Zebra
Aurora™ Vision

Aurora Vision Library 5.3

Function Reference

Created: 6/8/2023

Product version: 5.3.4.94078

Table of content:

41. Box
42. Geometry 3D Basics
43. Histogram Basics
44. Image Basics
45. Location
46. Point3DGrid Basics
47. Profile Basics
48. Deep Learning
49. Configuration
50. Shape Fitting 3D
51. Image Color Spaces
52. Image Local Transforms
53. Region Morphology
54. Geometry 3D Features
55. Geometry 3D Intersections
56. Image Thresholding
57. HandEyeCalibration Calibration
58. Geometry 2D Intersections
59. Geometry 2D Constructions
60. Geometry 3D Constructions
61. Geometry 2D Distance Metrics
62. Geometry 3D Distance Metrics
63. Path Classification
64. Region Relations
65. Clustering
66. Image Metrics
67. Path Global Transforms
68. Profile Local Transforms
69. Image Spatial Transforms
70. Shape Fitting
71. Region Basics
72. Data Classification Common
73. Image Tiling
74. Geometry 2D Relations
75. 1D Edge Detection
76. 1D Edge Detection 3D
77. Histogram Spatial Transforms
78. Profile Spatial Transforms
79. Image Pixel Statistics
80. Barcodes
81. Datacodes
82. 2D Edge Detection
83. Hough Transform
84. Shape Region Morphology
85. Image Features
86. Image Drawing
87. Image Segmentation
88. Region Global Transforms
89. Statistics
90. Surface Fitting
91. Format
92. Fourier Analysis
93. FTP
94. Histogram Features
95. Histogram Data Statistics
96. Histogram Metrics
97. HTTP
98. Nearest Neighbors
99. Texture Analysis
100. Geometry 3D Interpolations
101. Regression Analysis
102. Camera Calibration IO
103. Interoperability
104. Histogram IO

105. Path IO
106. Point3DGrid IO
107. Profile IO
108. Region IO
109. Surface IO
110. Image Vector Transforms
111. Multilayer Perceptron
112. Geometry 2D Normalizations
113. Path Features
114. Path Metrics
115. Point3DGrid Features
116. Polygon Features
117. Polygon Relations
118. Profile Metrics
119. Profile Features
120. Region Features
121. Region Point Transforms
122. Region Combinators
123. Region Metrics
124. Geometry 3D Normalizations
125. Surface Features
126. Surface Interpolations
127. Segmentation 3D
128. Histogram Local Transforms
129. Path Local Transforms
130. Support Vector Machines
131. Image Relations
132. Histogram Relations
133. Geometry 3D Relations
134. Profile Relations
135. Loop Generators
136. Matrix
137. String
138. Process
139. Time
140. Random
141. Unit Conversions
142. User Input
143. INI
144. Binary Data
145. Modbus TCP
146. Serial Port
147. TCP IP
148. Advantech
149. Advantech SUSI
150. AvSMART
151. AXIS
152. BitFlow
153. cxCam
154. Dahua
155. National Instruments
156. Ensenso
157. Flir
158. Gocator
159. Hikrobot
160. Hikvision
161. Hilscher
162. The Imaging Source
163. NET ICube
164. IDS
165. IFM
166. JAI
167. Kontron GPIO Board
168. LEX

169. Lumenera
170. Matrox
171. Microview
172. Euresys
173. MATRIX VISION
174. Neousys
175. Nodka
176. Omron
177. Kinect
178. OptoEngineeringHRAS
179. Photoneo
180. Basler
181. Roseek
182. SiliconSoftware
183. SmartRay
184. Smart
185. Flir Spinnaker
186. NET SynView
187. Thorlabs
188. Allied Vision Technologies
189. Imago Technologies
190. WebCamera
191. XIMEA
192. XSightSmartCamera
193. XSight
194. ZebraCameras
195. ZebraScanEngines
196. GigE Vision
197. GenApi
198. GenTL

1. AVL Common

Table of content:

CancelCurrentWork

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Set function cancellation flag. Function is thread-safe.

Syntax

```
void avl::CancelCurrentWork
(
)
```

CloseGPUProcessing

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Disables GPU processing.

Syntax

```
void avl::CloseGPUProcessing
(
)
```

DischargeAVLImageMemoryPools

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Clears images in the AVL memory pool.

Syntax

```
void avl::DischargeAVLImageMemoryPools
(
)
```

EnableAvlDiagnosticOutputs

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Enables computation of diagnostics outputs.

Syntax

```
void avl::EnableAvlDiagnosticOutputs
(
 bool enable
)
```

Parameters

Name	Type	Default	Description
enable	bool		

EnableAVX2Acceleration

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Enables AVX2 acceleration.

Syntax

```
void avl::EnableAVX2Acceleration
(
 bool enable
)
```

Parameters

	Name	Type	Default	Description
	enable	bool		

EnableGPUProcessing

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Enables GPU processing.

Syntax

```
void avl::EnableGPUProcessing
(
 bool inEnabled
)
```

Parameters

	Name	Type	Default	Description
➡	inEnabled	bool		

EnableNEONAcceleration

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Enables NEON acceleration. Only on ARM CPUs.

Syntax

```
void avl::EnableNEONAcceleration
(
 bool enable
)
```

Parameters

	Name	Type	Default	Description
	enable	bool		

EnableSSEAcceleration

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Enables SSE acceleration.

Syntax

```
void avl::EnableSSEAcceleration
(
 bool enable
)
```

Parameters

	Name	Type	Default	Description
	enable	bool		

GetAvlDiagnosticOutputsEnabled

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns AVL diagnostic state.

Syntax

```
bool avl::GetAvlDiagnosticOutputsEnabled
(
)
```

GetAVX2AccelerationEnabled

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns the state of the AVX2 acceleration.

Syntax

```
bool avl::GetAVX2AccelerationEnabled
(
)
```

GetGPUProcessingDiagInfo

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns GPU diagnostic information.

Syntax

```
bool avl::GetGPUProcessingDiagInfo
(
 atl::String& outDiagInfo
)
```

Parameters

	Name	Type	Default	Description
⬅	outDiagInfo	String&		

GetNEONAccelerationEnabled

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns the state of the NEON acceleration.

Syntax

```
bool avl::GetNEONAccelerationEnabled
(
)
```

GetParallelComputingThreadsCount

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns maximal number of threads to be used for computing.

Syntax

```
int avl::GetParallelComputingThreadsCount
(
)
```

GetSSEAccelerationEnabled

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns returns state of SSE acceleration.

Syntax

```
bool avl::GetSSEAccelerationEnabled
(
)
```

InitGPUProcessing

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Initializes GPU processing with a selected device.

Syntax

```
void avl::InitGPUProcessing
(
 atl::String& platformName = "",
 atl::String& deviceName = ""
)
```

Parameters

Name	Type	Default	Description
platformName	String&	""	
deviceName	String&	""	

IsCurrentWorkCancelled

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns true if function cancellation was requested.

Syntax

```
bool avl::IsCurrentWorkCancelled
(
)
```

ResetIsCurrentWorkCancelled

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Reset function cancellation flag.

Syntax

```
void avl::ResetIsCurrentWorkCancelled
(
)
```

SetParallelComputing

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Controls the state and number of available threads.

Syntax

```
void avl::SetParallelComputing
(
 bool enabled,
 int num_threads
)
```

Parameters

Name	Type	Default	Description
enabled	bool		
num_threads	int		Numbers of thread available for computation.

2. Debug Preview

Table of content:

Also in AVL Lite

DebugPreviewCloseAllWindows

Header: AVL.h
Namespace: avl
Module: FoundationLite

Closes all active debug preview windows and terminates the preview window subsystem.

Syntax

```
void avl::DebugPreviewCloseAllWindows
(
)
```

Description

This function will close all active preview windows, terminate background message pump thread and free debug preview subsystem resources.

After call to this function all preview window state objects become invalid.

See Also

- DebugPreviewWaitForWindowsClose – Waits until the user closes all active debug windows.

Also in AVL Lite

DebugPreviewCloseWindow

Header: AVL.h
Namespace: avl
Module: FoundationLite

Closes a debug preview window.

Syntax

```
void avl::DebugPreviewCloseWindow
(
 DebugPreviewWindowState& ioWindow
)
```

Parameters

	Name	Type	Default	Description
	ioWindow	DebugPreviewWindowState&		State object of window to be closed.

Description

This function will close the window bound to the specified state object (opened by DebugPreviewShowImage or DebugPreviewShowRegion) and free the resource occupied by the state object.

After call to this function, the state object will become invalid. Function invoked on an invalid state object ends with no results.

Also in AVL Lite

DebugPreviewShowImage

Header: AVL.h
Namespace: avl
Module: FoundationLite

Shows an image in a separated window. When called for the first time, it creates a new window with size fitted to the image.

Applications: This function is for development purpose only and requires Library Professional license.

Syntax

```
bool avl::DebugPreviewShowImage
(
 DebugPreviewWindowState& ioState,
 const Image& inImage,
 const wchar_t* inWindowName,
 int inPosX,
 int inPosY
)
```

Parameters

	Name	Type	Default	Description
	ioState	DebugPreviewWindowState&		Object used to maintain state of the function.
	inImage	const Image&		Image to preview.
	inWindowName	const wchar_t*	"Image Preview"	Name of newly opened window.
	inPosX	int	Windows default	X coordinate of initial windows position.
	inPosY	int	Windows default	Y coordinate of initial windows position.

Description

This function helps in debugging of Aurora Vision Library based applications by providing a quick but primitive way to preview internal application image-oriented data in floating windows.

A preview window is bound to the state object specified in the first argument of the function. On the first call on specified state object instance, this function will create and show a new floating window with size fitted to the specified image. Every subsequent call on the same state object instance will only update the image in preview.

A preview window will be closed as a result of the following actions:

- the user closes the window using its tool box (state object remains valid),
- the state object of the window is destructed,
- DebugPreviewCloseWindow is called on the window state object,
- DebugPreviewCloseAllWindows is called.

This function creates a background thread with message pump (single thread shared by all windows in the debug preview subsystem) that will stay active till application close (even when no more windows are active). To explicitly close the debug preview subsystem and free its resources call DebugPreviewCloseAllWindows function.

Leaving the start position set to its default value (-1x-1) will automatically arrange windows, starting from top left corner of the screen.

Return value

This function returns true after successful creation or update of an active window. After such window is closed by the user, subsequent calls to this function will return false.

See Also

- DebugPreviewShowRegion – Shows a region in a separated window. When called for the first time, creates a new window with size fitted to the region.
- DebugPreviewCloseWindow – Closes a debug preview window.

Also in AVL Lite

DebugPreviewShowNewImage

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a new separated window and shows an image in it.

Applications: This function is for development purpose only and requires Library Professional license.

Syntax

```
void avl::DebugPreviewShowNewImage
(
 const Image& inImage,
 const wchar_t* inWindowName,
 int inPosX,
 int inPosY
)
```

Parameters

	Name	Type	Default	Description
➡	inImage	const Image&		Image to preview.
➡	inWindowName	const wchar_t*	"Image Preview"	Name of newly opened window.
➡	inPosX	int	Windows default	X coordinate of initial windows position.
➡	inPosY	int	Windows default	Y coordinate of initial windows position.

Description

This function helps in debugging of Aurora Vision Library based applications by providing a quick but primitive way to preview internal application image-oriented data in floating windows.

Calling this function will create a new preview window with size fitted to the specified image. Such window is shown with the specified image but without returning any reference or handle. The new windows will remain active until it is closed by the user or by DebugPreviewCloseAllWindows function call. An application can call this function multiple times to open multiple windows as long as the limit of 32 simultaneously active windows is not exceeded.

This function creates a background thread with message pump (single thread shared by all windows in the debug preview subsystem) that will stay active till application close (even when no more windows are active). To explicitly close the debug preview subsystem and free its resources call DebugPreviewCloseAllWindows function.

Leaving the start position set to its default value (-1x-1) will automatically arrange windows, starting from top left corner of the screen.

See Also

- DebugPreviewShowNewRegion – Creates a new separated window and shows a region in it.
- DebugPreviewWaitForWindowsClose – Waits until the user closes all active debug windows.
- DebugPreviewCloseAllWindows – Closes all active debug preview windows and terminates the preview window subsystem.

Also in AVL Lite

DebugPreviewShowNewRegion

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a new separated window and shows a region in it.

Applications: This function is for development purpose only and requires Library Professional license.

Syntax

```
void avl::DebugPreviewShowNewRegion
(
 const Region& inRegion,
 const wchar_t* inWindowName,
 int inPosX,
 int inPosY
)
```

Parameters

	Name	Type	Default	Description
➡	inRegion	const Region&		Region to preview.
➡	inWindowName	const wchar_t*	"Region Preview"	Name of newly opened window.
➡	inPosX	int	Windows default	X coordinate of initial windows position.
➡	inPosY	int	Windows default	Y coordinate of initial windows position.

Description

This function helps in debugging of Aurora Vision Library based applications by providing a quick but primitive way to preview internal application image-oriented data in floating windows.

Calling this function will create a new preview window with size fitted to the specified region. Such window is shown with the specified region but without returning any reference or handle. The new windows will remain active until it is closed by the user or by DebugPreviewCloseAllWindows function call. An application can call this function multiple times to open multiple windows as long as the limit of 32 simultaneously active windows is not exceeded.

This function creates a background thread with message pump (single thread shared by all windows in the debug preview subsystem) that will stay active till application close (even when no more windows are active). To explicitly close the debug preview subsystem and free its resources call DebugPreviewCloseAllWindows function.

Leaving the start position set to its default value (-1x-1) will automatically arrange windows, starting from top left corner of the screen.

See Also

- DebugPreviewShowNewImage – Creates a new separated window and shows an image in it.
- DebugPreviewWaitForWindowsClose – Waits until the user closes all active debug windows.
- DebugPreviewCloseAllWindows – Closes all active debug preview windows and terminates the preview window subsystem.

Also in AVL Lite

DebugPreviewShowRegion

Header: AVL.h
Namespace: avl
Module: FoundationLite

Shows a region in a separated window. When called for the first time, creates a new window with size fitted to the region.

Applications: This function is for development purpose only and requires Library Professional license.

Syntax

```
bool avl::DebugPreviewShowRegion
(
 DebugPreviewWindowState& ioState,
 const Region& inRegion,
 const wchar_t* inWindowName,
 int inPosX,
 int inPosY
)
```

Parameters

	Name	Type	Default	Description
	ioState	DebugPreviewWindowState&		Object used to maintain state of the function.
	inRegion	const Region&		Region to preview.
	inWindowName	const wchar_t*	"Region Preview"	Name of newly opened window.
	inPosX	int	Windows default	X coordinate of initial windows position.
	inPosY	int	Windows default	Y coordinate of initial windows position.

Description

This function helps in debugging of Aurora Vision Library based applications by providing a quick but primitive way to preview internal application image-oriented data in floating windows.

A preview window is bound to the state object specified in the first argument of the function. On the first call on specified state object instance, this function will create and show a new floating window with size fitted to the specified region. Every subsequent call on the same state object instance will only update the region in preview.

A preview window will be closed as a result of the following actions:

- the user closes the window using its tool box (state object remains valid),
- the state object of the window is destructed,
- DebugPreviewCloseWindow is called on the window state object,
- DebugPreviewCloseAllWindows is called.

This function creates a background thread with message pump (single thread shared by all windows in the debug preview subsystem) that will stay active till application close (even when no more windows are active). To explicitly close the debug preview subsystem and free its resources call DebugPreviewCloseAllWindows function.

Leaving the start position set to its default value (-1x-1) will automatically arrange windows, starting from top left corner of the screen.

Return value

This function returns true after successful creation or update of an active window. After such window is closed by the user, subsequent calls to this function will return false.

See Also

- DebugPreviewShowImage – Shows an image in a separated window. When called for the first time, it creates a new window with size fitted to the image.
- DebugPreviewCloseWindow – Closes a debug preview window.

Also in AVL Lite

DebugPreviewWaitForAnyWindowClose

Header: AVL.h
Namespace: avl
Module: FoundationLite

Waits until the user closes any active debug windows.

Syntax

```
void avl::DebugPreviewWaitForAnyWindowClose
(
)
```

DebugPreviewWaitForWindowsClose

Header: AVL.h
Namespace: avl
Module: FoundationLite

Waits until the user closes all active debug windows.

Syntax

```
void avl::DebugPreviewWaitForWindowsClose
(
)
```

Description

This function will block until the user closes all active preview windows, or until the debug preview subsystem is terminated by an other thread. During the waiting period all preview windows remain responsive.

This function will not invalidate any window state objects.

See Also

- DebugPreviewCloseAllWindows – Closes all active debug preview windows and terminates the preview window subsystem.

3. File System

Table of content:

EnumerateObjects

Header: STD.h
Namespace: avl
Module: FoundationLite

Enumerates the avdata files present in a disk directory and sorts them according to the specified order and loads an object from a file.

Syntax

```
bool avl::EnumerateObjects
(
 EnumerateFilesState& ioState,
 const atl::Directory& inDirectory,
 avl::FileSortingOrder::Type inSortingOrder,
 bool inRepeat,
 bool inProcessSubdirectories,
 bool inInvert,
 avl::StreamMode::Type inStreamMode,
 atl::File& outFilePath,
 atl::String& outFileName,
 Type& outObject,
 atl::Optional<bool> outIsFirst = atl::NIL,
 atl::Optional<bool> outIsLast = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
↔	ioState	EnumerateFilesState&		Object used to maintain state of the function.
→	inDirectory	const Directory&		Input directory
→	inSortingOrder	FileSortingOrder::Type		Sorting order
→	inRepeat	bool		Determines whether to repeat reading directory after reading all files
→	inProcessSubdirectories	bool		Flag indicating whether to enumerate files from the subdirectories or not
→	inInvert	bool		Flag indicating whether to enumerate files backwards or not
→	inStreamMode	StreamMode::Type		Binary or text format of the files.
←	outFilePath	File&		Output file path
←	outFileName	String&		Output file name
←	outObject	Type&		
←	outIsFirst	Optional<bool>	NIL	Flag indicating the first iteration
←	outIsLast	Optional<bool>	NIL	Flag indicating the last iteration

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outIsFirst**, **outIsLast**.

Read more about Optional Outputs.

Remarks

Linux

Sorting output by file creation date when running AVL on Linux is not possible due to file system restrictions. When requested to do so EnumerateFiles will throw an exception.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported stream mode in EnumerateObjects function.

See Also

- LoadObject – Loads an AVL object from a file.

EnumerateFiles

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Enumerates the files present in a disk directory.

Syntax

```
bool avl::EnumerateFiles
(
 EnumerateFilesState& ioState,
 const atl::Directory& inDirectory,
 const atl::String& inExtensions,
 avl::FileSortingOrder::Type inSortingOrder,
 bool inRepeat,
 bool inProcessSubdirectories,
 bool inInvert,
 atl::File& outFilePath,
 atl::String& outFileName,
 atl::Optional<bool> outIsFirst = atl::NIL,
 atl::Optional<bool> outIsLast = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
↔	ioState	EnumerateFilesState&		Object used to maintain state of the function.
→	inDirectory	const Directory&		Input directory
→	inExtensions	const String&		Allowed extensions separated by semicolon
→	inSortingOrder	FileSortingOrder::Type		Sorting order
→	inRepeat	bool		Determines whether to repeat reading directory after reading all files
→	inProcessSubdirectories	bool		Flag indicating whether to enumerate files from the subdirectories or not
→	inInvert	bool		Flag indicating whether to enumerate files backwards or not
←	outFilePath	File&		Output file path
←	outFileName	String&		Output file name
←	outIsFirst	Optional<bool>	NIL	Flag indicating the first iteration
←	outIsLast	Optional<bool>	NIL	Flag indicating the last iteration

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outIsFirst**, **outIsLast**.

Read more about Optional Outputs.

Remarks

Linux

Sorting output by file creation date when running AVL on Linux is not possible due to file system restrictions. When requested to do so EnumerateFiles will throw an exception.

Errors

List of possible exceptions:

Error type	Description
DomainError	Cannot sort by file creation date on Linux.
DomainError	Directory doesn't exist: Directory path
DomainError	Empty string is not a valid directory path.

EnumerateFiles_Random

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Enumerates the files present in a disk directory sorted randomly.

Syntax

```
bool avl::EnumerateFiles_Random
(
 EnumerateFilesState& ioState,
 const atl::Directory& inDirectory,
 const atl::String& inExtensions,
 atl::Optional<int> inSeed,
 bool inRepeat,
 bool inProcessSubdirectories,
 bool inInvert,
 atl::File& outFilePath,
 atl::String& outFileName,
 atl::Optional<bool> outIsFirst = atl::NIL,
 atl::Optional<bool> outIsLast = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
↔	ioState	EnumerateFilesState&		Object used to maintain state of the function.
→	inDirectory	const Directory&		Input directory
→	inExtensions	const String&		Allowed extensions separated by semicolon
→	inSeed	Optional<int>		Random seed used to determine random sorting order
→	inRepeat	bool		Determines whether to repeat reading directory after reading all files
→	inProcessSubdirectories	bool		Flag indicating whether to enumerate files from the subdirectories or not
→	inInvert	bool		Flag indicating whether to enumerate files backwards or not
←	outFilePath	File&		Output file path
←	outFileName	String&		Output file name
←	outIsFirst	Optional<bool>	NIL	Flag indicating the first iteration
←	outIsLast	Optional<bool>	NIL	Flag indicating the last iteration

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outIsFirst**, **outIsLast**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Cannot sort by file creation date on Linux.
DomainError	Directory doesn't exist: Directory path
DomainError	Empty string is not a valid directory path.

CheckDiskSpace

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Determines the information about the filesystem on which the pathname inPath is located.

Syntax

```
void avl::CheckDiskSpace
(
 const atl::Directory& inPath,
 atl::int64& outCapacity,
 atl::int64& outFree,
 atl::int64& outAvailable
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Directory&	\".\"	Input path
←	outCapacity	int64&		Total size of the filesystem, in bytes
←	outFree	int64&		Free space on the filesystem, in bytes
←	outAvailable	int64&		Free space available to a non-privileged process (may be equal or less than outFree)

ConvertPathToAbsolute

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Makes the path absolute, inStartPath + inPath

Syntax

```
void avl::ConvertPathToAbsolute
(
 const atl::String& inPath,
 const atl::Optional<atl::Directory> inStartPath,
 atl::String& outPath
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const String&		relative path
→	inStartPath	const Optional<Directory>	NIL	starting path, by default the path of the project or where the program was executed in case of AVL
←	outPath	String&		the resulting absolute path

ConvertPathToRelative

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Makes the path relative to inBasePath

Syntax

```
void avl::ConvertPathToRelative
(
 const atl::String& inPath,
 const atl::Optional<atl::Directory> inBasePath,
 atl::String& outPath
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const String&		absolute path
→	inBasePath	const Optional<Directory>	NIL	base path, by default the path of the project or where the program was executed in case of AVL
←	outPath	String&		the resulting relative path

CopyFiles

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Copies files that match a pattern from a directory to other directory.

Syntax

```
void avl::CopyFiles
(
 const atl::Directory& inStartDirectory,
 const atl::String& inMask,
 const atl::Directory& inOutputDirectory,
 bool inProcessSubdirectories,
 bool inOverwrite,
 bool& outSuccess
)
```

Parameters

	Name	Type	Default	Description
→	inStartDirectory	const Directory&		
→	inMask	const String&		Wildcard pattern
→	inOutputDirectory	const Directory&		
→	inProcessSubdirectories	bool		
→	inOverwrite	bool		
←	outSuccess	bool&		

Errors

List of possible exceptions:

Error type	Description
DomainError	inMask needs to be filled in CopyFiles.
DomainError	inOutputDirectory needs to be filled in CopyFiles.
DomainError	inStartDirectory needs to be filled in CopyFiles.

CreateDirectories

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Creates a directory tree if it does not exist.

Syntax

```
void avl::CreateDirectories
(
 const atl::Directory& inDirectory,
 bool& outSuccess
)
```

Parameters

	Name	Type	Default	Description
→	inDirectory	const Directory&		
←	outSuccess	bool&		

FileAttributes

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Provides information about file, e.g. size, modification time

Syntax

```
void avl::FileAttributes
(
 const atl::File& inFile,
 int& outSizeInKBytes,
 atl::int64& outSizeInBytes,
 bool& outIsReadOnly,
 atl::String& outAccessTime,
 atl::String& outModificationTime,
 atl::String& outCreationTime
)
```

Parameters

	Name	Type	Default	Description
→	inFile	const File&		Input file
←	outSizeInKBytes	int&		File size in kilobytes
←	outSizeInBytes	int64&		File size in bytes
←	outIsReadOnly	bool&		Flag indicating whether the file is read-only
←	outAccessTime	String&		File access time
←	outModificationTime	String&		File modification time
←	outCreationTime	String&		File creation time

Remarks

Linux

File creation time is unavailable when running AVL on Linux due to file system restrictions. It will always report value 0 which translates to the following time: 1970-01-01T01:00:00.

FileChecksum

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Returns CRC checksum of the input file.

Syntax

```
void avl::FileChecksum
(
 const atl::File& inFile,
 atl::String& outChecksum
)
```

Parameters

	Name	Type	Default	Description
→	inFile	const File&		
←	outChecksum	String&		

FindDirectories

Header: STD.h
Namespace: avl
Module: FoundationLite

Returns subdirectories of the input directory.

Syntax

```
void avl::FindDirectories
(
 const atl::Directory& inStartDirectory,
 bool inRecursive,
 atl::Optional<avl::FileSortingOrder::Type> inSortingOrder,
 atl::Array<atl::Directory>& outDirectories,
 bool& outFound = atl::Dummy<bool>()
)
```

Parameters

	Name	Type	Default	Description
→	inStartDirectory	const Directory&	\".\"	
→	inRecursive	bool		
→	inSortingOrder	Optional<FileSortingOrder::Type>	NIL	Sorting order
←	outDirectories	Array<Directory>&		
←	outFound	bool&	Dummy<bool>()	

Errors

List of possible exceptions:

Error type	Description
DomainError	Given start directory is invalid in FindDirectories.

FindFiles

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Returns files of the input directory.

Syntax

```
void avl::FindFiles
(
 const atl::Directory& inStartDirectory,
 const atl::String& inMask,
 bool inSubdirs,
 atl::Optional<avl::FileSortingOrder::Type> inSortingOrder,
 atl::Array<atl::File>& outFilePaths,
 atl::Array<atl::String>& outFileNames,
 bool& outFound = atl::Dummy<bool>()
)
```

Parameters

	Name	Type	Default	Description
→	inStartDirectory	const Directory&	\".\"	Input directory
→	inMask	const String&	\"*\"	Wildcard pattern
→	inSubdirs	bool		Read subdirectories
→	inSortingOrder	Optional<FileSortingOrder::Type>	NIL	Sorting order
←	outFilePaths	Array<File>&		File paths
←	outFileNames	Array<String>&		File names
←	outFound	bool&	Dummy<bool>()	

Remarks

Working with Find Files

Start with defining a directory path in **inStartDirectory** port to choose where you want to look for files. To search files in subdirectories set **inSubdirs** to 'true'.

inMask pattern string

Port **inMask** specifies wildcard pattern that selects files this filter will operate on. Supported wildcards:

- * - any string of characters, including no characters
- ? - exactly one character

You can use any expression like examples below:

Examples:

- * - all files,
- *.jpg - files only with extension .jpg,
- Filename.* - files with name "Filename" and any extension, including no extension e.g. "Filename."
- Filename.?? - files with name "Filename" and two letter extension,

Errors

List of possible exceptions:

Error type	Description
DomainError	Given start directory is invalid in FindFiles.
IoError	Error opening start directory in FindFiles.

See Also

- TestFileExists – Checks if a given file is present.

JoinPathParts

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Joins path parts

Syntax

```
void avl::JoinPathParts
(
 const atl::String& inPath1,
 const atl::String& inPath2,
 const atl::Optional<atl::String>& inPath3,
 const atl::Optional<atl::String>& inPath4,
 atl::String& outPath
)
```

Parameters

	Name	Type	Default	Description
→	inPath1	const String&		First input path
→	inPath2	const String&		Second input path
→	inPath3	const Optional<String>&	NIL	
→	inPath4	const Optional<String>&	NIL	
←	outPath	String&		Output path

RemoveDirectory

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Removes a directory, with all files and subdirectories.

Syntax

```
void avl::RemoveDirectory
(
 const atl::Directory& inDirectory,
 bool& outSuccess
)
```

Parameters

	Name	Type	Default	Description
→	inDirectory	const Directory&		
←	outSuccess	bool&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Path is not a directory in RemoveDirectory

RemoveFile_Multiple

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Removes files that match a pattern from a directory.

Syntax

```
void avl::RemoveFile_Multiple
(
 const atl::Directory& inStartDirectory,
 const atl::String& inMask,
 bool inProcessSubdirectories,
 bool& outSuccess
)
```

Parameters

	Name	Type	Default	Description
→	inStartDirectory	const Directory&		
→	inMask	const String&		Wildcard pattern
→	inProcessSubdirectories	bool		
←	outSuccess	bool&		

Remarks

inMask pattern string

Port **inMask** specifies wildcard pattern that selects files this filter will operate on. Supported wildcards:

- * - any string of characters, including no characters
- ? - exactly one character

You can use any expression like examples below:

Examples:

- * - all files,
- *.jpg - files only with extension .jpg,
- Filename.* - files with name "Filename" and any extension, including no extension e.g. "Filename."
- Filename.?? - files with name "Filename" and two letter extension,

Errors

List of possible exceptions:

Error type	Description
DomainError	inMask needs to be filled in
DomainError	inStartDirectory needs to be filled in

See Also

- FindFiles – Returns files of the input directory.

RemoveFile_Single

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Removes single file.

Syntax

```
void avl::RemoveFile_Single
(
 const atl::File& inFile,
 bool& outSuccess
)
```

Parameters

	Name	Type	Default	Description
→	inFile	const File&		
←	outSuccess	bool&		

Remarks

inMask pattern string

Port **inMask** specifies wildcard pattern that selects files this filter will operate on. Supported wildcards:

- * - any string of characters, including no characters
- ? - exactly one character

You can use any expression like examples below:

Examples:

- * - all files,
- *.jpg - files only with extension .jpg,
- Filename.* - files with name "Filename" and any extension, including no extension e.g. "Filename."
- Filename.?? - files with name "Filename" and two letter extension,

Errors

List of possible exceptions:

Error type	Description
DomainError	inFile does not exist in RemoveFile.
DomainError	inFile is not a normal file in RemoveFile.

See Also

- FindFiles – Returns files of the input directory.

SplitFileNameToParts

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Splits the filename to base and extension, the extension is considered

Syntax

```
void avl::SplitFileNameToParts
(
 const atl::String& inPath,
 atl::String& outNameBase,
 atl::String& outExtension
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const String&		Input path
←	outNameBase	String&		
←	outExtension	String&		

SplitPathToNameParent

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Splits a file path into a file name and a parent directory

Syntax

```
void avl::SplitPathToNameParent
(
 const atl::String& inPath,
 atl::String& outRoot,
 atl::String& outName
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const String&		input path
←	outRoot	String&		the parent path
←	outName	String&		name, directory or file

SplitPathToParts

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Determines the information about the filesystem on which the pathname inPath is located.

Syntax

```
void avl::SplitPathToParts
(
 const atl::String& inPath,
 atl::Array<atl::String>& outPathParts
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const String&		Input path
←	outPathParts	Array<String>&		

TestDirectoryEmpty

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Checks if a given directory is empty.

Syntax

```
void avl::TestDirectoryEmpty
(
 const atl::Directory& inDirectory,
 bool& outEmpty
)
```

Parameters

	Name	Type	Default	Description
→	inDirectory	const Directory&		
←	outEmpty	bool&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Directory does not exist in TestDirectoryEmpty
DomainError	Path is not a directory in TestDirectoryEmpty

TestDirectoryExists

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Checks if a given directory is present.

Syntax

```
void avl::TestDirectoryExists
(
 const atl::Directory& inDirectory,
 bool& outExists
)
```

Parameters

	Name	Type	Default	Description
→	inDirectory	const Directory&		
←	outExists	bool&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Path is not a directory in TestDirectoryExists

TestFileEmpty

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Checks if the size of the file is equal to zero.

Syntax

```
void avl::TestFileEmpty
(
 const atl::File& inFile,
 bool& outEmpty
)
```

Parameters

	Name	Type	Default	Description
→	inFile	const File&		
←	outEmpty	bool&		

Errors

List of possible exceptions:

Error type	Description
DomainError	File does not exist in TestFileEmpty
DomainError	Path does not point to a file in TestFileEmpty

TestFileExists

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Checks if a given file is present.

Syntax

```
void avl::TestFileExists
(
 const atl::File& inFile,
 bool& outSuccess
)
```

Parameters

	Name	Type	Default	Description
→	inFile	const File&		
←	outSuccess	bool&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Path is not a file in TestFileExists

4. Image IO

Table of content:

Also in AVL Lite

FillImageBuffer

Header: AVL.h
Namespace: avl
Module: FoundationLite

Fills image buffer with provided data.

Syntax

```
avl::FillImageBuffer
(
avl::Image& inImage,
int inWidth,
int inHeight,
avl::PlainType::Type inType,
int inDepth,
int inPitch,
atl::byte * inData
)
```

Parameters

	Name	Type	Default	Description
➡	inImage	Image&		
➡	inWidth	int		
➡	inHeight	int		
➡	inType	PlainType::Type		
➡	inDepth	int		
➡	inPitch	int		
➡	inData	byte *		

Also in AVL Lite

FillImagePreviewBuffer

Header: AVL.h
Namespace: avl
Module: FoundationLite

Fills image buffer with provided data.

Syntax

```
avl::FillImagePreviewBuffer
(
avl::Image& inImage,
int inPreviewWidth,
int inPreviewHeight,
atl::byte* inBuffer,
int inBufferStride,
int inPitch,
atl::byte * inData
)
```

Parameters

	Name	Type	Default	Description
➡	inImage	Image&		
➡	inPreviewWidth	int		
➡	inPreviewHeight	int		
➡	inBuffer	byte*		
➡	inBufferStride	int		
➡	inPitch	int		
➡	inData	byte *		

EnumerateImages

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Scans a disk directory for image files and then returns the images one by one in consecutive iterations sorted according to the specified order.

Applications: Emulates image acquisition with images stored on disk.

Syntax

```
bool avl::EnumerateImages
(
 EnumerateFilesState& ioState,
 const atl::Directory& inDirectory,
 atl::Optional<avl::ImageFileFormat::Type> inFileType,
 avl::FileSortingOrder::Type inSortingOrder,
 bool inRepeat,
 bool inProcessSubdirectories,
 bool inInvert,
 bool inLoadAlphaChannel,
 const int inDelay,
 avl::Image& outImage,
 atl::File& outFilePath,
 atl::String& outFileName,
 atl::Optional<bool&> outIsFirst = atl::NIL,
 atl::Optional<bool&> outIsLast = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
	ioState	EnumerateFilesState&		Object used to maintain state of the function.
	inDirectory	const Directory&	\".\"	Input directory
	inFileType	Optional<ImageFileFormat::Type>	NIL	File format of the images
	inSortingOrder	FileSortingOrder::Type		Sorting order
	inRepeat	bool		Determines whether to repeat reading directory after reading all files
	inProcessSubdirectories	bool		Flag indicating whether to load images from the subdirectories or not
	inInvert	bool		Flag indicating whether to enumerate images backwards or not
	inLoadAlphaChannel	bool		Flag indicating whether to load alpha channel of the image or not
	inDelay	const int		Minimum time between iterations in milliseconds
	outImage	Image&		Output image
	outFilePath	File&		Output file path
	outFileName	String&		Output file name
	outIsFirst	Optional<bool&>	NIL	Flag indicating the first iteration
	outIsLast	Optional<bool&>	NIL	Flag indicating the last iteration

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outIsFirst**, **outIsLast**.

Read more about Optional Outputs.

Hints

- Set **inDirectory** to specify where on your disk are the images you want to load.
- This filter can also be added easily by dragging and dropping a disk directory from Windows Explorer to the Program Editor in Aurora Vision Studio.

EnumerateImages_Random

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Scans a disk directory for image files and then returns the images one by one in consecutive iterations sorted randomly.

Applications: Emulates image acquisition with images stored on disk.

Syntax

```
bool avl::EnumerateImages_Random
(
 EnumerateFilesRandomState& ioState,
 const atl::Directory& inDirectory,
 atl::Optional<avl::ImageFileFormat::Type> inFileType,
 atl::Optional<int> inSeed,
 bool inRepeat,
 bool inProcessSubdirectories,
 bool inInvert,
 bool inLoadAlphaChannel,
 const int inDelay,
 avl::Image& outImage,
 atl::File& outFilePath,
 atl::String& outFileName,
 atl::Optional<bool&> outIsFirst = atl::NIL,
 atl::Optional<bool&> outIsLast = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
↔	ioState	EnumerateFilesRandomState&		Object used to maintain state of the function.
→	inDirectory	const Directory&	\".\"	Input directory
→	inFileType	Optional<ImageFileFormat::Type>	NIL	File format of the images
→	inSeed	Optional<int>	NIL	Random seed used to determine random sorting order
→	inRepeat	bool		Determines whether to repeat reading directory after reading all files
→	inProcessSubdirectories	bool		Flag indicating whether to load images from the subdirectories or not
→	inInvert	bool		Flag indicating whether to enumerate images backwards or not
→	inLoadAlphaChannel	bool		Flag indicating whether to load alpha channel of the image or not
→	inDelay	const int		Minimum time between iterations in milliseconds
←	outImage	Image&		Output image
←	outFilePath	File&		Output file path
←	outFileName	String&		Output file name
←	outIsFirst	Optional<bool&>	NIL	Flag indicating the first iteration
←	outIsLast	Optional<bool&>	NIL	Flag indicating the last iteration

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outIsFirst**, **outIsLast**.

Read more about Optional Outputs.

GetAvailableVideoCompressors

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Lists FourCC names of available video compressors.

Syntax

```
void avl::GetAvailableVideoCompressors
(
 atl::Array<atl::String>& outCompressors
)
```

Parameters

	Name	Type	Default	Description
←	outCompressors	Array<String>&		Available compressors.

GrabImage_FromFiles

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Simulates capturing a frame from a camera.

Applications: Can be used as EnumerateImages, but its state is global in a program - does not reset when some task is finished.

Syntax

```
bool avl::GrabImage_FromFiles
(
 const atl::Directory& inDirectory,
 atl::Optional<avl::ImageFileFormat::Type> inFileType,
 avl::FileSortingOrder::Type inSortingOrder,
 bool inRepeat,
 bool inProcessSubdirectories,
 bool inInvert,
 bool inLoadAlphaChannel,
 int inDelay,
 avl::Image& outImage,
 atl::File& outFilePath,
 atl::String& outFileName
)
```

Parameters

	Name	Type	Range	Default	Description
→	inDirectory	const Directory&		\".\"	Input directory
→	inFileType	Optional<ImageFileFormat::Type>		NIL	File format of the images
→	inSortingOrder	FileSortingOrder::Type			Sorting order
→	inRepeat	bool			Determines whether to repeat reading directory after reading all files
→	inProcessSubdirectories	bool			Flag indicating whether to load images from the subdirectories or not
→	inInvert	bool			Flag indicating whether to enumerate images backwards or not
→	inLoadAlphaChannel	bool			Flag indicating whether to load alpha channel of the image or not
→	inDelay	int	0 - 2000		
←	outImage	Image&			Output image
←	outFilePath	File&			Output file path
←	outFileName	String&			Output file name

Description

This filter works similarly to EnumerateImages, but it behaves more like a camera: it preserves its state across the whole program.

For example, you can have two GrabImage_FromFiles filters in two different Task macrofilters. When execution of second Task starts, images loaded previously in the first Task are not loaded again.

Errors

List of possible exceptions:

Error type	Description
DomainError	All instances of GrabImage_FromFiles must have these same parameters.

See Also

- EnumerateImages – Scans a disk directory for image files and then returns the images one by one in consecutive iterations sorted according to the specified order.

GrabImage_FromFiles_ResetState

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Resets the global state of GrabImage_FromFiles filter.

Syntax

```
void avl::GrabImage_FromFiles_ResetState
(
)
```

GrabScreenshot

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Grabs a screenshot of the Desktop.

Syntax

```
void avl::GrabScreenshot
(
 const atl::Optional<avl::Box>& inSelection,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inSelection	const Optional<Box>&	NIL	Bounding box of screen area to capture, whole desktop if not specified.
←	outImage	Image&		Desktop screenshot image

LoadImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Loads a single image from a file.

Syntax

```
void avl::LoadImage
(
 const atl::File& inFile,
 bool inLoadAlphaChannel,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inFile	const File&		Path to a file
→	inLoadAlphaChannel	bool		Whether to load the alpha channel (if exists) as an additional image channel
←	outImage	Image&		Output image

Description

The operation loads an image from a file in one of the standard image file formats. Currently the filter supports the following formats:

- BMP (*.bmp),
- JPEG (*.jpg, *jpeg),
- PNG (*.png),
- PNM (*.pbm, *.pgm, *.ppm, *.pnm),
- TIFF (*.tif, *.tiff).

The format of the image file is recognized automatically based on the file header.

The resulting **outImage** will be three-channel image of UInt8 pixel type, or sometimes of UInt16 pixel type for 16-bit depth images of supported formats (PNG, TIFF).

Hints

- Set **inFile** to specify a path to the image file you want to load.
- This filter can also be added easily by dragging and dropping an image file from Windows Explorer to the Program Editor in Aurora Vision Studio.

Errors

List of possible exceptions:

Error type	Description
	Image is damaged or it is not a BMP, PNG, PNM, TIF or JPG file in LoadImage. File: File name
DomainError	Unable to open an image. Image header format is unrecognized. Image format is unknown or image is damaged. Load image supports only BMP, PNG, PNM, JPG and TIFF formats.

See Also

- SaveImage – Saves an image to a file.

LoadImageFormat

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Loads a single image format from a file without loading all image data.

Syntax

```
void avl::LoadImageFormat
(
 const atl::File& inFile,
 bool inLoadAlphaChannel,
 avl::ImageFormat& outImageFormat
)
```

Parameters

	Name	Type	Default	Description
→	inFile	const File&		Path to a file
→	inLoadAlphaChannel	bool		
←	outImageFormat	ImageFormat&		

Errors

List of possible exceptions:

Error type	Description
	Image is damaged or it is not a BMP, PNG, PNM, TIF or JPG file in LoadImageFormat. File: File name
DomainError	Unable to open an image. Image header format is unrecognized. Image format is unknown or image is damaged. Load image supports only BMP, PNG, PNM, JPG and TIFF formats.

LoadImageFromBuffer

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Loads a single image from a file stored in memory.

Applications: Use this filter when you received an image file through I/O communication, e.g. through a Tcplp connection.

Syntax

```
void avl::LoadImageFromBuffer
(
 const avl::ByteBuffer& inBuffer,
 int inOffset,
 atl::Optional<int> inLength,
 bool inLoadAlphaChannel,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inBuffer	const ByteBuffer&		Source buffer containing image file data
→	inOffset	int	0	Image data start position in source buffer
→	inLength	Optional<int>	NIL	Image data length in source buffer
→	inLoadAlphaChannel	bool		Whether to load the alpha channel (if exists) as an additional image channel
←	outImage	Image&		Output image

Description

This function loads an Image from a common format file similarly to LoadImage, but instead of accessing the file system it uses only the memory by loading the file content from ByteBuffer.

The operation loads an image from a file in one of the standard image file formats. Currently the filter supports the following formats:

- BMP (*.bmp),
- JPEG (*.jpg, *jpeg),
- PNG (*.png),
- PNM (*.pbm, *.pgm, *.ppm, *.pnm),
- TIFF (*.tif, *.tiff).

The format of the image file is recognized automatically based on the file header.

The resulting **outImage** will be three-channel image of UInt8 pixel type, or sometimes of UInt16 pixel type for 16-bit depth images of supported formats (PNG, TIFF).

Errors

List of possible exceptions:

Error type	Description
IoError	Reading beyond the end of the byte buffer in LoadImageFromBuffer.

See Also

- SaveImageToBuffer – Saves an image to a file stored in ByteBuffer.
- LoadImage – Loads a single image from a file.

LoadMultiplePagesTiffImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Loads a image array from a tiff file.

Syntax

```
void avl::LoadMultiplePagesTiffImage
(
 const atl::File& inFile,
 bool inLoadAlphaChannel,
 atl::Array<avl::Image>& outImages
)
```

Parameters

	Name	Type	Default	Description
→	inFile	const File&		Path to a file
→	inLoadAlphaChannel	bool		Whether to load the alpha channel (if exists) as an additional image channel
←	outImages	Array<Image>&		

Errors

List of possible exceptions:

Error type	Description
	Image is damaged or it is not a TIF file in LoadMultiplePagesTiffImage. File: File name
DomainError	Unable to open an image. Image header format is unrecognized. Image format is unknown or image is damaged. Load image supports only TIFF formats.

OpenInputVideoStream

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Opens and creates a video stream from a file.

Syntax

```
void avl::OpenInputVideoStream
(
 const atl::File& inFile,
 int inStartFrame,
 avl::InputVideoStream& outInputVideoStream,
 float& outFPS,
 int& outMaxFrame
)
```

Parameters

	Name	Type	Default	Description
→	inFile	const File&		Video file
→	inStartFrame	int		Number of first frame to fetch.
←	outInputVideoStream	InputVideoStream&		Initialized input video stream
←	outFPS	float&		Frames per second
←	outMaxFrame	int&		Last frame number

Errors

List of possible exceptions:

Error type	Description
DomainError	Function not available under the linux.
DomainError	Negative start frame in OpenInputVideoStream.
DomainError	Start frame too high for current video in OpenInputVideoStream.
RuntimeError	Unexpected error in OpenInputVideoStream.

Also in AVL Lite

OpenOutputVideoStream

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a video stream which will be saved to a selected file.

Syntax

```
void avl::OpenOutputVideoStream
(
 const atl::File& inFile,
 const avl::VideoEncoder::Type inVideoEncoder,
 const float inFPS,
 const int inFrameWidth,
 const int inFrameHeight,
 avl::OutputVideoStream& outOutputVideoStream
)
```

Parameters

	Name	Type	Default	Description
→	inFile	const File&		File path of the output file
→	inVideoEncoder	const VideoEncoder::Type		Video encoder
→	inFPS	const float		Desired frame rate of the produced video file
→	inFrameWidth	const int		Width of frame being added
→	inFrameHeight	const int		Height of frame being added
←	outOutputVideoStream	OutputVideoStream&		Initialized output video stream

Errors

List of possible exceptions:

Error type	Description
DomainError	Unsupported Video Encoder in OpenOutputVideoStream.

Also in AVL Lite

OpenOutputVideoStream_Any

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a video stream which will be saved to a selected file.

Syntax

```
void avl::OpenOutputVideoStream_Any
(
 const atl::File& inFile,
 const avl::VideoEncoderName& inVideoEncoderName,
 const float inFPS,
 const int inFrameWidth,
 const int inFrameHeight,
 avl::OutputVideoStream& outOutputVideoStream
)
```

Parameters

	Name	Type	Default	Description
→	inFile	const File&		File path of the output file
→	inVideoEncoderName	const VideoEncoderName&		Video encoder
→	inFPS	const float		Desired frame rate of the produced video file
→	inFrameWidth	const int		Width of frame being added
→	inFrameHeight	const int		Height of frame being added
←	outOutputVideoStream	OutputVideoStream&		Initialized output video stream

ReadVideoStream

Header: AVL.h
Namespace: avl
Module: FoundationLite

Acquires a frame from a previously opened video stream.

Syntax

```
bool avl::ReadVideoStream
(
 const avl::InputVideoStream& inInputVideoStream,
 bool inRepeat,
 int& outFrameNum,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inInputVideoStream	const InputVideoStream&		Previously opened input video stream
→	inRepeat	bool		Determines whether to repeat video playback
←	outFrameNum	int&		Current frame number
←	outImage	Image&		Current frame

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrectly initialized InputVideoStream object passed to ReadVideoStream

SaveImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Saves an image to a file.

Syntax

```
void avl::SaveImage
(
 const avl::Image& inImage,
 atl::Optional<avl::ImageFileFormat::Type> inImageFileFormat,
 const atl::File& inFile,
 bool inIgnoreErrors = false
)
```

Parameters

	Name	Type	Default	Description
➡	inImage	const Image&		An image to be saved
➡	inImageFileFormat	Optional<ImageFileFormat::Type>	NIL	If Nil the format will be chosen on the basis of extension
➡	inFile	const File&		Path to a file
➡	inIgnoreErrors	bool	false	Switches to re-trying and canceling in case of errors (e.g. when the disk is full)

Description

The operation saves an image to file encoded in one of the standard image file formats. Currently the filter supports the following formats:

- BMP (*.bmp)
- JPEG (*.jpg, *jpeg)
- PNG (*.png),
- PNM (*.pbm, *.pgm, *.ppm, *.pnm),
- TIFF (*.tif, *.tiff).

Because of the limitations of the standard image formats, the filter is capable of saving three-channel images of UInt8 pixel type for all formats, and UInt16 for supported formats only (PNG, TIFF). To alter the format of an image one can use the filters contained in the Image Conversions category.

The **inImageFileFormat** input can be used to explicitly select the file format to be used. When **inImageFileFormat** is set to Auto the recognition of the desired image file format is based on the extension of the file being written, so it is essential that the extension is present and accurate. When extension of the file is not specified, it will be appended according to **inImageFileFormat**.

If the selected file does not exist, it will be created on filter execution. If the selected file does exist, it will be overwritten.

Errors

List of possible exceptions:

Error type	Description
DomainError	Unsupported pixel type in SaveImage.

See Also

- LoadImage – Loads a single image from a file.
- SaveImage_Asynchronous – Saves an image to a file in the background thread.

SaveImageToBuffer

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Saves an image to a file stored in ByteBuffer.

Applications: Use this filter if you want to send image file through I/O communication, e.g. through a Tcplp connection.

Syntax

```
void avl::SaveImageToBuffer
(
 const avl::Image& inImage,
 avl::ImageFileFormat::Type inImageFileFormat,
 avl::ByteBuffer& outBuffer
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		An image to be saved
→	inImageFileFormat	ImageFileFormat::Type		Image file format
←	outBuffer	ByteBuffer&		Buffer containing image file data

Description

This function saves an Image to a common format file similarly to SaveImage, but instead of accessing the file system it stores the file in the memory by writing its content to ByteBuffer.

The operation saves an image to file encoded in one of the standard image file formats. Currently the filter supports the following formats:

- BMP (*.bmp)
- JPEG (*.jpg, *jpeg)
- PNG (*.png),
- PNM (*.pbm, *.pgm, *.ppm, *.pnm),
- TIFF (*.tif, *.tiff).

Because of the limitations of the standard image formats, the filter is capable of saving three-channel images of UInt8 pixel type for all formats, and UInt16 for supported formats only (PNG, TIFF). To alter the format of an image one can use the filters contained in the Image Conversions category.

See Also

- SaveImage – Saves an image to a file.
- LoadImageFromBuffer – Loads a single image from a file stored in memory.

SaveImageToJpeg

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Saves an image to a JPEG file.

Syntax

```
void avl::SaveImageToJpeg
(
 const avl::Image& inImage,
 const atl::File& inFile,
 atl::Optional<int> inQuality,
 bool inIgnoreErrors
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inImage	const Image&			An image to be saved
➡	inFile	const File&			Path to a file
➡	inQuality	Optional<int>	0 - 100	NIL	Quality
➡	inIgnoreErrors	bool			Switches to re-trying and canceling in case of errors (e.g. when the disk is full)

Requirements

For input **inImage** only pixel formats are supported: uint8, uint16.

Read more about pixel formats in Image documentation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported inImage pixel format in SaveImageToJpeg. Supported formats: UInt8, UInt16.

SaveImageToJpeg_Asynchronous

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Saves an image to a JPEG file in the background thread.

Syntax

```
void avl::SaveImageToJpeg_Asynchronous
(
 SaveImageState& ioState,
 int inThreadQueueSize,
 const avl::Image& inImage,
 const atl::File& inFile,
 atl::Optional<int> inQuality,
 bool inIgnoreErrors
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SaveImageState&			Object used to maintain state of the function.
	inThreadQueueSize	int	1 - ∞	3	Number of incoming frames that can be buffered before the thread is able to process them
	inImage	const Image&			An image to be saved
	inFile	const File&			Path to a file
	inQuality	Optional<int>	0 - 100	NIL	Quality
	inIgnoreErrors	bool			If false the error will be reported as soon as the filter instance is again executed

Requirements

For input **inImage** only pixel formats are supported: uint8, uint16.

Read more about pixel formats in Image documentation.

Remarks

This filter is executed in the background thread. Execution errors may be reported with a delay or ignored. Stopping of the program may be delayed, because of waiting for background work to complete.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported inImage pixel format in SaveImageToJpeg_Asynchronous. Supported formats: UInt8, UInt16.

See Also

- LoadImage – Loads a single image from a file.
- SaveImage – Saves an image to a file.

SaveImageToPng

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Saves an image to a PNG file.

Syntax

```
void avl::SaveImageToPng
(
 const avl::Image& inImage,
 const atl::File& inFile,
 atl::Optional<int> inCompressionLevel,
 bool inIgnoreErrors
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inImage	const Image&			An image to be saved
➡	inFile	const File&			Path to a file
➡	inCompressionLevel	Optional<int>	0 - 9	NIL	Compression level
➡	inIgnoreErrors	bool			Switches to re-trying and canceling in case of errors (e.g. when the disk is full)

Requirements

For input **inImage** only pixel formats are supported: uint8, uint16.

Read more about pixel formats in Image documentation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported inImage pixel format in SaveImageToPng. Supported formats: UInt8, UInt16.

SaveImageToPng_Asynchronous

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Saves an image to a PNG file in the background thread.

Syntax

```
void avl::SaveImageToPng_Asynchronous
(
 SaveImageState& ioState,
 int inThreadQueueSize,
 const avl::Image& inImage,
 const atl::File& inFile,
 atl::Optional<int> inCompressionLevel,
 bool inIgnoreErrors
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SaveImageState&			Object used to maintain state of the function.
	inThreadQueueSize	int	1 - ∞	3	Number of incoming frames that can be buffered before the thread is able to process them
	inImage	const Image&			An image to be saved
	inFile	const File&			Path to a file
	inCompressionLevel	Optional<int>	0 - 9	NIL	Compression level
	inIgnoreErrors	bool			If false the error will be reported as soon as the filter instance is again executed

Requirements

For input **inImage** only pixel formats are supported: uint8, uint16.

Read more about pixel formats in Image documentation.

Remarks

This filter is executed in the background thread. Execution errors may be reported with a delay or ignored. Stopping of the program may be delayed, because of waiting for background work to complete.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported inImage pixel format in SaveImageToPng_Asynchronous. Supported formats: UInt8, UInt16.

See Also

- LoadImage – Loads a single image from a file.
- SaveImage – Saves an image to a file.

SaveImageToTiff

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Saves an image to a TIFF file.

Syntax

```
void avl::SaveImageToTiff
(
 const avl::Image& inImage,
 const atl::File& inFile,
 atl::Optional<avl::TiffImageCompressionScheme::Type> inCompressionScheme,
 atl::Optional<int> inJpegQuality,
 bool inIgnoreErrors
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inImage	const Image&			An image to be saved
➡	inFile	const File&			Path to a file
➡	inCompressionScheme	Optional<TiffImageCompressionScheme::Type>		NIL	Compression scheme
➡	inJpegQuality	Optional<int>	0 - 100	NIL	Quality - used only for JPEG compression scheme
➡	inIgnoreErrors	bool			Switches to re-trying and canceling in case of errors (e.g. when the disk is full)

Requirements

For input **inImage** only pixel formats are supported: uint8, uint16, real, int32.

Read more about pixel formats in Image documentation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Compression is not supported for 32-bit pixel type in SaveImageToTiff.
DomainError	inJpegQuality set in SaveImageToTiff is valid only with inCompressionScheme set as JPEG.
DomainError	Path name cannot be empty in SaveImageToTiff.
DomainError	Not supported inImage pixel format in SaveImageToTiff. Supported formats: UInt8, UInt16, Real, Int32.

SaveImageToTiff_Asynchronous

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Saves an image to a TIFF file in the background thread.

Syntax

```
void avl::SaveImageToTiff_Asynchronous
(
 SaveImageState& ioState,
 int inThreadQueueSize,
 const avl::Image& inImage,
 const atl::File& inFile,
 atl::Optional<avl::TiffImageCompressionScheme::Type> inCompressionScheme,
 atl::Optional<int> inJpegQuality,
 bool inIgnoreErrors
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SaveImageState&			Object used to maintain state of the function.
	inThreadQueueSize	int	1 - ∞	3	Number of incoming frames that can be buffered before the thread is able to process them
	inImage	const Image&			An image to be saved
	inFile	const File&			Path to a file
	inCompressionScheme	Optional<TiffImageCompressionScheme::Type>		NIL	Compression scheme
	inJpegQuality	Optional<int>	0 - 100	NIL	Quality (0-100) - used only for JPEG compression scheme
	inIgnoreErrors	bool			If false the error will be reported as soon as the filter instance is again executed

Requirements

For input **inImage** only pixel formats are supported: uint8, uint16, real, int32.

Read more about pixel formats in Image documentation.

Remarks

This filter is executed in the background thread. Execution errors may be reported with a delay or ignored. Stopping of the program may be delayed, because of waiting for background work to complete.

Errors

List of possible exceptions:

Error type	Description
DomainError	Compression is not supported for 32-bit pixel type in SaveImageToTiff_Asynchronous.
DomainError	inJpegQuality set in SaveImageToTiff_Asynchronous is valid only with inCompressionScheme set as JPEG.
DomainError	Path name cannot be empty in SaveImageToTiff_Asynchronous.
DomainError	Not supported inImage pixel format in SaveImageToTiff_Asynchronous. Supported formats: UInt8, UInt16, Real, Int32.

See Also

- LoadImage – Loads a single image from a file.
- SaveImage – Saves an image to a file.

SaveImage_Asynchronous

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Saves an image to a file in the background thread.

Syntax

```
void avl::SaveImage_Asynchronous
(
 SaveImageState& ioState,
 int inThreadQueueSize,
 const avl::Image& inImage,
 atl::Optional<avl::ImageFileFormat::Type> inImageFileFormat,
 const atl::File& inFile,
 bool inIgnoreErrors
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SaveImageState&			Object used to maintain state of the function.
	inThreadQueueSize	int	1 - ∞	3	Number of incoming frames that can be buffered before the thread is able to process them
	inImage	const Image&			An image to be saved
	inImageFileFormat	Optional<ImageFileFormat::Type>		NIL	If Nil the format will be chosen on the basis of extension
	inFile	const File&			Path to a file
	inIgnoreErrors	bool			If false the error will be reported as soon as the filter instance is again executed

Description

The operation saves an image to file encoded in one of the standard image file formats. Currently the filter supports the following formats:

- BMP (*.bmp)
- JPEG (*.jpg, *jpeg)
- PNG (*.png),
- PNM (*.pbm, *.pgm, *.ppm, *.pnm),
- TIFF (*.tif, *.tiff).

Because of the limitations of the standard image formats, the filter is capable of saving three-channel images of UInt8 pixel type for all formats, and UInt16 for supported formats only (PNG, TIFF). To alter the format of an image one can use the filters contained in the Image Conversions category.

The **inImageFileFormat** input can be used to explicitly select the file format to be used. When **inImageFileFormat** is set to Auto the recognition of the desired image file format is based on the extension of the file being written, so it is essential that the extension is present and accurate. When extension of the file is not specified, it will be appended according to **inImageFileFormat**.

If the selected file does not exist, it will be created on filter execution. If the selected file does exist, it will be overwritten.

Remarks

This filter is executed in the background thread. Execution errors may be reported with a delay or ignored. Stopping of the program may be delayed, because of waiting for background work to complete.

Errors

List of possible exceptions:

Error type	Description
DomainError	Unsupported pixel type in SaveImage_Asynchronous.

See Also

- LoadImage – Loads a single image from a file.
- SaveImage – Saves an image to a file.

Also in AVL Lite

WriteVideoStream

Header: AVL.h
Namespace: avl
Module: FoundationLite

Writes an image to a previously opened video stream.

Syntax

```
void avl::WriteVideoStream
(
 const avl::OutputVideoStream& inOutputVideoStream,
 const avl::Image& inImage
)
```

Parameters

	Name	Type	Default	Description
→	inOutputVideoStream	const OutputVideoStream&		Previously opened output video stream
→	inImage	const Image&		Frame to be added to stream

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrectly initialized OutputVideoStream object passed to WriteVideoStream
DomainError	Incorrectly OutputVideoStream object passed to WriteVideoStream

Also in AVL Lite

DecodeVideo

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an image from video file with using FFmpeg library.

Syntax

```
bool avl::DecodeVideo
(
 DecodeVideo_State& ioState,
 const atl::File& inFilename,
 const atl::Optional<atl::int64> inStartFrameId,
 avl::Image& outImage,
 atl::int64& outFrameId
)
```

Parameters

	Name	Type	Default	Description
↔	ioState	DecodeVideo_State&		Object used to maintain state of the function.
→	inFilename	const File&		
→	inStartFrameId	const Optional<int64>	NIL	
←	outImage	Image&		Captured frame
←	outFrameId	int64&		

Remarks

Video decoder software

This filter is intended to convert video file to series of images using FFmpeg library. It is required to install FFmpeg software.

FFmpeg can be downloaded from the following website: https://ffmpeg.org/download.html. Please to download "shared" version of FFmpeg.

Recommended FFmpeg version for Aurora Vision Studio usage is **4.1.3**.

Add DLL path to system environment variable may be required.

This filter is not able to decode all video formats supported by FFmpeg.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

5. Histogram Point Transforms

Table of content:

AbsoluteHistogram

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Transforms each bin value to its absolute value.

Syntax

```
void avl::AbsoluteHistogram
(
 const avl::Histogram& inHistogram,
 avl::Histogram& outHistogram
)
```

Parameters

	Name	Type	Default	Description
→	inHistogram	const Histogram&		Input histogram
←	outHistogram	Histogram&		Output histogram

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inHistogram** and **outHistogram**

Read more about In-place Computation.

AddToHistogram

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Increases each bin value by a number.

Syntax

```
void avl::AddToHistogram
(
 const avl::Histogram& inHistogram,
 const double inValue,
 avl::Histogram& outHistogram
)
```

Parameters

	Name	Type	Default	Description
→	inHistogram	const Histogram&		Input histogram
→	inValue	const double	0.0D	Input value
←	outHistogram	Histogram&		Output histogram

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inHistogram** and **outHistogram**

Read more about In-place Computation.

DivideHistogram

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Divides each bin value by a number.

Syntax

```
void avl::DivideHistogram
(
 const avl::Histogram& inHistogram,
 const double inValue,
 avl::Histogram& outHistogram
)
```

Parameters

	Name	Type	Default	Description
→	inHistogram	const Histogram&		Input histogram
→	inValue	const double	2.0D	Input value
←	outHistogram	Histogram&		Output histogram

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inHistogram** and **outHistogram**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Divisor is equal to zero on input in DivideHistogram.

MultiplyHistogram

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Multiplies each bin value by a number.

Syntax

```
void avl::MultiplyHistogram
(
 const avl::Histogram& inHistogram,
 const double inValue,
 avl::Histogram& outHistogram
)
```

Parameters

	Name	Type	Default	Description
→	inHistogram	const Histogram&		Input histogram
→	inValue	const double	2.0D	Input value
←	outHistogram	Histogram&		Output histogram

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inHistogram** and **outHistogram**

Read more about In-place Computation.

NegateHistogram

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Negates each bin value.

Syntax

```
void avl::NegateHistogram
(
 const avl::Histogram& inHistogram,
 avl::Histogram& outHistogram
)
```

Parameters

	Name	Type	Default	Description
→	inHistogram	const Histogram&		Input histogram
←	outHistogram	Histogram&		Output histogram

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inHistogram** and **outHistogram**

Read more about In-place Computation.

SubtractFromHistogram

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Decreases each bin value by a number.

Syntax

```
void avl::SubtractFromHistogram
(
 const avl::Histogram& inHistogram,
 const double inValue,
 avl::Histogram& outHistogram
)
```

Parameters

	Name	Type	Default	Description
→	inHistogram	const Histogram&		Input histogram
→	inValue	const double	2.0D	Input value
←	outHistogram	Histogram&		Output histogram

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inHistogram** and **outHistogram**

Read more about In-place Computation.

6. Profile Point Transforms

Table of content:

AbsoluteProfile

Header: AVL.h
Namespace: avl
Module: FoundationPro

Transforms each profile value to its absolute value.

Syntax

```
void avl::AbsoluteProfile
(
 avl::Profile& ioProfile,
 atl::Optional<const avl::Range&> inRange
)
```

Parameters

	Name	Type	Default	Description
	ioProfile	Profile&		
	inRange	Optional<const Range&>	NIL	

Errors

List of possible exceptions:

Error type	Description
DomainError	Range exceeds the input profile in AbsoluteProfile.

AddToProfile

Header: AVL.h
Namespace: avl
Module: FoundationPro

Adds a scalar value to each element of a profile.

Syntax

```
void avl::AddToProfile
(
 avl::Profile& ioProfile,
 atl::Optional<const avl::Range&> inRange,
 float inValue
)
```

Parameters

	Name	Type	Default	Description
	ioProfile	Profile&		
	inRange	Optional<const Range&>	NIL	
	inValue	float		

Errors

List of possible exceptions:

Error type	Description
DomainError	Range exceeds the input profile in AddToProfile.

ClipProfileValues

Header: AVL.h
Namespace: avl
Module: FoundationPro

Applies limits to profile values.

Syntax

```
void avl::ClipProfileValues
(
 avl::Profile& ioProfile,
 atl::Optional<const avl::Range&> inRange,
 const atl::Optional<float> inLowValue,
 const atl::Optional<float> inHighValue
)
```

Parameters

	Name	Type	Default	Description
	ioProfile	Profile&		
	inRange	Optional<const Range&>	NIL	
	inLowValue	const Optional<float>	NIL	Value min threshold
	inHighValue	const Optional<float>	NIL	Value max threshold

Errors

List of possible exceptions:

Error type	Description
DomainError	Range exceeds the input profile in ClipProfileValues.
DomainError	Value of inHighValue is less than inLowValue in ClipProfileValues.

DivideProfile

Header: AVL.h
Namespace: avl
Module: FoundationPro

Divides each element of a profile by a scalar value.

Syntax

```
void avl::DivideProfile
(
 avl::Profile& ioProfile,
 atl::Optional<const avl::Range&> inRange,
 float inValue
)
```

Parameters

	Name	Type	Default	Description
	ioProfile	Profile&		
	inRange	Optional<const Range&>	NIL	
	inValue	float	2.0f	

Errors

List of possible exceptions:

Error type	Description
DomainError	Divisor is equal to zero on input in DivideProfile.
DomainError	Range exceeds the input profile in DivideProfile.

MultiplyProfile

Header: AVL.h
Namespace: avl
Module: FoundationPro

Multiplies each element of a profile by a scalar value.

Syntax

```
void avl::MultiplyProfile
(
 avl::Profile& ioProfile,
 atl::Optional<const avl::Range&> inRange,
 float inValue
)
```

Parameters

	Name	Type	Default	Description
	ioProfile	Profile&		
	inRange	Optional<const Range&>	NIL	
	inValue	float	2.0f	

Errors

List of possible exceptions:

Error type	Description
DomainError	Range exceeds the input profile in MultiplyProfile.

NegateProfile

Header: AVL.h
Namespace: avl
Module: FoundationPro

Transforms each profile value to its negation.

Syntax

```
void avl::NegateProfile
(
 avl::Profile& ioProfile,
 atl::Optional<const avl::Range&> inRange
)
```

Parameters

	Name	Type	Default	Description
	ioProfile	Profile&		
	inRange	Optional<const Range&>	NIL	

Errors

List of possible exceptions:

Error type	Description
DomainError	Range exceeds the input profile in NegateProfile.

NormalizeProfile

Header: AVL.h
Namespace: avl
Module: FoundationPro

Rescales a profile linearly, so that its minimum becomes inNewMinimum and its maximum becomes inNewMaximum.

Syntax

```
void avl::NormalizeProfile
(
 const avl::Profile& inProfile,
 atl::Optional<const avl::Range&> inRange,
 atl::Optional<float> inNewMinimum,
 atl::Optional<float> inNewMaximum,
 const float inSaturateHighestFraction,
 const float inSaturateLowestFraction,
 avl::Profile& outProfile,
 float& outA,
 float& outB
)
```

Parameters

	Name	Type	Range	Default	Description
→	inProfile	const Profile&			Input profile
→	inRange	Optional<const Range&>		NIL	
→	inNewMinimum	Optional<float>		NIL	Desired minimum value of the resulting profile (if set to Nil, the minimum of the input profile is used)
→	inNewMaximum	Optional<float>		NIL	Desired maximum value of the resulting profile (if set to Nil, the maximum of the input profile is used)
→	inSaturateHighestFraction	const float	0.0 - 1.0	0.0f	Fraction of the highest values skipped during normalization
→	inSaturateLowestFraction	const float	0.0 - 1.0	0.0f	Fraction of the lowest values skipped during normalization
←	outProfile	Profile&			Normalized profile
←	outA	float&			Multiplicative parameter of the applied linear transformation
←	outB	float&			Additive parameter of the applied linear transformation

Errors

List of possible exceptions:

Error type	Description
DomainError	Range exceeds inProfile in NormalizeProfile.
DomainError	The sum of inSaturateHighestFraction and inSaturateLowestFraction can't be greater than 1 in NormalizeProfile.

RescaleProfile

Header: AVL.h
Namespace: avl
Module: FoundationPro

Applies a linear transformation to a profile.

Syntax

```
void avl::RescaleProfile
(
 avl::Profile& ioProfile,
 atl::Optional<const avl::Range&> inRange,
 float inA,
 float inB
)
```

Parameters

	Name	Type	Default	Description
	ioProfile	Profile&		
	inRange	Optional<const Range&>	NIL	
	inA	float	1.0f	Value multiplied
	inB	float	0.0f	Value added

Errors

List of possible exceptions:

Error type	Description
DomainError	Range exceeds the input profile in RescaleProfile.

SubtractFromProfile

Header: AVL.h
Namespace: avl
Module: FoundationPro

Subtracts a scalar value from each element of a profile.

Syntax

```
void avl::SubtractFromProfile
(
 avl::Profile& ioProfile,
 atl::Optional<const avl::Range&> inRange,
 float inValue
)
```

Parameters

	Name	Type	Default	Description
	ioProfile	Profile&		
	inRange	Optional<const Range&>	NIL	
	inValue	float		

Errors

List of possible exceptions:

Error type	Description
DomainError	Range exceeds the input profile in SubtractFromProfile.

7. Surface Spatial Transforms

Table of content:

AbsoluteSurface

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Transforms Z coordinate of each surface point to its absolute value.

Syntax

```
void avl::AbsoluteSurface
(
 const avl::Surface& inSurface,
 avl::Surface& outSurface
)
```

Parameters

	Name	Type	Default	Description
→	inSurface	const Surface&		Input surface
←	outSurface	Surface&		Output surface

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

CloseSurfacePoints

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Reconstructs missing points of the input surface by interpolating neighboring points.

Syntax

```
void avl::CloseSurfacePoints
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 avl::KernelShape::Type inKernel,
 int inRadiusX,
 atl::Optional<int> inRadiusY,
 avl::Surface& outSurface
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inRoi	Optional<const Region&>		NIL	Region of interest
→	inKernel	KernelShape::Type			Kernel shape (predefined)
→	inRadiusX	int	0 - ∞	1	Nearly half of the kernel's width (2*R+1)
→	inRadiusY	Optional<int>	0 - ∞	NIL	Nearly half of the kernel's height (2*R+1), or same as inRadiusX
←	outSurface	Surface&			Output surface

Description

The operation reconstructs some of missing points of the input surface by interpolating their neighboring points. Internally the region of surface valid points is closed using defined kernel. Every missing point in location that have emerged because of the closing is substituted with the arithmetic mean of existing points in locations that would be covered by the kernel if the middle of the kernel was precisely over the missing point location.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region of interest exceeds an input surface in CloseSurfacePoints.

See Also

- SurfaceValidPointsRegion – Computes region of locations where points are valid in a surface and where they are invalid.
- CloseRegion – Performs a morphological closing on a region using selected predefined kernel.

CropSurface

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Removes from the surface points that are not contained in a given rectangular box.

Syntax

```
void avl::CropSurface
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 const avl::ValueLimits_f64& inXLimits,
 const avl::ValueLimits_f64& inYLimits,
 const avl::ValueLimits_f64& inZLimits,
 bool inPreserveDimensions,
 avl::Surface& outSurface,
 atl::Optional<avl::Region&> outRejected = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inSurface	const Surface&		Input surface
→	inRoi	Optional<const Region&>	NIL	Region of interest
→	inXLimits	const ValueLimits_f64&		
→	inYLimits	const ValueLimits_f64&		
→	inZLimits	const ValueLimits_f64&		
→	inPreserveDimensions	bool	False	Flag indicating whether the surface dimensions should be preserved or not
←	outSurface	Surface&		Output surface
←	outRejected	Optional<Region&>	NIL	Region of locations where points are not contained in a given rectangular box

Optional Outputs

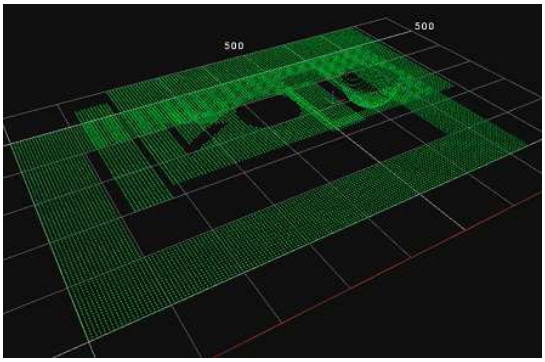
The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outRejected**.

Read more about Optional Outputs.

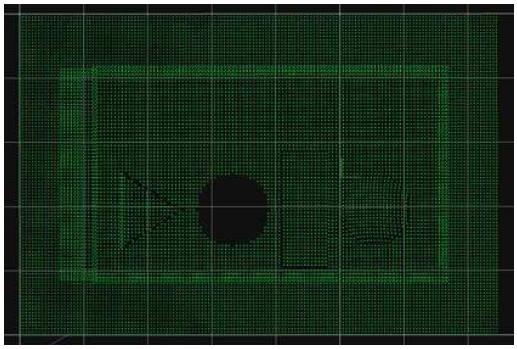
Description

The operation removes points from surface that are not contained in a given rectangular box within provided coordinates.
It is also possible to narrow this box by editing **inRoi**, Region of Interest of operation.
Coordinates are defined via acceptable limits for their values. E.g. **inXLimits.MinValue** is minimum and **inXLimits.MaxValue** is maximum value of the X coordinate. Every point with its X coordinate outside of this range is removed.
Flag **inPreserveDimensions** informs if the operation output should keep the original dimensions of its input. E.g. if **inSurface** has its Width=300, Height=200 and Pitch=608 and the flag is enabled, the **outSurface** will have exactly the same dimensions of 300x200x608. However, if the flag is disabled, **outSurface** will have its dimensions smaller to adjust them to surface size, but its XOffset, YOffset, ZOffset properties will be accordingly bigger in order to keep the surface in the same place in workspace.
Operation returns the cropped Surface as well as removed Region if one needs to perform further actions on removed points.

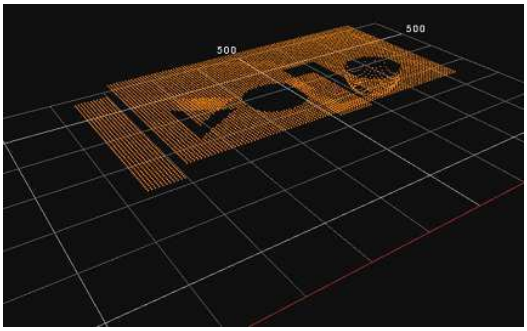
Examples



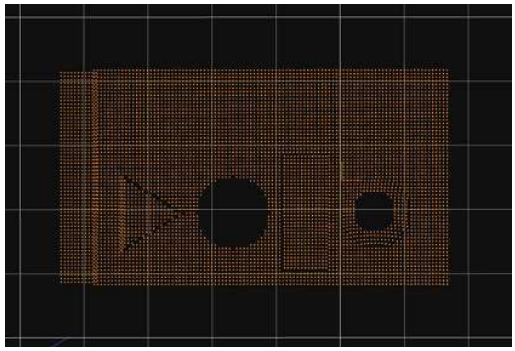
A sample Surface object.



The same Surface object seen from above.



A sample Surface object cropped through half of its height.



Cropped Surface object seen from above.

Name	Type	Value
⊟ (2)CropSurface		
⊟ outSurface	Surface	{...}
Width	Integer	150
Height	Integer	100
Pitch	Integer	304
Type	PlainType	Int16
PointByteSize	Integer	2
XOffset	Double	0,000
XScale	Double	5,000
YOffset	Double	0,000
YScale	Double	5,000
ZOffset	Double	0,000
ZScale	Double	1,000

Properties of a cropped surface with dimension preservation enabled.

Name	Type	Value
⊟ (3)CropSurface		
⊟ outSurface	Surface	{...}
Width	Integer	111
Height	Integer	61
Pitch	Integer	224
Type	PlainType	Int16
PointByteSize	Integer	2
XOffset	Double	100,000
XScale	Double	5,000
YOffset	Double	100,000
YScale	Double	5,000
ZOffset	Double	0,000
ZScale	Double	1,000

Properties of a cropped surface with dimension preservation disabled.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region of interest exceeds an input surface in CropSurface.

See Also

- CropSurfaceToRegion – Removes points that are not present in a given region.
- TestSurface – Returns a sample 3D surface.
- Point3DGrid

CropSurfaceByNeighborsProximity

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Removes from the surface points that are too distant from their neighbor points.

Syntax

```
void avl::CropSurfaceByNeighborsProximity
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 const int inNeighborRadius,
 const float inMaxDistance,
 avl::Metric3D::Type inMetric,
 const float inMinNeighborRatio,
 bool inPreserveDimensions,
 avl::Surface& outSurface,
 atl::Optional<avl::Region&> outRejected = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			
→	inRoi	Optional<const Region&>		NIL	Region of interest
→	inNeighborRadius	const int	1 - ∞	1	Radius of neighbors to search for real neighbors
→	inMaxDistance	const float	0.0 - ∞	2.0f	Maximal distance from another point to consider them real neighbors
→	inMetric	Metric3D::Type		Z	Metric used for measuring distance between points
→	inMinNeighborRatio	const float	0.0 - 1.0	1.0f	Fraction of valid neighbors in a given radius that have to be real neighbors
→	inPreserveDimensions	bool		False	Flag indicating whether the surface dimensions should be preserved or not
←	outSurface	Surface&			
←	outRejected	Optional<Region&>		NIL	Region of locations where points are too distant from their neighbors

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outRejected**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect metric in CropSurfaceByNeighborsProximity.
DomainError	Region of interest exceeds an input surface in CropSurfaceByNeighborsProximity.

CropSurfaceByPlaneProximity

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Removes from the surface points that are too distant from a given plane.

Syntax

```
void avl::CropSurfaceByPlaneProximity
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 const avl::Plane3D& inPlane,
 atl::Optional<float> inMinDistance,
 atl::Optional<float> inMaxDistance,
 bool inPreserveDimensions,
 avl::Surface& outSurface,
 atl::Optional<avl::Region&> outRejected = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inSurface	const Surface&		Input surface
→	inRoi	Optional<const Region&>	NIL	Region of interest
→	inPlane	const Plane3D&		Plane to which distance is measured
→	inMinDistance	Optional<float>	NIL	Minimal distance from a given plane
→	inMaxDistance	Optional<float>	1.0f	Maximal distance from a given plane
→	inPreserveDimensions	bool	False	Flag indicating whether the surface dimensions should be preserved or not
←	outSurface	Surface&		Output surface
←	outRejected	Optional<Region&>	NIL	Region of locations where points are too distant from the input plane

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outRejected**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite plane on input in CropSurfaceByPlaneProximity.
DomainError	Region of interest exceeds an input surface in CropSurfaceByPlaneProximity.

CropSurfaceToBox3D

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Removes from the surface points that are not contained in a given box in 3D.

Syntax

```
void avl::CropSurfaceToBox3D
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 const avl::Box3D& inBox3D,
 bool inPreserveDimensions,
 avl::Surface& outSurface,
 atl::Optional<avl::Region&> outRejected = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inSurface	const Surface&		Input surface
→	inRoi	Optional<const Region&>	NIL	Region of interest
→	inBox3D	const Box3D&		Box defining a subspace to be cropped
→	inPreserveDimensions	bool	False	Flag indicating whether the surface dimensions should be preserved or not
←	outSurface	Surface&		Output surface
←	outRejected	Optional<Region&>	NIL	Region of locations where points are not contained in a given box in 3D

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outRejected**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region of interest exceeds an input surface in CropSurfaceToBox3D.

CropSurfaceToRegion

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Removes points that are not present in a given region.

Syntax

```
void avl::CropSurfaceToRegion
(
 const avl::Surface& inSurface,
 const avl::Region& inRegion,
 bool inPreserveDimensions,
 avl::Surface& outSurface
)
```

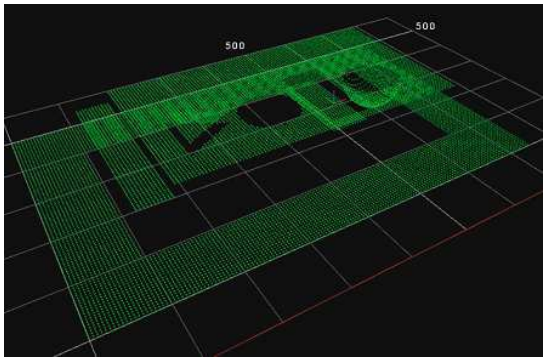
Parameters

	Name	Type	Default	Description
→	inSurface	const Surface&		Input surface
→	inRegion	const Region&		Region from which the points are not removed
→	inPreserveDimensions	bool	False	Flag indicating whether the surface dimensions should be preserved or not
←	outSurface	Surface&		Output surface

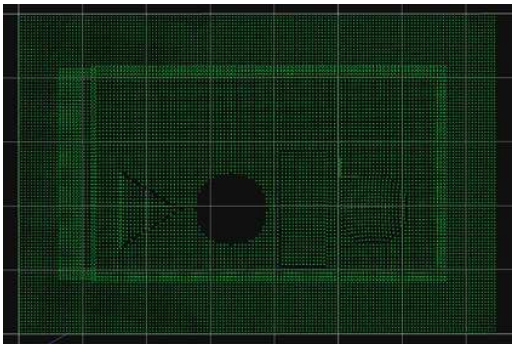
Description

The operation removes points from surface that are not contained in a given rectangular box within provided Region.
This operation is very similar to CropSurface, although it is slightly limited. Region within the surface will be cropped is viewed from above. This means that it is possible to edit the input within a two-dimensional space through the X and Y axis instead of X, Y and Z axis in CropSurface filter.
Flag **inPreserveDimensions** informs if the operation output should keep the original dimensions of its input. E.g. if **inSurface** has its Width=300, Height=200 and Pitch=608 and the flag is enabled, the **outSurface** will have exactly the same dimensions of 300x200x608. However, if the flag is disabled, **outSurface** will have its dimensions smaller to adjust them to surface size, but its XOffset, YOffset, ZOffset properties will be accordingly bigger in order to keep the surface in the same place in workspace.

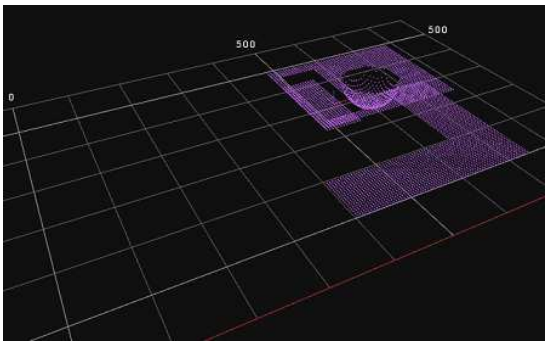
Examples



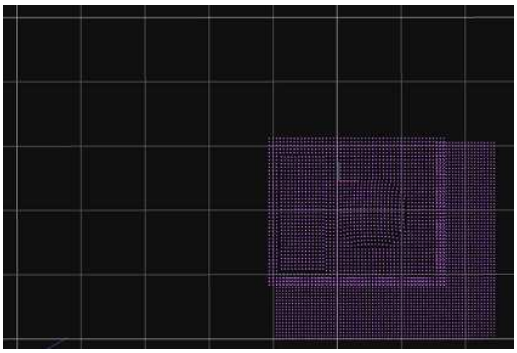
A sample Surface object.



The same Surface object seen from above.



A sample Surface object cropped within the Region of a bottom-right part of surface.



Cropped Surface object seen from above.

Name	Type	Value
⊟ (5)CropSurfaceToRegion		
⊟ outSurface	Surface	{...}
Width	Integer	150
Height	Integer	100
Pitch	Integer	304
Type	PlainType	Int16
PointByteSize	Integer	2
XOffset	Double	0,000
XScale	Double	5,000
YOffset	Double	0,000
YScale	Double	5,000
ZOffset	Double	0,000
ZScale	Double	1,000

Properties of a cropped surface with dimension preservation enabled.

Name	Type	Value
⊟ (4)CropSurfaceToRegion		
⊟ outSurface	Surface	{...}
Width	Integer	69
Height	Integer	62
Pitch	Integer	144
Type	PlainType	Int16
PointByteSize	Integer	2
XOffset	Double	405,000
XScale	Double	5,000
YOffset	Double	0,000
YScale	Double	5,000
ZOffset	Double	0,000
ZScale	Double	1,000

Properties of a cropped surface with dimension preservation disabled.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Input region exceeds surface dimensions in CropSurfaceToRegion.

See Also

- CropSurface – Removes from the surface points that are not contained in a given rectangular box.
- TestSurface – Returns a sample 3D surface.
- Point3DGrid

CropSurface_Dynamic

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Removes from the surface points that are relatively too distant from their average neighbor points in a local rectangular neighborhood.

Syntax

```
void avl::CropSurface_Dynamic
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 int inRadiusX,
 atl::Optional<int> inRadiusY,
 atl::Optional<float> inMinRelativeHeight,
 atl::Optional<float> inMaxRelativeHeight,
 float inHysteresis,
 bool inPreserveDimensions,
 avl::Surface& outSurface,
 atl::Optional<avl::Region&> outRejected = atl::NIL,
 avl::Surface& diagBaseSurface = atl::Dummy<avl::Surface>()
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inSurface	const Surface&			Input surface
➡	inRoi	Optional<const Region&>		NIL	Region of interest
➡	inRadiusX	int	0 - ∞	1	Horizontal radius of internal mean blur
➡	inRadiusY	Optional<int>	0 - ∞	NIL	Vertical radius of internal mean blur (Auto = inRadiusX)
➡	inMinRelativeHeight	Optional<float>		NIL	Minimum relative height of a point to be not removed
➡	inMaxRelativeHeight	Optional<float>		NIL	Maximum relative height of a point to be not removed
➡	inHysteresis	float	0.0 - ∞	0.0f	Defines how much the threshold criteria are lowered for points neighboring with other not removed points
➡	inPreserveDimensions	bool		False	Flag indicating whether the surface dimensions should be preserved or not
⬅	outSurface	Surface&			
⬅	outRejected	Optional<Region&>		NIL	Region of locations where points are removed
🔍	diagBaseSurface	Surface&			Diagnostic blurred input surface

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outRejected**.

Read more about Optional Outputs.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region of interest exceeds an input surface in CropSurface_Dynamic.

CropSurface_Relative

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Removes from the surface points that are relatively too distant from their average neighbor points in a local rectangular neighborhood.

Syntax

```
void avl::CropSurface_Relative
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 const avl::Surface& inBaseSurface,
 atl::Optional<float> inMinRelativeHeight,
 atl::Optional<float> inMaxRelativeHeight,
 float inHysteresis,
 bool inPreserveDimensions,
 avl::Surface& outSurface,
 atl::Optional<avl::Region&> outRejected = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inRoi	Optional<const Region&>		NIL	Region of interest
→	inBaseSurface	const Surface&			Heights of this surface are subtracted from inSurface heights before crop
→	inMinRelativeHeight	Optional<float>		NIL	Minimum relative height of a point to be not removed
→	inMaxRelativeHeight	Optional<float>		NIL	Maximum relative height of a point to be not removed
→	inHysteresis	float	0.0 - ∞	0.0f	Defines how much the threshold criteria are lowered for points neighboring with other not removed points
→	inPreserveDimensions	bool		False	Flag indicating whether the surface dimensions should be preserved or not
←	outSurface	Surface&			
←	outRejected	Optional<Region&>		NIL	Region of locations where points are removed

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outRejected**.

Read more about Optional Outputs.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region of interest exceeds an input surface in CropSurface_Relative.
DomainError	Surface formats are not the same in CropSurface_Relative.
DomainError	Surface sizes are not equal in CropSurface_Relative.

DilateSurfacePoints

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Reconstructs missing points of the input surface by interpolating neighboring points.

Syntax

```
void avl::DilateSurfacePoints
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 avl::KernelShape::Type inKernel,
 int inRadiusX,
 atl::Optional<int> inRadiusY,
 avl::Surface& outSurface
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inRoi	Optional<const Region&>		NIL	Region of interest
→	inKernel	KernelShape::Type			Kernel shape (predefined)
→	inRadiusX	int	0 - ∞	1	Nearly half of the kernel's width (2*R+1)
→	inRadiusY	Optional<int>	0 - ∞	NIL	Nearly half of the kernel's height (2*R+1), or same as inRadiusX
←	outSurface	Surface&			Output surface

Description

The operation reconstructs some of missing points of the input surface by interpolating their neighboring points. Internally the region of surface valid points is dilated using defined kernel. Every missing point in location that have emerged because of the dilation is substituted with the arithmetic mean of existing points in locations that would be covered by the kernel if the middle of the kernel was precisely over the missing point location.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region of interest exceeds an input surface in DilateSurfacePoints.

See Also

- SurfaceValidPointsRegion – Computes region of locations where points are valid in a surface and where they are invalid.
- DilateRegion – Performs a morphological dilation on a region using a predefined kernel.

ErodeSurfacePoints

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Removes some existing points from the input surface when some points in their vicinity are missing.

Syntax

```
void avl::ErodeSurfacePoints
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 avl::KernelShape::Type inKernel,
 int inRadiusX,
 atl::Optional<int> inRadiusY,
 bool inPreserveDimensions,
 avl::Surface& outSurface
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inSurface	const Surface&			Input surface
➡	inRoi	Optional<const Region&>		NIL	Region of interest
➡	inKernel	KernelShape::Type			Kernel shape (predefined)
➡	inRadiusX	int	0 - ∞	1	Nearly half of the kernel's width (2*R+1)
➡	inRadiusY	Optional<int>	0 - ∞	NIL	Nearly half of the kernel's height (2*R+1), or same as inRadiusX
➡	inPreserveDimensions	bool			Flag indicating whether the surface dimensions should be preserved or not
⬅	outSurface	Surface&			Output surface

Description

The operation removes some existing points from the input surface when some points in their vicinity are missing. Internally the region of surface valid points is eroded using defined kernel. The surface points in locations that have vanished because of the erosion are removed and substituted with indefinite points.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region of interest exceeds an input surface in ErodeSurfacePoints.

See Also

- SurfaceValidPointsRegion – Computes region of locations where points are valid in a surface and where they are invalid.
- ErodeRegion – Performs a morphological erosion on a region using a predefined kernel.

FlattenSurface

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Flattens a curved surface.

Syntax

```
void avl::FlattenSurface
(
 const avl::Surface& inSurface,
 const avl::Axis::Type inCurvatureAxis,
 const float inStdDev,
 avl::Surface& outSurface,
 avl::Surface& diagSmoothedSurface
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inCurvatureAxis	const Axis::Type			Axis along which the input surface is curved
→	inStdDev	const float	0.0 - ∞		Standard deviation for smoothing of the surface
←	outSurface	Surface&			Output surface
←	diagSmoothedSurface	Surface&			Surface smoothed with a Gaussian kernel

Errors

List of possible exceptions:

Error type	Description
DomainError	Unknown axis type in FlattenSurface.

FlattenSurface_WithScalePreserving

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Flattens a curved surface preserving the scale on the axes.

Syntax

```
void avl::FlattenSurface_WithScalePreserving
(
 const avl::Surface& inSurface,
 const avl::Axis::Type inCurvatureAxis,
 const float inStdDev,
 avl::Point3DGrid& outFlattenedGrid,
 avl::Surface& diagSmoothedSurface
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inCurvatureAxis	const Axis::Type			Axis along which the input surface is curved
→	inStdDev	const float	0.0 - ∞		Standard deviation for smoothing of the surface
←	outFlattenedGrid	Point3DGrid&			Output point grid
←	diagSmoothedSurface	Surface&			Surface smoothed with a Gaussian kernel

Errors

List of possible exceptions:

Error type	Description
DomainError	Unknown axis type in FlattenSurface_WithScalePreserving.

JoinSurfaces

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Combines two surfaces into one.

Syntax

```
void avl::JoinSurfaces
(
 const avl::Surface& inSurface1,
 const avl::Surface& inSurface2,
 avl::Surface& outSurface
)
```

Parameters

	Name	Type	Default	Description
→	inSurface1	const Surface&		First input surface to join
→	inSurface2	const Surface&		Second input surface to join
←	outSurface	Surface&		Combination of two input surfaces

Remarks

The joined surfaces must be compatible. Their scales must be equal and the difference between the corresponding offsets must be equal to the multiple of the related scale.

The filter will return the mean of the points where the surfaces overlap each other.

Errors

List of possible exceptions:

Error type	Description
DomainError	Different axis scales in JoinSurfaces.
DomainError	Different Surface types in JoinSurfaces.
DomainError	Incompatible axis offsets in JoinSurfaces.

MirrorSurface

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Transforms Z coordinate of each surface point to its opposite value.

Syntax

```
void avl::MirrorSurface
(
 const avl::Surface& inSurface,
 avl::Surface& outSurface,
 atl::Optional<avl::Matrix&> outTransform = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inSurface	const Surface&		Input surface
←	outSurface	Surface&		Mirrored surface
←	outTransform	Optional<Matrix&>	NIL	Matrix of the transform used to mirror the surface

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outTransform**.

Read more about Optional Outputs.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

OpenSurfacePoints

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Removes some existing points from the input surface when some points in their vicinity are missing.

Syntax

```
void avl::OpenSurfacePoints
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 avl::KernelShape::Type inKernel,
 int inRadiusX,
 atl::Optional<int> inRadiusY,
 bool inPreserveDimensions,
 avl::Surface& outSurface
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inRoi	Optional<const Region&>		NIL	Region of interest
→	inKernel	KernelShape::Type			Kernel shape (predefined)
→	inRadiusX	int	0 - ∞	1	Nearly half of the kernel's width (2*R+1)
→	inRadiusY	Optional<int>	0 - ∞	NIL	Nearly half of the kernel's height (2*R+1), or same as inRadiusX
→	inPreserveDimensions	bool			Flag indicating whether the surface dimensions should be preserved or not
←	outSurface	Surface&			Output surface

Description

The operation removes some existing points from the input surface when some points in their vicinity are missing. Internally the region of surface valid points is opened using defined kernel. The surface points in locations that have vanished because of the opening are removed and substituted with indefinite points.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region of interest exceeds an input surface in OpenSurfacePoints.

See Also

- SurfaceValidPointsRegion – Computes region of locations where points are valid in a surface and where they are invalid.
- OpenRegion – Performs a morphological opening on a region using a predefined kernel.

ProjectSurfaceOntoPlane

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Projects a surface on another plane, replacing Z coordinate of a point with its distance from a given plane.

Syntax

```
void avl::ProjectSurfaceOntoPlane
(
 const avl::Surface& inSurface,
 const avl::Plane3D& inPlane,
 const bool inUseAbsoluteDistance,
 const bool inPreserveOffsetsAndScales,
 avl::Surface& outSurface,
 atl::Optional<avl::Matrix&> outTransform = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inSurface	const Surface&		
→	inPlane	const Plane3D&		
→	inUseAbsoluteDistance	const bool		
→	inPreserveOffsetsAndScales	const bool	True	Flag indicating whether to preserve input surface offsets and scales or to adjust them accordingly
←	outSurface	Surface&		
←	outTransform	Optional<Matrix&>	NIL	Matrix of the transform used to compute the output surface; signed distance usage is assumed

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outTransform**.

Read more about Optional Outputs.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite plane on input in ProjectSurfaceOntoPlane.

ReduceSurface

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Reduces surface dimensions as much as possible.

Syntax

```
void avl::ReduceSurface
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 avl::Surface& outSurface
)
```

Parameters

	Name	Type	Default	Description
→	inSurface	const Surface&		Input surface
→	inRoi	Optional<const Region&>	NIL	Region of interest
←	outSurface	Surface&		Reduced surface

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region of interest exceeds an input surface in ReduceSurface.

RescaleSurface

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Changes the distances of surface points to a reference point.

Syntax

```
void avl::RescaleSurface
(
 const avl::Surface& inSurface,
 const avl::Point3D& inReferencePoint,
 float inScaleX,
 atl::Optional<float> inScaleY,
 atl::Optional<float> inScaleZ,
 bool inInverse,
 avl::Surface& outSurface
)
```

Parameters

	Name	Type	Default	Description
→	inSurface	const Surface&		Input surface
→	inReferencePoint	const Point3D&		Point to which the distances will be changed
→	inScaleX	float	1.0f	Scaling factor along X axis
→	inScaleY	Optional<float>	NIL	Scaling factor along Y axis
→	inScaleZ	Optional<float>	NIL	Scaling factor along Z axis
→	inInverse	bool		Switches to the inverse operation
←	outSurface	Surface&		Rescaled surface

Errors

List of possible exceptions:

Error type	Description
DomainError	Scale cannot be zero in an inverse rescaling in RescaleSurface.

SplitSurfaceByPlane

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Separates the surface points being on one side of the input plane from the others.

Syntax

```
void avl::SplitSurfaceByPlane
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 const avl::Plane3D& inPlane,
 bool inPreserveDimensions,
 avl::Surface& outSurface1,
 avl::Surface& outSurface2
)
```

Parameters

	Name	Type	Default	Description
→	inSurface	const Surface&		Input surface
→	inRoi	Optional<const Region&>	NIL	Region of interest
→	inPlane	const Plane3D&		Plane used for splitting
→	inPreserveDimensions	bool	False	Flag indicating whether the surface dimensions should be preserved or not
←	outSurface1	Surface&		Surface with points with positive signed distance to the input plane
←	outSurface2	Surface&		Surface with points with negative signed distance to the input plane

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite plane on input in SplitSurfaceByPlane.
DomainError	Region of interest exceeds an input surface in SplitSurfaceByPlane.

SubtractSurfaces

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Subtracts two surfaces point by point.

Syntax

```
void avl::SubtractSurfaces
(
 const avl::Surface& inSurface1,
 const avl::Surface& inSurface2,
 atl::Optional<const avl::Region&> inRoi,
 avl::Surface& outSurface
)
```

Parameters

	Name	Type	Default	Description
→	inSurface1	const Surface&		First input surface
→	inSurface2	const Surface&		Second input surface
→	inRoi	Optional<const Region&>	NIL	Region of interest
←	outSurface	Surface&		Output surface

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Different surface dimensions in SubtractSurfaces.
DomainError	Different surface scales or offsets in SubtractSurfaces.
DomainError	Different surface types in SubtractSurfaces.
DomainError	Region of interest exceeds an input surface in SubtractSurfaces.

TranslateSurface

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Translates a surface by a vector.

Syntax

```
void avl::TranslateSurface
(
 const avl::Surface& inSurface,
 const avl::Vector3D& inDelta,
 bool inInverse,
 avl::Surface& outSurface
)
```

Parameters

	Name	Type	Default	Description
→	inSurface	const Surface&		Input surface
→	inDelta	const Vector3D&		Translation vector
→	inInverse	bool		Switches to the inverse operation
←	outSurface	Surface&		Translated surface

8. Image Point Transforms

Table of content:

AbsoluteValueImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Transforms pixel values to their absolute values pixel by pixel.

Syntax

```
void avl::AbsoluteValueImage
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
←	outImage	Image&		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

Read more about In-place Computation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in AbsoluteValueImage.

AddNoiseToImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Adds random noise to the image.

Syntax

```
void avl::AddNoiseToImage
(
 RandomState& ioState,
 const avl::Image& inImage,
 const float inMinValue,
 const float inMaxValue,
 const float inNoiseStrength,
 const bool inColorNoise,
 atl::Optional<int> inSeed,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	RandomState&			Object used to maintain state of the function.
→	inImage	const Image&			Input image
→	inMinValue	const float		0.0f	Minimum value of noise pixel, inclusive
→	inMaxValue	const float		256.0f	Maximum value of noise pixel, exclusive
→	inNoiseStrength	const float	0.0 - 1.0	0.5f	Noise strength
→	inColorNoise	const bool			If 'True', noise will be generated separately for each channel
→	inSeed	Optional<int>		NIL	Random seed used to generate noise
←	outImage	Image&			Output image

AddToImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Adds a scalar value to each pixel.

Syntax

```
void avl::AddToImage
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 float inValue,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inValue	float	50.0f	Value to be added
←	outImage	Image&		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

Read more about In-place Computation.

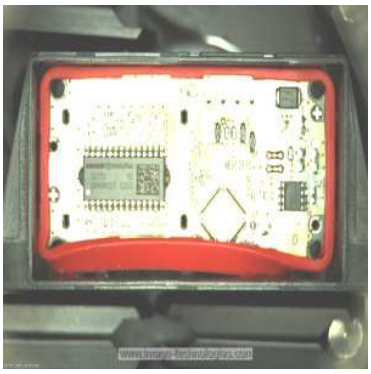
Description

The operation increases the brightness of the **inImage** by adding a fixed value to each of its pixels.

$$\forall_{i,j} \mathrm{OutImage}_{i,j} = \mathrm{InImage}_{i,j} + \mathrm{inValue}$$

Whenever the resulting value exceeds the range of pixel values, it is clipped to the nearest proper value.

Examples



*The **AddToImage** performed on the sample image with **inValue** = 50.0.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT8, SINT16, REAL.

This operation is optimized for NEON technology for pixels of types: UINT8, SINT8, UINT16, SINT16, SINT32, REAL.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in AddToImage.

See Also

- SubtractFromImage – Subtracts a scalar value from each pixel.

ClipPixels

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Sets pixels below the low value to the low value and above the high value to the high value.

Syntax

```
void avl::ClipPixels
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<float> inLowValue,
 atl::Optional<float> inHighValue,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inLowValue	Optional<float>	64.0f	The lower threshold for the image pixel values
→	inHighValue	Optional<float>	192.0f	The higher threshold for the image pixel values
←	outImage	Image&		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

Read more about In-place Computation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in ClipPixels.
DomainError	The low value cannot be higher than the high value in ClipPixels.

ColorizeImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Shows a monochromatic image in false colors.

Syntax

```
void avl::ColorizeImage
(
 ColorizeImageState& ioState,
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::ColorPalette::Type inPalette,
 bool inNegate,
 atl::Optional<int> inMinValue,
 atl::Optional<int> inMaxValue,
 avl::Image& outImage,
 avl::Image& diagPalette
)
```

Parameters

	Name	Type	Default	Description
	ioState	ColorizeImageState&		Object used to maintain state of the function.
	inImage	const Image&		Input image
	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
	inPalette	ColorPalette::Type	Iron	Palette of colors which is used to replace pixels
	inNegate	bool		Reversing palette colors
	inMinValue	Optional<int>	0	Minimal value of pixel that will be replaced by color from palette, otherwise first color from palette will be set
	inMaxValue	Optional<int>	255	Maximal value of pixel that will be replaced by color from palette, otherwise last color from palette will be set
	outImage	Image&		Output image
	diagPalette	Image&		Used palette preview

Requirements

For input **inImage** only pixel formats are supported: 1□uint8, 1□uint16.

Read more about pixel formats in Image documentation.

Description

This filter is especially useful for visualization purposes. **Hue** and **RedBlue** pallets are useful for scientific visualization and the **Iron** palette is prepared for showing infrared images in thermo-graphic appliances.

Parameters **inMinValue** and **inMaxValue** defines the scale that should be applied to image pixels. Pixels outside this range are reduced to minimal or maximal value in range.

If parameters **inMinValue** or **inMaxValue** is set to **Auto** the filter will find the maximal or minimal pixel value.

This filter works only with single channel images.

Hints

- Consider resizing image to smaller format if image does not require to contain details.
- This filter can be used to show differences between object and background noises.

Examples

*The ColorizeImage performed on the result of TestImage with **inPalette** set to **Iron**.*

Remarks

Filter pre-computes after changing its parameter which can affect the filter's execution speed. Try to avoid changing filter's parameter during program execution.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	inMinValue is greater than inMaxValue in ColorizeImage.
DomainError	Input image must have 1xUInt8 or 1xUInt16 format in ColorizeImage.
DomainError	Region exceeds an input image in ColorizeImage.
DomainError	Not supported inImage pixel format in ColorizeImage. Supported formats: 1xUInt8, 1xUInt16.

See Also

- LUTTransformImage – Changes pixel values for data stored in array.

CorrectGamma

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Performs gamma correction.

Applications: Image enhancement for human perception. For computer vision consider LogarithmImage.

Syntax

```
void avl::CorrectGamma
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 float inValue,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inValue	float	0.01 - 8.0	2.0f	Gamma coefficient, where 1.0 is neutral
←	outImage	Image&			Output image

Requirements

For input **inImage** only pixel formats are supported: int8, uint8, int16, uint16, int32.

Read more about pixel formats in Image documentation.

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

Read more about In-place Computation.

Description

The gamma correction is a standard nonlinear transformation of pixel brightness that was developed as a tool for the compensation of CRT display input-output characteristic. The operation scales brightness of each **inImage** pixel to the 0.0 - 1.0 range, exponentiates it to the power of **inValue**, and then scales the result back to the pixel values range.

Examples

*The **CorrectGamma** performed on the sample image with **inValue** = 2.0 (left image) and **inValue** = 0.5 (right image).*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported inImage pixel format in CorrectGamma. Supported formats: Int8, UInt8, Int16, UInt16, Int32.

See Also

- RescalePixels – Applies linear transformation to pixel values.

DivideImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Divides each pixel by a scalar value.

Syntax

```
void avl::DivideImage
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 float inValue,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inValue	float	2.0f	Divisor
←	outImage	Image&		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

Read more about In-place Computation.

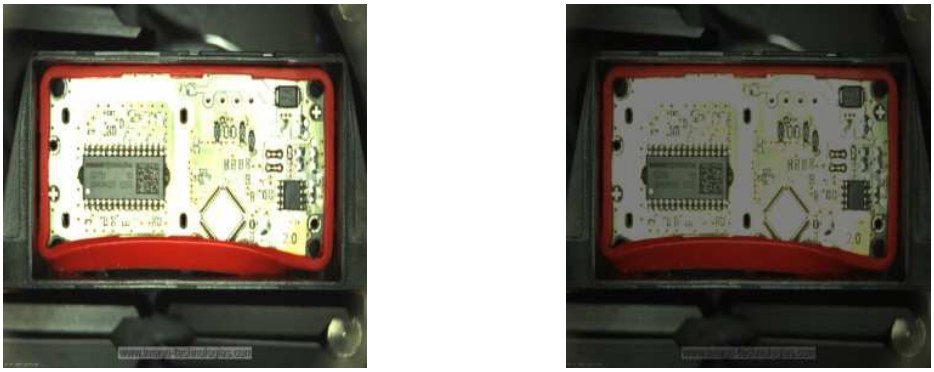
Description

The operation decreases the brightness of the **inImage** dividing each of its pixels by a fixed value.

$$\forall_{i,j} \mathbf{OutImage}_{i,j} = \frac{\mathbf{InImage}_{i,j}}{\mathbf{inValue}}$$

Whenever the resulting value exceeds the range of pixel values, it is clipped to the nearest proper value.

Examples



DivideImage *performed on the sample image with* ***inValue*** *= 2.0.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT8, SINT16, REAL.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Division by zero in DivideImage.
DomainError	Region exceeds an input image in DivideImage.

See Also

- MultiplyImage – Multiplies each pixel by a scalar value.

InvertImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Applies numeric inversion (1/x) of the pixel values.

Syntax

```
void avl::InvertImage
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 float inDividend,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inDividend	float		
←	outImage	Image&		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

Read more about In-place Computation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in InvertImage.

LogarithmImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes a natural logarithm of each pixel.

Applications: Transforms an image in such a way that a quotient on the input image becomes a difference on the output image. This can be useful for dealing with variable illumination.

Syntax

```
void avl::LogarithmImage
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<const float&> inScale,
 float inOffset,
 bool inNormalizeZero,
 avl::Image& outImage,
 avl::Profile& diagLutProfile
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Region of interest
→	inScale	Optional<const float&>		NIL	Scaling factor (default: 255)
→	inOffset	float	1.0 - ∞	1.0f	Offset factor
→	inNormalizeZero	bool			Specifies whether the output range should be rescaled to start from 0
←	outImage	Image&			Output image
🔍	diagLutProfile	Profile&			Profile depicting the resulting look-up table of the logarithm transform

Requirements

For input **inImage** only pixel formats are supported: int8, uint8, int16, uint16, int32.

Read more about pixel formats in Image documentation.

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

Read more about In-place Computation.

Description

The operation applies logarithmic operator to each pixel of an image. Logarithmic operator is defined as follows:

$$inScale \cdot \frac{log(inOffset + |P(x,y)|)}{log(inOffset + M)}$$

where:

- **M** is the maximum of the **inImage** type (i.e. 255 for UInt8, 127 for Int8).
- **inScale** is the expected maximum value of the transformation. If set to *Auto* it will result in value 127 for Int8 image and 255 for other image types.
- **inOffset** value corresponds to the camera's black level. Its default value is equal 1 and causes the strongest possible transform.

When **inNormalizeZero** is set to **True**, the result is not only scaled, but also normalized so that pixel value 0 is still transformed into value 0. This assures that the entire output value range is utilized.

Examples

*The **LogarithmImage** performed on the sample image with parameters **inScale** = 250, **inOffset** = 50. The middle image **inNormalizeZero** = **False** and the left image has **inNormalizeZero** = **True**.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported inImage pixel format in LogarithmImage. Supported formats: Int8, UInt8, Int16, UInt16, Int32.

LUTTransformImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Changes pixel values for data stored in array.

Syntax

```
void avl::LUTTransformImage
(
 LUTTransformImageState& ioState,
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 const atl::Array<int>& inColorPoints,
 const atl::Array<avl::Pixel>& inColors,
 const bool inComputeGradient,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
	ioState	LUTTransformImageState&		Object used to maintain state of the function.
	inImage	const Image&		Input image
	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
	inColorPoints	const Array<int>&		Last pixel value in which corresponding color will be replaced.
	inColors	const Array<Pixel>&		Colors corresponding to pixel levels defined in inColorPoints.
	inComputeGradient	const bool		Compute the linear change of pixels colors between next palette colors.
	outImage	Image&		Output image

Requirements

For input **inImage** only pixel formats are supported: 1□uint8, 1□uint16, 1□int8, 1□int16.

Read more about pixel formats in Image documentation.

Description

Filter applies pixel color transformation using the defined transformation colors (look-up table).

Input **inColorPoints** defines the maximal pixel values that should be replaced with colors defined in **inColors**. First color replace all pixels from 0 to **inColorPoints**[0], second color will replace all colors from **inColors**[1]+1 to **inColors**[2] and so on. All pixel values higher than last **inColorPoints** will be replaced with black pixel.

If the input **inComputeGradient** is set to **True** then the pixel value is computed as a linear interpolation between current and previous color defined in **inColors**. For the first color in **inColors** the previous color is set to minimal value of current pixel format.

Filter converts a single channel image to three RGB channels image.

This filter works only with single channel images.

Hints

- Consider resizing image to smaller format if image does not require to contain details.
- This filter can be used to show differences between object and background noises.

Examples

*The **LUTTransformImage** applied to a result of TestImage. In the middle image the **inComputeGradient** is set to **False** in the left image the **inComputeGradient** is set to **True**.*

Remarks

Filter pre-computes after changing its parameter which can affect the filter's execution speed. Try to avoid changing filter's parameter during program execution.

If no color values are provided in **inColorPoints** filter returns an empty image.

If a single color value is provided in **inColorPoints** filter returns an image filled with this color.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	inColorPoints and InColors has different sizes LUTTransformImage.
DomainError	inColorPoints must be sorted in raising order in LUTTransformImage.
DomainError	Input image must have 1xUInt8, 1xUInt16, 1xInt8 or 1xInt16 format in LUTTransformImage.
DomainError	Region exceeds an input image in LUTTransformImage.
DomainError	Not supported inImage pixel format in LUTTransformImage. Supported formats: 1xUInt8, 1xUInt16, 1xInt8, 1xInt16.

See Also

- ColorizeImage – Shows a monochromatic image in false colors.

MultiplyImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Multiplies each pixel by a scalar value.

Syntax

```
void avl::MultiplyImage
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 float inValue,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inValue	float	2.0f	Multiplier
←	outImage	Image&		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

Read more about In-place Computation.

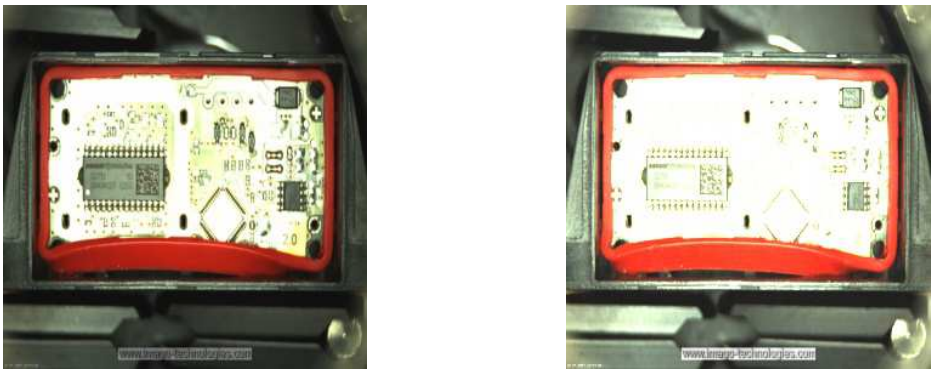
Description

The operation increases the brightness of an image by multiplying each of its pixels by a fixed value.

$$\forall_{i,j} \mathrm{OutImage}_{i,j} = \mathrm{InImage}_{i,j} \cdot \mathrm{inValue}$$

Whenever the resulting value exceeds the range of pixel values, it is clipped to the nearest proper value.

Examples



*The **MultiplyImage** performed on the sample image with **inValue** = 2.0.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT8, SINT16, REAL.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in MultiplyImage.

See Also

- DivideImage – Divides each pixel by a scalar value.

NegateImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Reverts the pixel value range (unsigned) or applies numeric negation (signed).

Syntax

```
void avl::NegateImage
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
➡	inImage	const Image&		Input image
➡	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
⬅	outImage	Image&		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

Read more about In-place Computation.

Description

The operation negates the pixel values of the **inImage**. Depending on the pixel type, the negation is defines as follows:

- For the signed pixel types: $\forall_{i,j} \mathbf{OutImage}_{i,j} = -\mathbf{InImage}_{i,j}$
- For the unsigned pixel types: $\forall_{i,j} \mathbf{OutImage}_{i,j} = \mathbf{MaximalValue} - \mathbf{InImage}_{i,j}$ (which can be thought of as *mirroring* the pixel values around the center of the pixel values range)

Examples

*The **NegateImage** performed on the sample image.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT8, SINT16, REAL NEON: UINT8, SINT8, UINT16, SINT16, SINT32, REAL.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in NegateImage.

PowerImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Exponentiates each pixel to the given power.

Syntax

```
void avl::PowerImage
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 float inValue,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inValue	float	2.0f	The exponent
←	outImage	Image&		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

Read more about In-place Computation.

Remarks

When diagnostic mode is used, this filter will check correctness of input data and throw an exception if possibility of NaN value in output image occur. When working in optimized mode, this check is omitted.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- CorrectGamma – Performs gamma correction.
- LogarithmImage – Computes a natural logarithm of each pixel.

ReplacePixels

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Substitutes pixels having the specified value with a new value.

Syntax

```
void avl::ReplacePixels
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 const avl::Pixel& inOldPixel,
 const avl::Pixel& inNewPixel,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inOldPixel	const Pixel&		
→	inNewPixel	const Pixel&		
←	outImage	Image&		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

Read more about In-place Computation.

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: 1xUINT8, 1xSINT16.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in ReplacePixels.

ResaturateImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Sets pixels below the low value to minimum, above the high value to maximum, and interpolates the rest.

Syntax

```
void avl::ResaturateImage
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 float inLowValue,
 float inHighValue,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inLowValue	float	64.0f	The value that will be changed to minimum (usually 0)
→	inHighValue	float	192.0f	The value that will be changed to maximum (usually 255)
←	outImage	Image&		Output image

Requirements

For input **inImage** only pixel formats are supported: int8, uint8, int16, uint16, int32.

Read more about pixel formats in Image documentation.

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

Read more about In-place Computation.

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT16.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in ResaturateImage.
DomainError	Not supported inImage pixel format in ResaturateImage. Supported formats: Int8, UInt8, Int16, UInt16, Int32.

See Also

- NormalizeImage – Rescales an image linearly, so that its minimum becomes inNewMinimum and the maximum of the remaining pixels becomes inNewMaximum.
- AddToImage – Adds a scalar value to each pixel.
- RescalePixels – Applies linear transformation to pixel values.

RescalePixels

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Applies linear transformation to pixel values.

Syntax

```
void avl::RescalePixels
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 float inA,
 float inB,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inA	float	1.0f	Value multiplied
→	inB	float	0.0f	Value added
←	outImage	Image&		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

Read more about In-place Computation.

Description

The operation applies linear transformation to pixel values.

$$\forall_{i,j} outImage_{i,j} = inA \cdot inImage_{i,j} + inB$$

Whenever the resulting value exceeds the range of pixel values, it is clipped to the nearest proper value.

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT8, SINT16, REAL.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in RescalePixels.

See Also

- ResaturateImage – Sets pixels below the low value to minimum, above the high value to maximum, and interpolates the rest.
- AddToImage – Adds a scalar value to each pixel.
- NormalizeImage – Rescales an image linearly, so that its minimum becomes inNewMinimum and the maximum of the remaining pixels becomes inNewMaximum.

RescalePixels_FixedRange

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Transforms pixels value from given range into new one.

Syntax

```
void avl::RescalePixels_FixedRange
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 float inMinValue,
 float inMaxValue,
 float inNewMinValue,
 float inNewMaxValue,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inMinValue	float		Declared minimum value of input image
→	inMaxValue	float		Declared maximum value of input image
→	inNewMinValue	float		Desired minimum value of the resulting image
→	inNewMaxValue	float		Desired maximum value of the resulting image
←	outImage	Image&		Output image

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT8, SINT16, REAL.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	A inMinValue cannot be equal to inMaxValue in RescalePixels_FixedRange.
DomainError	A inMinValue cannot be higher than the inMaxValue in RescalePixels_FixedRange.
DomainError	The inNewMinValue cannot be higher than the inNewMaxValue in RescalePixels_FixedRange.

SquareImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Raises pixel values to the second power pixel by pixel.

Syntax

```
void avl::SquareImage
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
←	outImage	Image&		Output image

In-place Processing

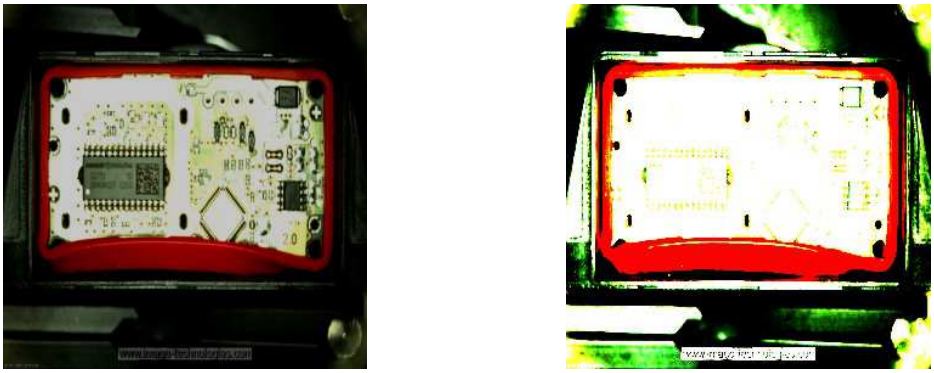
This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

Read more about In-place Computation.

Description

The operation increases the brightness of an image by squaring each pixel value. Whenever the resulting value exceeds the range of pixel values, it is clipped to the nearest proper value. Note that the absolute change of pixel brightness is usually very big.

Examples



*The **SquareImage** performed on the sample image.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT8, SINT16, REAL.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in SquareImage.

See Also

- SquareRootImage – Transforms pixel values to their square roots pixel by pixel.

SquareRootImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Transforms pixel values to their square roots pixel by pixel.

Syntax

```
void avl::SquareRootImage
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
←	outImage	Image&		Output image

Requirements

For input **inImage** only pixel formats are supported: uint8, uint16.

Read more about pixel formats in Image documentation.

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

Read more about In-place Computation.

Description

The operation decreases the brightness of an image by transforming each pixel value to its square root.

Examples

*The **SquareRootImage** performed on the sample image.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of type: UINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not an unsigned-valued image in SquareRootImage.
DomainError	Region exceeds an input image in SquareRootImage.
DomainError	Not supported inImage pixel format in SquareRootImage. Supported formats: UInt8, UInt16.

See Also

- SquareImage – Raises pixel values to the second power pixel by pixel.

SubtractFromImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Subtracts a scalar value from each pixel.

Syntax

```
void avl::SubtractFromImage
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 float inValue,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inValue	float	50.0f	Value to be subtracted
←	outImage	Image&		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

Read more about In-place Computation.

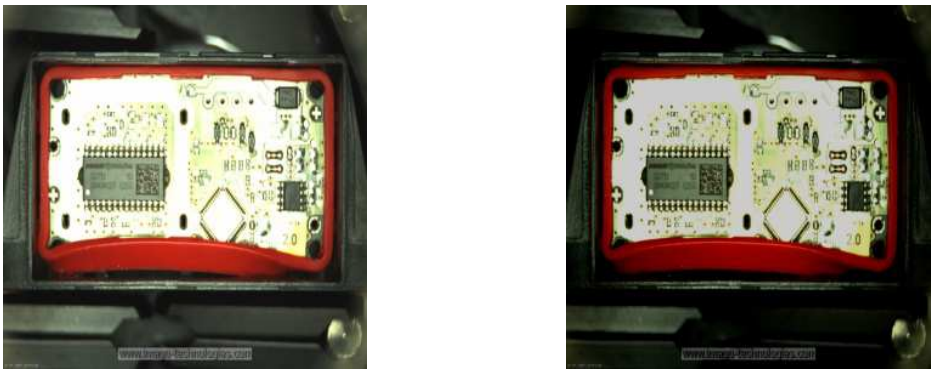
Description

The operation decreases the brightness of the **inImage** by subtracting a fixed value from each of its pixels.

$$\forall_{i,j} \mathrm{OutImage}_{i,j} = \mathrm{InImage}_{i,j} - \mathrm{inValue}$$

Whenever the resulting value exceeds the range of pixel values, it is clipped to the nearest proper value.

Examples



*The **SubtractFromImage** performed on the sample image with **inValue** = 50.0.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT8, SINT16, REAL.

This operation is optimized for NEON technology for pixels of types: UINT8, SINT8, UINT16, SINT16, SINT32, REAL.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in SubtractFromImage.

See Also

- AddToImage – Adds a scalar value to each pixel.

9. Optical Character Recognition

Table of content:

ExtractText

Header: AVL.h
Namespace: avl
Module: OCR

Ready-to-use tool for extracting and splitting character to single characters.

Syntax

```
void avl::ExtractText
(
 const avl::Image& inImage,
 const avl::Rectangle2D& inRoi,
 const avl::CoordinateSystem2D& inRoiAlignment,
 const avl::TextSegmentation& inSegmentationModel,
 atl::Array<avl::Region>& outCharacters,
 avl::Region& diagTextRegion,
 atl::Array<avl::Region>& diagAlignedCharacters,
 avl::Rectangle2D& diagAlignedRoi
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		An input image with text
→	inRoi	const Rectangle2D&		Location of the text
→	inRoiAlignment	const CoordinateSystem2D&		Adjusts the region of interest to the position of the inspected object
→	inSegmentationModel	const TextSegmentation&		Model used for separating text from background
←	outCharacters	Array<Region>&		Split characters aligned to ROI
🔍	diagTextRegion	Region&		Region of text after extraction
🔍	diagAlignedCharacters	Array<Region>&		Character regions preserving original image orientation
🔍	diagAlignedRoi	Rectangle2D&		ROI rectangle after alignment

Description

This filter distinguish characters from the background using a set of algorithms. This filter performs two steps: text extraction and text segmentation:

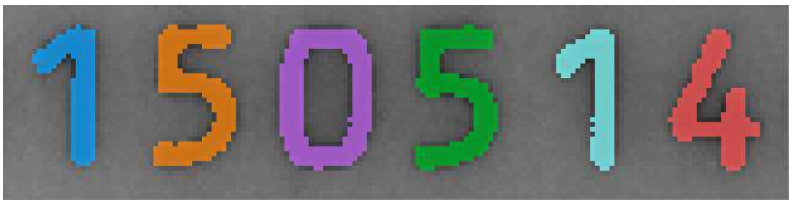
- **Text extraction** – in this step filter uses some basic thresholding methods to get a characters region from the image.
- **Text segmentation** – split previously found region into separate regions contains character.

Input **inRoi** should contains only a single line of text.

Hints

- If the object location is variable, pass an appropriate local coordinate system to **inRoiAlignment**.
- Before defining **inSegmentationModel** first specify **inRoi** – the rectangle in which characters will be extracted.

Examples



Result of reading text using the ExtractText.

Remarks

To read more about how to use OCR technique, refer to Machine Vision Guide: Optical Character Recognition

Errors

List of possible exceptions:

Error type	Description
DomainError	Invalid segmentation algorithm in ExtractText.
DomainError	Invalid thresholding algorithm in ExtractText.

See Also

- ReadText – Ready-to-use tool for reading text from images using the OCR technique.
- RecognizeCharacters – Classifies input regions into characters. Based on the Multi-Layer Perceptron model.

ExtractText2

Header: AVL.h
Namespace: avl
Module: OCR

Ready-to-use tool for extracting and splitting text elements to single characters.

Syntax

```
void avl::ExtractText2
(
 const avl::Image& inImage,
 const avl::Rectangle2D& inRoi,
 const avl::CoordinateSystem2D& inRoiAlignment,
 const avl::Polarity::Type inPolarity,
 const int inCharWidth,
 atl::Optional<int> inStrokeWidth,
 atl::Optional<int> inMinWordGap,
 atl::Array<avl::Region>& outCharacters,
 atl::Array<avl::Region>& diagAlignedCharacters
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			An input image with text
→	inRoi	const Rectangle2D&			Location of the text
→	inRoiAlignment	const CoordinateSystem2D&			Adjusts the region of interest to the position of the inspected object
→	inPolarity	const Polarity::Type			Text polarity
→	inCharWidth	const int	5 - 200	50	Width of a single character in pixels
→	inStrokeWidth	Optional<int>	1 - 50	NIL	Width of the stroke of the letters in pixels
→	inMinWordGap	Optional<int>	1 - 200	NIL	Width of the smallest gap between letters that is to be treated as a space in pixels
←	outCharacters	Array<Region>&			Regions representing individual characters aligned to the ROI
🔍	diagAlignedCharacters	Array<Region>&			Regions representing individual characters aligned to the Image

Errors

List of possible exceptions:

Error type	Description
DomainError	inPolarity must be specified as Dark or Bright in ExtractText2.
DomainError	inRoi is to narrow to fit a single character in ExtractText2.
DomainError	It is impossible for inStrokeWidth to be greater than inCharWidth in ExtractText2.

GroupRegionsByLines

Header: AVL.h
Namespace: avl
Module: OCR

Splits an array of blobs by distance to computed base lines.

Syntax

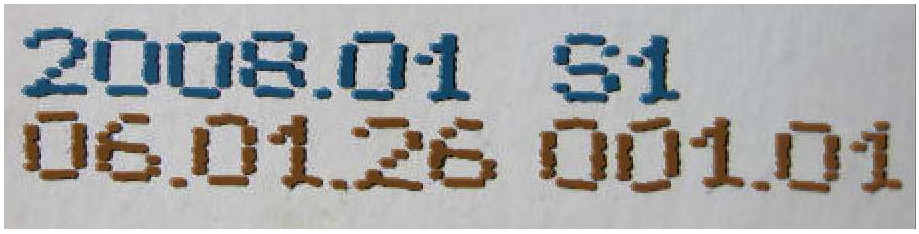
```
void avl::GroupRegionsByLines
(
 const atl::Array<avl::Region>& inRegions,
 const avl::Rectangle2D& inRoi,
 const atl::Optional<avl::CoordinateSystem2D>& inRoiAlignment,
 const int inLinesCount,
 const float inLineWidth,
 const bool inRemoveOutliers,
 atl::Array<atl::Array<avl::Region>>& outLines,
 atl::Array<avl::Region>& outLine0,
 atl::Array<avl::Region>& outLine1,
 atl::Array<avl::Region>& outLine2,
 atl::Array<avl::Region>& outLine3,
 avl::Rectangle2D& outAlignedRoi,
 atl::Array<avl::Segment2D>& diagLines,
 atl::Array<avl::Rectangle2D>& diagLineRanges,
 atl::Array<avl::Point2D>& diagRegionPoints
)
```

Parameters

	Name	Type	Range	Default	Description
→	inRegions	const Array<Region>&			Input regions (parts of characters)
→	inRoi	const Rectangle2D&			Range of character classification
→	inRoiAlignment	const Optional<CoordinateSystem2D>&		NIL	
→	inLinesCount	const int	0 - ∞	1	ROI is divided into inLinesCount lines
→	inLineWidth	const float	0.0 - ∞	1.0f	Value of line range used if inRemoveOutliers is set to TRUE
→	inRemoveOutliers	const bool		False	If this flag is set blobs with distance greater than inLineWidth are removed.
←	outLines	Array<Array<Region>>&			Lines in single Array
←	outLine0	Array<Region>&			
←	outLine1	Array<Region>&			
←	outLine2	Array<Region>&			
←	outLine3	Array<Region>&			
←	outAlignedRoi	Rectangle2D&			
🔍	diagLines	Array<Segment2D>&			Segments which indicates the position of lines
🔍	diagLineRanges	Array<Rectangle2D>&			Rectangles represents range of lines which are included if inRemoveOutliers is set
🔍	diagRegionPoints	Array<Point2D>&			Point which indicates region position

Examples

As you can see in the below picture, the filter's task is to divide the input region into several sub-regions by its placement in a line. As a result, each line represents a separate region.



Exemplary result of the filter's usage.

LoadOcrMlpModel

Header: AVL.h
Namespace: avl
Module: OCR

Loads serialized OcrMlpModel object from AVDATA file.

Syntax

```
void avl::LoadOcrMlpModel
(
 const atl::File& inFilename,
 avl::OcrMlpModel& outOcrModel
)
```

Parameters

	Name	Type	Default	Description
→	inFilename	const File&		Name of the source file
←	outOcrModel	OcrMlpModel&		Deserialized OcrMlpModel

LoadOcrModel

Header: AVL.h
Namespace: avl
Module: OCR

Loads serialized OcrMlpModel object from AVDATA file.

Syntax

```
void avl::LoadOcrModel
(
 const atl::File& inFilename,
 avl::OcrModel& outOcrModel
)
```

Parameters

	Name	Type	Default	Description
→	inFilename	const File&		Name of the source file
←	outOcrModel	OcrModel&		Deserialized OcrMlpModel

LoadOcrSvmModel

Header: AVL.h
Namespace: avl
Module: OCR

Loads serialized OcrSvmModel object from AVDATA file.

Syntax

```
void avl::LoadOcrSvmModel
(
 const atl::File& inFilename,
 avl::OcrSvmModel& outOcrModel
)
```

Parameters

	Name	Type	Default	Description
→	inFilename	const File&		Name of the source file
←	outOcrModel	OcrSvmModel&		Deserialized OcrSvmModel

OcrMlpModelToOcrModel

Header: AVL.h
Namespace: avl
Module: OCR

Converts OcrMlpModel(old type) to OcrModel.

Syntax

```
void avl::OcrMlpModelToOcrModel
(
 const avl::OcrMlpModel& inOcrMlpModel,
 avl::OcrModel& outOcrModel
)
```

Parameters

	Name	Type	Default	Description
→	inOcrMlpModel	const OcrMlpModel&		
←	outOcrModel	OcrModel&		

OcrModelToOcrMlpModel

Header: AVL.h
Namespace: avl
Module: OCR

Converts OcrModel to OcrMlpModel.

Syntax

```
void avl::OcrModelToOcrMlpModel
(
 const avl::OcrModel& inOcrModel,
 avl::OcrMlpModel& outOcrMlpModel
)
```

Parameters

	Name	Type	Default	Description
→	inOcrModel	const OcrModel&		
←	outOcrMlpModel	OcrMlpModel&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Provided OcrModel does not contain SVM model.

OcrModelToOcrSvmModel

Header: AVL.h
Namespace: avl
Module: OCR

Converts OcrModel to OcrSvmModel.

Syntax

```
void avl::OcrModelToOcrSvmModel
(
 const avl::OcrModel& inOcrModel,
 avl::OcrSvmModel& outOcrSvmModel
)
```

Parameters

	Name	Type	Default	Description
→	inOcrModel	const OcrModel&		
←	outOcrSvmModel	OcrSvmModel&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Provided OcrModel does not contain SVM model.

OcrSvmModelToOcrModel

Header: AVL.h
Namespace: avl
Module: OCR

Converts OcrSvmModel(old type) to OcrModel.

Syntax

```
void avl::OcrSvmModelToOcrModel
(
 const avl::OcrSvmModel& inOcrSvmModel,
 avl::OcrModel& outOcrModel
)
```

Parameters

	Name	Type	Default	Description
→	inOcrSvmModel	const OcrSvmModel&		
←	outOcrModel	OcrModel&		

ReadText

Header: AVL.h
Namespace: avl
Module: OCR

Ready-to-use tool for reading text from images using the OCR technique.

Syntax

```
void avl::ReadText
(
 const atl::Array<avl::Region>& inCharacters,
 const avl::OcrModel& inOcrModel,
 const float inMinScore,
 atl::String& outText,
 atl::Array<atl::Conditional<atl::String>>& outCharacters,
 atl::Array<float>& outScores,
 bool& outIsTextValid
)
```

Parameters

	Name	Type	Range	Default	Description
→	inCharacters	const Array<Region>&			Character regions
→	inOcrModel	const OcrModel&			OCR model specific to a particular font
→	inMinScore	const float	0.0 - 1.0		Minimal score of reading a character
←	outText	String&			Read text
←	outCharacters	Array<Conditional<String>>&			Array of characters. NIL indicates invalid read when inMinScore is set,
←	outScores	Array<float>&			Reading scores for each character
←	outIsTextValid	bool&			Returns False if any ad score smaller than inMinScore

Description

This operation reads a text from the array of regions. Each region corresponds to a single letter at the filter output **outText**. Empty regions are omitted.

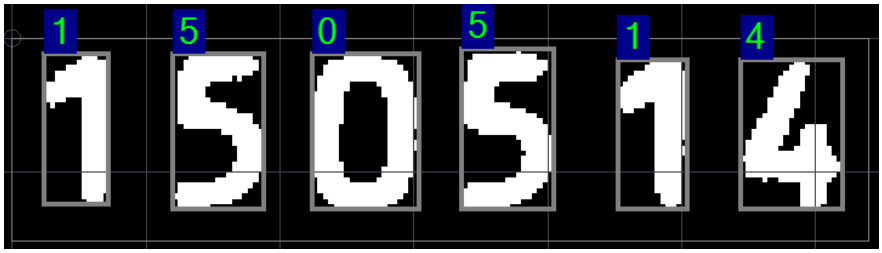
This filter uses a trained OcrModel which can be created using the TrainOcr_MLP or TrainOcr_SVM filter.

Typically this filter are connected with ExtractText which prepares input regions for reading.

Hints

- Pass an array of character regions to the **inCharacters** input. Usually it will be the output of the ExtractText filter.

Examples



Result of reading text using the ReadText and ExtractText.

Remarks

To read more about how to use OCR technique, refer to Machine Vision Guide: Optical Character Recognition

Errors

List of possible exceptions:

Error type	Description
DomainError	Uninitialized OCR model in ReadText. OCR model must be trained before use.

See Also

- ExtractText – Ready-to-use tool for extracting and splitting character to single characters.
- RecognizeCharacters – Classifies input regions into characters. Based on the Multi-Layer Perceptron model.
- TrainOcr_MLP – Trains an OCR multilayer perceptron classifier.
- TrainOcr_SVM – Trains an OCR support vector machines classifier.

RecognizeCharacters

Header: AVL.h
Namespace: avl
Module: OCR

Classifies input regions into characters. Based on the Multi-Layer Perceptron model.

Applications: Usually the last, yet the most important step of optical character recognition or verification.

Syntax

```
void avl::RecognizeCharacters
(
 const atl::Array<avl::Region>& inCharacterRegions,
 const avl::OcrModel& inOcrModel,
 atl::Optional<const avl::Size&> inCharacterSize,
 const bool inDotPrint,
 const avl::CharacterSortingOrder::Type inCharacterSorting,
 atl::Optional<float> inMinScore,
 atl::Optional<int> inMinSpaceWidth,
 atl::String& outCharacters,
 atl::Array<float>& outScores,
 atl::Array<atl::Array<avl::OcrCandidate> >& outCandidates,
 atl::Array<avl::Image>& diagNormalizedCharacters,
 atl::Array<avl::Box>& diagCharactersBoxes
)
```

Parameters

	Name	Type	Range	Default	Description
→	inCharacterRegions	const Array<Region>&			Array of character regions to recognize
→	inOcrModel	const OcrModel&			Trained OcrMlpModel used to recognize characters
→	inCharacterSize	Optional<const Size&>		NIL	Size of single monospaced character if needed
→	inDotPrint	const bool			Dot-printed characters preprocessing
→	inCharacterSorting	const CharacterSortingOrder::Type		LeftToRight	Sorting order of input characters
→	inMinScore	Optional<float>	0.0 - 1.0	NIL	Minimal value of accepted result. Otherwise char '*' will be placed.
→	inMinSpaceWidth	Optional<int>	0 - ∞	NIL	Minimal distance between characters where space character will be inserted
←	outCharacters	String&			Result of characters recognition
←	outScores	Array<float>&			Classification result score
←	outCandidates	Array<Array<OcrCandidate> >&			Array of a character classification results and their score
🔍	diagNormalizedCharacters	Array<Image>&			Images of normalized characters used in character recognition
🔍	diagCharactersBoxes	Array<Box>&			Bounding boxes of characters

Description

This operation performs reading a text from a set of regions. This operation requires a trained OCR classifier, which can be created using the TrainOcr_MLP or TrainOcr_SVM filter.

Filter requires regions specified in the **inCharacterRegions**. Each of the input region must contain single character. To get regions from an image use filter ExtractText.

The **inCharacterSorting** parameter defines the sorting order of the input characters provided in **inCharacterRegions**.

The parameter **inDotPrint** turns on the dedicated smoothing for characters printed using a jet printer in a dot-matrix form.

The parameter **inCharacterSize** defines size of monospaced (fixed-width) font. This size defines the cutting character bounding box.

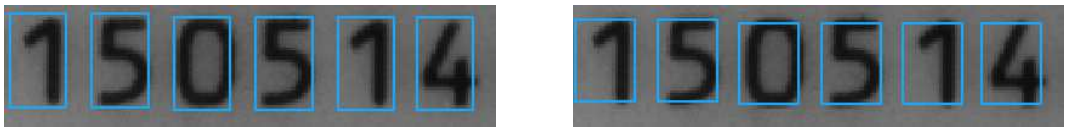
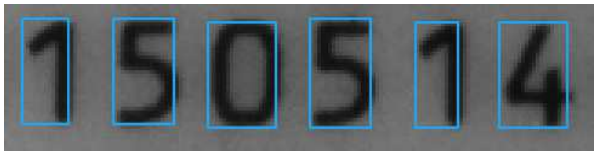


Image shows extracted character cutting boxes with different cutting size.

If the parameter is set to *auto* character will be recognized as proportional font. For further information about font types please refer to the documentation of filters TrainOcr_SVM or TrainOcr_MLP.



*Characters boxes with **inCharacterSize** set to **Auto**.*

The input **inMinSpaceWidth** value indicates a minimal distance between characters where space character will be inserted to result string. When the value is marked as **Auto** no spaces will be inserted. Distance is presented in pixels.

If **inMinScore** is set, the characters with smaller score will be replaced with *****

The output **outCharacters** contains recognized characters. The recognition score of each recognized character is stored in the **outScores**.

Remarks

To read more about how to use OCR technique, refer to Machine Vision Guide: Optical Character Recognition

Errors

List of possible exceptions:

Error type	Description
DomainError	Uninitialized OCR model in RecognizeCharacters. OCR model must be trained before use.

See Also

- TrainOcr_SVM – Trains an OCR support vector machines classifier.
- TrainOcr_MLP – Trains an OCR multilayer perceptron classifier.

SaveOcrMlpModel

Header: AVL.h
Namespace: avl
Module: OCR

Saves serialized OcrMlpModel object as AVDATA file.

Syntax

```
void avl::SaveOcrMlpModel
(
 const avl::OcrMlpModel& inOcrModel,
 const atl::File& inFilename
)
```

Parameters

	Name	Type	Default	Description
➡	inOcrModel	const OcrMlpModel&		Model to be serialized
➡	inFilename	const File&		Name of the target file

SaveOcrModel

Header: AVL.h
Namespace: avl
Module: OCR

Saves serialized OcrMlpModel object as AVDATA file.

Syntax

```
void avl::SaveOcrModel
(
 const avl::OcrModel& inOcrModel,
 const atl::File& inFilename
)
```

Parameters

	Name	Type	Default	Description
➡	inOcrModel	const OcrModel&		Model to be serialized
➡	inFilename	const File&		Name of the target file

SaveOcrSvmModel

Header: AVL.h
Namespace: avl
Module: OCR

Saves serialized OcrSvmModel object as AVDATA file.

Syntax

```
void avl::SaveOcrSvmModel
(
 const avl::OcrSvmModel& inOcrModel,
 const atl::File& inFilename
)
```

Parameters

	Name	Type	Default	Description
→	inOcrModel	const OcrSvmModel&		Model to be serialized
→	inFilename	const File&		Name of the target file

SplitRegionIntoExactlyNCharacters

Header: AVL.h
Namespace: avl
Module: OCR

Splits the input region into a fixed-size array of regions corresponding to individual characters.

Applications: Text segmentation when the number of characters is known, usually followed by a RecognizeCharacters filter.

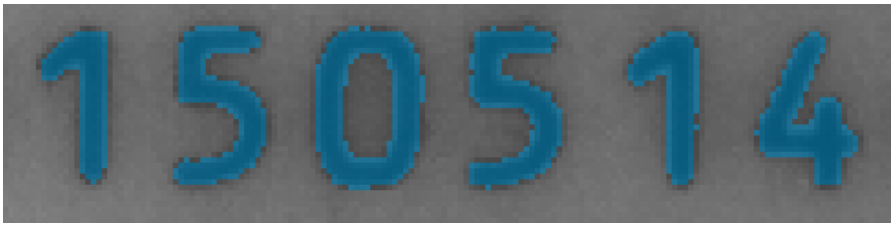
Syntax

```
void avl::SplitRegionIntoExactlyNCharacters
(
 const avl::Region& inRegion,
 const int inCharacterCount,
 const int inCharacterSpacing,
 const float inProjectionSmooth,
 atl::Conditional<atl::Array<avl::Region> >& outRegions,
 atl::Array<avl::Region>& diagClasses,
 avl::Profile& diagProjection
)
```

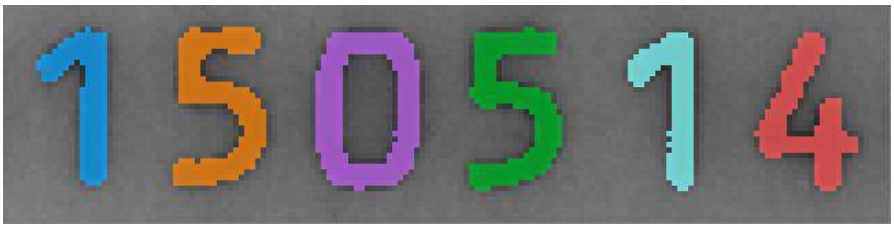
Parameters

	Name	Type	Range	Default	Description
→	inRegion	const Region&			Region which contains characters
→	inCharacterCount	const int	1 - ∞		Character count in provided region
→	inCharacterSpacing	const int	0 - ∞		Spacing size between characters
→	inProjectionSmooth	const float	0.0 - ∞	1.0f	Projection smoothing value used to remove noises from character region
←	outRegions	Conditional<Array<Region> >&			Output array of regions containing separated characters
🔍	diagClasses	Array<Region>&			Regions that contain location of split characters
🔍	diagProjection	Profile&			Profile of region projection used to distinguish characters

Examples



Input region.



Result of the filter's usage.

Errors

List of possible exceptions:

Error type	Description
DomainError	Value of inCharacterCount is greater than inRegion frame width.
DomainError	Value of inCharactersSpacing is greater than single character width.

SplitRegionIntoMultipleCharacters

Header: AVL.h
Namespace: avl
Module: OCR

Splits the input region into an array of regions corresponding to individual characters.

Applications: Text segmentation when the number of characters is unknown, usually followed by a RecognizeCharacters filter.

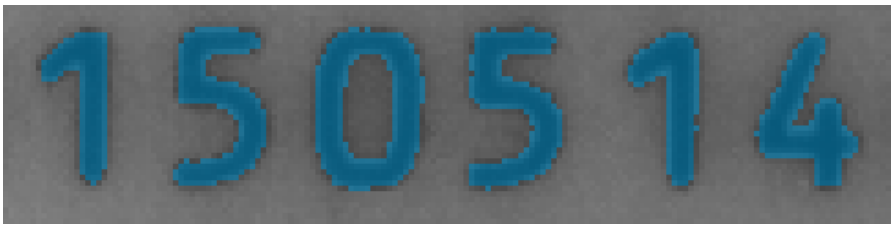
Syntax

```
void avl::SplitRegionIntoMultipleCharacters
(
 const avl::Region& inRegion,
 const float inProjectionSmooth,
 const int inCharacterWidth,
 atl::Array<avl::Region>& outRegions,
 atl::Array<avl::Region>& diagClasses,
 avl::Profile& diagProjection
)
```

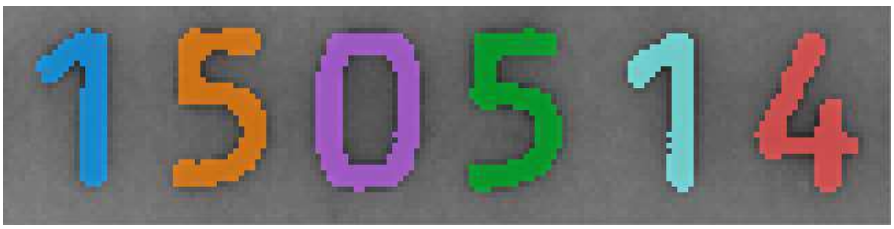
Parameters

	Name	Type	Range	Default	Description
→	inRegion	const Region&			Input region containing characters
→	inProjectionSmooth	const float	0.0 - ∞	1.0f	Value of smooth applied to region projection before splitting into separated characters
→	inCharacterWidth	const int	0 - ∞	15	Single character width
←	outRegions	Array<Region>&			Array or regions containing single characters
🔍	diagClasses	Array<Region>&			Regions in which characters parts will be connected into a single character region
🔍	diagProjection	Profile&			Profile of region projection used to distinguish characters

Examples



Input region.



Result of the filter's usage.

TrainOcr_MLP

Header: AVL.h
Namespace: avl
Module: OCR

Trains an OCR multilayer perceptron classifier.

Syntax

```
void avl::TrainOcr_MLP
(
 const atl::Array<avl::CharacterSample>& inCharacterSamples,
 const avl::Size& inNormalizationSize,
 atl::Optional<const atl::Array<int>& > inHiddenLayerSizes,
 atl::Optional<int> inRandomSeed,
 const avl::CharacterFeatures& inCharacterFeatures,
 float inLearningRate,
 float inMomentum,
 int inIterationCount,
 atl::Optional<const avl::Size&> inCharacterSize,
 avl::OcrModel& outOcrModel,
 float& outTrainingAccuracy,
 avl::Profile& diagError,
 atl::Array<avl::Image>& diagNormalizedCharacters
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inCharacterSamples	const Array<CharacterSample>&			Training font created from sample regions
➡	inNormalizationSize	const Size&		(Width: 16, Height: 16)	The character size after normalization
➡	inHiddenLayerSizes	Optional<const Array<int>& >		NIL	Internal structure of neuron layers used in classifier
➡	inRandomSeed	Optional<int>	0 - + ∞	NIL	Random seed used by MLP classifier
➡	inCharacterFeatures	const CharacterFeatures&		(Pixels: True)	Character features used to distinguish characters from each other
➡	inLearningRate	float	0.01 - 1.0	0.6f	Suppression level of changes during learning process
➡	inMomentum	float	0.0 - 1.0	0.75f	Value of classifier learning momentum
➡	inIterationCount	int	1 - + ∞	100	Learning iteration count
➡	inCharacterSize	Optional<const Size&>		NIL	Size of fixed width font
⬅	outOcrModel	OcrModel&			Trained OcrMlpModel used to recognize characters
⬅	outTrainingAccuracy	float&			The overall training score
🔍	diagError	Profile&			Changes of mean error level progress during learning process
🔍	diagNormalizedCharacters	Array<Image>&			Images of normalized characters used to train classifier

Description

This filter prepares a MLP classifier for the further OCR operations.

Filter requires a set of prepared CharacterSample which can be created using MakeCharacterSamples.

Parameter **inCharacterSize** defines the size of character cropping box. It is especially useful when characters are much bigger than normalization size. When it has **Nil** value the character is cropped to its bounding box.

Parameters **inHiddenLayerSizes**, **inRandomSeed** are used in the process of learning of a newly created MLP classifier. For further parameter description please refer to the documentation of the MLP_Init filter.

All the input regions in filters RecognizeCharacters and TrainOcr_MLP are resized to the size specified in the **inNormalizationSize** input parameter. The further classification is performed on the normalized regions. Therefore, it is important to select the appropriate normalization size.

The selection of too small normalization size may result in loss of character details. However, too large value of normalization size increases the classifier learning time. The best recognition results are obtained when the size of character is nearly the same as the normalization size.

The character classification depends on character features that are selected in the **inCharacterFeatures** parameter. At least one feature must be selected. By the default the feature **Pixels** is selected.

The table below contains the description of each available character feature:

Feature name	Description	Filter origin	Normalized
Pixels	Values of the image pixels after normalization.		False
NormalizedPixels	Values of the image pixels after normalization normalized to range <0, 1.0>.		True
Convexity	Ratio of the input region area to area of its convex hull.	RegionConvexity	True
Circularity	Ratio of the region area to area of its bounding circle.	RegionCircularity	True
NumberOfHoles	Number of holes found in the input region.	RegionHoles	True
AspectRatio	Ratio of input region width to its height.	RegionBoundingBox	False
Width	Region bounding box width.	RegionBoundingBox	False
Height	Region bounding box height.	RegionBoundingBox	False
AreaRatio	Ratio of the input region area to area of its bounding box.		True
DiameterRatio	Ratio of the input region diameter to diameter of its bounding box.	RegionDiameter	True
Elongation	Ratio of longer axis of the approximating ellipse to the shorter one.	RegionElongation	False
Orientation	Further details in the filter RegionOrientation documentation.	RegionOrientation	True
Zoning4x4	Normalized pixel values of region reduced to size 4x4 pixel.		True
HorizontalProjection	Values of normalized image projection normalized by region height.	ImageProjection	True
VerticalProjection	Values of normalized image projection normalized by region height.	ImageProjection	True
HoughCircles	Count of circles found in the normalized image.		True
Moment_11	Character geometric moment type M_{11}.	RegionMoment	False
Moment_20	Character geometric moment type M_{20}.	RegionMoment	False
Moment_02	Character geometric moment type M_{02}.	RegionMoment	False

Remarks

It is recommended not to set normalization size greater than 50 pixels in each dimension. That could make learning time too long.

For more remarks about using MLP classifier please refer to the documentation of the MLP_Init filter.

To read more about how to use OCR technique, refer to Machine Vision Guide: Optical Character Recognition

Errors

List of possible exceptions:

Error type	Description
DomainError	At least a single feature must be selected in inCharacterFeatures in TrainOcr_MLP.
DomainError	Hidden layer should have at least a single hidden layer in TrainOcr_MLP.
DomainError	Invalid character sample in TrainOcr_MLP.
DomainError	Invalid normalization size in InitOcr_MLP.

See Also

- TrainOcr_SVM – Trains an OCR support vector machines classifier.
- RecognizeCharacters – Classifies input regions into characters. Based on the Multi-Layer Perceptron model.

TrainOcr_SVM

Header: AVL.h
Namespace: avl
Module: OCR

Trains an OCR support vector machines classifier.

Syntax

```
void avl::TrainOcr_SVM
(
 const atl::Array<avl::CharacterSample>& inCharacterSamples,
 const avl::Size& inNormalizationSize,
 const atl::Optional<float>& inNu,
 const atl::Optional<float>& inKernelGamma,
 const float inRegularizationConstant,
 const float inStopEpsilon,
 bool inUseShrinkingHeuristics,
 atl::Optional<const avl::Size&> inCharacterSize,
 atl::Optional<int> inRandomSeed,
 const avl::CharacterFeatures& inCharacterFeatures,
 avl::OcrModel& outOcrModel,
 float& outTrainingAccuracy,
 atl::Array<avl::Image>& diagNormalizedCharacters
)
```

Parameters

	Name	Type	Range	Default	Description
→	inCharacterSamples	const Array<CharacterSample>&			Training font created from sample regions
→	inNormalizationSize	const Size&		(Width: 16, Height: 16)	The character size after normalization
→	inNu	const Optional<float>&	0.0 - 1.0	NIL	Trade-off between training accuracy and number of supported vectors
→	inKernelGamma	const Optional<float>&		NIL	Gamma parameter for RBF kernel
→	inRegularizationConstant	const float	0.0 - ∞	1.0f	Preventing overfitting
→	inStopEpsilon	const float		0.001f	Epsilon for stopping criterion
→	inUseShrinkingHeuristics	bool		True	Heuristics may speed up computations
→	inCharacterSize	Optional<const Size&>		NIL	Size of fixed width font
→	inRandomSeed	Optional<int>	0 - + ∞	NIL	Random seed used to train classifier
→	inCharacterFeatures	const CharacterFeatures&		(Pixels: True)	Character features used to identify characters
←	outOcrModel	OcrModel&			Trained OcrSvmModel used to recognize characters
←	outTrainingAccuracy	float&			The overall training score
🔍	diagNormalizedCharacters	Array<Image>&			Images of normalized characters used to train classifier

Description

This filter prepares a SVM classifier for the further OCR operations.

Filter requires a set of prepared CharacterSample which can be created using MakeCharacterSamples.

Parameter **inCharacterSize** defines the size of character cropping box. It is especially useful when characters are much bigger than normalization size. When it has **Nil** value the character is cropped to its bounding box.

The selection of too small normalization size may result in loss of character details. However, too large value of normalization size increases the classifier learning time. The best recognition results are obtained when the size of character is nearly the same as the normalization size.

The character classification depends on character features that are selected in the **inCharacterFeatures** parameter. At least one feature must be selected. By the default the feature **Pixels** is selected.

The table below contains the description of each available character feature:

Feature name	Description	Filter origin	Normalized
Pixels	Values of the image pixels after normalization.		False
NormalizedPixels	Values of the image pixels after normalization normalized to range <0, 1.0>.		True
Convexity	Ratio of the input region area to area of its convex hull.	RegionConvexity	True
Circularity	Ratio of the region area to area of its bounding circle.	RegionCircularity	True
NumberOfHoles	Number of holes found in the input region.	RegionHoles	True
AspectRatio	Ratio of input region width to its height.	RegionBoundingBox	False
Width	Region bounding box width.	RegionBoundingBox	False
Height	Region bounding box height.	RegionBoundingBox	False
AreaRatio	Ratio of the input region area to area of its bounding box.		True
DiameterRatio	Ratio of the input region diameter to diameter of its bounding box.	RegionDiameter	True
Elongation	Ratio of longer axis of the approximating ellipse to the shorter one.	RegionElongation	False
Orientation	Further details in the filter RegionOrientation documentation.	RegionOrientation	True
Zoning4x4	Normalized pixel values of region reduced to size 4x4 pixel.		True
HorizontalProjection	Values of normalized image projection normalized by region height.	ImageProjection	True
VerticalProjection	Values of normalized image projection normalized by region height.	ImageProjection	True
HoughCircles	Count of circles found in the normalized image.		True
Moment_11	Character geometric moment type M_{11}.	RegionMoment	False
Moment_20	Character geometric moment type M_{20}.	RegionMoment	False
Moment_02	Character geometric moment type M_{02}.	RegionMoment	False

Remarks

To read more about how to use OCR technique, refer to Machine Vision Guide: Optical Character Recognition

Errors

List of possible exceptions:

Error type	Description
DomainError	At least a single feature must be selected in inCharacterFeatures in TrainOcr_SVM.
DomainError	Invalid character sample in TrainOcr_SVM.
DomainError	Invalid OcrSvmModel in TrainOcr_SVM.

See Also

- TrainOcr_SVM – Trains an OCR support vector machines classifier.
- RecognizeCharacters – Classifies input regions into characters. Based on the Multi-Layer Perceptron model.

10. Template Matching

Table of content:

ControlEdgeModelLimits

Header: AVL.h
Namespace: avl
Module: MatchingPro

Limits the size of the EdgeModel objects created by CreateEdgeModel filter.

Syntax

```
void avl::ControlEdgeModelLimits
(
 atl::Optional<int> inMemoryLimit
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inMemoryLimit	Optional<int>	1 - ∞	NIL	Maximum number of megabytes an EdgeModel can have

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

CreateEdgeModel1

Header: AVL.h
Namespace: avl
Module: MatchingPro

Creates a model for edge-based template matching.

Applications: Dynamic creation of models in the runtime environment (normally they are created interactively in Studio).

Syntax

```
void avl::CreateEdgeModel1
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inTemplateRegion,
 atl::Optional<const avl::Rectangle2D&> inReferenceFrame,
 int inMinPyramidLevel,
 atl::Optional<int> inMaxPyramidLevel,
 float inSmoothingStdDev,
 float inEdgeThreshold,
 float inEdgeHysteresis,
 float inMinAngle,
 float inMaxAngle,
 float inAnglePrecision,
 float inMinScale,
 float inMaxScale,
 float inScalePrecision,
 float inEdgeCompleteness,
 atl::Conditional<avl::EdgeModel>& outEdgeModel,
 atl::Optional<atl::Conditional<avl::Point2D>&> outEdgeModelPoint = atl::NIL,
 atl::Optional<atl::Conditional<atl::Array<avl::Path>>&> outEdges = atl::NIL,
 atl::Conditional<atl::Array<avl::Image> >& diagEdgePyramid
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inImage	const Image&			Image from which model will be extracted
➡	inTemplateRegion	Optional<const Region&>		NIL	Region of the image from which model will be extracted
➡	inReferenceFrame	Optional<const Rectangle2D&>		NIL	Exact position of the model object in the image
➡	inMinPyramidLevel	int	0 - 12	0	Defines the index of the lowest reduced resolution level used to speed up computations
➡	inMaxPyramidLevel	Optional<int>	0 - 12	NIL	Defines the number of reduced resolution levels used to speed up computations
➡	inSmoothingStdDev	float	0.0 - ∞	0.0f	Standard deviation of the gaussian smoothing applied before edge extraction
➡	inEdgeThreshold	float	0.0 - ∞	35.0f	Higher threshold for edge magnitude
➡	inEdgeHysteresis	float	0.0 - ∞	15.0f	Threshold hysteresis value for edge magnitude
➡	inMinAngle	float		-180.0f	Start of range of possible rotations
➡	inMaxAngle	float		180.0f	End of range of possible rotations
➡	inAnglePrecision	float	0.001 - 10.0	1.0f	Defines angular resolution of the matching process
➡	inMinScale	float	0.0 - ∞	1.0f	Start of range of possible scales
➡	inMaxScale	float	0.0 - ∞	1.0f	End of range of possible scales
➡	inScalePrecision	float	0.001 - 10.0	1.0f	Defines scale resolution of the matching process
➡	inEdgeCompleteness	float	0.01 - 1.0	1.0f	Determines what fraction of the edges will be present in the created model
⬅	outEdgeModel	Conditional<EdgeModel>&			Created model that can be used by LocateMultipleObjects_Edges
⬅	outEdgeModelPoint	Optional<Conditional<Point2D>&>		NIL	The middle point of the created model
⬅	outEdges	Optional<Conditional<Array<Path>>&>		NIL	Visualization of the model edges found at the original resolution
🔍	diagEdgePyramid	Conditional<Array<Image> >&			Visualization of the edges found at different resolution levels

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outEdgeModelPoint**, **outEdges**.

Read more about Optional Outputs.

Description

The operation creates an Edge Matching model for the object represented in **inTemplateRegion** region in **inImage** image. The resulting model can be matched against any image using the LocateMultipleObjects_Edges1 filter.

The model consists of a pyramid of iteratively downsampled images, the original image being the first of them. The **inMaxPyramidLevel** parameter determines how many additional images of the pyramid shall be computed. Its value has great influence on computation speed. Therefore it is highly recommended to set its value as high as possible, at the same time ensuring that the model edges are at least 2^**inMaxPyramidLevel** pixels far from the **inImage** image frame. However, if it is set too high and no model edges are found on some pyramid level, an error with appropriate description occurs.

The **inEdgeThreshold** and **inEdgeHysteresis** parameters control the hysteresis threshold (as in ThresholdImage_Hysteresis) used in the edge extraction (as in DetectEdges_AsRegion) phase of the model creation. It is an important part of using the filter to set these parameters properly, because only found edge pixels determine how good the later matching process will be. Therefore the **outEdges** and **diagEdgePyramid** are the crucial parameters for experiments, because they show found edges in the model image and edge pixels found at different resolution levels, respectively.

The **inMinAngle** and **inMaxAngle** parameters describe possible rotation angles of the model, i.e. only those object occurrences will be later found by LocateMultipleObjects_Edges1 whose rotation angles are in the range <**inMinAngle**,**inMaxAngle**>. The **inAnglePrecision** parameter controls the angular resolution of the matching process. The model is created in several rotations. The angles of consecutive rotations differ by an angle step from each other. The value of this angle step is determined on the basis of **inAnglePrecision**. Its value represents the multiplicity of the automatically computed angle step used as an actual step. So the greater **inAnglePrecision** is, the greater accuracy can be achieved and the lower is the chance of missing object occurrences. In practice however increasing **inAnglePrecision** above a certain threshold (unique for every object) does not increase the accuracy, only the computation time.

The **inReferenceFrame** is a characteristic rectangle, the position of which will be returned by LocateMultipleObjects_Edges1 as an occurrence of the object. By default, it is set to the bounding box of the edges found in **inTemplateRegion**.

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

Additional information about Template Matching can be found in Machine Vision Guide: Template Matching

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect scale range in CreateEdgeModel.
DomainError	Minimal pyramid level cannot be greater than maximal pyramid level in CreateEdgeModel.
DomainError	Region of interest exceeds an input image in CreateEdgeModel.

See Also

- LocateSingleObject_Edges1 – Finds a single occurrence of a predefined template on an image by comparing object edges.
- LocateMultipleObjects_Edges1 – Finds all occurrences of a predefined template on an image by comparing object edges.

CreateEdgeModel2

Header: AVL.h
Namespace: avl
Module: MatchingPro

Creates a model for edge-based template matching.

Applications: Dynamic creation of models in the runtime environment (normally they are created interactively in Studio).

Syntax

```
void avl::CreateEdgeModel2
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inTemplateRegion,
 atl::Optional<const avl::Rectangle2D&> inReferenceFrame,
 int inMinPyramidLevel,
 atl::Optional<int> inMaxPyramidLevel,
 float inSmoothingStdDev,
 float inEdgeThreshold,
 float inEdgeHysteresis,
 float inMinAngle,
 float inMaxAngle,
 float inAnglePrecision,
 float inMinScale,
 float inMaxScale,
 float inScalePrecision,
 float inEdgeCompleteness,
 atl::Conditional<avl::EdgeModel2>& outEdgeModel,
 atl::Optional<atl::Conditional<avl::Point2D>&> outEdgeModelPoint = atl::NIL,
 atl::Optional<atl::Conditional<atl::Array<avl::Path>>&> outEdges = atl::NIL,
 atl::Conditional<atl::Array<avl::Image> >& diagEdgePyramid
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image from which model will be extracted
→	inTemplateRegion	Optional<const Region&>		NIL	Region of the image from which model will be extracted
→	inReferenceFrame	Optional<const Rectangle2D&>		NIL	Exact position of the model object in the image
→	inMinPyramidLevel	int	0 - 12	1	Defines the number of reduced resolution levels dynamically created during model creation
→	inMaxPyramidLevel	Optional<int>	0 - 12	NIL	Defines the number of reduced resolution levels used to speed up computations
→	inSmoothingStdDev	float	0.0 - ∞	0.0f	Standard deviation of the gaussian smoothing applied before edge extraction
→	inEdgeThreshold	float	0.0 - ∞	35.0f	Higher threshold for edge magnitude
→	inEdgeHysteresis	float	0.0 - ∞	15.0f	Threshold hysteresis value for edge magnitude
→	inMinAngle	float		-180.0f	Start of range of possible rotations
→	inMaxAngle	float		180.0f	End of range of possible rotations
→	inAnglePrecision	float	0.001 - 10.0	1.0f	Defines angular resolution of the matching process
→	inMinScale	float	0.0 - ∞	1.0f	Start of range of possible scales
→	inMaxScale	float	0.0 - ∞	1.0f	End of range of possible scales
→	inScalePrecision	float	0.001 - 10.0	1.0f	Defines scale resolution of the matching process
→	inEdgeCompleteness	float	0.01 - 1.0	1.0f	Determines what fraction of the edges will be present in the created model
←	outEdgeModel	Conditional<EdgeModel2>&			Created model that can be used by LocateMultipleObjects_Edges
←	outEdgeModelPoint	Optional<Conditional<Point2D>&>		NIL	The middle point of the created model
←	outEdges	Optional<Conditional<Array<Path>>&>		NIL	Visualization of the model edges found at the original resolution
🔍	diagEdgePyramid	Conditional<Array<Image> >&			Visualization of the edges found at different resolution levels

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outEdgeModelPoint**, **outEdges**.

Read more about Optional Outputs.

Description

The operation creates an Edge-based Template Matching model for the object represented in **inTemplateRegion** region in **inImage** image. The resulting model can be matched against any image using the LocateSingleObject_Edges2 or LocateMultipleObjects_Edges2 filter.

The model consists of a pyramid of iteratively downsampled images, the original image being the first of them. The **inMaxPyramidLevel** parameter determines how many additional images of the pyramid shall be computed. Its value has great influence on computation speed. Therefore it is highly recommended to set its value as high as possible, at the same time ensuring that the model edges are at least 2^**inMaxPyramidLevel** pixels far from the **inImage** image frame. However, if it is set too high and no model edges are found on some pyramid level, an error with appropriate description occurs.

The **inMinPyramidLevel** parameter determines what is the lowest pyramid level that is generated in the filter. Each level lower than that will be generated on demand in a locating objects filter that uses the model.

The **inEdgeThreshold** and **inEdgeHysteresis** parameters control the hysteresis threshold (as in ThresholdImage_Hysteresis) used in the edge extraction (as in DetectEdges_AsRegion) phase of the model creation. It is an important part of using the filter to set these parameters properly, because only found edge pixels determine how good the later matching process will be. Therefore the **outEdges** and **diagEdgePyramid** are the crucial parameters for experiments, because they show found edges in the model image and edge pixels found at different resolution levels, respectively.

The **inMinAngle** and **inMaxAngle** parameters describe possible rotation angles of the model, i.e. only those object occurrences will be later found by LocateMultipleObjects_Edges2 whose rotation angles are in the range <**inMinAngle**,**inMaxAngle**>. The **inAnglePrecision** parameter controls the angular resolution of the matching process. The model is created in several rotations. The angles of consecutive rotations differ by an angle step from each other. The value of this angle step is determined on the basis of **inAnglePrecision**. Its value represents the multiplicity of the automatically computed angle step used as an actual step. So the greater **inAnglePrecision** is, the greater accuracy can be achieved and the lower is the chance of missing object occurrences. In practice however increasing **inAnglePrecision** above a certain threshold (unique for every object) does not increase the accuracy, only the computation time.

The **inReferenceFrame** is a characteristic rectangle, the position of which will be returned by LocateMultipleObjects_Edges2 as an occurrence of the object. By default, it is set to the bounding box of the edges found in **inTemplateRegion**.

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

Additional information about Template Matching can be found in Machine Vision Guide: Template Matching

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- LocateSingleObject_Edges2 – Finds a single occurrence of a predefined template on an image by comparing object edges.
- LocateMultipleObjects_Edges2 – Finds all occurrences of a predefined template on an image by comparing object edges.

CreateGrayModel

Header: AVL.h
Namespace: avl
Module: MatchingBasic

Creates a model for NCC or SAD template matching.

Applications: Dynamic creation of models in the runtime environment (normally they are created interactively in Studio).

Syntax

```
void avl::CreateGrayModel
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inTemplateRegion,
 atl::Optional<const avl::Rectangle2D&> inReferenceFrame,
 int inMinPyramidLevel,
 atl::Optional<int> inMaxPyramidLevel,
 float inMinAngle,
 float inMaxAngle,
 float inAnglePrecision,
 float inMinScale,
 float inMaxScale,
 float inScalePrecision,
 avl::GrayModel& outGrayModel,
 atl::Optional<avl::Point2D&> outGrayModelPoint = atl::NIL,
 atl::Array<avl::Image>& diagTemplatePyramid
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image from which model will be extracted
→	inTemplateRegion	Optional<const Region&>		NIL	Region of the image from which model will be extracted
→	inReferenceFrame	Optional<const Rectangle2D&>		NIL	Exact position of the model object in the image
→	inMinPyramidLevel	int	0 - 12	0	Defines the index of the lowest reduced resolution level used to speed up computations
→	inMaxPyramidLevel	Optional<int>	0 - 12	NIL	Defines the number of reduced resolution levels used to speed up computations
→	inMinAngle	float		0.0f	Start of range of possible rotations
→	inMaxAngle	float		0.0f	End of range of possible rotations
→	inAnglePrecision	float	0.001 - 10.0	1.0f	Defines angular resolution of the matching process
→	inMinScale	float	0.0 - ∞	1.0f	Start of range of possible scales
→	inMaxScale	float	0.0 - ∞	1.0f	End of range of possible scales
→	inScalePrecision	float	0.001 - 10.0	1.0f	Defines scale resolution of the matching process
←	outGrayModel	GrayModel&			Created model that can be used by LocateMultipleObjects_NCC and LocateMultipleObjects_SAD filters
←	outGrayModelPoint	Optional<Point2D&>		NIL	The middle point of the created model
🔍	diagTemplatePyramid	Array<Image>&			Visualization of the model at different resolution levels

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outGrayModelPoint**.

Read more about Optional Outputs.

Description

The operation creates a Gray Matching model for the object represented in **inTemplateRegion** region in **inImage** image. The resulting model can be matched against any image using the LocateMultipleObjects_NCC filter or the LocateMultipleObjects_SAD filter.

The model consists of a pyramid of iteratively downsampled images, the original image being the first of them. The **inMaxPyramidLevel** parameter determines how many additional images of the pyramid shall be computed. Greater **inMaxPyramidLevel** values can speed up matching process considerably, but it should be set so the image on the highest pyramid level is not too distorted.

The **inMinAngle** and **inMaxAngle** parameters describe possible rotation angles of the model, i.e. only those object occurrences will be later found by LocateMultipleObjects_NCC (or LocateMultipleObjects_SAD) whose rotation angles are in the range <**inMinAngle**,**inMaxAngle**>. The **inAnglePrecision** parameter controls the angular resolution of the matching process. The model is created in several rotations. The angles of consecutive rotations differ by an angle step from each other. The value of this angle step is determined on the basis of **inAnglePrecision**. Its value represents the multiplicity of the automatically computed angle step used as an actual step. So the greater **inAnglePrecision** is, the greater accuracy can be achieved and the lower is the chance of missing object occurrences. In practice however increasing **inAnglePrecision** above a certain threshold (unique for every object) does not increase the accuracy, only the computation time.

The **inReferenceFrame** is a characteristic rectangle, the position of which will be returned by LocateMultipleObjects_NCC (or LocateMultipleObjects_SAD) as an occurrence of the object. By default, it is set to the bounding box of the **inTemplateRegion**.

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

Additional information about Template Matching can be found in Machine Vision Guide: Template Matching

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect scale range in CreateGrayModel.
DomainError	Minimal pyramid level cannot be greater than maximal pyramid level in CreateGrayModel.
DomainError	Region of interest exceeds an input image in CreateGrayModel.

See Also

- LocateSingleObject_NCC – Finds a single occurrence of a predefined template on an image by analysing the normalized correlation between pixel values.
- LocateMultipleObjects_NCC – Finds all occurrences of a predefined template on an image by analysing the normalized correlation between pixel values.
- LocateSingleObject_SAD – Finds a single occurrence of a predefined template on an image by analysing the Square Average Difference between pixel values.
- LocateMultipleObjects_SAD – Finds multiple occurrences of a predefined template on an image by analysing the Square Average Difference between pixel values.

CreateGrayModel2

Header: AVL.h
Namespace: avl
Module: MatchingBasic

Creates a model for NCC or SAD template matching.

Applications: Dynamic creation of models in the runtime environment (normally they are created interactively in Studio).

Syntax

```
void avl::CreateGrayModel2
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inTemplateRegion,
 atl::Optional<const avl::Rectangle2D&> inReferenceFrame,
 int inMinPyramidLevel,
 atl::Optional<int> inMaxPyramidLevel,
 float inMinAngle,
 float inMaxAngle,
 float inAnglePrecision,
 float inMinScale,
 float inMaxScale,
 float inScalePrecision,
 avl::GrayModel2& outGrayModel,
 atl::Optional<avl::Point2D&> outGrayModelPoint = atl::NIL,
 atl::Array<avl::Image>& diagTemplatePyramid = atl::Dummy<atl::Array<avl::Image>>()
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image from which model will be extracted
→	inTemplateRegion	Optional<const Region&>		NIL	Region of the image from which model will be extracted
→	inReferenceFrame	Optional<const Rectangle2D&>		NIL	Exact position of the model object in the image
→	inMinPyramidLevel	int	0 - 12	0	Defines the index of the lowest reduced resolution level used to speed up computations
→	inMaxPyramidLevel	Optional<int>	0 - 12	NIL	Defines the number of reduced resolution levels used to speed up computations
→	inMinAngle	float		0.0f	Start of range of possible rotations
→	inMaxAngle	float		0.0f	End of range of possible rotations
→	inAnglePrecision	float	0.001 - 10.0	1.0f	Defines angular resolution of the matching process
→	inMinScale	float	0.2 - 5.0	1.0f	Start of range of possible scales
→	inMaxScale	float	0.2 - 5.0	1.0f	End of range of possible scales
→	inScalePrecision	float	0.001 - 10.0	1.0f	Defines scale resolution of the matching process
←	outGrayModel	GrayModel2&			Created model that can be used by LocateMultipleObjects_NCC and LocateMultipleObjects_SAD filters
←	outGrayModelPoint	Optional<Point2D&>		NIL	The middle point of the created model
🔍	diagTemplatePyramid	Array<Image>&			Visualization of the model at different resolution levels

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outGrayModelPoint**.

Read more about Optional Outputs.

Description

The operation creates a Gray-based Template Matching model for the object represented in **inTemplateRegion** region in **inImage** image. The resulting model can be matched against any image using the LocateSingleObject_NCC2 or the LocateMultipleObjects_NCC2 filter.

The model consists of a pyramid of iteratively downsampled images, the original image being the first of them. The **inMaxPyramidLevel** parameter determines how many additional images of the pyramid shall be computed. Greater **inMaxPyramidLevel** values can speed up matching process considerably, but it should be set so the image on the highest pyramid level is not too distorted.

The **inMinAngle** and **inMaxAngle** parameters describe possible rotation angles of the model, i.e. only those object occurrences will be later found by LocateSingleObject_NCC2 (or LocateMultipleObjects_NCC2) whose rotation angles are in the range <**inMinAngle**,**inMaxAngle**>. The **inAnglePrecision** parameter controls the angular resolution of the matching process. The model is created in several rotations. The angles of consecutive rotations differ by an angle step from each other. The value of this angle step is determined on the basis of **inAnglePrecision**. Its value represents the multiplicity of the automatically computed angle step used as an actual step. So the greater **inAnglePrecision** is, the greater accuracy can be achieved and the lower is the chance of missing object occurrences. In practice however increasing **inAnglePrecision** above a certain threshold (unique for every object) does not increase the accuracy, only the computation time.

The **inReferenceFrame** is a characteristic rectangle, the position of which will be returned by LocateSingleObject_NCC2 (or LocateMultipleObjects_NCC2) as an occurrence of the object. By default, it is set to the bounding box of the **inTemplateRegion**.

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

Additional information about Template Matching can be found in Machine Vision Guide: Template Matching

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- LocateSingleObject_NCC2 – Finds a single occurrence of a predefined template on an image by analysing the normalized correlation between pixel values.
- LocateMultipleObjects_NCC2 – Finds all occurrences of a predefined template on an image by analysing the normalized correlation between pixel values.

EdgeModel2ByteSize

Header: AVL.h
Namespace: avl
Module: MatchingPro

Returns estimated size of the EdgeModel2 in bytes.

Syntax

```
void avl::EdgeModel2ByteSize
(
 const avl::EdgeModel2& inEdgeModel,
 atl::int64& outByteSize
)
```

Parameters

	Name	Type	Default	Description
→	inEdgeModel	const EdgeModel2&		
←	outByteSize	int64&		

EnhanceMultipleObjectMatches

Header: AVL.h
Namespace: avl
Module: MatchingPro

Improves accuracy of multiple object matching by adding a subpixel-precise adjustment.

Applications: Usually used after an edge-based template matching tool.

Syntax

```
void avl::EnhanceMultipleObjectMatches
(
 const avl::Image& inImage,
 const atl::Array<avl::Object2D>& inObjects,
 const atl::Array<atl::Array<avl::Path>>& inObjectEdges,
 bool inAllowScale,
 atl::Array<avl::Object2D>& outObjects,
 atl::Array<atl::Array<avl::Path>>& outObjectEdges
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inObjects	const Array<Object2D>&		Input objects
→	inObjectEdges	const Array<Array<Path>>&		Input objects edges
→	inAllowScale	bool		Determines if the object scale can be adjusted
←	outObjects	Array<Object2D>&		Objects with enhanced accuracy
←	outObjectEdges	Array<Array<Path>>&		Edges of objects with enhanced accuracy

Hints

- In order to use the filter to fine-tune the results of template matching, connect the **outObjects** and **outObjectEdges** outputs of the template matching filter to the **inObjects** and **inObjectEdges** inputs, and the same input image to the **inImage** input. By default, the **outObjects** and **outObjectEdges** outputs of the template matching filter are hidden, so one should use "Show/Hide Ports" option to make them visible.
- When using template matching with **EnhanceMultipleObjectMatches** filter, you may try to speed up template matching by increasing its **inMinPyramidLevel** parameter. The potential loss of template matching precision should be mitigated by the **EnhanceMultipleObjectMatches** filter.

Hardware Acceleration

This operation is optimized for SSSE3 technology for pixels of type: UINT8.

This operation is optimized for AVX2 technology for pixels of type: UINT8.

This operation is optimized for NEON technology for pixels of type: UINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	inObjects and inObjectEdges sizes differ

See Also

- LocateMultipleObjects_Edges1 – Finds all occurrences of a predefined template on an image by comparing object edges.
- EnhanceSingleObjectMatch – Improves accuracy of single object matching by adding a subpixel-precise adjustment.

EnhanceSingleObjectMatch

Header: AVL.h
Namespace: avl
Module: MatchingPro

Improves accuracy of single object matching by adding a subpixel-precise adjustment.

Applications: Usually used after an edge-based template matching tool.

Syntax

```
void avl::EnhanceSingleObjectMatch
(
 const avl::Image& inImage,
 const avl::Object2D& inObject,
 const atl::Array<avl::Path>& inObjectEdges,
 bool inAllowScale,
 avl::Object2D& outObject,
 atl::Array<avl::Path>& outObjectEdges
)
```

Parameters

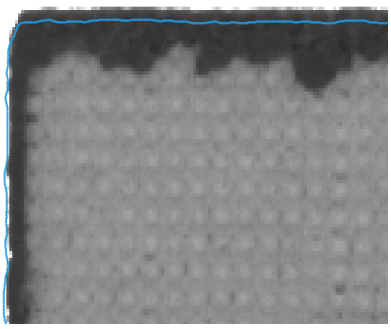
	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inObject	const Object2D&		Input object
→	inObjectEdges	const Array<Path>&		Input object edges
→	inAllowScale	bool		Determines if the object scale can be adjusted
←	outObject	Object2D&		Object with enhanced accuracy
←	outObjectEdges	Array<Path>&		Edges of the object with enhanced accuracy

Hints

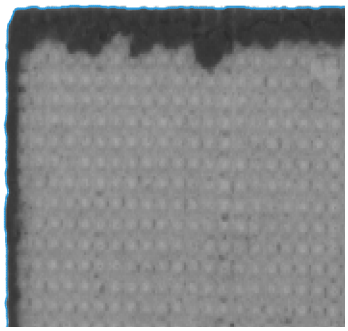
- In order to use the filter to fine-tune the results of template matching, connect the **outObject** and **outObjectEdges** outputs of the template matching filter to the **inObject** and **inObjectEdges** inputs, and the same input image to the **inImage** input. By default, the **outObject** and **outObjectEdges** outputs of the template matching filter are hidden, so one should use "Show/Hide Ports" option to make them visible.

Examples

When using template matching with **EnhanceSingleObjectMatch** filter, you may try to speed up template matching by increasing its **inMinPyramidLevel** parameter. The potential loss of template matching precision should be mitigated by the **EnhanceSingleObjectMatch** filter, as shown in the images below:



Without enhancement



With enhancement

Hardware Acceleration

This operation is optimized for SSSE3 technology for pixels of type: UINT8.

This operation is optimized for AVX2 technology for pixels of type: UINT8.

This operation is optimized for NEON technology for pixels of type: UINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- LocateSingleObject_Edges1 – Finds a single occurrence of a predefined template on an image by comparing object edges.
- EnhanceMultipleObjectMatches – Improves accuracy of multiple object matching by adding a subpixel-precise adjustment.

GrayModel2ByteSize

Header: AVL.h
Namespace: avl
Module: MatchingPro

Returns estimated size of the GrayModel2 in bytes.

Syntax

```
void avl::GrayModel2ByteSize
(
 const avl::GrayModel2& inGrayModel,
 atl::int64& outByteSize
)
```

Parameters

	Name	Type	Default	Description
→	inGrayModel	const GrayModel2&		
←	outByteSize	int64&		

LoadEdgeModel

Header: AVL.h
Namespace: avl
Module: MatchingPro

Loads serialized template matching EdgeModel object from AVDATA file.

Syntax

```
void avl::LoadEdgeModel
(
 const atl::File& inFilename,
 avl::EdgeModel& outEdgeModel
)
```

Parameters

	Name	Type	Default	Description
→	inFilename	const File&		Name of the source file
←	outEdgeModel	EdgeModel&		Deserialized output model

LoadEdgeModel2

Header: AVL.h
Namespace: avl
Module: MatchingPro

Loads serialized template matching EdgeModel2 object from AVDATA file.

Syntax

```
void avl::LoadEdgeModel2
(
 const atl::File& inFilename,
 avl::EdgeModel2& outEdgeModel
)
```

Parameters

	Name	Type	Default	Description
→	inFilename	const File&		Name of the source file
←	outEdgeModel	EdgeModel2&		Deserialized output model

LoadGrayModel

Header: AVL.h
Namespace: avl
Module: MatchingBasic

Loads serialized template matching GrayModel object from AVDATA file.

Syntax

```
void avl::LoadGrayModel
(
 const atl::File& inFilename,
 avl::GrayModel& outGrayModel
)
```

Parameters

	Name	Type	Default	Description
→	inFilename	const File&		Name of the source file
←	outGrayModel	GrayModel&		Deserialized output model

LoadGrayModel2

Header: AVL.h
Namespace: avl
Module: MatchingBasic

Loads serialized template matching GrayModel2 object from AVDATA file.

Syntax

```
void avl::LoadGrayModel2
(
 const atl::File& inFilename,
 avl::GrayModel2& outGrayModel
)
```

Parameters

	Name	Type	Default	Description
→	inFilename	const File&		Name of the source file
←	outGrayModel	GrayModel2&		Deserialized output model

LocateMultipleObjects_Edges1

Header: AVL.h
Namespace: avl
Module: MatchingPro

Finds all occurrences of a predefined template on an image by comparing object edges.

Applications: Detection of multiple objects whose outlines are sharp and rigid. Often one of the first filters in a program.

Syntax

```
void avl::LocateMultipleObjects_Edges1
(
 const avl::Image& inImage,
 atl::Optional<const avl::ShapeRegion&> inSearchRegion,
 atl::Optional<const avl::CoordinateSystem2D&> inSearchRegionAlignment,
 const avl::EdgeModel& inEdgeModel,
 int inMinPyramidLevel,
 atl::Optional<int> inMaxPyramidLevel,
 float inEdgeThreshold,
 avl::EdgePolarityMode::Type inEdgePolarityMode,
 avl::EdgeNoiseLevel::Type inEdgeNoiseLevel,
 bool inIgnoreBoundaryObjects,
 float inMinScore,
 float inMinDistance,
 atl::Array<avl::Object2D>& outObjects,
 atl::Optional<atl::Array<atl::Array<avl::Path>>&> outObjectEdges = atl::NIL,
 atl::Optional<int&> outPyramidHeight = atl::NIL,
 atl::Optional<avl::ShapeRegion&> outAlignedSearchRegion = atl::NIL,
 atl::Array<avl::Image>& diagEdgePyramid,
 atl::Array<avl::Image>& diagMatchPyramid,
 atl::Array<atl::Array<float> >& diagScores
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image on which object occurrences will be searched
→	inSearchRegion	Optional<const ShapeRegion&>		NIL	Region of possible object centers
→	inSearchRegionAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the region of interest to the position of the inspected object
→	inEdgeModel	const EdgeModel&			Model of objects to be searched
→	inMinPyramidLevel	int	0 - 12	0	Defines the lowest pyramid level at which object position is still refined
→	inMaxPyramidLevel	Optional<int>	0 - 12	3	Defines the total number of reduced resolution levels that can be used to speed up computations
→	inEdgeThreshold	float	0.01 - ∞	10.0f	Minimum strength of edges used for matching with the model
→	inEdgePolarityMode	EdgePolarityMode::Type		MatchStrictly	Defines how edges with reversed polarity will contribute to the object score
→	inEdgeNoiseLevel	EdgeNoiseLevel::Type		High	Defines how much noise the objects edges have
→	inIgnoreBoundaryObjects	bool		False	Flag indicating whether objects crossing image boundary should be ignored or not
→	inMinScore	float	0.0 - 1.0	0.7f	Minimum score of object candidates accepted at each pyramid level
→	inMinDistance	float	0.0 - ∞	10.0f	Minimum distance between two found objects
←	outObjects	Array<Object2D>&			Found objects
←	outObjectEdges	Optional<Array<Array<Path>>&>		NIL	Model edges of the found objects
←	outPyramidHeight	Optional<int&>		NIL	Highest pyramid level used to speed up computations
←	outAlignedSearchRegion	Optional<ShapeRegion&>		NIL	Transformed input shape region
🔍	diagEdgePyramid	Array<Image>&			Image edges used for matching at each pyramid level
🔍	diagMatchPyramid	Array<Image>&			Candidate object locations found at each pyramid level
🔍	diagScores	Array<Array<float> >&			Scores of the found objects at each pyramid level

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outObjectEdges**, **outPyramidHeight**, **outAlignedSearchRegion**.

Read more about Optional Outputs.

Description

The operation matches the object model, **inEdgeModel**, against the input image, **inImage**. The **inSearchRegion** region restricts the search area so that only in this region the centers of the objects can be presented. The **inMinScore** parameter determines the minimum score of the valid object occurrence. The **inMinDistance** parameter determines minimum distance between any two valid occurrences (if two occurrences lie closer than **inMinDistance** from each other, the one with greater score is considered to be valid).

In the **inImage** every pixel with gradient's magnitude at least **inEdgeThreshold** is considered an edge pixel. Only those are later used during the matching process. To establish the value of this threshold properly the **diagEdgePyramid** output which shows all the edges with sufficient gradient's magnitude may be used. If the **inEdgePolarityMode** parameter is set to Ignore, the object occurrences in the **inImage** image do not have necessarily to have the same contrast as the object in the model image.

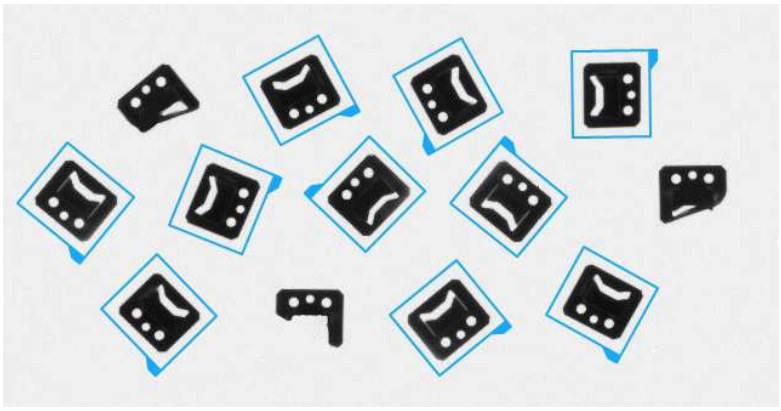
The computation time of the filter depends on the size of the model, the sizes of **inImage** and **inSearchRegion**, but also on the value of **inMinScore**. This parameter is a score threshold. Based on its value some partial computation can be sufficient to reject some locations as valid object instances. Moreover, the pyramid of the images is used. Thus, only the highest pyramid level is searched exhaustively, and potential candidates are later validated at lower levels. The **inMinPyramidLevel** parameter determines the lowest pyramid level used to validate such candidates. Setting this parameter to a value greater than 0 may speed up the computation significantly, especially for higher resolution images. However, the accuracy of the found object occurrences can be reduced. Larger **inMinScore** generates less potential candidates on the highest level to verify on lower levels. It should be noted that some valid occurrences with score above this score threshold can be missed. On higher levels score can be slightly lower than on lower levels. Thus, some valid object occurrences which on the lowest level would be deemed to be valid object instances can be incorrectly missed on some higher level. The **diagMatchPyramid** output represents all potential candidates recognized on each pyramid level and can be helpful during the difficult process of the proper parameter setting.

The **outObjects.Point** array contains the model reference points of the matched object occurrences. The corresponding **outObjects.Angle** array contains the rotation angles of the objects. The corresponding **outObjects.Match** array provides information about both the position and the angle of each match combined into value of Rectangle2D type. Each element of the **outObjects.Alignment** array contains informations about the transform required for geometrical objects defined in the context of template image to be transformed into object in the context of corresponding **outObjects.Match** position. This array can be later used e.g. by 1D Edge Detection or Shape Fitting categories filters.

Hints

- You can use EnhanceMultipleObjectMatches filter to fine-tune the results. A great example of usage is presented in the CreateGoldenTemplate2 filter.
- If an object is not detected, first try decreasing **inMaxPyramidLevel**, then try decreasing **inMinScore**.
- If all the expected objects are correctly detected, try achieving higher performance by increasing **inMaxPyramidLevel** and **inMinScore**.
- Please note, that due to the pyramid strategy used for speeding-up computations, some objects whose score would finally be above **inMinScore** may be not found. This may be surprising, but this is by design. The reason is that the minimum value is used at different pyramid levels and many objects exhibit lower score at higher pyramid levels. They get discarded before they can be evaluated at the lowest pyramid level. Decrease **inMinScore** experimentally until all objects are found.
- If precision of matching is not very important, you can also gain some performance by increasing **inMinPyramidLevel**.
- If the performance is still not satisfactory, go back to model definition and try reducing the range of rotations and scaling as well as the precision-related parameters: **inAnglePrecision**, **inScalePrecision** and **inEdgeCompleteness**.
- To obtain high accuracy of matching make sure that the edges visible on the **diagEdgePyramid** output are not too thick. Increase **inEdgeThreshold** to make them thinner.
- Also consider increasing **inEdgeThreshold** if there is much noise on the **diagEdgePyramid** output.
- Verify accuracy of the matching by checking the **outObjectEdges** output. If the accuracy is lower than expected, make sure that **inMinPyramidLevel** = 0, try increasing **inEdgeThreshold** and also consider higher values for **inAnglePrecision** during model creation.
- To further improve matching accuracy use the EnhanceMultipleObjectMatches filter.
- If the object being detected can be darker or brighter than the background on different images, use the **inEdgePolarityMode** parameter set to Ignore.
- If the object being detected is only partially visible or its shape and the model shape are not identical, use the **inEdgePolarityMode** parameter set to MatchWeakly.
- If there is only low noise in the object edges, use the **inEdgeNoiseLevel** parameter set to Low.
- Define **inSearchRegion** to limit possible object locations and speed-up computations. Please note, that this is the region where the **outObject.Point** results belong to (and NOT the region within which the entire object has to be contained).

Examples



Locating multiple objects with the edge-based method.

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

Additional information about Template Matching can be found in Machine Vision Guide: Template Matching

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Unsupported IgnoreGlobally edge polarity mode in LocateMultipleObjects_Edges1.

See Also

- LocateSingleObject_Edges1 – Finds a single occurrence of a predefined template on an image by comparing object edges.
- CreateEdgeModel1 – Creates a model for edge-based template matching.
- EnhanceMultipleObjectMatches – Improves accuracy of multiple object matching by adding a subpixel-precise adjustment.

LocateMultipleObjects_Edges2

Header: AVL.h
Namespace: avl
Module: MatchingPro

Finds all occurrences of a predefined template on an image by comparing object edges.

Applications: Detection of multiple objects whose outlines are sharp and rigid. Often one of the first filters in a program.

Syntax

```
void avl::LocateMultipleObjects_Edges2
(
 const avl::Image& inImage,
 atl::Optional<const avl::ShapeRegion&> inSearchRegion,
 atl::Optional<const avl::CoordinateSystem2D&> inSearchRegionAlignment,
 const avl::EdgeModel2& inEdgeModel,
 int inMinPyramidLevel,
 atl::Optional<int> inMaxPyramidLevel,
 float inEdgeThreshold,
 avl::EdgePolarityMode::Type inEdgePolarityMode,
 avl::EdgeNoiseLevel::Type inEdgeNoiseLevel,
 bool inIgnoreBoundaryObjects,
 int inMaxDeformation,
 float inMinScore,
 float inMinDistance,
 atl::Array<avl::Object2D>& outObjects,
 atl::Optional<atl::Array<atl::Array<avl::Path>>&> outObjectEdges = atl::NIL,
 atl::Optional<int&> outPyramidHeight = atl::NIL,
 atl::Optional<avl::ShapeRegion&> outAlignedSearchRegion = atl::NIL,
 atl::Array<avl::Image>& diagEdgePyramid,
 atl::Array<avl::Image>& diagMatchPyramid,
 atl::Array<atl::Array<float> >& diagScores
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inImage	const Image&			Image on which object occurrences will be searched
➡	inSearchRegion	Optional<const ShapeRegion&>		NIL	Region of possible object centers
➡	inSearchRegionAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the region of interest to the position of the inspected object
➡	inEdgeModel	const EdgeModel2&			Model of objects to be searched
➡	inMinPyramidLevel	int	0 - 12	0	Defines the lowest pyramid level at which object position is still refined
➡	inMaxPyramidLevel	Optional<int>	0 - 12	3	Defines the total number of reduced resolution levels that can be used to speed up computations
➡	inEdgeThreshold	float	0.01 - ∞	10.0f	Minimum strength of edges used for matching with the model
➡	inEdgePolarityMode	EdgePolarityMode::Type		MatchStrictly	Defines how edges with reversed polarity will contribute to the object score
➡	inEdgeNoiseLevel	EdgeNoiseLevel::Type		High	Defines how much noise the objects edges have
➡	inIgnoreBoundaryObjects	bool		False	Flag indicating whether objects crossing image boundary should be ignored or not
➡	inMaxDeformation	int	0 - ∞	0	Maximal distance in pixels between model edge and an edge visible in the input image
➡	inMinScore	float	0.0 - 1.0	0.7f	Minimum score of object candidates accepted at each pyramid level
➡	inMinDistance	float	0.0 - ∞	10.0f	Minimum distance between two found objects
⬅	outObjects	Array<Object2D>&			Found objects
⬅	outObjectEdges	Optional<Array<Array<Path>>&>		NIL	Model edges of the found objects
⬅	outPyramidHeight	Optional<int&>		NIL	Highest pyramid level used to speed up computations
⬅	outAlignedSearchRegion	Optional<ShapeRegion&>		NIL	Transformed input shape region
🔍	diagEdgePyramid	Array<Image>&			Image edges used for matching at each pyramid level
🔍	diagMatchPyramid	Array<Image>&			Candidate object locations found at each pyramid level
🔍	diagScores	Array<Array<float> >&			Scores of the found objects at each pyramid level

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outObjectEdges**, **outPyramidHeight**, **outAlignedSearchRegion**.

Read more about Optional Outputs.

Description

The operation matches the object model, **inEdgeModel**, against the input image, **inImage**. The **inSearchRegion** region restricts the search area so that only in this region the centers of the objects can be presented. The **inMinScore** parameter determines the minimum score of the valid object occurrence. The **inMinDistance** parameter determines minimum distance between any two valid occurrences (if two occurrences lie closer than **inMinDistance** from each other, the one with greater score is considered to be valid).

In the **inImage** every pixel with gradient's magnitude at least **inEdgeThreshold** is considered an edge pixel. Only those are later used during the matching process. To establish the value of this threshold properly the **diagEdgePyramid** output which shows all the edges with sufficient gradient's magnitude may be used. If the **inEdgePolarityMode** parameter is set to Ignore, the object occurrences in the **inImage** image do not have necessarily to have the same contrast as the object in the model image.

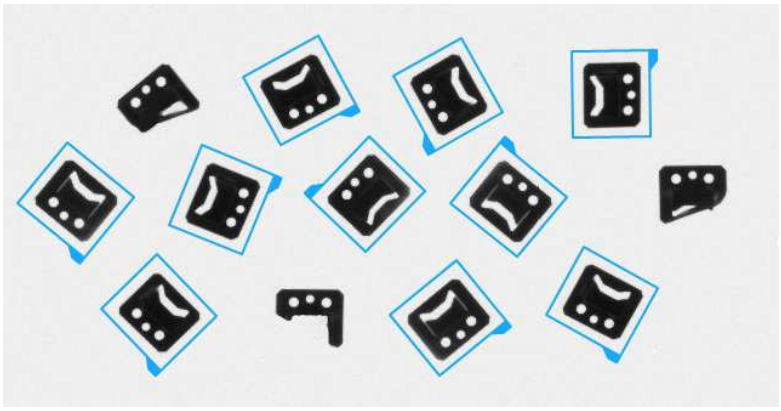
The computation time of the filter depends on the size of the model, the sizes of **inImage** and **inSearchRegion**, but also on the value of **inMinScore**. This parameter is a score threshold. Based on its value some partial computation can be sufficient to reject some locations as valid object instances. Moreover, the pyramid of the images is used. Thus, only the highest pyramid level is searched exhaustively, and potential candidates are later validated at lower levels. The **inMinPyramidLevel** parameter determines the lowest pyramid level used to validate such candidates. Setting this parameter to a value greater than 0 may speed up the computation significantly, especially for higher resolution images. However, the accuracy of the found object occurrences can be reduced. Larger **inMinScore** generates less potential candidates on the highest level to verify on lower levels. It should be noted that some valid occurrences with score above this score threshold can be missed. On higher levels score can be slightly lower than on lower levels. Thus, some valid object occurrences which on the lowest level would be deemed to be valid object instances can be incorrectly missed on some higher level. The **diagMatchPyramid** output represents all potential candidates recognized on each pyramid level and can be helpful during the difficult process of the proper parameter setting.

The **outObjects.Point** array contains the model reference points of the matched object occurrences. The corresponding **outObjects.Angle** array contains the rotation angles of the objects. The corresponding **outObjects.Match** array provides information about both the position and the angle of each match combined into value of Rectangle2D type. Each element of the **outObjects.Alignment** array contains informations about the transform required for geometrical objects defined in the context of template image to be transformed into object in the context of corresponding **outObjects.Match** position. This array can be later used e.g. by 1D Edge Detection or Shape Fitting categories filters.

Hints

- You can use EnhanceMultipleObjectMatches filter to fine-tune the results. A great example of usage is presented in the CreateGoldenTemplate2 filter.
- If an object is not detected, first try decreasing **inMaxPyramidLevel**, then try decreasing **inMinScore**.
- If all the expected objects are correctly detected, try achieving higher performance by increasing **inMaxPyramidLevel** and **inMinScore**.
- Please note, that due to the pyramid strategy used for speeding-up computations, some objects whose score would finally be above **inMinScore** may be not found. This may be surprising, but this is by design. The reason is that the minimum value is used at different pyramid levels and many objects exhibit lower score at higher pyramid levels. They get discarded before they can be evaluated at the lowest pyramid level. Decrease **inMinScore** experimentally until all objects are found.
- If precision of matching is not very important, you can also gain some performance by increasing **inMinPyramidLevel**.
- If the performance is still not satisfactory, go back to model definition and try reducing the range of rotations and scaling as well as the precision-related parameters: **inAnglePrecision**, **inScalePrecision** and **inEdgeCompleteness**.
- To obtain high accuracy of matching make sure that the edges visible on the **diagEdgePyramid** output are not too thick. Increase **inEdgeThreshold** to make them thinner.
- Also consider increasing **inEdgeThreshold** if there is much noise on the **diagEdgePyramid** output.
- Verify accuracy of the matching by checking the **outObjectEdges** output. If the accuracy is lower than expected, make sure that **inMinPyramidLevel** = 0, try increasing **inEdgeThreshold** and also consider higher values for **inAnglePrecision** during model creation.
- To further improve matching accuracy use the EnhanceMultipleObjectMatches filter.
- If the object being detected can be darker or brighter than the background on different images, use the **inEdgePolarityMode** parameter set to Ignore.
- If the object being detected is only partially visible or its shape and the model shape are not identical, use the **inEdgePolarityMode** parameter set to MatchWeakly.
- If there is only low noise in the object edges, use the **inEdgeNoiseLevel** parameter set to Low.
- Define **inSearchRegion** to limit possible object locations and speed-up computations. Please note, that this is the region where the **outObject.Point** results belong to (and NOT the region within which the entire object has to be contained).

Examples



Locating multiple objects with the edge-based method.

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

Additional information about Template Matching can be found in Machine Vision Guide: Template Matching

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- LocateSingleObject_Edges2 – Finds a single occurrence of a predefined template on an image by comparing object edges.
- CreateEdgeModel2 – Creates a model for edge-based template matching.
- EnhanceMultipleObjectMatches – Improves accuracy of multiple object matching by adding a subpixel-precise adjustment.

LocateMultipleObjects_NCC

Header: AVL.h
Namespace: avl
Module: MatchingBasic

Finds all occurrences of a predefined template on an image by analysing the normalized correlation between pixel values.

Applications: Detection of objects with blurred or unclear edges. Often one of the first filters in a program.

Syntax

```
void avl::LocateMultipleObjects_NCC
(
 const avl::Image& inImage,
 atl::Optional<const avl::ShapeRegion&> inSearchRegion,
 atl::Optional<const avl::CoordinateSystem2D&> inSearchRegionAlignment,
 const avl::GrayModel& inGrayModel,
 int inMinPyramidLevel,
 atl::Optional<int> inMaxPyramidLevel,
 bool inIgnoreBoundaryObjects,
 float inMinScore,
 float inMinDistance,
 atl::Optional<float> inMaxBrightnessRatio,
 atl::Optional<float> inMaxContrastRatio,
 atl::Array<avl::Object2D>& outObjects,
 atl::Optional<int&> outPyramidHeight = atl::NIL,
 atl::Optional<avl::ShapeRegion&> outAlignedSearchRegion = atl::NIL,
 atl::Array<avl::Image>& diagImagePyramid = atl::Dummy<atl::Array<avl::Image>>(),
 atl::Array<avl::Image>& diagMatchPyramid = atl::Dummy<atl::Array<avl::Image>>(),
 atl::Array<atl::Array<float>>& diagScores = atl::Dummy<atl::Array<atl::Array<float>>>()
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inImage	const Image&			Image on which model occurrences will be searched
➡	inSearchRegion	Optional<const ShapeRegion&>		NIL	Range of possible object centers
➡	inSearchRegionAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the region of interest to the position of the inspected object
➡	inGrayModel	const GrayModel&			Model of objects to be searched
➡	inMinPyramidLevel	int	0 - 12	0	Defines the lowest pyramid level at which object position is still refined
➡	inMaxPyramidLevel	Optional<int>	0 - 12	3	Defines the total number of reduced resolution levels that can be used to speed up computations
➡	inIgnoreBoundaryObjects	bool		False	Flag indicating whether objects crossing image boundary should be ignored or not
➡	inMinScore	float	-1.0 - 1.0	0.7f	Minimum score of object candidates accepted at each pyramid level
➡	inMinDistance	float	0.0 - ∞	10.0f	Minimum distance between two found objects
➡	inMaxBrightnessRatio	Optional<float>	1.0 - ∞	NIL	Defines the maximal deviation of the mean brightness of the model object and the object present in the image
➡	inMaxContrastRatio	Optional<float>	1.0 - ∞	NIL	Defines the maximal deviation of the brightness standard deviation of the model object and the object present in the image
⬅	outObjects	Array<Object2D>&			Found objects
⬅	outPyramidHeight	Optional<int&>		NIL	Highest pyramid level used to speed up computations
⬅	outAlignedSearchRegion	Optional<ShapeRegion&>		NIL	Transformed input shape region
🔍	diagImagePyramid	Array<Image>&			Pyramid of iteratively downsampled input image
🔍	diagMatchPyramid	Array<Image>&			Candidate object locations found at each pyramid level
🔍	diagScores	Array<Array<float>>&			Scores of the found objects at each pyramid level

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outPyramidHeight**, **outAlignedSearchRegion**.

Read more about Optional Outputs.

Description

The operation matches the object model, **inGrayModel**, against the input image, **inImage**. The **inSearchRegion** region restricts the search area so that only in this region the centers of the objects can be presented. The **inMinScore** parameter determines the minimum score of the valid object occurrence. The **inMinDistance** parameter determines minimum distance between any two valid occurrences (if two occurrences lie closer than **inMinDistance** from each other, the one with greater score is considered to be valid).

The computation time of the filter depends on the size of the model, the sizes of **inImage** and **inSearchRegion**, but also on the value of **inMinScore**. This parameter is a score threshold. Based on its value some partial computation can be sufficient to reject some locations as valid object instances. Moreover, the pyramid of the images is used. Thus, only the highest pyramid level is searched exhaustively, and potential candidates are later validated at lower levels. The **inMinPyramidLevel** parameter determines the lowest pyramid level used to validate such candidates. Setting this parameter to a value greater than 0 may speed up the computation significantly, especially for higher resolution images. However, the accuracy of the found object occurrences can be reduced. Larger **inMinScore** generates less potential candidates on the highest level to verify on lower levels. It should be noted that some valid occurrences with score above this score threshold can be missed. On higher levels score can be slightly lower than on lower levels. Thus, some valid object occurrences which on the lowest level would be deemed to be valid object instances can be incorrectly missed on some higher level. The **diagMatchPyramid** output represents all potential candidates recognized on each pyramid level and can be helpful during the difficult process of the proper parameter setting.

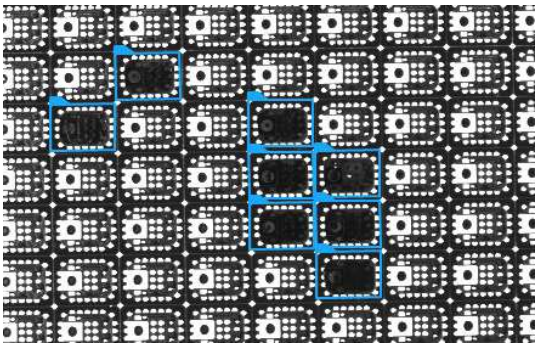
To be able to locate objects which are partially outside the image, the filter assumes that there are only black pixels beyond the image border.

The **outObjects.Point** array contains the model reference points of the matched object occurrences. The corresponding **outObjects.Angle** array contains the rotation angles of the objects. The corresponding **outObjects.Match** array provides information about both the position and the angle of each match combined into value of Rectangle2D type. Each element of the **outObjects.Alignment** array contains informations about the transform required for geometrical objects defined in the context of template image to be transformed into object in the context of corresponding **outObjects.Match** position. This array can be later used e.g. by 1D Edge Detection or Shape Fitting categories filters.

Hints

- If an object is not detected, first try decreasing **inMaxPyramidLevel**, then try decreasing **inMinScore**. However, if you need to lower **inMinScore** below 0.5, then probably something else is wrong.
- If all the expected objects are correctly detected, try achieving higher performance by increasing **inMaxPyramidLevel** and **inMinScore**.
- Please note, that due to the pyramid strategy used for speeding-up computations, some objects whose score would finally be above **inMinScore** may be not found. This may be surprising, but this is by design. The reason is that the minimum value is used at different pyramid levels and many objects exhibit lower score at higher pyramid levels. They get discarded before they can be evaluated at the lowest pyramid level. Decrease **inMinScore** experimentally until all objects are found.
- If precision of matching is not very important, you can also gain some performance by increasing **inMinPyramidLevel**.
- If the performance is still not satisfactory, go back to model definition and try reducing the range of rotations and scaling as well as the precision-related parameters: **inAnglePrecision** and **inScalePrecision**.
- Define **inSearchRegion** to limit possible object locations and speed-up computations. Please note, that this is the region where the **outObject.Point** results belong to (and NOT the region within which the entire object has to be contained).
- Smoothing of the input image can significantly improve performance of this tool.

Examples



*Locating multiple objects with the gray-based method (**inMaxPyramidLevel** = 2). Please note, that the edge-based method might not work properly here, because what we are looking for (a missing chip) is a sub-object of what we are not looking for (a correct chip).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

Additional information about Template Matching can be found in Machine Vision Guide: Template Matching

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of type: UINT8.

This operation is optimized for AVX2 technology for pixels of type: UINT8.

This operation is optimized for NEON technology for pixels of type: UINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- LocateSingleObject_NCC – Finds a single occurrence of a predefined template on an image by analysing the normalized correlation between pixel values.
- CreateGrayModel – Creates a model for NCC or SAD template matching.
- LocateMultipleObjects_SAD – Finds multiple occurrences of a predefined template on an image by analysing the Square Average Difference between pixel values.

LocateMultipleObjects_NCC2

Header: AVL.h
Namespace: avl
Module: MatchingBasic

Finds all occurrences of a predefined template on an image by analysing the normalized correlation between pixel values.

Applications: Detection of objects with blurred or unclear edges. Often one of the first filters in a program.

Syntax

```
void avl::LocateMultipleObjects_NCC2
(
 const avl::Image& inImage,
 atl::Optional<const avl::ShapeRegion&> inSearchRegion,
 atl::Optional<const avl::CoordinateSystem2D&> inSearchRegionAlignment,
 const avl::GrayModel2& inGrayModel,
 int inMinPyramidLevel,
 atl::Optional<int> inMaxPyramidLevel,
 bool inIgnoreBoundaryObjects,
 float inMinScore,
 float inMinDistance,
 atl::Optional<float> inMaxBrightnessRatio,
 atl::Optional<float> inMaxContrastRatio,
 atl::Array<avl::Object2D>& outObjects,
 atl::Optional<int&> outPyramidHeight = atl::NIL,
 atl::Optional<avl::ShapeRegion&> outAlignedSearchRegion = atl::NIL,
 atl::Array<avl::Image>& diagImagePyramid = atl::Dummy<atl::Array<avl::Image>>(),
 atl::Array<avl::Image>& diagMatchPyramid = atl::Dummy<atl::Array<avl::Image>>(),
 atl::Array<atl::Array<float>>& diagScores = atl::Dummy<atl::Array<atl::Array<float>>>()
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inImage	const Image&			Image on which model occurrences will be searched
➡	inSearchRegion	Optional<const ShapeRegion&>		NIL	Range of possible object centers
➡	inSearchRegionAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the region of interest to the position of the inspected object
➡	inGrayModel	const GrayModel2&			Model of objects to be searched
➡	inMinPyramidLevel	int	0 - 12	0	Defines the lowest pyramid level at which object position is still refined
➡	inMaxPyramidLevel	Optional<int>	0 - 12	3	Defines the total number of reduced resolution levels that can be used to speed up computations
➡	inIgnoreBoundaryObjects	bool		False	Flag indicating whether objects crossing image boundary should be ignored or not
➡	inMinScore	float	-1.0 - 1.0	0.7f	Minimum score of object candidates accepted at each pyramid level
➡	inMinDistance	float	0.0 - ∞	10.0f	Minimum distance between two found objects
➡	inMaxBrightnessRatio	Optional<float>	1.0 - ∞	NIL	Defines the maximal deviation of the mean brightness of the model object and the object present in the image
➡	inMaxContrastRatio	Optional<float>	1.0 - ∞	NIL	Defines the maximal deviation of the brightness standard deviation of the model object and the object present in the image
⬅	outObjects	Array<Object2D>&			Found objects
⬅	outPyramidHeight	Optional<int&>		NIL	Highest pyramid level used to speed up computations
⬅	outAlignedSearchRegion	Optional<ShapeRegion&>		NIL	Transformed input shape region
🔍	diagImagePyramid	Array<Image>&			Pyramid of iteratively downsampled input image
🔍	diagMatchPyramid	Array<Image>&			Candidate object locations found at each pyramid level
🔍	diagScores	Array<Array<float>>&			Scores of the found objects at each pyramid level

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outPyramidHeight**, **outAlignedSearchRegion**.

Read more about Optional Outputs.

Description

The operation matches the object model, **inGrayModel**, against the input image, **inImage**. The **inSearchRegion** region restricts the search area so that only in this region the centers of the objects can be presented. The **inMinScore** parameter determines the minimum score of the valid object occurrence. The **inMinDistance** parameter determines minimum distance between any two valid occurrences (if two occurrences lie closer than **inMinDistance** from each other, the one with greater score is considered to be valid).

The computation time of the filter depends on the size of the model, the sizes of **inImage** and **inSearchRegion**, but also on the value of **inMinScore**. This parameter is a score threshold. Based on its value some partial computation can be sufficient to reject some locations as valid object instances. Moreover, the pyramid of the images is used. Thus, only the highest pyramid level is searched exhaustively, and potential candidates are later validated at lower levels. The **inMinPyramidLevel** parameter determines the lowest pyramid level used to validate such candidates. Setting this parameter to a value greater than 0 may speed up the computation significantly, especially for higher resolution images. However, the accuracy of the found object occurrences can be reduced. Larger **inMinScore** generates less potential candidates on the highest level to verify on lower levels. It should be noted that some valid occurrences with score above this score threshold can be missed. On higher levels score can be slightly lower than on lower levels. Thus, some valid object occurrences which on the lowest level would be deemed to be valid object instances can be incorrectly missed on some higher level. The **diagMatchPyramid** output represents all potential candidates recognized on each pyramid level and can be helpful during the difficult process of the proper parameter setting.

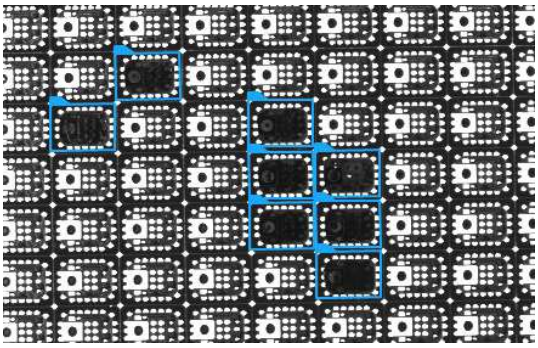
To be able to locate objects which are partially outside the image, the filter assumes that there are only black pixels beyond the image border.

The **outObjects.Point** array contains the model reference points of the matched object occurrences. The corresponding **outObjects.Angle** array contains the rotation angles of the objects. The corresponding **outObjects.Match** array provides information about both the position and the angle of each match combined into value of Rectangle2D type. Each element of the **outObjects.Alignment** array contains informations about the transform required for geometrical objects defined in the context of template image to be transformed into object in the context of corresponding **outObjects.Match** position. This array can be later used e.g. by 1D Edge Detection or Shape Fitting categories filters.

Hints

- If an object is not detected, first try decreasing **inMaxPyramidLevel**, then try decreasing **inMinScore**. However, if you need to lower **inMinScore** below 0.5, then probably something else is wrong.
- If all the expected objects are correctly detected, try achieving higher performance by increasing **inMaxPyramidLevel** and **inMinScore**.
- Please note, that due to the pyramid strategy used for speeding-up computations, some objects whose score would finally be above **inMinScore** may be not found. This may be surprising, but this is by design. The reason is that the minimum value is used at different pyramid levels and many objects exhibit lower score at higher pyramid levels. They get discarded before they can be evaluated at the lowest pyramid level. Decrease **inMinScore** experimentally until all objects are found.
- If precision of matching is not very important, you can also gain some performance by increasing **inMinPyramidLevel**.
- If the performance is still not satisfactory, go back to model definition and try reducing the range of rotations and scaling as well as the precision-related parameters: **inAnglePrecision** and **inScalePrecision**.
- Define **inSearchRegion** to limit possible object locations and speed-up computations. Please note, that this is the region where the **outObject.Point** results belong to (and NOT the region within which the entire object has to be contained).
- Smoothing of the input image can significantly improve performance of this tool.

Examples



*Locating multiple objects with the gray-based method (**inMaxPyramidLevel** = 2). Please note, that the edge-based method might not work properly here, because what we are looking for (a missing chip) is a sub-object of what we are not looking for (a correct chip).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

Additional information about Template Matching can be found in Machine Vision Guide: Template Matching

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of type: UINT8.

This operation is optimized for AVX2 technology for pixels of type: UINT8.

This operation is optimized for NEON technology for pixels of type: UINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Invalid GrayModel2 in LocateMultipleObjects_NCC2. Create a proper model first using the Template Matching plugin or CreateGrayModel2 filter.

See Also

- LocateSingleObject_NCC2 – Finds a single occurrence of a predefined template on an image by analysing the normalized correlation between pixel values.
- CreateGrayModel2 – Creates a model for NCC or SAD template matching.

LocateMultipleObjects_SAD

Header: AVL.h
Namespace: avl
Module: MatchingBasic

Finds multiple occurrences of a predefined template on an image by analysing the Square Average Difference between pixel values.

Applications: Almost always inferior to NCC, so rarely used in real applications.

Syntax

```
void avl::LocateMultipleObjects_SAD
(
 const avl::Image& inImage,
 atl::Optional<const avl::ShapeRegion&> inSearchRegion,
 atl::Optional<const avl::CoordinateSystem2D&> inSearchRegionAlignment,
 const avl::GrayModel& inGrayModel,
 int inMinPyramidLevel,
 atl::Optional<int> inMaxPyramidLevel,
 bool inIgnoreBoundaryObjects,
 float inMaxDifference,
 float inMinDistance,
 atl::Array<avl::Object2D>& outObjects,
 atl::Optional<int&> outPyramidHeight = atl::NIL,
 atl::Optional<avl::ShapeRegion&> outAlignedSearchRegion = atl::NIL,
 atl::Array<avl::Image>& diagImagePyramid = atl::Dummy<atl::Array<avl::Image>>(),
 atl::Array<avl::Image>& diagMatchPyramid = atl::Dummy<atl::Array<avl::Image>>(),
 atl::Array<atl::Array<float> >& diagScores = atl::Dummy<atl::Array<atl::Array<float>>>()
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image on which model occurrences will be searched
→	inSearchRegion	Optional<const ShapeRegion&>		NIL	Possible centers of the object occurrences
→	inSearchRegionAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the region of interest to the position of the inspected object
→	inGrayModel	const GrayModel&			Model which will be sought
→	inMinPyramidLevel	int	0 - 12	0	Defines the highest resolution level
→	inMaxPyramidLevel	Optional<int>	0 - 12	3	Defines the number of reduced resolution levels that can be used to speed up computations
→	inIgnoreBoundaryObjects	bool		False	Flag indicating whether objects crossing image boundary should be ignored or not
→	inMaxDifference	float	0.0 - ∞	5.0f	Maximum accepted average difference between pixel values
→	inMinDistance	float	0.0 - ∞	10.0f	Minimum distance between two matches
←	outObjects	Array<Object2D>&			Found objects
←	outPyramidHeight	Optional<int&>		NIL	Highest pyramid level used to speed up computations
←	outAlignedSearchRegion	Optional<ShapeRegion&>		NIL	Transformed input shape region
🔍	diagImagePyramid	Array<Image>&			Pyramid of iteratively downsampled input image
🔍	diagMatchPyramid	Array<Image>&			Locations found on each pyramid level
🔍	diagScores	Array<Array<float> >&			Scores of found matches on each pyramid level

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outPyramidHeight**, **outAlignedSearchRegion**.

Read more about Optional Outputs.

Description

The operation matches the object model, **inGrayModel**, against the input image, **inImage**. The **inSearchRegion** region restricts the search area so that only in this region the centers of the objects can be presented. The **inMaxDifference** parameter determines the maximum average difference between corresponding pixel values of the valid object occurrence. The **inMinDistance** parameter determines minimum distance between any two valid occurrences (if two occurrences lie closer than **inMinDistance** from each other, the one with greater score is considered to be valid).

The computation time of the filter depends on the size of the model, the sizes of **inImage** and **inSearchRegion**, but also on the value of **inMaxDifference**. This parameter is a score threshold. Based on its value some partial computation can be sufficient to reject some locations as valid object instances. Moreover, the pyramid of the images is used. Thus, only the highest pyramid level is searched exhaustively, and potential candidates are later validated at lower levels. The **inMinPyramidLevel** parameter determines the lowest pyramid level used to validate such candidates. Setting this parameter to a value greater than 0 may speed up the computation significantly, especially for higher resolution images. However, the accuracy of the found object occurrences can be reduced. Lower **inMaxDifference** generates less potential candidates on the highest level to verify on lower levels. It should be noted that some valid occurrences with score above this score threshold can be missed. On higher levels score can be slightly lower than on lower levels. Thus, some valid object occurrences which on the lowest level would be deemed to be valid object instances can be incorrectly missed on some higher level. The **diagMatchPyramid** output represents all potential candidates recognized on each pyramid level and can be helpful during the difficult process of the proper parameter setting.

To be able to locate objects which are partially outside the image, the filter assumes that there are only black pixels beyond the image border.

The **outObjects.Point** array contains the model reference points of the matched object occurrences. The corresponding **outObjects.Angle** array contains the rotation angles of the objects. The corresponding **outObjects.Match** array provides information about both the position and the angle of each match combined into value of Rectangle2D type. Each element of the **outObjects.Alignment** array contains informations about the transform required for geometrical objects defined in the context of template image to be transformed into object in the context of corresponding **outObjects.Match** position. This array can be later used e.g. by 1D Edge Detection or Shape Fitting categories filters.

The SAD (Sum of Absolute Differences) method can be significantly slower than NCC (Normalized Cross-Correlation) method. Moreover, it is not illumination-invariant, as it is required in most applications. Thus, it is highly recommended to use the latter, NCC method instead.

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

Additional information about Template Matching can be found in Machine Vision Guide: Template Matching

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- LocateSingleObject_SAD – Finds a single occurrence of a predefined template on an image by analysing the Square Average Difference between pixel values.
- CreateGrayModel – Creates a model for NCC or SAD template matching.
- LocateMultipleObjects_NCC – Finds all occurrences of a predefined template on an image by analysing the normalized correlation between pixel values.

LocateSingleObject_Edges1

Header: AVL.h
Namespace: avl
Module: MatchingPro

Finds a single occurrence of a predefined template on an image by comparing object edges.

Applications: Detection of an object whose outlines are sharp and rigid. Often one of the first filters in a program.

Syntax

```
void avl::LocateSingleObject_Edges1
(
 const avl::Image& inImage,
 atl::Optional<const avl::ShapeRegion&> inSearchRegion,
 atl::Optional<const avl::CoordinateSystem2D&> inSearchRegionAlignment,
 const avl::EdgeModel& inEdgeModel,
 int inMinPyramidLevel,
 atl::Optional<int> inMaxPyramidLevel,
 float inEdgeThreshold,
 avl::EdgePolarityMode::Type inEdgePolarityMode,
 avl::EdgeNoiseLevel::Type inEdgeNoiseLevel,
 bool inIgnoreBoundaryObjects,
 float inMinScore,
 atl::Conditional<avl::Object2D>& outObject,
 atl::Optional<atl::Conditional<atl::Array<avl::Path>>&> outObjectEdges = atl::NIL,
 atl::Optional<int&> outPyramidHeight = atl::NIL,
 atl::Optional<avl::ShapeRegion&> outAlignedSearchRegion = atl::NIL,
 atl::Array<avl::Image>& diagEdgePyramid,
 atl::Array<avl::Image>& diagMatchPyramid,
 atl::Conditional<atl::Array<float> >& diagScores
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image on which object occurrence will be searched
→	inSearchRegion	Optional<const ShapeRegion&>		NIL	Region of possible object centers
→	inSearchRegionAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the region of interest to the position of the inspected object
→	inEdgeModel	const EdgeModel&			Model of objects to be searched
→	inMinPyramidLevel	int	0 - 12	0	Defines the lowest pyramid level at which object position is still refined
→	inMaxPyramidLevel	Optional<int>	0 - 12	3	Defines the total number of reduced resolution levels that can be used to speed up computations
→	inEdgeThreshold	float	0.01 - ∞	10.0f	Minimum strength of edges used for matching with the model
→	inEdgePolarityMode	EdgePolarityMode::Type		MatchStrictly	Defines how edges with reversed polarity will contribute to the object score
→	inEdgeNoiseLevel	EdgeNoiseLevel::Type		High	Defines how much noise the object edges have
→	inIgnoreBoundaryObjects	bool		False	Flag indicating whether objects crossing image boundary should be ignored or not
→	inMinScore	float	0.0 - 1.0	0.7f	Minimum score of object candidates accepted at each pyramid level
←	outObject	Conditional<Object2D>&			Found object
←	outObjectEdges	Optional<Conditional<Array<Path>>&>		NIL	Model edges of the found object
←	outPyramidHeight	Optional<int&>		NIL	Highest pyramid level used to speed up computations
←	outAlignedSearchRegion	Optional<ShapeRegion&>		NIL	Transformed input shape region
🔍	diagEdgePyramid	Array<Image>&			Image edges used for matching at each pyramid level
🔍	diagMatchPyramid	Array<Image>&			Candidate object locations found at each pyramid level
🔍	diagScores	Conditional<Array<float> >&			Scores of the found object at each pyramid level

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outObjectEdges**, **outPyramidHeight**, **outAlignedSearchRegion**.

Read more about Optional Outputs.

Description

The operation matches the object model, **inEdgeModel**, against the input image, **inImage**. The **inSearchRegion** region restricts the search area so that only in this region the centers of the objects can be presented. The **inMinScore** parameter determines the minimum score of the valid object occurrence.

In the **inImage** every pixel with gradient's magnitude at least **inEdgeThreshold** is considered an edge pixel. Only those are later used during the matching process. To establish the value of this threshold properly the **diagEdgePyramid** output which shows all the edges with sufficient gradient's magnitude may be used. If the **inEdgePolarityMode** parameter is set to Ignore, the object occurrences in the **inImage** image do not have necessarily to have the same contrast as the object in the model image.

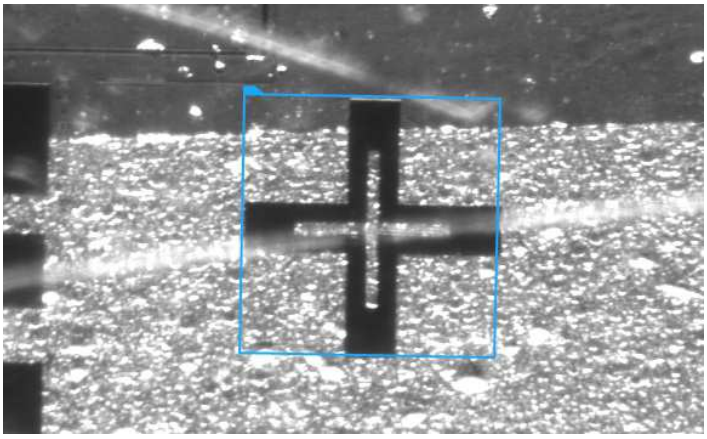
The computation time of the filter depends on the size of the model, the sizes of **inImage** and **inSearchRegion**, but also on the value of **inMinScore**. This parameter is a score threshold. Based on its value some partial computation can be sufficient to reject some locations as valid object instances. Moreover, the pyramid of the images is used. Thus, only the highest pyramid level is searched exhaustively, and potential candidates are later validated at lower levels. The **inMinPyramidLevel** parameter determines the lowest pyramid level used to validate such candidates. Setting this parameter to a value greater than 0 may speed up the computation significantly, especially for higher resolution images. However, the accuracy of the found object occurrences can be reduced. Larger **inMinScore** generates less potential candidates on the highest level to verify on lower levels. It should be noted that some valid occurrences with score above this score threshold can be missed. On higher levels score can be slightly lower than on lower levels. Thus, some valid object occurrences which on the lowest level would be deemed to be valid object instances can be incorrectly missed on some higher level. The **diagMatchPyramid** output represents all potential candidates recognized on each pyramid level and can be helpful during the difficult process of the proper parameter setting.

The **outObject.Point** contains the model reference point of the matched object occurrence. The **outObject.Angle** contains the rotation angle of the object. The **outObject.Match** provides information about both the position and the angle of the found match combined into value of Rectangle2D type. The **outObject.Alignment** contains information about the transform required for geometrical objects defined in the context of template image to be transformed into object in the context of **outObject.Match** position. This value can be later used e.g. by 1D Edge Detection or Shape Fitting categories filters.

Hints

- You can use EnhanceSingleObjectMatch filter to fine-tune the results. A great example of usage is presented in the CreateGoldenTemplate2 filter.
- If an object is not detected, first try decreasing **inMaxPyramidLevel**, then try decreasing **inMinScore**.
- If all the expected objects are correctly detected, try achieving higher performance by increasing **inMaxPyramidLevel** and **inMinScore**.
- Please note, that due to the pyramid strategy used for speeding-up computations, some objects whose score would finally be above **inMinScore** may be not found. This may be surprising, but this is by design. The reason is that the minimum value is used at different pyramid levels and many objects exhibit lower score at higher pyramid levels. They get discarded before they can be evaluated at the lowest pyramid level. Decrease **inMinScore** experimentally until all objects are found.
- If precision of matching is not very important, you can also gain some performance by increasing **inMinPyramidLevel**.
- If the performance is still not satisfactory, go back to model definition and try reducing the range of rotations and scaling as well as the precision-related parameters: **inAnglePrecision**, **inScalePrecision** and **inEdgeCompleteness**.
- To obtain high accuracy of matching make sure that the edges visible on the **diagEdgePyramid** output are not too thick. Increase **inEdgeThreshold** to make them thinner.
- Also consider increasing **inEdgeThreshold** if there is much noise on the **diagEdgePyramid** output.
- Verify accuracy of the matching by checking the **outObjectEdges** output. If the accuracy is lower than expected, make sure that **inMinPyramidLevel** = 0, try increasing **inEdgeThreshold** and also consider higher values for **inAnglePrecision** during model creation.
- To further improve matching accuracy use the EnhanceSingleObjectMatch filter.
- If the object being detected can be darker or brighter than the background on different images, use the **inEdgePolarityMode** parameter set to Ignore.
- If the object being detected is only partially visible or its shape and the model shape are not identical, use the **inEdgePolarityMode** parameter set to MatchWeakly.
- If there is only low noise in the object edges, use the **inEdgeNoiseLevel** parameter set to Low.
- Define **inSearchRegion** to limit possible object locations and speed-up computations. Please note, that this is the region where the **outObject.Point** results belong to (and NOT the region within which the entire object has to be contained).

Examples



*Locating a single object with the edge-based method (**inMaxPyramidLevel** = 3). Background clutter and glares are efficiently ignored.*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

Additional information about Template Matching can be found in Machine Vision Guide: Template Matching

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Unsupported IgnoreGlobally edge polarity mode in LocateSingleObject_Edges1.

See Also

- LocateMultipleObjects_Edges1 – Finds all occurrences of a predefined template on an image by comparing object edges.
- CreateEdgeModel1 – Creates a model for edge-based template matching.
- EnhanceSingleObjectMatch – Improves accuracy of single object matching by adding a subpixel-precise adjustment.

LocateSingleObject_Edges2

Header: AVL.h
Namespace: avl
Module: MatchingPro

Finds a single occurrence of a predefined template on an image by comparing object edges.

Applications: Detection of an object whose outlines are sharp and rigid. Often one of the first filters in a program.

Syntax

```
void avl::LocateSingleObject_Edges2
(
 const avl::Image& inImage,
 atl::Optional<const avl::ShapeRegion&> inSearchRegion,
 atl::Optional<const avl::CoordinateSystem2D&> inSearchRegionAlignment,
 const avl::EdgeModel2& inEdgeModel,
 int inMinPyramidLevel,
 atl::Optional<int> inMaxPyramidLevel,
 float inEdgeThreshold,
 avl::EdgePolarityMode::Type inEdgePolarityMode,
 avl::EdgeNoiseLevel::Type inEdgeNoiseLevel,
 bool inIgnoreBoundaryObjects,
 int inMaxDeformation,
 float inMinScore,
 atl::Conditional<avl::Object2D>& outObject,
 atl::Optional<atl::Conditional<atl::Array<avl::Path>>&> outObjectEdges = atl::NIL,
 atl::Optional<int&> outPyramidHeight = atl::NIL,
 atl::Optional<avl::ShapeRegion&> outAlignedSearchRegion = atl::NIL,
 atl::Array<avl::Image>& diagEdgePyramid,
 atl::Array<avl::Image>& diagMatchPyramid,
 atl::Conditional<atl::Array<float> >& diagScores
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image on which object occurrence will be searched
→	inSearchRegion	Optional<const ShapeRegion&>		NIL	Region of possible object centers
→	inSearchRegionAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the region of interest to the position of the inspected object
→	inEdgeModel	const EdgeModel2&			Model of objects to be searched
→	inMinPyramidLevel	int	0 - 12	0	Defines the lowest pyramid level at which object position is still refined
→	inMaxPyramidLevel	Optional<int>	0 - 12	3	Defines the total number of reduced resolution levels that can be used to speed up computations
→	inEdgeThreshold	float	0.01 - ∞	10.0f	Minimum strength of edges used for matching with the model
→	inEdgePolarityMode	EdgePolarityMode::Type		MatchStrictly	Defines how edges with reversed polarity will contribute to the object score
→	inEdgeNoiseLevel	EdgeNoiseLevel::Type		High	Defines how much noise the object edges have
→	inIgnoreBoundaryObjects	bool		False	Flag indicating whether objects crossing image boundary should be ignored or not
→	inMaxDeformation	int	0 - ∞	0	Maximal distance in pixels between model edge and an edge visible in the input image
→	inMinScore	float	0.0 - 1.0	0.7f	Minimum score of object candidates accepted at each pyramid level
←	outObject	Conditional<Object2D>&			Found object
←	outObjectEdges	Optional<Conditional<Array<Path>>&>		NIL	Model edges of the found object
←	outPyramidHeight	Optional<int&>		NIL	Highest pyramid level used to speed up computations
←	outAlignedSearchRegion	Optional<ShapeRegion&>		NIL	Transformed input shape region
🔍	diagEdgePyramid	Array<Image>&			Image edges used for matching at each pyramid level
🔍	diagMatchPyramid	Array<Image>&			Candidate object locations found at each pyramid level
🔍	diagScores	Conditional<Array<float> >&			Scores of the found object at each pyramid level

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outObjectEdges**, **outPyramidHeight**, **outAlignedSearchRegion**.

Read more about Optional Outputs.

Description

The operation matches the object model, **inEdgeModel**, against the input image, **inImage**. The **inSearchRegion** region restricts the search area so that only in this region the centers of the objects can be presented. The **inMinScore** parameter determines the minimum score of the valid object occurrence.

In the **inImage** every pixel with gradient's magnitude at least **inEdgeThreshold** is considered an edge pixel. Only those are later used during the matching process. To establish the value of this threshold properly the **diagEdgePyramid** output which shows all the edges with sufficient gradient's magnitude may be used. If the **inEdgePolarityMode** parameter is set to Ignore, the object occurrences in the **inImage** image do not have necessarily to have the same contrast as the object in the model image.

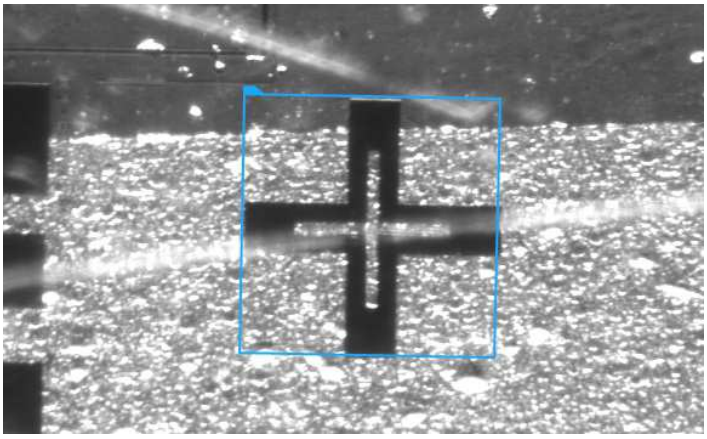
The computation time of the filter depends on the size of the model, the sizes of **inImage** and **inSearchRegion**, but also on the value of **inMinScore**. This parameter is a score threshold. Based on its value some partial computation can be sufficient to reject some locations as valid object instances. Moreover, the pyramid of the images is used. Thus, only the highest pyramid level is searched exhaustively, and potential candidates are later validated at lower levels. The **inMinPyramidLevel** parameter determines the lowest pyramid level used to validate such candidates. Setting this parameter to a value greater than 0 may speed up the computation significantly, especially for higher resolution images. However, the accuracy of the found object occurrences can be reduced. Larger **inMinScore** generates less potential candidates on the highest level to verify on lower levels. It should be noted that some valid occurrences with score above this score threshold can be missed. On higher levels score can be slightly lower than on lower levels. Thus, some valid object occurrences which on the lowest level would be deemed to be valid object instances can be incorrectly missed on some higher level. The **diagMatchPyramid** output represents all potential candidates recognized on each pyramid level and can be helpful during the difficult process of the proper parameter setting.

The **outObject.Point** contains the model reference point of the matched object occurrence. The **outObject.Angle** contains the rotation angle of the object. The **outObject.Match** provides information about both the position and the angle of the found match combined into value of Rectangle2D type. The **outObject.Alignment** contains information about the transform required for geometrical objects defined in the context of template image to be transformed into object in the context of **outObject.Match** position. This value can be later used e.g. by 1D Edge Detection or Shape Fitting categories filters.

Hints

- You can use EnhanceSingleObjectMatch filter to fine-tune the results. A great example of usage is presented in the CreateGoldenTemplate2 filter.
- If an object is not detected, first try decreasing **inMaxPyramidLevel**, then try decreasing **inMinScore**.
- If all the expected objects are correctly detected, try achieving higher performance by increasing **inMaxPyramidLevel** and **inMinScore**.
- Please note, that due to the pyramid strategy used for speeding-up computations, some objects whose score would finally be above **inMinScore** may be not found. This may be surprising, but this is by design. The reason is that the minimum value is used at different pyramid levels and many objects exhibit lower score at higher pyramid levels. They get discarded before they can be evaluated at the lowest pyramid level. Decrease **inMinScore** experimentally until all objects are found.
- If precision of matching is not very important, you can also gain some performance by increasing **inMinPyramidLevel**.
- If the performance is still not satisfactory, go back to model definition and try reducing the range of rotations and scaling as well as the precision-related parameters: **inAnglePrecision**, **inScalePrecision** and **inEdgeCompleteness**.
- To obtain high accuracy of matching make sure that the edges visible on the **diagEdgePyramid** output are not too thick. Increase **inEdgeThreshold** to make them thinner.
- Also consider increasing **inEdgeThreshold** if there is much noise on the **diagEdgePyramid** output.
- Verify accuracy of the matching by checking the **outObjectEdges** output. If the accuracy is lower than expected, make sure that **inMinPyramidLevel** = 0, try increasing **inEdgeThreshold** and also consider higher values for **inAnglePrecision** during model creation.
- To further improve matching accuracy use the EnhanceSingleObjectMatch filter.
- If the object being detected can be darker or brighter than the background on different images, use the **inEdgePolarityMode** parameter set to Ignore.
- If the object being detected is only partially visible or its shape and the model shape are not identical, use the **inEdgePolarityMode** parameter set to MatchWeakly.
- If there is only low noise in the object edges, use the **inEdgeNoiseLevel** parameter set to Low.
- Define **inSearchRegion** to limit possible object locations and speed-up computations. Please note, that this is the region where the **outObject.Point** results belong to (and NOT the region within which the entire object has to be contained).

Examples



*Locating a single object with the edge-based method (**inMaxPyramidLevel** = 3). Background clutter and glares are efficiently ignored.*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

Additional information about Template Matching can be found in Machine Vision Guide: Template Matching

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- LocateMultipleObjects_Edges2 – Finds all occurrences of a predefined template on an image by comparing object edges.
- CreateEdgeModel2 – Creates a model for edge-based template matching.
- EnhanceSingleObjectMatch – Improves accuracy of single object matching by adding a subpixel-precise adjustment.

LocateSingleObject_NCC

Header: AVL.h
Namespace: avl
Module: MatchingBasic

Finds a single occurrence of a predefined template on an image by analysing the normalized correlation between pixel values.

Applications: Detection of an object with blurred or unclear edges. Often one of the first filters in a program.

Syntax

```
void avl::LocateSingleObject_NCC
(
 const avl::Image& inImage,
 atl::Optional<const avl::ShapeRegion&> inSearchRegion,
 atl::Optional<const avl::CoordinateSystem2D&> inSearchRegionAlignment,
 const avl::GrayModel& inGrayModel,
 int inMinPyramidLevel,
 atl::Optional<int> inMaxPyramidLevel,
 bool inIgnoreBoundaryObjects,
 float inMinScore,
 atl::Optional<float> inMaxBrightnessRatio,
 atl::Optional<float> inMaxContrastRatio,
 atl::Conditional<avl::Object2D>& outObject,
 atl::Optional<int&> outPyramidHeight = atl::NIL,
 atl::Optional<avl::ShapeRegion&> outAlignedSearchRegion = atl::NIL,
 atl::Array<avl::Image>& diagImagePyramid,
 atl::Array<avl::Image>& diagMatchPyramid,
 atl::Conditional<atl::Array<float> >& diagScores
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image on which object occurrence will be searched
→	inSearchRegion	Optional<const ShapeRegion&>		NIL	Range of possible object centers
→	inSearchRegionAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the region of interest to the position of the inspected object
→	inGrayModel	const GrayModel&			Model of objects to be searched
→	inMinPyramidLevel	int	0 - 12	0	Defines the lowest pyramid level at which object position is still refined
→	inMaxPyramidLevel	Optional<int>	0 - 12	3	Defines the total number of reduced resolution levels that can be used to speed up computations
→	inIgnoreBoundaryObjects	bool		False	Flag indicating whether objects crossing image boundary should be ignored or not
→	inMinScore	float	-1.0 - 1.0	0.7f	Minimum score of object candidates accepted at each pyramid level
→	inMaxBrightnessRatio	Optional<float>	1.0 - ∞	NIL	Defines the maximal deviation of the mean brightness of the model object and the object present in the image
→	inMaxContrastRatio	Optional<float>	1.0 - ∞	NIL	Defines the maximal deviation of the brightness standard deviation of the model object and the object present in the image
←	outObject	Conditional<Object2D>&			Found object
←	outPyramidHeight	Optional<int&>		NIL	Highest pyramid level used to speed up computations
←	outAlignedSearchRegion	Optional<ShapeRegion&>		NIL	Transformed input shape region
🔍	diagImagePyramid	Array<Image>&			Pyramid of iteratively downsampled input image
🔍	diagMatchPyramid	Array<Image>&			Candidate object locations found at each pyramid level
🔍	diagScores	Conditional<Array<float> >&			Scores of the found object at each pyramid level

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outPyramidHeight**, **outAlignedSearchRegion**.

Read more about Optional Outputs.

Description

The operation matches the object model, **inGrayModel**, against the input image, **inImage**. The **inSearchRegion** region restricts the search area so that only in this region the centers of the objects can be presented. The **inMinScore** parameter determines the minimum score of the valid object occurrence.

The computation time of the filter depends on the size of the model, the sizes of **inImage** and **inSearchRegion**, but also on the value of **inMinScore**. This parameter is a score threshold. Based on its value some partial computation can be sufficient to reject some locations as valid object instances. Moreover, the pyramid of the images is used. Thus, only the highest pyramid level is searched exhaustively, and potential candidates are later validated at lower levels. The **inMinPyramidLevel** parameter determines the lowest pyramid level used to validate such candidates. Setting this parameter to a value greater than 0 may speed up the computation significantly, especially for higher resolution images. However, the accuracy of the found object occurrences can be reduced. Larger **inMinScore** generates less potential candidates on the highest level to verify on lower levels. It should be noted that some valid occurrences with score above this score threshold can be missed. On higher levels score can be slightly lower than on lower levels. Thus, some valid object occurrences which on the lowest level would be deemed to be valid object instances can be incorrectly missed on some higher level. The **diagMatchPyramid** output represents all potential candidates recognized on each pyramid level and can be helpful during the difficult process of the proper parameter setting.

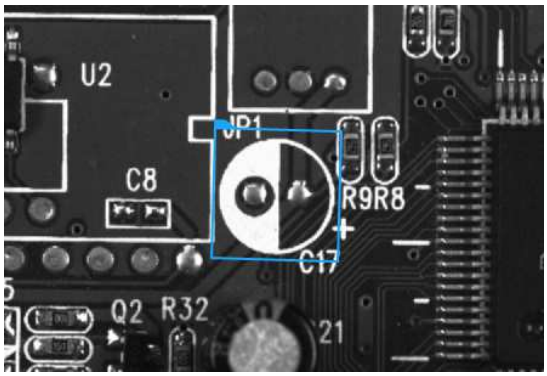
To be able to locate objects which are partially outside the image, the filter assumes that there are only black pixels beyond the image border.

The **outObject.Point** contains the model reference point of the matched object occurrence. The **outObject.Angle** contains the rotation angle of the object. The **outObject.Match** provides information about both the position and the angle of the found match combined into value of Rectangle2D type. The **outObject.Alignment** contains information about the transform required for geometrical objects defined in the context of template image to be transformed into object in the context of **outObject.Match** position. This value can be later used e.g. by 1D Edge Detection or Shape Fitting categories filters.

Hints

- If an object is not detected, first try decreasing **inMaxPyramidLevel**, then try decreasing **inMinScore**. However, if you need to lower **inMinScore** below 0.5, then probably something else is wrong.
- If all the expected objects are correctly detected, try achieving higher performance by increasing **inMaxPyramidLevel** and **inMinScore**.
- Please note, that due to the pyramid strategy used for speeding-up computations, some objects whose score would finally be above **inMinScore** may be not found. This may be surprising, but this is by design. The reason is that the minimum value is used at different pyramid levels and many objects exhibit lower score at higher pyramid levels. They get discarded before they can be evaluated at the lowest pyramid level. Decrease **inMinScore** experimentally until all objects are found.
- If precision of matching is not very important, you can also gain some performance by increasing **inMinPyramidLevel**.
- If the performance is still not satisfactory, go back to model definition and try reducing the range of rotations and scaling as well as the precision-related parameters: **inAnglePrecision** and **inScalePrecision**.
- Define **inSearchRegion** to limit possible object locations and speed-up computations. Please note, that this is the region where the **outObject.Point** results belong to (and NOT the region within which the entire object has to be contained).
- Smoothing of the input image can significantly improve performance of this tool.

Examples



*Locating single object with the gray-based method (**inMaxPyramidLevel** = 4).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

Additional information about Template Matching can be found in Machine Vision Guide: Template Matching

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of type: UINT8.

This operation is optimized for AVX2 technology for pixels of type: UINT8.

This operation is optimized for NEON technology for pixels of type: UINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- LocateMultipleObjects_NCC – Finds all occurrences of a predefined template on an image by analysing the normalized correlation between pixel values.
- CreateGrayModel – Creates a model for NCC or SAD template matching.
- LocateSingleObject_SAD – Finds a single occurrence of a predefined template on an image by analysing the Square Average Difference between pixel values.

LocateSingleObject_NCC2

Header: AVL.h
Namespace: avl
Module: MatchingBasic

Finds a single occurrence of a predefined template on an image by analysing the normalized correlation between pixel values.

Applications: Detection of objects with blurred or unclear edges. Often one of the first filters in a program.

Syntax

```
void avl::LocateSingleObject_NCC2
(
 const avl::Image& inImage,
 atl::Optional<const avl::ShapeRegion&> inSearchRegion,
 atl::Optional<const avl::CoordinateSystem2D&> inSearchRegionAlignment,
 const avl::GrayModel2& inGrayModel,
 int inMinPyramidLevel,
 atl::Optional<int> inMaxPyramidLevel,
 bool inIgnoreBoundaryObjects,
 float inMinScore,
 atl::Optional<float> inMaxBrightnessRatio,
 atl::Optional<float> inMaxContrastRatio,
 atl::Conditional<avl::Object2D>& outObject,
 atl::Optional<int&> outPyramidHeight = atl::NIL,
 atl::Optional<avl::ShapeRegion&> outAlignedSearchRegion = atl::NIL,
 atl::Array<avl::Image>& diagImagePyramid = atl::Dummy<atl::Array<avl::Image>>(),
 atl::Array<avl::Image>& diagMatchPyramid = atl::Dummy<atl::Array<avl::Image>>(),
 atl::Conditional<atl::Array<float>>& diagScores = atl::Dummy<atl::Conditional<atl::Array<float>>>()
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image on which model occurrences will be searched
→	inSearchRegion	Optional<const ShapeRegion&>		NIL	Range of possible object centers
→	inSearchRegionAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the region of interest to the position of the inspected object
→	inGrayModel	const GrayModel2&			Model of objects to be searched
→	inMinPyramidLevel	int	0 - 12	0	Defines the lowest pyramid level at which object position is still refined
→	inMaxPyramidLevel	Optional<int>	0 - 12	3	Defines the total number of reduced resolution levels that can be used to speed up computations
→	inIgnoreBoundaryObjects	bool		False	Flag indicating whether objects crossing image boundary should be ignored or not
→	inMinScore	float	-1.0 - 1.0	0.7f	Minimum score of object candidates accepted at each pyramid level
→	inMaxBrightnessRatio	Optional<float>	1.0 - ∞	NIL	Defines the maximal deviation of the mean brightness of the model object and the object present in the image
→	inMaxContrastRatio	Optional<float>	1.0 - ∞	NIL	Defines the maximal deviation of the brightness standard deviation of the model object and the object present in the image
←	outObject	Conditional<Object2D>&			Found object
←	outPyramidHeight	Optional<int&>		NIL	Highest pyramid level used to speed up computations
←	outAlignedSearchRegion	Optional<ShapeRegion&>		NIL	Transformed input shape region
🔍	diagImagePyramid	Array<Image>&			Pyramid of iteratively downsampled input image
🔍	diagMatchPyramid	Array<Image>&			Candidate object locations found at each pyramid level
🔍	diagScores	Conditional<Array<float>>&			Scores of the found object at each pyramid level

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outPyramidHeight**, **outAlignedSearchRegion**.

Read more about Optional Outputs.

Description

The operation matches the object model, **inGrayModel**, against the input image, **inImage**. The **inSearchRegion** region restricts the search area so that only in this region the centers of the objects can be presented. The **inMinScore** parameter determines the minimum score of the valid object occurrence.

The computation time of the filter depends on the size of the model, the sizes of **inImage** and **inSearchRegion**, but also on the value of **inMinScore**. This parameter is a score threshold. Based on its value some partial computation can be sufficient to reject some locations as valid object instances. Moreover, the pyramid of the images is used. Thus, only the highest pyramid level is searched exhaustively, and potential candidates are later validated at lower levels. The **inMinPyramidLevel** parameter determines the lowest pyramid level used to validate such candidates. Setting this parameter to a value greater than 0 may speed up the computation significantly, especially for higher resolution images. However, the accuracy of the found object occurrences can be reduced. Larger **inMinScore** generates less potential candidates on the highest level to verify on lower levels. It should be noted that some valid occurrences with score above this score threshold can be missed. On higher levels score can be slightly lower than on lower levels. Thus, some valid object occurrences which on the lowest level would be deemed to be valid object instances can be incorrectly missed on some higher level. The **diagMatchPyramid** output represents all potential candidates recognized on each pyramid level and can be helpful during the difficult process of the proper parameter setting.

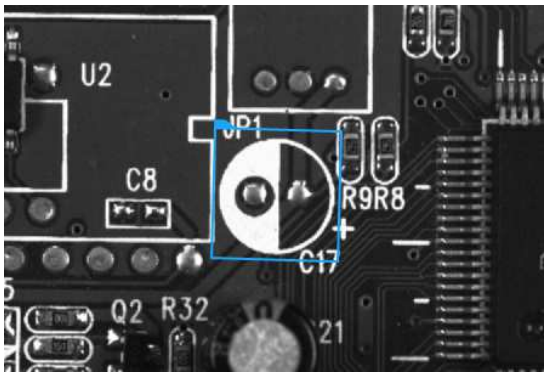
To be able to locate objects which are partially outside the image, the filter assumes that there are only black pixels beyond the image border.

The **outObject.Point** contains the model reference point of the matched object occurrence. The **outObject.Angle** contains the rotation angle of the object. The **outObject.Match** provides information about both the position and the angle of the found match combined into value of Rectangle2D type. The **outObject.Alignment** contains information about the transform required for geometrical objects defined in the context of template image to be transformed into object in the context of **outObject.Match** position. This value can be later used e.g. by 1D Edge Detection or Shape Fitting categories filters.

Hints

- If an object is not detected, first try decreasing **inMaxPyramidLevel**, then try decreasing **inMinScore**. However, if you need to lower **inMinScore** below 0.5, then probably something else is wrong.
- If all the expected objects are correctly detected, try achieving higher performance by increasing **inMaxPyramidLevel** and **inMinScore**.
- Please note, that due to the pyramid strategy used for speeding-up computations, some objects whose score would finally be above **inMinScore** may be not found. This may be surprising, but this is by design. The reason is that the minimum value is used at different pyramid levels and many objects exhibit lower score at higher pyramid levels. They get discarded before they can be evaluated at the lowest pyramid level. Decrease **inMinScore** experimentally until all objects are found.
- If precision of matching is not very important, you can also gain some performance by increasing **inMinPyramidLevel**.
- If the performance is still not satisfactory, go back to model definition and try reducing the range of rotations and scaling as well as the precision-related parameters: **inAnglePrecision** and **inScalePrecision**.
- Define **inSearchRegion** to limit possible object locations and speed-up computations. Please note, that this is the region where the **outObject.Point** results belong to (and NOT the region within which the entire object has to be contained).
- Smoothing of the input image can significantly improve performance of this tool.

Examples



*Locating single object with the gray-based method (**inMaxPyramidLevel** = 4).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

Additional information about Template Matching can be found in Machine Vision Guide: Template Matching

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of type: UINT8.

This operation is optimized for AVX2 technology for pixels of type: UINT8.

This operation is optimized for NEON technology for pixels of type: UINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Invalid GrayModel2 in LocateSingleObject_NCC2. Create a proper model first using the Template Matching plugin or CreateGrayModel2 filter.

See Also

- LocateMultipleObjects_NCC2 – Finds all occurrences of a predefined template on an image by analysing the normalized correlation between pixel values.
- CreateGrayModel2 – Creates a model for NCC or SAD template matching.

LocateSingleObject_SAD

Header: AVL.h
Namespace: avl
Module: MatchingBasic

Finds a single occurrence of a predefined template on an image by analysing the Square Average Difference between pixel values.

Applications: Almost always inferior to NCC, so rarely used in real applications.

Syntax

```
void avl::LocateSingleObject_SAD
(
 const avl::Image& inImage,
 atl::Optional<const avl::ShapeRegion&> inSearchRegion,
 atl::Optional<const avl::CoordinateSystem2D&> inSearchRegionAlignment,
 const avl::GrayModel& inGrayModel,
 int inMinPyramidLevel,
 atl::Optional<int> inMaxPyramidLevel,
 bool inIgnoreBoundaryObjects,
 float inMaxDifference,
 atl::Conditional<avl::Object2D>& outObject,
 atl::Optional<int&> outPyramidHeight = atl::NIL,
 atl::Optional<avl::ShapeRegion&> outAlignedSearchRegion = atl::NIL,
 atl::Array<avl::Image>& diagImagePyramid,
 atl::Array<avl::Image>& diagMatchPyramid,
 atl::Conditional<atl::Array<float>>& diagScores
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image on which model occurrence will be searched
→	inSearchRegion	Optional<const ShapeRegion&>		NIL	Possible centers of the object occurrence
→	inSearchRegionAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the region of interest to the position of the inspected object
→	inGrayModel	const GrayModel&			Model which will be sought
→	inMinPyramidLevel	int	0 - 12	0	Defines the highest resolution level
→	inMaxPyramidLevel	Optional<int>	0 - 12	3	Defines the number of reduced resolution levels that can be used to speed up computations
→	inIgnoreBoundaryObjects	bool		False	Flag indicating whether objects crossing image boundary should be ignored or not
→	inMaxDifference	float	0.0 - ∞	0.0f	Maximum accepted average difference between pixel values
←	outObject	Conditional<Object2D>&			Found object
←	outPyramidHeight	Optional<int&>		NIL	Highest pyramid level used to speed up computations
←	outAlignedSearchRegion	Optional<ShapeRegion&>		NIL	Transformed input shape region
🔍	diagImagePyramid	Array<Image>&			Pyramid of iteratively downsampled input image
🔍	diagMatchPyramid	Array<Image>&			Locations found on each pyramid level
🔍	diagScores	Conditional<Array<float>>&			Scores of found match on each pyramid level

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outPyramidHeight**, **outAlignedSearchRegion**.

Read more about Optional Outputs.

Description

The operation matches the object model, **inGrayModel**, against the input image, **inImage**. The **inSearchRegion** region restricts the search area so that only in this region the centers of the objects can be presented. The **inMaxDifference** parameter determines the maximum average difference between corresponding pixel values of the valid object occurrence.

The computation time of the filter depends on the size of the model, the sizes of **inImage** and **inSearchRegion**, but also on the value of **inMaxDifference**. This parameter is a score threshold. Based on its value some partial computation can be sufficient to reject some locations as valid object instances. Moreover, the pyramid of the images is used. Thus, only the highest pyramid level is searched exhaustively, and potential candidates are later validated at lower levels. The **inMinPyramidLevel** parameter determines the lowest pyramid level used to validate such candidates. Setting this parameter to a value greater than 0 may speed up the computation significantly, especially for higher resolution images. However, the accuracy of the found object occurrences can be reduced. Lower **inMaxDifference** generates less potential candidates on the highest level to verify on lower levels. It should be noted that some valid occurrences with score above this score threshold can be missed. On higher levels score can be slightly lower than on lower levels. Thus, some valid object occurrences which on the lowest level would be deemed to be valid object instances can be incorrectly missed on some higher level. The **diagMatchPyramid** output represents all potential candidates recognized on each pyramid level and can be helpful during the difficult process of the proper parameter setting.

To be able to locate objects which are partially outside the image, the filter assumes that there are only black pixels beyond the image border.

The **outObject.Point** contains the model reference point of the matched object occurrence. The **outObject.Angle** contains the rotation angle of the object. The **outObject.Match** provides information about both the position and the angle of the found match combined into value of Rectangle2D type. The **outObject.Alignment** contains information about the transform required for geometrical objects defined in the context of template image to be transformed into object in the context of **outObject.Match** position. This value can be later used e.g. by 1D Edge Detection or Shape Fitting categories filters.

The SAD (Sum of Absolute Differences) method can be significantly slower than NCC (Normalized Cross-Correlation) method. Moreover, it is not illumination-invariant, as it is required in most applications. Thus, it is highly recommended to use the latter, NCC method instead.

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

Additional information about Template Matching can be found in Machine Vision Guide: Template Matching

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- LocateMultipleObjects_SAD – Finds multiple occurrences of a predefined template on an image by analysing the Square Average Difference between pixel values.
- CreateGrayModel – Creates a model for NCC or SAD template matching.
- LocateSingleObject_NCC – Finds a single occurrence of a predefined template on an image by analysing the normalized correlation between pixel values.

MergeMultipleLocationResults

Header: AVL.h
Namespace: avl
Module: MatchingBasic

Combines results from multiple LocateMultipleObject instances.

Syntax

```
void avl::MergeMultipleLocationResults
(
 const atl::Array<avl::Object2D>& inObjects1,
 const atl::Array<avl::Object2D>& inObjects2,
 const atl::Array<avl::Object2D>& inObjects3,
 const atl::Array<avl::Object2D>& inObjects4,
 float inMinDistance,
 float inMinScore,
 atl::Array<avl::Object2D>& outObjects,
 atl::Array<int>& outIndices
)
```

Parameters

	Name	Type	Range	Default	Description
→	inObjects1	const Array<Object2D>&			
→	inObjects2	const Array<Object2D>&			
→	inObjects3	const Array<Object2D>&			
→	inObjects4	const Array<Object2D>&			
→	inMinDistance	float	0.0 - ∞	10.0f	
→	inMinScore	float	-1.0 - 1.0	0.0f	
←	outObjects	Array<Object2D>&			
←	outIndices	Array<int>&			

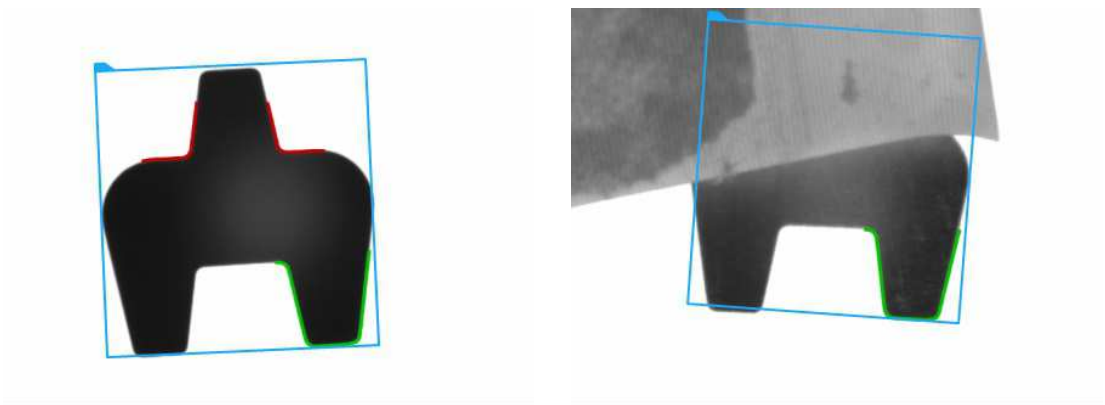
Description

Filter selects the best matching object from the given matching objects.

This filter is especially useful for finding big objects by using smaller template models instead of the large model.

This filter will return a valid object location even if object is partially covered.

Examples



Object location found based on two matching objects(red and green).

Remarks

Each template matching object which is used in filter must have set this same **reference frame**.

See Also

- LocateMultipleObjects_Edges1 – Finds all occurrences of a predefined template on an image by comparing object edges.
- LocateSingleObject_Edges1 – Finds a single occurrence of a predefined template on an image by comparing object edges.

MergeSingleLocationResults

Header: AVL.h
Namespace: avl
Module: MatchingBasic

Combines results from multiple LocateSingleObject instances.

Syntax

```
void avl::MergeSingleLocationResults
(
 const atl::Conditional<avl::Object2D>& inObject1,
 const atl::Conditional<avl::Object2D>& inObject2,
 const atl::Conditional<avl::Object2D>& inObject3,
 const atl::Conditional<avl::Object2D>& inObject4,
 float inMinScore,
 atl::Conditional<avl::Object2D>& outObject,
 atl::Conditional<int>& outIndex
)
```

Parameters

	Name	Type	Range	Default	Description
→	inObject1	const Conditional<Object2D>&			
→	inObject2	const Conditional<Object2D>&			
→	inObject3	const Conditional<Object2D>&			
→	inObject4	const Conditional<Object2D>&			
→	inMinScore	float	-1.0 - 1.0	0.0f	
←	outObject	Conditional<Object2D>&			
←	outIndex	Conditional<int>&			

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inObject1** and **outObject**, **inObject2** and **outObject**, **inObject3** and **outObject**, **inObject4** and **outObject**

Read more about In-place Computation.

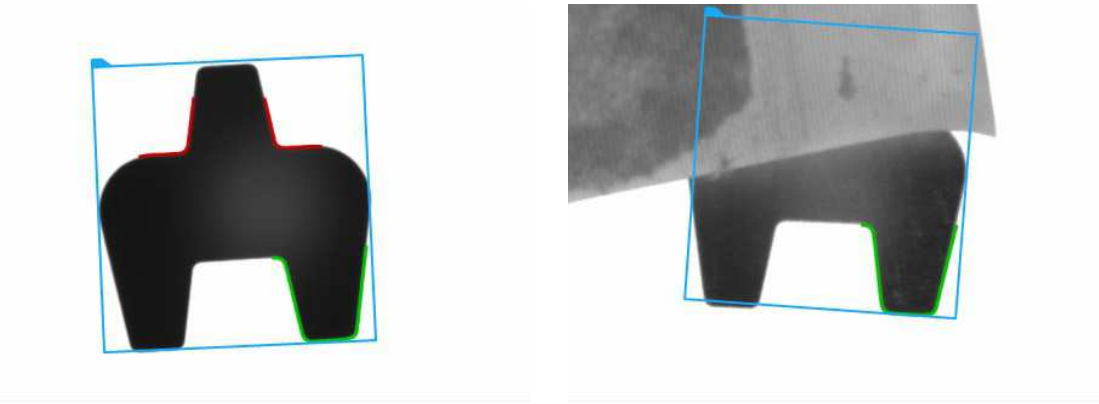
Description

Filter selects the best matching object from the given matching objects.

This filter is especially useful for finding big objects by using smaller template models instead of the large model.

This filter will return a valid object location even if object is partially covered.

Examples



Object location found based on two matching objects(red and green).

Remarks

Each template matching object which is used in filter must have set this same **reference frame**.

See Also

- LocateMultipleObjects_Edges1 – Finds all occurrences of a predefined template on an image by comparing object edges.
- LocateSingleObject_Edges1 – Finds a single occurrence of a predefined template on an image by comparing object edges.

SaveEdgeModel

Header: AVL.h
Namespace: avl
Module: MatchingPro

Saves serialized template matching EdgeModel object as AVDATA file.

Syntax

```
void avl::SaveEdgeModel
(
 const avl::EdgeModel& inEdgeModel,
 const atl::File& inFilename
)
```

Parameters

	Name	Type	Default	Description
➡	inEdgeModel	const EdgeModel&		Model to be serialized
➡	inFilename	const File&		Name of the target file

SaveEdgeModel2

Header: AVL.h
Namespace: avl
Module: MatchingPro

Saves serialized template matching EdgeModel2 object as AVDATA file.

Syntax

```
void avl::SaveEdgeModel2
(
 const avl::EdgeModel2& inEdgeModel,
 const atl::File& inFilename
)
```

Parameters

	Name	Type	Default	Description
➡	inEdgeModel	const EdgeModel2&		Model to be serialized
➡	inFilename	const File&		Name of the target file

SaveGrayModel

Header: AVL.h
Namespace: avl
Module: MatchingBasic

Saves serialized template matching GrayModel object as AVDATA file.

Syntax

```
void avl::SaveGrayModel
(
 const avl::GrayModel& inGrayModel,
 const atl::File& inFilename
)
```

Parameters

	Name	Type	Default	Description
➡	inGrayModel	const GrayModel&		Model to be serialized
➡	inFilename	const File&		Name of the target file

SaveGrayModel2

Header: AVL.h
Namespace: avl
Module: MatchingBasic

Saves serialized template matching GrayModel2 object as AVDATA file.

Syntax

```
void avl::SaveGrayModel2
(
 const avl::GrayModel2& inGrayModel,
 const atl::File& inFilename
)
```

Parameters

	Name	Type	Default	Description
➡	inGrayModel	const GrayModel2&		Model to be serialized
➡	inFilename	const File&		Name of the target file

11. Image Analysis

Table of content:

CheckPresence_EdgeAmount

Header: AVL.h
Namespace: avl
Module: FoundationPro

Verifies object presence by analysing the amount of edges in the specified region.

Applications: Quick and easy presence verification, e.g. for missing caps, screws, labels.

Syntax

```
void avl::CheckPresence_EdgeAmount
(
 const avl::Image& inImage,
 const avl::ShapeRegion& inRoi,
 atl::Optional<const avl::CoordinateSystem2D&> inRoiAlignment,
 avl::GradientMaskOperator::Type inEdgeOperator,
 avl::MagnitudeMeasure::Type inEdgeMeasure,
 const int inEdgeScale,
 int inMinStrength,
 float inMinAmount,
 float inMaxAmount,
 bool& outIsPresent,
 atl::Optional<float&> outAmount = atl::NIL,
 atl::Optional<avl::Region&> outForeground = atl::NIL,
 atl::Optional<avl::ShapeRegion&> outAlignedRoi = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	const ShapeRegion&			Location at which object presence is being checked
→	inRoiAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the region of interest to the position of the inspected object
→	inEdgeOperator	GradientMaskOperator::Type			Selected gradient operator
→	inEdgeMeasure	MagnitudeMeasure::Type		Sum	Selected method of gradient magnitude computation
→	inEdgeScale	const int	1 - 16	1	Scales the resulting gradient magnitudes
→	inMinStrength	int	0 - 255	15	Lowest acceptable edge magnitude
→	inMinAmount	float	0.0 - 1.0	0.2f	Lowest acceptable fraction of pixels meeting the criteria
→	inMaxAmount	float	0.0 - 1.0	1.0f	Highest acceptable fraction of pixels meeting the criteria
←	outIsPresent	bool&			Flag indicating whether the object is present or not
←	outAmount	Optional<float&>		NIL	Fraction of pixels from meeting the criteria
←	outForeground	Optional<Region&>		NIL	Region of pixels meeting the criteria
←	outAlignedRoi	Optional<ShapeRegion&>		NIL	Input ROI after transformation (in the image coordinates)

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAmount**, **outForeground**, **outAlignedRoi**.

Read more about Optional Outputs.

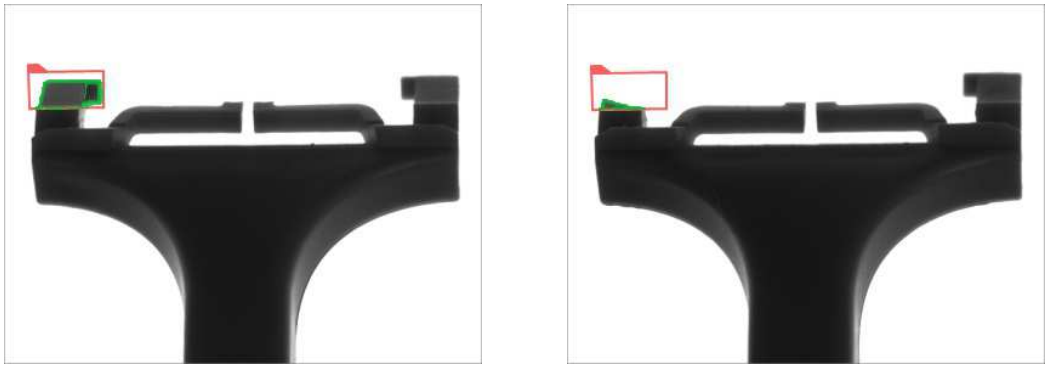
Description

The filter extracts foreground pixels and checks if their number comparing to the area of the whole ROI fits the range (**inMinAmount**, **inMaxAmount**). The pixel is considered a foreground pixel if and only if its gradient magnitude is at least **inMinStrength**.

Hints

- If the object location is variable, Pass an appropriate local coordinate system to **inRoiAlignment**.
- Define **inRoi** to specify the image location at which the object presence will be checked.
- Set **inMinStrength** to a value that results in good edges visible on the **outForeground** output.
- Investigate the values that appear on the **outAmount** output, then set **inMinAmount** and **inMaxAmount** to values appropriate for correct objects.
- When creating data previews, use **outAlignedRoi** and NOT **inRoi** as only the former will be properly aligned to the object location.

Examples



*CheckPresence_EdgeAmount performed on sample images with **inMinAmount** = 0.2. In the left image the object is present, while in the right image it is not. Green pixels are foreground pixels.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in CheckPresence_EdgeAmount.

See Also

- CheckPresence_Intensity – Verifies object presence by analysing pixel intensities in the specified region.
- CheckPresence_EdgeAmount – Verifies object presence by analysing the amount of edges in the specified region.
- ThresholdToRegion_HSx – Creates a region containing image pixels which belongs to specified region in HSV, HSL or HSI space.
- ThresholdToRegion – Creates a region containing image pixels with values within the specified range.

CheckPresence_Intensity

Header: AVL.h
Namespace: avl
Module: FoundationPro

Verifies object presence by analysing pixel intensities in the specified region.

Applications: Quick and easy presence verification, e.g. for missing caps, screws, labels.

Syntax

```
void avl::CheckPresence_Intensity
(
 const avl::Image& inImage,
 const avl::ShapeRegion& inRoi,
 atl::Optional<const avl::CoordinateSystem2D&> inRoiAlignment,
 atl::Optional<float> inMinIntensity,
 atl::Optional<float> inMaxIntensity,
 float inMinContrast,
 atl::Optional<float> inMaxContrast,
 bool& outIsPresent,
 atl::Optional<float&> outIntensity = atl::NIL,
 atl::Optional<float&> outContrast = atl::NIL,
 atl::Optional<avl::ShapeRegion&> outAlignedRoi = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	const ShapeRegion&			Location at which object presence is being checked
→	inRoiAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the region of interest to the position of the inspected object
→	inMinIntensity	Optional<float>		NIL	Lowest acceptable value for the average pixel value
→	inMaxIntensity	Optional<float>		NIL	Highest acceptable value for the average pixel value
→	inMinContrast	float	0.0 - ∞		Lowest acceptable value for the standard deviation of the pixel values
→	inMaxContrast	Optional<float>	0.0 - ∞	NIL	Highest acceptable value for the standard deviation of the pixel values
←	outIsPresent	bool&			Flag indicating whether the object is present or not
←	outIntensity	Optional<float&>		NIL	Average pixel value
←	outContrast	Optional<float&>		NIL	Standard deviation of the pixel values
←	outAlignedRoi	Optional<ShapeRegion&>		NIL	Input ROI after transformation (in the image coordinates)

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outIntensity**, **outContrast**, **outAlignedRoi**.

Read more about Optional Outputs.

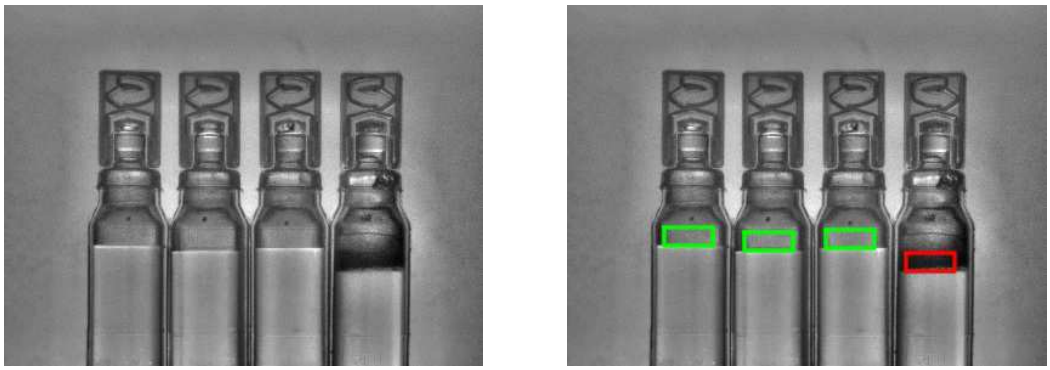
Description

The filter computes basic statistics of the image pixels in selected ROI and checks if they fit the defined ranges. The used statistics are average and standard deviation of the pixel values.

Hints

- If the object location is variable, Pass an appropriate local coordinate system to **inRoiAlignment**.
- Define **inRoi** to specify the image location at which the object presence will be checked.
- Investigate the values that appear on **outIntensity** and **outContrast** outputs. With this information set **inMinIntensity**, **inMaxIntensity**, **inMinContrast** and **inMaxContrast** to values which are appropriate for correct objects.
- When creating data previews, use **outAlignedRoi** and NOT **inRoi** as only the former will be properly aligned to the object location.

Examples



*CheckPresence_Intensity performed on sample image with **inMinIntensity** = 80. The foam is present in red rectangle.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in CheckPresence_Intensity.

See Also

- CheckPresence_PixelAmount – Verifies object presence by analysing the amount of pixels that meet the specified criteria.
- CheckPresence_EdgeAmount – Verifies object presence by analysing the amount of edges in the specified region.
- ImageAverage – Computes the average of the image pixel values.

CheckPresence_PixelAmount

Header: AVL.h
Namespace: avl
Module: FoundationPro

Verifies object presence by analysing the amount of pixels that meet the specified criteria.

Applications: Quick and easy presence verification, e.g. for missing caps, screws, labels.

Syntax

```
void avl::CheckPresence_PixelAmount
(
 const avl::Image& inImage,
 const avl::ShapeRegion& inRoi,
 atl::Optional<const avl::CoordinateSystem2D&> inRoiAlignment,
 avl::HSxColorModel::Type inColorModel,
 int inBeginHue,
 int inEndHue,
 int inMinSaturation,
 atl::Optional<int> inMaxSaturation,
 atl::Optional<float> inMinBrightness,
 atl::Optional<float> inMaxBrightness,
 float inMinAmount,
 float inMaxAmount,
 bool& outIsPresent,
 atl::Optional<float&> outAmount = atl::NIL,
 atl::Optional<avl::Region&> outForeground = atl::NIL,
 atl::Optional<avl::ShapeRegion&> outAlignedRoi = atl::NIL,
 avl::Image& diagHsxImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	const ShapeRegion&			Location at which object presence is being checked
→	inRoiAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the region of interest to the position of the inspected object
→	inColorModel	HSxColorModel::Type			Selected color model
→	inBeginHue	int	0 - 255	0	Begin of the range of acceptable hue
→	inEndHue	int	0 - 255	255	End of the range of acceptable hue
→	inMinSaturation	int	0 - 255	128	Lowest acceptable saturation
→	inMaxSaturation	Optional<int>	0 - 255	NIL	Highest acceptable saturation
→	inMinBrightness	Optional<float>	0.0 - ∞	128.0f	Lowest acceptable brightness
→	inMaxBrightness	Optional<float>	0.0 - ∞	NIL	Highest acceptable brightness
→	inMinAmount	float	0.0 - 1.0	0.5f	Lowest acceptable fraction of pixels meeting the criteria
→	inMaxAmount	float	0.0 - 1.0	1.0f	Highest acceptable fraction of pixels meeting the criteria
←	outIsPresent	bool&			Flag indicating whether the object is present or not
←	outAmount	Optional<float&>		NIL	Fraction of pixels meeting the criteria
←	outForeground	Optional<Region&>		NIL	Region of pixels meeting the criteria
←	outAlignedRoi	Optional<ShapeRegion&>		NIL	Input ROI after transformation (in the image coordinates)
🔍	diagHsxImage	Image&			Image represented in chosen color model

Requirements

For input **inImage** only pixel formats are supported: 3□uint8, 1□uint8, 1□int8, 1□uint16, 1□int16, 1□int32, 1□real.

Read more about pixel formats in Image documentation.

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAmount**, **outForeground**, **outAlignedRoi**.

Read more about Optional Outputs.

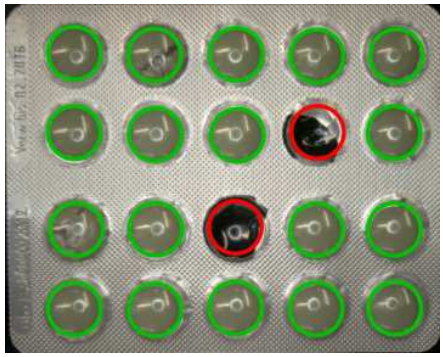
Description

The filter extracts foreground pixels by means of image thresholding and checks if their number comparing to the area of the whole ROI fits the range (**inMinAmount**, **inMaxAmount**). If the input image has 3 channels, the ThresholdToRegion_HSx filter is used for thresholding, and if the image has 1 channel (i.e. it is monochromatic), the simple ThresholdToRegion filter is performed. In the latter case only **inMinBrightness** and **inMaxBrightness** parameters matter.

Hints

- If the object location is variable, Pass an appropriate local coordinate system to **inRoiAlignment**.
- Define **inRoi** to specify the image location at which the object presence will be checked.
- Set **inBeginHue** and **inEndHue** to values appropriate for correct objects.
- Limit the ranges of **inMinSaturation**–**inMaxSaturation** and **inMinBrightness**–**inMaxBrightness**.
- Verify that the **outForeground** output represents pixels belonging to correct objects.
- Investigate the values that appear on the **outAmount** output, then set **inMinAmount** and **inMaxAmount** to values appropriate for correct objects.
- When creating data previews, use **outAlignedRoi** and NOT **inRoi** as only the former will be properly aligned to the object location.

Examples



*CheckPresence_PixelAmount performed on sample image with **inMinAmount** = 0.7. The defects are present in red circles.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect inMaxBrightness value in CheckPresence_PixelAmount.
DomainError	Incorrect inMinBrightness value in CheckPresence_PixelAmount.
DomainError	Region exceeds an input image in CheckPresence_PixelAmount.
DomainError	Not supported inImage pixel format in CheckPresence_PixelAmount. Supported formats: 3xUInt8, 1xUInt8, 1xInt8, 1xUInt16, 1xInt16, 1xInt32, 1xReal.

See Also

- CheckPresence_Intensity – Verifies object presence by analysing pixel intensities in the specified region.
- CheckPresence_EdgeAmount – Verifies object presence by analysing the amount of edges in the specified region.
- ThresholdToRegion_HSx – Creates a region containing image pixels which belongs to specified region in HSV, HSL or HSI space.
- ThresholdToRegion – Creates a region containing image pixels with values within the specified range.

CompareGoldenTemplate2

Header: AVL.h
Namespace: avl
Module: FoundationPro

Compares an image with a multi-image model using an ensemble of image features approach.

Applications: Finding general object defects by analyzing brightness deviations from a template image.

Syntax

```
void avl::CompareGoldenTemplate2
(
 const avl::Image& inImage,
 const avl::GoldenTemplate2Model& inModel,
 float inSensitivityA,
 float inSensitivityB,
 avl::Region& outDefects
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inModel	const GoldenTemplate2Model&			
→	inSensitivityA	float	0.0 - ∞	1.0f	Usually influences small, distinctive defects.
→	inSensitivityB	float	0.0 - ∞	0.95f	Usually influences bigger, extensive defects.
←	outDefects	Region&			

Description

This filter compares pixels of the input images against a template created by CreateGoldenTemplate2 and creates a region containing only pixels that are different.

The filter expects the object to be positioned precisely and in the same way as object images used to construct the **inModel** - see CreateGoldenTemplate2.

Sensitivity of the filter can be adjusted using **inSensitivityA** and **inSensitivityB** parameters.

See Also

- CreateGoldenTemplate2 – Create a model to be used with CompareGoldenTemplate2 filter.

CompareGoldenTemplate_Edges

Header: AVL.h
Namespace: avl
Module: FoundationPro

Compares image edges with the edges of a perfect template. Significant differences are considered defects.

Applications: Finding general object defects by analyzing missing or excessive edges.

Syntax

```
void avl::CompareGoldenTemplate_Edges
(
 const avl::Image& inImage,
 const avl::EdgeGoldenTemplate& inGoldenTemplate,
 const avl::CoordinateSystem2D& inGoldenTemplateAlignment,
 float inStdDevX,
 atl::Optional<float> inStdDevY,
 float inEdgeThreshold,
 float inEdgeHysteresis,
 int inMaxDistance,
 avl::Region& outDefects,
 bool& outDefectsPresent,
 avl::Region& outMissingEdges,
 avl::Region& outExcessiveEdges,
 atl::Optional<avl::Region&> outImageEdges = atl::NIL,
 atl::Optional<avl::Region&> outGoldenEdges = atl::NIL,
 atl::Optional<avl::Region&> outMatchingEdges = atl::NIL,
 atl::Optional<avl::Rectangle2D&> outObjectPosition = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inGoldenTemplate	const EdgeGoldenTemplate&			Golden edge template containing image with no defects
→	inGoldenTemplateAlignment	const CoordinateSystem2D&			Adjusts the golden template to the position of the inspected object
→	inStdDevX	float	0.0 - ∞	2.0f	Amount of horizontal smoothing used by the edge filter
→	inStdDevY	Optional<float>	0.0 - ∞	NIL	Amount of vertical smoothing used by the edge filter (Auto = inStdDevX)
→	inEdgeThreshold	float	0.0 - ∞	35.0f	Sufficient edge strength; edges of that strength will always be detected on the input image
→	inEdgeHysteresis	float	0.0 - ∞	15.0f	Value by which the edge threshold is decreased for edge points neighboring with sufficiently strong edges
→	inMaxDistance	int	0 - ∞	3	Maximal allowed distance between corresponding edges on the input and golden image
←	outDefects	Region&			Region of detected defects
←	outDefectsPresent	bool&			Flag indicating whether any defects were detected
←	outMissingEdges	Region&			Edges present on the golden image that are missing on the input image
←	outExcessiveEdges	Region&			Edges that are not present on the golden image
←	outImageEdges	Optional<Region&>		NIL	Edges on the input image
←	outGoldenEdges	Optional<Region&>		NIL	Edges on the golden image
←	outMatchingEdges	Optional<Region&>		NIL	Golden edges present on the input image
←	outObjectPosition	Optional<Rectangle2D&>		NIL	Position of the object being compared

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outImageEdges**, **outGoldenEdges**, **outMatchingEdges**, **outObjectPosition**.

Read more about Optional Outputs.

Description

This filter compares edges given input image with an image stored in a golden template and then, as a result, creates a region containing only pixels where edges are different.

This method is especially useful for finding defects of the object shape. Due to invulnerability to color changes, it may be used in appliances with changing light conditions.

Parameter **inMaxDistance** defines the maximal distances of two edges that should be treated as the same edge.

To prepare image and area to be compared in it you can use CreateGoldenTemplate_Edges filter or property window.

Mostly the **inEdgeThreshold** and **inEdgeHysteresis** parameters should have the same value as in CreateGoldenTemplate_Edges filter. The values should be changed however if the brightness of the input image is significantly different from the brightness of the golden image used in CreateGoldenTemplate_Edges. The **inEdgeThreshold** can be also decreased slightly if one wants to concentrate on missing edges or increased to concentrate on excessive edges.

More information about this technique can be found in Machine Vision Guide: Golden Template.

Hints

- A golden template comparison filter should be preceded with filters finding the object in certain location. Most typically we do this with LocateSingleObject_Edges1 filter.
- Connect the **inImage** input with the image containing the object.
- Select an image with a perfect object and use it to create golden template using the CreateGoldenTemplate_Edges filter.
- Set **inEdgeThreshold**, **inEdgeHysteresis**, **inStdDevX** and **inStdDevY** as in DetectEdges_AsRegion filter. These will control detection of the edges used for object comparison.
- Set **inMaxDistance** to specify the maximum spatial deviation between the edges on the input image and the edges on the template.

Remarks

Due to performance, it is recommended to create a template using the CreateGoldenTemplate_Edges filter outside a main loop of a program. It will create a model only once, instead of each iteration.

Errors

List of possible exceptions:

Error type	Description
DomainError	No valid golden template on input in CompareGoldenTemplate_Edges.

See Also

- CreateGoldenTemplate_Intensity – Creates golden template for application in CompareGoldenTemplate_Intensity filter.
- CompareGoldenTemplate_Intensity – Compares an image with a template image considered to have no defects.

CompareGoldenTemplate_Intensity

Header: AVL.h
Namespace: avl
Module: FoundationPro

Compares an image with a template image considered to have no defects.

Applications: Finding general object defects by analyzing brightness deviations from a template image.

Syntax

```
void avl::CompareGoldenTemplate_Intensity
(
 const avl::Image& inImage,
 const avl::GrayGoldenTemplate& inGoldenTemplate,
 atl::Optional<const avl::CoordinateSystem2D&> inGoldenTemplateAlignment,
 float inMaxDifference,
 int inMinDefectRadius,
 avl::Region& outDefects,
 avl::Region& outDifferenceRegion,
 bool& outDefectsPresent,
 avl::Region& outEdgeRegion,
 atl::Optional<avl::Rectangle2D&> outObjectPosition = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inGoldenTemplate	const GrayGoldenTemplate&			Golden gray template containing image of an object with no defects
→	inGoldenTemplateAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the golden template to the position of the inspected object
→	inMaxDifference	float	0.0 - ∞	20.0f	Maximal allowed difference between corresponding pixels of the input and golden images
→	inMinDefectRadius	int	0 - ∞	1	Minimal radius of a defect
←	outDefects	Region&			Region of detected defects
←	outDifferenceRegion	Region&			Region of pixels differing too much between the golden image and the input image
←	outDefectsPresent	bool&			Flag indicating whether any defects were detected
←	outEdgeRegion	Region&			Region of pixels that will not be compared
←	outObjectPosition	Optional<Rectangle2D&>		NIL	Position of the object being compared

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outObjectPosition**.

Read more about Optional Outputs.

Description

This filter compares pixels of the input images against a template image stored in passed **inGoldenTemplate** input. Then creates a region containing only pixels in which intensity difference is higher than **inMaxDifference** value as a result. This method is especially useful for finding defects like: smudges, noises and dust particles. It can be used for finding missing holes or changes in complex shapes.

When the defected pixels are found only consistent regions are selected. Minimal radius of accepted region is set in **inMinDefectRadius**.

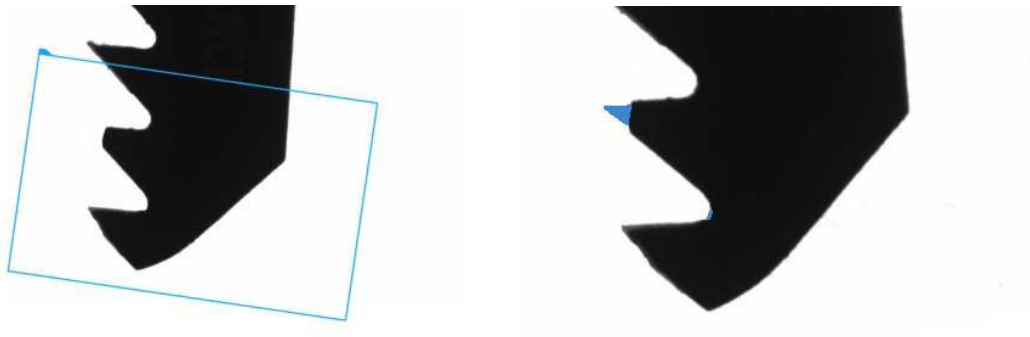
You can define a part of an image when defining **inGoldenTemplate**.

More information about this technique can be found in Machine Vision Guide: Golden Template.

Hints

- A golden template comparison filter should be preceded with filters finding the object in certain location. Most typically we do this with LocateSingleObject_Edges1 filter.
- Connect the **inImage** input with the image containing the object.
- Select an image with a perfect object and use it to create golden template using the CreateGoldenTemplate_Intensity filter.
- Set **inMaxDifference** to a value that assures detection of all true defects and no false ones. Verify this with the **outDefects** output.
- Dismiss very small defects by setting **inMinDefectRadius**.

Examples



CompareGoldenTemplate_Intensity performed on sample image. A part of the object is missing – it is marked in blue on the right.

Remarks

Due to performance, it is recommended to create a template using the CreateGoldenTemplate_Intensity filter outside a main loop of a program. It will create a model only once, instead of each iteration.

Errors

List of possible exceptions:

Error type	Description
DomainError	Input image format is incompatible with golden template image format in CompareGoldenTemplate_Intensity.
DomainError	No valid golden template on input in CompareGoldenTemplate_Intensity.

See Also

- CompareGoldenTemplate_Edges – Compares image edges with the edges of a perfect template. Significant differences are considered defects.
- CreateGoldenTemplate_Intensity – Creates golden template for application in CompareGoldenTemplate_Intensity filter.

CreateGoldenTemplate2

Header: AVL.h
Namespace: avl
Module: FoundationPro

Create a model to be used with CompareGoldenTemplate2 filter.

Syntax

```
void avl::CreateGoldenTemplate2
(
 const atl::Array<avl::Image>& inImages,
 const atl::Optional<const avl::Region&>& inObjectMask,
 int inDownscale,
 int inMaxDisplacement,
 int inLargeDefectSize,
 int inBrightnessAugmentation,
 int inNoiseAugmentation,
 float inSmoothingAugmentationStdDev,
 avl::GoldenTemplate2Model& outModel
)
```

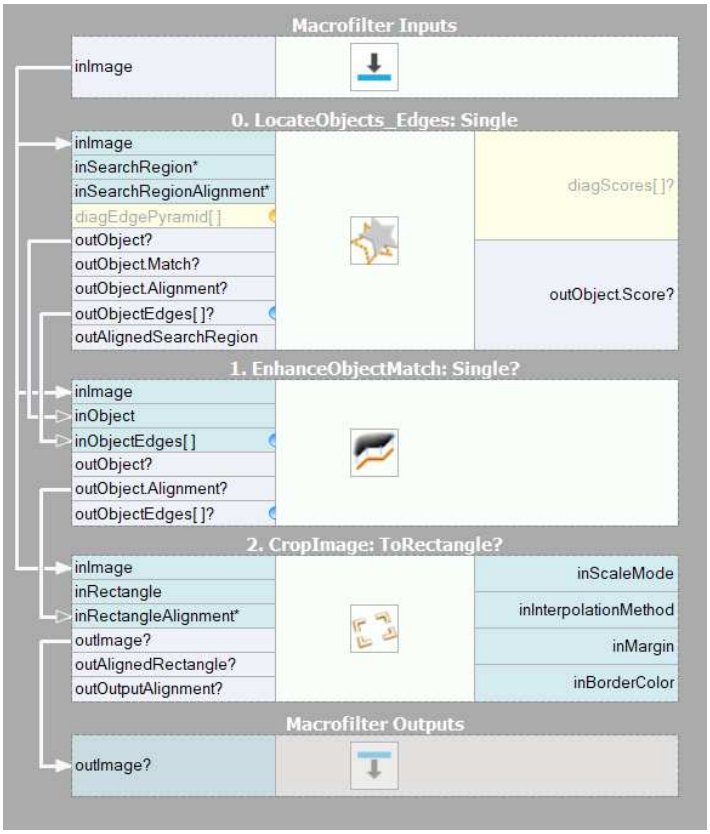
Parameters

	Name	Type	Range	Default	Description
→	inImages	const Array<Image>&			List of input images that has to be uniform in terms of size and format.
→	inObjectMask	const Optional<const Region&>&		NIL	
→	inDownscale	int	1 - ∞	2	Shrink the input for processing by dividing by specified value. Reduces sensitivity to minuscule (pixel-size) defects. Greatly improves processing speed.
→	inMaxDisplacement	int	0 - ∞	2	Error in object positioning. If in doubt, it is better to set this value too high. If set too low, subtle defects won't be detected, or no defects may not be detected at all. High values may impair detection of small defects, especially near edges.
→	inLargeDefectSize	int	0 - ∞	50	Expected size (diameter) of largest, extensive defects.
→	inBrightnessAugmentation	int		0	Allows for greater (additional to the value inferred from inImages training set) brightness deviation in inspected images.
→	inNoiseAugmentation	int		0	Allows for greater (additional to the value inferred from inImages training set) noise presence in inspected images. Uses a uniform noise with specified distribution width.
→	inSmoothingAugmentationStdDev	float	0.0 - ∞	0.0f	Allows for greater (additional to the value inferred from inImages training set) image smoothing in inspected images. Uses gaussian smoothing with specified standard deviation.
←	outModel	GoldenTemplate2Model&			

Description

This filter creates a template that can be later used in CompareGoldenTemplate2 filter.

Multiple images (at least 3, however it is recommended to use at least 5) of inspected object are required to properly create the model. Also, the filter expects the object to be positioned precisely. The positioning of inspected object can be achieved in Aurora Vision Studio by using following filter sequence.



Please note, that it is recommended to enable the scale adjustment in EnhanceSingleObjectMatch filter, as all input images for Golden Template model creation need to have the same dimensions.

The CreateGoldenTemplate2 filter also requires an object mask - pixels outside this mask won't be taken into consideration, however when object is visible against an uniform and constant background it is recommended to include some of the background in order to improve defect detection near the object edges.

It is important to set the **inMaxDisplacement** parameter properly, which describes the object positioning error, in pixels. If it is set too low, subtle defects won't be detected, or no defects may not be detected at all. On the other hand, high values may impair detection of small defects, especially near edges.

The **augmentation** could be used when it is known that images subjected to inspection will be different in *some specific manner* from the training image set. Namely, **inBrightnessAugmentation** is to be used when inspected images could differ in brightness from the training set, **inNoiseAugmentation** - for noise content difference and **inSmoothingAugmentationStdDev** - when inspected images could be blurred w.r.t the training set. Note, that it is always better to obtain a representative training set than to use augmentation parameters. Also note, that the overall sensitivity of the defect detection is governed by CompareGoldenTemplate2 filter sensitivity parameters.

Errors

List of possible exceptions:

Error type	Description
DomainError	inImages - inObjectMask dimensions differ in CreateGoldenTemplate2.
DomainError	inImages array must contain at least 3 different images of the object, 5 is recommended in CreateGoldenTemplate2
DomainError	Input image dimensions differ in CreateGoldenTemplate2.
DomainError	Input image formats differ in CreateGoldenTemplate2.

See Also

- CompareGoldenTemplate2 – Compares an image with a multi-image model using an ensemble of image features approach.

CreateGoldenTemplate_Edges

Header: AVL.h
Namespace: avl
Module: FoundationPro

Creates golden template for application in CompareGoldenTemplate_Edges filter.

Syntax

```
void avl::CreateGoldenTemplate_Edges
(
 const avl::Image& inTemplateImage,
 atl::Optional<const avl::Rectangle2D&> inTemplateArea,
 atl::Optional<const avl::Region&> inMask,
 atl::Optional<const avl::CoordinateSystem2D&> inTemplateAreaAlignment,
 float inStdDevX,
 atl::Optional<float> inStdDevY,
 float inEdgeThreshold,
 float inEdgeHysteresis,
 avl::EdgeGoldenTemplate& outGoldenTemplate,
 avl::Image& diagCroppedImage,
 avl::Region& diagCroppedEdges,
 avl::Region& diagEdges
)
```

Parameters

	Name	Type	Range	Default	Description
→	inTemplateImage	const Image&			Template image containing an object with no defects
→	inTemplateArea	Optional<const Rectangle2D&>		NIL	Desired area to compare
→	inMask	Optional<const Region&>		NIL	Range of pixels to compare
→	inTemplateAreaAlignment	Optional<const CoordinateSystem2D&>		NIL	Alignment of template bounded by inTemplateArea
→	inStdDevX	float	0.0 - ∞	2.0f	Amount of horizontal smoothing used by the edge filter
→	inStdDevY	Optional<float>	0.0 - ∞	NIL	Amount of vertical smoothing used by the edge filter (Auto = inStdDevX)
→	inEdgeThreshold	float	0.0 - ∞	35.0f	Sufficient edge strength; edges of that strength will always be detected on the template image
→	inEdgeHysteresis	float	0.0 - ∞	15.0f	Value by which the edge threshold is decreased for edge points neighboring with sufficiently strong edges
←	outGoldenTemplate	EdgeGoldenTemplate&			The output golden template structure
🔍	diagCroppedImage	Image&			Cropped image to be analyzed
🔍	diagCroppedEdges	Region&			Edges found on the cropped image
🔍	diagEdges	Region&			Edges projected onto the input image

Description

This filter creates a template structure that can be later used in CompareGoldenTemplate_Edges filter.

This method is especially useful for finding defects of the object shape. Due to invulnerability to color changes, it may be used in appliances with changing light conditions.

More information about this technique can be found in Machine Vision Guide: Golden Template.

Hints

- A golden template comparison filter should be preceded with filters finding the object in certain location. Most typically we do this with LocateSingleObject_Edges1 filter.
- Connect the **inTemplateImage** input with the image containing the perfect object.
- Set **inEdgeThreshold**, **inEdgeHysteresis**, **inStdDevX** and **inStdDevY** as in DetectEdges_AsRegion filter. These will control detection of the edges used for object comparison.

Remarks

Due to performance, it is recommended to create a template outside a main loop of a program. It will create a model only once, instead of each iteration.

See Also

- CompareGoldenTemplate_Edges – Compares image edges with the edges of a perfect template. Significant differences are considered defects.
- CreateGoldenTemplate_Intensity – Creates golden template for application in CompareGoldenTemplate_Intensity filter.

CreateGoldenTemplate_Intensity

Header: AVL.h
Namespace: avl
Module: FoundationPro

Creates golden template for application in CompareGoldenTemplate_Intensity filter.

Syntax

```
void avl::CreateGoldenTemplate_Intensity
(
 const avl::Image& inTemplateImage,
 atl::Optional<const avl::Rectangle2D&> inTemplateArea,
 atl::Optional<const avl::Region&> inMask,
 atl::Optional<const avl::CoordinateSystem2D&> inTemplateAreaAlignment,
 const float inEdgeThreshold,
 const int inEdgeDilation,
 avl::GrayGoldenTemplate& outGoldenTemplate,
 avl::Image& diagCroppedImage,
 avl::Region& diagCroppedEdgeRegion,
 avl::Region& diagEdgeRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inTemplateImage	const Image&			Template image containing an object with no defects
→	inTemplateArea	Optional<const Rectangle2D&>		NIL	Desired area to compare
→	inMask	Optional<const Region&>		NIL	Range of pixels to compare
→	inTemplateAreaAlignment	Optional<const CoordinateSystem2D&>		NIL	Alignment of template bounded by inTemplateArea
→	inEdgeThreshold	const float	0.0 - ∞	10.0f	Minimum strength of edges on the golden image near which comparison is NOT performed
→	inEdgeDilation	const int	0 - 1000000	1	Defines for how far from the detected edges comparison is NOT performed
←	outGoldenTemplate	GrayGoldenTemplate&			The output golden template structure
🔍	diagCroppedImage	Image&			Cropped image to be analyzed
🔍	diagCroppedEdgeRegion	Region&			Region of pixels that will not be compared
🔍	diagEdgeRegion	Region&			Edges projected onto the input image

Description

This filter creates a template structure that can be later used in CompareGoldenTemplate_Intensity filter. This golden template method is especially useful for finding defects like: smudges, noises and dust particles. It can be used for finding missing holes or changes in complex shapes.

Filter finds edges on an image and removes nearby pixels to avoid comparing pixels near edges which may contains some distortions due to shadows or changes in lightning. Input **inEdgeDilation** defines the width of the quiet zone around the edges and the parameter **inEdgeThreshold** define how strong must be change of the color between pixels to treat them as an edge.

More information about this technique can be found in Machine Vision Guide: Golden Template.

Hints

- A golden template comparison filter should be preceded with filters finding the object in certain location. Most typically we do this with LocateSingleObject_Edges1 filter.
- Connect the **inTemplateImage** input with the image containing the perfect object.
- Set **inEdgeThreshold** to a value that assures detection of all object edges, so that defects appearing near the edges can be ignored. Verify this setting with the **diagEdgeRegion** output – it should be present near the edges.

Remarks

Due to performance, it is recommended to create a template outside a main loop of a program. It will create a model only once, instead of each iteration.

See Also

- CompareGoldenTemplate_Intensity – Compares an image with a template image considered to have no defects.
- CreateGoldenTemplate_Edges – Creates golden template for application in CompareGoldenTemplate_Edges filter.

CreateMeasurementMap

Header: AVL.h
Namespace: avl
Module: FoundationPro

(Pre)computes image sampling locations used by MeasureObjectWidth function.

Syntax

```
void avl::CreateMeasurementMap
(
 const avl::ImageFormat& inImageFormat,
 const avl::SegmentScanField& inScanField,
 atl::Optional<const avl::CoordinateSystem2D&> inScanFieldAlignment,
 int inScanCount,
 int inScanWidth,
 avl::InterpolationMethod::Type inImageInterpolation,
 atl::Array<avl::ScanMap>& outMeasurementMap,
 atl::Optional<avl::SegmentScanField&> outAlignedScanField = atl::NIL,
 atl::Array<avl::Segment2D>& diagScanSegments
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImageFormat	const ImageFormat&			Information about dimensions, depth and pixel type of the scan image
→	inScanField	const SegmentScanField&			Field in which scans will be performed
→	inScanFieldAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the scan field to the position of the inspected object
→	inScanCount	int	2 - ∞	5	Number of scans to be performed
→	inScanWidth	int	1 - ∞	5	Width of the scan area
→	inImageInterpolation	InterpolationMethod::Type		Bilinear	Interpolation method used in extraction of image pixel values
←	outMeasurementMap	Array<ScanMap>&			
←	outAlignedScanField	Optional<SegmentScanField&>		NIL	Field in which the scans will be performed
🔍	diagScanSegments	Array<Segment2D>&			Array of scan segments

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedScanField**.

Read more about Optional Outputs.

DetectCorners_CornerResponse

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Detects corners using corner response method.

Applications: Detection of characteristic points on an image.

Syntax

```
void avl::DetectCorners_CornerResponse
(
 const avl::Image& inMonoImage,
 atl::Optional<const avl::Region&> inRoi,
 const avl::CornerResponseMethod::Type inCornerResponseMethod,
 const int inKernelSize,
 atl::Optional<float> inThreshold,
 atl::Optional<atl::Array<avl::Point2D>&> outCorners,
 atl::Optional<avl::Image&> outCornerResponseImage = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inMonoImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inCornerResponseMethod	const CornerResponseMethod::Type			Method for computing corner response
→	inKernelSize	const int	1 - 10	3	Method kernel size
→	inThreshold	Optional<float>	0.0 - 255.0	50.0f	Threshold for corner response value, between 0 and 255, default value is taken from SelectThresholdValue on outCornerResponseImage and entropy method
←	outCorners	Optional<Array<Point2D>&>			Found corner points
←	outCornerResponseImage	Optional<Image&>		NIL	

Requirements

For input **inMonoImage** only pixel formats are supported: 1□uint8, 1□int8, 1□uint16, 1□int16, 1□int32, 1□real.

Read more about pixel formats in Image documentation.

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outCorners**, **outCornerResponseImage**.

Read more about Optional Outputs.

Description

The operation detects corners using either Harris or Kanade-Tomasi corner response method, depending on **inCornerResponseMethod**. For every square window of size **inKernelSize** a *convolution matrix* is computed:

$$M = \begin{pmatrix} \sum g_r^2 & \sum g_c g_r \\ \sum g_c g_r & \sum g_c^2 \end{pmatrix}$$

where the summation is performed over the whole window and g_r, g_c denote horizontal and vertical gradient respectively at the point.
Harris' corner response is computed the following way:

$$H = \det(M) - k\,\mathrm{tr}^2(M)$$

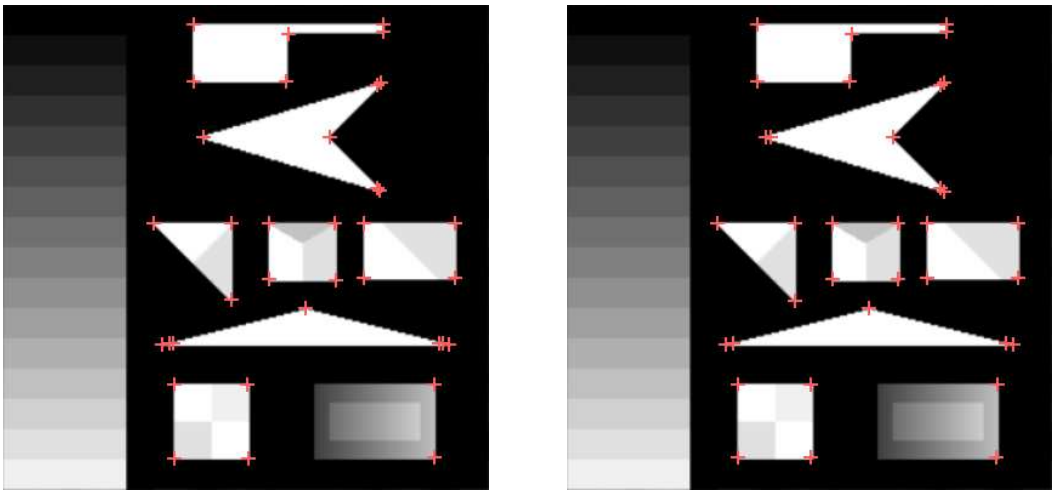
where k is a constant set to 0.01.
Kanade-Tomasi corner response is given by:

$$KT = \min(\lambda_1, \lambda_2)$$

where λ_1, λ_2 are eigenvalues of the convolution matrix.
Values H or KT give corner response image.

Then a few steps are performed in order to extract corner points. First, the normalized corner response function is thresholded with **inThreshold** with a small hysteresis, then the remained points are split into connected regions (blobs) and the center of each blob is determined.

Examples



*DetectCorners_CornerResponse with **inThreshold**=50, methods Harris and Kanade-Tomasi respectively.*

Remarks

Both methods give similar results and are quite fast. However, they tend to be less accurate than DetectCorners_Foerstner.

Errors

List of possible exceptions:

Error type	Description
DomainError	Unknown algorithm in DetectCorners_CornerResponse.
DomainError	Not supported inMonoImage pixel format in DetectCorners_CornerResponse. Supported formats: 1xUInt8, 1xInt8, 1xUInt16, 1xInt16, 1xInt32, 1xReal.

See Also

- DetectCorners_Foerstner – Detects corners using the Foerstner algorithm.

DetectCorners_Foerstner

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Detects corners using the Foerstner algorithm.

Applications: Detection of characteristic points on an image.

Syntax

```
void avl::DetectCorners_Foerstner
(
 const avl::Image& inMonoImage,
 atl::Optional<const avl::Region&> inRoi,
 const float inCornerQuality,
 const float inStrengthThreshold,
 const int inLocalness,
 atl::Array<avl::Point2D>& outPoints,
 avl::Image& diagRoundnessImage,
 avl::Image& diagStrengthImage
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inMonoImage	const Image&			Input image
➡	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
➡	inCornerQuality	const float	0.0 - 1.0	0.8f	Threshold on regularity of the corner
➡	inStrengthThreshold	const float	0.0 - 255.0	50.0f	Threshold on contrast of gradients forming the corner
➡	inLocalness	const int	1 - 11	3	How big-scaled the corners should be
⬅	outPoints	Array<Point2D>&			Found corner points
🔍	diagRoundnessImage	Image&			Calculated roundness for each input pixel
🔍	diagStrengthImage	Image&			Calculated strength for each pixel

Requirements

For input **inMonoImage** only pixel formats are supported: 1□uint8, 1□int8, 1□uint16, 1□int16, 1□int32, 1□real.

Read more about pixel formats in Image documentation.

Description

The operation detects corners using Foerstner algorithm. Its goal is to find corners defined as crossings of image edges.
For every square window of size 2*__inLocalness__+1 a *convolution matrix* is computed:

$$M = \begin{pmatrix} \sum g_r^2 & \sum g_c g_r \\ \sum g_c g_r & \sum g_c^2 \end{pmatrix}$$

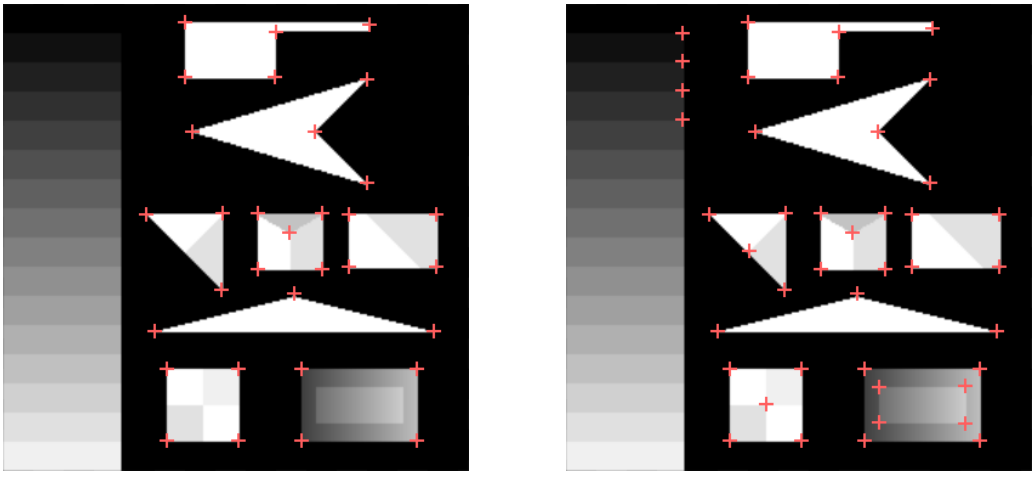
where the summation is performed over the whole window and g_r, g_c denote horizontal and vertical gradient respectively at the point.
Then the strength and so called *roundness* of the window is computed, where:

$$strength = \mathrm{tr}(M)$$

$$roundness = \mathrm{tr}(M)^2 / (4 * \det(M))$$

Roundness measures how similar are the gradients which form the corner.
Only windows with strength greater than **inStrengthThreshold** and *roundness* greater than **inCornerQuality** are considered. Then the non-maximum suppression on the window *roundness* is performed.
Finally the window's candidate for the corner is determined. It is done by minimizing square distance to all tangent lines (i.e. perpendicular to gradients) within the window, with distances weighted with gradient lengths.

Examples



***DetectCorners_Foerstner** with different **inStrengthThreshold** values.*

Remarks

Strength of the window corresponds to average gradient strength within the window.
Roundness of the window is always between 0 and 1. For most applications **inCornerQuality** values below 0.5 are not recommended.
Higher **inLocalness** values (e.g. greater than 4) may help to get rid of noise on the image, but decrease precision.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported inMonoImage pixel format in DetectCorners_Foerstner. Supported formats: 1xUInt8, 1xInt8, 1xUInt16, 1xInt16, 1xInt32, 1xReal.

See Also

- DetectCorners_CornerResponse – Detects corners using corner response method.

DetectLinePeak

Header: AVL.h
Namespace: avl
Module: FoundationPro

Finds line peaks on an image.

Applications: Created for applications with laser line detection.

Syntax

```
void avl::DetectLinePeak
(
 const avl::Image& inImage,
 avl::LinePeakDetectionMethod::Type inDetectionMethod,
 const float inThreshold,
 atl::Array<atl::Conditional<avl::Point2D> >& outLinePeakPoints,
 avl::Profile& diagLinePeakProfile
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inDetectionMethod	LinePeakDetectionMethod::Type		MaximalPixel	Method used to determine exact line peak position
→	inThreshold	const float	0.0 - ∞	128.0f	Minimal value of a bright pixel
←	outLinePeakPoints	Array<Conditional<Point2D> >&			Line peak positions
🔍	diagLinePeakProfile	Profile&			Profile of line peak positions

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Unknown input detection method in DetectLinePeak.

DetectLinePeak_Gauss

Header: AVL.h
Namespace: avl
Module: FoundationPro

Finds line peaks on an image.

Applications: Created for applications with laser line detection.

Syntax

```
void avl::DetectLinePeak_Gauss
(
 const avl::Image& inImage,
 const float inStdDev,
 const float inKernelRelativeSize,
 const float inThreshold,
 atl::Array<atl::Conditional<avl::Point2D> >& outLinePeakPoints,
 avl::Profile& diagLinePeakProfile
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inStdDev	const float	0.0 - ∞	4.0f	Standard deviation of the gaussian kernel
→	inKernelRelativeSize	const float	0.0 - 3.0	2.0f	A multiple of the standard deviation determining the size of the kernel
→	inThreshold	const float	0.0 - ∞	128.0f	Minimal value of a bright pixel
←	outLinePeakPoints	Array<Conditional<Point2D> >&			Line peak positions
🔍	diagLinePeakProfile	Profile&			Profile of line peak positions

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

MeasureObjectWidth

Header: AVL.h
Namespace: avl
Module: MetrologyBasic

Measures the width of an object using stripe detection.

Applications: Easy and precise measurement of distances between to straight parallel edges.

Syntax

```
void avl::MeasureObjectWidth
(
 const avl::Image& inImage,
 const atl::Array<avl::ScanMap>& inMeasurementMap,
 const avl::StripeScanParams& inStripeScanParams,
 avl::MeasureObjectMethod::Type inMeasureMethod,
 avl::Selection::Type inStripeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<avl::MEstimator::Type> inOutlierSuppression,
 int inOutlierCount,
 atl::Conditional<float>& outObjectWidth,
 atl::Conditional<avl::Segment2D>& outSegment1,
 atl::Conditional<avl::Segment2D>& outSegment2,
 atl::Optional<atl::Array<atl::Conditional<avl::Stripe1D>>&> outStripes = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outBrightnessProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inMeasurementMap	const Array<ScanMap>&		Input measurement map
→	inStripeScanParams	const StripeScanParams&		Parameters controlling the object stripe extraction process
→	inMeasureMethod	MeasureObjectMethod::Type		Method used to measure the object
→	inStripeSelection	Selection::Type	Selection::Best	Selection mode of edges of the object
→	inLocalBlindness	Optional<const LocalBlindness&>	NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inOutlierSuppression	Optional<MEstimator::Type>	NIL	Selects a method for ignoring incorrectly detected points
→	inOutlierCount	int		Determines how many outlying points are rejected before the width is measured
←	outObjectWidth	Conditional<float>&		Width of the object
←	outSegment1	Conditional<Segment2D>&		First edge of the object
←	outSegment2	Conditional<Segment2D>&		Second edge of the object
←	outStripes	Optional<Array<Conditional<Stripe1D>>&>	NIL	Detected stripes
←	outBrightnessProfiles	Optional<Array<Profile>&>	NIL	Extracted image profiles
←	outResponseProfiles	Optional<Array<Profile>&>	NIL	Profiles of the edge (derivative) operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outStripes**, **outBrightnessProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

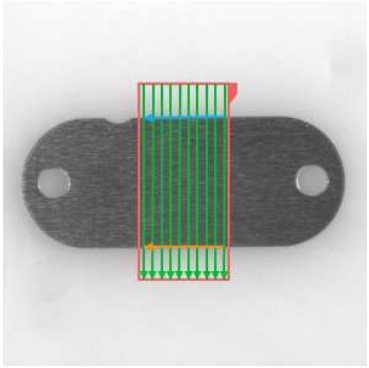
Description

The filter measures the width of an object present in an image. Internally, it performs a series of scans with ScanSingleStripe filter using **inMeasurementMap** and **inStripeScanParams**. The so obtained points are then used for computing two parallel segments by means of a slightly modified segment fitting routine. The process is supported by **inOutlierSuppression** parameter. Finally, having the aforementioned stripe widths and fitted segments' direction, the object width can be computed using a selected **inMeasureMethod** method. The **inOutlierCount** stripe widths most differing from the median width are not used in this step. The **inMeasurementMap** should be created with CreateMeasurementMap function. For the filter to work properly, the scan segments do not have to be necessarily perpendicular to the object edges.

Hints

- Define **inStripeScanParams.StripePolarity** to detect a particular edge type, and only that type.
- If some points are not found, try decreasing **inStripeScanParams.MinMagnitude**.
- If the object is more narrow than 6 pixels, change **inStripeScanParams.ProfileInterpolation** to **Quadratic3**.
- Experiment with **inOutlierSuppression** and **inOutlierCount** to deal with some amount of incorrectly detected points.
- Experiment with various values for **inMeasureMethod** to obtain best possible precision.

Examples



*MeasureObjectWidth performed on the sample image. Green segments are **diagScanSegments**, the other two are the found object edges.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Invalid MeasurementMap in MeasureObjectWidth function. Use CreateMeasurementMap function to create it properly.

See Also

- ScanSingleStripe – Locates the strongest pair of edges across a given path.
- FitSegmentToEdges – Performs a series of 1D edge detections and finds a segment that best matches the detected points.

PhotometricStereo_ComputeHeightMap

Header: AVL.h
Namespace: avl
Module: Photometric

Computes the shape of the image object using surface normals.

Syntax

```
void avl::PhotometricStereo_ComputeHeightMap
(
 const avl::Image& inSurfaceNormals,
 avl::Image& outHeightMap
)
```

Parameters

	Name	Type	Default	Description
→	inSurfaceNormals	const Image&		
←	outHeightMap	Image&		

Requirements

For input **inSurfaceNormals** only pixel formats are supported: 2□real.

Read more about pixel formats in Image documentation.

Errors

List of possible exceptions:

Error type	Description
DomainError	No proper structure available to compute heights for given image dimensions in PhotometricStereo_ComputeHeightMap. Please use PhotometricStereo_InitializeHeightMapStruct before using this function.
DomainError	Not supported inSurfaceNormals pixel format in PhotometricStereo_ComputeHeightMap. Supported formats: 2xReal.

PhotometricStereo_DestroyHeightMapStruct

Header: AVL.h
Namespace: avl
Module: Photometric

Destroys a structure used for computing height map in photometric stereo filter.

Syntax

```
void avl::PhotometricStereo_DestroyHeightMapStruct
(
 atl::Optional<int> inImageWidth,
 atl::Optional<int> inImageHeight
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImageWidth	Optional<int>	1 - ∞	NIL	Width of the input images
→	inImageHeight	Optional<int>	1 - ∞	NIL	Height of the input images

PhotometricStereo_General_Perform

Header: AVL.h
Namespace: avl
Module: Photometric

Computes surface normals using images with light source in different places.

Syntax

```
void avl::PhotometricStereo_General_Perform
(
 const atl::Array<avl::Image>& inImages,
 const atl::Array<float>& inLightXYAngles,
 const atl::Array<float>& inLightZAngles,
 avl::Image& outSurfaceNormals,
 avl::Image& outIntensities,
 avl::Image& outAlbedo
)
```

Parameters

	Name	Type	Default	Description
→	inImages	const Array<Image>&		
→	inLightXYAngles	const Array<float>&		Angle of light source on XY plane
→	inLightZAngles	const Array<float>&		Angle between light source and normal, where normal is camera optical axis
←	outSurfaceNormals	Image&		
←	outIntensities	Image&		
←	outAlbedo	Image&		

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Different image dimensions on input in PhotometricStereo_General_Perform.
DomainError	Different types of images on input in PhotometricStereo_General_Perform.
DomainError	Inconsistent array sizes in PhotometricStereo_General_Perform.
DomainError	Not enough images on input in PhotometricStereo_General_Perform.
DomainError	Not single-channel image on input in PhotometricStereo_General_Perform.

PhotometricStereo_InitializeHeightMapStruct

Header: AVL.h
Namespace: avl
Module: Photometric

Creates a structure used for computing height map in photometric stereo filter.

Syntax

```
void avl::PhotometricStereo_InitializeHeightMapStruct
(
 const int inImageWidth,
 const int inImageHeight
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImageWidth	const int	1 - ∞		Width of the input images
→	inImageHeight	const int	1 - ∞		Height of the input images

PhotometricStereo_Perform

Header: AVL.h
Namespace: avl
Module: Photometric

Computes surface normals using four images with light source in different places.

Syntax

```
void avl::PhotometricStereo_Perform
(
 const avl::Image& inImage1,
 const avl::Image& inImage2,
 const avl::Image& inImage3,
 const avl::Image& inImage4,
 const float inXYAngleOffset,
 const float inZAngle,
 avl::Image& outSurfaceNormals,
 avl::Image& outIntensities,
 avl::Image& outAlbedo
)
```

Parameters

	Name	Type	Default	Description
→	inImage1	const Image&		First input image
→	inImage2	const Image&		Second input image
→	inImage3	const Image&		Third input image
→	inImage4	const Image&		Fourth input image
→	inXYAngleOffset	const float	0.0f	Angle of first light source on XY plane
→	inZAngle	const float	45.0f	Angle between light source and normal, where normal is camera optical axis
←	outSurfaceNormals	Image&		
←	outIntensities	Image&		
←	outAlbedo	Image&		

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

PhotometricStereo_SurfaceCurvature

Header: AVL.h
Namespace: avl
Module: Photometric

Computes the surface curvature using previously computed normals.

Syntax

```
void avl::PhotometricStereo_SurfaceCurvature
(
 const avl::Image& inSurfaceNormals,
 avl::CurvatureMeasure::Type inCurvatureMeasure,
 float inSmoothingStdDevX,
 atl::Optional<float> inSmoothingStdDevY,
 float inScale,
 float inOffset,
 avl::Image& outSurfaceCurvature
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurfaceNormals	const Image&			
→	inCurvatureMeasure	CurvatureMeasure::Type		Gauss	Method used for measuring the surface curvature
→	inSmoothingStdDevX	float	0.0 - ∞	1.0f	Horizontal standard deviation of the gaussian smoothing applied to surface normals
→	inSmoothingStdDevY	Optional<float>	0.0 - ∞	NIL	Vertical standard deviation of the gaussian smoothing applied to surface normals
→	inScale	float	0.0 - ∞	1.0f	Scaling factor
→	inOffset	float		50.0f	Value added to rescaled curvature value
←	outSurfaceCurvature	Image&			

Requirements

For input **inSurfaceNormals** only pixel formats are supported: 2□uint8, 2□int8, 2□uint16, 2□int16, 2□int32, 2□real.

Read more about pixel formats in Image documentation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported inSurfaceNormals pixel format in PhotometricStereo_SurfaceCurvature. Supported formats: 2xUInt8, 2xInt8, 2xUInt16, 2xInt16, 2xInt32, 2xReal.

12. Image Spatial Transforms Maps

Table of content:

AddSpatialMaps

Header: AVL.h
Namespace: avl
Module: Calibration

Combines two spatial maps.

Applications: Makes it possible to use single RemapImage filter even if there are two or more image remapping operations (e.g. lens distortion + perspective distortion).

Syntax

```
void avl::AddSpatialMaps
(
 const avl::SpatialMap& inSpatialMap1,
 const avl::SpatialMap& inSpatialMap2,
 avl::SpatialMap& outSpatialMap
)
```

Parameters

	Name	Type	Default	Description
→	inSpatialMap1	const SpatialMap&		
→	inSpatialMap2	const SpatialMap&		
←	outSpatialMap	SpatialMap&		

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Different spatial maps image types in AddSpatialMaps.

CombineSpatialMaps

Header: AVL.h
Namespace: avl
Module: Calibration

Combines two spatial maps.

Applications: Makes it possible to use single RemapImage filter even if there are two or more image remapping operations (e.g. lens distortion + perspective distortion).

Syntax

```
void avl::CombineSpatialMaps
(
 const avl::SpatialMap& inSpatialMap1,
 const avl::SpatialMap& inSpatialMap2,
 avl::SpatialMap& outSpatialMap
)
```

Parameters

	Name	Type	Default	Description
→	inSpatialMap1	const SpatialMap&		
→	inSpatialMap2	const SpatialMap&		
←	outSpatialMap	SpatialMap&		

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Different spatial map interpolations in CombineSpatialMaps.
DomainError	Incompatible spatial map dimensions in CombineSpatialMaps.
DomainError	Incompatible spatial map pixel types in CombineSpatialMaps.

ConvertMatrixMapsToSpatialMap

Header: AVL.h
Namespace: avl
Module: Calibration

Joins two matrices of coordinates to produce a SpatialMap for use in RemapImage.

Syntax

```
void avl::ConvertMatrixMapsToSpatialMap
(
 const avl::ImageFormat& inImageFormat,
 const avl::Matrix& inMatrixX,
 const avl::Matrix& inMatrixY,
 avl::InterpolationMethod::Type inInterpolationMethod,
 bool inRoundingOpenCV,
 avl::SpatialMap& outSpatialMap,
 atl::Optional<avl::Region&> outOutputRegion = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inImageFormat	const ImageFormat&		Information about dimensions, depth and pixel type of the image
→	inMatrixX	const Matrix&		Map of real X coordinates
→	inMatrixY	const Matrix&		Map of real Y coordinates
→	inInterpolationMethod	InterpolationMethod::Type	Bilinear	
→	inRoundingOpenCV	bool		Use same interpolation convention as cvRemap
←	outSpatialMap	SpatialMap&		Output spatial map
←	outOutputRegion	Optional<Region&>	NIL	Pixels set by the spatial map application

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outOutputRegion**.

Read more about Optional Outputs.

Remarks

The two source matrices contain real-valued coordinates - if **inMatrixX[a, b]** = *x* and **inMatrixY[a, b]** = *y*, then the RemapImage filter applied with this map will generate the **outImage[a, b]** pixel based on pixels around (*x*, *y*) in **inImage**.

One example of application of this filter is the conversion of matrix maps generated with the OpenCV filter OpenCV **InitUndistortRectifyMap** function.

The **inRoundingOpenCV** parameter should be set to true if the matrices passed to this filter were obtained from an OpenCV function, like OpenCV **InitUndistortRectifyMap** function. This ensures that the results of RemapImage will be identical to OpenCV **Remap** function used with the same interpolation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Format of an empty image on input in ConvertMatrixMapsToSpatialMap.
DomainError	Input matrices must have equal sizes in ConvertMatrixMapsToSpatialMap.

See Also

- RemapImage – Applies a precomputed image transform, defined by a spatial map object.
- ConvertSpatialMapToMatrixMaps – Splits a spatial map into two matrices of source coordinates.

ConvertSpatialMap

Header: AVL.h
Namespace: avl
Module: Calibration

Converts a spatial map to a different format.

Syntax

```
void avl::ConvertSpatialMap
(
 const avl::SpatialMap& inSpatialMap,
 atl::Optional<avl::PlainType::Type> inNewPixelType,
 atl::Optional<int> inNewDepth,
 const int inNewPitchAlignment,
 avl::SpatialMap& outSpatialMap
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSpatialMap	const SpatialMap&			
→	inNewPixelType	Optional<PlainType::Type>		NIL	
→	inNewDepth	Optional<int>	1 - 4	NIL	
→	inNewPitchAlignment	const int	0 - +∞	16	
←	outSpatialMap	SpatialMap&			

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect inSpatialMap map in ConvertSpatialMap.

ConvertSpatialMapToMatrixMaps

Header: AVL.h
Namespace: avl
Module: Calibration

Splits a spatial map into two matrices of source coordinates.

Syntax

```
void avl::ConvertSpatialMapToMatrixMaps
(
 const avl::SpatialMap& inSpatialMap,
 bool inRoundingOpenCV,
 avl::Matrix& outMatrixX,
 avl::Matrix& outMatrixY
)
```

Parameters

	Name	Type	Default	Description
→	inSpatialMap	const SpatialMap&		
→	inRoundingOpenCV	bool		Use same interpolation convention as cvRemap
←	outMatrixX	Matrix&		Map of real X coordinates
←	outMatrixY	Matrix&		Map of real Y coordinates

Description

This operation allows to inspect the accessed coordinates in the image being remapped.

If the input SpatialMap uses the **nearest neighbor interpolation**, it is not possible to recover the exact source coordinates. In that case the pixel center, or the top left corner when using **inRoundingOpenCV** set to **True**, is taken as an approximation.

Remarks

For pixels in the input spatial map which are not well defined, an artificial pair of invalid coordinates, (-10, -10), is returned.

The **inRoundingOpenCV** parameter should be set to True if the matrices will be used with OpenCV **Remap** function. This ensures the results of OpenCV **Remap** function will be the same as of RemapImage used with the input SpatialMap.

See Also

- RemapImage – Applies a precomputed image transform, defined by a spatial map object.
- ConvertMatrixMapsToSpatialMap – Joins two matrices of coordinates to produce a SpatialMap for use in RemapImage.
- ConvertSpatialMap_ToNearest – Converts any spatial map to NearestNeighbour interpolation.

ConvertSpatialMap_ToNearest

Header: AVL.h
Namespace: avl
Module: Calibration

Converts any spatial map to NearestNeighbour interpolation.

Applications: Reducing quality for the purpose of speed.

Syntax

```
void avl::ConvertSpatialMap_ToNearest
(
 const avl::SpatialMap& inSpatialMap,
 avl::SpatialMap& outSpatialMap
)
```

Parameters

	Name	Type	Default	Description
→	inSpatialMap	const SpatialMap&		
←	outSpatialMap	SpatialMap&		

Description

This operation can be used to convert a SpatialMap using Bilinear interpolation to one using nearest neighbour interpolation. If the input SpatialMap is already nearest neighbour, it is returned unchanged.

Remarks

The compatible ImageFormat remains the same as for the input SpatialMap in any case.

This conversion is not reversible, because a SpatialMap with nearest Neighbour interpolation holds less information than one with Bilinear interpolation.

See Also

- RemapImage – Applies a precomputed image transform, defined by a spatial map object.
- ConvertMatrixMapsToSpatialMap – Joins two matrices of coordinates to produce a SpatialMap for use in RemapImage.
- ConvertSpatialMapToMatrixMaps – Splits a spatial map into two matrices of source coordinates.

CreateCylinderMap

Header: AVL.h
Namespace: avl
Module: Calibration

Creates a spatial map for transformations from a cylinder surface to a flat rectangle.

Applications: Inspection of the surface of bottles and other cylindrical objects. The result is used by RemapImage.

Syntax

```
void avl::CreateCylinderMap
(
 const avl::ImageFormat& inImageFormat,
 const avl::Rectangle2D& inCylinderRectangle,
 const float inCylinderRadiusCorrection,
 atl::Optional<avl::Point2D> inOpticalAxis,
 atl::Optional<int> inNewWidth,
 atl::Optional<int> inNewHeight,
 const int inMargin,
 avl::InterpolationMethod::Type inInterpolationMethod,
 avl::CylinderMappingMode::Type inCylinderMappingMode,
 avl::SpatialMap& outSpatialMap,
 atl::Optional<avl::Region&> outOutputRegion = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImageFormat	const ImageFormat&			Information about dimensions, depth and pixel type of the image
→	inCylinderRectangle	const Rectangle2D&			Bounding rectangle of the cylinder
→	inCylinderRadiusCorrection	const float	0.0 - ∞	0.0f	How many pixels the cylinder radius is larger than the visible circle radius
→	inOpticalAxis	Optional<Point2D>		NIL	Coordinates of the camera optical axis (Auto = image center)
→	inNewWidth	Optional<int>	1 - ∞	NIL	Width of an image created by output spatial map application
→	inNewHeight	Optional<int>	1 - ∞	NIL	Height of an image created by output spatial map application
→	inMargin	const int	0 - ∞	0	Width of the cylinder extreme points zone excluded from spatial map
→	inInterpolationMethod	InterpolationMethod::Type		Bilinear	Interpolation method used in extraction of image pixel values
→	inCylinderMappingMode	CylinderMappingMode::Type			Determines which pixels of the mapped cylinder have to be within the given rectangle.
←	outSpatialMap	SpatialMap&			Output spatial map
←	outOutputRegion	Optional<Region&>		NIL	Pixels set by the spatial map application

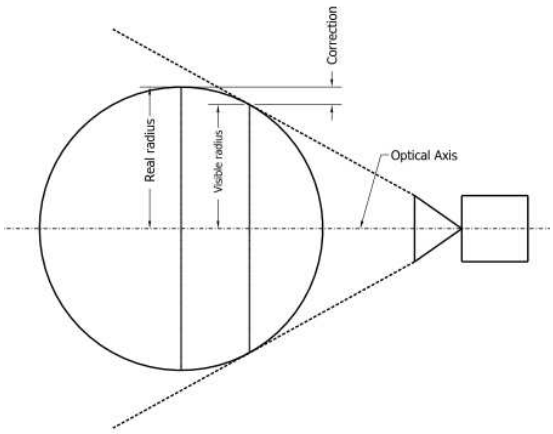
Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outOutputRegion**.

Read more about Optional Outputs.

Description

The filter creates a spatial map that allows to transform a cylinder surface to a flat rectangle. The **inCylinderRectangle** should be the minimal rectangle that contains the given cylinder. Because of the presence of a camera, the cylinder radius is not equal to but greater than the width of the input rectangle. To compensate the difference, **inCylinderRadiusCorrection** has to be properly set experimentally. Another important parameter is the **inOpticalAxis** input that represents the coordinates of the camera optical axis, i.e. it shows which pixel is directly under the camera. It is set to the input image center by default.



Examples

*Results of applying RemapImage with a spatial map created with the **CreateCylinderMap** filter.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Cylinder stripe within rectangle thinner then a single pixel in CreateCylinderMap.
DomainError	Empty domain in CreateCylinderMap.
DomainError	Mapped image width too large in CreateCylinderMap.
DomainError	Margin too large in CreateCylinderMap.

CreateImageInversePolarTransformMap

Header: AVL.h
Namespace: avl
Module: Calibration

Creates a spatial map representing an image inverse polar transform.

Applications: Data preprocessing for fast inverse polar transform. The result is used by RemapImage.

Syntax

```
void avl::CreateImageInversePolarTransformMap
(
 const avl::ImageFormat& inImageFormat,
 const avl::Point2D& inCenter,
 avl::PolarSpaceType::Type inSpaceType,
 avl::InterpolationMethod::Type inInterpolationMethod,
 avl::SpatialMap& outPolarMap,
 atl::Optional<avl::Region&> outOutputRegion = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inImageFormat	const ImageFormat&		Information about dimensions, depth and pixel type of the image
→	inCenter	const Point2D&		
→	inSpaceType	PolarSpaceType::Type		Method of transformation
→	inInterpolationMethod	InterpolationMethod::Type	Bilinear	Interpolation method used in extraction of image pixel values
←	outPolarMap	SpatialMap&		Output spatial map
←	outOutputRegion	Optional<Region&>	NIL	Pixels set by the spatial map application

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outOutputRegion**.

Read more about Optional Outputs.

Description

The operation generates map that describes image transformation from polar or log-polar space to euclidean space. Two modes of pixel interpolation are available, the bilinear filtering being more precise and computationally expensive. Usually creating map and then using RemapImage is faster than ImageInversePolarTransform. For more information see ImageInversePolarTransform.

See Also

- RemapImage – Applies a precomputed image transform, defined by a spatial map object.
- ImagePolarTransform – Transforms an image to polar or log-polar space.

CreateImagePolarTransformMap

Header: AVL.h
Namespace: avl
Module: Calibration

Creates a spatial map representing an image polar transform.

Applications: Data preprocessing for fast image polar transform. The result is used by RemapImage.

Syntax

```
void avl::CreateImagePolarTransformMap
(
 const avl::ImageFormat& inImageFormat,
 const avl::Point2D& inCenter,
 avl::PolarSpaceType::Type inSpaceType,
 avl::InterpolationMethod::Type inInterpolationMethod,
 avl::SpatialMap& outPolarMap,
 atl::Optional<avl::Region&> outOutputRegion = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inImageFormat	const ImageFormat&		Information about dimensions, depth and pixel type of the image
→	inCenter	const Point2D&		
→	inSpaceType	PolarSpaceType::Type		Method of transformation
→	inInterpolationMethod	InterpolationMethod::Type	Bilinear	Interpolation method used in extraction of image pixel values
←	outPolarMap	SpatialMap&		Output spatial map
←	outOutputRegion	Optional<Region&>	NIL	Pixels set by the spatial map application

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outOutputRegion**.

Read more about Optional Outputs.

Description

The operation generates map that describes image transformation to polar or log-polar space. Two modes of pixel interpolation are available, the bilinear filtering being more precise and computationally expensive. Usually creating map and then using RemapImage is faster than ImagePolarTransform. For more information see ImagePolarTransform.

See Also

- RemapImage – Applies a precomputed image transform, defined by a spatial map object.
- ImagePolarTransform – Transforms an image to polar or log-polar space.

CreateImageResizeMap

Header: AVL.h
Namespace: avl
Module: Calibration

Creates a spatial map representing an image resizing.

Applications: Data preprocessing for fast image resize between two constant sizes. The result is used by RemapImage.

Syntax

```
void avl::CreateImageResizeMap
(
 const avl::ImageFormat& inImageFormat,
 int inNewWidth,
 int inNewHeight,
 avl::InterpolationMethod::Type inInterpolationMethod,
 avl::SpatialMap& outResizeMap,
 atl::Optional<avl::Region&> outOutputRegion = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImageFormat	const ImageFormat&			Information about dimensions, depth and pixel type of the image
→	inNewWidth	int	1 - ∞		Width of an image created by output spatial map application
→	inNewHeight	int	1 - ∞		Height of an image created by output spatial map application
→	inInterpolationMethod	InterpolationMethod::Type		Bilinear	Interpolation method used in extraction of image pixel values
←	outResizeMap	SpatialMap&			Output spatial map
←	outOutputRegion	Optional<Region&>		NIL	Pixels set by the spatial map application

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outOutputRegion**.

Read more about Optional Outputs.

Description

The operation generates map that stretches or shrinks the image. Two modes of pixel interpolation are available, the bilinear filtering being more precise and computationally expensive. Usually creating map and then using RemapImage is faster than ResizeImage.

See Also

- RemapImage – Applies a precomputed image transform, defined by a spatial map object.
- ResizeImage – Enlarges or shrinks an image to new dimensions.

CreateImageRotationMap

Header: AVL.h
Namespace: avl
Module: Calibration

Creates a spatial map representing an image rotation.

Applications: Preprocessing data for fast image rotation by a constant angle. The result is used by RemapImage.

Syntax

```
void avl::CreateImageRotationMap
(
 const avl::ImageFormat& inImageFormat,
 float inAngle,
 avl::RotationSizeMode::Type inSizeMode,
 avl::InterpolationMethod::Type inInterpolationMethod,
 avl::SpatialMap& outRotationMap,
 atl::Optional<avl::Region&> outOutputRegion = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inImageFormat	const ImageFormat&		Information about dimensions, depth and pixel type of the image
→	inAngle	float	45.0f	The angle of rotation
→	inSizeMode	RotationSizeMode::Type		
→	inInterpolationMethod	InterpolationMethod::Type	Bilinear	Interpolation method used in extraction of image pixel values
←	outRotationMap	SpatialMap&		Output spatial map
←	outOutputRegion	Optional<Region&>	NIL	Pixels set by the spatial map application

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outOutputRegion**.

Read more about Optional Outputs.

Description

The operation generates map that describes rotation mapping. Dimensions of the resulting image depends on **inSizeMode**. In 'Fit' mode size is extended to fit the rotated image. In 'Preserve' mode size of source image is left unchanged and part of rotated image may be lost. Two modes of pixel interpolation are available, the bilinear filtering being more precise and computationally expensive. Usually creating map and then using RemapImage is faster than RotateImage. **outOutputRegion** return region, which can be calculated by RemapImage.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- RemapImage – Applies a precomputed image transform, defined by a spatial map object.
- RotateImage – Rotates an image clockwise.

CreateMatrixTransformMap

Header: AVL.h
Namespace: avl
Module: Calibration

Creates a spatial map which performs transform defined as a 3x3 homography matrix.

Syntax

```
void avl::CreateMatrixTransformMap
(
 const avl::ImageFormat& inImageFormat,
 atl::Optional<const avl::Region&> inRoi,
 const avl::Matrix& inTransformMatrix,
 const atl::Optional<avl::Size>& inNewSize,
 avl::InterpolationMethod::Type inInterpolationMethod,
 avl::SpatialMap& outSpatialMap
)
```

Parameters

	Name	Type	Default	Description
➡	inImageFormat	const ImageFormat&		
➡	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
➡	inTransformMatrix	const Matrix&		3x3 homography matrix.
➡	inNewSize	const Optional<Size>&	NIL	New image size after remapping
➡	inInterpolationMethod	InterpolationMethod::Type		Interpolation method used in extraction of image pixel values
⬅	outSpatialMap	SpatialMap&		Calculated spatial map

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Input transformation matrix must have dimensions 3x3 in CreateMatrixTransformMap.

CreatePerspectiveMap_Path

Header: AVL.h
Namespace: avl
Module: Calibration

Creates a perspective transform map from planes defined by paths.

Applications: Data preprocessing for fast perspective correction. The result is used by RemapImage.

Syntax

```
void avl::CreatePerspectiveMap_Path
(
 const avl::ImageFormat& inImageFormat,
 atl::Optional<const avl::Region&> inRoi,
 const avl::Path& inImagePath,
 atl::Optional<const avl::Path&> inTargetPath,
 atl::Optional<const avl::Size&> inNewSize,
 const avl::InterpolationMethod::Type inInterpolationMethod,
 avl::SpatialMap& outSpatialMap,
 avl::Matrix& outTransformMatrix
)
```

Parameters

	Name	Type	Default	Description
➡	inImageFormat	const ImageFormat&		
➡	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
➡	inImagePath	const Path&		Plane defined by a closed path made of 4 points
➡	inTargetPath	Optional<const Path&>	NIL	Target plane. If NIL then image corners are used starting from (0,0) (Width, 0) (Width, Height), (0, Height)
➡	inNewSize	Optional<const Size&>	NIL	New image size after remapping
➡	inInterpolationMethod	const InterpolationMethod::Type		Interpolation method used in extraction of image pixel values
⬅	outSpatialMap	SpatialMap&		Calculated spatial map
⬅	outTransformMatrix	Matrix&		Used transform matrix

Description

This operation computes a SpatialMap which can be later used for removing a perspective distortion from an image.

The operation maps the input path (**inImagePath**) into the target path (**inTargetPath**). If the **inTargetPath** input is set to Auto this path will be made from the corner points of the input image. Both paths must be closed and must be made of four points.

The input **inNewSize** allows rescaling of the output image.

The **inImageFormat** format is necessary for preparation of a spatial map.

The **outTransformMatrix** output allows verifying the found transformation.

Examples

Image before and after the perspective transform created by CreatePerspectiveMap_Path.

Remarks

This filter is a good choice for local perspective distortion removal - such as "unwrapping" boxes, as is depicted by the example above. Applications concerned with observing real-world flat surfaces (such as observing conveyor belts) should use methods that are more accurate. Please refer to: **Machine Vision Guide - Camera Calibration and World Coordinates**

Notice that both **inImagePath** path and **inTargetPath** must be made of four points. This filter creates point to point transform so the changing points order in paths may yield an unexpected result.

The best way to understand the relation between **inImagePath** and **inTargetPath** is to present the first of them on the input image and the second on the background of remapped image.

*The image before remapping on the left (with **inImagePath**) and the image after on the right (with **inTargetPath**).*

Errors

List of possible exceptions:

Error type	Description
DomainError	inImagePath input must contain a closed path created from 4 points in CreatePerspectiveMap_Path.
DomainError	inTargetPath input must contain a closed path created from 4 points in CreatePerspectiveMap_Path.

See Also

- CreatePerspectiveMap_Points – Creates a perspective transform map from four points denoting a rectangle in the world coordinates.
- RemapImage – Applies a precomputed image transform, defined by a spatial map object.

CreatePerspectiveMap_Points

Header: AVL.h
Namespace: avl
Module: Calibration

Creates a perspective transform map from four points denoting a rectangle in the world coordinates.

Applications: Data preprocessing for fast perspective correction. The result is used by RemapImage.

Syntax

```
void avl::CreatePerspectiveMap_Points
(
 const avl::ImageFormat& inImageFormat,
 atl::Optional<const avl::Region&> inRoi,
 const atl::Array<avl::Point2D>& inImagePoints,
 atl::Optional<const atl::Array<avl::Point2D>&> inTargetPoints,
 atl::Optional<const avl::Size&> inNewSize,
 const avl::InterpolationMethod::Type inInterpolationMethod,
 avl::SpatialMap& outSpatialMap,
 avl::Matrix& outTransformMatrix
)
```

Parameters

	Name	Type	Default	Description
→	inImageFormat	const ImageFormat&		
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inImagePoints	const Array<Point2D>&		Points define real object plane corners.
→	inTargetPoints	Optional<const Array<Point2D>&>	NIL	Points define target plane corners. If NIL then image corners are used starting from (0,0) (Width, 0) (Width, Height), (0, Height)
→	inNewSize	Optional<const Size&>	NIL	New image size after remapping
→	inInterpolationMethod	const InterpolationMethod::Type		Interpolation method used in extraction of image pixel values
←	outSpatialMap	SpatialMap&		Created SpatialMap with perspective transform
←	outTransformMatrix	Matrix&		Used transform matrix

Description

This operation computes a SpatialMap which can be later used for removing a perspective distortion from an image.

The operation maps the input image points (**inImagePoints**) onto locations described by the target points (**inTargetPoints**). If the **inTargetPoints** input is set to Auto this array will be made from the corner points of the input image. Both array must contains four points.

The input **inNewSize** allows rescaling of the output image.

The **inImageFormat** format is necessary for preparation of a spatial map.

The **outTransformMatrix** output allows verifying the found transformation.

Remarks

This filter is a good choice for local perspective distortion removal - such as "unwrapping" boxes, as is depicted by the example above. Applications concerned with observing real-world flat surfaces (such as observing conveyor belts) should use methods that are more accurate. Please refer to: **Machine Vision Guide - Camera Calibration and World Coordinates**

Notice that both **inImagePoints** array and **inTargetPoints** must be made of four points. This filter creates point to point transform so changing the points' order in the array may yield an unexpected result.

The best way to understand the relation between **inImagePoints** and **inTargetPoints** is to present the first of them on the input image and the second on the background of the remapped image.

*The image before remapping on the left (with **inImagePoints**) and the image after on the right (with **inTargetPoints**) created by CreatePerspectiveMap_Path.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Each of input array must contain four points in CreatePerspectiveMap_Points.

See Also

- CreatePerspectiveMap_Path – Creates a perspective transform map from planes defined by paths.
- RemapImage – Applies a precomputed image transform, defined by a spatial map object.

CreatePincushionMap

Header: AVL.h
Namespace: avl
Module: Calibration

Creates a spatial map for removing/adding pincushion distortion, using divisional lens distortion model.

Applications: The easiest way to remove typical lens distortion. Use together with RemapImage.

Syntax

```
void avl::CreatePincushionMap
(
 const avl::ImageFormat& inImageFormat,
 const float inKappa,
 const atl::Optional<avl::Point2D>& inCenter,
 avl::InterpolationMethod::Type inInterpolationMethod,
 avl::SpatialMap& outMap
)
```

Parameters

	Name	Type	Default	Description
→	inImageFormat	const ImageFormat&		
→	inKappa	const float		
→	inCenter	const Optional<Point2D>&	NIL	
→	inInterpolationMethod	InterpolationMethod::Type		
←	outMap	SpatialMap&		

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

CreateSphereMap

Header: AVL.h
Namespace: avl
Module: Calibration

Creates a spatial map for transformations from a sphere surface to a flat rectangle.

Applications: Inspection of the surface of balls and other spherical objects. The result is used by RemapImage.

Syntax

```
void avl::CreateSphereMap
(
 const avl::ImageFormat& inImageFormat,
 const avl::Rectangle2D& inSphereRectangle,
 const float inSphereRadiusCorrection,
 atl::Optional<int> inNewDimension,
 const int inMargin,
 avl::InterpolationMethod::Type inInterpolationMethod,
 avl::SpatialMap& outSpatialMap,
 atl::Optional<avl::Region&> outOutputRegion = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImageFormat	const ImageFormat&			Information about dimensions, depth and pixel type of the image
→	inSphereRectangle	const Rectangle2D&			Bounding rectangle of a sphere
→	inSphereRadiusCorrection	const float	0.0 - ∞	0.0f	How many pixels the sphere radius is larger than the visible circle radius
→	inNewDimension	Optional<int>	1 - ∞	NIL	Width and height of an image created by output spatial map application
→	inMargin	const int	0 - ∞	0	Width of the sphere extreme points zone excluded from spatial map
→	inInterpolationMethod	InterpolationMethod::Type		Bilinear	Interpolation method used in extraction of image pixel values
←	outSpatialMap	SpatialMap&			Output spatial map
←	outOutputRegion	Optional<Region&>		NIL	Pixels set by the spatial map application

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outOutputRegion**.

Read more about Optional Outputs.

Examples

*Results of applying RemapImage with a spatial map created with the **CreateSphereMap** filter.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty rectangle on input in CreateSphereMap.

CropSpatialMap

Header: AVL.h
Namespace: avl
Module: Calibration

Crops a spatial map to the specified input and output boxes.

Syntax

```
void avl::CropSpatialMap
(
 const avl::SpatialMap& inSpatialMap,
 const avl::Box& inInputCroppingBox,
 const avl::Box& inOutputCroppingBox,
 avl::SpatialMap& outCroppedSpatialMap
)
```

Parameters

	Name	Type	Default	Description
→	inSpatialMap	const SpatialMap&		
→	inInputCroppingBox	const Box&		
→	inOutputCroppingBox	const Box&		
←	outCroppedSpatialMap	SpatialMap&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect inSpatialMap map in CropSpatialMap.

LoadSpatialMap

Header: AVL.h
Namespace: avl
Module: Calibration

Deserializes a SpatialMap object from a AVDATA file.

Syntax

```
void avl::LoadSpatialMap
(
 const atl::File& inFilename,
 avl::SpatialMap& outSpatialMap
)
```

Parameters

	Name	Type	Default	Description
→	inFilename	const File&		Name of the source file
←	outSpatialMap	SpatialMap&		Deserialized SpatialMap

RemapImage

Header: AVL.h
Namespace: avl
Module: Calibration

Applies a precomputed image transform, defined by a spatial map object.

Applications: Fast (precomputed) image transformations, especially for view undistortion or object geometry correction (e.g. pos recognition of labels on cylindrical bottles).

Syntax

```
void avl::RemapImage
(
 const avl::Image& inImage,
 const avl::SpatialMap& inSpatialMap,
 atl::Optional<const avl::Region&> inMapRoi,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inSpatialMap	const SpatialMap&		Definition of the transformation
→	inMapRoi	Optional<const Region&>	NIL	Defines which elements of the spatial map are valid
←	outImage	Image&		Output image

Description

The operation applies an arbitrary spatial transformation to an image using a SpatialMap object. The input image has to be compatible with the given map. You can check if a map is compatible with an image using TestSpatialMapApplicability.

Hints

- Connect the **inSpatialMap** input with a spatial map. This can be for example the result of the CreateSphereMap function or similar.

Remarks

Read more about how to handle images from camera in Camera Calibration article.

Hardware Acceleration

This operation is optimized for SSE4 technology for pixels of types: 1xUINT8, 3xUINT8.

This operation is optimized for AVX2 technology for pixels of type: 3xUINT8.

This operation is optimized for NEON technology for pixels of types: 1xUINT8, 3xUINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect SpatialMap on input in RemapImage.

See Also

- TestSpatialMapApplicability – Checks if a spatial map may be applied to transform the given image.

SaveSpatialMap

Header: AVL.h
Namespace: avl
Module: Calibration

Serializes a SpatialMap object to an AVDATA file.

Syntax

```
void avl::SaveSpatialMap
(
 const avl::SpatialMap& inSpatialMap,
 const atl::File& inFilename
)
```

Parameters

	Name	Type	Default	Description
→	inSpatialMap	const SpatialMap&		SpatialMap to be serialized
→	inFilename	const File&		Name of the target file

TestSpatialMapApplicability

Header: AVL.h
Namespace: avl
Module: Calibration

Checks if a spatial map may be applied to transform the given image.

Syntax

```
void avl::TestSpatialMapApplicability
(
 const avl::Image& inImage,
 const avl::SpatialMap& inSpatialMap,
 bool& outAreCompatible
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inSpatialMap	const SpatialMap&		
←	outAreCompatible	bool&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect SpatialMap on input in TestSpatialMapApplicability.

UnmapPoint

Header: AVL.h
Namespace: avl
Module: Calibration

Computes locations of image point on the source image from image point location on remapped image.

Syntax

```
void avl::UnmapPoint
(
 const avl::SpatialMap& inSpatialMap,
 const avl::Point2D& inPoint,
 atl::Conditional<avl::Point2D>& outPoint
)
```

Parameters

	Name	Type	Default	Description
→	inSpatialMap	const SpatialMap&		
→	inPoint	const Point2D&		
←	outPoint	Conditional<Point2D>&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect spatial map in UnmapPoint.

13. Profile Combinators

Table of content:

AccumulateProfile

Header: AVL.h
Namespace: avl
Module: FoundationPro

Creates a profile in a loop by concatenating individual values.

Syntax

```
void avl::AccumulateProfile
(
 const float inValue,
 atl::Optional<int> inMaxCount,
 bool inReset,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Range	Default	Description
→	inValue	const float			
→	inMaxCount	Optional<int>	0 - ∞	NIL	Number of last values that are remembered
→	inReset	bool			Reset accumulator state
←	outProfile	Profile&			Output profile

Errors

List of possible exceptions:

Error type	Description
DomainError	inMaxCount cannot be negative in AccumulateProfile.

AddProfiles

Header: AVL.h
Namespace: avl
Module: FoundationPro

Adds two profiles value by value.

Syntax

```
void avl::AddProfiles
(
 const avl::Profile& inProfile1,
 const avl::Profile& inProfile2,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Default	Description
→	inProfile1	const Profile&		First input profile
→	inProfile2	const Profile&		Second input profile
←	outProfile	Profile&		Output profile

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inProfile1** and **outProfile**, **inProfile2** and **outProfile**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Input profiles have different lengths in AddProfiles.
DomainError	Input profiles have different X coordinates in AddProfiles.

AddProfiles_OfArray

Header: AVL.h
Namespace: avl
Module: FoundationPro

Adds profiles of an array value by value.

Syntax

```
void avl::AddProfiles_OfArray
(
 const atl::Array<avl::Profile>& inProfileArray,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Default	Description
→	inProfileArray	const Array<Profile>&		
←	outProfile	Profile&		Output profile

Description

Array version of AddProfiles.

See Also

- AddProfiles – Adds two profiles value by value.

AddProfiles_OfLoop

Header: AVL.h
Namespace: avl
Module: FoundationPro

Adds profiles appearing in consecutive iterations value by value.

Syntax

```
void avl::AddProfiles_OfLoop
(
 ProfileCombinators_OfLoopState& ioState,
 const avl::Profile& inProfile,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Default	Description
↔	ioState	ProfileCombinators_OfLoopState&		Object used to maintain state of the function.
→	inProfile	const Profile&		Input profile
←	outProfile	Profile&		Output profile

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inProfile** and **outProfile**

Read more about In-place Computation.

Description

Loop version of AddProfiles.

See Also

- AddProfiles – Adds two profiles value by value.

DifferenceProfile

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes profile representing difference between corresponding values of given profiles.

Syntax

```
void avl::DifferenceProfile
(
 const avl::Profile& inProfile1,
 const avl::Profile& inProfile2,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Default	Description
→	inProfile1	const Profile&		First input profile
→	inProfile2	const Profile&		Second input profile
←	outProfile	Profile&		Output profile

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inProfile1** and **outProfile**, **inProfile2** and **outProfile**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Input profiles have different lengths in DifferenceProfile.
DomainError	Input profiles have different X coordinates in DifferenceProfile.

DivideProfiles

Header: AVL.h
Namespace: avl
Module: FoundationPro

Divides two profiles value by value.

Syntax

```
void avl::DivideProfiles
(
 const avl::Profile& inProfile1,
 const avl::Profile& inProfile2,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Default	Description
→	inProfile1	const Profile&		First input profile
→	inProfile2	const Profile&		Second input profile
←	outProfile	Profile&		Output profile

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inProfile1** and **outProfile**, **inProfile2** and **outProfile**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Input profiles have different lengths in DivideProfiles.
DomainError	Input profiles have different X coordinates in DivideProfiles.
DomainError	One of the second profile's values equals zero in DivideProfiles.

JoinProfiles

Header: AVL.h
Namespace: avl
Module: FoundationPro

Concatenates two profiles into one.

Syntax

```
void avl::JoinProfiles
(
 const avl::Profile& inProfile1,
 const avl::Profile& inProfile2,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Default	Description
→	inProfile1	const Profile&		First input profile
→	inProfile2	const Profile&		Second input profile
←	outProfile	Profile&		Output profile

MaximumProfile

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes the maximum of two profiles point by point.

Syntax

```
void avl::MaximumProfile
(
 const avl::Profile& inProfile1,
 const avl::Profile& inProfile2,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Default	Description
→	inProfile1	const Profile&		First input profile
→	inProfile2	const Profile&		Second input profile
←	outProfile	Profile&		Output profile

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inProfile1** and **outProfile**, **inProfile2** and **outProfile**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Input profiles have different lengths in MaximumProfile.
DomainError	Input profiles have different X coordinates in MaximumProfile.

MaximumProfile_OfArray

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes the maximum of profiles of an array.

Syntax

```
void avl::MaximumProfile_OfArray
(
 const atl::Array<avl::Profile>& inProfileArray,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Default	Description
→	inProfileArray	const Array<Profile>&		
←	outProfile	Profile&		Output profile

Description

Array version of MaximumProfile.

See Also

- MaximumProfile – Computes the maximum of two profiles point by point.

MaximumProfile_OfLoop

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes the maximum of profiles appearing in consecutive iterations.

Syntax

```
void avl::MaximumProfile_OfLoop
(
 ProfileCombinators_OfLoopState& ioState,
 const avl::Profile& inProfile,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Default	Description
↔	ioState	ProfileCombinators_OfLoopState&		Object used to maintain state of the function.
→	inProfile	const Profile&		Input profile
←	outProfile	Profile&		Output profile

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inProfile** and **outProfile**

Read more about In-place Computation.

Description

Loop version of MaximumProfile.

See Also

- MaximumProfile – Computes the maximum of two profiles point by point.

MinimumProfile

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes the minimum of two profiles point by point.

Syntax

```
void avl::MinimumProfile
(
 const avl::Profile& inProfile1,
 const avl::Profile& inProfile2,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Default	Description
→	inProfile1	const Profile&		First input profile
→	inProfile2	const Profile&		Second input profile
←	outProfile	Profile&		Output profile

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inProfile1** and **outProfile**, **inProfile2** and **outProfile**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Input profiles have different lengths in MinimumProfile.
DomainError	Input profiles have different X coordinates in MinimumProfile.

MinimumProfile_OfArray

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes the minimum of profiles of an array.

Syntax

```
void avl::MinimumProfile_OfArray
(
 const atl::Array<avl::Profile>& inProfileArray,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Default	Description
→	inProfileArray	const Array<Profile>&		
←	outProfile	Profile&		Output profile

Description

Array version of MinimumProfile.

See Also

- MinimumProfile – Computes the minimum of two profiles point by point.

MinimumProfile_OfLoop

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes the minimum of profiles appearing in consecutive iterations.

Syntax

```
void avl::MinimumProfile_OfLoop
(
 ProfileCombinators_OfLoopState& ioState,
 const avl::Profile& inProfile,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Default	Description
	ioState	ProfileCombinators_OfLoopState&		Object used to maintain state of the function.
	inProfile	const Profile&		Input profile
	outProfile	Profile&		Output profile

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inProfile** and **outProfile**

Read more about In-place Computation.

Description

Loop version of MinimumProfile.

See Also

- MinimumProfile – Computes the minimum of two profiles point by point.

MultiplyProfiles

Header: AVL.h
Namespace: avl
Module: FoundationPro

Multiplies two profiles value by value.

Syntax

```
void avl::MultiplyProfiles
(
 const avl::Profile& inProfile1,
 const avl::Profile& inProfile2,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Default	Description
	inProfile1	const Profile&		First input profile
	inProfile2	const Profile&		Second input profile
	outProfile	Profile&		Output profile

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inProfile1** and **outProfile**, **inProfile2** and **outProfile**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Input profiles have different lengths in MultiplyProfiles.
DomainError	Input profiles have different X coordinates in MultiplyProfiles.

MultiplyProfiles_OfArray

Header: AVL.h
Namespace: avl
Module: FoundationPro

Multiplies profiles of an array value by value.

Syntax

```
void avl::MultiplyProfiles_OfArray
(
 const atl::Array<avl::Profile>& inProfileArray,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Default	Description
→	inProfileArray	const Array<Profile>&		
←	outProfile	Profile&		Output profile

Description

Array version of MultiplyProfiles.

See Also

- MultiplyProfiles – Multiplies two profiles value by value.

MultiplyProfiles_OfLoop

Header: AVL.h
Namespace: avl
Module: FoundationPro

Multiplies profiles appearing in consecutive iterations value by value.

Syntax

```
void avl::MultiplyProfiles_OfLoop
(
 ProfileCombinators_OfLoopState& ioState,
 const avl::Profile& inProfile,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Default	Description
↔	ioState	ProfileCombinators_OfLoopState&		Object used to maintain state of the function.
→	inProfile	const Profile&		Input profile
←	outProfile	Profile&		Output profile

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inProfile** and **outProfile**

Read more about In-place Computation.

Description

Loop version of MultiplyProfiles.

See Also

- MultiplyProfiles – Multiplies two profiles value by value.

SubtractProfiles

Header: AVL.h
Namespace: avl
Module: FoundationPro

Subtracts two profiles value by value.

Syntax

```
void avl::SubtractProfiles
(
 const avl::Profile& inProfile1,
 const avl::Profile& inProfile2,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Default	Description
→	inProfile1	const Profile&		First input profile
→	inProfile2	const Profile&		Second input profile
←	outProfile	Profile&		Output profile

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inProfile1** and **outProfile**, **inProfile2** and **outProfile**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Input profiles have different lengths in SubtractProfiles.
DomainError	Input profiles have different X coordinates in SubtractProfiles.

14. Image Conversions

Table of content:

AddChannels

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a monochromatic image by summing the values of the input image channels.

Syntax

```
void avl::AddChannels
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
←	outImage	Image&		output mono image

Description

This operation sums values of the given **inImage** image channels to obtain a monochromatic image.

Examples

Image with one channel on the input of the filter (on the left) doesn't affect the image - see the image on the right.

Input RGB image (on the left) converted to single-channel image (on the right) with pixel values obtained from summing values of input image channels. For example, if a pixel from input image in each R, G and B channel has values {255, 0, 0}, corresponding pixel in the output image will have value {255} (because 255 + 0 + 0 = 255).

Hardware Acceleration

This operation is optimized for AVX2 technology for pixels of types: 2xUINT8, 3xUINT8, 4xUINT8, 2xUINT16, 3xUINT16, 4xUINT16.

This operation is optimized for SSSE3 technology for pixels of types: 2xUINT8, 3xUINT8, 4xUINT8, 2xUINT16, 3xUINT16, 4xUINT16.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in AddChannels.

See Also

- AddChannels_Saturation – Creates a monochromatic image by summing the values of the input image channels with saturation.
- AverageChannels – Creates a monochromatic image by averaging the input image channels.
- AverageChannels_Weighted – Creates a monochromatic image from weighted averages of the input image channels.

AddChannels_Saturation

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a monochromatic image by summing the values of the input image channels with saturation.

Syntax

```
void avl::AddChannels_Saturation
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
←	outImage	Image&		output mono image

Description

This operation sums with saturation values of the given **inImage** image channels to obtain a monochromatic image.

Firstly, the filter performs ordinary addition of pixel values from each channel. Then, comparison with maximal allowable value for single channel is performed. Such value depends on the pixel type - for example, if the pixel type is UInt8, maximal allowable value for each single channel is 255. The result of addition with saturation is chosen as a minimal value of computed sum and allowable maximum - for example, if the input image has three R, G, B channels and pixel type of UInt8, the result of addition with saturation is chosen as min(R + G + B, 255).

Examples

Image with one channel on the input of the filter (on the left) doesn't affect the image - see the image on the right.

Input RGB image (on the left) converted to single-channel image (on the right) with pixel values obtained from summing with saturation values of input image channels. In the picture on the left-hand side, pixel values in the colorful squares are {255, 255, 0}, {255, 0, 255} and {0, 255, 255}, and their type is UInt8. After addition with saturation, values of corresponding pixels in the output image will be {255} - because maximal value for UInt8 pixels is 255 and min(255 + 0 + 255, 255) is 255.

Hardware Acceleration

This operation is optimized for AVX2 technology for pixels of types: 2xUINT8, 3xUINT8, 4xUINT8, 2xUINT16, 3xUINT16, 4xUINT16.

This operation is optimized for SSSE3 technology for pixels of types: 2xUINT8, 3xUINT8, 4xUINT8, 2xUINT16, 3xUINT16, 4xUINT16.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in AddChannels_Saturation.

AppendImageChannel

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Appends a single valued channel to the input image.

Syntax

```
void avl::AppendImageChannel
(
 const avl::Image& inImage,
 const float inAppendedChannelValue,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inAppendedChannelValue	const float		
←	outImage	Image&		Output image

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Input image cannot have 4 channels in AppendImageChannel.

AverageChannels

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a monochromatic image by averaging the input image channels.

Syntax

```
void avl::AverageChannels
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::Image& outImage
)
```

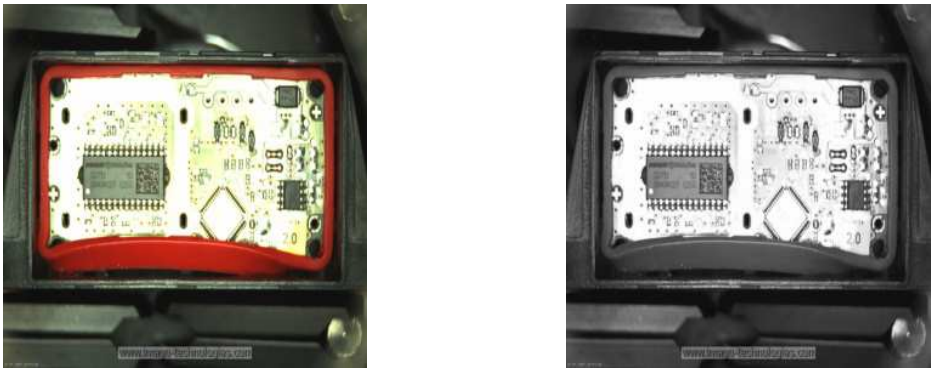
Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
←	outImage	Image&		Output image

Description

The operation computes the monochromatic average of **inImage** color channels. Average being computed at each pixel is a standard arithmetic mean.

Examples



AverageChannels *run on example image.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of type: 3xUINT8.

This operation is optimized for SSSE3 technology for pixels of types: 2xUINT8, 4xUINT8.

This operation is optimized for AVX2 technology for pixels of types: 2xUINT8, 3xUINT8, 4xUINT8.

This operation is optimized for NEON technology for pixels of type: UINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in AverageChannels.

See Also

- AverageChannels_Weighted – Creates a monochromatic image from weighted averages of the input image channels.

AverageChannels_121

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a monochromatic image by averaging the input image channels.

Applications: Conversion to mono which is more consistent with human perception (our eyes are more sensitive to green).

Syntax

```
void avl::AverageChannels_121
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
←	outImage	Image&		Output image

Requirements

For input **inImage** only pixel formats are supported: 3□uint8, 3□int8, 3□uint16, 3□int16, 3□int32, 3□real.

Read more about pixel formats in Image documentation.

Description

The operation computes the monochromatic average of **inImage** color channels. Average being computed at each pixel is a mean with weight (1, 2, 1).

Hardware Acceleration

This operation is optimized for SSSE3 technology for pixels of type: 3xUINT8.

This operation is optimized for AVX2 technology for pixels of type: 3xUINT8.

This operation is optimized for NEON technology for pixels of type: 3xUINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in AverageChannels_121.
DomainError	Not supported inImage pixel format in AverageChannels_121. Supported formats: 3xUInt8, 3xInt8, 3xUInt16, 3xInt16, 3xInt32, 3xReal.

See Also

- AverageChannels_Weighted – Creates a monochromatic image from weighted averages of the input image channels.
- AverageChannels – Creates a monochromatic image by averaging the input image channels.

AverageChannels_Weighted

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a monochromatic image from weighted averages of the input image channels.

Syntax

```
void avl::AverageChannels_Weighted
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 int inWeight1,
 int inWeight2,
 int inWeight3,
 int inWeight4,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inWeight1	int		Weight of the first channel
→	inWeight2	int		Weight of the second channel
→	inWeight3	int		Weight of the third channel
→	inWeight4	int		Weight of the fourth channel
←	outImage	Image&		Output image

Description

The operation computes the monochromatic average of the **inImage** color channels. Average being computed at each pixel is a weighted arithmetic mean.

Weights of the channels are determined by the parameters **inWeight1**, **inWeight2**, **inWeight3**, **inWeight4**. For images having less than four channels, it is required that weights assigned to non-existing channels equal zero.

Examples

AverageChannels_Weighted *run on example image with* ***inWeight1*** *= 1,* ***inWeight2*** *= 0,* ***inWeight3*** *= 0,* ***inWeight4*** *= 0, which for a RGB image is equivalent to the extraction of the Red channel.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Non-zero weight of not existing image channel in AverageChannels_Weighted.
DomainError	Region exceeds an input image in AverageChannels_Weighted.
DomainError	Sum of weights equals zero in AverageChannels_Weighted.

See Also

- AverageChannels – Creates a monochromatic image by averaging the input image channels.
- MixChannels – Calculates a linear combination of image channels.

ConvertPixelType

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Changes the type of pixel components.

Syntax

```
void avl::ConvertPixelType
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::PlainType::Type inNewType,
 int inDepthDelta,
 avl::Image& outImage
)
```

Parameters

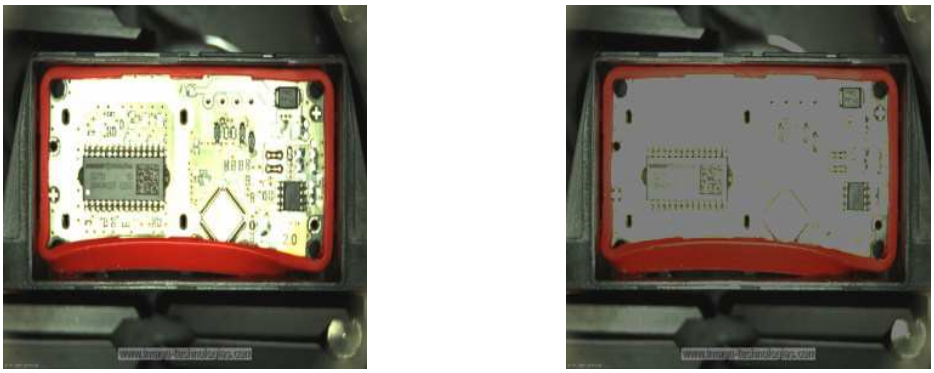
	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inNewType	PlainType::Type		UInt8	Type of the output image
→	inDepthDelta	int	-30 - 30	0	Pixel values will be multiplied by 2^inDepthDelta. For example, use -4 to convert 12-bit to 8-bit images.
←	outImage	Image&			Output image

Description

The operation alters the pixel component format of the **inImage**. Available formats are listed in the documentation of the Image data type.

If the value of pixel component doesn't fit in the range of the new type, it is clipped to the nearest proper value, which can lead to the significant loss of information, as demonstrated in the example.

Examples



ConvertPixelType *run on example image of UInt8 type with* ***inNewType*** *being Int8. Note that bright pixels suffered from clipping, while dark ones remained unaltered.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

This operation is optimized for SSE2 technology for pixels of types: UINT16 (when converting to UINT8 with negative delta).

This operation is optimized for AVX2 technology for pixels of types: UINT16 (when converting to UINT8 with negative delta).

This operation is optimized for NEON technology for pixels of types: UINT16 (when converting to UINT8 with negative delta).

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in ConvertPixelType.

ConvertToMultichannel

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a multichannel image from a monochromatic one by replicating its channel.

Syntax

```
void avl::ConvertToMultichannel
(
 const avl::Image& inMonoImage,
 atl::Optional<const avl::Region&> inRoi,
 int inNewDepth,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inMonoImage	const Image&			
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inNewDepth	int	1 - 4	3	
←	outImage	Image&			Output image

Description

The operation computes multichannel image equivalent to monochromatic **inMonoImage**. The number of channels in the resulting image is determined by the **inNewDepth** parameter. Each of the channels in **outImage** will be equal to the only channel of the **inMonoImage**.

Note that the operation does not alter the appearance of the image in the IDE.

Hardware Acceleration

This operation is optimized for SSSE3 technology for pixels of types: 3xUINT8, 3xSINT8, 3xUINT16, 3xSINT16, 3xSINT32, 3xREAL.

This operation is optimized for NEON technology for pixels of types: 3xUINT8, 3xSINT8, 3xUINT16, 3xSINT16, 3xSINT32, 3xREAL.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not a monochromatic input in ConvertToMultichannel.
DomainError	Region exceeds an input image in ConvertToMultichannel.

MaxChannels

Header: AVL.h
Namespace: avl
Module: FoundationPro

Creates a monochromatic image by taking the maximum value of all the input image channels.

Syntax

```
void avl::MaxChannels
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
←	outImage	Image&		output mono image

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of type: 2xUINT8.

This operation is optimized for SSSE3 technology for pixels of types: 3xUINT8, 4xUINT8.

This operation is optimized for AVX2 technology for pixels of types: 2xUINT8, 3xUINT8, 4xUINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in MaxChannels.

MergeChannels

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a multichannel image from several monochromatic ones.

Syntax

```
void avl::MergeChannels
(
 const avl::Image& inMonoImage1,
 const avl::Image& inMonoImage2,
 atl::Optional<const avl::Image&> inMonoImage3,
 atl::Optional<const avl::Image&> inMonoImage4,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inMonoImage1	const Image&		An image that becomes the first channel
→	inMonoImage2	const Image&		An image that becomes the second channel
→	inMonoImage3	Optional<const Image&>	NIL	An image that becomes the third channel
→	inMonoImage4	Optional<const Image&>	NIL	An image that becomes the fourth channel
←	outImage	Image&		Output image

Requirements

For input **inMonoImage1** only pixel formats are supported: 1□uint8, 1□int8, 1□uint16, 1□int16, 1□int32, 1□real.

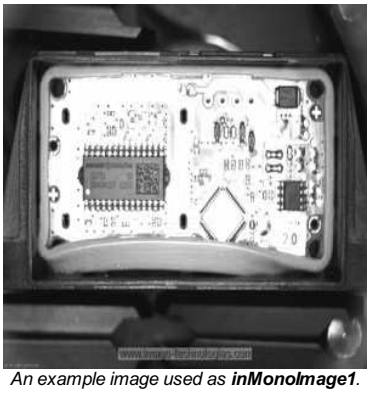
For input **inMonoImage2** only pixel formats are supported: 1□uint8, 1□int8, 1□uint16, 1□int16, 1□int32, 1□real.

Read more about pixel formats in Image documentation.

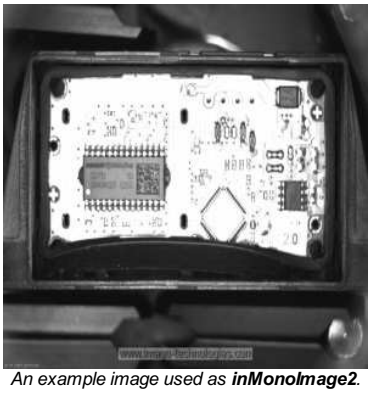
Description

The operation combines the given monochromatic images to obtain a color image, each of its channels equal to the only channel of the corresponding input image.

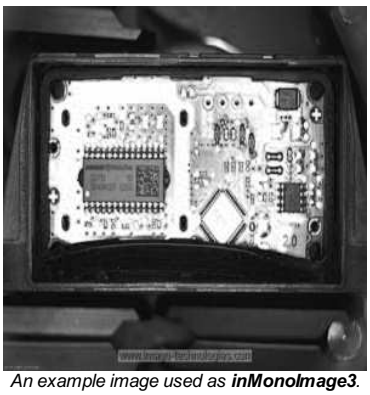
Examples



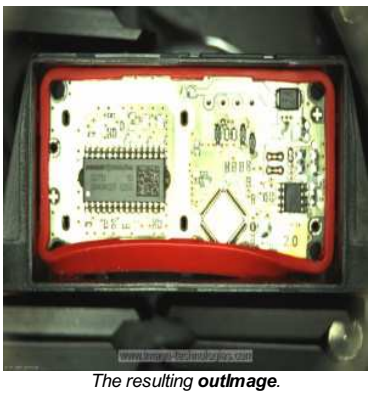
*An example image used as **inMonoImage1**.*



*An example image used as **inMonoImage2**.*



*An example image used as **inMonoImage3**.*



*The resulting **outImage**.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Image sizes are not equal in MergeChannels.
DomainError	Input image is not monochromatic in MergeChannels.
DomainError	Pixel types of the input images are not the same in MergeChannels.
DomainError	Not supported inMonoImage1 pixel format in MergeChannels. Supported formats: 1xUInt8, 1xInt8, 1xUInt16, 1xInt16, 1xInt32, 1xReal.
DomainError	Not supported inMonoImage2 pixel format in MergeChannels. Supported formats: 1xUInt8, 1xInt8, 1xUInt16, 1xInt16, 1xInt32, 1xReal.

See Also

- SplitChannels – Creates several monochromatic images from individual channels of the input image.

MinChannels

Header: AVL.h
Namespace: avl
Module: FoundationPro

Creates a monochromatic image by taking the minimum value of all the input image channels.

Syntax

```
void avl::MinChannels
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
←	outImage	Image&		output mono image

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of type: 2xUINT8.

This operation is optimized for SSSE3 technology for pixels of types: 3xUINT8, 4xUINT8.

This operation is optimized for AVX2 technology for pixels of types: 2xUINT8, 3xUINT8, 4xUINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in MinChannels.

Also in AVL Lite

MixChannels

Header: AVL.h
Namespace: avl
Module: FoundationLite

Calculates a linear combination of image channels.

Syntax

```
void avl::MixChannels
(
 const avl::Image& inImage,
 int inXCoeff,
 int inYCoeff,
 int inZCoeff,
 int inWCoeff,
 int inDivider,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inXCoeff	int			
→	inYCoeff	int			
→	inZCoeff	int			
→	inWCoeff	int			
→	inDivider	int	1 - ∞		
←	outImage	Image&			Output image

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

ReverseChannels

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Reverses the order of channels in an image.

Applications: E.g. when one has an RGB image, but needs to convert to BGR.

Syntax

```
void avl::ReverseChannels
(
 const avl::Image& inImage,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
←	outImage	Image&		Output image with reversed channels

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

Read more about In-place Computation.

Examples

Channels reversed from RGB to BGR on the Lena image.

Hardware Acceleration

This operation is optimized for SSSE3 technology for pixels of type: 3xUINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- SelectChannel – Creates an image from a single channel of the input image.
- MergeChannels – Creates a multichannel image from several monochromatic ones.

SelectChannel

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates an image from a single channel of the input image.

Syntax

```
void avl::SelectChannel
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 int inChannelIndex,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inChannelIndex	int	0 - 3		
←	outImage	Image&			Output image

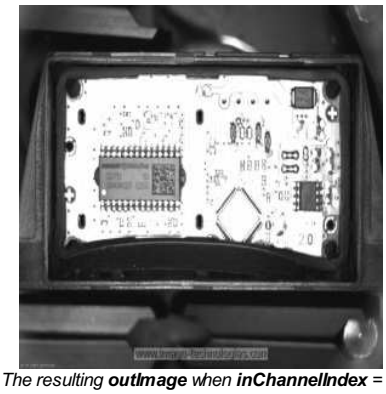
Description

The operation extracts the selected color channel of **inImage**.

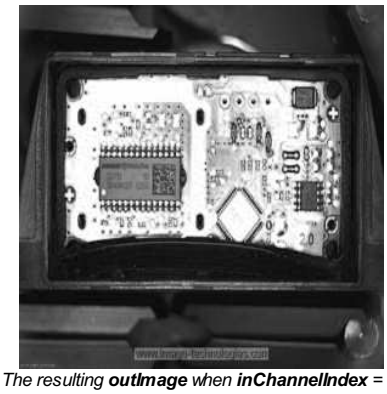
Examples

*An example image used as **inImage**.*

*The resulting **outImage** when **inChannelIndex** = 0.*



*The resulting **outImage** when **inChannelIndex** = 1.*



*The resulting **outImage** when **inChannelIndex** = 2.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Channel index out of range in SelectChannel.
DomainError	Region exceeds an input image in SelectChannel.

See Also

- SplitChannels – Creates several monochromatic images from individual channels of the input image.

SplitChannels

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates several monochromatic images from individual channels of the input image.

Syntax

```
void avl::SplitChannels
(
 const avl::Image& inImage,
 atl::Optional<avl::Image&> outMonoImage1 = atl::NIL,
 atl::Optional<avl::Image&> outMonoImage2 = atl::NIL,
 atl::Optional<avl::Image&> outMonoImage3 = atl::NIL,
 atl::Optional<avl::Image&> outMonoImage4 = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
←	outMonoImage1	Optional<Image&>	NIL	Image of the first channel
←	outMonoImage2	Optional<Image&>	NIL	Image of the second channel (if exists)
←	outMonoImage3	Optional<Image&>	NIL	Image of the third channel (if exists)
←	outMonoImage4	Optional<Image&>	NIL	Image of the fourth channel (if exists)

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outMonoImage1**, **outMonoImage2**, **outMonoImage3**, **outMonoImage4**.

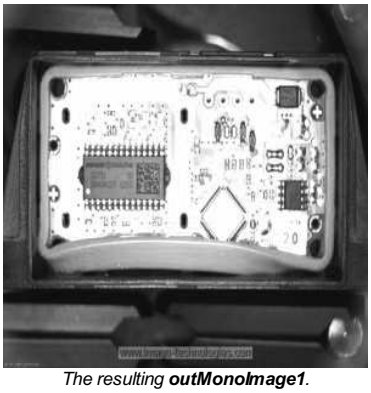
Read more about Optional Outputs.

Description

The operation extracts the color channels of **inImage** as separate monochromatic images. Outputs corresponding to non-existing channels of an image are set to empty images.

Examples

*An example image used as **inImage**.*



*The resulting **outMonoImage1**.*

*The resulting **outMonoImage2**.*

*The resulting **outMonoImage3**.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- SelectChannel – Creates an image from a single channel of the input image.
- MergeChannels – Creates a multichannel image from several monochromatic ones.
- SplitChannels_OrNil – Creates several monochromatic images from individual channels of the input image and sets Nil for channels that are not present.

SplitChannels_OrNil

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates several monochromatic images from individual channels of the input image and sets Nil for channels that are not present.

Syntax

```
void avl::SplitChannels_OrNil
(
 const avl::Image& inImage,
 atl::Optional<atl::Conditional<avl::Image>&> outMonoImage1 = atl::NIL,
 atl::Optional<atl::Conditional<avl::Image>&> outMonoImage2 = atl::NIL,
 atl::Optional<atl::Conditional<avl::Image>&> outMonoImage3 = atl::NIL,
 atl::Optional<atl::Conditional<avl::Image>&> outMonoImage4 = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
←	outMonoImage1	Optional<Conditional<Image>&>	NIL	Image of the first channel
←	outMonoImage2	Optional<Conditional<Image>&>	NIL	Image of the second channel (if exists)
←	outMonoImage3	Optional<Conditional<Image>&>	NIL	Image of the third channel (if exists)
←	outMonoImage4	Optional<Conditional<Image>&>	NIL	Image of the fourth channel (if exists)

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outMonoImage1**, **outMonoImage2**, **outMonoImage3**, **outMonoImage4**.

Read more about Optional Outputs.

Description

The operation extracts the color channels of **inImage** as separate monochromatic images. Outputs corresponding to non-existing channels of an image are set to Nil.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- SelectChannel – Creates an image from a single channel of the input image.
- MergeChannels – Creates a multichannel image from several monochromatic ones.
- SplitChannels – Creates several monochromatic images from individual channels of the input image.

15. Histogram Combinators

Table of content:

AddHistograms

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Adds two histograms bar by bar.

Syntax

```
void avl::AddHistograms
(
 const avl::Histogram& inHistogram1,
 const avl::Histogram& inHistogram2,
 avl::Histogram& outHistogram
)
```

Parameters

	Name	Type	Default	Description
→	inHistogram1	const Histogram&		Input histogram1
→	inHistogram2	const Histogram&		Input histogram2
←	outHistogram	Histogram&		Output histogram

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inHistogram1** and **outHistogram**, **inHistogram2** and **outHistogram**

Read more about In-place Computation.

Description

The operation computes the sum of two histograms. That is, each bin of the resulting histogram equals the sum of corresponding bins of the input histograms.

Errors

List of possible exceptions:

Error type	Description
DomainError	Input histograms formats are not the same in AddHistograms.

AddHistograms_OfArray

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Adds histograms of an array bar by bar.

Syntax

```
void avl::AddHistograms_OfArray
(
 const atl::Array<avl::Histogram>& inHistogramArray,
 avl::Histogram& outHistogram
)
```

Parameters

	Name	Type	Default	Description
→	inHistogramArray	const Array<Histogram>&		
←	outHistogram	Histogram&		Output histogram

Description

Array version of AddHistograms.

See Also

- AddHistograms – Adds two histograms bar by bar.

AddHistograms_OfLoop

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Adds histograms appearing in consecutive iterations bar by bar.

Syntax

```
void avl::AddHistograms_OfLoop
(
 HistogramCombinators_OfLoopState& ioState,
 const avl::Histogram& inHistogram,
 avl::Histogram& outHistogram
)
```

Parameters

	Name	Type	Default	Description
	ioState	HistogramCombinators_OfLoopState&		Object used to maintain state of the function.
	inHistogram	const Histogram&		Input histogram
	outHistogram	Histogram&		Output histogram

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inHistogram** and **outHistogram**

Read more about In-place Computation.

Description

Loop version of AddHistograms.

See Also

- AddHistograms – Adds two histograms bar by bar.

DifferenceHistogram

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes histogram representing difference between corresponding bars of given histograms.

Syntax

```
void avl::DifferenceHistogram
(
 const avl::Histogram& inHistogram1,
 const avl::Histogram& inHistogram2,
 avl::Histogram& outHistogram
)
```

Parameters

	Name	Type	Default	Description
	inHistogram1	const Histogram&		Input histogram1
	inHistogram2	const Histogram&		Input histogram2
	outHistogram	Histogram&		Output histogram

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inHistogram1** and **outHistogram**, **inHistogram2** and **outHistogram**

Read more about In-place Computation.

Description

The operation computes the absolute difference between histograms. That is, each bin of the resulting histogram equals the absolute value of difference between corresponding bins of the input histograms.

Errors

List of possible exceptions:

Error type	Description
DomainError	Input histograms formats are not the same in DifferenceHistogram.

DivideHistograms

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Divides two histograms bar by bar.

Syntax

```
void avl::DivideHistograms
(
 const avl::Histogram& inHistogram1,
 const avl::Histogram& inHistogram2,
 avl::Histogram& outHistogram
)
```

Parameters

	Name	Type	Default	Description
→	inHistogram1	const Histogram&		Input histogram1
→	inHistogram2	const Histogram&		Input histogram2
←	outHistogram	Histogram&		Output histogram

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inHistogram1** and **outHistogram**, **inHistogram2** and **outHistogram**

Read more about In-place Computation.

Description

The operation computes the quotient of two histograms. That is, each bin of the resulting histogram equals the quotient of corresponding bins of the input histograms.

Errors

List of possible exceptions:

Error type	Description
DomainError	Input histograms formats are not the same in DivideHistograms.
DomainError	One of inHistogram2 bins equals zero in DivideHistograms.

MaximumHistogram

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the maximum of two histograms bar by bar.

Syntax

```
void avl::MaximumHistogram
(
 const avl::Histogram& inHistogram1,
 const avl::Histogram& inHistogram2,
 avl::Histogram& outHistogram
)
```

Parameters

	Name	Type	Default	Description
→	inHistogram1	const Histogram&		Input histogram1
→	inHistogram2	const Histogram&		Input histogram2
←	outHistogram	Histogram&		Output histogram

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inHistogram1** and **outHistogram**, **inHistogram2** and **outHistogram**

Read more about In-place Computation.

Description

The operation computes the maximum of two histograms. That is, each bin of the resulting histogram equals the maximum of corresponding bins of the input histograms.

Errors

List of possible exceptions:

Error type	Description
DomainError	Input histograms formats are not the same in MaximumHistogram.

MaximumHistogram_OfArray

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the maximum of histograms of an array bar by bar.

Syntax

```
void avl::MaximumHistogram_OfArray
(
 const atl::Array<avl::Histogram>& inHistogramArray,
 avl::Histogram& outHistogram
)
```

Parameters

	Name	Type	Default	Description
→	inHistogramArray	const Array<Histogram>&		
←	outHistogram	Histogram&		Output histogram

Description

Array version of MaximumHistogram.

See Also

- MaximumHistogram – Computes the maximum of two histograms bar by bar.

MaximumHistogram_OfLoop

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the maximum of histograms appearing in consecutive iterations bar by bar.

Syntax

```
void avl::MaximumHistogram_OfLoop
(
 HistogramCombinators_OfLoopState& ioState,
 const avl::Histogram& inHistogram,
 avl::Histogram& outHistogram
)
```

Parameters

	Name	Type	Default	Description
	ioState	HistogramCombinators_OfLoopState&		Object used to maintain state of the function.
	inHistogram	const Histogram&		Input histogram
	outHistogram	Histogram&		Output histogram

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inHistogram** and **outHistogram**

Read more about In-place Computation.

Description

Loop version of MaximumHistogram.

See Also

- MaximumHistogram – Computes the maximum of two histograms bar by bar.

MinimumHistogram

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the minimum of two histograms bar by bar.

Syntax

```
void avl::MinimumHistogram
(
 const avl::Histogram& inHistogram1,
 const avl::Histogram& inHistogram2,
 avl::Histogram& outHistogram
)
```

Parameters

	Name	Type	Default	Description
	inHistogram1	const Histogram&		Input histogram1
	inHistogram2	const Histogram&		Input histogram2
	outHistogram	Histogram&		Output histogram

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inHistogram1** and **outHistogram**, **inHistogram2** and **outHistogram**

Read more about In-place Computation.

Description

The operation computes the minimum of two histograms. That is, each bin of the resulting histogram equals the minimum of corresponding bins of the input histograms.

Errors

List of possible exceptions:

Error type	Description
DomainError	Input histograms formats are not the same in MinimumHistogram.

MinimumHistogram_OfArray

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the minimum of histograms of an array bar by bar.

Syntax

```
void avl::MinimumHistogram_OfArray
(
 const atl::Array<avl::Histogram>& inHistogramArray,
 avl::Histogram& outHistogram
)
```

Parameters

	Name	Type	Default	Description
→	inHistogramArray	const Array<Histogram>&		
←	outHistogram	Histogram&		Output histogram

Description

Array version of MinimumHistogram.

See Also

- MinimumHistogram – Computes the minimum of two histograms bar by bar.

MinimumHistogram_OfLoop

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the minimum of histograms appearing in consecutive iterations bar by bar.

Syntax

```
void avl::MinimumHistogram_OfLoop
(
 HistogramCombinators_OfLoopState& ioState,
 const avl::Histogram& inHistogram,
 avl::Histogram& outHistogram
)
```

Parameters

	Name	Type	Default	Description
↔	ioState	HistogramCombinators_OfLoopState&		Object used to maintain state of the function.
→	inHistogram	const Histogram&		Input histogram
←	outHistogram	Histogram&		Output histogram

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inHistogram** and **outHistogram**

Read more about In-place Computation.

Description

Loop version of MinimumHistogram.

See Also

- MinimumHistogram – Computes the minimum of two histograms bar by bar.

MultiplyHistograms

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Multiplies two histograms bar by bar.

Syntax

```
void avl::MultiplyHistograms
(
 const avl::Histogram& inHistogram1,
 const avl::Histogram& inHistogram2,
 avl::Histogram& outHistogram
)
```

Parameters

	Name	Type	Default	Description
→	inHistogram1	const Histogram&		Input histogram1
→	inHistogram2	const Histogram&		Input histogram2
←	outHistogram	Histogram&		Output histogram

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inHistogram1** and **outHistogram**, **inHistogram2** and **outHistogram**

Read more about In-place Computation.

Description

The operation computes the product of two histograms. That is, each bin of the resulting histogram equals the product of corresponding bins of the input histograms.

Errors

List of possible exceptions:

Error type	Description
DomainError	Input histograms formats are not the same in MultiplyHistograms.

MultiplyHistograms_OfArray

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Multiply histograms of an array bar by bar.

Syntax

```
void avl::MultiplyHistograms_OfArray
(
 const atl::Array<avl::Histogram>& inHistogramArray,
 avl::Histogram& outHistogram
)
```

Parameters

	Name	Type	Default	Description
→	inHistogramArray	const Array<Histogram>&		
←	outHistogram	Histogram&		Output histogram

Description

Array version of MultiplyHistograms.

See Also

- MultiplyHistograms – Multiplies two histograms bar by bar.

MultiplyHistograms_OfLoop

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Multiply histograms appearing in consecutive iterations bar by bar.

Syntax

```
void avl::MultiplyHistograms_OfLoop
(
 HistogramCombinators_OfLoopState& ioState,
 const avl::Histogram& inHistogram,
 avl::Histogram& outHistogram
)
```

Parameters

	Name	Type	Default	Description
	ioState	HistogramCombinators_OfLoopState&		Object used to maintain state of the function.
	inHistogram	const Histogram&		Input histogram
	outHistogram	Histogram&		Output histogram

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inHistogram** and **outHistogram**

Read more about In-place Computation.

Description

Loop version of MultiplyHistograms.

See Also

- MultiplyHistograms – Multiplies two histograms bar by bar.

SubtractHistograms

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Subtracts two histograms bar by bar.

Syntax

```
void avl::SubtractHistograms
(
 const avl::Histogram& inHistogram1,
 const avl::Histogram& inHistogram2,
 avl::Histogram& outHistogram
)
```

Parameters

	Name	Type	Default	Description
	inHistogram1	const Histogram&		Input histogram1
	inHistogram2	const Histogram&		Input histogram2
	outHistogram	Histogram&		Output histogram

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inHistogram1** and **outHistogram**, **inHistogram2** and **outHistogram**

Read more about In-place Computation.

Description

The operation subtracts two histograms. That is, each bin of the resulting histogram equals the difference of corresponding bins of the input histograms.

Errors

List of possible exceptions:

Error type	Description
DomainError	Input histograms formats are not the same in SubtractHistograms.

16. Image Combinators

Table of content:

AddImages

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Adds two images pixel by pixel.

Syntax

```
void avl::AddImages
(
 const avl::Image& inImage1,
 const avl::Image& inImage2,
 atl::Optional<const avl::Region&> inRoi,
 float inScale,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage1	const Image&		First input image
→	inImage2	const Image&		Second input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inScale	float	1.0f	Output image scaling factor
←	outImage	Image&		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage1** and **outImage**, **inImage2** and **outImage**

Read more about In-place Computation.

Description

The operation computes the sum of two images. Each **outImage** pixel is equal to the sum of the corresponding pixels of the input images.

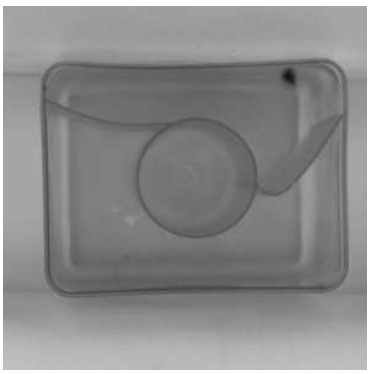
$$\forall_{i,j} \mathbf{OutImage}_{i,j} = (\mathbf{InImage1}_{i,j} + \mathbf{InImage2}_{i,j}) \cdot \mathbf{InScale}$$

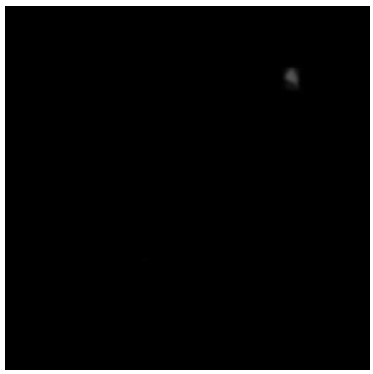
The result of the operation is multiplied by **inScale** factor, which may be used to avoid clipping when the produced values exceed the range of correct pixel values. However it should be noted that when the value of **inScale** is other than 1.0, the filter uses a slower, not SSE-optimized implementation.

Whenever the resulting value exceeds the range of pixel values, it is clipped to the nearest proper value. In multichannel (color) images each pixel channel is processed separately.

The operation requires that the images being processed have equal format and dimensions, otherwise an error with appropriate description occurs. To obtain an image of desired dimensions one can use ResizeImage or CropImage filter. To alter the pixel type of an image one can use ConvertPixelType filter. ConvertToMultichannel and AverageChannels filters allow to alter the number of image channels.

Examples





AddImages *performed on the sample images.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT8, SINT16, REAL, SINT8(for inScale=1), UINT16(for inScale=1).

This operation is optimized for SSE41 technology for pixels of types: UINT16(for inScale!=1).

This operation is optimized for AVX2 technology for pixels of types: UINT8, SINT8, SINT16, UINT16, REAL.

This operation is optimized for NEON technology for pixels of types: UINT8, SINT8, UINT16, SINT16, REAL(for inScale!=1), SINT32(for inScale!=1).

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Image formats are not the same in AddImages.
DomainError	Image sizes are not equal in AddImages.
DomainError	Region exceeds an input image in AddImages.

See Also

- SubtractImages – Subtracts two images pixel by pixel. The result is signed.

AddImages_OfArray

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Adds images of an array pixel by pixel.

Syntax

```
void avl::AddImages_OfArray
(
 const atl::Array<avl::Image>& inImageArray,
 atl::Optional<const avl::Region&> inRoi,
 float inScale,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImageArray	const Array<Image>&		
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inScale	float	1.0f	
←	outImage	Image&		Output image

Description

Array version of AddImages.

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT8, SINT16, REAL, SINT8(for inScale=1), UINT16(for inScale=1).

This operation is optimized for SSE41 technology for pixels of types: UINT16(for inScale!=1).

This operation is optimized for AVX2 technology for pixels of types: UINT8, SINT8, SINT16, UINT16, REAL.

This operation is optimized for NEON technology for pixels of types: UINT8, SINT8, UINT16, SINT16, REAL, SINT32(for inScale!=1).

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- AddImages – Adds two images pixel by pixel.

AddImages_OfLoop

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Adds images appearing in consecutive iterations pixel by pixel.

Syntax

```
void avl::AddImages_OfLoop
(
 ImageCombinators_OfLoopState& ioState,
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 float inScale,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
	ioState	ImageCombinators_OfLoopState&		Object used to maintain state of the function.
	inImage	const Image&		Input image
	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
	inScale	float	1.0f	
	outImage	Image&		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

Read more about In-place Computation.

Description

Loop version of AddImages.

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT8, SINT16, REAL, SINT8(for inScale=1), UINT16(for inScale=1).

This operation is optimized for SSE41 technology for pixels of types: UINT16(for inScale!=1).

This operation is optimized for AVX2 technology for pixels of types: UINT8, SINT8, SINT16, UINT16, REAL.

This operation is optimized for NEON technology for pixels of types: UINT8, SINT8, UINT16, SINT16, REAL, SINT32(for inScale!=1).

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- AddImages – Adds two images pixel by pixel.

AverageImages

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Averages two images pixel by pixel.

Syntax

```
void avl::AverageImages
(
 const avl::Image& inImage1,
 const avl::Image& inImage2,
 atl::Optional<const avl::Region&> inRoi,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
	inImage1	const Image&		First input image
	inImage2	const Image&		Second input image
	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
	outImage	Image&		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage1** and **outImage**, **inImage2** and **outImage**

Read more about In-place Computation.

Description

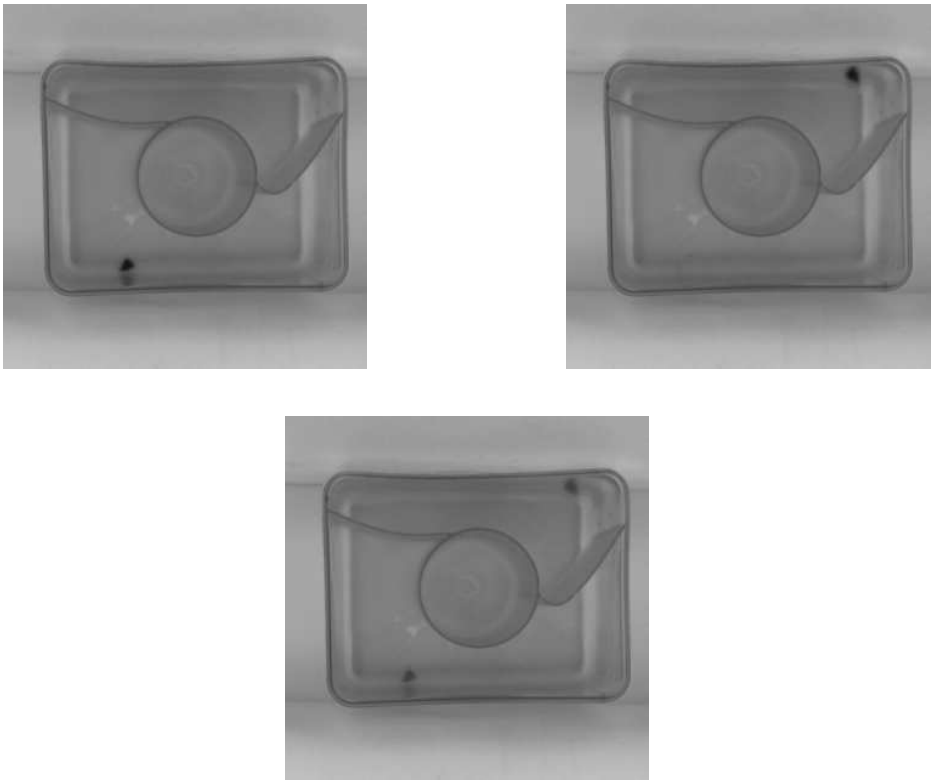
The operation computes the average of two images. Each **outImage** pixel is equal to the average of the corresponding pixels of the input images.

$$\forall_{i,j} \mathbf{OutImage}_{i,j} = \frac{\mathbf{InImage1}_{i,j} + \mathbf{InImage2}_{i,j}}{2}$$

In multichannel (color) images each pixel channel is processed separately.

The operation requires that the images being processed have equal format and dimensions, otherwise an error with appropriate description occurs. To obtain an image of desired dimensions one can use ResizeImage or CropImage filter. To alter the pixel type of an image one can use ConvertPixelType filter. ConvertToMultichannel and AverageChannels filters allow to alter the number of image channels.

Examples



AverageImages *performed on the sample images.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT16, UINT16, REAL.

This operation is optimized for AVX2 technology for pixels of types: UINT8, SINT16, UINT16, REAL.

This operation is optimized for NEON technology for pixels of types: UINT8, SINT8, UINT16, SINT16, SINT32.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Image formats are not the same in AverageImages.
DomainError	Image sizes are not equal in AverageImages.
DomainError	Region exceeds an input image in AverageImages.

See Also

- BlendImages – Computes weighted sum pixel by pixel.
- LerpImages – Interpolates two images linearly pixel by pixel.

AverageImages_OfArray

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Averages images from an array pixel by pixel.

Syntax

```
void avl::AverageImages_OfArray
(
 const atl::Array<avl::Image>& inImages,
 atl::Optional<const avl::Region&> inRoi,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImages	const Array<Image>&		
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
←	outImage	Image&		Output image

Description

Array version of AverageImages.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array of images in AverageImages_OfArray.
DomainError	Image dimensions are not equal in AverageImages_OfArray.
DomainError	Image formats are not the same in AverageImages_OfArray.
DomainError	Not supported image type in AverageImages_OfArray.
DomainError	Region exceeds an input image in AverageImages_OfArray.

See Also

- AverageImages – Averages two images pixel by pixel.

AverageImages_OfLoop

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Averages images appearing in consecutive iterations pixel by pixel.

Syntax

```
void avl::AverageImages_OfLoop
(
 AverageImages_OfLoopState& ioState,
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<avl::Image&> outImage
)
```

Parameters

	Name	Type	Default	Description
	ioState	AverageImages_OfLoopState&		Object used to maintain state of the function.
	inImage	const Image&		Input image
	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
	outImage	Optional<Image&>		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

Read more about In-place Computation.

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outImage**.

Read more about Optional Outputs.

Description

Loop version of AverageImages.

Errors

List of possible exceptions:

Error type	Description
DomainError	Image dimensions are not equal in AverageImages_OfLoop.
DomainError	Image formats are not the same in AverageImages_OfLoop.
DomainError	Region exceeds an input image in AverageImages_OfLoop.

See Also

- AverageImages – Averages two images pixel by pixel.

BlendImages

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes weighted sum pixel by pixel.

Syntax

```
void avl::BlendImages
(
 const avl::Image& inImage1,
 const avl::Image& inImage2,
 atl::Optional<const avl::Region&> inRoi,
 float inAmount1,
 float inAmount2,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage1	const Image&			First input image
→	inImage2	const Image&			Second input image
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inAmount1	float	$-\infty$ - ∞	0.5f	
→	inAmount2	float	$-\infty$ - ∞	0.5f	
←	outImage	Image&			Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage1** and **outImage**, **inImage2** and **outImage**

Read more about In-place Computation.

Description

The operation computes the weighted sum of the corresponding pixels of the input images. Note that the sum of selected weight need not to equal 1.0, but only when the sum is less or equal to 1.0 the resulting value is guaranteed to fit in the range of pixel values.

$$\forall_{i,j} \mathbf{OutImage}_{i,j} = \mathbf{InAmount1} \cdot \mathbf{InImage1}_{i,j} + \mathbf{InAmount2} \cdot \mathbf{InImage2}_{i,j}$$

Whenever the resulting value exceeds the range of pixel values, it is clipped to the nearest proper value. In multichannel (color) images each pixel channel is processed separately.

The operation requires that the images being processed have equal format and dimensions, otherwise an error with appropriate description occurs. To obtain an image of desired dimensions one can use ResizeImage or CropImage filter. To alter the pixel type of an image one can use ConvertPixelType filter. ConvertToMultichannel and AverageChannels filters allow to alter the number of image channels.

Examples

BlendImages *performed on the sample images with* ***inAmount1*** *= 0.3,* ***inAmount2*** *= 0.7.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT16, REAL.

This operation is optimized for AVX2 technology for pixels of types: UINT8, SINT16, REAL.

This operation is optimized for NEON technology for pixels of types: UINT8, SINT8, UINT16, SINT16, SINT32.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Image formats are not the same in BlendImages.
DomainError	Image sizes are not equal in BlendImages.
DomainError	Region exceeds an input image in BlendImages.

See Also

- AverageImages – Averages two images pixel by pixel.
- LerpImages – Interpolates two images linearly pixel by pixel.

ComposeImages

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Takes pixels from the first image within the specified region and from the other one elsewhere.

Syntax

```
void avl::ComposeImages
(
 const avl::Image& inImage1,
 const avl::Image& inImage2,
 const avl::Region& inRegion,
 atl::Optional<const avl::Region&> inRoi,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
➡	inImage1	const Image&		First source of pixels marked in inRegion
➡	inImage2	const Image&		Background image
➡	inRegion	const Region&		Region which describes which pixels should be taken from inImage1
➡	inRoi	Optional<const Region&>	NIL	Region in which pixels should be processed otherwise black pixels are placed
⬅	outImage	Image&		Image composed of tho input images

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage1** and **outImage**, **inImage2** and **outImage**

Read more about In-place Computation.

Description

The operation computes the composition of two images. Each **outImage** pixel is equal to the corresponding pixel of **inImage1** iff its location lies inside the **inRegion**; otherwise the pixel is equal to the corresponding pixel of the **inImage2**.

$$\forall_{i,j}\mathbf{OutImage}_{i,j} = \begin{cases} \mathbf{InImage1}_{i,j} & \text{if } (i,j) \in \mathbf{InRegion} \\ \mathbf{InImage2}_{i,j} & \text{otherwise} \end{cases}$$

The operation requires that the images being processed have equal format and dimensions, otherwise an error with appropriate description occurs. To obtain an image of desired dimensions one can use ResizeImage or CropImage filter. To alter the pixel type of an image one can use ConvertPixelType filter. ConvertToMultichannel and AverageChannels filters allow to alter the number of image channels.

Examples

*Example **inImage1** and **inImage2**.*

*Example **inRegion** and the result of **ComposeImages***

Errors

List of possible exceptions:

Error type	Description
DomainError	Image formats are not the same in ComposeImages.
DomainError	Image sizes are not equal in ComposeImages.
DomainError	Input region exceeds an input image in ComposeImages.
DomainError	Input ROI exceeds an input image in ComposeImages.

See Also

- JoinImages – Creates a single image by glueing together the two input images in horizontal or vertical direction.
- DrawGridImage – Draws an image as a tile on an image considered to be a grid of tiles.

DifferenceImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the non-negative distances between corresponding pixel values.

Applications: Useful for things like comparing an image against a template or for detecting differences between consecutive video frames.

Syntax

```
void avl::DifferenceImage
(
 const avl::Image& inImage1,
 const avl::Image& inImage2,
 atl::Optional<const avl::Region&> inRoi,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage1	const Image&		First input image
→	inImage2	const Image&		Second input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
←	outImage	Image&		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage1** and **outImage**, **inImage2** and **outImage**

Read more about In-place Computation.

Description

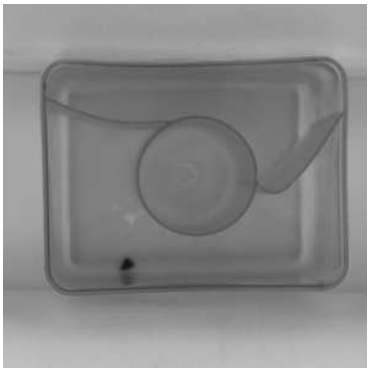
The operation computes the absolute difference between images. Each **outImage** pixel is equal to the absolute value of difference between corresponding pixels of the input images.

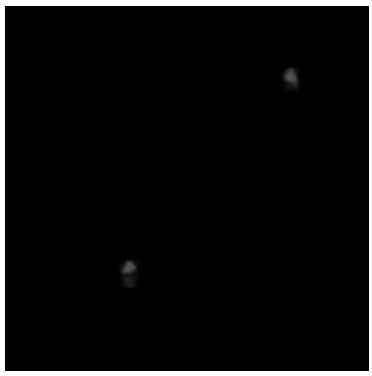
$$\forall_{i,j} \mathbf{OutImage}_{i,j} = |\mathbf{InImage1}_{i,j} - \mathbf{InImage2}_{i,j}|$$

In multichannel (color) images each pixel channel is processed separately.

The operation requires that the images being processed have equal format and dimensions, otherwise an error with appropriate description occurs. To obtain an image of desired dimensions one can use ResizeImage or CropImage filter. To alter the pixel type of an image one can use ConvertPixelType filter. ConvertToMultichannel and AverageChannels filters allow to alter the number of image channels.

Examples





DifferenceImage *performed on the sample images.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT8, UINT16, SINT16, REAL.

This operation is optimized for AVX2 technology for pixels of types: UINT8, SINT8, UINT16, SINT16, REAL.

This operation is optimized for NEON technology for pixels of types: UINT8, SINT8, UINT16, SINT16, SINT32, REAL.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Image formats are not the same in DifferenceImage.
DomainError	Image sizes are not equal in DifferenceImage.
DomainError	Region exceeds an input image in DifferenceImage.

See Also

- SubtractImages – Subtracts two images pixel by pixel. The result is signed.
- DifferenceImage_Shifted – Computes the non-negative distances between corresponding pixel values.
- DifferenceImage_Flex – Computes the non-negative distances between corresponding pixel values using tiles.

DifferenceImage_Flex

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes the non-negative distances between corresponding pixel values using tiles.

Applications: Useful for things like comparing an image against a template or for detecting differences between consecutive video frames.

Syntax

```
void avl::DifferenceImage_Flex
(
 const avl::Image& inImage1,
 const avl::Image& inImage2,
 atl::Optional<const avl::Box&> inBox1,
 atl::Optional<const avl::Box&> inBox2,
 int inTileWidth,
 atl::Optional<int> inTileHeight,
 avl::TileTranslationMode::Type inTileTranslationMode,
 const bool inOutputFromColorImage,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage1	const Image&			First input image
→	inImage2	const Image&			Second input image
→	inBox1	Optional<const Box&>		NIL	
→	inBox2	Optional<const Box&>		NIL	
→	inTileWidth	int	1 - ∞	16	
→	inTileHeight	Optional<int>	1 - ∞	NIL	
→	inTileTranslationMode	TileTranslationMode::Type		FourDirections	
→	inOutputFromColorImage	const bool		False	Flag indicating whether to use every channel of the input images separately to compute results or only channels average
←	outImage	Image&			Output image

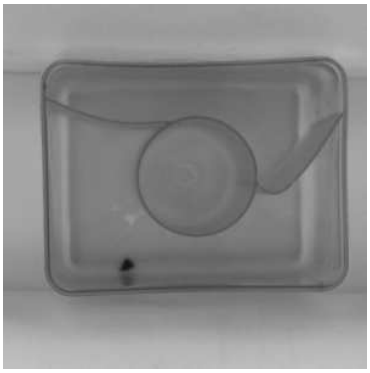
Description

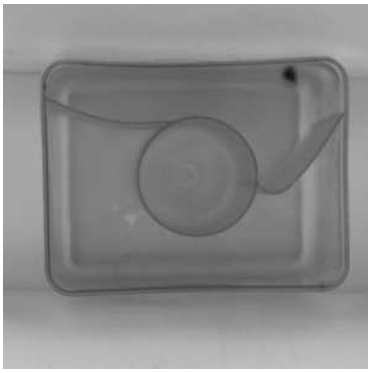
The operation computes the absolute difference between images. First the input images are divided into tiles. Each tile is then considered separately. It is translated by a minimal vector in one of the number of directions (depending on **inTileTranslationMode**, 4 or 8) and the translation with minimum overall difference between input images in this tile is considered to be the proper one. Finally, the output image values in the tile are computed using the so computed translation and the same formula like in DifferenceImage_Shifted.

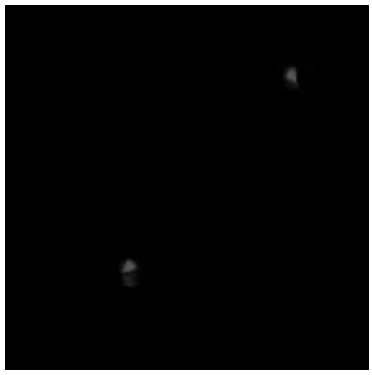
In multichannel (color) images each pixel channel is processed separately.

The operation requires that the images being processed have equal format and dimensions, otherwise an error with appropriate description occurs. To obtain an image of desired dimensions one can use ResizeImage or CropImage filter. To alter the pixel type of an image one can use ConvertPixelType filter. ConvertToMultichannel and AverageChannels filters allow to alter the number of image channels.

Examples







*DifferenceImage_Flex performed on the sample images with **inTileWidth** = **inTileHeight** = 16 and **inTileTranslationMode** = EightDirections.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Image formats are not the same in DifferenceImage_Flex.
DomainError	Image fragments dimensions are not equal in DifferenceImage_Flex.
DomainError	Input box exceeds image dimensions in DifferenceImage_Flex.

See Also

- DifferenceImage – Computes the non-negative distances between corresponding pixel values.
- DifferenceImage_Shifted – Computes the non-negative distances between corresponding pixel values.
- SubtractImages – Subtracts two images pixel by pixel. The result is signed.

DifferenceImage_Shifted

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the non-negative distances between corresponding pixel values.

Applications: Useful for things like comparing an image against a template or for detecting differences between consecutive video frames.

Syntax

```
void avl::DifferenceImage_Shifted
(
 const avl::Image& inImage1,
 const avl::Image& inImage2,
 const avl::Box& inBox1,
 const avl::Box& inBox2,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage1	const Image&		First input image
→	inImage2	const Image&		Second input image
→	inBox1	const Box&		
→	inBox2	const Box&		
←	outImage	Image&		Output image

Description

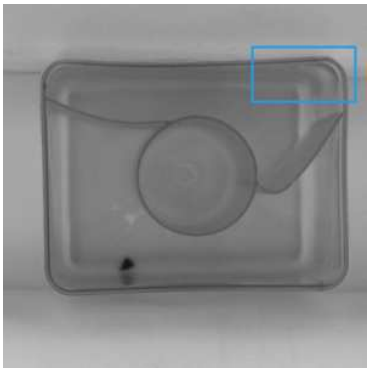
The operation computes the absolute difference between image pixels inside the input boxes. Each **outImage** pixel is equal to the absolute value of difference between corresponding pixels of the input images:

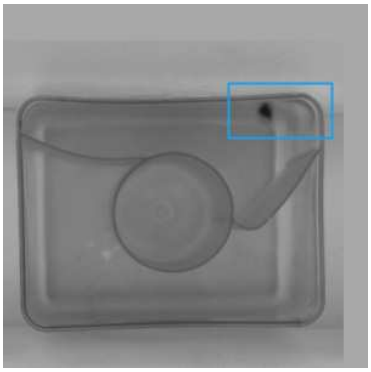
$$\forall_{i,j} \mathrm{OutImage}_{i,j} = |\mathrm{InImage1}_{i+box1.x,j+box1.y} - \mathrm{InImage2}_{i+box2.x,j+box2.y}|$$

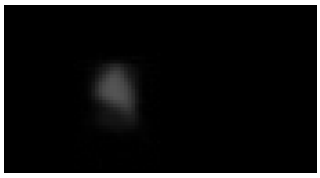
In multichannel (color) images each pixel channel is processed separately.

The operation requires that the images being processed have equal format and the input boxes have equal dimensions, otherwise an error with appropriate description occurs. To alter the pixel type of an image one can use ConvertPixelType filter. ConvertToMultichannel and AverageChannels filters allow to alter the number of image channels.

Examples







DifferenceImage_Shifted *performed on the sample images.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT8, UINT16, SINT16, REAL.

This operation is optimized for AVX2 technology for pixels of types: UINT8, SINT8, UINT16, SINT16, REAL.

This operation is optimized for NEON technology for pixels of types: UINT8, SINT8, UINT16, SINT16, SINT32, REAL.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Image formats are not the same in DifferenceImage_Shifted.
DomainError	Input boxes are not completely inside the input images in DifferenceImage_Shifted.
DomainError	Not equal box dimensions in DifferenceImage_Shifted.

See Also

- DifferenceImage – Computes the non-negative distances between corresponding pixel values.
- DifferenceImage_Flex – Computes the non-negative distances between corresponding pixel values using tiles.
- SubtractImages – Subtracts two images pixel by pixel. The result is signed.

DivideImages

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Divides two images pixel by pixel.

Applications: Can be used for flat field correction.

Syntax

```
void avl::DivideImages
(
 const avl::Image& inImage1,
 const avl::Image& inImage2,
 atl::Optional<const avl::Region&> inRoi,
 float inScale,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage1	const Image&		First input image
→	inImage2	const Image&		Second input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inScale	float	1.0f	Output image scaling factor
←	outImage	Image&		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage1** and **outImage**, **inImage2** and **outImage**

Read more about In-place Computation.

Description

The operation computes the quotient of two images. Each **outImage** pixel is equal to the quotient of the corresponding pixels of the input images.

$$\forall_{i,j}\mathbf{OutImage}_{i,j} = \left(\frac{\mathbf{InImage1}_{i,j}}{\mathbf{InImage2}_{i,j}}\right) \cdot \mathbf{InScale}$$

The result of the operation is multiplied by **inScale** factor, which may be used to avoid clipping when the produced values exceed the range of correct pixel values. However it should be noted that when the value of **inScale** is other than 1.0, the filter uses a slower, not SSE-optimized implementation.

In multichannel (color) images each pixel channel is processed separately.

The operation requires that the images being processed have equal format and dimensions, otherwise an error with appropriate description occurs. To obtain an image of desired dimensions one can use ResizeImage or CropImage filter. To alter the pixel type of an image one can use ConvertPixelType filter. ConvertToMultichannel and AverageChannels filters allow to alter the number of image channels.

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT16, REAL.

This operation is optimized for SSE41 technology for pixels of type: UINT16.

This operation is optimized for AVX2 technology for pixels of types: UINT8, UINT16, SINT16, REAL.

This operation is optimized for NEON technology for pixels of types: UINT8, SINT8, UINT16, SINT16, SINT32(for inScale!=1), REAL(for inScale!=1).

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Image formats are not the same in DivideImages.
DomainError	Image sizes are not equal in DivideImages.
DomainError	Region exceeds an input image in DivideImages.

See Also

- MultiplyImages – Multiplies two images pixel by pixel.

LerpImages

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Interpolates two images linearly pixel by pixel.

Syntax

```
void avl::LerpImages
(
 const avl::Image& inImage1,
 const avl::Image& inImage2,
 atl::Optional<const avl::Region&> inRoi,
 float inLambda,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage1	const Image&			First input image
→	inImage2	const Image&			Second input image
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inLambda	float	0.0 - 1.0	0.5f	Interpolation between the input images where 0.0 value is equal to inImage1 and 1.0 to inImage2
←	outImage	Image&			Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage1** and **outImage**, **inImage2** and **outImage**

Read more about In-place Computation.

Description

The operation computes the linear interpolation of two images. Each pixel of the output image is computed as follows.

$$outImage[i,j] = (1 - inLambda) \cdot inImage0[i,j] + inLambda \cdot inImage1[i,j]$$

In multichannel (color) images each pixel channel is processed separately.

The operation requires that the images being processed have equal format and dimensions, otherwise an error with appropriate description occurs. To obtain an image of desired dimensions one can use ResizeImage or CropImage filter. To alter the pixel type of an image one can use ConvertPixelType filter. ConvertToMultichannel and AverageChannels filters allow to alter the number of image channels.

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT16, REAL.

This operation is optimized for AVX2 technology for pixels of types: UINT8, SINT16, REAL.

This operation is optimized for NEON technology for pixels of types: UINT8, SINT8, UINT16, SINT16, SINT32.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Image formats are not the same in LerpImages.
DomainError	Image sizes are not equal in LerpImages.
DomainError	Region exceeds an input image in LerpImages.

See Also

- AverageImages – Averages two images pixel by pixel.
- BlendImages – Computes weighted sum pixel by pixel.

LerpImages_ByImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Interpolates two images linearly pixel by pixel according to another image values.

Syntax

```
void avl::LerpImages_ByImage
(
 const avl::Image& inImage1,
 const avl::Image& inImage2,
 atl::Optional<const avl::Region&> inRoi,
 const avl::Image& inLambdaImage,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage1	const Image&		First input image
→	inImage2	const Image&		Second input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inLambdaImage	const Image&		Values of interpolation between the input images where 0.0 value is equal to inImage1 and 1.0 to inImage2
←	outImage	Image&		Output image

Requirements

For input **inLambdaImage** only pixel formats are supported: 1□real.

Read more about pixel formats in Image documentation.

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage1** and **outImage**, **inImage2** and **outImage**

Read more about In-place Computation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Image formats are not the same in LerpImages_ByImage.
DomainError	Image sizes are not equal in LerpImages_ByImage.
DomainError	Region exceeds an input image in LerpImages_ByImage.
DomainError	Not supported inLambdaImage pixel format in LerpImages_ByImage. Supported formats: 1xReal.

MaximumImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates an image from the higher pixel values of each corresponding pair.

Syntax

```
void avl::MaximumImage
(
 const avl::Image& inImage1,
 const avl::Image& inImage2,
 atl::Optional<const avl::Region&> inRoi,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage1	const Image&		First input image
→	inImage2	const Image&		Second input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
←	outImage	Image&		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage1** and **outImage**, **inImage2** and **outImage**

Read more about In-place Computation.

Description

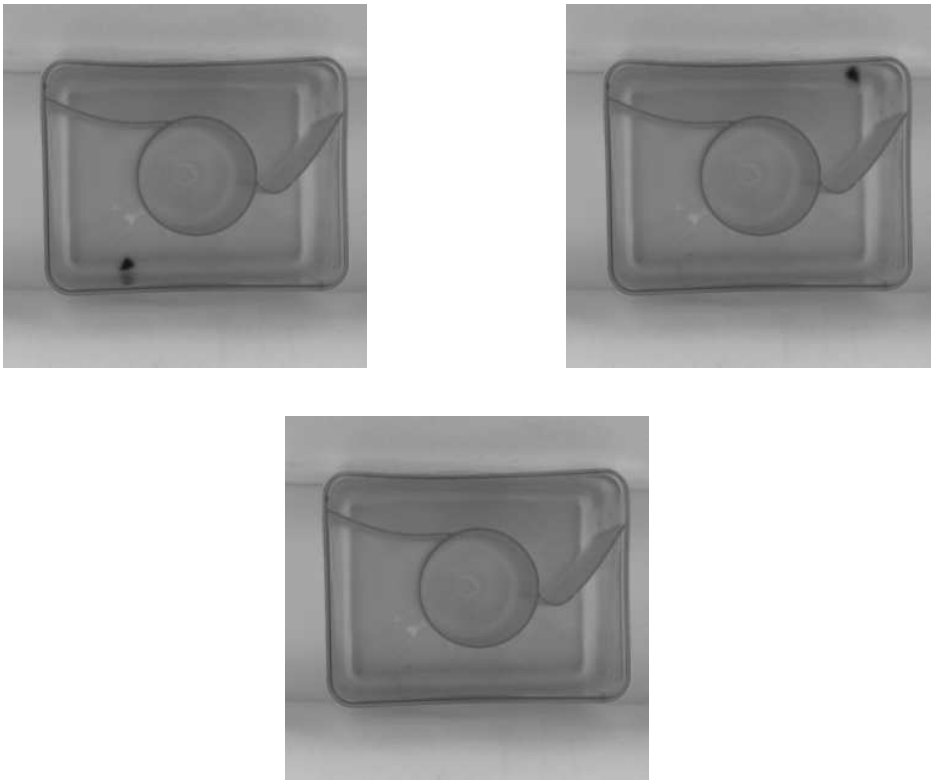
The operation computes the maximum of two images. Each **outImage** pixel is equal to the brighter of the corresponding pixels of the input images.

$$\forall_{i,j} \mathbf{OutImage}_{i,j} = \mathrm{Max}(\mathbf{InImage1}_{i,j}, \mathbf{InImage2}_{i,j})$$

In multichannel (color) images each pixel channel is processed separately.

The operation requires that the images being processed have equal format and dimensions, otherwise an error with appropriate description occurs. To obtain an image of desired dimensions one can use ResizeImage or CropImage filter. To alter the pixel type of an image one can use ConvertPixelType filter. ConvertToMultichannel and AverageChannels filters allow to alter the number of image channels.

Examples



MaximumImage *performed on the sample images.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT16, REAL.

This operation is optimized for SSE4.1 technology for pixels of types: SINT8, UINT16, SINT32.

This operation is optimized for AVX2 technology for pixels of types: UINT8, SINT8, UINT16, SINT16, SINT32, REAL.

This operation is optimized for NEON technology for pixels of types: UINT8, SINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Image formats are not the same in MaximumImage.
DomainError	Image sizes are not equal in MaximumImage.
DomainError	Region exceeds an input image in MaximumImage.

See Also

- MinimumImage – Creates an image from the lower pixel values of each corresponding pair.

MaximumImage_OfArray

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the maximum of images of an array pixel by pixel.

Syntax

```
void avl::MaximumImage_OfArray
(
 const atl::Array<avl::Image>& inImageArray,
 atl::Optional<const avl::Region&> inRoi,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImageArray	const Array<Image>&		
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
←	outImage	Image&		Output image

Description

Array version of MaximumImage.

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT16, REAL.

This operation is optimized for SSE4.1 technology for pixels of types: SINT8, UINT16, SINT32.

This operation is optimized for AVX2 technology for pixels of types: UINT8, SINT8, UINT16, SINT16, SINT32, REAL.

This operation is optimized for NEON technology for pixels of types: UINT8, SINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- MaximumImage – Creates an image from the higher pixel values of each corresponding pair.

MaximumImage_OfLoop

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the maximum of images appearing in consecutive iterations pixel by pixel.

Syntax

```
void avl::MaximumImage_OfLoop
(
 ImageCombinators_OfLoopState& ioState,
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
	ioState	ImageCombinators_OfLoopState&		Object used to maintain state of the function.
	inImage	const Image&		Input image
	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
	outImage	Image&		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

Read more about In-place Computation.

Description

Loop version of MaximumImage.

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT16, REAL.

This operation is optimized for SSE4.1 technology for pixels of types: SINT8, UINT16, SINT32.

This operation is optimized for AVX2 technology for pixels of types: UINT8, SINT8, UINT16, SINT16, SINT32, REAL.

This operation is optimized for NEON technology for pixels of types: UINT8, SINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- MaximumImage – Creates an image from the higher pixel values of each corresponding pair.

MedianImages_OfArray

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Compute median value for each pixel of images from an array.

Syntax

```
void avl::MedianImages_OfArray
(
 const atl::Array<avl::Image>& inImages,
 atl::Optional<const avl::Region&> inRoi,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImages	const Array<Image>&		
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
←	outImage	Image&		Output image

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array of images in MedianImages_OfArray.
DomainError	Image dimensions are not equal in MedianImages_OfArray.
DomainError	Image formats are not the same in MedianImages_OfArray.
DomainError	Not supported image type in MedianImages_OfArray.
DomainError	Region exceeds an input image in MedianImages_OfArray.

MinimumImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates an image from the lower pixel values of each corresponding pair.

Syntax

```
void avl::MinimumImage
(
 const avl::Image& inImage1,
 const avl::Image& inImage2,
 atl::Optional<const avl::Region&> inRoi,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage1	const Image&		First input image
→	inImage2	const Image&		Second input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
←	outImage	Image&		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage1** and **outImage**, **inImage2** and **outImage**

Read more about In-place Computation.

Description

The operation computes the minimum of two images. Each **outImage** pixel is equal to the darker of the corresponding pixels of the input images.

$$\forall_{i,j}\mathbf{OutImage}_{i,j} = \mathrm{Min}(\mathbf{InImage1}_{i,j}, \mathbf{InImage2}_{i,j})$$

In multichannel (color) images each pixel channel is processed separately.

The operation requires that the images being processed have equal format and dimensions, otherwise an error with appropriate description occurs. To obtain an image of desired dimensions one can use ResizeImage or CropImage filter. To alter the pixel type of an image one can use ConvertPixelType filter. ConvertToMultichannel and AverageChannels filters allow to alter the number of image channels.

Examples

MinimumImage *performed on the sample images.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT16, REAL.

This operation is optimized for SSE4.1 technology for pixels of types: SINT8, UINT16, SINT32.

This operation is optimized for AVX2 technology for pixels of types: UINT8, SINT8, UINT16, SINT16, SINT32, REAL.

This operation is optimized for NEON technology for pixels of types: UINT8, SINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Image formats are not the same in MinimumImage.
DomainError	Image sizes are not equal in MinimumImage.
DomainError	Region exceeds an input image in MinimumImage.

See Also

- MaximumImage – Creates an image from the higher pixel values of each corresponding pair.

MinimumImage_OfArray

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the minimum of images of an array pixel by pixel.

Syntax

```
void avl::MinimumImage_OfArray
(
 const atl::Array<avl::Image>& inImageArray,
 atl::Optional<const avl::Region&> inRoi,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImageArray	const Array<Image>&		
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
←	outImage	Image&		Output image

Description

Array version of MinimumImage.

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, UINT16, REAL.

This operation is optimized for SSE4.1 technology for pixels of types: SINT8, UINT16, SINT32.

This operation is optimized for AVX2 technology for pixels of types: UINT8, SINT8, UINT16, SINT16, SINT32, REAL.

This operation is optimized for NEON technology for pixels of types: UINT8, SINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- MinimumImage – Creates an image from the lower pixel values of each corresponding pair.

MinimumImage_OfLoop

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the minimum of images appearing in consecutive iterations pixel by pixel.

Syntax

```
void avl::MinimumImage_OfLoop
(
 ImageCombinators_OfLoopState& ioState,
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
	ioState	ImageCombinators_OfLoopState&		Object used to maintain state of the function.
	inImage	const Image&		Input image
	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
	outImage	Image&		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

Read more about In-place Computation.

Description

Loop version of MinimumImage.

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT16, REAL.

This operation is optimized for SSE4.1 technology for pixels of types: SINT8, UINT16, SINT32.

This operation is optimized for AVX2 technology for pixels of types: UINT8, SINT8, UINT16, SINT16, SINT32, REAL.

This operation is optimized for NEON technology for pixels of types: UINT8, SINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- MinimumImage – Creates an image from the lower pixel values of each corresponding pair.

MultiplyImages

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Multiplies two images pixel by pixel.

Syntax

```
void avl::MultiplyImages
(
 const avl::Image& inImage1,
 const avl::Image& inImage2,
 atl::Optional<const avl::Region&> inRoi,
 float inScale,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage1	const Image&		First input image
→	inImage2	const Image&		Second input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inScale	float	1.0f	Output image scaling factor
←	outImage	Image&		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage1** and **outImage**, **inImage2** and **outImage**

Read more about In-place Computation.

Description

The operation computes the product of two images. Each **outImage** pixel is equal to the product of the corresponding pixels of the input images.

$$\forall_{i,j}\mathbf{OutImage}_{i,j} = (\mathbf{InImage1}_{i,j} \cdot \mathbf{InImage2}_{i,j}) \cdot \mathbf{InScale}$$

The result of the operation is multiplied by **inScale** factor, which may be used to avoid clipping when the produced values exceed the range of correct pixel values. However it should be noted that when the value of **inScale** is other than 1.0, the filter uses a slower, not SSE-optimized implementation.

Whenever the resulting value exceeds the range of pixel values, it is clipped to the nearest proper value. In multichannel (color) images each pixel channel is processed separately.

The operation requires that the images being processed have equal format and dimensions, otherwise an error with appropriate description occurs. To obtain an image of desired dimensions one can use ResizeImage or CropImage filter. To alter the pixel type of an image one can use ConvertPixelType filter. ConvertToMultichannel and AverageChannels filters allow to alter the number of image channels.

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT16, REAL, SINT8(for inScale=1).

This operation is optimized for SSE41 technology for pixels of type: UINT16.

This operation is optimized for AVX2 technology for pixels of types: UINT8, UINT16, SINT16, REAL, SINT8(for inScale=1).

This operation is optimized for NEON technology for pixels of types: UINT8(for inScale!=1), SINT8(for inScale!=1), UINT16(for inScale!=1), SINT16(for inScale!=1), SINT32(for inScale!=1).

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Image formats are not the same in MultiplyImages.
DomainError	Image sizes are not equal in MultiplyImages.
DomainError	Region exceeds an input image in MultiplyImages.

See Also

- DivideImages – Divides two images pixel by pixel.

MultiplyImages_OfArray

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Multiplies images of an array pixel by pixel.

Syntax

```
void avl::MultiplyImages_OfArray
(
 const atl::Array<avl::Image>& inImageArray,
 atl::Optional<const avl::Region&> inRoi,
 float inScale,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImageArray	const Array<Image>&		
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inScale	float	1.0f	
←	outImage	Image&		Output image

Description

Array version of MultiplyImages.

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT16, REAL, SINT8(for inScale=1).

This operation is optimized for SSE41 technology for pixels of type: UINT16.

This operation is optimized for AVX2 technology for pixels of types: UINT8, UINT16, SINT16, REAL, SINT8(for inScale=1).

This operation is optimized for NEON technology for pixels of types: UINT8(for inScale!=1), SINT8(for inScale!=1), UINT16(for inScale!=1), SINT16(for inScale!=1), SINT32(for inScale!=1).

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- MultiplyImages – Multiplies two images pixel by pixel.

MultiplyImages_OfLoop

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Multiplies images appearing in consecutive iterations pixel by pixel.

Syntax

```
void avl::MultiplyImages_OfLoop
(
 ImageCombinators_OfLoopState& ioState,
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 float inScale,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
	ioState	ImageCombinators_OfLoopState&		Object used to maintain state of the function.
	inImage	const Image&		Input image
	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
	inScale	float	1.0f	
	outImage	Image&		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

Read more about In-place Computation.

Description

Loop version of MultiplyImages.

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT16, REAL, SINT8(for inScale=1).

This operation is optimized for SSE41 technology for pixels of type: UINT16.

This operation is optimized for AVX2 technology for pixels of types: UINT8, UINT16, SINT16, REAL, SINT8(for inScale=1).

This operation is optimized for NEON technology for pixels of types: UINT8(for inScale!=1), SINT8(for inScale!=1), UINT16(for inScale!=1), SINT16(for inScale!=1), SINT32(for inScale!=1).

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- MultiplyImages – Multiplies two images pixel by pixel.

NthImage_OfArray

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

For each pixel location, get value of n-th (increasing) pixel among input images.

Syntax

```
void avl::NthImage_OfArray
(
 const atl::Array<avl::Image>& inImages,
 const atl::Optional<const atl::Array<avl::Region>&>& inSourceRois,
 int inN,
 bool inReverse,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImages	const Array<Image>&		
→	inSourceRois	const Optional<const Array<Region>&>&	NIL	
→	inN	int		
→	inReverse	bool	False	Reverse the ordering of pixel values to decreasing.
←	outImage	Image&		Output image

Errors

List of possible exceptions:

Error type	Description
DomainError	ROI exceeds an input image in NthImage_OfArray.
DomainError	Sizes of image array and ROI array differ in NthImage_OfArray.
DomainError	The inN parameter cant be negative.
DomainError	The inN parameter must be smaller than input image array size.

RollingAverageImages_OfArray

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Averages images appearing in an array pixel by pixel using exponential rolling average.

Syntax

```
void avl::RollingAverageImages_OfArray
(
 const atl::Array<avl::Image>& inImageArray,
 atl::Optional<const avl::Region&> inRoi,
 const float inAlpha,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImageArray	const Array<Image>&			
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inAlpha	const float	0.0 - 1.0	0.5f	
←	outImage	Image&			Output image

RollingAverageImages_OfLoop

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Averages images appearing in consecutive iterations pixel by pixel using exponential rolling average.

Syntax

```
void avl::RollingAverageImages_OfLoop
(
 ImageCombinators_OfLoopState& ioState,
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 const float inAlpha,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	ImageCombinators_OfLoopState&			Object used to maintain state of the function.
	inImage	const Image&			Input image
	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
	inAlpha	const float	0.0 - 1.0	0.5f	
	outImage	Image&			Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

Read more about In-place Computation.

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT16, REAL, SINT8(for inScale=1), UINT16(for inScale=1).

This operation is optimized for SSE41 technology for pixels of types: UINT16(for inScale!=1).

This operation is optimized for AVX2 technology for pixels of types: UINT8, SINT16, UINT16, REAL, SINT8(for inScale=1).

This operation supports automatic parallelization for multicore and multiprocessor systems.

SubtractImages

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Subtracts two images pixel by pixel. The result is signed.

Syntax

```
void avl::SubtractImages
(
 const avl::Image& inImage1,
 const avl::Image& inImage2,
 atl::Optional<const avl::Region&> inRoi,
 float inScale,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
	inImage1	const Image&		First input image
	inImage2	const Image&		Second input image
	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
	inScale	float	1.0f	Output image scaling factor
	outImage	Image&		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage1** and **outImage**, **inImage2** and **outImage**

Read more about In-place Computation.

Description

Operation subtracts two images. Each **outImage** pixel is equal to the difference of corresponding pixels of the input images.

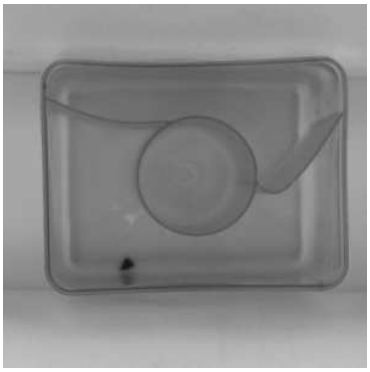
$$\forall_{i,j} \mathbf{OutImage}_{i,j} = \left(\mathbf{InImage1}_{i,j} - \mathbf{InImage2}_{i,j}\right) \cdot \mathbf{InScale}$$

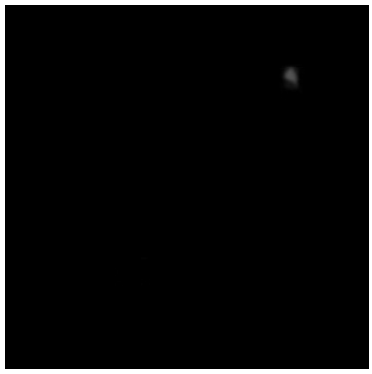
The result of the operation is multiplied by **inScale** factor, which may be used to avoid clipping when the produced values exceed the range of correct pixel values. However it should be noted that when the value of **inScale** is other than 1.0, the filter uses a slower, not SSE-optimized implementation.

Whenever the resulting value exceeds the range of pixel values, it is clipped to the nearest proper value. In multichannel (color) images each pixel channel is processed separately.

The operation requires that the images being processed have equal format and dimensions, otherwise an error with appropriate description occurs. To obtain an image of desired dimensions one can use ResizeImage or CropImage filter. To alter the pixel type of an image one can use ConvertPixelType filter. ConvertToMultichannel and AverageChannels filters allow to alter the number of image channels.

Examples





SubtractImages *performed on the sample images.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT16, REAL, SINT8(for inScale=1), UINT16(for inScale=1).

This operation is optimized for SSE41 technology for pixels of types: UINT16(for inScale!=1).

This operation is optimized for AVX2 technology for pixels of types: UINT8, SINT16, UINT16, REAL, SINT8(for inScale=1).

This operation is optimized for NEON technology for pixels of types: UINT8, SINT8, UINT16, SINT16, SINT32.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Image formats are not the same in SubtractImages.
DomainError	Image sizes are not equal in SubtractImages.
DomainError	Region exceeds an input image in SubtractImages.

See Also

- AddImages – Adds two images pixel by pixel.
- DifferenceImage – Computes the non-negative distances between corresponding pixel values.

17. Shape Adjustment

Table of content:

AdjustPathArraysToEdges

Header: AVL.h
Namespace: avl
Module: FoundationPro

Translates, rotates and scales multiple contour sets (PathArray), each separately, to the edges of the input image with subpixel precision.

Applications: Fine-tune results of edge-based template matching.

Syntax

```
void avl::AdjustPathArraysToEdges
(
 const avl::Image& inImage,
 const atl::Array<atl::Array<avl::Path>>& inPaths,
 atl::Optional<float> inPointSpacing,
 atl::Optional<const atl::Array<avl::CoordinateSystem2D>&> inAlignments,
 float inAttractionRadius,
 avl::AdjustmentMetric::Type inAdjustmentMetric,
 bool inAdjustTranslation,
 bool inAdjustRotation,
 bool inAdjustScale,
 int inIterationCount,
 float inBaseGradient,
 atl::Array<atl::Array<avl::Path>>& outAdjustedPaths,
 atl::Array<avl::CoordinateSystem2D>& outAlignments,
 atl::Array<atl::Array<avl::Segment2D>>& diagAttractionVectors
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image.
→	inPaths	const Array<Array<Path>>&			Paths to be adjusted. For sparse, synthetic paths the inPointSpacing needs to be specified.
→	inPointSpacing	Optional<float>	1.0 - ∞	NIL	If set, sampling points of the input path will be equidistant with specified spacing. Useful for increasing density of synthetic, sparse paths.
→	inAlignments	Optional<const Array<CoordinateSystem2D>&>		NIL	Alignments to be corrected, usually connect to outObjects.Alignment[] of LocateObjects:Multiple filter.
→	inAttractionRadius	float	0.1 - ∞	2.0f	Expected initial distance between inPaths and edges of the input image.
→	inAdjustmentMetric	AdjustmentMetric::Type		SegmentDistance	Metric used for path attraction. The SegmentDistance minimizes distances along path normal vectors, and thus adjustment is more accurate. PointDistance_* minimize euclidean distance, adjustment tends to be more stable.
→	inAdjustTranslation	bool		True	Compute the translation part of adjustment transform. Must be true when using AttractPathTangents estimation method.
→	inAdjustRotation	bool		True	Compute the rotation part of adjustment transform.
→	inAdjustScale	bool		False	Compute the scale part of adjustment transform.
→	inIterationCount	int	0 - 100	6	Number of iterations of internal adjustment algorithm.
→	inBaseGradient	float	0.1 - ∞	1.0f	Threshold for suppression of weak input image gradients. Increase for very noisy images.
←	outAdjustedPaths	Array<Array<Path>>&			Adjusted output paths.
←	outAlignments	Array<CoordinateSystem2D>&			Corrected alignments - inAlignments input modified with estimated adjustment parameters.
🔍	diagAttractionVectors	Array<Array<Segment2D>>&			Attraction vectors from first iteration.

Hints

- This filter is a multiple object version of AdjustPathArrayToEdges, however it achieves much higher performance than AdjustPathArrayToEdges executed in a loop.
- Please refer to AdjustPathArrayToEdges documentation for more information.

Hardware Acceleration

This operation is optimized for SSSE3 technology for pixels of type: UINT8.

This operation is optimized for AVX2 technology for pixels of type: UINT8.

This operation is optimized for NEON technology for pixels of type: UINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	inPaths and inAlignments have different size
DomainError	To use SegmentDistance adjustment metric, inAdjustTranslation must be enabled.

See Also

- AdjustPathArrayToEdges – Translates, rotates and scales the given contour set to the edges of the input image with subpixel precision.
- EnhanceSingleObjectMatch – Improves accuracy of single object matching by adding a subpixel-precise adjustment.
- EnhanceMultipleObjectMatches – Improves accuracy of multiple object matching by adding a subpixel-precise adjustment.

AdjustPathArrayToEdges

Header: AVL.h
Namespace: avl
Module: FoundationPro

Translates, rotates and scales the given contour set to the edges of the input image with subpixel precision.

Applications: Fine-tune results of edge-based template matching.

Syntax

```
void avl::AdjustPathArrayToEdges
(
 const avl::Image& inImage,
 const atl::Array< avl::Path >& inPaths,
 atl::Optional<float> inPointSpacing,
 atl::Optional<const avl::CoordinateSystem2D&> inAlignment,
 float inAttractionRadius,
 avl::AdjustmentMetric::Type inAdjustmentMetric,
 bool inAdjustTranslation,
 bool inAdjustRotation,
 bool inAdjustScale,
 int inIterationCount,
 float inBaseGradient,
 atl::Array< avl::Path >& outAdjustedPaths,
 avl::CoordinateSystem2D& outAlignment,
 atl::Array<avl::Segment2D>& diagAttractionVectors
)
```

Parameters

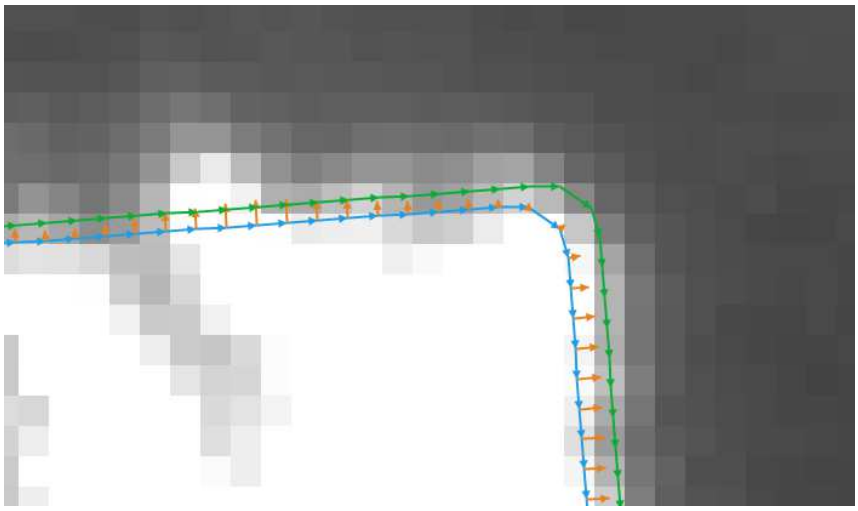
	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image.
→	inPaths	const Array< Path >&			Paths to be adjusted. For sparse, synthetic paths the inPointSpacing needs to be specified.
→	inPointSpacing	Optional<float>	1.0 - ∞	NIL	If set, sampling points of the input path will be equidistant with specified spacing. Useful for increasing density of synthetic, sparse paths.
→	inAlignment	Optional<const CoordinateSystem2D&>		NIL	Alignment to be corrected, usually connects to outObject.Alignment of LocateObjects filter.
→	inAttractionRadius	float	0.1 - ∞	2.0f	Expected initial distance between inPaths and edges of the input image.
→	inAdjustmentMetric	AdjustmentMetric::Type		SegmentDistance	Metric used for path attraction. The SegmentDistance minimizes distances along path normal vectors, and thus adjustment is more accurate. PointDistance_* minimize euclidean distance, adjustment tends to be more stable.
→	inAdjustTranslation	bool		True	Compute the translation part of adjustment transform. Must be true when using AttractPathTangents estimation method.
→	inAdjustRotation	bool		True	Compute the rotation part of adjustment transform.
→	inAdjustScale	bool		False	Compute the scale part of adjustment transform.
→	inIterationCount	int	0 - 100	6	Number of iterations of internal adjustment algorithm.
→	inBaseGradient	float	0.1 - ∞	1.0f	Threshold for suppression of weak input image gradients. Increase for very noisy images.
←	outAdjustedPaths	Array< Path >&			Adjusted output paths.
←	outAlignment	CoordinateSystem2D&			Corrected alignment - the inAlignment input modified with estimated adjustment parameters.
🔍	diagAttractionVectors	Array<Segment2D>&			Attraction vectors from first iteration.

Hints

- In order to use the filter to fine-tune the results of template matching, connect the **outObjectEdges** output of the template matching filter to the **inPaths** input, **outObject.Alignment** to **inAlignment** and the same input image to the **inImage** input.
- When using template matching with AdjustPathArrayToEdges filter, you may try to speed up template matching by increasing its **inMinPyramidLevel** parameter. The potential loss of template matching precision should be mitigated by the AdjustPathArrayToEdges filter.
- The **inAttractionRadius** should be set to the average initial distance between inPaths and edges of the input image. Although the filter tolerance to this parameter is high, setting it too low may result in adjustment failure, setting it too high will result in reduced accuracy. The parameter directly influences attraction vectors, which may be observed via diagnostic output **diagAttractionVectors**.
- **SegmentDistance** is the default value of **inAdjustmentMetric** parameter and usually it is the best choice. However, in some special cases one should consider other options:
  - **inImage** contains glares – the **PointDistance_Median** method may perform better as the median error metric is more robust than least squares.
  - **inPaths** is a degenerate path array, such as a 1D straight line – the **SegmentDistance** could fail, the **PointDistance_*** methods may work.

  Please note that you need to increase **inIterationCount** to **10** when using **PointDistance_*** adjustment metrices, as the default value of **6** is fine-tuned to the **SegmentDistance** metric.
- Increasing **inIterationCount** may result in more accurate results, however gains are diminishing quickly. The only downside of increasing **inIterationCount** in higher computation time.
- Sparse paths - with a low amount of points - need to be densened to obtain acceptable accuracy and stability of the filter. Set the **inPointSpacing** parameter to a desired distance between consecutive path points.
- For paths with large amount of points, the filter execution may be considerably accelerated without perceivable influence on the accuracy. Set the **inPointSpacing** parameter to a value higher than average spacing of input paths.
- Noise on the input image may introduce false adjustment vectors, reducing accuracy. To mitigate the influence of noise, the **inBaseGradient** parameter may be used, which governs suppression of weak input image gradients. In order to tune the parameter, observe the **diagAttractionVectors** output. Note that its better to have some noise in attraction vectors directions than to considerably reduce their strength.
- When dealing with multiple objects on a single image, use the AdjustPathArraysToEdges for improved performance.

Examples



Improving results of template matching. Blue – input paths, orange – attraction vectors, green – corrected paths.

Hardware Acceleration

This operation is optimized for SSSE3 technology for pixels of type: UINT8.

This operation is optimized for AVX2 technology for pixels of type: UINT8.

This operation is optimized for NEON technology for pixels of type: UINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- EnhanceSingleObjectMatch – Improves accuracy of single object matching by adding a subpixel-precise adjustment.
- AdjustPathArraysToEdges – Translates, rotates and scales multiple contour sets (PathArray), each separately, to the edges of the input image with subpixel precision.
- EnhanceMultipleObjectMatches – Improves accuracy of multiple object matching by adding a subpixel-precise adjustment.

18. Geometry 2D Fitting

Table of content:

AdjustPointArrays

Header: AVL.h
Namespace: avl
Module: FoundationPro

Aligns a point array to match best the input point array.

Syntax

```
void avl::AdjustPointArrays
(
 const atl::Array<avl::Point2D>& inPoints,
 const atl::Array<avl::Point2D>& inReferencePoints,
 const bool inAllowRotation,
 const bool inAllowScale,
 const int inMaxIterationCount,
 const float inMatchFraction,
 const int inInitialTransformCount,
 atl::Conditional<atl::Array<avl::Point2D> >& outAlignedPoints,
 atl::Conditional<avl::CoordinateSystem2D>& outAlignment
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoints	const Array<Point2D>&			Points to be aligned
→	inReferencePoints	const Array<Point2D>&			Points to align to
→	inAllowRotation	const bool		True	Flag indicating whether rotation is allowed as a part of output alignment
→	inAllowScale	const bool		False	Flag indicating whether scale is allowed as a part of output alignment
→	inMaxIterationCount	const int	1 - ∞	10	Maximal number of iteration for the algorithm
→	inMatchFraction	const float	0.0 - 1.0	0.75f	Defines fraction of input points that is being fitted in every iteration
→	inInitialTransformCount	const int	1 - ∞	10	Number of initial transforms to be tried out by the algorithm
←	outAlignedPoints	Conditional<Array<Point2D> >&			The aligned input points
←	outAlignment	Conditional<CoordinateSystem2D>&			The transform that aligns best the input points

Errors

List of possible exceptions:

Error type	Description
DomainError	Input point array is empty in AdjustPointArrays.

DetectPointSegments

Header: AVL.h
Namespace: avl
Module: FoundationPro

Detect points that lie along multiple segments.

Syntax

```
void avl::DetectPointSegments
(
 const atl::Array<avl::Point2D>& inPoints,
 const float inMaxDistance,
 const int inMaxRank,
 atl::Optional<float> inMaxRelativeDistance,
 const float inMaxTurnAngle,
 const int inMinPointCount,
 atl::Array<avl::Segment2D>& outPointSegments,
 atl::Array<avl::Path>& outPaths
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoints	const Array<Point2D>&			Points to connect
→	inMaxDistance	const float	0.0 - ∞	10.0f	Maximum distance between connected points
→	inMaxRank	const int	1 - ∞	4	Maximum number of neighbour candidates considered when joining points
→	inMaxRelativeDistance	Optional<float>	1.0 - ∞	2.0f	Maximum distance in relation to the shortest distance for a point
→	inMaxTurnAngle	const float	0.0 - 90.0	5.0f	Maximum angle between consecutive path segments
→	inMinPointCount	const int	2 - ∞	3	Minimum number of points in one path
←	outPointSegments	Array<Segment2D>&			Segments connecting first and last point of each path
←	outPaths	Array<Path>&			Paths of connected points

Description

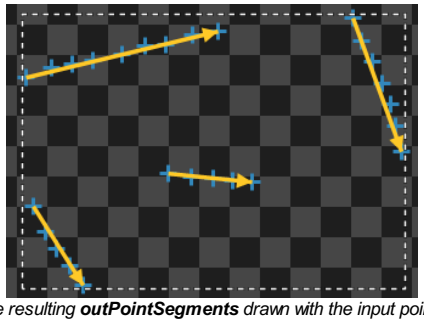
Use this filter to detect multiple segments or paths at the same time.

Set **inMaxDistance** to smaller value to avoid connecting distant points.
Set **inMaxRank** to larger value to allow detecting paths that lie close to each other.
Set **inMaxTurnAngle** to larger value when points do not lie precisely on a segment.
Set **inMinPointCount** to filter out too short paths (useful when some of the input points are noise).

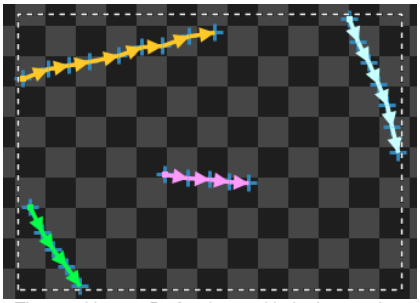
Filter returns array of paths as well as segments connecting first and last point of each path.

Examples

*Results for input parameters: **inMaxDistance** = 30, **inMaxRank** = 4, **inMaxTurnAngle** = 20, **inMinPointCount** = 4.*



*The resulting **outPointSegments** drawn with the input points.*



*The resulting **outPaths** drawn with the input points.*

See Also

- FitSegmentToPoints – Approximates points with a segment using selected outliers suppression method.

DetectPointSegments_Ex

Header: AVL.h
Namespace: avl
Module: FoundationPro

Detect points that lie along multiple segments.

Syntax

```
void avl::DetectPointSegments_Ex
(
 const atl::Array<avl::Point2D>& inPoints,
 const float inMaxDistance,
 const int inMaxRank,
 atl::Optional<float> inMaxRelativeDistance,
 const float inMaxTurnAngle,
 const int inMinPointCount,
 atl::Array<avl::Segment2D>& outPointSegments,
 atl::Array<avl::Path>& outPaths
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoints	const Array<Point2D>&			Points to connect
→	inMaxDistance	const float	0.0 - ∞	10.0f	Maximum distance between connected points
→	inMaxRank	const int	1 - ∞	4	Maximum number of neighbour candidates considered when joining points
→	inMaxRelativeDistance	Optional<float>	1.0 - ∞	2.0f	Maximum distance in relation to the shortest distance for a point
→	inMaxTurnAngle	const float	0.0 - 90.0	5.0f	Maximum angle between consecutive path segments
→	inMinPointCount	const int	2 - ∞	3	Minimum number of points in one path
←	outPointSegments	Array<Segment2D>&			Segments connecting first and last point of each path
←	outPaths	Array<Path>&			Paths of connected points

FitArcToPath

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Approximates path by an arc using the selected outliers suppression method and considering path's start and end.

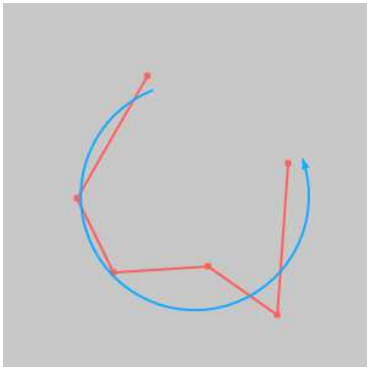
Syntax

```
void avl::FitArcToPath
(
 const avl::Path& inPath,
 avl::CircleFittingMethod::Type inFittingMethod,
 atl::Optional<avl::MEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Arc2D>& outArc
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
→	inFittingMethod	CircleFittingMethod::Type		
→	inOutlierSuppression	Optional<MEstimator::Type>	NIL	
←	outArc	Conditional<Arc2D>&		Fitted arc or nothing if the computations failed to converge

Examples



*The resulting **outArc** drawn with the input path, **inFittingMethod** = AlgebraicKasa and **inOutlierSuppression** = Auto.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Outlier suppression is supported only in algebraic fitting methods.

FitArcToPoints

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Approximates points with an arc using the selected outliers suppression method.

Syntax

```
void avl::FitArcToPoints
(
 const atl::Array<avl::Point2D>& inPoints,
 atl::Optional<const avl::Range&> inRange,
 avl::CircleFittingMethod::Type inFittingMethod,
 atl::Optional<avl::MEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Arc2D>& outArc,
 atl::Optional<atl::Array<avl::Point2D>&> outInliers = atl::NIL
)
```

Parameters

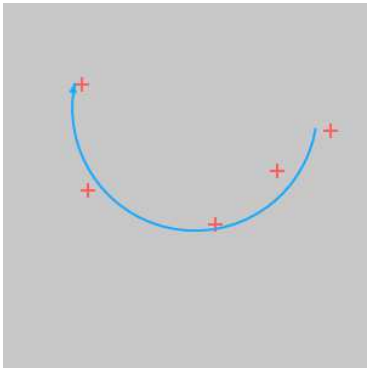
	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		
→	inRange	Optional<const Range&>	NIL	Determines which array points take part in fitting process
→	inFittingMethod	CircleFittingMethod::Type		
→	inOutlierSuppression	Optional<MEstimator::Type>	NIL	
←	outArc	Conditional<Arc2D>&		Fitted arc or nothing if the computations failed to converge
←	outInliers	Optional<Array<Point2D>&>	NIL	Points matching the fitting arc

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outInliers**.

Read more about Optional Outputs.

Examples



*The resulting **outArc** drawn with the input points, **inFittingMethod** = AlgebraicKasa and **inOutlierSuppression** = Auto.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Outlier suppression is supported only in algebraic fitting methods.
DomainError	Range exceeds the input point array in FitArcToPoints.

FitCircleToPoints

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Approximates points with a circle using selected outliers suppression method.

Syntax

```
void avl::FitCircleToPoints
(
 const atl::Array<avl::Point2D>& inPoints,
 atl::Optional<const avl::Range&> inRange,
 avl::CircleFittingMethod::Type inFittingMethod,
 atl::Optional<avl::MEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Circle2D>& outCircle,
 atl::Optional<atl::Array<avl::Point2D>&> outInliers = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		
→	inRange	Optional<const Range&>	NIL	Determines which array points take part in fitting process
→	inFittingMethod	CircleFittingMethod::Type		
→	inOutlierSuppression	Optional<MEstimator::Type>	NIL	
←	outCircle	Conditional<Circle2D>&		Fitted circle or nothing if method failed to converge
←	outInliers	Optional<Array<Point2D>&>	NIL	Points matching the computed circle

Optional Outputs

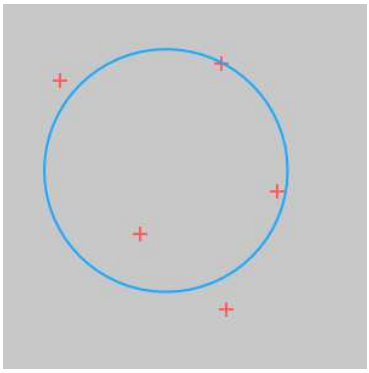
The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outInliers**.

Read more about Optional Outputs.

Description

The operation computes a circle which approximates the input points best. Several methods are available, **AlgebraicKasa** being the fastest one. It is also the most inaccurate when the input points are sampled along small arc only.

Examples



*The resulting **outCircle** drawn with the input points, **inFittingMethod** = AlgebraicKasa and **inOutlierSuppression** = Auto.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Outlier suppression is supported only in algebraic fitting methods.
DomainError	Range exceeds the input point array in FitCircleToPoints.

See Also

- FitArcToPoints – Approximates points with an arc using the selected outliers suppression method.
- FitLineToPoints – Approximates points with a line using the Least Squares method.

FitLineToPoints

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Approximates points with a line using the Least Squares method.

Syntax

```
void avl::FitLineToPoints
(
 const atl::Array<avl::Point2D>& inPoints,
 atl::Optional<const avl::Range&> inRange,
 avl::Line2D& outLine,
 atl::Optional<float&> outError = atl::NIL
)
```

Parameters

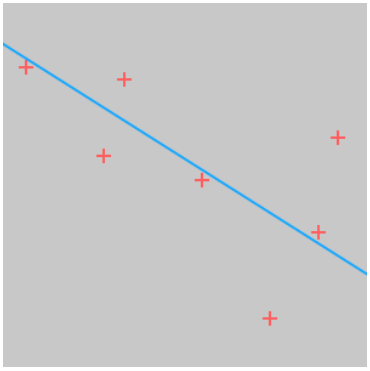
	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		
→	inRange	Optional<const Range&>	NIL	Determines which array points take part in fitting process
←	outLine	Line2D&		
←	outError	Optional<float&>	NIL	Sum of the point distances from the output line

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outError**.

Read more about Optional Outputs.

Examples



*The resulting **outLine** drawn with the input points.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array of points in FitLineToPoints.
DomainError	Range exceeds the input point array in FitLineToPoints.

FitLineToPoints_LTE

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Approximates points with a line using Least Trimmed Error algorithm.

Applications: Brute-force finding of a line that best matches a subset of the input points. Very efficient against outliers, but possibly slow for bigger subsets.

Syntax

```
void avl::FitLineToPoints_LTE
(
 const atl::Array<avl::Point2D>& inPoints,
 atl::Optional<const avl::Range&> inRange,
 int inSeedSubsetSize,
 atl::Optional<int> inEvalSubsetSize,
 avl::Line2D& outLine,
 atl::Optional<atl::Array<avl::Point2D>&> outLTInliers = atl::NIL,
 atl::Optional<float&> outLTError = atl::NIL,
 int& diagIterationCount = atl::Dummy<int>()
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoints	const Array<Point2D>&			Input points
→	inRange	Optional<const Range&>		NIL	Determines which array points take part in fitting process
→	inSeedSubsetSize	int	2 - 10	3	Number of points in one combination for getting a sample line
→	inEvalSubsetSize	Optional<int>	3 - ∞	NIL	Number of closest points used for evaluation of a sample line, or Auto if seed points are to be used
←	outLine	Line2D&			Fitted line
←	outLTInliers	Optional<Array<Point2D>&>		NIL	Inlying points of the best LTE line
←	outLTError	Optional<float&>		NIL	The Least Trimmed Error
🔍	diagIterationCount	int&			Number of combinations considered

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outLTInliers**, **outLTError**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array of points in FitLineToPoints_LTE.
DomainError	Range exceeds the input point array in FitLineToPoints_LTE.

FitLineToPoints_M

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Approximates points with a line using selected M-estimator for outlier suppression.

Applications: Finding a locally optimal line. Good enough when the number of outliers is small.

Syntax

```
void avl::FitLineToPoints_M
(
 const atl::Array<avl::Point2D>& inPoints,
 atl::Optional<const avl::Range&> inRange,
 avl::MEstimator::Type inOutlierSuppression,
 float inClippingFactor,
 int inIterationCount,
 atl::Optional<const avl::Line2D&> inInitialLine,
 avl::Line2D& outLine,
 atl::Optional<atl::Array<avl::Point2D>&> outInliers = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoints	const Array<Point2D>&			
→	inRange	Optional<const Range&>		NIL	Determines which array points take part in fitting process
→	inOutlierSuppression	MEstimator::Type			
→	inClippingFactor	float	0.675 - 6.0	2.5f	Multitude of standard deviation within which points are considered inliers
→	inIterationCount	int	0 - ∞	5	Number of iterations of outlier suppressing algorithm
→	inInitialLine	Optional<const Line2D&>		NIL	Initial approximation (if available)
←	outLine	Line2D&			
←	outInliers	Optional<Array<Point2D>&>		NIL	Points matching the computed line

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outInliers**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array of points in FitLineToPoints_M
DomainError	Range exceeds the input point array in FitLineToPoints_M

FitLineToPoints_RANSAC

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Approximates points with a line using a RANSAC algorithm.

Applications: Finds a well matching line, but for handling outliers requires a distance threshold that may be difficult to set.

Syntax

```
void avl::FitLineToPoints_RANSAC
(
 const atl::Array<avl::Point2D>& inPoints,
 atl::Optional<const avl::Range&> inRange,
 atl::Optional<int> inMaxOutlierCount,
 float inMaxInlierDistance,
 atl::Optional<int> inIterationCount,
 atl::Conditional<avl::Line2D>& outLine
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoints	const Array<Point2D>&			
→	inRange	Optional<const Range&>		NIL	Determines which array points take part in fitting process
→	inMaxOutlierCount	Optional<int>	0 - ∞	0	Determines how many outlier points can be present to end the search
→	inMaxInlierDistance	float	0.0 - ∞	3.0f	Distance from the output line for a point to be considered an inlier
→	inIterationCount	Optional<int>	1 - ∞	42	Number of iterations; Auto means that all point pairs will be used
←	outLine	Conditional<Line2D>&			

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array of points in FitLineToPoints_RANSAC.
DomainError	Range exceeds the input point array in FitLineToPoints_RANSAC.

FitLineToPoints_TheilSen

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Approximates points with a line using TheilSen algorithm, optionally with Siegel's improvement.

Applications: Finds a well matching line, ignoring up to 29.3% (TheilSen) or 50.0% (Siegel) outliers. Outliers do have some influence on accuracy.

Syntax

```
void avl::FitLineToPoints_TheilSen
(
 const atl::Array<avl::Point2D>& inPoints,
 atl::Optional<const avl::Range&> inRange,
 avl::TheilSenVariant::Type inVariant,
 atl::Optional<int> inSampleLimit,
 atl::Optional<float> inOutlierRatio,
 avl::Line2D& outLine,
 atl::Array<float>& diagOrientations
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoints	const Array<Point2D>&			Input points
→	inRange	Optional<const Range&>		NIL	Determines which array points take part in fitting process
→	inVariant	TheilSenVariant::Type			Switches between Theil-Sen and Siegel methods
→	inSampleLimit	Optional<int>	5 - ∞	NIL	How many pairs of points are used to estimate orientation
→	inOutlierRatio	Optional<float>	0.0 - 0.99	NIL	
←	outLine	Line2D&			Fitted line
🔍	diagOrientations	Array<float>&			Sample orientations used to determine the output line orientation

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array of points in FitLineToPoints_TheilSen.
DomainError	Range exceeds the input point array in FitLineToPoints_TheilSen.

FitSegmentToPoints

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Approximates points with a segment using selected outliers suppression method.

Applications: Finding a locally optimal segment. Good enough when the number of outliers is small.

Syntax

```
void avl::FitSegmentToPoints
(
 const atl::Array<avl::Point2D>& inPoints,
 atl::Optional<const avl::Range&> inRange,
 atl::Optional<avl::MEstimator::Type> inOutlierSuppression,
 avl::Segment2D& outSegment,
 atl::Optional<atl::Array<avl::Point2D>&> outInliers = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		
→	inRange	Optional<const Range&>	NIL	Determines which array points take part in fitting process
→	inOutlierSuppression	Optional<MEstimator::Type>	NIL	
←	outSegment	Segment2D&		
←	outInliers	Optional<Array<Point2D>&>	NIL	

Optional Outputs

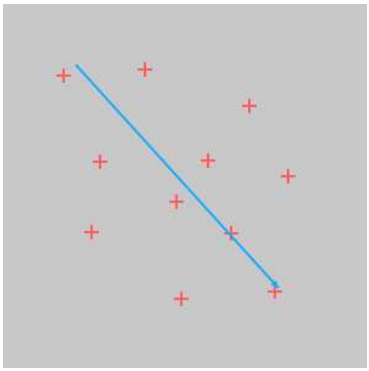
The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outInliers**.

Read more about Optional Outputs.

Description

The orientation of the resulting **outSegment** is always between 0 and 180 degrees.

Examples



*The resulting **outSegment** drawn with the input points, **inOutlierSuppression** = Auto.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array of points in FitSegmentToPoints.
DomainError	Range exceeds the input point array in FitSegmentToPoints.

FitSegmentToPoints_LTE

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Approximates points with a segment using Least Trimmed Error algorithm.

Applications: Brute-force finding of a segment that best matches a subset of the input points. Very efficient against outliers, but possibly slow for bigger subsets.

Syntax

```
void avl::FitSegmentToPoints_LTE
(
 const atl::Array<avl::Point2D>& inPoints,
 atl::Optional<const avl::Range&> inRange,
 int inSeedSubsetSize,
 atl::Optional<int> inEvalSubsetSize,
 avl::Segment2D& outSegment,
 atl::Optional<atl::Array<avl::Point2D>&> outLTInliers = atl::NIL,
 atl::Optional<float&> outLTError = atl::NIL,
 int& diagIterationCount = atl::Dummy<int>()
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoints	const Array<Point2D>&			Input points
→	inRange	Optional<const Range&>		NIL	Determines which array points take part in fitting process
→	inSeedSubsetSize	int	2 - 10	3	Number of points in one combination for getting a sample segment
→	inEvalSubsetSize	Optional<int>	3 - ∞	NIL	Number of closest points used for evaluation of a sample segment, or Auto if seed points are to be used
←	outSegment	Segment2D&			Fitted segment
←	outLTInliers	Optional<Array<Point2D>&>		NIL	Inlying points of the best LTE segment
←	outLTError	Optional<float&>		NIL	The Least Trimmed Error
🔍	diagIterationCount	int&			Number of combinations considered

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outLTInliers**, **outLTError**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array of points in FitSegmentToPoints_LTE.
DomainError	Range exceeds the input point array in FitSegmentToPoints_LTE.

FitSegmentToPoints_RANSAC

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Approximates points with a segment using a RANSAC algorithm.

Applications: Finds a well matching segments, but for handling outliers requires a distance threshold that may be difficult to set.

Syntax

```
void avl::FitSegmentToPoints_RANSAC
(
 const atl::Array<avl::Point2D>& inPoints,
 atl::Optional<const avl::Range&> inRange,
 atl::Optional<int> inMaxOutlierCount,
 float inMaxInlierDistance,
 atl::Optional<int> inIterationCount,
 atl::Conditional<avl::Segment2D>& outSegment
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoints	const Array<Point2D>&			
→	inRange	Optional<const Range&>		NIL	Determines which array points take part in fitting process
→	inMaxOutlierCount	Optional<int>	0 - ∞	0	Determines how many outlier points can be present to end the search
→	inMaxInlierDistance	float	0.0 - ∞	3.0f	Distance from the output segment for a point to be considered an inlier
→	inIterationCount	Optional<int>	1 - ∞	42	Number of iterations; Auto means that all point pairs will be used
←	outSegment	Conditional<Segment2D>&			

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array of points in FitSegmentToPoints_RANSAC.
DomainError	Range exceeds the input point array in FitSegmentToPoints_RANSAC.

FitSegmentToPoints_TheilSen

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Approximates points with a segment using TheilSen algorithm, optionally with Siegel's improvement.

Applications: Finds a well matching segment, ignoring up to 29.3% (TheilSen) or 50.0% (Siegel) outliers. Outliers do have some influence on accuracy.

Syntax

```
void avl::FitSegmentToPoints_TheilSen
(
 const atl::Array<avl::Point2D>& inPoints,
 atl::Optional<const avl::Range&> inRange,
 avl::TheilSenVariant::Type inVariant,
 atl::Optional<int> inSampleLimit,
 atl::Optional<float> inOutlierRatio,
 avl::Segment2D& outSegment,
 atl::Array<float>& diagOrientations
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoints	const Array<Point2D>&			Input points
→	inRange	Optional<const Range&>		NIL	Determines which array points take part in fitting process
→	inVariant	TheilSenVariant::Type			Switches between Theil-Sen and Siegel methods
→	inSampleLimit	Optional<int>	5 - ∞	NIL	How many pairs of points are used to estimate orientation
→	inOutlierRatio	Optional<float>	0.0 - 0.99	NIL	
←	outSegment	Segment2D&			Fitted segment
↔	diagOrientations	Array<float>&			Sample orientations used to determine the output line orientation

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array of points in FitSegmentToPoints_TheilSen.
DomainError	Range exceeds the input point array in FitSegmentToPoints_TheilSen.

FitSegmentToRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Approximates a region with a segment using selected outliers suppression method.

Applications: Finding a locally optimal segment. Good enough when the number of outliers is small.

Syntax

```
void avl::FitSegmentToRegion
(
 const avl::Region& inRegion,
 const avl::RegionSegmentFittingMethod::Type inMethod,
 avl::Segment2D& outSegment
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inMethod	const RegionSegmentFittingMethod::Type		
←	outSegment	Segment2D&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Degenerate convex hull on input in FitSegmentToRegion.
DomainError	Empty region on input in FitSegmentToRegion.

LocateMultiplePointPatterns

Header: AVL.h
Namespace: avl
Module: FoundationPro

Finds occurrences of a pattern in a 2D cloud of (feature) points.

Applications: Can be used to find entire objects after finding their characteristic points with tools such as Template Matching or DL_LocatePoints.

Syntax

```
void avl::LocateMultiplePointPatterns
(
 const atl::Array<avl::Point2D>& inPoints,
 atl::Optional<const atl::Array<int>&> inPointLabels,
 const atl::Array<avl::Point2D>& inPattern,
 atl::Optional<const atl::Array<int>&> inPatternLabels,
 atl::Optional<const avl::Rectangle2D&> inReferenceFrame,
 const bool inAllowRotation,
 const float inMinAngle,
 const float inMaxAngle,
 const bool inAllowScale,
 const float inMinScale,
 const float inMaxScale,
 const float inTilingFactor,
 const float inMinInitialScore,
 const float inMaxDeviation,
 const float inMinScore,
 const float inMinDistance,
 const bool inDisjointObjectsOnly,
 atl::Array<avl::Rectangle2D>& outObjects,
 atl::Array<atl::Array<avl::Point2D> >& outAlignedPatterns,
 atl::Array<avl::CoordinateSystem2D>& outAlignments,
 atl::Array<atl::Array<avl::Segment2D> >& outPatternSkeletons
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoints	const Array<Point2D>&			Input points
→	inPointLabels	Optional<const Array<int>&>		NIL	Categories that the input points belong to
→	inPattern	const Array<Point2D>&			Point pattern to be found
→	inPatternLabels	Optional<const Array<int>&>		NIL	Categories that the pattern points belong to
→	inReferenceFrame	Optional<const Rectangle2D&>		NIL	Exact position of the model object associated with the pattern in the image
→	inAllowRotation	const bool		True	Flag indicating whether rotation is allowed as a part of output alignment
→	inMinAngle	const float		-180.0f	Start of range of possible rotations
→	inMaxAngle	const float		180.0f	End of range of possible rotations
→	inAllowScale	const bool		False	Flag indicating whether scale is allowed as a part of output alignment
→	inMinScale	const float	0.0 - ∞	0.8f	Start of range of possible scales
→	inMaxScale	const float	0.0 - ∞	1.25f	End of range of possible scales
→	inTilingFactor	const float	0.000001 - 1.0	0.2f	Defines relative size of the square tile on the plane during initial detection
→	inMinInitialScore	const float	0.0 - 1.0	0.75f	The minimum proportion of points correctly matched during initial detection
→	inMaxDeviation	const float	0.0 - ∞	5.0f	Maximal distance between two points considered matched
→	inMinScore	const float	0.0 - 1.0	0.75f	The minimum proportion of points correctly matched
→	inMinDistance	const float	0.0 - ∞	10.0f	Minimal distance between centers of two found occurrences
→	inDisjointObjectsOnly	const bool		False	Flag indicating whether found occurrences can have common points
←	outObjects	Array<Rectangle2D>&			Bounding rectangles of the found pattern occurrences
←	outAlignedPatterns	Array<Array<Point2D> >&			The aligned input pattern points
←	outAlignments	Array<CoordinateSystem2D>&			The transforms that align the input pattern to the input points
←	outPatternSkeletons	Array<Array<Segment2D> >&			The skeletons of the aligned input pattern points

Description

The filter finds locations of a pattern of points in the set of input points. The possible rotations and scales of the found occurrences can be fully controlled using proper values of **inAllowRotation**, **inMinAngle**, **inMaxAngle**, **inAllowScale**, **inMinScale** and **inMaxScale** parameters.

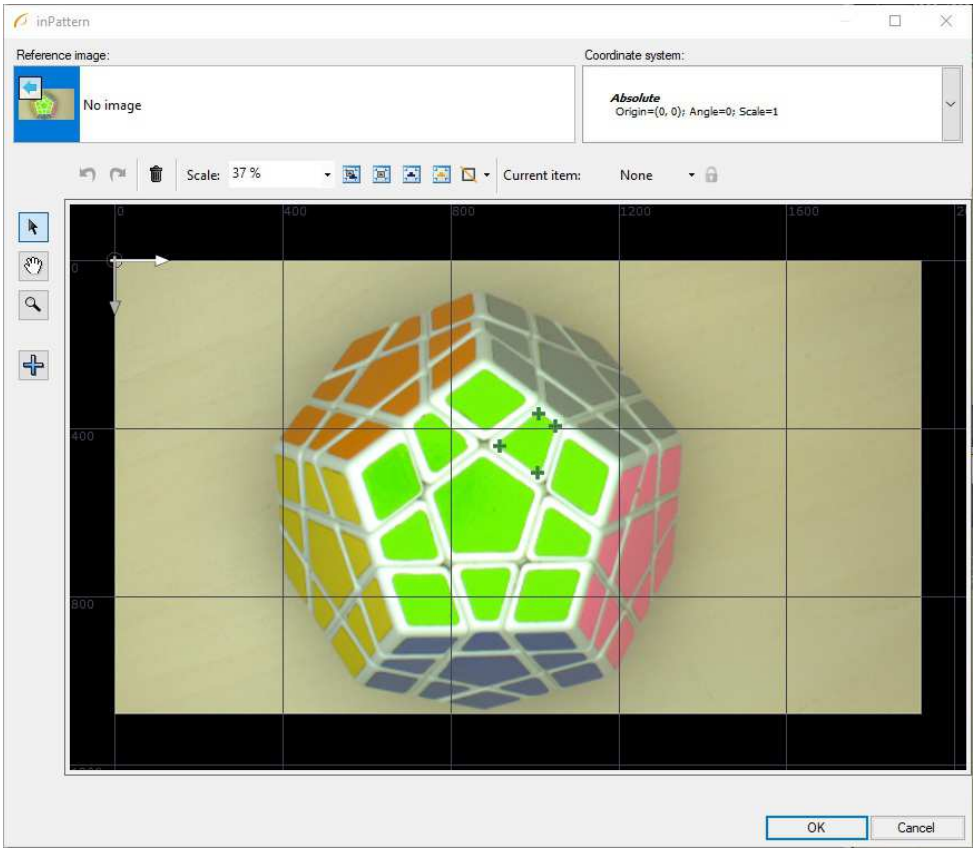
The location routine consists of three phases. Only the **inTilingFactor** and **inMinInitialScore** parameters have an effect on the initial phase. Internally, the whole plane is then divided into square tiles which size depends on **inTilingFactor** and the average distance between two points from **inPoints**. A transformation becomes a candidate to be a valid pattern occurrence if at least **inMinInitialScore** fraction of pattern points reside in the right tiles. The candidate transformation proceeds to the second phase, where it is refined to be possibly best fitted to the data points. The result transformation is considered to be a valid output alignment if at least **inMinScore** fraction of aligned pattern points are at most **inMaxDeviation** away from their closest data points.

The end phase's purpose is to select the output alignments according to the **inMinDistance** and **inDisjointObjectsOnly** parameters values. The ultimate output is chosen so no two aligned pattern centers are closer than **inMinDistance** from themselves and, if **inDisjointObjectsOnly** parameter is set, no two aligned patterns cover the same data point.

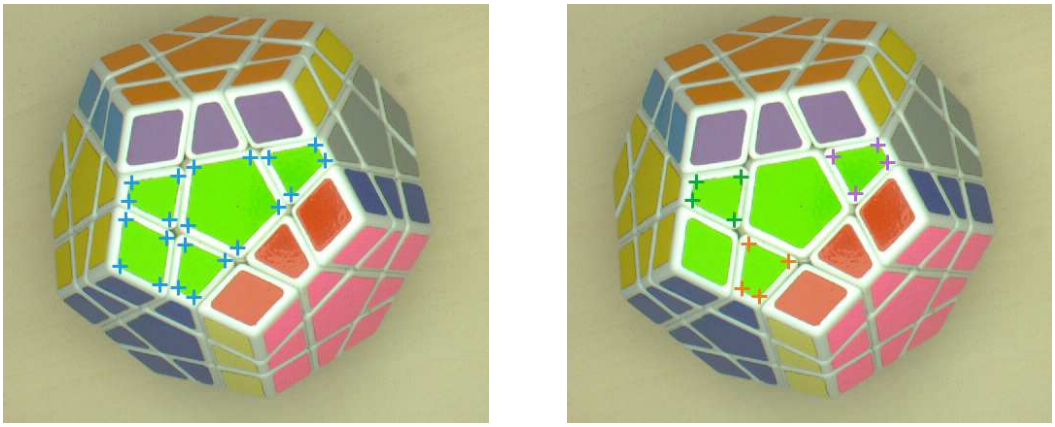
The most difficult part to achieve reliable results seems to be the proper setting of the **inTilingFactor** parameter. Because of its existence, the algorithm will work especially well when the pattern consists of not too many points and distances between them are more or less equal i.e. the ratio of the greatest distance between pattern points and the smallest distance between pattern points is small. If this is not the case and the default value for **inTilingFactor** does not work well, one should try to adjust the value keeping in mind that the greater values should be used when the pattern visible in the **inPoints** is distorted and the smaller values will work best when only small distortion is present. In case of further problems with choosing the right **inTilingFactor** value, one can also try lowering **inMinInitialScore** value.

The filter performance depends heavily on the number of the pattern points. Because of that fact, it is highly advisable for the pattern to be as small as possible. The performance can be poor even for patterns with more than 15 points. Note that the shape of the pattern also matters. The execution time for symmetrical patterns is generally bigger than for asymmetrical ones.

Examples



inPattern *input with selected point pattern to be found*



LocateMultiplePointPatterns *performed on points acquired by* ***DL_LocatePoints****.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Input pattern and input pattern labels sizes are not equal in LocateMultiplePointPatterns.
DomainError	Input pattern is empty in LocateMultiplePointPatterns.
DomainError	Input points and input point labels sizes are not equal in LocateMultiplePointPatterns.
DomainError	Point labels should be both present or both absent in LocateMultiplePointPatterns.

See Also

- LocateSinglePointPattern – Finds an occurrence of the pattern in the input points.

LocateSinglePointPattern

Header: AVL.h
Namespace: avl
Module: FoundationPro

Finds an occurrence of the pattern in the input points.

Applications: Can be used to find an entire object after finding its characteristic points with tools such as Template Matching or DL_LocatePoints.

Syntax

```
void avl::LocateSinglePointPattern
(
 const atl::Array<avl::Point2D>& inPoints,
 atl::Optional<const atl::Array<int>&> inPointLabels,
 const atl::Array<avl::Point2D>& inPattern,
 atl::Optional<const atl::Array<int>&> inPatternLabels,
 atl::Optional<const avl::Rectangle2D&> inReferenceFrame,
 const bool inAllowRotation,
 const float inMinAngle,
 const float inMaxAngle,
 const bool inAllowScale,
 const float inMinScale,
 const float inMaxScale,
 const float inTilingFactor,
 const float inMinInitialScore,
 const float inMaxDeviation,
 const float inMinScore,
 atl::Conditional<avl::Rectangle2D>& outObject,
 atl::Conditional<atl::Array<avl::Point2D> >& outAlignedPattern,
 atl::Conditional<avl::CoordinateSystem2D>& outAlignment,
 atl::Conditional<atl::Array<avl::Segment2D> >& outPatternSkeleton
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoints	const Array<Point2D>&			Input points
→	inPointLabels	Optional<const Array<int>&>		NIL	Categories that the input points belong to
→	inPattern	const Array<Point2D>&			Point pattern to be found
→	inPatternLabels	Optional<const Array<int>&>		NIL	Categories that the pattern points belong to
→	inReferenceFrame	Optional<const Rectangle2D&>		NIL	Exact position of the model object associated with the pattern in the image
→	inAllowRotation	const bool		True	Flag indicating whether rotation is allowed as a part of output alignment
→	inMinAngle	const float		-180.0f	Start of range of possible rotations
→	inMaxAngle	const float		180.0f	End of range of possible rotations
→	inAllowScale	const bool		False	Flag indicating whether scale is allowed as a part of output alignment
→	inMinScale	const float	0.0 - ∞	0.8f	Start of range of possible scales
→	inMaxScale	const float	0.0 - ∞	1.25f	End of range of possible scales
→	inTilingFactor	const float	0.000001 - 1.0	0.2f	Defines relative size of the square tile on the plane during initial detection
→	inMinInitialScore	const float	0.0 - 1.0	0.75f	The minimum proportion of points correctly matched during initial detection
→	inMaxDeviation	const float	0.0 - ∞	5.0f	Maximal distance between two points considered matched
→	inMinScore	const float	0.0 - 1.0	0.75f	The minimum proportion of points correctly matched
←	outObject	Conditional<Rectangle2D>&			Bounding rectangle of the found pattern occurrence
←	outAlignedPattern	Conditional<Array<Point2D> >&			The aligned input pattern points
←	outAlignment	Conditional<CoordinateSystem2D>&			The transform that align the input pattern to the input points
←	outPatternSkeleton	Conditional<Array<Segment2D> >&			The skeleton of the aligned input pattern points

Description

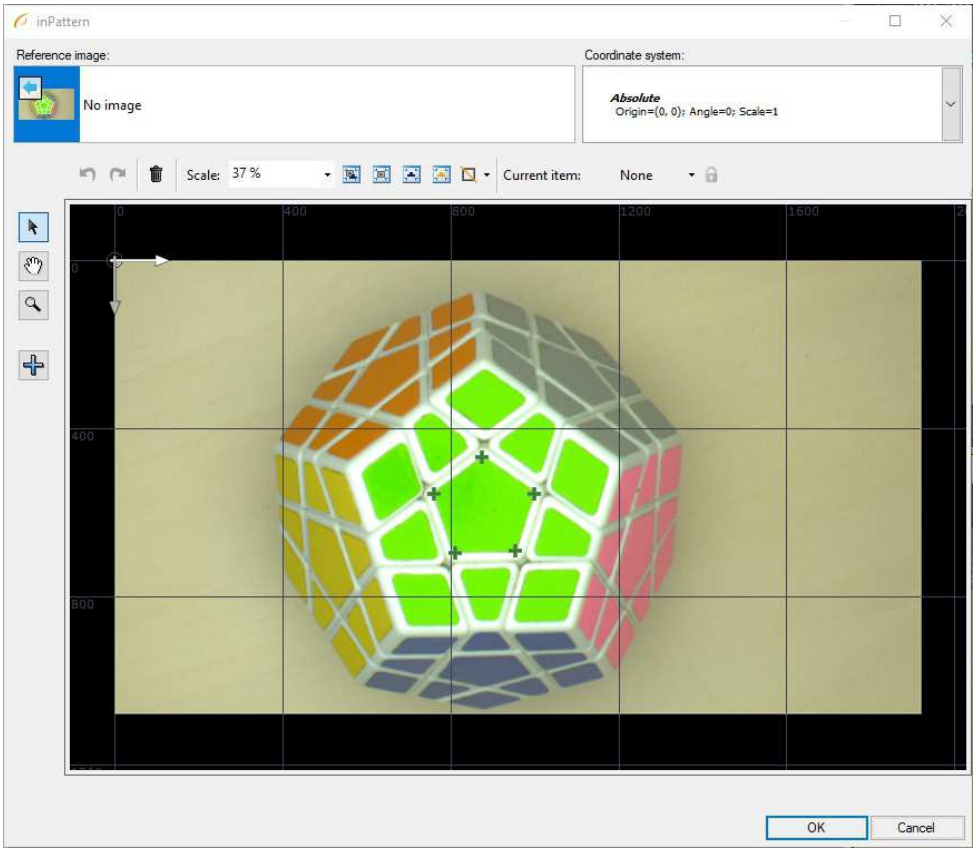
The filter finds location of a pattern of points in the set of input points. The possible rotation and scale of the found occurrence can be fully controlled using proper values of **inAllowRotation**, **inMinAngle**, **inMaxAngle**, **inAllowScale**, **inMinScale** and **inMaxScale** parameters.

The location routine consists of two phases. Only the **inTilingFactor** and **inMinInitialScore** parameters have an effect on the initial phase. Internally, the whole plane is then divided into square tiles which size depends on **inTilingFactor** and the average distance between two points from **inPoints**. A transformation becomes a candidate to be a valid pattern occurrence if at least **inMinInitialScore** fraction of pattern points reside in the right tiles. The candidate transformation proceeds to the second phase, where it is refined to be possibly best fitted to the data points. The result transformation is considered to be a valid output alignment if at least **inMinScore** fraction of aligned pattern points are at most **inMaxDeviation** away from their closest data points.

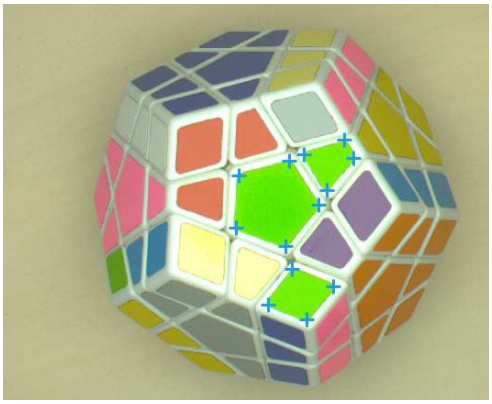
The most difficult part to achieve reliable results seems to be the proper setting of the **inTilingFactor** parameter. Because of its existence, the algorithm will work especially well when the pattern consists of not too many points and distances between them are more or less equal i.e. the ratio of the greatest distance between pattern points and the smallest distance between pattern points is small. If this is not the case and the default value for **inTilingFactor** does not work well, one should try to adjust the value keeping in mind that the greater values should be used when the pattern visible in the **inPoints** is distorted and the smaller values will work best when only small distortion is present. In case of further problems with choosing the right **inTilingFactor** value, one can also try lowering **inMinInitialScore** value.

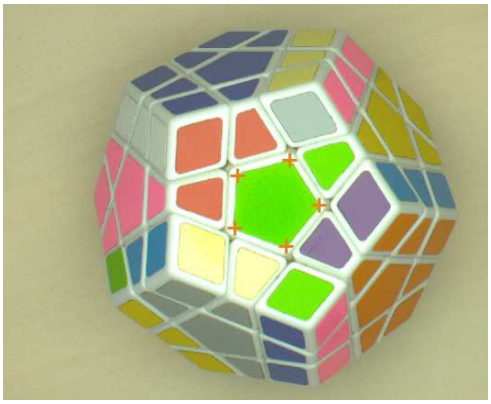
The filter performance depends heavily on the number of the pattern points. Because of that fact, it is highly advisable for the pattern to be as small as possible. The performance can be poor even for patterns with more than 15 points. Note that the shape of the pattern also matters. The execution time for symmetrical patterns is generally bigger than for asymmetrical ones.

Examples



inPattern *input with selected point pattern to be found*





LocateSinglePointPattern *performed on points acquired by* ***DL_LocatePoints***

Errors

List of possible exceptions:

Error type	Description
DomainError	Input pattern and input pattern labels sizes are not equal in LocateSinglePointPattern.
DomainError	Input pattern is empty in LocateSinglePointPattern.
DomainError	Input points and input point labels sizes are not equal in LocateSinglePointPattern.
DomainError	Point labels should be both present or both absent in LocateSinglePointPattern.

See Also

- LocateMultiplePointPatterns – Finds occurrences of a pattern in a 2D cloud of (feature) points.

19. Point3DGrid Fitting

Table of content:

AdjustPointGrids3D

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Aligns (rotation + translation) a point grid to match best the reference point grid.

Syntax

```
void avl::AdjustPointGrids3D
(
 const avl::Point3DGrid& inPoints,
 const avl::Point3DGrid& inReferencePoints,
 const int inMaxIterationCount,
 const float inMatchFraction,
 const float inDiscardFurthestFraction,
 avl::Point3DGrid& outAlignedPoints,
 avl::Matrix& outAlignment,
 atl::Array<avl::Point3D>& diagPointsWorkingSet,
 atl::Array<avl::Point3D>& diagReferencePointsWorkingSet,
 atl::Array<atl::Array<avl::Segment3D>>& diagAttractionSegments
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoints	const Point3DGrid&			Data points to be aligned, may contain background clutter.
→	inReferencePoints	const Point3DGrid&			Reference points to align to, may only contain points belonging to the object of interest.
→	inMaxIterationCount	const int	1 - ∞	100	Maximal number of iteration for the algorithm
→	inMatchFraction	const float	0.0 - 1.0	0.1111111111111111f	Defines fraction of input points that is being fitted in every iteration
→	inDiscardFurthestFraction	const float	0.0 - 1.0	0.05f	Fraction of point pairs to be discarded during internal ICP loop. Furthest pairs are discarded, and only in last 10% of algorithm iterations. Useful for handling outliers in the inReferencePoints, as well as gaps/holes in the inPoints data.
←	outAlignedPoints	Point3DGrid&			The aligned input points
←	outAlignment	Matrix&			The transform that aligns best the input points to the reference points
🔍	diagPointsWorkingSet	Array<Point3D>&			inPoints after decimation by inMatchFraction, i.e. points which will be fitted in every iteration
🔍	diagReferencePointsWorkingSet	Array<Point3D>&			inReferencePoints after decimation by inMatchFraction, i.e. points which will be fitted in every iteration
🔍	diagAttractionSegments	Array<Array<Segment3D>>&			Attraction segments from every iteration. Note that the algorithm internally moves inReferencePoints towards inPoints (and returns inverse transformation), so the segments show movement of reference towards data.

Hints

- Usually the **inPoints** Point3DGrid data contains an object with background - a flat plate or some other clutter. The filter handles such cases, however its best to remove this clutter beforehand. Note: The **inReferencePoints** input must contain only points belonging to the object.
- The filter can not be used as a pattern localization tool, as it performs well only when adjustments to be made are relatively small. For problems where large adjustments are needed (especially rotations) the filter may fail.

Errors

List of possible exceptions:

Error type	Description
DomainError	Input point array is empty in AdjustPointGrids3D.

AdjustPointGrids3DGlobal

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Aligns (rotation + translation) a point grid to match best the reference point grid using a global registration algorithm.

Syntax

```
void avl::AdjustPointGrids3DGlobal
(
 const avl::Point3DGrid& inPoints,
 const avl::Point3DGrid& inReferencePoints,
 const int inSampleCount,
 const float inOverlap,
 const float inDeltaCorrection,
 atl::Optional<const float&> inMaxAngle,
 atl::Optional<const float&> inMaxTranslationDistance,
 atl::Optional<const int&> inSeed,
 avl::Point3DGrid& outAlignedPoints,
 avl::Matrix& outAlignment,
 float& outDelta
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoints	const Point3DGrid&			Data points to be aligned, may contain background clutter.
→	inReferencePoints	const Point3DGrid&			Reference points to align to, may contain background clutter.
→	inSampleCount	const int	1 - ∞	500	Number of random samples used by the registration algorithm. Recommended values are from 200 to several thousands.
→	inOverlap	const float	0.0 - 1.0	0.8f	Defines the estimated overlap between two input grids (0 = no overlap, 1.0 = every point in inPoints is also in inReferencePoints).
→	inDeltaCorrection	const float	0.0 - ∞	9.3f	Defines the accuracy of the final alignment. With smaller delta the amount of allocated memory increases.
→	inMaxAngle	Optional<const float&>	0.0 - 360.0	NIL	Limits the maximum rotation angle of the final transform.
→	inMaxTranslationDistance	Optional<const float&>	0.0 - ∞	NIL	Limits the length of the translation vector of the final transform.
→	inSeed	Optional<const int&>		NIL	Seed for a random generator used by the algorithm.
←	outAlignedPoints	Point3DGrid&			The aligned input points.
←	outAlignment	Matrix&			The transform that aligns best the input points to the reference points.
←	outDelta	float&			Returns estimated value of the parameter delta.

Errors

List of possible exceptions:

Error type	Description
DomainError	Input point array is empty in AdjustPointGrids3DGlobal.
DomainError	Parameter inDeltaCorrection in AdjustPointGrids3DGlobal has to be greater than 0.0.

FitPlaneToPoint3DGrid

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Approximates points of the input grid with a plane using the Least Squares method.

Syntax

```
void avl::FitPlaneToPoint3DGrid
(
 const avl::Point3DGrid& inPointGrid,
 atl::Optional<const avl::Region&> inRoi,
 avl::Plane3D& outPlane,
 atl::Optional<atl::Array<float>&> outDistances = atl::NIL,
 atl::Optional<float&> outSignedDistanceSum = atl::NIL,
 atl::Optional<float&> outDistanceSum = atl::NIL,
 atl::Optional<atl::Array<float>&> outSquaredDistances = atl::NIL,
 atl::Optional<float&> outSquaredDistanceSum = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPointGrid	const Point3DGrid&		Input point grid
→	inRoi	Optional<const Region&>	NIL	Region of interest
←	outPlane	Plane3D&		Fitted plane
←	outDistances	Optional<Array<float>&>	NIL	Distances of the input grid points to a resulting plane
←	outSignedDistanceSum	Optional<float&>	NIL	Sum of signed distances of the input grid points to a resulting plane
←	outDistanceSum	Optional<float&>	NIL	Sum of distances of the input grid points to a resulting plane
←	outSquaredDistances	Optional<Array<float>&>	NIL	Squared distances of the input grid points to a resulting plane
←	outSquaredDistanceSum	Optional<float&>	NIL	Sum of squared distances of the input grid points to a resulting plane

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outDistances**, **outSignedDistanceSum**, **outDistanceSum**, **outSquaredDistances**, **outSquaredDistanceSum**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	No points to fit the plane to in FitPlaneToPoint3DGrid.

FitPlaneToPoint3DGrid_M

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Approximates points of the input grid with a plane using selected M-estimator for outlier suppression.

Applications: Finding a locally optimal plane. Good enough when the number of outliers is small.

Syntax

```
void avl::FitPlaneToPoint3DGrid_M
(
 const avl::Point3DGrid& inPointGrid,
 atl::Optional<const avl::Region&> inRoi,
 avl::MEstimator::Type inOutlierSuppression,
 float inClippingFactor,
 int inIterationCount,
 atl::Optional<const avl::Plane3D&> inInitialPlane,
 avl::Plane3D& outPlane,
 atl::Optional<atl::Array<avl::Point3D>&> outInliers = atl::NIL,
 atl::Optional<atl::Array<float>&> outDistances = atl::NIL,
 atl::Optional<float&> outSignedDistanceSum = atl::NIL,
 atl::Optional<float&> outDistanceSum = atl::NIL,
 atl::Optional<atl::Array<float>&> outSquaredDistances = atl::NIL,
 atl::Optional<float&> outSquaredDistanceSum = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPointGrid	const Point3DGrid&			Input point grid
→	inRoi	Optional<const Region&>		NIL	Region of interest
→	inOutlierSuppression	MEstimator::Type			Selects a method for ignoring points being much different from the rest
→	inClippingFactor	float	0.675 - 6.0	2.5f	Multitude of standard deviation within which points are considered inliers
→	inIterationCount	int	0 - ∞	5	Number of iterations of outlier suppressing algorithm
→	inInitialPlane	Optional<const Plane3D&>		NIL	Initial approximation (if available)
←	outPlane	Plane3D&			Fitted plane
←	outInliers	Optional<Array<Point3D>&>		NIL	Points matching the computed plane
←	outDistances	Optional<Array<float>&>		NIL	Distances of the input grid points to a resulting plane
←	outSignedDistanceSum	Optional<float&>		NIL	Sum of signed distances of the input grid points to a resulting plane
←	outDistanceSum	Optional<float&>		NIL	Sum of distances of the input grid points to a resulting plane
←	outSquaredDistances	Optional<Array<float>&>		NIL	Squared distances of the input grid points to a resulting plane
←	outSquaredDistanceSum	Optional<float&>		NIL	Sum of squared distances of the input grid points to a resulting plane

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outInliers**, **outDistances**, **outSignedDistanceSum**, **outDistanceSum**, **outSquaredDistances**, **outSquaredDistanceSum**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	No points to fit the plane to in FitPlaneToPoint3DGrid_M.

GoldenTemplate3D

Header: AVL.h
Namespace: avl
Module: Vision3DPro

Compares points on the input with the golden object. Any significant differences are considered defects.

Syntax

```
void avl::GoldenTemplate3D
(
 const avl::Point3DGrid& inObject,
 const avl::Point3DGrid& inGoldenObject,
 const float inMaxDistance,
 const float inVoxelSize,
 const int inNeighborCount,
 const int inSampleCount,
 const float inDeltaCorrection,
 atl::Optional<const float&> inMaxAngle,
 atl::Optional<const float&> inMaxTranslationDistance,
 const float inOverlap,
 atl::Array<avl::Point3D>& outMissing,
 atl::Array<avl::Point3D>& outExcessive,
 avl::Matrix& outAlignment,
 avl::Point3DGrid& diagCleanedGoldenObject,
 avl::Point3DGrid& diagAlignedObject
)
```

Parameters

	Name	Type	Range	Default	Description
→	inObject	const Point3DGrid&			Input Point3DGrid
→	inGoldenObject	const Point3DGrid&			Point3DGrid with the golden object
→	inMaxDistance	const float	0.0 - ∞		Maximal allowed distance between corresponding vertices of the input and the golden object
→	inVoxelSize	const float	0.0 - ∞	0.0f	Defines a voxel size used to subsample both grids.
→	inNeighborCount	const int	1 - ∞	15	Defines number of neighbors used to compute average distances between vertices during noise removal.
→	inSampleCount	const int	1 - ∞	5000	Defines number of samples used by the registration algorithm.
→	inDeltaCorrection	const float	0.0 - ∞	9.3f	Defines the accuracy of the alignment. With smaller delta the amount of allocated memory increases.
→	inMaxAngle	Optional<const float&>	0.0 - 360.0	NIL	Limits the maximum rotation angle of the final transform.
→	inMaxTranslationDistance	Optional<const float&>	0.0 - ∞	NIL	Limits the length of the translation vector of the final transform.
→	inOverlap	const float	0.0 - 1.0	0.95f	Defines the estimated overlap between two input grids (0 = no overlap, 1.0 = every point in inObject is also in inGoldenObject).
←	outMissing	Array<Point3D>&			Points from the golden object not present in the input object.
←	outExcessive	Array<Point3D>&			Points from the input object not present in the golden object.
←	outAlignment	Matrix&			The transform that aligns best the input object to the golden object.
🔍	diagCleanedGoldenObject	Point3DGrid&			Golden object after subsampling and noise removal.
🔍	diagAlignedObject	Point3DGrid&			Aligned and cleaned input object

Errors

List of possible exceptions:

Error type	Description
DomainError	Grids are empty after noise removal in GoldenTemplate3D.
DomainError	inNeighborCount is larger than the number of valid points in GoldenTemplate3D.
DomainError	Input grid has no valid points in GoldenTemplate3D.
DomainError	Input grid is empty in GoldenTemplate3D.
DomainError	Parameter inDeltaCorrection in GoldenTemplate3D has to be greater than 0.0
DomainError	Parameter inVoxelSize in GoldenTemplate3D has to be greater than or equal to 0.0

Point3DGridDistance

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Computes distances between two input grids.

Syntax

```
void avl::Point3DGridDistance
(
 const avl::Point3DGrid& inReference,
 const avl::Point3DGrid& inCompare,
 atl::Optional<const avl::Region&> inRoi,
 const float inThreshold,
 atl::Array<avl::Point3D>& outCorrectPoints,
 atl::Array<avl::Point3D>& outIncorrectPoints,
 float& outMaxDistance,
 float& outMinDistance,
 atl::Array<float>& outDistances
)
```

Parameters

	Name	Type	Range	Default	Description
→	inReference	const Point3DGrid&			Reference grid
→	inCompare	const Point3DGrid&			Data points for which the distances will be computed.
→	inRoi	Optional<const Region&>		NIL	Range of points for which the distance will be computed.
→	inThreshold	const float	0.0 - ∞		Defines for each data point a maximum allowed distance to the reference grid.
←	outCorrectPoints	Array<Point3D>&			All data points with distance to the reference grid smaller that the given threshold
←	outIncorrectPoints	Array<Point3D>&			All data points with distance to reference grid greater that the given threshold
←	outMaxDistance	float&			Maximum computed distance
←	outMinDistance	float&			Minimum computed distance
←	outDistances	Array<float>&			All computed distances

Errors

List of possible exceptions:

Error type	Description
DomainError	Input point array is empty in Point3DGridDistance.
DomainError	Region of interest exceeds an input Point3DGrid in Point3DGridDistance.

Point3DGridRMSE

Header: AVL.h
Namespace: avl
Module: Vision3DPro

Computes point-to-point RMSE between two point grids.

Syntax

```
void avl::Point3DGridRMSE
(
 const avl::Point3DGrid& inReference,
 const avl::Point3DGrid& inPoints,
 double& outRMSE
)
```

Parameters

	Name	Type	Default	Description
→	inReference	const Point3DGrid&		Reference grid
→	inPoints	const Point3DGrid&		Data points for which the RMSE will be computed.
←	outRMSE	double&		Root-mean-square error of distances between inPoints and inReference.

Errors

List of possible exceptions:

Error type	Description
DomainError	Input grid has no valid points in Point3DGridRMSE.
DomainError	Input grid is empty inPoint3DGridRMSE.

RemoveOutliersFromPoint3DGrid

Header: AVL.h
Namespace: avl
Module: Vision3DPro

Removes noise from a given Point3DGrid. All points that are too far from their neighbors are removed.

Syntax

```
void avl::RemoveOutliersFromPoint3DGrid
(
 const avl::Point3DGrid& inPointGrid,
 atl::Optional<const avl::Region&> inRoi,
 const float inStdDevMultiple,
 const int inNeighborCount,
 avl::Point3DGrid& outGrid,
 float& diagThreshold,
 float& diagAverageDistance
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPointGrid	const Point3DGrid&			Input Point3DGrid
→	inRoi	Optional<const Region&>		NIL	Region of interest
→	inStdDevMultiple	const float	0.0 - ∞	1.5f	Allows to set a threshold based on standard deviation of average distances between points on input.
→	inNeighborCount	const int	1 - ∞	10	Defines number of neighbors used to compute average distances between vertices.
←	outGrid	Point3DGrid&			Cleaned input
🔍	diagThreshold	float&			Threshold used by the algorithm.
🔍	diagAverageDistance	float&			Average distance between input points and their neighborhood

Errors

List of possible exceptions:

Error type	Description
DomainError	inNeighborCount is larger than the number of valid points in RemoveOutliersFromPoint3DGrid.
DomainError	Input grid has no valid points in RemoveOutliersFromPoint3DGrid.
DomainError	Input grid is empty in RemoveOutliersFromPoint3DGrid.

VoxelizePoint3DGrid

Header: AVL.h
Namespace: avl
Module: Vision3DPro

Keeps one point from each occupied voxel of a given size. Value of all remaining grid points is set to invalid.

Syntax

```
void avl::VoxelizePoint3DGrid
(
 const avl::Point3DGrid& inPointGrid,
 atl::Optional<const avl::Region&> inRoi,
 const float inVoxelSize,
 avl::Point3DGrid& outGrid
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPointGrid	const Point3DGrid&			Input Point3DGrid
→	inRoi	Optional<const Region&>		NIL	Region of interest
→	inVoxelSize	const float	0.0 - ∞	0.5f	Defines a voxel size used to subsample the grid.
←	outGrid	Point3DGrid&			Subsampled points

Errors

List of possible exceptions:

Error type	Description
DomainError	Input grid is empty in VoxelizePoint3DGrid.
DomainError	Parameter inVoxelSize in VoxelizePoint3DGrid has to be greater than 0.0

20. Geometry 2D Spatial Transforms

Table of content:

- SplitSegment
- TranslateArc
- TranslateCircle
- TranslateCoordinateSystem
- TranslateLine
- TranslatePoint
- TranslatePointArray
- TranslatePoint_Toward
- TranslateRectangle
- TranslateSegment
- TrimLine
- TrimLineToRectangle
- TrimPath
- TrimPathArray
- TrimPathArrayToRectangle
- TrimPathToRectangle
- TrimPointArray
- TrimPointArrayToRegion
- TrimSegment
- TrimSegmentToCircle
- TrimSegmentToPolygon
- TrimSegmentToRectangle

AlignArc

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Moves an arc from a local coordinate system to the absolute one.

Applications: Required when there is an arc defined in a local coordinate system, but the next image-related filter in the program does not have any inAlignment input.

Syntax

```
void avl::AlignArc
(
 const avl::Arc2D& inArc,
 const avl::CoordinateSystem2D& inAlignment,
 bool inInverse,
 avl::Arc2D& outAlignedArc
)
```

Parameters

	Name	Type	Default	Description
→	inArc	const Arc2D&		
→	inAlignment	const CoordinateSystem2D&		Coordinate system to align to
→	inInverse	bool		Switches to the inverse transform
←	outAlignedArc	Arc2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inArc** and **outAlignedArc**

Read more about In-place Computation.

Description

AlignArc aligns the **inArc** to the **inAlignment** coordinate system. The input arc is translated, rotated and scaled.

The **inAlignment** is usually a coordinate system found by some template matching algorithm.

Examples

AlignArc *performed on the sample arc. The* ***inAlignment*** *is drawn on the first image in blue.*

See Also

- RotateArc – Rotates an arc clockwise around center point.
- TranslateArc – Translates an arc by a vector.
- AlignPath – Moves a path from a local coordinate system to the absolute one.

AlignCircle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Moves a circle from a local coordinate system to the absolute one.

Applications: Required when there is a circle defined in a local coordinate system, but the next image-related filter in the program does not have any inAlignment input.

Syntax

```
void avl::AlignCircle
(
 const avl::Circle2D& inCircle,
 const avl::CoordinateSystem2D& inAlignment,
 bool inInverse,
 avl::Circle2D& outAlignedCircle
)
```

Parameters

	Name	Type	Default	Description
→	inCircle	const Circle2D&		
→	inAlignment	const CoordinateSystem2D&		Coordinate system to align to
→	inInverse	bool		Switches to the inverse transform
←	outAlignedCircle	Circle2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inCircle** and **outAlignedCircle**

Read more about In-place Computation.

Description

AlignCircle aligns the **inCircle** to the **inAlignment** coordinate system. The input circle is translated, rotated and scaled.

The **inAlignment** is usually a coordinate system found by some template matching algorithm.

Examples

AlignCircle *performed on the sample circle. The* ***inAlignment*** *is drawn on the first image in blue.*

See Also

- RotateCircle – Rotates a circle clockwise around a center point.
- TranslateCircle – Translates a circle by a vector.
- AlignPath – Moves a path from a local coordinate system to the absolute one.

AlignCoordinateSystem

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Moves a nested coordinate system from its local coordinate system to the absolute one.

Applications: This filter is required e.g. when we first locate an object and then we locate its parts within it.

Syntax

```
void avl::AlignCoordinateSystem
(
 const avl::CoordinateSystem2D& inCoordinateSystem,
 const avl::CoordinateSystem2D& inAlignment,
 bool inInverse,
 avl::CoordinateSystem2D& outAlignedCoordinateSystem
)
```

Parameters

	Name	Type	Default	Description
→	inCoordinateSystem	const CoordinateSystem2D&		
→	inAlignment	const CoordinateSystem2D&		
→	inInverse	bool		Switches to the inverse operation
←	outAlignedCoordinateSystem	CoordinateSystem2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inCoordinateSystem** and **outAlignedCoordinateSystem**, **inAlignment** and **outAlignedCoordinateSystem**

Read more about In-place Computation.

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

AlignLine

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Moves a line from a local coordinate system to the absolute one.

Applications: Required when there is a line defined in a local coordinate system, but the next image-related filter in the program does not have any inAlignment input.

Syntax

```
void avl::AlignLine
(
 const avl::Line2D& inLine,
 const avl::CoordinateSystem2D& inAlignment,
 bool inInverse,
 avl::Line2D& outAlignedLine
)
```

Parameters

	Name	Type	Default	Description
→	inLine	const Line2D&		
→	inAlignment	const CoordinateSystem2D&		Coordinate system to align to
→	inInverse	bool		Switches to the inverse transform
←	outAlignedLine	Line2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inLine** and **outAlignedLine**

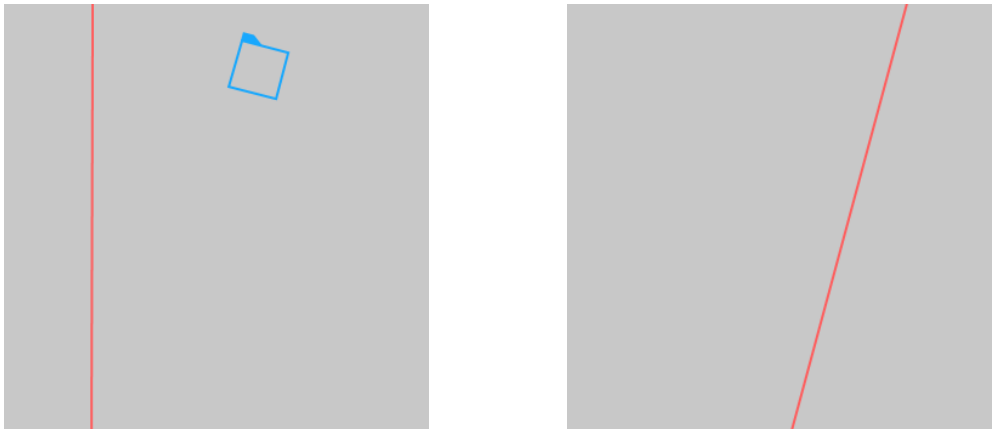
Read more about In-place Computation.

Description

AlignLine aligns the **inLine** to the **inAlignment** coordinate system. The input line is translated, rotated and scaled.

The **inAlignment** is usually a coordinate system found by some template matching algorithm.

Examples



***AlignLine** performed on the sample line. The **inAlignment** is drawn on the first image in blue.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite line on input in AlignLine.

See Also

- RotateLine – Rotates a line clockwise around a center point.
- TranslateLine – Translates a line by a vector.
- AlignPath – Moves a path from a local coordinate system to the absolute one.

AlignPoint

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Moves a point from a local coordinate system to the absolute one.

Applications: Required when there is a point defined in a local coordinate system, but the next image-related filter in the program does not have any inAlignment input.

Syntax

```
void avl::AlignPoint
(
 const avl::Point2D& inPoint,
 const avl::CoordinateSystem2D& inAlignment,
 bool inInverse,
 avl::Point2D& outAlignedPoint
)
```

Parameters

	Name	Type	Default	Description
→	inPoint	const Point2D&		
→	inAlignment	const CoordinateSystem2D&		Coordinate system to align to
→	inInverse	bool		Switches to the inverse transform
←	outAlignedPoint	Point2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inPoint** and **outAlignedPoint**

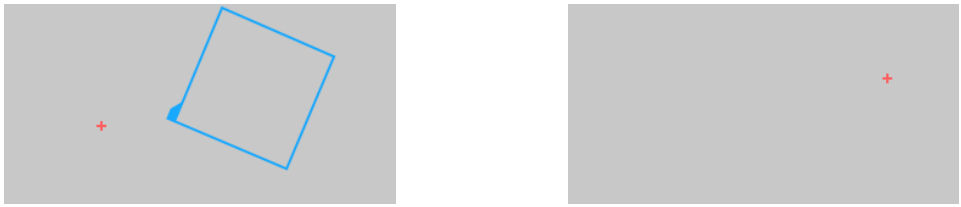
Read more about In-place Computation.

Description

AlignPoint aligns the **inPoint** to the **inAlignment** coordinate system. The input point is translated, rotated and scaled.

The **inAlignment** is usually a coordinate system found by some template matching algorithm.

Examples



AlignPoint *performed on the sample point. The* ***inAlignment*** *is drawn on the first image in blue.*

See Also

- RotatePoint – Rotates a point clockwise around a center point.
- TranslatePoint – Translates a point by a vector.
- AlignPath – Moves a path from a local coordinate system to the absolute one.

AlignPointArray

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Moves an array of points from a local coordinate system to the absolute one.

Applications: Required when there are points defined in a local coordinate system, but the next image-related filter in the program does not have any inAlignment input.

Syntax

```
void avl::AlignPointArray
(
 const atl::Array<avl::Point2D>& inPoints,
 const avl::CoordinateSystem2D& inAlignment,
 bool inInverse,
 atl::Array<avl::Point2D>& outAlignedPoints
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		
→	inAlignment	const CoordinateSystem2D&		Coordinate system to align to
→	inInverse	bool		Switches to the inverse transform
←	outAlignedPoints	Array<Point2D>&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inPoints** and **outAlignedPoints**

Read more about In-place Computation.

AlignRectangle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Moves a rectangle from a local coordinate system to the absolute one.

Applications: Required when there is a rectangle defined in a local coordinate system, but the next image-related filter in the program does not have any inAlignment input.

Syntax

```
void avl::AlignRectangle
(
 const avl::Rectangle2D& inRectangle,
 const avl::CoordinateSystem2D& inAlignment,
 bool inInverse,
 avl::Rectangle2D& outAlignedRectangle
)
```

Parameters

	Name	Type	Default	Description
➡	inRectangle	const Rectangle2D&		
➡	inAlignment	const CoordinateSystem2D&		Coordinate system to align to
➡	inInverse	bool		Switches to the inverse transform
⬅	outAlignedRectangle	Rectangle2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inRectangle** and **outAlignedRectangle**

Read more about In-place Computation.

Description

AlignRectangle aligns the **inRectangle** to the **inAlignment** coordinate system. The input rectangle is translated, rotated and scaled.

The **inAlignment** is usually a coordinate system found by some template matching algorithm.

Examples

AlignRectangle *performed on the sample rectangle. The* ***inAlignment*** *is drawn on the first image in blue.*

See Also

- RotateRectangle – Rotates a rectangle clockwise around a center point.
- TranslateRectangle – Translates a rectangle by a vector.
- AlignPath – Moves a path from a local coordinate system to the absolute one.

AlignSegment

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Moves a segment from a local coordinate system to the absolute one.

Applications: Required when there is a segment defined in a local coordinate system, but the next image-related filter in the program does not have any inAlignment input.

Syntax

```
void avl::AlignSegment
(
 const avl::Segment2D& inSegment,
 const avl::CoordinateSystem2D& inAlignment,
 bool inInverse,
 avl::Segment2D& outAlignedSegment
)
```

Parameters

	Name	Type	Default	Description
➡	inSegment	const Segment2D&		
➡	inAlignment	const CoordinateSystem2D&		Coordinate system to align to
➡	inInverse	bool		Switches to the inverse transform
⬅	outAlignedSegment	Segment2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inSegment** and **outAlignedSegment**

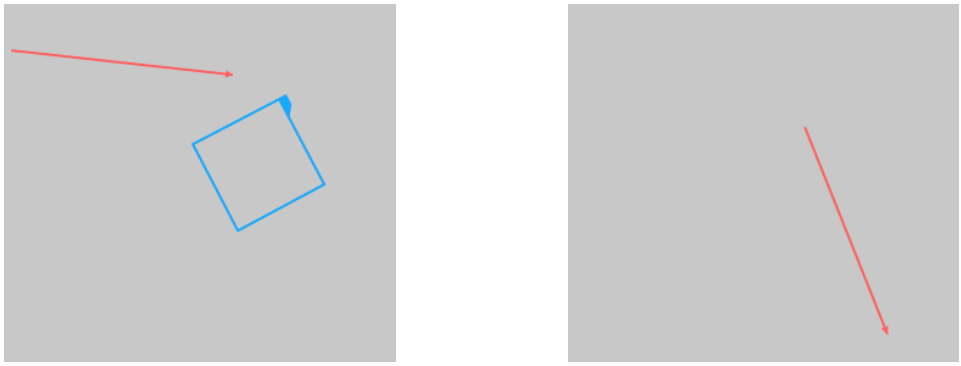
Read more about In-place Computation.

Description

AlignSegment aligns the **inSegment** to the **inAlignment** coordinate system. The input segment is translated, rotated and scaled.

The **inAlignment** is usually a coordinate system found by some template matching algorithm.

Examples



***AlignSegment** performed on the sample segment. The **inAlignment** is drawn on the first image in blue.*

See Also

- RotateSegment – Rotates a segment clockwise around a center point.
- TranslateSegment – Translates a segment by a vector.
- AlignPath – Moves a path from a local coordinate system to the absolute one.

Also in AVL Lite

InflateRectangle

Header: AVL.h
Namespace: avl
Module: FoundationLite

Enlarges a rectangle by a given margin.

Syntax

```
void avl::InflateRectangle
(
 const avl::Rectangle2D& inRectangle,
 float inMargin,
 avl::Rectangle2D& outRectangle
)
```

Parameters

	Name	Type	Default	Description
→	inRectangle	const Rectangle2D&		
→	inMargin	float		
←	outRectangle	Rectangle2D&		

Also in AVL Lite

InvertCoordinateSystem

Header: AVL.h
Namespace: avl
Module: FoundationLite

Inverts a coordinate system.

Syntax

```
void avl::InvertCoordinateSystem
(
 const avl::CoordinateSystem2D& inCoordinateSystem,
 avl::CoordinateSystem2D& outInvertedCoordinateSystem
)
```

Parameters

	Name	Type	Default	Description
→	inCoordinateSystem	const CoordinateSystem2D&		
←	outInvertedCoordinateSystem	CoordinateSystem2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inCoordinateSystem** and **outInvertedCoordinateSystem**

Read more about In-place Computation.

Also in AVL Lite

PointAlongArc

Header: AVL.h
Namespace: avl
Module: FoundationLite

Transforms a point to a coordinate system in which the 'axis' arc is vertical or horizontal.

Applications: Usually used to revert an ImageAlongArc transformation.

Syntax

```
void avl::PointAlongArc
(
 const avl::Point2D& inPoint,
 const avl::Arc2D& inAxis,
 avl::Axis::Type inAxisType,
 float inAxisCoordinate,
 bool inInverse,
 avl::Point2D& outPoint
)
```

Parameters

	Name	Type	Default	Description
→	inPoint	const Point2D&		Input point
→	inAxis	const Arc2D&		Input axis arc
→	inAxisType	Axis::Type	Y	Type of axis the input axis arc is parallel to
→	inAxisCoordinate	float	0.0f	Coordinate of the axis arc
→	inInverse	bool	True	Switches to the inverse operation
←	outPoint	Point2D&		Transformed point

Description

This operation transforms a point - **inPoint** - between two coordinate systems: the one linked with 'axis' arc **inAxis** and the original one. Direction of this conversion is based on **inInverse** value:

- if it is true, **inPoint** is converted *to* original coordinate system
- if it is false, **inPoint** is converted *from* original coordinate system

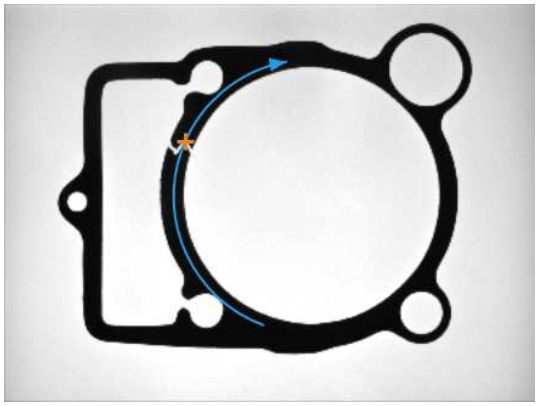
Coordinate system linked with **inAxis** is some transformation of the original one, satisfying these conditions:

- 'axis' arc is represented as a straight line
- 'axis' arc is horizontal if **inAxisType** is set to X or vertical if **inAxisType** is set to Y
- if **inAxisType** is set to X, the Y coordinate of 'axis' arc is equal to **inAxisCoordinate**. If **inAxisType** is set to Y, the X coordinate of 'axis' arc is equal to **inAxisCoordinate**

Hints

- If this filter is used to reverse **ImageAlongArc** transformation you should set **inAxis** and **inAxisType** to the same values as corresponding ones in **ImageAlongArc**. **inAxisCoordinate** should be set to a half of **ImageAlongArc.inScanWidth**.

Examples



ImageAlongArc *performed on the sample image with* ***inAxisType*** *= X and* ***inScanWidth*** *= 50. Point marked with green cross on the output image was calculated by* ***ScanSingleEdge****. Original point (marked with blue cross) was calculated using* ***PointAlongArc*** *with* ***inAxisType*** *= X,* ***inAxisCoordinate*** *= 25,* ***inInverse*** *= true and position of detected point.* ***ImageAlongArc*** *and* ***PointAlongArc*** *used the same arc for transformations.*

See Also

- ImageAlongArc – Creates an image from pixels traversed along an arc.
- PathAlongArc – Transforms a path to a coordinate system in which the 'axis' arc is vertical or horizontal.
- PointAlongPath – Transforms a point to a coordinate system in which the 'axis' path is vertical or horizontal.

PointAlongPath

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Transforms a point to a coordinate system in which the 'axis' path is vertical or horizontal.

Applications: Usually used to revert an ImageAlongPath transformation.

Syntax

```
void avl::PointAlongPath
(
 const avl::Point2D& inPoint,
 const avl::Path& inAxis,
 avl::Axis::Type inAxisType,
 float inAxisCoordinate,
 bool inInverse,
 avl::Point2D& outPoint
)
```

Parameters

	Name	Type	Default	Description
→	inPoint	const Point2D&		Input point
→	inAxis	const Path&		Input axis path
→	inAxisType	Axis::Type	Y	Type of axis the input axis path is parallel to
→	inAxisCoordinate	float	0.0f	Coordinate of the axis path
→	inInverse	bool	True	Switches to the inverse operation
←	outPoint	Point2D&		Transformed point

Description

This operation transforms a point - **inPoint** - between two coordinate systems: the one linked with 'axis' path **inAxis** and the original one. Direction of this conversion is based on **inInverse** value:

- if it is true, **inPoint** is converted *to* original coordinate system
- if it is false, **inPoint** is converted *from* original coordinate system

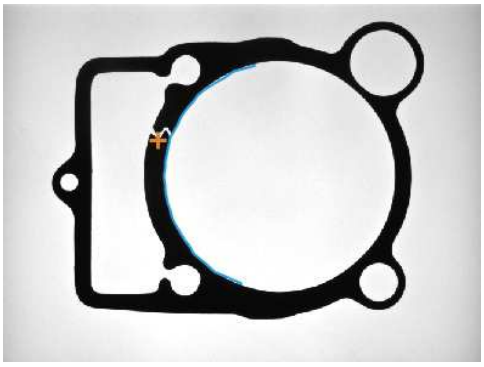
Coordinate system linked with **inAxis** is some transformation of the original one, satisfying these conditions:

- 'axis' path is represented as a straight line
- 'axis' path is horizontal if **inAxisType** is set to X or vertical if **inAxisType** is set to Y
- if **inAxisType** is set to X, the Y coordinate of 'axis' path is equal to **inAxisCoordinate**. If **inAxisType** is set to Y, the X coordinate of 'axis' path is equal to **inAxisCoordinate**

Hints

- If this filter is used to reverse **ImageAlongPath** transformation you should set **inAxis** and **inAxisType** to the same values as corresponding ones in **ImageAlongPath**. **inAxisCoordinate** should be set to a half of **ImageAlongPath.inScanWidth**.

Examples



ImageAlongPath *performed on the sample image with* ***inAxisType*** *= X,* ***inScanWidth*** *= 50 and path marked with blue. Point marked with blue cross on the output image was calculated by* ***ScanSingleEdge****. Original point (marked with orange cross) was calculated using* ***PointAlongPath*** *with* ***inAxisType*** *= X,* ***inAxisCoordinate*** *= 25,* ***inInverse*** *= true and position of detected point.* ***ImageAlongPath*** *and* ***PointAlongPath*** *used the same path for transformations.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty path on input in PointAlongPath.

See Also

- ImageAlongPath – Creates an image from pixels traversed along a path.
- PathAlongPath – Transforms a path to a coordinate system in which the 'axis' path is vertical or horizontal.
- PointAlongArc – Transforms a point to a coordinate system in which the 'axis' arc is vertical or horizontal.

RescaleArc

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Changes radius of an arc and translates its center in relation to a reference point.

Syntax

```
void avl::RescaleArc
(
 const avl::Arc2D& inArc,
 atl::Optional<const avl::Point2D&> inReferencePoint,
 float inScale,
 bool inInverse,
 avl::Arc2D& outArc
)
```

Parameters

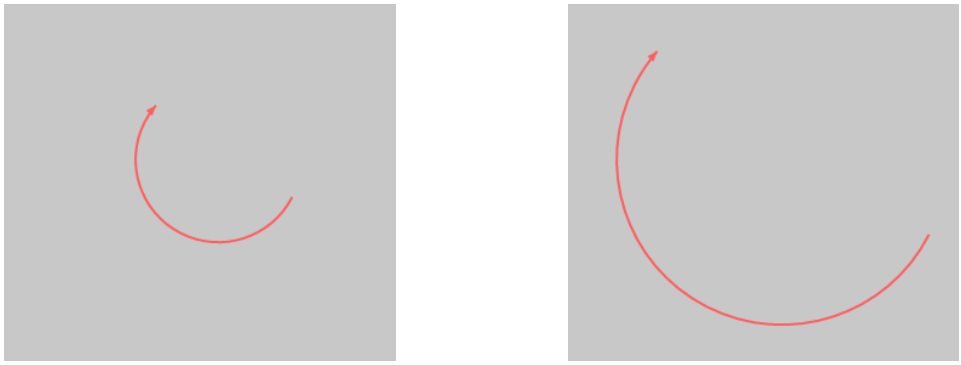
	Name	Type	Default	Description
→	inArc	const Arc2D&		
→	inReferencePoint	Optional<const Point2D&>	NIL	The point to which the distance of the arc center is changed (no change by default)
→	inScale	float	1.0f	Scaling factor
→	inInverse	bool		Switches to the inverse operation
←	outArc	Arc2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inArc** and **outArc**

Read more about In-place Computation.

Examples



RescaleArc *performed on the sample arc,* ***inReferencePoint*** *= auto,* ***inScale*** *= 2.0 and* ***inInverse*** *= false.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Scale cannot be zero in an inverse scaling of an arc in RescaleArc.

RescaleCircle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Changes radius of a circle and translates its center in relation to a reference point.

Syntax

```
void avl::RescaleCircle
(
 const avl::Circle2D& inCircle,
 atl::Optional<const avl::Point2D&> inReferencePoint,
 float inScale,
 bool inInverse,
 avl::Circle2D& outCircle
)
```

Parameters

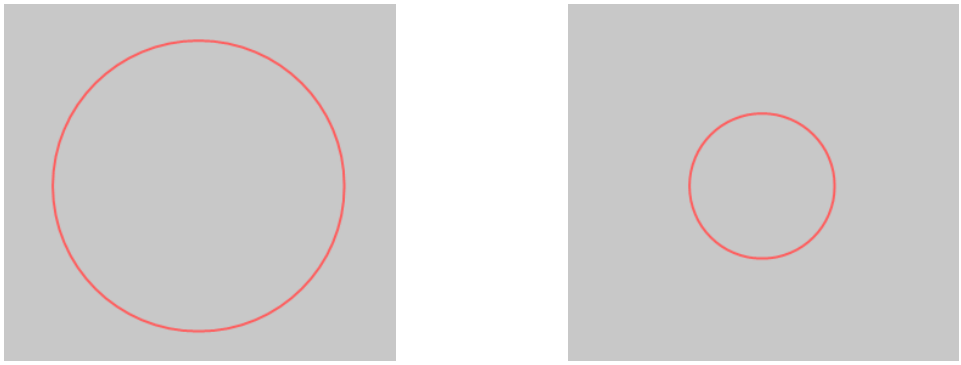
	Name	Type	Default	Description
→	inCircle	const Circle2D&		
→	inReferencePoint	Optional<const Point2D&>	NIL	The point to which the distance of the circle center is changed (no change by default)
→	inScale	float	1.0f	Scaling factor
→	inInverse	bool		Switches to the inverse operation
←	outCircle	Circle2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inCircle** and **outCircle**

Read more about In-place Computation.

Examples



RescaleCircle *performed on the sample circle,* ***inReferencePoint*** *= auto,* ***inScale*** *= 0.5 and* ***inInverse*** *= false.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Scale cannot be zero in an inverse rescaling of a circle in RescaleCircle.

RescaleLine

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Changes the distance of a line to a reference point.

Syntax

```
void avl::RescaleLine
(
 const avl::Line2D& inLine,
 const avl::Point2D& inReferencePoint,
 float inScale,
 bool inInverse,
 avl::Line2D& outLine
)
```

Parameters

	Name	Type	Default	Description
→	inLine	const Line2D&		
→	inReferencePoint	const Point2D&		The point to which all distances change linearly
→	inScale	float	1.0f	Scaling factor
→	inInverse	bool		Switches to the inverse operation
←	outLine	Line2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inLine** and **outLine**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite line on input in RescaleLine.
DomainError	Scale cannot be zero in an inverse rescaling of a line in RescaleLine.

RescalePoint

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Changes the distance of a point to a reference point.

Syntax

```
void avl::RescalePoint
(
 const avl::Point2D& inPoint,
 const avl::Point2D& inReferencePoint,
 float inScale,
 bool inInverse,
 avl::Point2D& outPoint
)
```

Parameters

	Name	Type	Default	Description
→	inPoint	const Point2D&		
→	inReferencePoint	const Point2D&		Point to which the distance will be changed
→	inScale	float	1.0f	Scaling factor
→	inInverse	bool		Switches to the inverse operation
←	outPoint	Point2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inPoint** and **outPoint**

Read more about In-place Computation.

Examples

RescalePoint *performed on the sample point,* ***inReferencePoint*** *= (150.0, 150.0),* ***inScale*** *= -1.5 and* ***inInverse*** *= false. The* ***inReferencePoint*** *point is drawn on the first image in blue.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Scale cannot be zero in an inverse rescaling of a point in RescalePoint.

RescalePointArray

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Changes the distances of points from an array to a reference point.

Syntax

```
void avl::RescalePointArray
(
 const atl::Array<avl::Point2D>& inPoints,
 atl::Optional<const avl::Point2D&> inReferencePoint,
 float inScale,
 bool inInverse,
 atl::Array<avl::Point2D>& outPoints
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		
→	inReferencePoint	Optional<const Point2D&>	NIL	Point to which the distances will be changed (the mass center by default)
→	inScale	float	1.0f	Scaling factor
→	inInverse	bool		Switches to the inverse operation
←	outPoints	Array<Point2D>&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inPoints** and **outPoints**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Scale cannot be zero in an inverse rescaling of a point in RescalePointArray.

RescaleRectangle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Changes the corners and the dimensions of a rectangle.

Syntax

```
void avl::RescaleRectangle
(
 const avl::Rectangle2D& inRectangle,
 atl::Optional<const avl::Point2D&> inReferencePoint,
 float inScale,
 bool inInverse,
 avl::Rectangle2D& outRectangle
)
```

Parameters

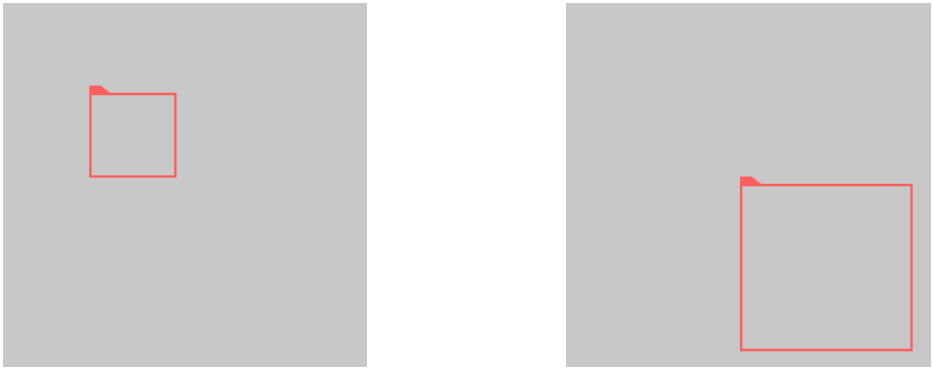
	Name	Type	Default	Description
→	inRectangle	const Rectangle2D&		
→	inReferencePoint	Optional<const Point2D&>	NIL	The point, to which all distance will be changed proportionally
→	inScale	float	1.0f	Scaling factor
→	inInverse	bool		Switches to the inverse operation
←	outRectangle	Rectangle2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inRectangle** and **outRectangle**

Read more about In-place Computation.

Examples



RescaleRectangle *performed on the sample rectangle,* ***inReferencePoint*** *= (0.0, 0.0),* ***inScale*** *= 2.0 and* ***inInverse*** *= false.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Scale cannot be zero in inverse scaling of rectangle in RescaleRectangle.

RescaleSegment

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Lengthens or shortens a segment relatively.

Syntax

```
void avl::RescaleSegment
(
 const avl::Segment2D& inSegment,
 atl::Optional<const avl::Point2D&> inReferencePoint,
 float inScale,
 bool inInverse,
 avl::Segment2D& outSegment
)
```

Parameters

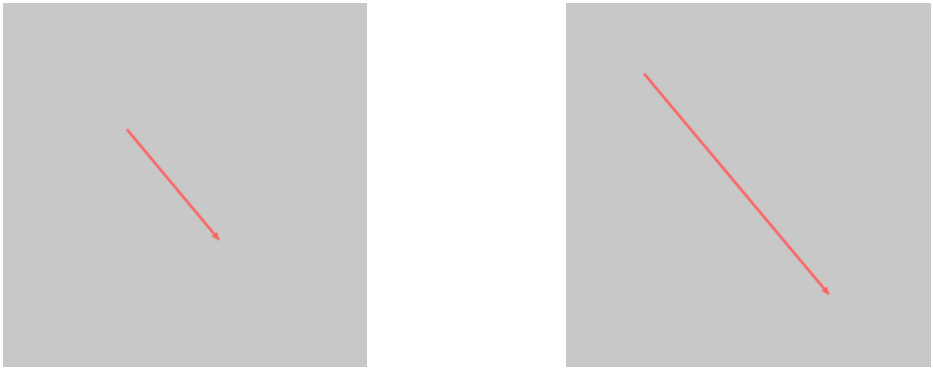
	Name	Type	Default	Description
→	inSegment	const Segment2D&		
→	inReferencePoint	Optional<const Point2D&>	NIL	The point to which all distances change linearly (the mass center by default)
→	inScale	float	1.0f	Scaling factor (negative values invert the segment)
→	inInverse	bool		Switches to the inverse operation
←	outSegment	Segment2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inSegment** and **outSegment**

Read more about In-place Computation.

Examples



RescaleSegment *performed on the sample segment,* ***inReferencePoint*** *= auto,* ***inScale*** *= 2.0 and* ***inInverse*** *= false.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Scale cannot be zero in an inverse rescaling of a segment in RescaleSegment.

RescaleVector

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Lengthens or shortens a vector relatively preserving its direction.

Syntax

```
void avl::RescaleVector
(
 const avl::Vector2D& inVector,
 float inScale,
 bool inInverse,
 avl::Vector2D& outVector
)
```

Parameters

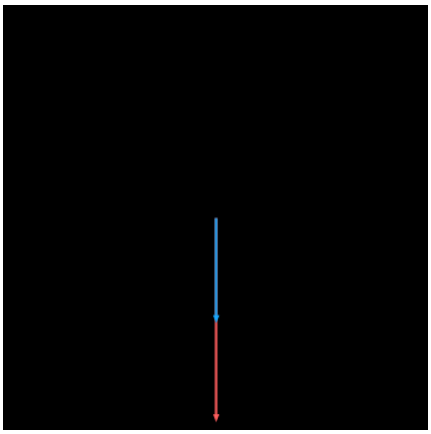
	Name	Type	Default	Description
→	inVector	const Vector2D&		
→	inScale	float	1.0f	Scaling factor
→	inInverse	bool		Switches to the inverse operation
←	outVector	Vector2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inVector** and **outVector**

Read more about In-place Computation.

Examples



RescaleVector *performed on a vector (red one) with* ***DeltaX*** *= 0,* ***DeltaY*** *= 10 with parameters* ***inScale*** *= 5,* ***inInverse*** *= True.* ***outVector*** *returns vector (blue one)*

Errors

List of possible exceptions:

Error type	Description
DomainError	Scale cannot be zero in an inverse scaling of vector in RescaleVector.

ResizeArc

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Changes radius of an arc.

Syntax

```
void avl::ResizeArc
(
 const avl::Arc2D& inArc,
 const float inNewSize,
 avl::Arc2D& outArc
)
```

Parameters

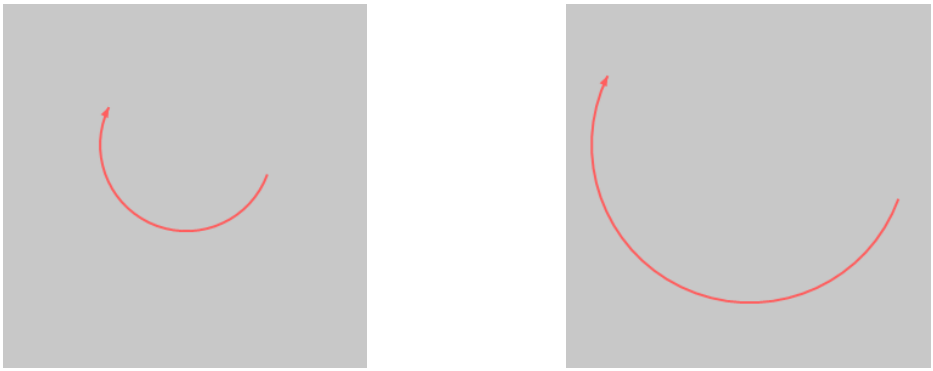
	Name	Type	Range	Default	Description
→	inArc	const Arc2D&			
→	inNewSize	const float	0.0 - ∞	1.0f	
←	outArc	Arc2D&			

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inArc** and **outArc**

Read more about In-place Computation.

Examples



ResizeArc *performed on the sample arc,* ***inNewSize*** *= 130.*

ResizeArc_Delta

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Changes radius of an arc by adding a value.

Syntax

```
void avl::ResizeArc_Delta
(
 const avl::Arc2D& inArc,
 const float inDelta,
 avl::Arc2D& outArc
)
```

Parameters

	Name	Type	Default	Description
→	inArc	const Arc2D&		
→	inDelta	const float	0.0f	Value added to arc radius
←	outArc	Arc2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inArc** and **outArc**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Negative arc radius in ResizeArc_Delta.

ResizeCircle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Changes radius of a circle.

Syntax

```
void avl::ResizeCircle
(
 const avl::Circle2D& inCircle,
 float inNewSize,
 avl::Circle2D& outCircle
)
```

Parameters

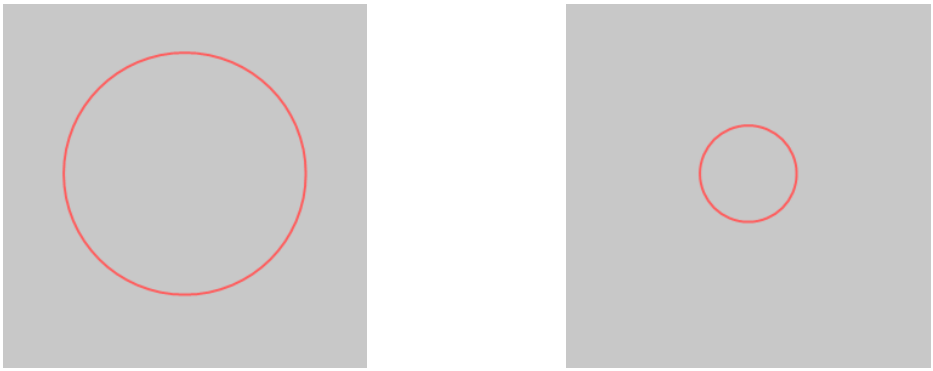
	Name	Type	Range	Default	Description
→	inCircle	const Circle2D&			
→	inNewSize	float	0.0 - ∞	1.0f	New value for radius
←	outCircle	Circle2D&			

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inCircle** and **outCircle**

Read more about In-place Computation.

Examples



ResizeCircle *performed on the sample circle,* ***inNewSize*** *= 40.*

ResizeCircle_Delta

Header: AVL.h
Namespace: avl
Module: FoundationLite

Changes radius of a circle by adding a value.

Syntax

```
void avl::ResizeCircle_Delta
(
 const avl::Circle2D& inCircle,
 float inDelta,
 avl::Circle2D& outCircle
)
```

Parameters

	Name	Type	Default	Description
→	inCircle	const Circle2D&		
→	inDelta	float	0.0f	Value added to circle radius
←	outCircle	Circle2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inCircle** and **outCircle**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Negative circle radius in ResizeCircle_Delta.

ResizeRectangle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Changes dimensions of a rectangle.

Syntax

```
void avl::ResizeRectangle
(
 const avl::Rectangle2D& inRectangle,
 avl::Anchor2D::Type inAnchor,
 atl::Optional<float> inNewWidth,
 atl::Optional<float> inNewHeight,
 avl::Rectangle2D& outRectangle
)
```

Parameters

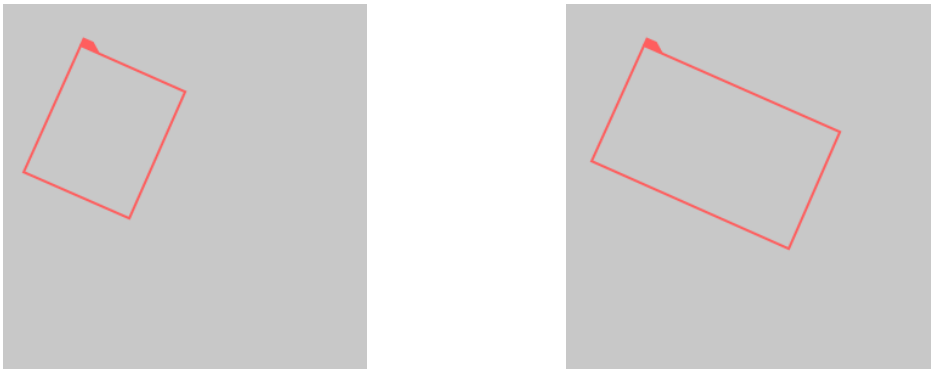
	Name	Type	Range	Default	Description
→	inRectangle	const Rectangle2D&			
→	inAnchor	Anchor2D::Type		TopLeft	
→	inNewWidth	Optional<float>	0.0 - ∞	NIL	
→	inNewHeight	Optional<float>	0.0 - ∞	NIL	
←	outRectangle	Rectangle2D&			

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inRectangle** and **outRectangle**

Read more about In-place Computation.

Examples



***ResizeRectangle** performed on the sample rectangle, **inNewWidth** = 177.0 and **inNewHeight** = 104.0.*

ResizeRectangle_Delta

Header: AVL.h
Namespace: avl
Module: FoundationLite

Changes dimensions of a rectangle by adding some values.

Syntax

```
void avl::ResizeRectangle_Delta
(
 const avl::Rectangle2D& inRectangle,
 avl::Anchor2D::Type inAnchor,
 float inWidthDelta,
 float inHeightDelta,
 avl::Rectangle2D& outRectangle
)
```

Parameters

	Name	Type	Default	Description
→	inRectangle	const Rectangle2D&		
→	inAnchor	Anchor2D::Type	TopLeft	
→	inWidthDelta	float	0.0f	Value added to width of the rectangle
→	inHeightDelta	float	0.0f	Value added to height of the rectangle
←	outRectangle	Rectangle2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inRectangle** and **outRectangle**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Negative rectangle dimensions in ResizeRectangle_Delta.

ResizeRectangle_Relative

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Changes dimensions of a rectangle relatively.

Syntax

```
void avl::ResizeRectangle_Relative
(
 const avl::Rectangle2D& inRectangle,
 avl::Anchor2D::Type inAnchor,
 float inWidthScale,
 float inHeightScale,
 avl::Rectangle2D& outRectangle
)
```

Parameters

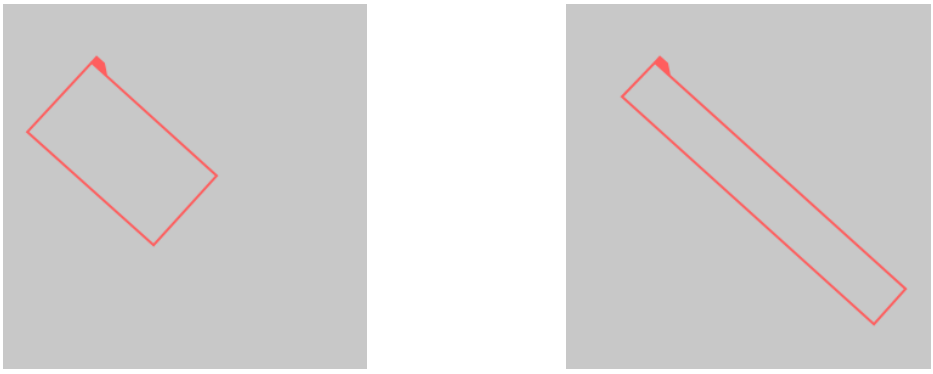
	Name	Type	Range	Default	Description
→	inRectangle	const Rectangle2D&			
→	inAnchor	Anchor2D::Type		TopLeft	
→	inWidthScale	float	0.0 - ∞	1.0f	
→	inHeightScale	float	0.0 - ∞	1.0f	
←	outRectangle	Rectangle2D&			

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inRectangle** and **outRectangle**

Read more about In-place Computation.

Examples



ResizeRectangle_Relative *performed on the sample rectangle,* ***inWidthScale*** *= 2.0 and* ***inHeightScale*** *= 0.5.*

ResizeSegment

Header: AVL.h
Namespace: avl
Module: FoundationLite

Lengthens or shortens a segment to a new length preserving its orientation and center point.

Syntax

```
void avl::ResizeSegment
(
 const avl::Segment2D& inSegment,
 float inNewLength,
 float inAnchor,
 avl::Segment2D& outSegment
)
```

Parameters

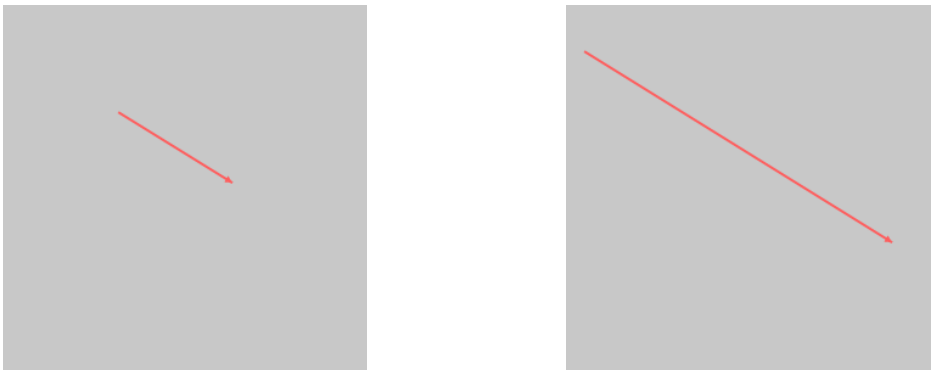
	Name	Type	Range	Default	Description
→	inSegment	const Segment2D&			
→	inNewLength	float	0.0 - ∞	1.0f	
→	inAnchor	float	-∞ - ∞	0.5f	
←	outSegment	Segment2D&			

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inSegment** and **outSegment**

Read more about In-place Computation.

Examples



ResizeSegment *performed on the sample segment,* ***inNewLength*** *= 298.0.*

ResizeSegment_Delta

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Changes length of a segment by adding a value preserving its orientation and center point.

Syntax

```
void avl::ResizeSegment_Delta
(
 const avl::Segment2D& inSegment,
 float inDelta,
 float inAnchor,
 avl::Segment2D& outSegment
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSegment	const Segment2D&			
→	inDelta	float		0.0f	Value added to segment length
→	inAnchor	float	-∞ -∞	0.5f	
←	outSegment	Segment2D&			

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inSegment** and **outSegment**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Negative segment length in ResizeSegment_Delta.

ResizeVector

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Lengthens or shortens a vector preserving its direction.

Syntax

```
void avl::ResizeVector
(
 const avl::Vector2D& inVector,
 float inNewLength,
 avl::Vector2D& outVector
)
```

Parameters

	Name	Type	Range	Default	Description
→	inVector	const Vector2D&			
→	inNewLength	float	0.0 - ∞	1.0f	
←	outVector	Vector2D&			

In-place Processing

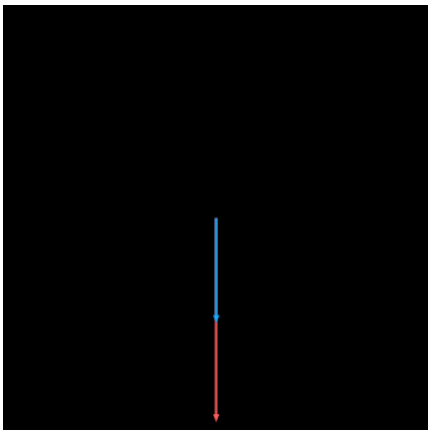
This function supports in-place data processing - you can pass the same reference to **inVector** and **outVector**

Read more about In-place Computation.

Description

Note that because of inaccuracies of floating-point arithmetic, some geometric operations (including this one) may lead to unpredictable results for degenerated cases. In this filter such a case occurs when a zero vector is given on input.

Examples



***ResizeVector** performed on a vector (red one) with **DeltaX** = 0, **DeltaY** = 10 with parameter **inNewLength** = 5. **outVector** returns vector (blue one)*

ResizeVector_Delta

Header: AVL.h
Namespace: avl
Module: FoundationLite

Extends length of a vector by adding a value preserving its direction.

Syntax

```
void avl::ResizeVector_Delta
(
 const avl::Vector2D& inVector,
 float inDelta,
 avl::Vector2D& outVector
)
```

Parameters

	Name	Type	Default	Description
→	inVector	const Vector2D&		
→	inDelta	float	0.0f	Value added to vector length
←	outVector	Vector2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inVector** and **outVector**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Negative vector length in ResizeVector_Delta.

ReverseArc

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Inverts the direction of the arc.

Syntax

```
void avl::ReverseArc
(
 const avl::Arc2D& inArc,
 avl::Arc2D& outArc
)
```

Parameters

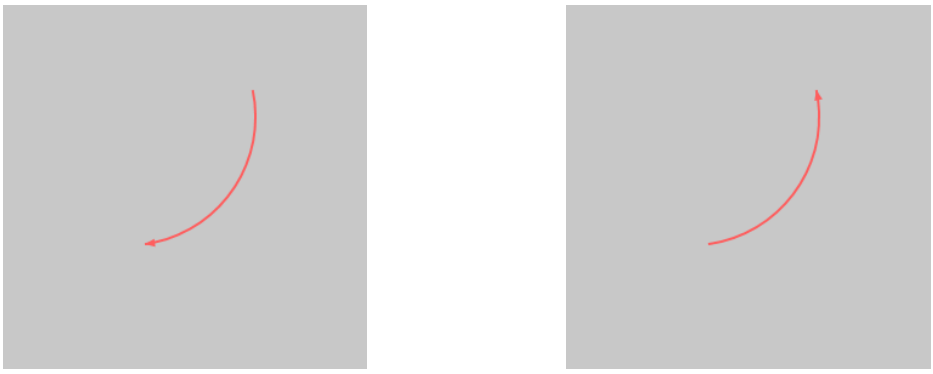
	Name	Type	Default	Description
→	inArc	const Arc2D&		
←	outArc	Arc2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inArc** and **outArc**

Read more about In-place Computation.

Examples



***ReverseArc** performed on the sample arc.*

ReverseSegment

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Swaps the two endpoints of a segment.

Syntax

```
void avl::ReverseSegment
(
 const avl::Segment2D& inSegment,
 avl::Segment2D& outSegment
)
```

Parameters

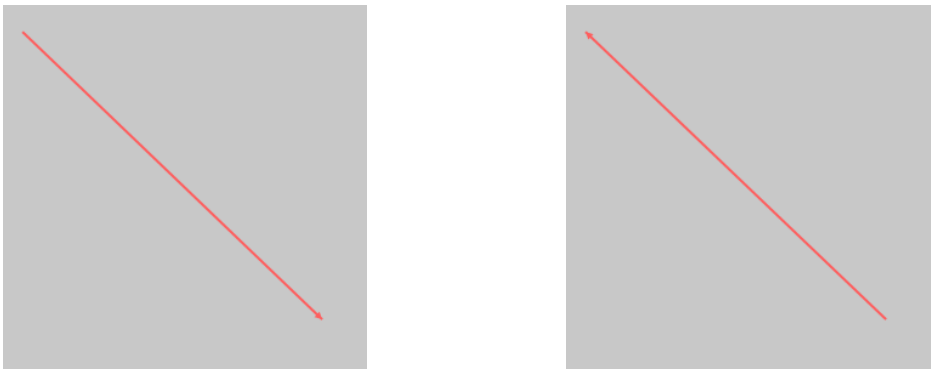
	Name	Type	Default	Description
→	inSegment	const Segment2D&		
←	outSegment	Segment2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inSegment** and **outSegment**

Read more about In-place Computation.

Examples



ReverseSegment *performed on the sample segment.*

RotateAngle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Adds two angles.

Syntax

```
void avl::RotateAngle
(
 float inAngle,
 float inRotationAngle,
 avl::AngleRange::Type inAngleRange,
 bool inInverse,
 float& outAngle
)
```

Parameters

	Name	Type	Default	Description
→	inAngle	float		
→	inRotationAngle	float		
→	inAngleRange	AngleRange::Type	_0_180	
→	inInverse	bool		
←	outAngle	float&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inAngle** and **outAngle**

Read more about In-place Computation.

RotateAngle_Toward

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Changes the input direction in the direction of minimum rotation toward the target direction.

Syntax

```
void avl::RotateAngle_Toward
(
 float inAngle,
 float inTargetAngle,
 avl::AngleRange::Type inAngleRange,
 float inRotationAngle,
 bool inInverse,
 float& outAngle
)
```

Parameters

	Name	Type	Default	Description
→	inAngle	float		
→	inTargetAngle	float		
→	inAngleRange	AngleRange::Type	_0_180	
→	inRotationAngle	float		
→	inInverse	bool		
←	outAngle	float&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inAngle** and **outAngle**

Read more about In-place Computation.

RotateArc

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Rotates an arc clockwise around center point.

Syntax

```
void avl::RotateArc
(
 const avl::Arc2D& inArc,
 atl::Optional<const avl::Point2D&> inCenter,
 float inAngle,
 bool inInverse,
 avl::Arc2D& outArc
)
```

Parameters

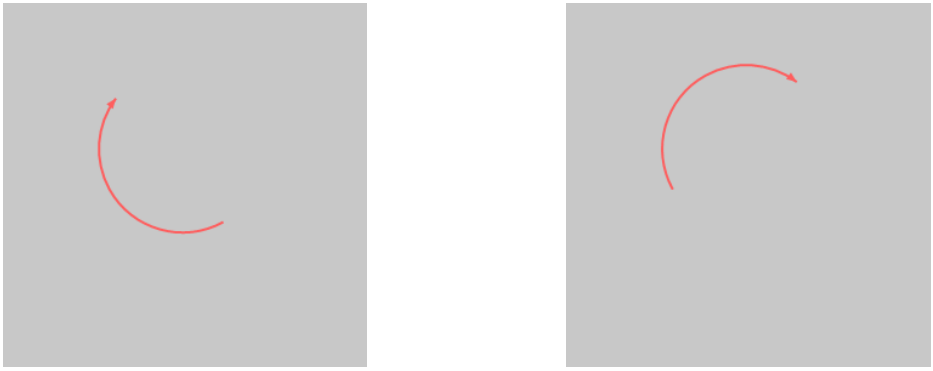
	Name	Type	Default	Description
→	inArc	const Arc2D&		
→	inCenter	Optional<const Point2D&>	NIL	Center of rotation (the arc's center by default)
→	inAngle	float		Clockwise rotation angle
→	inInverse	bool		Switches to the inverse operation
←	outArc	Arc2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inArc** and **outArc**

Read more about In-place Computation.

Examples



RotateArc *performed on the sample arc,* ***inCenter*** *= Auto,* ***inAngle*** *= 90 and* ***inInverse*** *= False.*

RotateCircle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Rotates a circle clockwise around a center point.

Syntax

```
void avl::RotateCircle
(
 const avl::Circle2D& inCircle,
 const avl::Point2D& inCenter,
 float inAngle,
 bool inInverse,
 avl::Circle2D& outCircle
)
```

Parameters

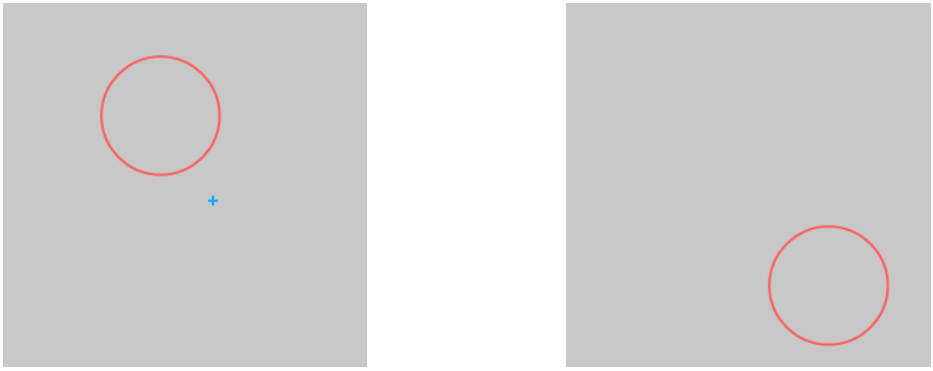
	Name	Type	Default	Description
→	inCircle	const Circle2D&		
→	inCenter	const Point2D&		Center of rotation
→	inAngle	float		Clockwise rotation angle
→	inInverse	bool		Switches to the inverse operation
←	outCircle	Circle2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inCircle** and **outCircle**

Read more about In-place Computation.

Examples



RotateCircle *performed on the sample circle,* ***inCenter*** *= (173.0, 163.0),* ***inAngle*** *= 180.0 and* ***inInverse*** *= False. The* ***inCenter*** *point is drawn on the first image in blue*

RotateCoordinateSystem

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Rotates a coordinate system around a center point.

Syntax

```
void avl::RotateCoordinateSystem
(
 const avl::CoordinateSystem2D& inCoordinateSystem,
 atl::Optional<const avl::Point2D&> inCenter,
 float inAngle,
 bool inInverse,
 avl::CoordinateSystem2D& outCoordinateSystem
)
```

Parameters

	Name	Type	Default	Description
→	inCoordinateSystem	const CoordinateSystem2D&		
→	inCenter	Optional<const Point2D&>	NIL	Center of rotation (the coordinate system's center by default)
→	inAngle	float		Clockwise rotation angle
→	inInverse	bool		Switches to the inverse operation
←	outCoordinateSystem	CoordinateSystem2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inCoordinateSystem** and **outCoordinateSystem**

Read more about In-place Computation.

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

RotateLine

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Rotates a line clockwise around a center point.

Syntax

```
void avl::RotateLine
(
 const avl::Line2D& inLine,
 const avl::Point2D& inCenter,
 float inAngle,
 bool inInverse,
 avl::Line2D& outLine
)
```

Parameters

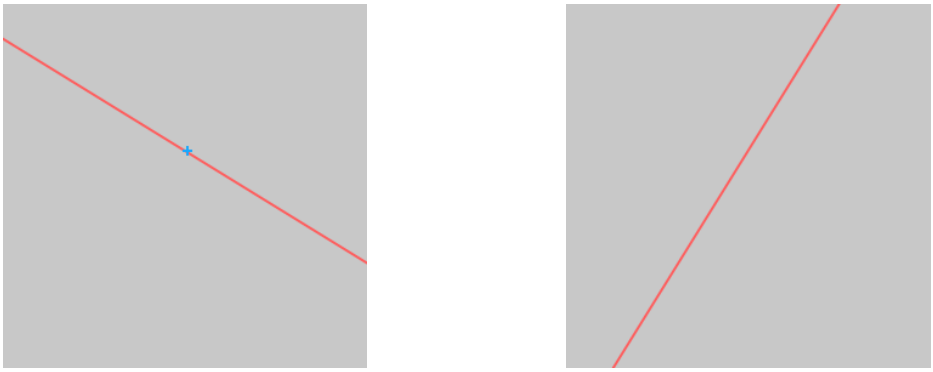
	Name	Type	Default	Description
→	inLine	const Line2D&		
→	inCenter	const Point2D&		Center of rotation
→	inAngle	float		Clockwise angle of rotation
→	inInverse	bool		Switches to the inverse operation
←	outLine	Line2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inLine** and **outLine**

Read more about In-place Computation.

Examples



RotateLine *performed on the sample line,* ***inCenter*** *= (152.0, 121.0),* ***inAngle*** *= 90.0 and* ***inInverse*** *= False. The* ***inCenter*** *point is drawn on the first image in blue.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite line on input in RotateLine.

RotatePoint

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Rotates a point clockwise around a center point.

Syntax

```
void avl::RotatePoint
(
 const avl::Point2D& inPoint,
 const avl::Point2D& inCenter,
 float inAngle,
 bool inInverse,
 avl::Point2D& outPoint
)
```

Parameters

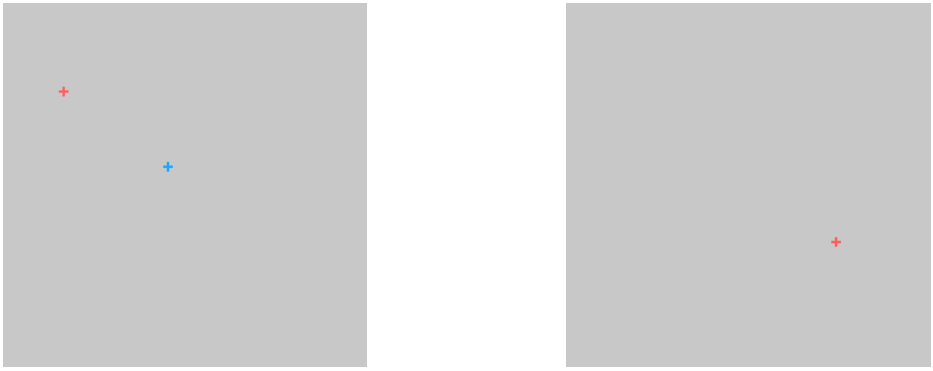
	Name	Type	Default	Description
→	inPoint	const Point2D&		
→	inCenter	const Point2D&		Center of rotation
→	inAngle	float		Clockwise rotation angle
→	inInverse	bool		Switches to the inverse operation
←	outPoint	Point2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inPoint** and **outPoint**, **inCenter** and **outPoint**

Read more about In-place Computation.

Examples



RotatePoint *performed on the sample point,* ***inCenter*** *= (136.0, 135.0),* ***inAngle*** *= 180.0 and* ***inInverse*** *= False. The* ***inCenter*** *point is drawn on the first image in blue.*

RotatePointArray

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Rotates an array of points clockwise around a center point.

Syntax

```
void avl::RotatePointArray
(
 const atl::Array<avl::Point2D>& inPoints,
 atl::Optional<const avl::Point2D&> inCenter,
 float inAngle,
 bool inInverse,
 atl::Array<avl::Point2D>& outPoints
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		
→	inCenter	Optional<const Point2D&>	NIL	Center of rotation (the mass center by default)
→	inAngle	float		Clockwise rotation angle
→	inInverse	bool		Switches to the inverse operation
←	outPoints	Array<Point2D>&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inPoints** and **outPoints**

Read more about In-place Computation.

RotateRectangle

Header: AVL.h
Namespace: avl
Module: FoundationLite

Rotates a rectangle clockwise around a center point.

Syntax

```
void avl::RotateRectangle
(
 const avl::Rectangle2D& inRectangle,
 atl::Optional<const avl::Point2D&> inCenter,
 float inAngle,
 bool inInverse,
 avl::Rectangle2D& outRectangle
)
```

Parameters

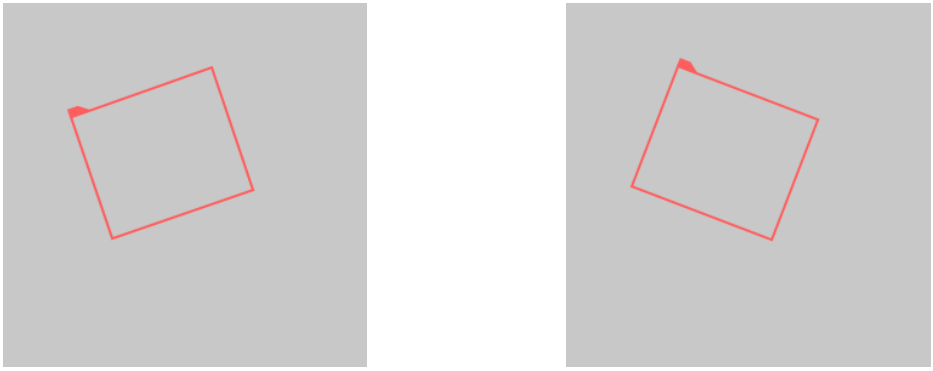
	Name	Type	Default	Description
→	inRectangle	const Rectangle2D&		
→	inCenter	Optional<const Point2D&>	NIL	Center of rotation (the rectangle's point by default)
→	inAngle	float		Clockwise rotation angle
→	inInverse	bool		Switches to the inverse operation
←	outRectangle	Rectangle2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inRectangle** and **outRectangle**

Read more about In-place Computation.

Examples



RotateRectangle *performed on the sample rectangle,* ***inCenter*** *= Auto,* ***inAngle*** *= 40.0 and* ***inInverse*** *= False.*

RotateSegment

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Rotates a segment clockwise around a center point.

Syntax

```
void avl::RotateSegment
(
 const avl::Segment2D& inSegment,
 atl::Optional<const avl::Point2D&> inCenter,
 float inAngle,
 bool inInverse,
 avl::Segment2D& outSegment
)
```

Parameters

	Name	Type	Default	Description
→	inSegment	const Segment2D&		
→	inCenter	Optional<const Point2D&>	NIL	Center of rotation (by default the mass center)
→	inAngle	float		Clockwise angle of rotation
→	inInverse	bool		Switches to the inverse operation
←	outSegment	Segment2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inSegment** and **outSegment**

Read more about In-place Computation.

Examples

RotateSegment *performed on the sample segment,* ***inCenter*** *= Auto,* ***inAngle*** *= 70.0 and* ***inInverse*** *= False.*

RotateVector

Header: AVL.h
Namespace: avl
Module: FoundationLite

Rotates a vector clockwise.

Syntax

```
void avl::RotateVector
(
 const avl::Vector2D& inVector,
 float inAngle,
 bool inInverse,
 avl::Vector2D& outVector
)
```

Parameters

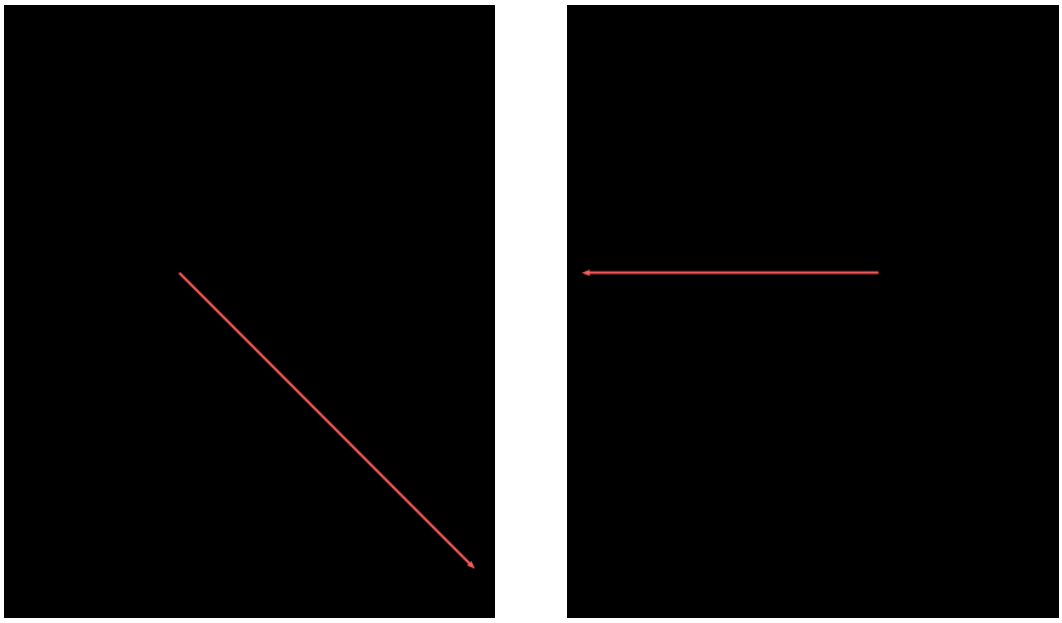
	Name	Type	Default	Description
→	inVector	const Vector2D&		
→	inAngle	float		Clockwise angle of rotation
→	inInverse	bool		Switches to the inverse operation
←	outVector	Vector2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inVector** and **outVector**

Read more about In-place Computation.

Examples



RotateVector *performed on the sample vector,* ***inAngle*** *= 135.0 and* ***inInverse*** *= False.*

SplitPointsByLine

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Separates the points being on one side of the input line from the others.

Syntax

```
void avl::SplitPointsByLine
(
 const atl::Array<avl::Point2D>& inPoints,
 const avl::Line2D& inLine,
 atl::Array<avl::Point2D>& outPoints1,
 atl::Array<avl::Point2D>& outPoints2
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		Input points
→	inLine	const Line2D&		Line used for splitting
←	outPoints1	Array<Point2D>&		Points with positive signed distance to the input line
←	outPoints2	Array<Point2D>&		Points with negative signed distance to the input line

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite line on input in SplitPointsByLine.

SplitSegment

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Splits a segment into several parts of equal length.

Syntax

```
void avl::SplitSegment
(
 const avl::Segment2D& inSegment,
 const int inCount,
 atl::Array<avl::Segment2D>& outSegments
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSegment	const Segment2D&			
→	inCount	const int	1 - ∞	2	Number of segments after segment split
←	outSegments	Array<Segment2D>&			

Description

Filter computes **outSegments** using following formulas:

$$\mathbf{outSegments}[0].x_{begin} = \mathbf{inSegment}.x_{begin}$$

$$\mathbf{outSegments}[0].y_{begin} = \mathbf{inSegment}.y_{begin}$$

$$\mathbf{outSegments}[0].x_{end} = \mathbf{inSegment}.x_{begin} + \Delta x$$

$$\mathbf{outSegments}[0].y_{end} = \mathbf{inSegment}.y_{begin} + \Delta y$$

$$\mathbf{outSegments}[i].x_{begin} = \mathbf{outSegments}[i-1].x_{begin} + \Delta x$$

$$\mathbf{outSegments}[i].y_{begin} = \mathbf{outSegments}[i-1].y_{begin} + \Delta y$$

$$\mathbf{outSegments}[i].x_{end} = \mathbf{outSegments}[i-1].x_{end} + \Delta x$$

$$\mathbf{outSegments}[i].y_{end} = \mathbf{outSegments}[i-1].y_{end} + \Delta y$$

$$i = 1, 2, .., \mathbf{inCount} - 1$$

where

$$\Delta x = \frac{\mathbf{inSegment}.x_{end} - \mathbf{inSegment}.x_{begin}}{\mathbf{inCount}}$$

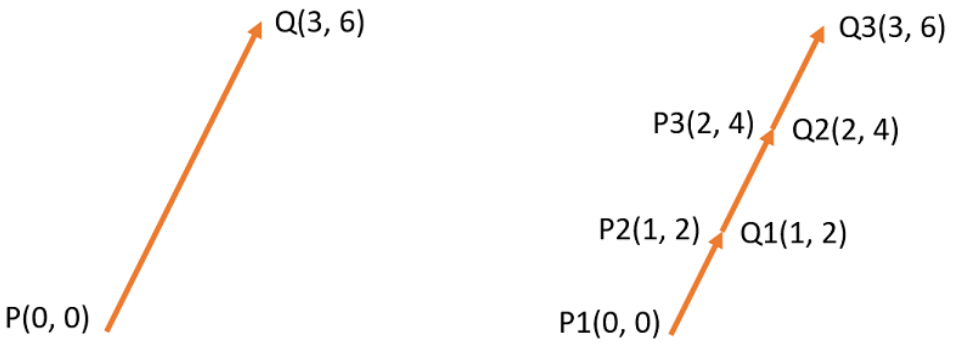
$$\Delta y = \frac{\mathbf{inSegment}.y_{end} - \mathbf{inSegment}.y_{begin}}{\mathbf{inCount}}$$

Examples

Assume segment described with two points:

- Start point P(0, 0)
- End point Q(3, 6).

To split segment into three parts of equal length, set **inCount** = 3.
Result will be as follows:



$$\mathbf{outSegments}[0] = \vec{P_1Q_1} = [1, 2] \quad \mathbf{outSegments}[1] = \vec{P_2Q_2} = [1, 2] \quad \mathbf{outSegments}[2] = \vec{P_3Q_3} = [1, 2]$$

TranslateArc

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Translates an arc by a vector.

Syntax

```
void avl::TranslateArc
(
 const avl::Arc2D& inArc,
 const avl::Vector2D& inDelta,
 bool inInverse,
 avl::Arc2D& outArc
)
```

Parameters

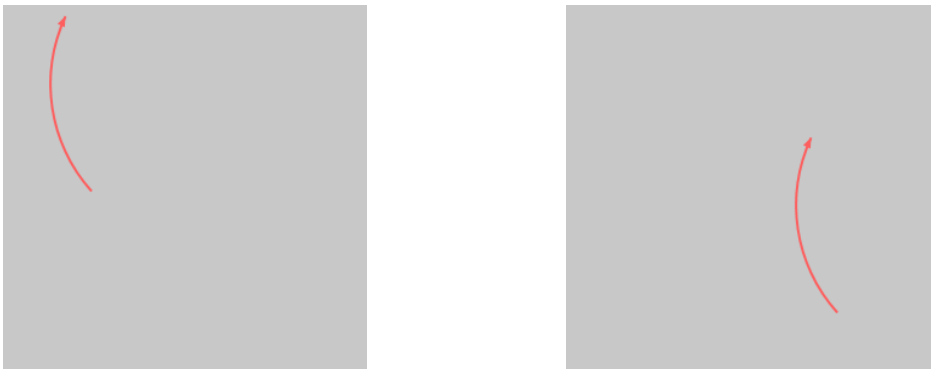
	Name	Type	Default	Description
→	inArc	const Arc2D&		
→	inDelta	const Vector2D&		Translation vector
→	inInverse	bool		Switches to the inverse operation
←	outArc	Arc2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inArc** and **outArc**

Read more about In-place Computation.

Examples



TranslateArc *performed on the sample arc,* ***inDelta*** *= (150.0, 100.0) and* ***inInverse*** *= False.*

TranslateCircle

Header: AVL.h
Namespace: avl
Module: FoundationLite

Translates a circle by a vector.

Syntax

```
void avl::TranslateCircle
(
 const avl::Circle2D& inCircle,
 const avl::Vector2D& inDelta,
 bool inInverse,
 avl::Circle2D& outCircle
)
```

Parameters

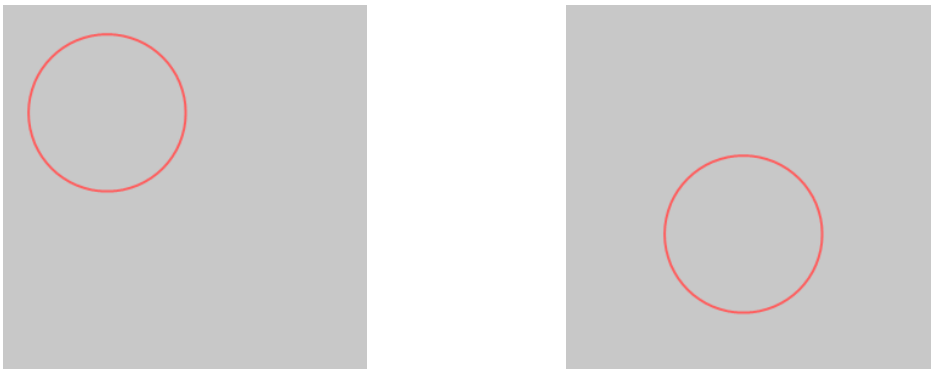
	Name	Type	Default	Description
→	inCircle	const Circle2D&		
→	inDelta	const Vector2D&		Vector of translation
→	inInverse	bool		Switches to the inverse operation
←	outCircle	Circle2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inCircle** and **outCircle**

Read more about In-place Computation.

Examples



TranslateCircle *performed on the sample circle,* ***inDelta*** *= (60.0, 100.0) and* ***inInverse*** *= False.*

TranslateCoordinateSystem

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Translates a coordinate system by a vector.

Syntax

```
void avl::TranslateCoordinateSystem
(
 const avl::CoordinateSystem2D& inCoordinateSystem,
 const avl::Vector2D& inDelta,
 bool inInverse,
 avl::CoordinateSystem2D& outCoordinateSystem
)
```

Parameters

	Name	Type	Default	Description
→	inCoordinateSystem	const CoordinateSystem2D&		
→	inDelta	const Vector2D&		Translation vector
→	inInverse	bool		Switches to the inverse operation
←	outCoordinateSystem	CoordinateSystem2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inCoordinateSystem** and **outCoordinateSystem**

Read more about In-place Computation.

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

TranslateLine

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Translates a line by a vector.

Syntax

```
void avl::TranslateLine
(
 const avl::Line2D& inLine,
 const avl::Vector2D& inDelta,
 bool inInverse,
 const avl::TranslateAlignment::Type inDeltaAlignment,
 avl::Line2D& outLine
)
```

Parameters

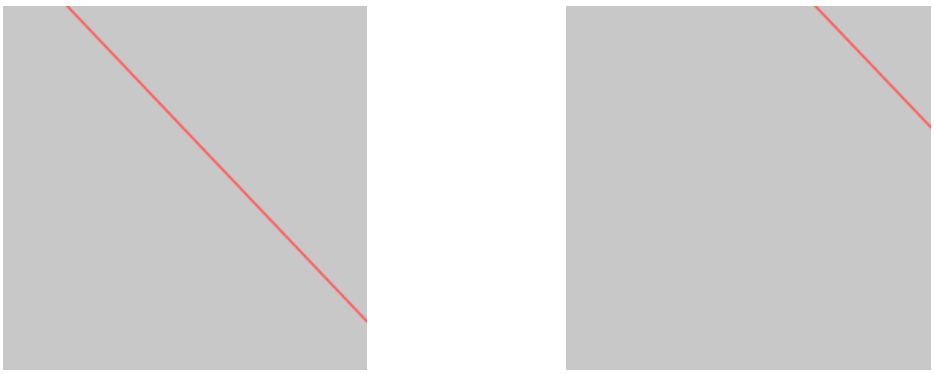
	Name	Type	Default	Description
→	inLine	const Line2D&		
→	inDelta	const Vector2D&		Vector of translation
→	inInverse	bool		Switches to the inverse operation
→	inDeltaAlignment	const TranslateAlignment::Type		
←	outLine	Line2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inLine** and **outLine**

Read more about In-place Computation.

Examples



TranslateLine *performed on the sample line,* ***inDelta*** *= (200.0, 50.0) and* ***inInverse*** *= False.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite line on input in TranslateLine.

TranslatePoint

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Translates a point by a vector.

Syntax

```
void avl::TranslatePoint
(
 const avl::Point2D& inPoint,
 const avl::Vector2D& inDelta,
 bool inInverse,
 avl::Point2D& outPoint
)
```

Parameters

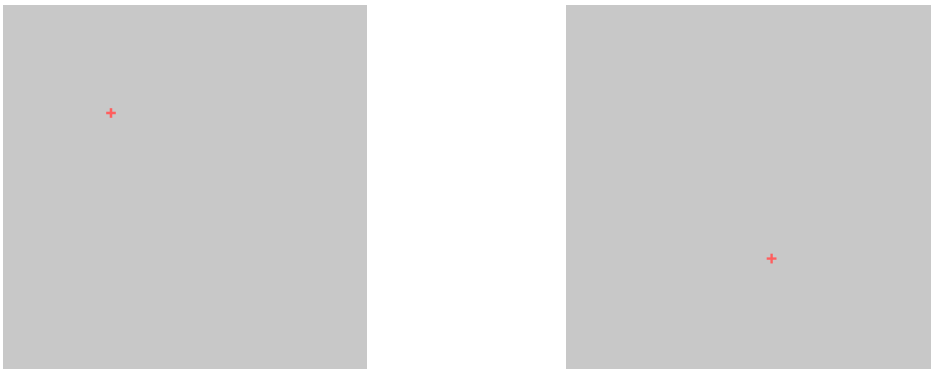
	Name	Type	Default	Description
→	inPoint	const Point2D&		
→	inDelta	const Vector2D&		Translation vector
→	inInverse	bool		Switches to the inverse operation
←	outPoint	Point2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inPoint** and **outPoint**

Read more about In-place Computation.

Examples



TranslatePoint *performed on the sample point,* ***inDelta*** *= (80.0, 120.0) and* ***inInverse*** *= False.*

TranslatePointArray

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Translates an array of points by a vector.

Syntax

```
void avl::TranslatePointArray
(
 const atl::Array<avl::Point2D>& inPoints,
 const avl::Vector2D& inDelta,
 bool inInverse,
 atl::Array<avl::Point2D>& outPoints
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		
→	inDelta	const Vector2D&		Translation vector
→	inInverse	bool		Switches to the inverse operation
←	outPoints	Array<Point2D>&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inPoints** and **outPoints**

Read more about In-place Computation.

TranslatePoint_Toward

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Translates a point towards another point by a specified distance.

Syntax

```
void avl::TranslatePoint_Toward
(
 const avl::Point2D& inPoint,
 const avl::Point2D& inTargetPoint,
 float inDistance,
 bool inInverse,
 avl::Point2D& outPoint
)
```

Parameters

	Name	Type	Default	Description
→	inPoint	const Point2D&		
→	inTargetPoint	const Point2D&		Defines the direction of the translation
→	inDistance	float		The distance between inPoint and outPoint
→	inInverse	bool		Switches to the inverse operation
←	outPoint	Point2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inPoint** and **outPoint**

Read more about In-place Computation.

TranslateRectangle

Header: AVL.h
Namespace: avl
Module: FoundationLite

Translates a rectangle by a vector.

Syntax

```
void avl::TranslateRectangle
(
 const avl::Rectangle2D& inRectangle,
 const avl::Vector2D& inDelta,
 bool inInverse,
 avl::Rectangle2D& outRectangle
)
```

Parameters

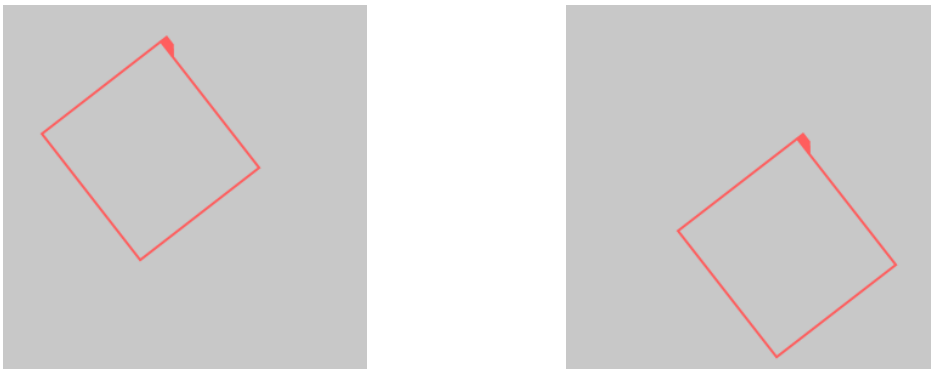
	Name	Type	Default	Description
→	inRectangle	const Rectangle2D&		
→	inDelta	const Vector2D&		Translation vector
→	inInverse	bool		Switches to the inverse operation
←	outRectangle	Rectangle2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inRectangle** and **outRectangle**

Read more about In-place Computation.

Examples



TranslateRectangle *performed on the sample rectangle,* ***inDelta*** *= (60.0, 80.0) and* ***inInverse*** *= False.*

TranslateSegment

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Translates a segment by a vector.

Syntax

```
void avl::TranslateSegment
(
 const avl::Segment2D& inSegment,
 const avl::Vector2D& inDelta,
 bool inInverse,
 const avl::TranslateAlignment::Type inDeltaAlignment,
 avl::Segment2D& outSegment
)
```

Parameters

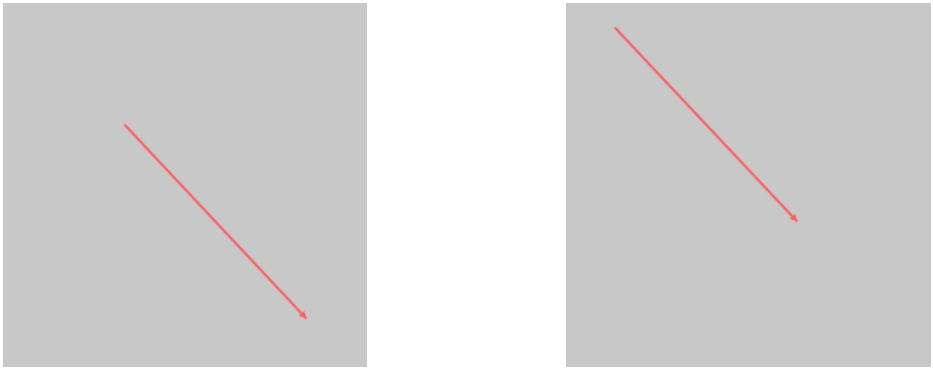
	Name	Type	Default	Description
→	inSegment	const Segment2D&		
→	inDelta	const Vector2D&		Translation vector
→	inInverse	bool		Switches to the inverse operation
→	inDeltaAlignment	const TranslateAlignment::Type		
←	outSegment	Segment2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inSegment** and **outSegment**

Read more about In-place Computation.

Examples



TranslateSegment *performed on the sample segment,* ***inDelta*** *= (-60.0, -80.0) and* ***inInverse*** *= False.*

TrimLine

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a segment contained in a box from a line.

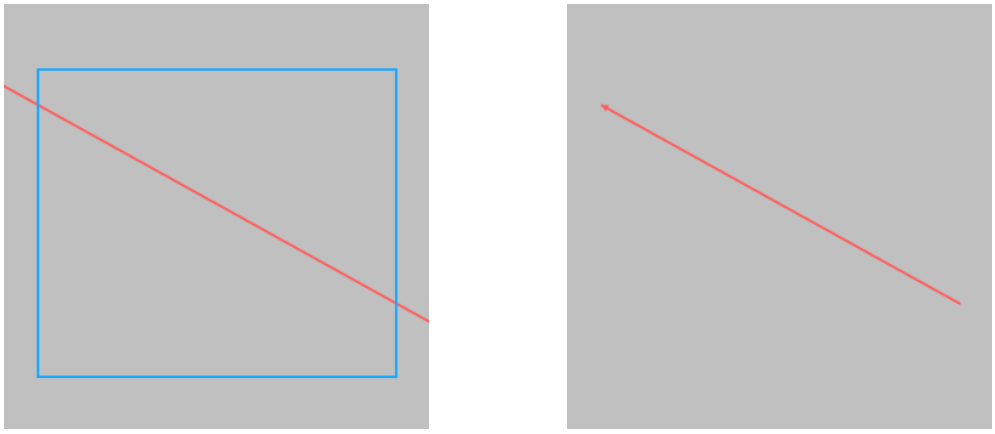
Syntax

```
void avl::TrimLine
(
 const avl::Line2D& inLine,
 const avl::Box& inBox,
 atl::Conditional<avl::Segment2D>& outSegment
)
```

Parameters

	Name	Type	Default	Description
→	inLine	const Line2D&		
→	inBox	const Box&		Box defining a region the input line will be cropped to
←	outSegment	Conditional<Segment2D>&		

Examples



TrimLine performed on line and box.

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite line on input in TrimLine.

TrimLineToRectangle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a segment contained in a rectangle from a line.

Syntax

```
void avl::TrimLineToRectangle
(
 const avl::Line2D& inLine,
 const avl::Rectangle2D& inRectangle,
 atl::Conditional<avl::Segment2D>& outSegment
)
```

Parameters

	Name	Type	Default	Description
→	inLine	const Line2D&		
→	inRectangle	const Rectangle2D&		Rectangle defining a region the input line will be cropped to
←	outSegment	Conditional<Segment2D>&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite line on input in TrimLineToRectangle.

TrimPath

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a path contained in a box from another path.

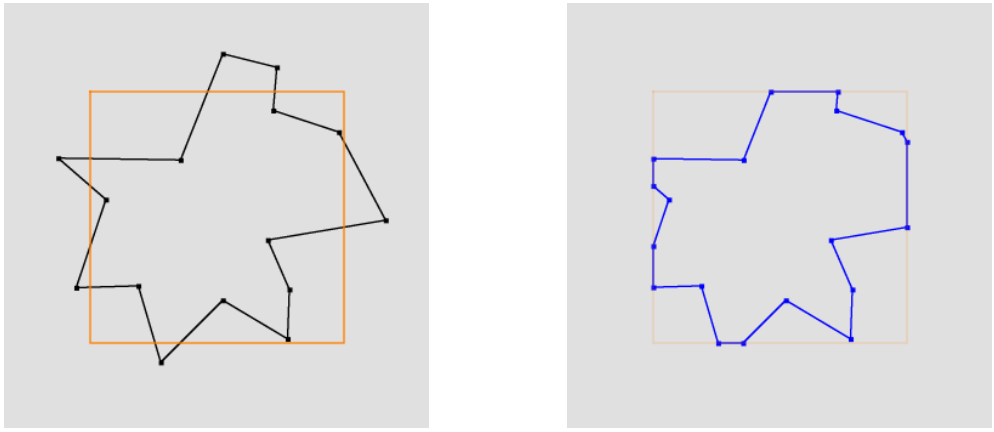
Syntax

```
void avl::TrimPath
(
 const avl::Path& inPath,
 const avl::Box& inBox,
 avl::TrimPathMethod::Type inTrimPathMethod,
 avl::Path& outPath
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
→	inBox	const Box&		
→	inTrimPathMethod	TrimPathMethod::Type	PointToPoint	
←	outPath	Path&		Output path

Examples



TrimPath *performed on the sample box and path.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Unknown method for cropping a path in TrimPath.

TrimPathArray

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates an array of paths contained in a box from another array of paths.

Syntax

```
void avl::TrimPathArray
(
 const atl::Array<avl::Path>& inPaths,
 const avl::Box& inBox,
 avl::TrimPathMethod::Type inTrimPathMethod,
 atl::Array<avl::Path>& outPaths
)
```

Parameters

	Name	Type	Default	Description
→	inPaths	const Array<Path>&		
→	inBox	const Box&		
→	inTrimPathMethod	TrimPathMethod::Type	PointToPoint	
←	outPaths	Array<Path>&		

TrimPathArrayToRectangle

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates an array of paths contained in a rectangle from another array of paths.

Syntax

```
void avl::TrimPathArrayToRectangle
(
 const atl::Array<avl::Path>& inPaths,
 const avl::Rectangle2D& inRectangle,
 avl::TrimPathMethod::Type inTrimPathMethod,
 atl::Array<avl::Path>& outPaths
)
```

Parameters

	Name	Type	Default	Description
→	inPaths	const Array<Path>&		
→	inRectangle	const Rectangle2D&		
→	inTrimPathMethod	TrimPathMethod::Type	PointToPoint	
←	outPaths	Array<Path>&		

TrimPathToRectangle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a path contained in a rectangle from another path.

Syntax

```
void avl::TrimPathToRectangle
(
 const avl::Path& inPath,
 const avl::Rectangle2D& inRectangle,
 avl::TrimPathMethod::Type inTrimPathMethod,
 avl::Path& outPath
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
→	inRectangle	const Rectangle2D&		
→	inTrimPathMethod	TrimPathMethod::Type	PointToPoint	
←	outPath	Path&		Output path

Errors

List of possible exceptions:

Error type	Description
DomainError	Unknown method for cropping a path in TrimPathToRectangle.

TrimPointArray

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Removes points not contained in a box from an array.

Syntax

```
void avl::TrimPointArray
(
 const atl::Array<avl::Point2D>& inPoints,
 const avl::Box& inBox,
 atl::Array<avl::Point2D>& outPoints
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		
→	inBox	const Box&		
←	outPoints	Array<Point2D>&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inPoints** and **outPoints**

Read more about In-place Computation.

TrimPointArrayToRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Trims an array of points to a region.

Syntax

```
void avl::TrimPointArrayToRegion
(
 const atl::Array<avl::Point2D>& inPoints,
 const avl::Region& inRegion,
 atl::Array<avl::Point2D>& outPoints
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		
→	inRegion	const Region&		Input region
←	outPoints	Array<Point2D>&		

TrimSegment

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a segment contained in a box from another segment.

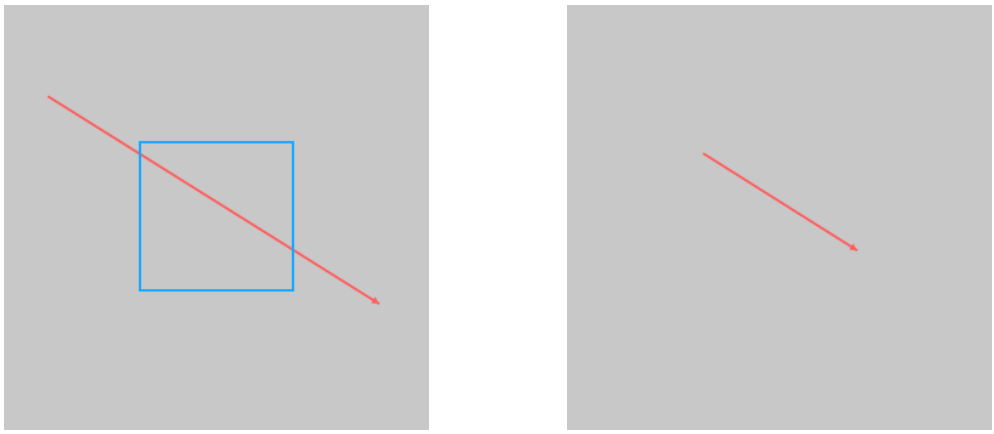
Syntax

```
void avl::TrimSegment
(
 const avl::Segment2D& inSegment,
 const avl::Box& inBox,
 atl::Conditional<avl::Segment2D>& outSegment
)
```

Parameters

	Name	Type	Default	Description
→	inSegment	const Segment2D&		
→	inBox	const Box&		
←	outSegment	Conditional<Segment2D>&		

Examples



TrimSegment *performed on the sample box and segment.*

TrimSegmentToCircle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a segment contained in a circle from another segment.

Syntax

```
void avl::TrimSegmentToCircle
(
 const avl::Segment2D& inSegment,
 const avl::Circle2D& inCircle,
 atl::Conditional<avl::Segment2D>& outSegment
)
```

Parameters

	Name	Type	Default	Description
→	inSegment	const Segment2D&		
→	inCircle	const Circle2D&		Circle defining a region the input segment will be cropped to
←	outSegment	Conditional<Segment2D>&		

TrimSegmentToPolygon

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Finds all subsegments that are inside a given polygon.

Syntax

```
void avl::TrimSegmentToPolygon
(
 const avl::Path& inPolygon,
 const avl::Segment2D& inSegment,
 atl::Array<avl::Segment2D>& outSubsegments
)
```

Parameters

	Name	Type	Default	Description
→	inPolygon	const Path&		
→	inSegment	const Segment2D&		
←	outSubsegments	Array<Segment2D>&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty path on input in TrimSegmentToPolygon.
DomainError	Open path on input in TrimSegmentToPolygon.

TrimSegmentToRectangle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a segment contained in a rectangle from another segment.

Syntax

```
void avl::TrimSegmentToRectangle
(
 const avl::Segment2D& inSegment,
 const avl::Rectangle2D& inRectangle,
 atl::Conditional<avl::Segment2D>& outSegment
)
```

Parameters

	Name	Type	Default	Description
→	inSegment	const Segment2D&		
→	inRectangle	const Rectangle2D&		Rectangle defining a region the input segment will be cropped to
←	outSegment	Conditional<Segment2D>&		

21. Path Spatial Transforms

Table of content:

AlignPath

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Moves a path from a local coordinate system to the absolute one.

Applications: Required when there is a path defined in a local coordinate system, but the next image-related filter in the program does not have any inAlignment input.

Syntax

```
void avl::AlignPath
(
 const avl::Path& inPath,
 const avl::CoordinateSystem2D& inAlignment,
 bool inInverse,
 avl::Path& outAlignedPath
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
→	inAlignment	const CoordinateSystem2D&		Coordinate system to align to
→	inInverse	bool		Switches to the inverse transform
←	outAlignedPath	Path&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inPath** and **outAlignedPath**

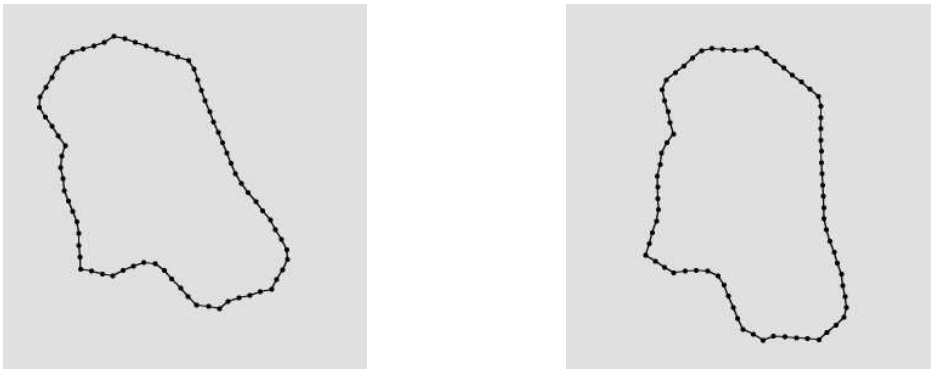
Read more about In-place Computation.

Description

AlignPath aligns the **inPath** to the **inAlignment** coordinate system. The input path is translated, rotated and scaled.

The **inAlignment** is usually an alignment of an object found by some template matching algorithm.

Examples



AlignPath *run on the sample paths.*

See Also

- TranslatePath – Translates a path by a vector.
- RotatePath – Rotates a path clockwise around a center point.

AlignPathArray

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Moves an array of paths from a local coordinate system to the absolute one.

Applications: Required when there are paths defined in a local coordinate system, but the next image-related filter in the program does not have any inAlignment input.

Syntax

```
void avl::AlignPathArray
(
 const atl::Array<avl::Path>& inPaths,
 const avl::CoordinateSystem2D& inAlignment,
 bool inInverse,
 atl::Array<avl::Path>& outAlignedPaths
)
```

Parameters

	Name	Type	Default	Description
→	inPaths	const Array<Path>&		
→	inAlignment	const CoordinateSystem2D&		Coordinate system to align to
→	inInverse	bool		Switches to the inverse transform
←	outAlignedPaths	Array<Path>&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inPaths** and **outAlignedPaths**

Read more about In-place Computation.

FitPathToPath

Header: AVL.h
Namespace: avl
Module: FoundationPro

Rotates and shifts a path to minimize average distance between its points and a reference path.

Syntax

```
void avl::FitPathToPath
(
 const avl::Path& inPath,
 const avl::Path& inReferencePath,
 avl::PathOrientationAlignment::Type inPathOrientationAlignment,
 const int inIterations,
 const float inFirstShift,
 const float inFirstRotation,
 avl::Path& outPath,
 avl::CoordinateSystem2D& outAlignment,
 float& outPathDistance
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPath	const Path&			Path to be aligned
→	inReferencePath	const Path&			Path to align to
→	inPathOrientationAlignment	PathOrientationAlignment::Type		EllipticAxes	Determines how to align paths orientation before the main algorithm
→	inIterations	const int	1 - ∞	5	Number of algorithm steps
→	inFirstShift	const float		10.0f	Magnitude of possible shift at the first step in pixels
→	inFirstRotation	const float		10.0f	Magnitude of possible rotation at the first step in degrees
←	outPath	Path&			Aligned path
←	outAlignment	CoordinateSystem2D&			The coordinate system that geometrical objects defined in the context of the path should be aligned to
←	outPathDistance	float&			Average distance of characteristic points of the input path from the reference path

Description

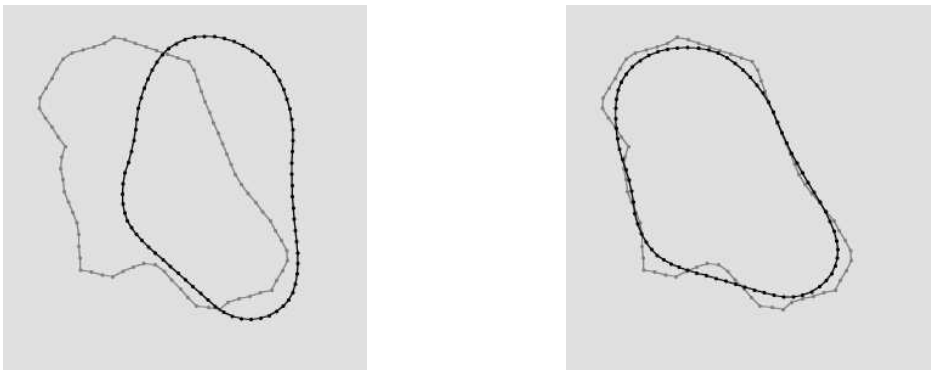
The operations repeatedly performs shifts and rotations on the **inPath** minimizing the average distance between its characteristic points and **inReferencePath**.

Initially the **inPath** is shifted so that its mass center is equals the mass center of **inReferencePath**. Then the alignment is performed in **inIterations** steps. At each step the algorithm considers:

1. Shifts of the processed path in four major directions by **ShiftMagnitude** pixels
2. Rotations of the processed path in clockwise and counter-clockwise direction by **RotationMagnitude** degrees

At each substep the operation performs shift or rotation that produces the best distance minimization results. The initial value of **ShiftMagnitude** is **inFirstShift** and at each step it is divided by 2. Analogically the initial value of **RotationMagnitude** is **inFirstRotation** and at each step it is divided by 2.

Examples



FitPathToPath *run on the sample paths with* ***inIterations*** *= 5,* ***inFirstShift*** *= 10,* ***inFirstRotation*** *= 10. The* ***inReferencePath*** *plotted in grey, the* ***inPath/outPath*** *in black.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty path on input in FitPathToPath.

InflatePath

Header: AVL.h
Namespace: avl
Module: FoundationPro

Enlarges a path by a given margin (points in the input path must be sorted clockwise).

Syntax

```
void avl::InflatePath
(
 const avl::Path& inPath,
 float inMargin,
 float inMaxPointDisplacement,
 atl::Conditional<avl::Path>& outPath
)
```

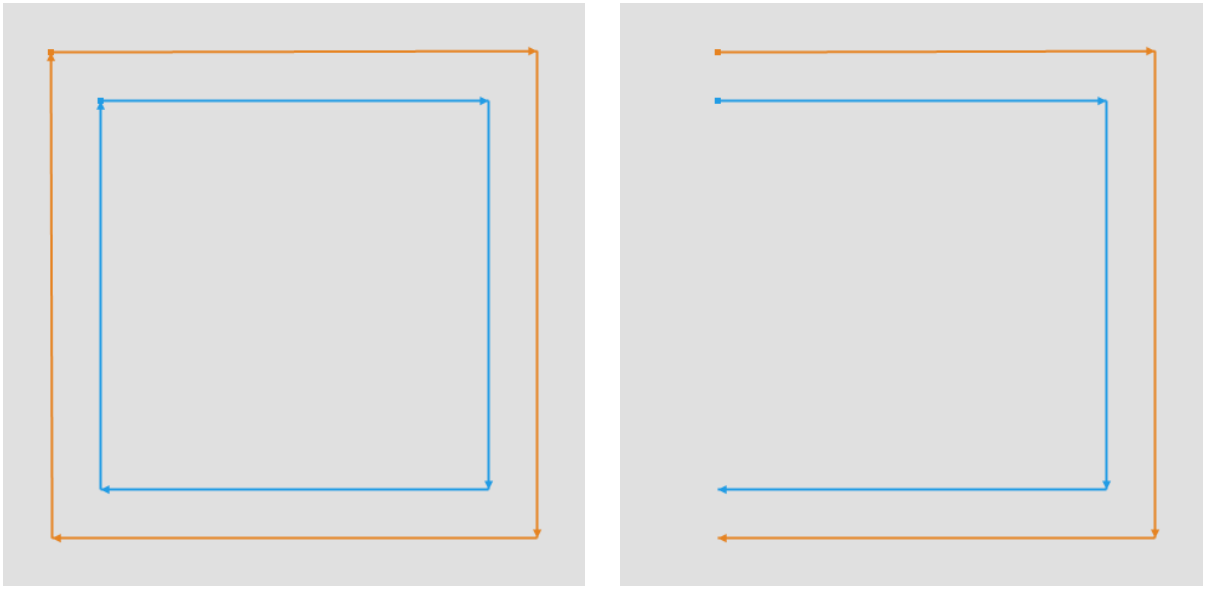
Parameters

	Name	Type	Range	Default	Description
→	inPath	const Path&			Input path
→	inMargin	float			Margin of inflation
→	inMaxPointDisplacement	float	0.0 - ∞	1000.0f	Maximal displacement of single point
←	outPath	Conditional<Path>&			Output path

Description

Creates a new path by enlarging the existing one. Distance between corresponding segments of **inPath** and **outPath** is defined by **inMargin**.

Examples



Example of inflation of initial path (blue). Result is in orange. Note that results for closed (on left) and open path (on right) are slightly different.

PathAlongArc

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Transforms a path to a coordinate system in which the 'axis' arc is vertical or horizontal.

Applications: Usually used to revert an ImageAlongArc transformation.

Syntax

```
void avl::PathAlongArc
(
 const avl::Path& inPath,
 const avl::Arc2D& inAxis,
 avl::Axis::Type inAxisType,
 float inAxisCoordinate,
 bool inInverse,
 avl::Path& outPath
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
→	inAxis	const Arc2D&		Input axis arc
→	inAxisType	Axis::Type	Y	Type of axis the input axis arc is parallel to
→	inAxisCoordinate	float	0.0f	Coordinate of the axis arc
→	inInverse	bool	True	Switches to the inverse operation
←	outPath	Path&		Transformed path

In-place Processing

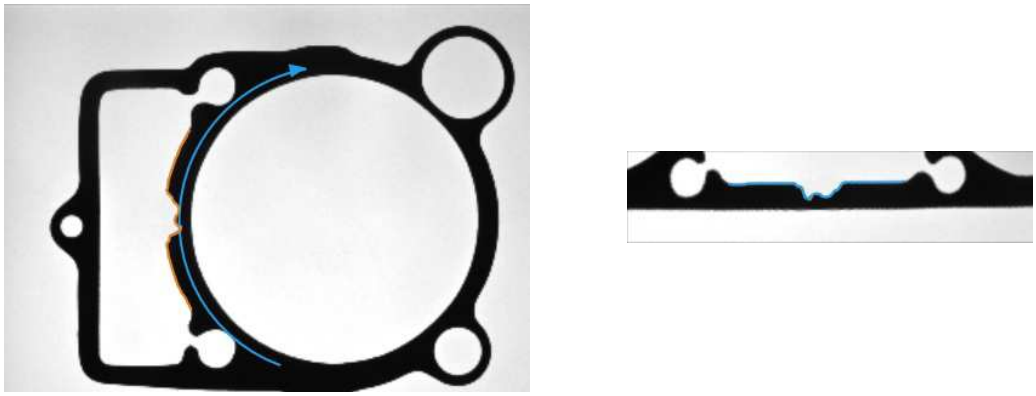
This function supports in-place data processing - you can pass the same reference to **inPath** and **outPath**

Read more about In-place Computation.

Description

This filter transforms each point in **inPath** the same way how **PointAlongArc** does.

Examples



***ImageAlongArc** performed on the sample image with **inAxisType** = X and **inScanWidth** = 50. Blue path on output image was reprojected on original image (as orange path) using **PathAlongArc** with **inAxisType** = X, **inAxisCoordinate** = 25, **inInverse** = true. **ImageAlongArc** and **PathAlongArc** used the same arc for transformations.*

Remarks

No new points are being added to input path during transformation, and this may lead to some "distortions" of path. This is especially visible on long, straight portions of input path, which are described with only 2 points: the beginning one and the ending one. After transformation, such part of path is still a line, but it may not be corresponding to the transformed image. Easy way to overcome this problem is to increase number of points creating an transformed path.

See Also

- ImageAlongArc – Creates an image from pixels traversed along an arc.
- PointAlongArc – Transforms a point to a coordinate system in which the 'axis' arc is vertical or horizontal.
- PathAlongPath – Transforms a path to a coordinate system in which the 'axis' path is vertical or horizontal.

PathAlongPath

Header: AVL.h
Namespace: avl
Module: FoundationPro

Transforms a path to a coordinate system in which the 'axis' path is vertical or horizontal.

Applications: Usually used to revert an ImageAlongPath transformation.

Syntax

```
void avl::PathAlongPath
(
 const avl::Path& inPath,
 const avl::Path& inAxis,
 avl::Axis::Type inAxisType,
 float inAxisCoordinate,
 bool inInverse,
 avl::Path& outPath
)
```

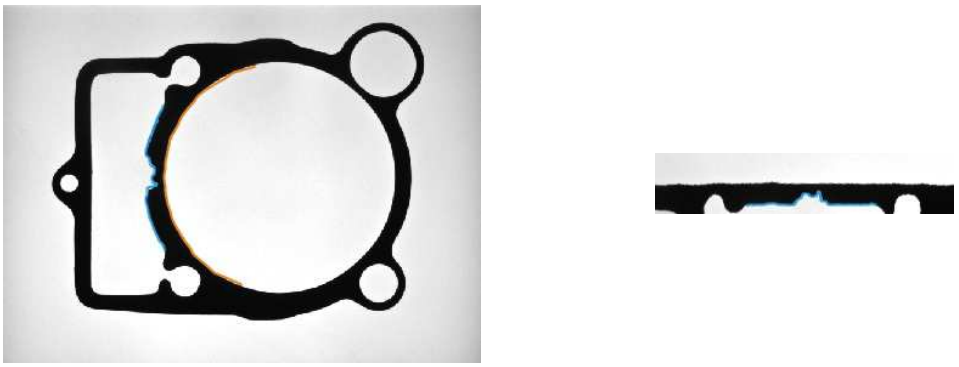
Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
→	inAxis	const Path&		Input axis path
→	inAxisType	Axis::Type	Y	Type of axis the input axis path is parallel to
→	inAxisCoordinate	float	0.0f	Coordinate of the axis path
→	inInverse	bool	True	Switches to the inverse operation
←	outPath	Path&		Transformed path

Description

This filter transforms each point in **inPath** the same way how **PointAlongPath** does.

Examples



ImageAlongPath *performed on the sample image with* ***inAxisType*** *= X and* ***inScanWidth*** *= 50. Blue path on output image was reprojected on original image using* ***PathAlongPath*** *with* ***inAxisType*** *= X,* ***inAxisCoordinate*** *= 25,* ***inInverse*** *= true.* ***ImageAlongPath*** *and* ***PathAlongPath*** *used the same path (the orange one) for transformations.*

Remarks

No new points are being added to input path during transformation, and this may lead to some "distortions" of path. This is especially visible on long, straight portions of input path, which are described with only 2 points: the beginning one and the ending one. After transformation, such part of path is still a line, but it may not be corresponding to the transformed image. Easy way to overcome this problem is to increase number of points creating an transformed path.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty path on input in PathAlongPath.

See Also

- ImageAlongPath – Creates an image from pixels traversed along a path.
- PointAlongPath – Transforms a point to a coordinate system in which the 'axis' path is vertical or horizontal.
- PathAlongArc – Transforms a path to a coordinate system in which the 'axis' arc is vertical or horizontal.

PathProjectionProfile

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes the distance from the line to the path.

Syntax

```
void avl::PathProjectionProfile
(
 const avl::Path& inPath,
 const avl::Line2D& inLine,
 avl::Profile& outProjectionProfile,
 atl::Array<avl::Segment2D>& diagProjectionSegments
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
→	inLine	const Line2D&		Input line to project to
←	outProjectionProfile	Profile&		Output path projection
🔍	diagProjectionSegments	Array<Segment2D>&		Segments corresponding to every profile value

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

RescalePath

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Translates each point of a path proportionally to its distance to a reference point.

Syntax

```
void avl::RescalePath
(
 const avl::Path& inPath,
 atl::Optional<const avl::Point2D&> inReferencePoint,
 float inScale,
 bool inInverse,
 avl::Path& outPath
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
→	inReferencePoint	Optional<const Point2D&>	NIL	The point to which all distances change linearly (the mass center by default)
→	inScale	float	1.0f	Scaling factor
→	inInverse	bool		Switches to the inverse operation
←	outPath	Path&		Output path

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inPath** and **outPath**

Read more about In-place Computation.

RescalePathArray

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Translates each point of each path proportionally to its distance to a reference point.

Syntax

```
void avl::RescalePathArray
(
 const atl::Array<avl::Path>& inPaths,
 atl::Optional<const avl::Point2D&> inReferencePoint,
 float inScale,
 bool inInverse,
 atl::Array<avl::Path>& outPaths
)
```

Parameters

	Name	Type	Default	Description
→	inPaths	const Array<Path>&		
→	inReferencePoint	Optional<const Point2D&>	NIL	The point to which all distances change linearly (the mass center by default)
→	inScale	float	1.0f	Scaling factor
→	inInverse	bool		Switches to the inverse operation
←	outPaths	Array<Path>&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inPaths** and **outPaths**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Reference point cannot be computed in RescalePathArray.

ReversePath

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Reverses the order of points in a path.

Syntax

```
void avl::ReversePath
(
 avl::Path& ioPath
)
```

Parameters

	Name	Type	Default	Description
↔	ioPath	Path&		

RotatePath

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Rotates a path clockwise around a center point.

Syntax

```
void avl::RotatePath
(
 const avl::Path& inPath,
 atl::Optional<const avl::Point2D&> inCenter,
 float inAngle,
 bool inInverse,
 avl::Path& outPath
)
```

Parameters

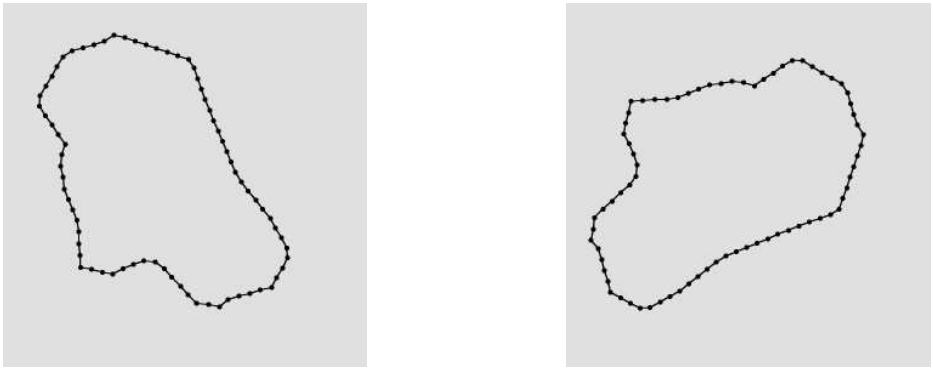
	Name	Type	Default	Description
→	inPath	const Path&		Input path
→	inCenter	Optional<const Point2D&>	NIL	Center of rotation (the mass center by default)
→	inAngle	float		Clockwise rotation angle
→	inInverse	bool		Switches to the inverse operation
←	outPath	Path&		Output path

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inPath** and **outPath**

Read more about In-place Computation.

Examples



RotatePath *run on the sample paths with* ***inAngle*** *= 90,* ***inCenter*** *= Nil.*

See Also

- TranslatePath – Translates a path by a vector.
- AlignPath – Moves a path from a local coordinate system to the absolute one.

RotatePathArray

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Rotates an array of paths clockwise around a center point.

Syntax

```
void avl::RotatePathArray
(
 const atl::Array<avl::Path>& inPaths,
 atl::Optional<const avl::Point2D&> inCenter,
 float inAngle,
 bool inInverse,
 atl::Array<avl::Path>& outPaths
)
```

Parameters

	Name	Type	Default	Description
→	inPaths	const Array<Path>&		
→	inCenter	Optional<const Point2D&>	NIL	Center of rotation (the mass center by default)
→	inAngle	float		Clockwise rotation angle
→	inInverse	bool		Switches to the inverse operation
←	outPaths	Array<Path>&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inPaths** and **outPaths**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Center of rotation cannot be computed in RotatePathArray.

ShiftPath

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Moves every vertex of path along bisector of the angle between incident segments.

Syntax

```
void avl::ShiftPath
(
 const avl::Path& inPath,
 float inDistance,
 avl::ShiftDirection::Type inDirection,
 avl::Path& outPath
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
→	inDistance	float		
→	inDirection	ShiftDirection::Type		
←	outPath	Path&		Output path

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inPath** and **outPath**

Read more about In-place Computation.

TranslatePath

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Translates a path by a vector.

Syntax

```
void avl::TranslatePath
(
 const avl::Path& inPath,
 const avl::Vector2D& inDelta,
 bool inInverse,
 avl::Path& outPath
)
```

Parameters

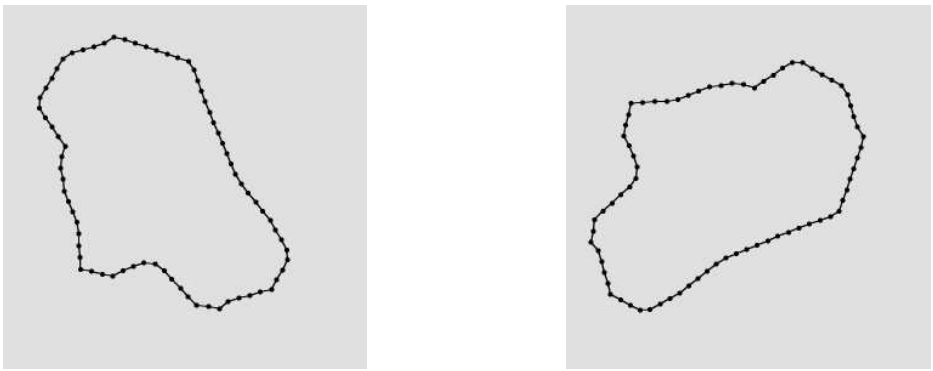
	Name	Type	Default	Description
→	inPath	const Path&		Input path
→	inDelta	const Vector2D&		Translation vector
→	inInverse	bool		Switches to the inverse operation
←	outPath	Path&		Output path

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inPath** and **outPath**

Read more about In-place Computation.

Examples



TranslatePath *run on the sample paths with* ***inDelta*** *= Vector2D(50,0).*

See Also

- RotatePath – Rotates a path clockwise around a center point.
- AlignPath – Moves a path from a local coordinate system to the absolute one.

TranslatePathArray

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Translates an array of paths by a vector.

Syntax

```
void avl::TranslatePathArray
(
 const atl::Array<avl::Path>& inPaths,
 const avl::Vector2D& inDelta,
 bool inInverse,
 atl::Array<avl::Path>& outPaths
)
```

Parameters

	Name	Type	Default	Description
→	inPaths	const Array<Path>&		
→	inDelta	const Vector2D&		Translation vector
→	inInverse	bool		Switches to the inverse operation
←	outPaths	Array<Path>&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inPaths** and **outPaths**

Read more about In-place Computation.

TransposePath

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Flips and rotates a path so that x-coordinates are exchanged with y-coordinates.

Syntax

```
void avl::TransposePath
(
 const avl::Path& inPath,
 avl::Path& outPath
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
←	outPath	Path&		Output path

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inPath** and **outPath**

Read more about In-place Computation.

22. Geometry 3D Spatial Transforms

Table of content:

- RotatePoint3DArray_AroundSegment
- RotatePoint3D_AroundSegment
- RotateSegment3D
- RotateSegment3D_AroundSegment
- RotateVector3D
- Segment2DToSegment3D_AxesPlane
- Segment3DToSegment2D_AxesPlane
- Segments2DToSegments3D_AxesPlane
- Segments3DToSegments2D_AxesPlane
- SplitPointsByPlane
- TransformCircle3D
- TransformLine3D
- TransformPlane
- TransformPoint3D
- TransformPoint3DArray
- TransformSegment3D
- TransformVector3D
- TranslateBox3D
- TranslateCircle3D
- TranslateLine3D
- TranslatePlane
- TranslatePoint3D
- TranslatePoint3DArray
- TranslatePoint3D_Toward
- TranslateSegment3D
- TrimLine3D
- TrimPoint3DArray
- TrimSegment3D
- Vector2DToVector3D_AxesPlane
- Vector3DToVector2D_AxesPlane

AlignPoint3DArrayToPlane

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Rotates an array of 3D points with a rotation that transforms the input plane to be parallel to XY plane.

Syntax

```
void avl::AlignPoint3DArrayToPlane
(
 const atl::Array<avl::Point3D>& inPoints,
 const avl::Plane3D& inPlane,
 atl::Array<avl::Point3D>& outPoints,
 atl::Optional<avl::Matrix&> outTransform = atl::NIL,
 atl::Optional<avl::Matrix&> outInvertedTransform = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point3D>&		
→	inPlane	const Plane3D&		The plane that determines rotation transform
←	outPoints	Array<Point3D>&		
←	outTransform	Optional<Matrix&>	NIL	Matrix of the transform used to align the input points
←	outInvertedTransform	Optional<Matrix&>	NIL	Matrix of the transform rotating the output points to the input ones

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outTransform**, **outInvertedTransform**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite plane on input in AlignPoint3DArrayToPlane.

Box2DToBox3D_AxesPlane

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Projects a 2D box onto a plane defined by 3D coordinate axes.

Syntax

```
void avl::Box2DToBox3D_AxesPlane
(
 const avl::Box& inBox2D,
 avl::PlaneType::Type inPlaneType,
 float inCoordinateValue,
 avl::Box3D& outBox3D,
 atl::Optional<avl::Plane3D&> outProjectionPlane = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inBox2D	const Box&		
→	inPlaneType	PlaneType::Type		
→	inCoordinateValue	float		Coordinate of the chosen axes plane
←	outBox3D	Box3D&		
←	outProjectionPlane	Optional<Plane3D&>	NIL	Plane the input is projected onto

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outProjectionPlane**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect plane type.

Box3DToBox2D_AxesPlane

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Projects a 3D box onto a plane defined by coordinate axes.

Syntax

```
void avl::Box3DToBox2D_AxesPlane
(
 const avl::Box3D& inBox3D,
 avl::PlaneType::Type inPlaneType,
 avl::Box& outBox2D,
 atl::Optional<avl::Rectangle2D&> outRectangle2D = atl::NIL,
 atl::Optional<avl::Plane3D&> outProjectionPlane = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inBox3D	const Box3D&		
→	inPlaneType	PlaneType::Type		
←	outBox2D	Box&		
←	outRectangle2D	Optional<Rectangle2D&>	NIL	
←	outProjectionPlane	Optional<Plane3D&>	NIL	Plane the input is projected onto

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outRectangle2D**, **outProjectionPlane**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect plane type.

Circle2DToCircle3D_AxesPlane

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Projects a 2D circle onto a plane defined by 3D coordinate axes.

Syntax

```
void avl::Circle2DToCircle3D_AxesPlane
(
 const avl::Circle2D& inCircle2D,
 avl::PlaneType::Type inPlaneType,
 float inCoordinateValue,
 avl::Circle3D& outCircle3D,
 atl::Optional<avl::Plane3D&> outProjectionPlane = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inCircle2D	const Circle2D&		
→	inPlaneType	PlaneType::Type		
→	inCoordinateValue	float		Coordinate of the chosen axes plane
←	outCircle3D	Circle3D&		
←	outProjectionPlane	Optional<Plane3D&>	NIL	Plane the input is projected onto

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outProjectionPlane**.

Read more about Optional Outputs.

Circle3DToCircle2D_AxesPlane

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Projects a 3D circle onto a plane defined by coordinate axes.

Syntax

```
void avl::Circle3DToCircle2D_AxesPlane
(
 const avl::Circle3D& inCircle3D,
 avl::PlaneType::Type inPlaneType,
 avl::Circle2D& outCircle2D,
 atl::Optional<avl::Plane3D&> outProjectionPlane = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inCircle3D	const Circle3D&		
→	inPlaneType	PlaneType::Type		
←	outCircle2D	Circle2D&		
←	outProjectionPlane	Optional<Plane3D&>	NIL	Plane the input is projected onto

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outProjectionPlane**.

Read more about Optional Outputs.

Line2DToLine3D_AxesPlane

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Projects a 2D line onto a plane defined by 3D coordinate axes.

Syntax

```
void avl::Line2DToLine3D_AxesPlane
(
 const avl::Line2D& inLine2D,
 avl::PlaneType::Type inPlaneType,
 float inCoordinateValue,
 avl::Line3D& outLine3D,
 atl::Optional<avl::Plane3D&> outProjectionPlane = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inLine2D	const Line2D&		
→	inPlaneType	PlaneType::Type		
→	inCoordinateValue	float		Coordinate of the chosen axes plane
←	outLine3D	Line3D&		
←	outProjectionPlane	Optional<Plane3D&>	NIL	Plane the input is projected onto

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outProjectionPlane**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect plane type.

Line3DToLine2D_AxesPlane

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Projects a 3D line onto a plane defined by coordinate axes.

Syntax

```
void avl::Line3DToLine2D_AxesPlane
(
 const avl::Line3D& inLine3D,
 avl::PlaneType::Type inPlaneType,
 avl::Line2D& outLine2D,
 atl::Optional<avl::Plane3D&> outProjectionPlane = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inLine3D	const Line3D&		
→	inPlaneType	PlaneType::Type		
←	outLine2D	Line2D&		
←	outProjectionPlane	Optional<Plane3D&>	NIL	Plane the input is projected onto

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outProjectionPlane**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect plane type.

MirrorPoint3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Mirrors a point according to the input plane.

Syntax

```
void avl::MirrorPoint3D
(
 const avl::Point3D& inPoint3D,
 const avl::Plane3D& inPlane,
 avl::Point3D& outPoint3D,
 atl::Optional<avl::Matrix&> outTransform = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPoint3D	const Point3D&		
→	inPlane	const Plane3D&		The plane being a mirror
←	outPoint3D	Point3D&		
←	outTransform	Optional<Matrix&>	NIL	Matrix of the transform used to mirror the input point

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outTransform**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite plane on input in MirrorPoint3D.

MirrorPoint3DArray

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Mirrors each point from the array according to the input plane.

Syntax

```
void avl::MirrorPoint3DArray
(
 const atl::Array<avl::Point3D>& inPoints3D,
 const avl::Plane3D& inPlane,
 atl::Array<avl::Point3D>& outPoints3D,
 atl::Optional<avl::Matrix&> outTransform = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPoints3D	const Array<Point3D>&		
→	inPlane	const Plane3D&		The plane being a mirror
←	outPoints3D	Array<Point3D>&		
←	outTransform	Optional<Matrix&>	NIL	Matrix of the transform used to mirror the input points

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outTransform**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite plane on input in MirrorPoint3DArray.

Point2DToPoint3D_AxesPlane

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Projects a 2D point onto a plane defined by 3D coordinate axes.

Syntax

```
void avl::Point2DToPoint3D_AxesPlane
(
 const avl::Point2D& inPoint2D,
 avl::PlaneType::Type inPlaneType,
 float inCoordinateValue,
 avl::Point3D& outPoint3D,
 atl::Optional<avl::Plane3D&> outProjectionPlane = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPoint2D	const Point2D&		
→	inPlaneType	PlaneType::Type		
→	inCoordinateValue	float		Coordinate of the chosen axes plane
←	outPoint3D	Point3D&		
←	outProjectionPlane	Optional<Plane3D&>	NIL	Plane the input is projected onto

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outProjectionPlane**.

Read more about Optional Outputs.

Point3DToPoint2D_AxesPlane

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Projects a 3D point onto a plane defined by coordinate axes.

Syntax

```
void avl::Point3DToPoint2D_AxesPlane
(
 const avl::Point3D& inPoint3D,
 avl::PlaneType::Type inPlaneType,
 avl::Point2D& outPoint2D,
 atl::Optional<avl::Plane3D&> outProjectionPlane = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPoint3D	const Point3D&		
→	inPlaneType	PlaneType::Type		
←	outPoint2D	Point2D&		
←	outProjectionPlane	Optional<Plane3D&>	NIL	Plane the input is projected onto

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outProjectionPlane**.

Read more about Optional Outputs.

Points2DToPoints3D_AxesPlane

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Projects an array of 2D points onto a plane defined by 3D coordinate axes.

Syntax

```
void avl::Points2DToPoints3D_AxesPlane
(
 const atl::Array<avl::Point2D>& inPoints2D,
 avl::PlaneType::Type inPlaneType,
 float inCoordinateValue,
 atl::Array<avl::Point3D>& outPoints3D,
 atl::Optional<avl::Plane3D&> outProjectionPlane = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPoints2D	const Array<Point2D>&		
→	inPlaneType	PlaneType::Type		
→	inCoordinateValue	float		Coordinate of the chosen axes plane
←	outPoints3D	Array<Point3D>&		
←	outProjectionPlane	Optional<Plane3D&>	NIL	Plane the input is projected onto

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outProjectionPlane**.

Read more about Optional Outputs.

Points3DToPoints2D_AxesPlane

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Projects an array of 3D points onto a plane defined by coordinate axes.

Syntax

```
void avl::Points3DToPoints2D_AxesPlane
(
 const atl::Array<avl::Point3D>& inPoints3D,
 avl::PlaneType::Type inPlaneType,
 atl::Array<avl::Point2D>& outPoints2D,
 atl::Optional<avl::Plane3D&> outProjectionPlane = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPoints3D	const Array<Point3D>&		
→	inPlaneType	PlaneType::Type		
←	outPoints2D	Array<Point2D>&		
←	outProjectionPlane	Optional<Plane3D&>	NIL	Plane the input is projected onto

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outProjectionPlane**.

Read more about Optional Outputs.

ProjectLineOntoPlane3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Projects a line 3D onto a plane.

Syntax

```
void avl::ProjectLineOntoPlane3D
(
 const avl::Line3D& inLine3D,
 const avl::Plane3D& inPlane,
 avl::Line3D& outLineOnPlane
)
```

Parameters

	Name	Type	Default	Description
→	inLine3D	const Line3D&		
→	inPlane	const Plane3D&		
←	outLineOnPlane	Line3D&		Line in 3D projected on a plane orthogonally

Errors

List of possible exceptions:

Error type	Description
DomainError	Cannot project line perpendicular to plane in ProjectLineOntoPlane3D.
DomainError	Indefinite line on input in ProjectLineOntoPlane3D.
DomainError	Indefinite plane on input in ProjectLineOntoPlane3D.

ProjectPoint2DOntoPlane3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Projects a point on a plane Z=0 onto a given plane, translating it parallel to Z axis.

Syntax

```
void avl::ProjectPoint2DOntoPlane3D
(
 const avl::Point2D& inPoint2D,
 const avl::Plane3D& inPlane,
 avl::Point3D& outPointOnPlane
)
```

Parameters

	Name	Type	Default	Description
→	inPoint2D	const Point2D&		
→	inPlane	const Plane3D&		
←	outPointOnPlane	Point3D&		Point projected on a plane parallel to Z axis

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite plane on input in ProjectPoint2DOntoPlane3D.
DomainError	Point cannot be projected in ProjectPoint2DOntoPlane3D.

ProjectPointOnLine3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Projects a point 3D onto a line in 3D.

Syntax

```
void avl::ProjectPointOnLine3D
(
 const avl::Point3D& inPoint3D,
 const avl::Line3D& inLine3D,
 avl::Point3D& outPointOnLine3D
)
```

Parameters

	Name	Type	Default	Description
→	inPoint3D	const Point3D&		
→	inLine3D	const Line3D&		
←	outPointOnLine3D	Point3D&		Point projected on a line orthogonally

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite line on input in ProjectPointOnLine3D.

ProjectPointOntoPlane3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Projects a point 3D onto a plane.

Syntax

```
void avl::ProjectPointOntoPlane3D
(
 const avl::Point3D& inPoint3D,
 const avl::Plane3D& inPlane,
 avl::Point3D& outPointOnPlane
)
```

Parameters

	Name	Type	Default	Description
→	inPoint3D	const Point3D&		
→	inPlane	const Plane3D&		
←	outPointOnPlane	Point3D&		Point projected on a plane orthogonally

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite plane on input in ProjectPointOntoPlane3D.

ProjectPointOrthogonal

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Converts a point through a simple orthogonal projection, parallel to the Z axis.

Syntax

```
void avl::ProjectPointOrthogonal
(
 const avl::Point3D& inPoint3D,
 const avl::Point2D& inCenter,
 avl::Point2D& outPoint2D
)
```

Parameters

	Name	Type	Default	Description
→	inPoint3D	const Point3D&		
→	inCenter	const Point2D&		Center of projection, i.e. the target for projected points (0, 0, z)
←	outPoint2D	Point2D&		

ProjectPointPerspective

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Converts a point through a perspective projection, parallel to the Z axis.

Syntax

```
void avl::ProjectPointPerspective
(
 const avl::Point3D& inPoint3D,
 const avl::Point2D& inCenter,
 const avl::Point3D& inFocalPoint,
 float inFocalLength,
 avl::Point2D& outPoint2D
)
```

Parameters

	Name	Type	Default	Description
→	inPoint3D	const Point3D&		
→	inCenter	const Point2D&		Center of projection, i.e. the target for projected points (x_focal, y_focal, z)
→	inFocalPoint	const Point3D&		The camera center, i.e. the point we measure (x, y, z) against
→	inFocalLength	float		The multiplier for the x and y coordinates, which is divided by z. If negative, projects in opposite direction.
←	outPoint2D	Point2D&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Projected point coplanar with the focal point, but not equal to it in ProjectPointPerspective.

RescaleBox3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Changes the distances of box in 3D to a reference point.

Syntax

```
void avl::RescaleBox3D
(
 const avl::Box3D& inBox3D,
 const avl::Point3D& inReferencePoint,
 float inScale,
 bool inInverse,
 avl::Box3D& outBox3D
)
```

Parameters

	Name	Type	Default	Description
→	inBox3D	const Box3D&		
→	inReferencePoint	const Point3D&		Point to which the distances will be changed
→	inScale	float	1.0f	Scaling factor
→	inInverse	bool		Switches to the inverse operation
←	outBox3D	Box3D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inBox3D** and **outBox3D**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Scale cannot be zero in an inverse rescaling of a box in RescaleBox3D.

RescaleCircle3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Changes radius of a circle in 3D and translates its center in relation to a reference point.

Syntax

```
void avl::RescaleCircle3D
(
 const avl::Circle3D& inCircle3D,
 atl::Optional<const avl::Point3D&> inReferencePoint,
 float inScale,
 bool inInverse,
 avl::Circle3D& outCircle3D
)
```

Parameters

	Name	Type	Default	Description
→	inCircle3D	const Circle3D&		
→	inReferencePoint	Optional<const Point3D&>	NIL	The point to which the distance of the circle center is changed (circle center by default)
→	inScale	float	1.0f	Scaling factor
→	inInverse	bool		Switches to the inverse operation
←	outCircle3D	Circle3D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inCircle3D** and **outCircle3D**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite circle on input in RescaleCircle3D.
DomainError	Scale cannot be zero in an inverse scaling of a circle in RescaleCircle3D.

RescaleLine3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Changes the input line in 3D distance to a reference point.

Syntax

```
void avl::RescaleLine3D
(
 const avl::Line3D& inLine3D,
 const avl::Point3D& inReferencePoint,
 float inScale,
 bool inInverse,
 avl::Line3D& outLine3D
)
```

Parameters

	Name	Type	Default	Description
→	inLine3D	const Line3D&		Input line in 3D
→	inReferencePoint	const Point3D&		The point to which the distance of line in 3D is changed
→	inScale	float	1.0f	Scaling factor
→	inInverse	bool		Switches to the inverse operation
←	outLine3D	Line3D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inLine3D** and **outLine3D**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite line on input in RescaleLine3D.
DomainError	Scale cannot be zero in an inverse rescaling of a line in RescaleLine3D.

RescalePlane

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Changes the input plane distance to a reference point.

Syntax

```
void avl::RescalePlane
(
 const avl::Plane3D& inPlane,
 const avl::Point3D& inReferencePoint,
 float inScale,
 bool inInverse,
 avl::Plane3D& outPlane
)
```

Parameters

	Name	Type	Default	Description
→	inPlane	const Plane3D&		
→	inReferencePoint	const Point3D&		The point to which the distance of plane is changed
→	inScale	float	1.0f	Scaling factor
→	inInverse	bool		Switches to the inverse operation
←	outPlane	Plane3D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inPlane** and **outPlane**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite plane on input in RescalePlane.
DomainError	Scale cannot be zero in an inverse rescaling of a plane in RescalePlane.

RescalePoint3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Changes the distance of a point to a reference point.

Syntax

```
void avl::RescalePoint3D
(
 const avl::Point3D& inPoint3D,
 const avl::Point3D& inReferencePoint,
 float inScale,
 bool inInverse,
 avl::Point3D& outPoint3D
)
```

Parameters

	Name	Type	Default	Description
→	inPoint3D	const Point3D&		
→	inReferencePoint	const Point3D&		Point to which the distance will be changed
→	inScale	float	1.0f	Scaling factor
→	inInverse	bool		Switches to the inverse operation
←	outPoint3D	Point3D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inPoint3D** and **outPoint3D**, **inReferencePoint** and **outPoint3D**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Scale cannot be zero in an inverse rescaling of a point.

RescalePoint3DArray

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Changes the distances of array points to a reference point.

Syntax

```
void avl::RescalePoint3DArray
(
 const atl::Array<avl::Point3D>& inPoints3D,
 const avl::Point3D& inReferencePoint,
 float inScale,
 bool inInverse,
 atl::Array<avl::Point3D>& outPoints3D
)
```

Parameters

	Name	Type	Default	Description
→	inPoints3D	const Array<Point3D>&		
→	inReferencePoint	const Point3D&		Point to which the distances will be changed
→	inScale	float	1.0f	Scaling factor
→	inInverse	bool		Switches to the inverse operation
←	outPoints3D	Array<Point3D>&		

Hardware Acceleration

This operation is optimized for SSE2 technology.

This operation is optimized for AVX2 technology.

This operation is optimized for NEON technology.

Errors

List of possible exceptions:

Error type	Description
DomainError	Scale cannot be zero in an inverse rescaling of a point.

RescaleSegment3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Lengthens or shortens a segment in 3D relatively.

Syntax

```
void avl::RescaleSegment3D
(
 const avl::Segment3D& inSegment3D,
 atl::Optional<const avl::Point3D&> inReferencePoint,
 float inScale,
 bool inInverse,
 avl::Segment3D& outSegment3D
)
```

Parameters

	Name	Type	Default	Description
→	inSegment3D	const Segment3D&		Input segment in 3D
→	inReferencePoint	Optional<const Point3D&>	NIL	The point to which all distances change linearly (the mass center by default)
→	inScale	float	1.0f	Scaling factor (negative values invert the segment)
→	inInverse	bool		Switches to the inverse operation
←	outSegment3D	Segment3D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inSegment3D** and **outSegment3D**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Scale cannot be zero in an inverse rescaling of a segment in RescaleSegment3D.

RescaleVector3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Lengthens or shortens a 3D vector relatively preserving its direction.

Syntax

```
void avl::RescaleVector3D
(
 const avl::Vector3D& inVector3D,
 float inScale,
 bool inInverse,
 avl::Vector3D& outVector3D
)
```

Parameters

	Name	Type	Default	Description
→	inVector3D	const Vector3D&		
→	inScale	float	1.0f	Scaling factor
→	inInverse	bool		Switches to the inverse operation
←	outVector3D	Vector3D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inVector3D** and **outVector3D**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Scale cannot be zero in an inverse scaling of vector in RescaleVector3D.

ResizeBox3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Changes the dimensions of a box in 3D.

Syntax

```
void avl::ResizeBox3D
(
 const avl::Box3D& inBox3D,
 const avl::Anchor3D& inAnchor,
 atl::Optional<float> inNewXLength,
 atl::Optional<float> inNewYLength,
 atl::Optional<float> inNewZLength,
 avl::Box3D& outBox3D
)
```

Parameters

	Name	Type	Range	Default	Description
→	inBox3D	const Box3D&			Input box in 3D
→	inAnchor	const Anchor3D&			Point of the box in 3D which position will not change
→	inNewXLength	Optional<float>	0.0 - ∞	NIL	Target length of the box in x-coordinate
→	inNewYLength	Optional<float>	0.0 - ∞	NIL	Target length of the box in y-coordinate
→	inNewZLength	Optional<float>	0.0 - ∞	NIL	Target length of the box in z-coordinate
←	outBox3D	Box3D&			Resized box

ResizeBox3D_Delta

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Changes the dimensions of a box in 3D by adding some values.

Syntax

```
void avl::ResizeBox3D_Delta
(
 const avl::Box3D& inBox3D,
 const avl::Anchor3D& inAnchor,
 float inXLengthDelta,
 float inYLengthDelta,
 float inZLengthDelta,
 avl::Box3D& outBox3D
)
```

Parameters

	Name	Type	Default	Description
→	inBox3D	const Box3D&		Input box
→	inAnchor	const Anchor3D&		Point of the box in 3D which position will not change
→	inXLengthDelta	float	0.0f	Value added to length of the box in X axis
→	inYLengthDelta	float	0.0f	Value added to length of the box in X axis
→	inZLengthDelta	float	0.0f	Value added to length of the box in X axis
←	outBox3D	Box3D&		Resized box

Errors

List of possible exceptions:

Error type	Description
DomainError	Negative box dimensions in ResizeBox3D_Delta.

ResizeBox3D_Relative

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Resizes a box in 3D to relatively defined dimensions.

Syntax

```
void avl::ResizeBox3D_Relative
(
 const avl::Box3D& inBox3D,
 const avl::Anchor3D& inAnchor,
 float inXScale,
 float inYScale,
 float inZScale,
 avl::Box3D& outBox3D
)
```

Parameters

	Name	Type	Range	Default	Description
→	inBox3D	const Box3D&			Input box in 3D
→	inAnchor	const Anchor3D&			Point of the box in 3D which position will not change
→	inXScale	float	0.0 - ∞	1.0f	Scale factor in X axis
→	inYScale	float	0.0 - ∞	1.0f	Scale factor in Y axis
→	inZScale	float	0.0 - ∞	1.0f	Scale factor in Z axis
←	outBox3D	Box3D&			Resized box

ResizeCircle3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Changes radius of a circle in 3D.

Syntax

```
void avl::ResizeCircle3D
(
 const avl::Circle3D& inCircle3D,
 float inNewSize,
 avl::Circle3D& outCircle3D
)
```

Parameters

	Name	Type	Range	Default	Description
→	inCircle3D	const Circle3D&			
→	inNewSize	float	0.0 - ∞	1.0f	New value for radius
←	outCircle3D	Circle3D&			

ResizeCircle3D_Delta

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Changes radius of a circle in 3D by adding a value.

Syntax

```
void avl::ResizeCircle3D_Delta
(
 const avl::Circle3D& inCircle3D,
 float inDelta,
 avl::Circle3D& outCircle3D
)
```

Parameters

	Name	Type	Default	Description
→	inCircle3D	const Circle3D&		
→	inDelta	float	0.0f	Value added to circle radius
←	outCircle3D	Circle3D&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Negative circle radius in ResizeCircle3D_Delta.

ResizeSegment3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Lengthens or shortens a segment in 3D to a new length preserving its orientation and center point.

Syntax

```
void avl::ResizeSegment3D
(
 const avl::Segment3D& inSegment3D,
 float inNewLength,
 float inAnchor,
 avl::Segment3D& outSegment3D
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inSegment3D	const Segment3D&			
➡	inNewLength	float	0.0 - ∞	1.0f	
➡	inAnchor	float	-∞ - ∞	0.5f	Anchor point, 0 means the beginning, 1 means the end of the segment
⬅	outSegment3D	Segment3D&			

ResizeSegment3D_Delta

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Changes length of a segment in 3D by adding a value preserving its orientation and center point.

Syntax

```
void avl::ResizeSegment3D_Delta
(
 const avl::Segment3D& inSegment3D,
 float inDelta,
 float inAnchor,
 avl::Segment3D& outSegment3D
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inSegment3D	const Segment3D&			
➡	inDelta	float		0.0f	Value added to segment length
➡	inAnchor	float	-∞ - ∞	0.5f	Anchor point, 0 means the beginning, 1 means the end of the segment
⬅	outSegment3D	Segment3D&			

Errors

List of possible exceptions:

Error type	Description
DomainError	Negative segment length in ResizeSegment3D_Delta.

ResizeVector3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Lengthens or shortens a 3D vector preserving its direction.

Syntax

```
void avl::ResizeVector3D
(
 const avl::Vector3D& inVector3D,
 float inNewLength,
 avl::Vector3D& outVector3D
)
```

Parameters

	Name	Type	Range	Default	Description
→	inVector3D	const Vector3D&			
→	inNewLength	float	0.0 - ∞	1.0f	
←	outVector3D	Vector3D&			

ResizeVector3D_Delta

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Extends length of a 3D vector by adding a value preserving its direction.

Syntax

```
void avl::ResizeVector3D_Delta
(
 const avl::Vector3D& inVector3D,
 float inDelta,
 avl::Vector3D& outVector3D
)
```

Parameters

	Name	Type	Default	Description
→	inVector3D	const Vector3D&		
→	inDelta	float	0.0f	Value added to vector length
←	outVector3D	Vector3D&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Negative vector length in ResizeVector3D_Delta.

ReverseSegment3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Swaps the two endpoints of a segment in 3D.

Syntax

```
void avl::ReverseSegment3D
(
 const avl::Segment3D& inSegment3D,
 avl::Segment3D& outSegment3D
)
```

Parameters

	Name	Type	Default	Description
→	inSegment3D	const Segment3D&		
←	outSegment3D	Segment3D&		

RotateCircle3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Rotates a circle in 3D around an axis in 3D.

Syntax

```
void avl::RotateCircle3D
(
 const avl::Circle3D& inCircle3D,
 const avl::Line3D& inAxis,
 float inAngle,
 bool inInverse,
 avl::Circle3D& outCircle3D
)
```

Parameters

	Name	Type	Default	Description
→	inCircle3D	const Circle3D&		
→	inAxis	const Line3D&		The rotation axis
→	inAngle	float		Clockwise angle of rotation
→	inInverse	bool		Switches to the inverse operation
←	outCircle3D	Circle3D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inCircle3D** and **outCircle3D**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite circle on input in RotateCircle3D.
DomainError	Indefinite line on input in RotateCircle3D.

RotateCircle3D_AroundSegment

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Rotates a circle in 3D around a segment (the axis) in 3D.

Syntax

```
void avl::RotateCircle3D_AroundSegment
(
 const avl::Circle3D& inCircle3D,
 const avl::Segment3D& inAxisSegment,
 float inAngle,
 bool inInverse,
 avl::Circle3D& outCircle3D
)
```

Parameters

	Name	Type	Default	Description
→	inCircle3D	const Circle3D&		
→	inAxisSegment	const Segment3D&		The rotation axis
→	inAngle	float		Clockwise angle of rotation
→	inInverse	bool		Switches to the inverse operation
←	outCircle3D	Circle3D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inCircle3D** and **outCircle3D**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty segment on input in RotateCircle3D_AroundSegment.
DomainError	Indefinite circle on input in RotateCircle3D_AroundSegment.

RotateLine3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Rotates a line in 3D around an axis in 3D.

Syntax

```
void avl::RotateLine3D
(
 const avl::Line3D& inLine3D,
 const avl::Line3D& inAxis,
 float inAngle,
 bool inInverse,
 avl::Line3D& outLine3D
)
```

Parameters

	Name	Type	Default	Description
→	inLine3D	const Line3D&		
→	inAxis	const Line3D&		The rotation axis
→	inAngle	float		Clockwise angle of rotation
→	inInverse	bool		Switches to the inverse operation
←	outLine3D	Line3D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inLine3D** and **outLine3D**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite line on input in RotateLine3D.

RotateLine3D_AroundSegment

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Rotates a line in 3D around a segment (the axis) in 3D.

Syntax

```
void avl::RotateLine3D_AroundSegment
(
 const avl::Line3D& inLine3D,
 const avl::Segment3D& inAxisSegment,
 float inAngle,
 bool inInverse,
 avl::Line3D& outLine3D
)
```

Parameters

	Name	Type	Default	Description
→	inLine3D	const Line3D&		
→	inAxisSegment	const Segment3D&		The rotation axis
→	inAngle	float		Clockwise angle of rotation
→	inInverse	bool		Switches to the inverse operation
←	outLine3D	Line3D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inLine3D** and **outLine3D**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty segment on input in RotateLine3D_AroundSegment.
DomainError	Indefinite line on input in RotateLine3D_AroundSegment.

RotatePlane

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Rotates a plane in 3D around an axis in 3D.

Syntax

```
void avl::RotatePlane
(
 const avl::Plane3D& inPlane,
 const avl::Line3D& inAxis,
 float inAngle,
 bool inInverse,
 avl::Plane3D& outPlane
)
```

Parameters

	Name	Type	Default	Description
→	inPlane	const Plane3D&		
→	inAxis	const Line3D&		The rotation axis
→	inAngle	float		Clockwise angle of rotation
→	inInverse	bool		Switches to the inverse operation
←	outPlane	Plane3D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inPlane** and **outPlane**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite line on input in RotatePlane.
DomainError	Indefinite plane on input in RotatePlane.

RotatePlane_AroundSegment

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Rotates a plane in 3D around a segment (the axis) in 3D.

Syntax

```
void avl::RotatePlane_AroundSegment
(
 const avl::Plane3D& inPlane,
 const avl::Segment3D& inAxisSegment,
 float inAngle,
 bool inInverse,
 avl::Plane3D& outPlane
)
```

Parameters

	Name	Type	Default	Description
→	inPlane	const Plane3D&		
→	inAxisSegment	const Segment3D&		The rotation axis
→	inAngle	float		Clockwise angle of rotation
→	inInverse	bool		Switches to the inverse operation
←	outPlane	Plane3D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inPlane** and **outPlane**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty segment on input in RotatePlane_AroundSegment.
DomainError	Indefinite plane on input in RotatePlane_AroundSegment.

RotatePoint3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Rotates a 3D point around an axis in 3D.

Syntax

```
void avl::RotatePoint3D
(
 const avl::Point3D& inPoint3D,
 const avl::Line3D& inAxis,
 float inAngle,
 bool inInverse,
 avl::Point3D& outPoint3D,
 atl::Optional<avl::Matrix&> outTransform = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPoint3D	const Point3D&		
→	inAxis	const Line3D&		The rotation axis
→	inAngle	float		Clockwise angle of rotation
→	inInverse	bool		Switches to the inverse operation
←	outPoint3D	Point3D&		
←	outTransform	Optional<Matrix&>	NIL	

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inPoint3D** and **outPoint3D**

Read more about In-place Computation.

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outTransform**.

Read more about Optional Outputs.

RotatePoint3DArray

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Rotates an array of 3D points around an axis in 3D.

Syntax

```
void avl::RotatePoint3DArray
(
 const atl::Array<avl::Point3D>& inPoints3D,
 const avl::Line3D& inAxis,
 float inAngle,
 bool inInverse,
 atl::Array<avl::Point3D>& outPoints3D,
 atl::Optional<avl::Matrix&> outTransform = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
➡	inPoints3D	const Array<Point3D>&		
➡	inAxis	const Line3D&		The rotation axis
➡	inAngle	float		Clockwise angle of rotation
➡	inInverse	bool		Switches to the inverse operation
⬅	outPoints3D	Array<Point3D>&		
⬅	outTransform	Optional<Matrix&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outTransform**.

Read more about Optional Outputs.

Hardware Acceleration

This operation is optimized for SSE41 technology.

This operation is optimized for AVX2 technology.

This operation is optimized for NEON technology.

This operation supports automatic parallelization for multicore and multiprocessor systems.

RotatePoint3DArray_AroundSegment

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Rotates an array of 3D points around a segment (the axis).

Syntax

```
void avl::RotatePoint3DArray_AroundSegment
(
 const atl::Array<avl::Point3D>& inPoints3D,
 const avl::Segment3D& inAxisSegment,
 float inAngle,
 bool inInverse,
 atl::Array<avl::Point3D>& outPoints3D,
 atl::Optional<avl::Matrix&> outTransform = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPoints3D	const Array<Point3D>&		
→	inAxisSegment	const Segment3D&		The rotation axis
→	inAngle	float		Clockwise angle of rotation
→	inInverse	bool		Switches to the inverse operation
←	outPoints3D	Array<Point3D>&		
←	outTransform	Optional<Matrix&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outTransform**.

Read more about Optional Outputs.

Hardware Acceleration

This operation is optimized for SSE41 technology.

This operation is optimized for AVX2 technology.

This operation is optimized for NEON technology.

This operation supports automatic parallelization for multicore and multiprocessor systems.

RotatePoint3D_AroundSegment

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Rotates a 3D point around a segment (the axis).

Syntax

```
void avl::RotatePoint3D_AroundSegment
(
 const avl::Point3D& inPoint3D,
 const avl::Segment3D& inAxisSegment,
 float inAngle,
 bool inInverse,
 avl::Point3D& outPoint3D,
 atl::Optional<avl::Matrix&> outTransform = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPoint3D	const Point3D&		
→	inAxisSegment	const Segment3D&		The rotation axis
→	inAngle	float		Clockwise angle of rotation
→	inInverse	bool		Switches to the inverse operation
←	outPoint3D	Point3D&		
←	outTransform	Optional<Matrix&>	NIL	

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inPoint3D** and **outPoint3D**

Read more about In-place Computation.

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outTransform**.

Read more about Optional Outputs.

RotateSegment3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Rotates a segment in 3D around an axis in 3D.

Syntax

```
void avl::RotateSegment3D
(
 const avl::Segment3D& inSegment3D,
 const avl::Line3D& inAxis,
 float inAngle,
 bool inInverse,
 avl::Segment3D& outSegment3D
)
```

Parameters

	Name	Type	Default	Description
→	inSegment3D	const Segment3D&		
→	inAxis	const Line3D&		The rotation axis
→	inAngle	float		Clockwise angle of rotation
→	inInverse	bool		Switches to the inverse operation
←	outSegment3D	Segment3D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inSegment3D** and **outSegment3D**

Read more about In-place Computation.

RotateSegment3D_AroundSegment

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Rotates a segment in 3D around a segment (the axis) in 3D.

Syntax

```
void avl::RotateSegment3D_AroundSegment
(
 const avl::Segment3D& inSegment3D,
 const avl::Segment3D& inAxisSegment,
 float inAngle,
 bool inInverse,
 avl::Segment3D& outSegment3D
)
```

Parameters

	Name	Type	Default	Description
→	inSegment3D	const Segment3D&		
→	inAxisSegment	const Segment3D&		The rotation axis
→	inAngle	float		Clockwise angle of rotation
→	inInverse	bool		Switches to the inverse operation
←	outSegment3D	Segment3D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inSegment3D** and **outSegment3D**

Read more about In-place Computation.

RotateVector3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Rotates a 3D vector around another 3D vector (the axis).

Syntax

```
void avl::RotateVector3D
(
 const avl::Vector3D& inVector3D,
 const avl::Vector3D& inAxisVector,
 float inAngle,
 bool inInverse,
 avl::Vector3D& outVector3D
)
```

Parameters

	Name	Type	Default	Description
→	inVector3D	const Vector3D&		
→	inAxisVector	const Vector3D&		Vector to rotate around
→	inAngle	float		Clockwise angle of rotation
→	inInverse	bool		Switches to the inverse operation
←	outVector3D	Vector3D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inVector3D** and **outVector3D**, **inAxisVector** and **outVector3D**

Read more about In-place Computation.

Segment2DToSegment3D_AxesPlane

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Projects a 2D segment onto a plane defined by 3D coordinate axes.

Syntax

```
void avl::Segment2DToSegment3D_AxesPlane
(
 const avl::Segment2D& inSegment2D,
 avl::PlaneType::Type inPlaneType,
 float inCoordinateValue,
 avl::Segment3D& outSegment3D,
 atl::Optional<avl::Plane3D&> outProjectionPlane = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inSegment2D	const Segment2D&		
→	inPlaneType	PlaneType::Type		
→	inCoordinateValue	float		Coordinate of the chosen axes plane
←	outSegment3D	Segment3D&		
←	outProjectionPlane	Optional<Plane3D&>	NIL	Plane the input is projected onto

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outProjectionPlane**.

Read more about Optional Outputs.

Segment3DToSegment2D_AxesPlane

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Projects a 3D segment onto a plane defined by coordinate axes.

Syntax

```
void avl::Segment3DToSegment2D_AxesPlane
(
 const avl::Segment3D& inSegment3D,
 avl::PlaneType::Type inPlaneType,
 avl::Segment2D& outSegment2D,
 atl::Optional<avl::Plane3D&> outProjectionPlane = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inSegment3D	const Segment3D&		
→	inPlaneType	PlaneType::Type		
←	outSegment2D	Segment2D&		
←	outProjectionPlane	Optional<Plane3D&>	NIL	Plane the input is projected onto

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outProjectionPlane**.

Read more about Optional Outputs.

Segments2DToSegments3D_AxesPlane

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Projects an array of 2D segments onto a plane defined by 3D coordinate axes.

Syntax

```
void avl::Segments2DToSegments3D_AxesPlane
(
 const atl::Array<avl::Segment2D>& inSegments2D,
 avl::PlaneType::Type inPlaneType,
 float inCoordinateValue,
 atl::Array<avl::Segment3D>& outSegments3D,
 atl::Optional<avl::Plane3D&> outProjectionPlane = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inSegments2D	const Array<Segment2D>&		
→	inPlaneType	PlaneType::Type		
→	inCoordinateValue	float		Coordinate of the chosen axes plane
←	outSegments3D	Array<Segment3D>&		
←	outProjectionPlane	Optional<Plane3D&>	NIL	Plane the input is projected onto

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outProjectionPlane**.

Read more about Optional Outputs.

Segments3DToSegments2D_AxesPlane

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Projects an array of 3D segments onto a plane defined by coordinate axes.

Syntax

```
void avl::Segments3DToSegments2D_AxesPlane
(
 const atl::Array<avl::Segment3D>& inSegments3D,
 avl::PlaneType::Type inPlaneType,
 atl::Array<avl::Segment2D>& outSegments2D,
 atl::Optional<avl::Plane3D&> outProjectionPlane = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inSegments3D	const Array<Segment3D>&		
→	inPlaneType	PlaneType::Type		
←	outSegments2D	Array<Segment2D>&		
←	outProjectionPlane	Optional<Plane3D&>	NIL	Plane the input is projected onto

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outProjectionPlane**.

Read more about Optional Outputs.

SplitPointsByPlane

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Separates the points being on one side of the input plane from the others.

Syntax

```
void avl::SplitPointsByPlane
(
 const atl::Array<avl::Point3D>& inPoints,
 const avl::Plane3D& inPlane,
 atl::Array<avl::Point3D>& outPoints1,
 atl::Array<avl::Point3D>& outPoints2
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point3D>&		Input points
→	inPlane	const Plane3D&		Plane used for splitting
←	outPoints1	Array<Point3D>&		Points with positive signed distance to the input plane
←	outPoints2	Array<Point3D>&		Points with negative signed distance to the input plane

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite plane on input in SplitPointsByPlane.

TransformCircle3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Applies a general transformation expressed by a matrix to a 3D circle.

Syntax

```
void avl::TransformCircle3D
(
 const avl::Circle3D& inCircle3D,
 const avl::Matrix& inTransform,
 bool inInverse,
 avl::Circle3D& outCircle3D
)
```

Parameters

	Name	Type	Default	Description
→	inCircle3D	const Circle3D&		
→	inTransform	const Matrix&		Transformation 3x3 or 4x4 matrix
→	inInverse	bool		Switches to the inverse operation
←	outCircle3D	Circle3D&		

TransformLine3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Applies a general transformation expressed by a matrix to a 3D line.

Syntax

```
void avl::TransformLine3D
(
 const avl::Line3D& inLine3D,
 const avl::Matrix& inTransform,
 bool inInverse,
 avl::Line3D& outLine3D
)
```

Parameters

	Name	Type	Default	Description
→	inLine3D	const Line3D&		
→	inTransform	const Matrix&		Transformation 3x3 or 4x4 matrix
→	inInverse	bool		Switches to the inverse operation
←	outLine3D	Line3D&		

TransformPlane

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Applies a general transformation expressed by a matrix to a plane.

Syntax

```
void avl::TransformPlane
(
 const avl::Plane3D& inPlane,
 const avl::Matrix& inTransform,
 bool inInverse,
 avl::Plane3D& outPlane
)
```

Parameters

	Name	Type	Default	Description
→	inPlane	const Plane3D&		
→	inTransform	const Matrix&		Transformation 3x3 or 4x4 matrix
→	inInverse	bool		Switches to the inverse operation
←	outPlane	Plane3D&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Singular matrix not allowed in TransformPlane.
DomainError	Transformation matrix size must be 3x3 or 4x4 in TransformPlane.

TransformPoint3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Applies a general transformation expressed by a matrix to a 3D point.

Syntax

```
void avl::TransformPoint3D
(
 const avl::Point3D& inPoint3D,
 const avl::Matrix& inTransform,
 bool inInverse,
 avl::Point3D& outPoint3D
)
```

Parameters

	Name	Type	Default	Description
→	inPoint3D	const Point3D&		
→	inTransform	const Matrix&		Transformation 3x3 or 4x4 matrix
→	inInverse	bool		Switches to the inverse operation
←	outPoint3D	Point3D&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Transformation matrix size must be 3x3 or 4x4 in 3D point transform.

TransformPoint3DArray

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Applies a general transformation expressed by a matrix to an array of 3D points.

Syntax

```
void avl::TransformPoint3DArray
(
 const atl::Array<avl::Point3D>& inPoints3D,
 const avl::Matrix& inTransform,
 bool inInverse,
 atl::Array<avl::Point3D>& outPoints3D
)
```

Parameters

	Name	Type	Default	Description
→	inPoints3D	const Array<Point3D>&		
→	inTransform	const Matrix&		Transformation 3x3 or 4x4 matrix
→	inInverse	bool		Switches to the inverse operation
←	outPoints3D	Array<Point3D>&		

Hardware Acceleration

This operation is optimized for SSE41 technology.

This operation is optimized for AVX2 technology.

This operation is optimized for NEON technology.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Transformation matrix size must be 3x3 or 4x4 in 3D point transform.

TransformSegment3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Applies a general transformation expressed by a matrix to a 3D segment.

Syntax

```
void avl::TransformSegment3D
(
 const avl::Segment3D& inSegment3D,
 const avl::Matrix& inTransform,
 bool inInverse,
 avl::Segment3D& outSegment3D
)
```

Parameters

	Name	Type	Default	Description
→	inSegment3D	const Segment3D&		
→	inTransform	const Matrix&		Transformation 3x3 or 4x4 matrix
→	inInverse	bool		Switches to the inverse operation
←	outSegment3D	Segment3D&		

TransformVector3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Applies a general transformation expressed by a matrix to a 3D vector.

Syntax

```
void avl::TransformVector3D
(
 const avl::Vector3D& inVector3D,
 const avl::Matrix& inTransform,
 bool inInverse,
 avl::Vector3D& outVector3D
)
```

Parameters

	Name	Type	Default	Description
→	inVector3D	const Vector3D&		
→	inTransform	const Matrix&		Transformation 3x3 or 4x4 matrix
→	inInverse	bool		Switches to the inverse operation
←	outVector3D	Vector3D&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Singular matrix not allowed for inverse transform in TransformVector3D.
DomainError	Transformation matrix size must be 3x3 or 4x4 in TransformVector3D.

TranslateBox3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Shifts a box in 3D by a vector.

Syntax

```
void avl::TranslateBox3D
(
 const avl::Box3D& inBox3D,
 const avl::Vector3D& inDelta,
 bool inInverse,
 avl::Box3D& outBox3D
)
```

Parameters

	Name	Type	Default	Description
→	inBox3D	const Box3D&		Input box in 3D
→	inDelta	const Vector3D&		Translation vector
→	inInverse	bool		Switches to the inverse operation
←	outBox3D	Box3D&		Shifted box

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inBox3D** and **outBox3D**

Read more about In-place Computation.

TranslateCircle3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Shifts a circle in 3D by a vector.

Syntax

```
void avl::TranslateCircle3D
(
 const avl::Circle3D& inCircle3D,
 const avl::Vector3D& inDelta,
 bool inInverse,
 avl::Circle3D& outCircle3D
)
```

Parameters

	Name	Type	Default	Description
→	inCircle3D	const Circle3D&		Input circle
→	inDelta	const Vector3D&		Translation vector
→	inInverse	bool		Switches to the inverse operation
←	outCircle3D	Circle3D&		Shifted circle

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inCircle3D** and **outCircle3D**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite circle on input in TranslateCircle3D.

TranslateLine3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Shifts a line in 3D by a vector.

Syntax

```
void avl::TranslateLine3D
(
 const avl::Line3D& inLine3D,
 const avl::Vector3D& inDelta,
 bool inInverse,
 avl::Line3D& outLine3D
)
```

Parameters

	Name	Type	Default	Description
→	inLine3D	const Line3D&		Input line in 3D
→	inDelta	const Vector3D&		Translation vector
→	inInverse	bool		Switches to the inverse operation
←	outLine3D	Line3D&		Shifted line

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inLine3D** and **outLine3D**

Read more about In-place Computation.

TranslatePlane

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Shifts a plane in 3D by a vector.

Syntax

```
void avl::TranslatePlane
(
 const avl::Plane3D& inPlane,
 const avl::Vector3D& inDelta,
 bool inInverse,
 avl::Plane3D& outPlane
)
```

Parameters

	Name	Type	Default	Description
→	inPlane	const Plane3D&		Input plane
→	inDelta	const Vector3D&		Translation vector
→	inInverse	bool		Switches to the inverse operation
←	outPlane	Plane3D&		Shifted plane

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inPlane** and **outPlane**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite plane on input in TranslatePlane.

TranslatePoint3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Translates a point by a vector.

Syntax

```
void avl::TranslatePoint3D
(
 const avl::Point3D& inPoint3D,
 const avl::Vector3D& inDelta,
 bool inInverse,
 avl::Point3D& outPoint3D
)
```

Parameters

	Name	Type	Default	Description
→	inPoint3D	const Point3D&		
→	inDelta	const Vector3D&		Translation vector
→	inInverse	bool		Switches to the inverse operation
←	outPoint3D	Point3D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inPoint3D** and **outPoint3D**

Read more about In-place Computation.

TranslatePoint3DArray

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Translates an array of points by a vector.

Syntax

```
void avl::TranslatePoint3DArray
(
 const atl::Array<avl::Point3D>& inPoints3D,
 const avl::Vector3D& inDelta,
 bool inInverse,
 atl::Array<avl::Point3D>& outPoints3D
)
```

Parameters

	Name	Type	Default	Description
→	inPoints3D	const Array<Point3D>&		
→	inDelta	const Vector3D&		Translation vector
→	inInverse	bool		Switches to the inverse operation
←	outPoints3D	Array<Point3D>&		

Hardware Acceleration

This operation is optimized for SSE2 technology.

This operation is optimized for AVX2 technology.

This operation is optimized for NEON technology.

TranslatePoint3D_Toward

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Translates a point in 3D towards another point in 3D by a specified distance.

Syntax

```
void avl::TranslatePoint3D_Toward
(
 const avl::Point3D& inPoint3D,
 const avl::Point3D& inTargetPoint,
 float inDistance,
 bool inInverse,
 avl::Point3D& outPoint3D
)
```

Parameters

	Name	Type	Default	Description
→	inPoint3D	const Point3D&		
→	inTargetPoint	const Point3D&		Defines the direction of the translation
→	inDistance	float		The distance between inPoint3D and outPoint3D
→	inInverse	bool		Switches to the inverse operation
←	outPoint3D	Point3D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inTargetPoint** and **outPoint3D**, **inPoint3D** and **outPoint3D**

Read more about In-place Computation.

TranslateSegment3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Shifts a segment in 3D by a vector.

Syntax

```
void avl::TranslateSegment3D
(
 const avl::Segment3D& inSegment3D,
 const avl::Vector3D& inDelta,
 bool inInverse,
 avl::Segment3D& outSegment3D
)
```

Parameters

	Name	Type	Default	Description
→	inSegment3D	const Segment3D&		Input segment in 3D
→	inDelta	const Vector3D&		Translation vector
→	inInverse	bool		Switches to the inverse operation
←	outSegment3D	Segment3D&		Shifted segment

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inSegment3D** and **outSegment3D**

Read more about In-place Computation.

TrimLine3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Creates a segment contained in a box in 3D from a line in 3D.

Syntax

```
void avl::TrimLine3D
(
 const avl::Line3D& inLine3D,
 const avl::Box3D& inBox3D,
 atl::Conditional<avl::Segment3D>& outSegment3D
)
```

Parameters

	Name	Type	Default	Description
➡	inLine3D	const Line3D&		
➡	inBox3D	const Box3D&		
⬅	outSegment3D	Conditional<Segment3D>&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite line on input in TrimLine3D.

TrimPoint3DArray

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Removes points not contained in a box in 3D from an array.

Syntax

```
void avl::TrimPoint3DArray
(
 const atl::Array<avl::Point3D>& inPoints3D,
 const avl::Box3D& inBox3D,
 atl::Array<avl::Point3D>& outPoints3D
)
```

Parameters

	Name	Type	Default	Description
➡	inPoints3D	const Array<Point3D>&		
➡	inBox3D	const Box3D&		
⬅	outPoints3D	Array<Point3D>&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inPoints3D** and **outPoints3D**

Read more about In-place Computation.

TrimSegment3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Creates a segment contained in a box in 3D from another segment in 3D.

Syntax

```
void avl::TrimSegment3D
(
 const avl::Segment3D& inSegment3D,
 const avl::Box3D& inBox3D,
 atl::Conditional<avl::Segment3D>& outSegment3D
)
```

Parameters

	Name	Type	Default	Description
→	inSegment3D	const Segment3D&		
→	inBox3D	const Box3D&		
←	outSegment3D	Conditional<Segment3D>&		

Vector2DToVector3D_AxesPlane

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Projects a 2D vector onto a plane defined by 3D coordinate axes.

Syntax

```
void avl::Vector2DToVector3D_AxesPlane
(
 const avl::Vector2D& inVector2D,
 avl::PlaneType::Type inPlaneType,
 float inCoordinateValue,
 avl::Vector3D& outVector3D,
 atl::Optional<avl::Plane3D&> outProjectionPlane = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inVector2D	const Vector2D&		
→	inPlaneType	PlaneType::Type		
→	inCoordinateValue	float		Coordinate of the chosen axes plane
←	outVector3D	Vector3D&		
←	outProjectionPlane	Optional<Plane3D&>	NIL	Plane the input is projected onto

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outProjectionPlane**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect plane type.

Vector3DToVector2D_AxesPlane

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Projects a 3D vector onto a plane defined by coordinate axes.

Syntax

```
void avl::Vector3DToVector2D_AxesPlane
(
 const avl::Vector3D& inVector3D,
 avl::PlaneType::Type inPlaneType,
 avl::Vector2D& outVector2D,
 atl::Optional<avl::Plane3D&> outProjectionPlane = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inVector3D	const Vector3D&		
→	inPlaneType	PlaneType::Type		
←	outVector2D	Vector2D&		
←	outProjectionPlane	Optional<Plane3D&>	NIL	Plane the input is projected onto

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outProjectionPlane**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect plane type.

23. Point3DGrid Spatial Transforms

Table of content:

AlignPoint3DGridToPlane

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Rotates a grid of 3D points with a rotation that transforms the input plane to be parallel to XY plane.

Syntax

```
void avl::AlignPoint3DGridToPlane
(
 const avl::Point3DGrid& inGrid,
 const avl::Plane3D& inPlane,
 avl::Point3DGrid& outGrid,
 atl::Optional<avl::Matrix&> outTransform = atl::NIL,
 atl::Optional<avl::Matrix&> outInvertedTransform = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inGrid	const Point3DGrid&		
→	inPlane	const Plane3D&		The plane that determines rotation transform
←	outGrid	Point3DGrid&		
←	outTransform	Optional<Matrix&>	NIL	Matrix of the transform used to align the input grid
←	outInvertedTransform	Optional<Matrix&>	NIL	Matrix of the transform rotating the output grid to the input one

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outTransform**, **outInvertedTransform**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite plane on input in AlignPoint3DGridToPlane.

CropPoint3DGrid

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Removes from the grid points that are not contained in a given rectangular box.

Syntax

```
void avl::CropPoint3DGrid
(
 const avl::Point3DGrid& inPoint3DGrid,
 const avl::ValueLimits& inXLimits,
 const avl::ValueLimits& inYLimits,
 const avl::ValueLimits& inZLimits,
 avl::Point3DGrid& outPoint3DGrid
)
```

Parameters

	Name	Type	Default	Description
→	inPoint3DGrid	const Point3DGrid&		
→	inXLimits	const ValueLimits&		
→	inYLimits	const ValueLimits&		
→	inZLimits	const ValueLimits&		
←	outPoint3DGrid	Point3DGrid&		

CropPoint3DGridByNeighborsProximity

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Removes from the grid points that are too distant from their neighbor points.

Syntax

```
void avl::CropPoint3DGridByNeighborsProximity
(
 const avl::Point3DGrid& inPoint3DGrid,
 const int inNeighborRadius,
 const float inMaxDistance,
 avl::Metric3D::Type inMetric,
 const float inMinNeighborRatio,
 avl::Point3DGrid& outPoint3DGrid
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoint3DGrid	const Point3DGrid&			
→	inNeighborRadius	const int	1 - ∞	1	Radius of neighbors to search for real neighbors
→	inMaxDistance	const float	0.0 - ∞	2.0f	Maximal distance from another point to consider them real neighbors
→	inMetric	Metric3D::Type		Z	Metric used for measuring distance between points
→	inMinNeighborRatio	const float	0.0 - 1.0	1.0f	Fraction of valid neighbors in a given radius that have to be real neighbors
←	outPoint3DGrid	Point3DGrid&			

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect metric in CropPoint3DGridByNeighborsProximity.

CropPoint3DGridByPlaneProximity

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Removes from the grid points that are too distant from a given plane.

Syntax

```
void avl::CropPoint3DGridByPlaneProximity
(
 const avl::Point3DGrid& inPoint3DGrid,
 const avl::Plane3D& inPlane,
 const float inMaxDistance,
 avl::Point3DGrid& outPoint3DGrid
)
```

Parameters

	Name	Type	Default	Description
→	inPoint3DGrid	const Point3DGrid&		
→	inPlane	const Plane3D&		
→	inMaxDistance	const float		Maximal distance from a given plane
←	outPoint3DGrid	Point3DGrid&		

CropPoint3DGridToRegion

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Removes points that are not represented in a given region of the input grid.

Syntax

```
void avl::CropPoint3DGridToRegion
(
 const avl::Point3DGrid& inPoint3DGrid,
 const avl::Region& inRegion,
 avl::Point3DGrid& outPoint3DGrid
)
```

Parameters

	Name	Type	Default	Description
➡	inPoint3DGrid	const Point3DGrid&		
➡	inRegion	const Region&		Input region
⬅	outPoint3DGrid	Point3DGrid&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Input region exceeds Point3DGrid dimensions in CropPoint3DGridToRegion.

JoinPoint3DGrids

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Combines two point grids into one.

Syntax

```
void avl::JoinPoint3DGrids
(
 const avl::Point3DGrid& inPoint3DGrid1,
 const avl::Point3DGrid& inPoint3DGrid2,
 atl::Optional<int> inDeltaX,
 atl::Optional<int> inDeltaY,
 avl::Point3DGrid& outPoint3DGrid
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inPoint3DGrid1	const Point3DGrid&			
➡	inPoint3DGrid2	const Point3DGrid&			
➡	inDeltaX	Optional<int>	0 - ∞	NIL	First index offset for second grid; if set to Nil, first grid's width is chosen
➡	inDeltaY	Optional<int>	0 - ∞	0	Second index offset for second grid; if set to Nil, first grid's height is chosen
⬅	outPoint3DGrid	Point3DGrid&			

Description

The operation adds points from the second input grid to the first one, increasing the grid's dimensions if necessary. The **inDeltaX** and **inDeltaY** parameters determine the offsets added to indices of the second grid. The second grid's points will never overwrite points from the first grid, so if both grids have same dimensions and both **inDeltaX** and **inDeltaY** equal 0, no point from the second grid will be added. If **inDeltaX** is set to Nil, the width of the first input grid is chosen. Similarly, if **inDeltaY** is set to Nil, the height of the first input grid is chosen as the offset.

If the intention is to keep all points from both input grids, one of the offsets should be set to Nil.

MirrorPoint3DGrid

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Mirrors each grid point according to the input plane.

Syntax

```
void avl::MirrorPoint3DGrid
(
 const avl::Point3DGrid& inGrid,
 const avl::Plane3D& inPlane,
 avl::Point3DGrid& outGrid,
 atl::Optional<avl::Matrix&> outTransform = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inGrid	const Point3DGrid&		
→	inPlane	const Plane3D&		The plane being a mirror
←	outGrid	Point3DGrid&		
←	outTransform	Optional<Matrix&>	NIL	Matrix of the transform used to mirror the input grid

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outTransform**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite plane on input in MirrorPoint3DGrid.

RescalePoint3DGrid

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Changes the distances of grid points to a reference point.

Syntax

```
void avl::RescalePoint3DGrid
(
 const avl::Point3DGrid& inGrid,
 const avl::Point3D& inReferencePoint,
 float inScale,
 bool inInverse,
 avl::Point3DGrid& outGrid
)
```

Parameters

	Name	Type	Default	Description
→	inGrid	const Point3DGrid&		
→	inReferencePoint	const Point3D&		Point to which the distances will be changed
→	inScale	float	1.0f	Scaling factor
→	inInverse	bool		Switches to the inverse operation
←	outGrid	Point3DGrid&		

Hardware Acceleration

This operation is optimized for SSE41 technology.

This operation is optimized for AVX2 technology.

This operation is optimized for NEON technology.

Errors

List of possible exceptions:

Error type	Description
DomainError	Scale cannot be zero in an inverse rescaling of a point.

RotatePoint3DGrid

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Rotates a grid of 3D points around an axis in 3D.

Syntax

```
void avl::RotatePoint3DGrid
(
 const avl::Point3DGrid& inGrid,
 const avl::Line3D& inAxis,
 float inAngle,
 bool inInverse,
 avl::Point3DGrid& outGrid,
 atl::Optional<avl::Matrix&> outTransform = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inGrid	const Point3DGrid&		
→	inAxis	const Line3D&		The rotation axis
→	inAngle	float		Clockwise angle of rotation
→	inInverse	bool		Switches to the inverse operation
←	outGrid	Point3DGrid&		
←	outTransform	Optional<Matrix&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outTransform**.

Read more about Optional Outputs.

Hardware Acceleration

This operation is optimized for SSE41 technology.

This operation is optimized for AVX2 technology.

This operation is optimized for NEON technology.

This operation supports automatic parallelization for multicore and multiprocessor systems.

RotatePoint3DGrid_AroundSegment

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Rotates a grid of 3D points around an segment3D (the axis) in 3D.

Syntax

```
void avl::RotatePoint3DGrid_AroundSegment
(
 const avl::Point3DGrid& inGrid,
 const avl::Segment3D& inAxisSegment,
 float inAngle,
 bool inInverse,
 avl::Point3DGrid& outGrid,
 atl::Optional<avl::Matrix&> outTransform = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inGrid	const Point3DGrid&		
→	inAxisSegment	const Segment3D&		The rotation axis
→	inAngle	float		Angle of rotation
→	inInverse	bool		Switches to the inverse operation
←	outGrid	Point3DGrid&		
←	outTransform	Optional<Matrix&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outTransform**.

Read more about Optional Outputs.

Hardware Acceleration

This operation is optimized for SSE41 technology.

This operation is optimized for AVX2 technology.

This operation is optimized for NEON technology.

This operation supports automatic parallelization for multicore and multiprocessor systems.

SplitPoint3DGridByPlane

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Separates the points being on one side of the input plane from the others.

Syntax

```
void avl::SplitPoint3DGridByPlane
(
 const avl::Point3DGrid& inPoint3DGrid,
 atl::Optional<const avl::Region&> inRoi,
 const avl::Plane3D& inPlane,
 avl::Point3DGrid& outPoint3DGrid1,
 avl::Point3DGrid& outPoint3DGrid2
)
```

Parameters

	Name	Type	Default	Description
→	inPoint3DGrid	const Point3DGrid&		
→	inRoi	Optional<const Region&>	NIL	Region of interest
→	inPlane	const Plane3D&		Plane used for splitting
←	outPoint3DGrid1	Point3DGrid&		Grid of points with positive signed distance to the input plane
←	outPoint3DGrid2	Point3DGrid&		Grid of points with negative signed distance to the input plane

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite plane on input in SplitPoint3DGridByPlane.

TransformPoint3DGrid

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Applies a general transformation expressed by a matrix to a grid of 3D points.

Syntax

```
void avl::TransformPoint3DGrid
(
 const avl::Point3DGrid& inGrid,
 const avl::Matrix& inTransform,
 bool inInverse,
 avl::Point3DGrid& outGrid
)
```

Parameters

	Name	Type	Default	Description
→	inGrid	const Point3DGrid&		
→	inTransform	const Matrix&		Transformation 3x3 or 4x4 matrix
→	inInverse	bool		Switches to the inverse operation
←	outGrid	Point3DGrid&		

Hardware Acceleration

This operation is optimized for SSE41 technology.

This operation is optimized for AVX2 technology.

This operation is optimized for NEON technology.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Transformation matrix size must be 3x3 or 4x4 in 3D point transform.

TranslatePoint3DGrid

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Translates a grid of points by a vector.

Syntax

```
void avl::TranslatePoint3DGrid
(
 const avl::Point3DGrid& inGrid,
 const avl::Vector3D& inDelta,
 bool inInverse,
 avl::Point3DGrid& outGrid
)
```

Parameters

	Name	Type	Default	Description
→	inGrid	const Point3DGrid&		
→	inDelta	const Vector3D&		Translation vector
→	inInverse	bool		Switches to the inverse operation
←	outGrid	Point3DGrid&		

Hardware Acceleration

This operation is optimized for SSE41 technology.

This operation is optimized for AVX2 technology.

This operation is optimized for NEON technology.

24. Region Spatial Transforms

Table of content:

AlignRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Aligns a region to a coordinate system.

Applications: Most typically used, when there is a region extracted after performing CropImageToRectangle. This filter makes it possible to align the region with the original image.

Syntax

```
void avl::AlignRegion
(
 const avl::Region& inRegion,
 const avl::CoordinateSystem2D& inAlignment,
 bool inInverse,
 atl::Optional<int> inFrameWidth,
 atl::Optional<int> inFrameHeight,
 avl::Region& outAlignedRegion
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inRegion	const Region&			Input region
➡	inAlignment	const CoordinateSystem2D&			Coordinate system to align to
➡	inInverse	bool			Switches to the inverse transform
➡	inFrameWidth	Optional<int>	0 - 65535	NIL	Output region's frame width
➡	inFrameHeight	Optional<int>	0 - 65535	NIL	Output region's frame height
⬅	outAlignedRegion	Region&			

In-place Processing

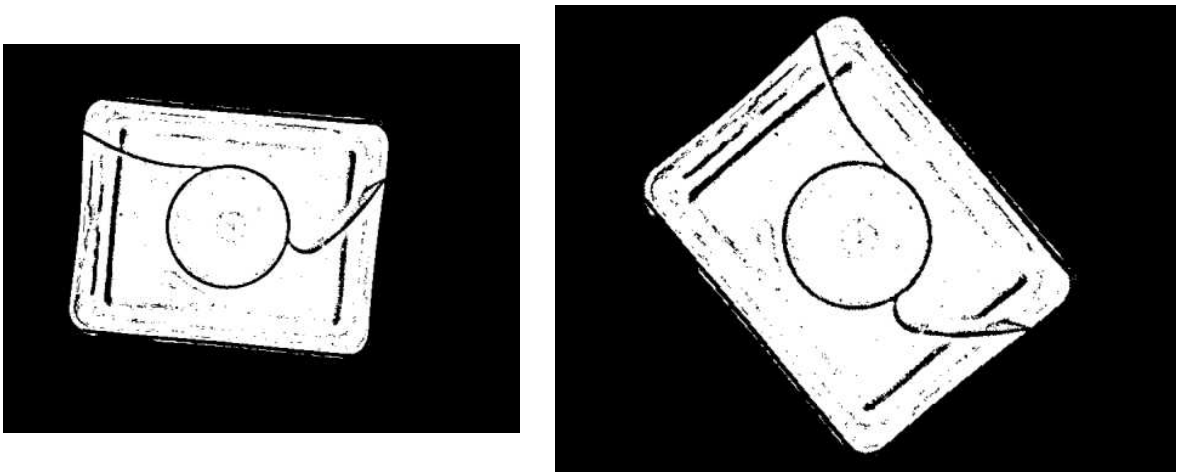
This function supports in-place data processing - you can pass the same reference to **inRegion** and **outAlignedRegion**

Read more about In-place Computation.

Description

AlignRegion applies **inAlignment** transform to an input region.

Examples



AlignRegion *performed on the sample region,* ***inAlignment.Origin*** *= (280, -120),* ***inAlignment.Angle*** *= 45,* ***inAlignment.Scale*** *= 1.0 and* ***inInverse*** *= False. The* ***inAlignment.Origin*** *is drawn on the first image in blue.*

Remarks

Region is a pixel-precise object, so geometrical transform may cause its deformation. Consider using a closed Path instead, which can be converted to a region in the last step.

See Also

- MirrorRegion – Mirrors a region across vertical or horizontal axis.
- TransposeRegion – Flips and rotates a region so that x-coordinates are exchanged with y-coordinates.
- RotateRegion – Rotates a region around a specified point.

CropRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a region from a rectangular fragment of another one.

Applications: Can be used to obtain a part of a bigger region, but also to enforce specific region frame.

Syntax

```
void avl::CropRegion
(
 const avl::Region& inRegion,
 const avl::Box& inSelection,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inSelection	const Box&		Box defining the range of cropping
←	outRegion	Region&		Output region

Examples

***CropRegion** performed on the sample region with **inSelection** = Box(0,0,200,200).*

See Also

- SetRegionFrame – Changes the width and the height of a region's frame (but does not rescale the content).
- UncropRegion – Inverse of CropRegion.

CropRegionToQuadrangle

Header: AVL.h
Namespace: avl
Module: FoundationPro

Creates a region from a quadrangular fragment of another one. The quadrangle must be convex.

Syntax

```
void avl::CropRegionToQuadrangle
(
 const avl::Region& inRegion,
 const avl::Path& inQuadrangle,
 atl::Optional<const avl::Size&> inOutputSize,
 atl::Optional<const avl::CoordinateSystem2D&> inQuadrangleAlignment,
 avl::Region& outRegion,
 atl::Optional<avl::Path&> outAlignedShape = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inQuadrangle	const Path&		A convex quadrangle defining a region to be cropped (points must be clockwise)
→	inOutputSize	Optional<const Size&>	NIL	Width and height of the output region
→	inQuadrangleAlignment	Optional<const CoordinateSystem2D&>	NIL	Adjusts the quadrangle to the position of the inspected object
←	outRegion	Region&		Output region
←	outAlignedShape	Optional<Path&>	NIL	The input quadrangle after transformation (in the region coordinates)

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inRegion** and **outRegion**

Read more about In-place Computation.

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedShape**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Path given as an argument to CropRegionToQuadrangle is not closed.
DomainError	Path given as an argument to CropRegionToQuadrangle should have exactly 4 points.

CropRegionToRectangle

Header: AVL.h
Namespace: avl
Module: FoundationPro

Creates a region from a rectangular fragment of another one.

Applications: Can be used to obtain a part of a bigger region, but also to enforce specific region frame.

Syntax

```
void avl::CropRegionToRectangle
(
 const avl::Region& inRegion,
 const avl::Rectangle2D& inRectangle,
 atl::Optional<const avl::CoordinateSystem2D&> inRectangleAlignment,
 avl::Region& outRegion,
 atl::Optional<avl::Rectangle2D&> outAlignedRectangle = atl::NIL,
 atl::Optional<avl::CoordinateSystem2D&> outOutputAlignment = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inRectangle	const Rectangle2D&		Rectangle defining a rotated subregion
→	inRectangleAlignment	Optional<const CoordinateSystem2D&>	NIL	Adjusts the rectangle to the position of the inspected object
←	outRegion	Region&		Output region
←	outAlignedRectangle	Optional<Rectangle2D&>	NIL	Input rectangle after transformation (in the region coordinates)
←	outOutputAlignment	Optional<CoordinateSystem2D&>	NIL	Alignment of the output region

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inRegion** and **outRegion**

Read more about In-place Computation.

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedRectangle**, **outOutputAlignment**.

Read more about Optional Outputs.

DownsampleRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Shrinks a region by the factor of two along each axis.

Syntax

```
void avl::DownsampleRegion
(
 const avl::Region& inRegion,
 avl::DownsampleRegionMode::Type inDownsampleRegionMode,
 const int inScaleStep,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inRegion	const Region&			Input region
→	inDownsampleRegionMode	DownsampleRegionMode::Type			
→	inScaleStep	const int	0 - 12	1	Defines how many times the region size is divided by 2
←	outRegion	Region&			Output region

See Also

- ShrinkRegionNTimes – Shrinks a region by a natural factor along each axis.
- ResizeRegion – Enlarges or shrinks a region to new dimensions.
- ResizeRegion_Relative – Resizes region relatively along each axis.
- EnlargeRegionNTimes – Enlarges a region by a natural factor.
- DownsampleImage – Shrinks an image by the factor of two along each axis.

EnlargeRegionNTimes

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Enlarges a region by a natural factor.

Syntax

```
void avl::EnlargeRegionNTimes
(
 const avl::Region& inRegion,
 int inN,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inRegion	const Region&			Input region
➡	inN	int	1 - ∞	2	The scaling coefficient
⬅	outRegion	Region&			Output region

Examples

EnlargeRegionNTimes *performed on the sample region.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Input and output regions are not distinct in EnlargeRegionNTimes.
DomainError	Output region too big in EnlargeRegionNTimes.

MirrorRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Mirrors a region across vertical or horizontal axis.

Syntax

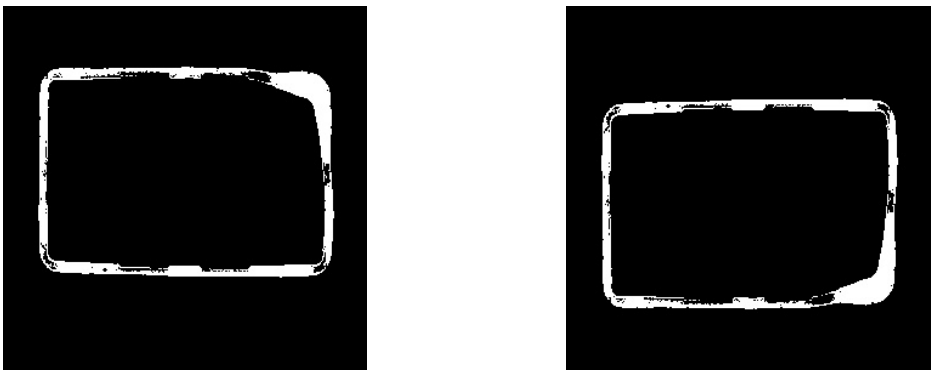
```
void avl::MirrorRegion
(
 const avl::Region& inRegion,
 const avl::MirrorDirection::Type inMirrorDirection,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inMrrorDirection	const MrrorDirection::Type	Horizontal	Reverse the order of region columns (horizontal direction) or rows (vertical direction).
←	outRegion	Region&		Output region

Examples

***MirrorRegion** with **inMirrorDirection** = Horizontal.*



***MirrorRegion** with **inMirrorDirection** = Vertical.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Input and output regions are not distinct in MrrorRegion.

See Also

- TransposeRegion – Flips and rotates a region so that x-coordinates are exchanged with y-coordinates.

ReflectRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Reflects a region through the given location.

Syntax

```
void avl::ReflectRegion
(
 const avl::Region& inRegion,
 const avl::Location& inReflectionCenter,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Default	Description
➡	inRegion	const Region&		Input region
➡	inReflectionCenter	const Location&		
⬅	outRegion	Region&		Output region

Examples

ReflectRegion *performed on the sample region.*

See Also

- MirrorRegion – Mirrors a region across vertical or horizontal axis.
- TransposeRegion – Flips and rotates a region so that x-coordinates are exchanged with y-coordinates.

ResizeRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Enlarges or shrinks a region to new dimensions.

Syntax

```
void avl::ResizeRegion
(
 const avl::Region& inRegion,
 atl::Optional<int> inNewWidth,
 atl::Optional<int> inNewHeight,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inRegion	const Region&			Input region
→	inNewWidth	Optional<int>	1 - 65535	NIL	
→	inNewHeight	Optional<int>	1 - 65535	NIL	
←	outRegion	Region&			Output region

Examples

ResizeRegion *performed on the sample region with* ***inNewWidth*** *= 200,* ***inNewHeight*** *= 100.*

See Also

- ResizeRegion_Relative – Resizes region relatively along each axis.
- DownsampleRegion – Shrinks a region by the factor of two along each axis.
- ShrinkRegionNTimes – Shrinks a region by a natural factor along each axis.
- EnlargeRegionNTimes – Enlarges a region by a natural factor.
- ResizeImage – Enlarges or shrinks an image to new dimensions.

ResizeRegion_Relative

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Resizes region relatively along each axis.

Syntax

```
void avl::ResizeRegion_Relative
(
 const avl::Region& inRegion,
 float inHorizontalScale,
 float inVerticalScale,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inRegion	const Region&			Input region
→	inHorizontalScale	float	0.0 - ∞	1.0f	
→	inVerticalScale	float	0.0 - ∞	1.0f	
←	outRegion	Region&			Output region

Examples

ResizeRegion_Relative *performed on the sample region with* ***inHorizontalScale*** *and* ***inVerticalScale*** *both set to 0,5.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Output region too big in ResizeRegion_Relative.

See Also

- ResizeRegion – Enlarges or shrinks a region to new dimensions.
- DownsampleRegion – Shrinks a region by the factor of two along each axis.
- ShrinkRegionNTimes – Shrinks a region by a natural factor along each axis.
- EnlargeRegionNTimes – Enlarges a region by a natural factor.
- ResizeImage – Enlarges or shrinks an image to new dimensions.

RotateRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Rotates a region around a specified point.

Syntax

```
void avl::RotateRegion
(
 const avl::Region& inRegion,
 atl::Optional<const avl::Point2D&> inCenter,
 float inAngle,
 const bool inInverse,
 avl::RotationSizeMode::Type inSizeMode,
 atl::Optional<int> inFrameWidth,
 atl::Optional<int> inFrameHeight,
 avl::Region& outRegion,
 atl::Optional<avl::CoordinateSystem2D&> outOutputAlignment = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inRegion	const Region&			Input region
→	inCenter	Optional<const Point2D&>		NIL	Center of rotation
→	inAngle	float			Clockwise rotation angle
→	inInverse	const bool			Switches to counter-clockwise rotation
→	inSizeMode	RotationSizeMode::Type		Preserve	Determines whether to extent the region size to fit the rotated region.
→	inFrameWidth	Optional<int>	0 - 65535	NIL	Output region frame width, ignored when inSizeMode is set to RotationSizeMode::Fit.
→	inFrameHeight	Optional<int>	0 - 65535	NIL	Output region frame height, ignored when inSizeMode is set to RotationSizeMode::Fit.
←	outRegion	Region&			Output region
←	outOutputAlignment	Optional<CoordinateSystem2D&>		NIL	Alignment of the output region

Optional Outputs

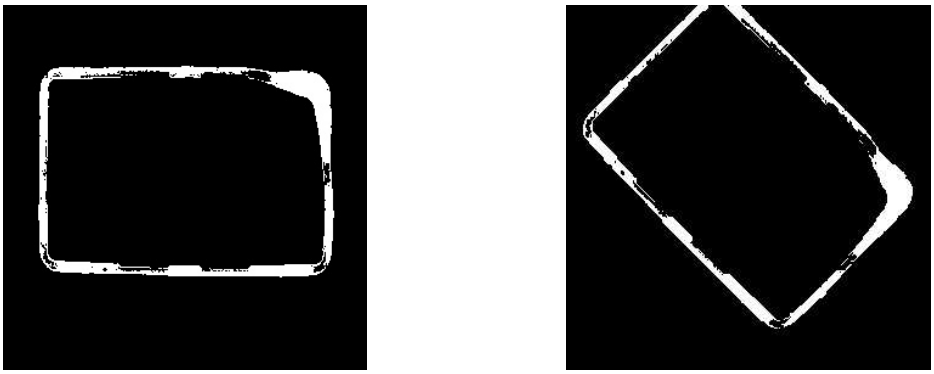
The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outOutputAlignment**.

Read more about Optional Outputs.

Description

The operation rotates a region by the **inAngle** degrees around **inCenter** point. If the **inCenter** is not provided, the rotation is conducted around the mass center of the region.

Examples



RotateRegion *performed on the sample region with* ***inAngle*** *set to 45.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty region on input when rotation center is Nil in RotateRegion.

See Also

- TranslateRegion – Translates a region by a given number of pixels along each axis.

ShearRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes a leant region.

Syntax

```
void avl::ShearRegion
(
 const avl::Region& inRegion,
 float inShear,
 avl::Axis::Type inAxis,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inRegion	const Region&			Input region
→	inShear	float	$-\infty$ $-\infty$		The relative shift of each consecutive row or column
→	inAxis	Axis::Type			Switches between shifting rows or columns
←	outRegion	Region&			Output region

Description

The filter **ShearRegion** applies basic affine transform to each regions's pixel.

Shear affine transform when **X axis** is selected:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & inShear \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Shear affine transform when **Y axis** is selected:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ inShear & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

See Also

- ShearImage – Computes a leant image (shifts the rows).

ShrinkRegionNTimes

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Shrinks a region by a natural factor along each axis.

Syntax

```
void avl::ShrinkRegionNTimes
(
 const avl::Region& inRegion,
 int inNX,
 const atl::Optional<int>& inNY,
 float inThreshold,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inRegion	const Region&			Input region
→	inNX	int	1 - ∞	2	
→	inNY	const Optional<int>&	1 - ∞	NIL	
→	inThreshold	float	0.0 - 1.0	0.5f	How much of the input pixels must be present to produce output pixel. When 0 - any pixel, when 1 - all pixels.
←	outRegion	Region&			Output region

See Also

- DownsampleRegion – Shrinks a region by the factor of two along each axis.
- ResizeRegion – Enlarges or shrinks a region to new dimensions.
- ResizeRegion_Relative – Resizes region relatively along each axis.
- EnlargeRegionNTimes – Enlarges a region by a natural factor.
- ShrinkImageNTimes – Shrinks an image by a natural factor along each axis.
- ShrinkProfileNTimes – Reduces the length of a profile N-times by averaging each N consecutive elements.

TranslateRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Translates a region by a given number of pixels along each axis.

Syntax

```
void avl::TranslateRegion
(
 const avl::Region& inRegion,
 int inDeltaX,
 int inDeltaY,
 bool inInverse,
 atl::Optional<int> inFrameWidth,
 atl::Optional<int> inFrameHeight,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inRegion	const Region&			Input region
→	inDeltaX	int			Horizontal shift
→	inDeltaY	int			Vertical shift
→	inInverse	bool			Negates the delta values
→	inFrameWidth	Optional<int>	0 - 65535	NIL	Output region frame width
→	inFrameHeight	Optional<int>	0 - 65535	NIL	Output region frame height
←	outRegion	Region&			Output region

Examples

TranslateRegion *performed on the sample region with* ***inDeltaX*** *and* ***inDeltaY*** *both set to 50.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Input and output regions are not distinct in TranslateRegion.

See Also

- RotateRegion – Rotates a region around a specified point.

TransposeRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Flips and rotates a region so that x-coordinates are exchanged with y-coordinates.

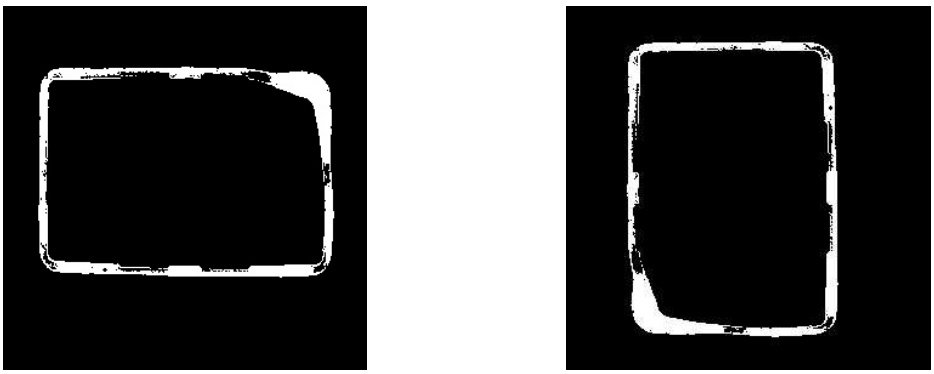
Syntax

```
void avl::TransposeRegion
(
 const avl::Region& inRegion,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outRegion	Region&		Output region

Examples



TransposeRegion *performed on the sample region.*

See Also

- MirrorRegion – Mirrors a region across vertical or horizontal axis.

TrimRegionToPolygon

Header: AVL.h
Namespace: avl
Module: FoundationPro

Removes region elements lying beyond the specified polygon.

Syntax

```
void avl::TrimRegionToPolygon
(
 const avl::Region& inRegion,
 const avl::Path& inPolygon,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inPolygon	const Path&		
←	outRegion	Region&		Output region

Errors

List of possible exceptions:

Error type	Description
DomainError	Path is not closed in TrimRegionToPolygon.

TrimRegionToRectangle

Header: AVL.h
Namespace: avl
Module: FoundationPro

Limits a region to a rectangular area.

Syntax

```
void avl::TrimRegionToRectangle
(
 const avl::Region& inRegion,
 const avl::Rectangle2D& inRectangle,
 atl::Optional<const avl::CoordinateSystem2D&> inRectangleAlignment,
 avl::Region& outRegion,
 atl::Optional<avl::Rectangle2D&> outAlignedRectangle = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inRectangle	const Rectangle2D&		Trimming rectangle
→	inRectangleAlignment	Optional<const CoordinateSystem2D&>	NIL	Adjusts the rectangle to the position of the inspected object
←	outRegion	Region&		Output region
←	outAlignedRectangle	Optional<Rectangle2D&>	NIL	Input rectangle after transformation (in the region coordinates)

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedRectangle**.

Read more about Optional Outputs.

UncropRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Inverse of CropRegion.

Syntax

```
void avl::UncropRegion
(
 const avl::Region& inRegion,
 const avl::Box& inSelection,
 int inWidth,
 int inHeight,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inRegion	const Region&			Input region
→	inSelection	const Box&			Box defining the range of the original cropping
→	inWidth	int	0 - 65535		Width of the uncropped region
→	inHeight	int	0 - 65535		Height of the uncropped region
←	outRegion	Region&			Output region

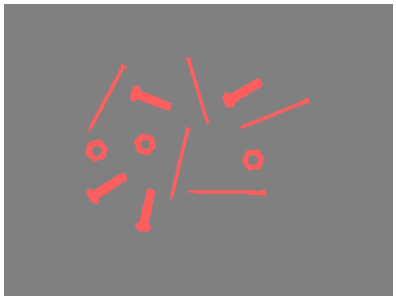
Description

The operation translates a region to original coordinates. The value of **inSelection** describes region coordinates before the cropping.

Dimensions of **outRegion** depends on **inWidth** and **inHeight**.

Examples

Images below show a typical usage of the filter **CropRegion**.



An input region.

A region after CropRegion.

Result of applying **UncropRegion** on the cropped region.

Remarks

This operation is an inversion of filter CropRegion.

See Also

- CropRegion – Creates a region from a rectangular fragment of another one.

25. Surface Basics

Table of content:

AlignRegionToSurfaceCoordinatesFormat

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Returns region covering the same area for the surface with another coordinates format.

Applications: Transformation of regions defined on the image associated with the surface to image associated with another surface.

Syntax

```
void avl::AlignRegionToSurfaceCoordinatesFormat
(
 const avl::Region& inRegion,
 const avl::SurfaceCoordinatesFormat& inInputSurfaceCoordinatesFormat,
 const avl::SurfaceCoordinatesFormat& inOutputSurfaceCoordinatesFormat,
 atl::Optional<int> inFrameWidth,
 atl::Optional<int> inFrameHeight,
 avl::Region& outAlignedRegion
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inInputSurfaceCoordinatesFormat	const SurfaceCoordinatesFormat&		Offsets and scales on X and Y axes of the surface associated with the image the input region were defined on
→	inOutputSurfaceCoordinatesFormat	const SurfaceCoordinatesFormat&		Offsets and scales on X and Y axes of the surface associated with the image the output region is defined on
→	inFrameWidth	Optional<int>	NIL	Output region's frame width
→	inFrameHeight	Optional<int>	NIL	Output region's frame height
←	outAlignedRegion	Region&		

AlignRegionToSurfaceFormat

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Returns region covering the same area for the surface with another format.

Applications: Transformation of regions defined on the image associated with the surface to image associated with another surface.

Syntax

```
void avl::AlignRegionToSurfaceFormat
(
 const avl::Region& inRegion,
 const avl::SurfaceCoordinatesFormat& inInputSurfaceCoordinatesFormat,
 const avl::SurfaceFormat& inOutputSurfaceFormat,
 avl::Region& outAlignedRegion
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inInputSurfaceCoordinatesFormat	const SurfaceCoordinatesFormat&		Offsets and scales on X and Y axes of the surface associated with the image the input region were defined on
→	inOutputSurfaceFormat	const SurfaceFormat&		Format of the surface associated with the image the output region is defined on
←	outAlignedRegion	Region&		

ArrangePoint3DArray

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Creates a surface structure from Point3D array taking into account X and Y coordinates.

Syntax

```
void avl::ArrangePoint3DArray
(
 const atl::Array<avl::Point3D>& inPoints,
 const avl::ValueLimits_f64& inXLimits,
 const double inXScale,
 const avl::ValueLimits_f64& inYLimits,
 const double inYScale,
 const avl::ValueLimits_f64& inZLimits,
 const double inZOffset,
 const double inZScale,
 avl::PlainType::Type inPointType,
 avl::SurfaceMultipointHeight::Type inMultipointHeight,
 avl::Surface& outSurface,
 atl::Optional<double&> outMinX = atl::NIL,
 atl::Optional<double&> outMinY = atl::NIL,
 avl::Region& diagSurfaceValidPointsRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoints	const Array<Point3D>&			
→	inXLimits	const ValueLimits_f64&			
→	inXScale	const double	0.000001 - ∞	1.0D	
→	inYLimits	const ValueLimits_f64&			
→	inYScale	const double	0.000001 - ∞	1.0D	
→	inZLimits	const ValueLimits_f64&			
→	inZOffset	const double			
→	inZScale	const double	0.000001 - ∞	1.0D	
→	inPointType	PlainType::Type		Int16	Type of single point Z coordinate
→	inMultipointHeight	SurfaceMultipointHeight::Type		Mean	Determines the Z coordinate of a surface point created from more than one point
←	outSurface	Surface&			
←	outMinX	Optional<double&>		NIL	
←	outMinY	Optional<double&>		NIL	
🔍	diagSurfaceValidPointsRegion	Region&			Region of locations where the surface points are valid

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outMinX**, **outMinY**.

Read more about Optional Outputs.

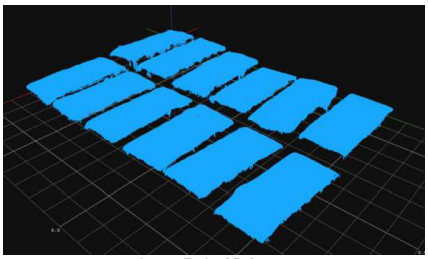
Description

The operation creates a surface object basing on the input points in 3D. The X and Y coordinates of the output surface object points are very regular, so the whole output object has neatly organized structure. Internally, the XY plane is divided into rectangular tiles with dimensions equal to **inXScale** and **inYScale**. Each tile will represent one output surface point. The point is computed as an average of all input points that are located in the corresponding tile. If none of the input points is present in a tile, the output point for such a tile is indefinite and set to the point in infinity.

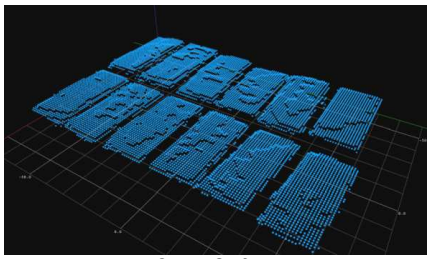
Hints

- The filter can be used to reduce the size of the input data. The higher the **inXScale** and **inYScale** values are, the smaller output surface size is.

Examples



Input Point3DArray.



Output Surface.

ArrangePoint3DArray performed on a sample surface.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty point array in ArrangePoint3DArray.

See Also

- ArrangePoint3DGrid – Creates a surface structure from Point3DGrid taking into account X and Y coordinates.

ArrangePoint3DGrid

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Creates a surface structure from Point3DGrid taking into account X and Y coordinates.

Syntax

```
void avl::ArrangePoint3DGrid
(
 const avl::Point3DGrid& inPoint3DGrid,
 const avl::ValueLimits_f64& inXLimits,
 const double inXScale,
 const avl::ValueLimits_f64& inYLimits,
 const double inYScale,
 const avl::ValueLimits_f64& inZLimits,
 const double inZOffset,
 const double inZScale,
 avl::PlainType::Type inPointType,
 avl::SurfaceMultipointHeight::Type inMultipointHeight,
 avl::Surface& outSurface,
 atl::Optional<double&> outMinX = atl::NIL,
 atl::Optional<double&> outMinY = atl::NIL,
 avl::Region& diagSurfaceValidPointsRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoint3DGrid	const Point3DGrid&			
→	inXLimits	const ValueLimits_f64&			
→	inXScale	const double	0.000001 - ∞	1.0D	
→	inYLimits	const ValueLimits_f64&			
→	inYScale	const double	0.000001 - ∞	1.0D	
→	inZLimits	const ValueLimits_f64&			
→	inZOffset	const double			
→	inZScale	const double	0.000001 - ∞	1.0D	
→	inPointType	PlainType::Type		Int16	Type of single point Z coordinate
→	inMultipointHeight	SurfaceMultipointHeight::Type		Mean	Determines the Z coordinate of a surface point created from more than one point
←	outSurface	Surface&			
←	outMinX	Optional<double&>		NIL	
←	outMinY	Optional<double&>		NIL	
🔍	diagSurfaceValidPointsRegion	Region&			Region of locations where the surface points are valid

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outMinX**, **outMinY**.

Read more about Optional Outputs.

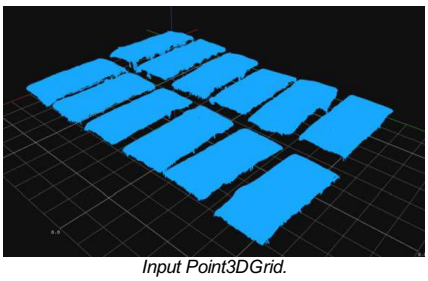
Description

The operation creates a surface object basing on the input point 3D grid. The X and Y coordinates of the output surface object points are very regular, so the whole output object has neatly organized structure. Internally, the XY plane is divided into rectangular tiles with dimensions equal to **inXScale** and **inYScale**. Each tile will represent one output surface point. The point is computed as an average of all points from the input grid that are located in the corresponding tile. If none of the input grid points is present in a tile, the output point for such a tile is indefinite and set to the point in infinity.

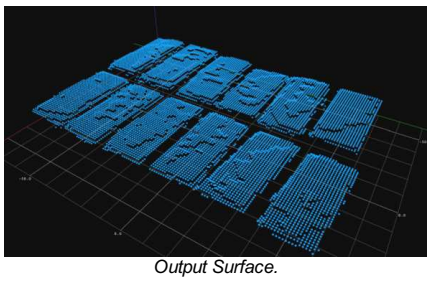
Hints

- The filter can be used to reduce the size of the input data. The higher the **inXScale** and **inYScale** values are, the smaller output surface size is.

Examples



Input Point3DGrid.



Output Surface.

ArrangePoint3DGrid performed on a sample point grid.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty point grid in ArrangePoint3DGrid.

See Also

- ArrangePoint3DArray – Creates a surface structure from Point3D array taking into account X and Y coordinates.
- CreateSurfaceFromImage – Creates a Surface structure from coordinates encoded in pixels of an image.

ConvertCoordinateSystem2DTo3D

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Converts a coordinate system connected with the surface image to a coordinate system connected with the surface.

Syntax

```
void avl::ConvertCoordinateSystem2DTo3D
(
 const avl::CoordinateSystem2D& inCoordinateSystem,
 const avl::SurfaceFormat& inSurfaceFormat,
 avl::CoordinateSystem2D& outCoordinateSystem
)
```

Parameters

	Name	Type	Default	Description
→	inCoordinateSystem	const CoordinateSystem2D&		Coordinate system connected with the surface image
→	inSurfaceFormat	const SurfaceFormat&		Format of the surface
←	outCoordinateSystem	CoordinateSystem2D&		Coordinate system connected with the surface

ConvertSurfaceType

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Changes the point Z coordinate type.

Syntax

```
void avl::ConvertSurfaceType
(
 const avl::Surface& inSurface,
 avl::PlainType::Type inNewPointType,
 atl::Optional<double> inNewZOffset,
 atl::Optional<double> inNewZScale,
 avl::Surface& outSurface
)
```

Parameters

	Name	Type	Default	Description
→	inSurface	const Surface&		Input surface
→	inNewPointType	PlainType::Type		Point Z coordinate type of the output surface
→	inNewZOffset	Optional<double>	NIL	Offset along Z axis in output surface
→	inNewZScale	Optional<double>	NIL	Scale along Z axis in output surface
←	outSurface	Surface&		Surface with changed internal surface type

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

CreateFlatSurface

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Creates a uniform surface.

Syntax

```
void avl::CreateFlatSurface
(
 const int inWidth,
 const int inHeight,
 const avl::PlainType::Type& inPointType,
 const double inXOffset,
 const double inXScale,
 const double inYOffset,
 const double inYScale,
 const double inZOffset,
 const double inZScale,
 atl::Optional<double> inZ,
 avl::Surface& outSurface
)
```

Parameters

	Name	Type	Range	Default	Description
→	inWidth	const int	1 - 65535	320	Width of the created surface
→	inHeight	const int	1 - 65535	240	Height of the created surface
→	inPointType	const PlainType::Type&		Int16	Type of single point Z coordinate
→	inXOffset	const double		0.0D	
→	inXScale	const double	0.001 - ∞	1.0D	
→	inYOffset	const double		0.0D	
→	inYScale	const double	0.001 - ∞	1.0D	
→	inZOffset	const double		0.0D	
→	inZScale	const double	0.001 - ∞	1.0D	
→	inZ	Optional<double>		0.0D	Z coordinate of all points of the created surface
←	outSurface	Surface&			

CreateSurfaceFromImage

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Creates a Surface structure from coordinates encoded in pixels of an image.

Applications: Creating a Surface structure out of an image obtained from a 3D camera or other external sources that encodes 3D surface as pixel components of 2D image.

Syntax

```
void avl::CreateSurfaceFromImage
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 const avl::PointCloudCoordinateTransform& inXCoordinateTransform,
 const avl::PointCloudCoordinateTransform& inYCoordinateTransform,
 const avl::PointCloudCoordinateTransform& inZCoordinateTransform,
 const avl::OutputSurfaceFormat& inCreatedSurfaceFormat,
 avl::Surface& outSurface,
 avl::Region& diagSurfaceValidPointsRegion,
 avl::Point3DGrid& diagPoint3DGrid
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Source image with per pixel encoded XYZ coordinates
→	inRoi	Optional<const Region&>	NIL	Region determining valid points in resulting point grid
→	inXCoordinateTransform	const PointCloudCoordinateTransform&	PointCloudCoordinateTransform (Offset: 0.000000D ValueCoordinateTransform: ImageValueCoordinateTransform(ChannelIndex: 0 Scale: 1.0D InvalidValues: Nil) LocationCoordinateTransform: Nil Limits: ValueLimits_f64(MinValue: Nil, MaxValue: Nil))	Description of the creation of the X coordinate
→	inYCoordinateTransform	const PointCloudCoordinateTransform&	PointCloudCoordinateTransform (Offset: 0.000000D ValueCoordinateTransform: ImageValueCoordinateTransform(ChannelIndex: 1 Scale: 1.0D InvalidValues: Nil) LocationCoordinateTransform: Nil Limits: ValueLimits_f64(MinValue: Nil, MaxValue: Nil))	Description of the creation of the Y coordinate
→	inZCoordinateTransform	const PointCloudCoordinateTransform&	PointCloudCoordinateTransform (Offset: 0.000000D ValueCoordinateTransform: ImageValueCoordinateTransform(ChannelIndex: 2 Scale: 1.0D InvalidValues: Nil) LocationCoordinateTransform: Nil Limits: ValueLimits_f64(MinValue: Nil, MaxValue: Nil))	Description of the creation of the Z coordinate
→	inCreatedSurfaceFormat	const OutputSurfaceFormat&	OutputSurfaceFormat (XScale: 1.000000D YScale: 1.000000D ZScale: 1.000000D ZOffset: 0.000000D PointType: UInt16 MultipointHeight: Mean)	Parameters for arranging points into Surface
←	outSurface	Surface&		
🔍	diagSurfaceValidPointsRegion	Region&		Region of locations where the surface points are valid
🔍	diagPoint3DGrid	Point3DGrid&		Points decoded before arranging them into Surface

Description

The operation creates a point cloud object based on the input point image. It can be thought of as a combination of two operations: CreatePoint3DGridFromImage and ArrangePoint3DArray.

See Also

- ArrangePoint3DArray – Creates a surface structure from Point3D array taking into account X and Y coordinates.
- CreatePoint3DGridFromImage – Creates a Point3DGrid structure from coordinates encoded in pixels of image.

GetSurfaceElement_Interpolated

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Returns an interpolated single point of a surface given its surface grid coordinates.

Syntax

```
void avl::GetSurfaceElement_Interpolated
(
 const avl::Surface& inSurface,
 const avl::Point2D& inPoint,
 int inInterpolationRadius,
 avl::Point3D& outPoint
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inPoint	const Point2D&			Surface grid coordinates of the input point
→	inInterpolationRadius	int	0 - 65535	0	Radius of vicinity being taken into account to interpolate not existing point
←	outPoint	Point3D&			Output point

GetSurfacePath

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Returns a path consisting of interpolated single points of a surface given their surface grid coordinates.

Syntax

```
void avl::GetSurfacePath
(
 const avl::Surface& inSurface,
 const avl::Path& inPath,
 avl::Path& outPath
)
```

Parameters

	Name	Type	Default	Description
→	inSurface	const Surface&		Input surface
→	inPath	const Path&		Surface grid coordinates of the input path points
←	outPath	Path&		Output path in surface coordinates

ImageBoxToSurfaceCoordinates

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Finds the surface coordinates of image box.

Applications: Transformation of results found on the image associated with the surface to the surface coordinates.

Syntax

```
void avl::ImageBoxToSurfaceCoordinates
(
 const avl::Box& inImageBox,
 const avl::SurfaceCoordinatesFormat& inSurfaceCoordinatesFormat,
 avl::Box& outSurfaceBox,
 atl::Optional<avl::Box3D&> outSurfaceBox3D = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inImageBox	const Box&		
→	inSurfaceCoordinatesFormat	const SurfaceCoordinatesFormat&		Offsets and scales on X and Y axes of the surface associated with the image objects were found on
←	outSurfaceBox	Box&		
←	outSurfaceBox3D	Optional<Box3D&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outSurfaceBox3D**.

Read more about Optional Outputs.

ImageCircleToSurfaceCoordinates

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Finds the surface coordinates of image circle.

Applications: Transformation of results found on the image associated with the surface to the surface coordinates.

Syntax

```
void avl::ImageCircleToSurfaceCoordinates
(
 const avl::Circle2D& inImageCircle,
 const avl::SurfaceCoordinatesFormat& inSurfaceCoordinatesFormat,
 avl::Circle2D& outSurfaceCircle,
 atl::Optional<avl::Circle3D&> outSurfaceCircle3D = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inImageCircle	const Circle2D&		
→	inSurfaceCoordinatesFormat	const SurfaceCoordinatesFormat&		Offsets and scales on X and Y axes of the surface associated with the image objects were found on
←	outSurfaceCircle	Circle2D&		
←	outSurfaceCircle3D	Optional<Circle3D&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outSurfaceCircle3D**.

Read more about Optional Outputs.

ImageCoordinateSystemToSurfaceCoordinates

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Finds the surface coordinates of image coordinate system.

Applications: Transformation of results found on the image associated with the surface to the surface coordinates.

Syntax

```
void avl::ImageCoordinateSystemToSurfaceCoordinates
(
 const avl::CoordinateSystem2D& inImageCoordinateSystem,
 const avl::SurfaceCoordinatesFormat& inSurfaceCoordinatesFormat,
 avl::CoordinateSystem2D& outSurfaceCoordinateSystem
)
```

Parameters

	Name	Type	Default	Description
→	inImageCoordinateSystem	const CoordinateSystem2D&		
→	inSurfaceCoordinatesFormat	const SurfaceCoordinatesFormat&		Offsets and scales on X and Y axes of the surface associated with the image objects were found on
←	outSurfaceCoordinateSystem	CoordinateSystem2D&		

ImageLineToSurfaceCoordinates

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Finds the surface coordinates of image line.

Applications: Transformation of results found on the image associated with the surface to the surface coordinates.

Syntax

```
void avl::ImageLineToSurfaceCoordinates
(
 const avl::Line2D& inImageLine,
 const avl::SurfaceCoordinatesFormat& inSurfaceCoordinatesFormat,
 avl::Line2D& outSurfaceLine,
 atl::Optional<avl::Line3D&> outSurfaceLine3D = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inImageLine	const Line2D&		
→	inSurfaceCoordinatesFormat	const SurfaceCoordinatesFormat&		Offsets and scales on X and Y axes of the surface associated with the image objects were found on
←	outSurfaceLine	Line2D&		
←	outSurfaceLine3D	Optional<Line3D&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outSurfaceLine3D**.

Read more about Optional Outputs.

ImagePathsToSurfaceCoordinates

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Finds the surface coordinates of image paths.

Applications: Transformation of results found on the image associated with the surface to the surface coordinates.

Syntax

```
void avl::ImagePathsToSurfaceCoordinates
(
 const atl::Array<avl::Path>& inImagePaths,
 const avl::SurfaceCoordinatesFormat& inSurfaceCoordinatesFormat,
 atl::Array<avl::Path>& outSurfacePaths
)
```

Parameters

	Name	Type	Default	Description
→	inImagePaths	const Array<Path>&		
→	inSurfaceCoordinatesFormat	const SurfaceCoordinatesFormat&		Offsets and scales on X and Y axes of the surface associated with the image objects were found on
←	outSurfacePaths	Array<Path>&		

ImagePathToSurfaceCoordinates

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Finds the surface coordinates of image path.

Applications: Transformation of results found on the image associated with the surface to the surface coordinates.

Syntax

```
void avl::ImagePathToSurfaceCoordinates
(
 const avl::Path& inImagePath,
 const avl::SurfaceCoordinatesFormat& inSurfaceCoordinatesFormat,
 avl::Path& outSurfacePath
)
```

Parameters

	Name	Type	Default	Description
→	inImagePath	const Path&		
→	inSurfaceCoordinatesFormat	const SurfaceCoordinatesFormat&		Offsets and scales on X and Y axes of the surface associated with the image objects were found on
←	outSurfacePath	Path&		

ImagePointsToSurfaceCoordinates

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Finds the surface coordinates of image points.

Applications: Transformation of results found on the image associated with the surface to the surface coordinates.

Syntax

```
void avl::ImagePointsToSurfaceCoordinates
(
 const atl::Array<avl::Point2D>& inImagePoints,
 const avl::SurfaceCoordinatesFormat& inSurfaceCoordinatesFormat,
 atl::Array<avl::Point2D>& outSurfacePoints
)
```

Parameters

	Name	Type	Default	Description
➡	inImagePoints	const Array<Point2D>&		
➡	inSurfaceCoordinatesFormat	const SurfaceCoordinatesFormat&		Offsets and scales on X and Y axes of the surface associated with the image objects were found on
⬅	outSurfacePoints	Array<Point2D>&		

ImagePointToSurfaceCoordinates

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Finds the surface coordinates of image point.

Applications: Transformation of results found on the image associated with the surface to the surface coordinates.

Syntax

```
void avl::ImagePointToSurfaceCoordinates
(
 const avl::Point2D& inImagePoint,
 const avl::SurfaceCoordinatesFormat& inSurfaceCoordinatesFormat,
 avl::Point2D& outSurfacePoint,
 atl::Optional<avl::Point3D&> outSurfacePoint3D = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
➡	inImagePoint	const Point2D&		
➡	inSurfaceCoordinatesFormat	const SurfaceCoordinatesFormat&		Offsets and scales on X and Y axes of the surface associated with the image objects were found on
⬅	outSurfacePoint	Point2D&		
⬅	outSurfacePoint3D	Optional<Point3D&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outSurfacePoint3D**.

Read more about Optional Outputs.

ImageSegmentsToSurfaceCoordinates

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Finds the surface coordinates of image segments.

Applications: Transformation of results found on the image associated with the surface to the surface coordinates.

Syntax

```
void avl::ImageSegmentsToSurfaceCoordinates
(
 const atl::Array<avl::Segment2D>& inImageSegments,
 const avl::SurfaceCoordinatesFormat& inSurfaceCoordinatesFormat,
 atl::Array<avl::Segment2D>& outSurfaceSegments
)
```

Parameters

	Name	Type	Default	Description
→	inImageSegments	const Array<Segment2D>&		
→	inSurfaceCoordinatesFormat	const SurfaceCoordinatesFormat&		Offsets and scales on X and Y axes of the surface associated with the image objects were found on
←	outSurfaceSegments	Array<Segment2D>&		

ImageSegmentToSurfaceCoordinates

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Finds the surface coordinates of image segment.

Applications: Transformation of results found on the image associated with the surface to the surface coordinates.

Syntax

```
void avl::ImageSegmentToSurfaceCoordinates
(
 const avl::Segment2D& inImageSegment,
 const avl::SurfaceCoordinatesFormat& inSurfaceCoordinatesFormat,
 avl::Segment2D& outSurfaceSegment,
 atl::Optional<avl::Segment3D&> outSurfaceSegment3D = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inImageSegment	const Segment2D&		
→	inSurfaceCoordinatesFormat	const SurfaceCoordinatesFormat&		Offsets and scales on X and Y axes of the surface associated with the image objects were found on
←	outSurfaceSegment	Segment2D&		
←	outSurfaceSegment3D	Optional<Segment3D&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outSurfaceSegment3D**.

Read more about Optional Outputs.

ImageVectorToSurfaceCoordinates

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Finds the surface coordinates of image vector.

Applications: Transformation of results found on the image associated with the surface to the surface coordinates.

Syntax

```
void avl::ImageVectorToSurfaceCoordinates
(
 const avl::Vector2D& inImageVector,
 const avl::SurfaceCoordinatesFormat& inSurfaceCoordinatesFormat,
 avl::Vector2D& outSurfaceVector,
 atl::Optional<avl::Vector3D&> outSurfaceVector3D = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inImageVector	const Vector2D&		
→	inSurfaceCoordinatesFormat	const SurfaceCoordinatesFormat&		Offsets and scales on X and Y axes of the surface associated with the image objects were found on
←	outSurfaceVector	Vector2D&		
←	outSurfaceVector3D	Optional<Vector3D&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outSurfaceVector3D**.

Read more about Optional Outputs.

ProjectPathOntoSurface

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Returns a path consisting of interpolated single points of a surface given their coordinates in surface coordinate system.

Syntax

```
void avl::ProjectPathOntoSurface
(
 const avl::Surface& inSurface,
 const avl::Path& inPath,
 avl::Path& outSurfaceGridPath
)
```

Parameters

	Name	Type	Default	Description
→	inSurface	const Surface&		Input surface
→	inPath	const Path&		Coordinates of the input path points in surface coordinate system
←	outSurfaceGridPath	Path&		Output path in surface grid coordinate system

ProjectPointOntoSurface

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Returns an interpolated single point of a surface given its coordinates in surface coordinate system.

Syntax

```
void avl::ProjectPointOntoSurface
(
 const avl::Surface& inSurface,
 const avl::Point2D& inPoint,
 int inInterpolationRadius,
 atl::Optional<avl::Point3D&> outSurfaceRealPoint = atl::NIL,
 atl::Optional<avl::Point2D&> outSurfaceGridPoint = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inPoint	const Point2D&			Coordinates of the input point in surface coordinate system
→	inInterpolationRadius	int	0 - 65535	0	Radius of vicinity being taken into account to interpolate not existing point
←	outSurfaceRealPoint	Optional<Point3D&>		NIL	Output point in surface coordinate system
←	outSurfaceGridPoint	Optional<Point2D&>		NIL	Output point in surface grid coordinate system

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outSurfaceRealPoint**, **outSurfaceGridPoint**.

Read more about Optional Outputs.

ProjectSurfaceOntoAxesPlane

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Projects a surface onto one of three axes planes.

Syntax

```
void avl::ProjectSurfaceOntoAxesPlane
(
 const avl::Surface& inSurface,
 avl::PlaneType::Type inAxesPlaneType,
 atl::Optional<double> inOutputXScale,
 const avl::ValueLimits_f64& inOutputXLimits,
 atl::Optional<double> inOutputYScale,
 const avl::ValueLimits_f64& inOutputYLimits,
 const double inOutputZOffset,
 atl::Optional<double> inOutputZScale,
 const avl::ValueLimits_f64& inOutputZLimits,
 avl::PlainType::Type inPointType,
 avl::SurfaceMultipointHeight::Type inMultipointHeight,
 avl::Surface& outSurface,
 atl::Optional<double&> outMinOutputX = atl::NIL,
 atl::Optional<double&> outMinOutputY = atl::NIL,
 avl::Region& diagSurfaceValidPointsRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			
→	inAxesPlaneType	PlaneType::Type		XZ	
→	inOutputXScale	Optional<double>	0.000001 - ∞	1.0D	
→	inOutputXLimits	const ValueLimits_f64&			
→	inOutputYScale	Optional<double>	0.000001 - ∞	1.0D	
→	inOutputYLimits	const ValueLimits_f64&			
→	inOutputZOffset	const double			
→	inOutputZScale	Optional<double>	0.000001 - ∞	1.0D	
→	inOutputZLimits	const ValueLimits_f64&			
→	inPointType	PlainType::Type		Int16	Type of single point Z coordinate
→	inMultipointHeight	SurfaceMultipointHeight::Type		Mean	Determines the Z coordinate of a surface point created from more than one point
←	outSurface	Surface&			
←	outMinOutputX	Optional<double&>		NIL	
←	outMinOutputY	Optional<double&>		NIL	
🔍	diagSurfaceValidPointsRegion	Region&			Region of locations where the surface points are valid

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outMinOutputX**, **outMinOutputY**.

Read more about Optional Outputs.

ResamplePoint3DGrid

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Creates a surface structure from Point3DGrid taking into account X and Y coordinates and trying to preserve continuity of the surface.

Syntax

```
void avl::ResamplePoint3DGrid
(
 const avl::Point3DGrid& inPoint3DGrid,
 const avl::ValueLimits_f64& inXLimits,
 const avl::ValueLimits_f64& inYLimits,
 const double inZOffset,
 const double inZScale,
 avl::PlainType::Type inPointType,
 avl::SurfaceMultipointHeight::Type inMultipointHeight,
 avl::Surface& outSurface,
 atl::Optional<double&> outMinX = atl::NIL,
 atl::Optional<double&> outMinY = atl::NIL,
 avl::Region& diagSurfaceValidPointsRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoint3DGrid	const Point3DGrid&			
→	inXLimits	const ValueLimits_f64&			
→	inYLimits	const ValueLimits_f64&			
→	inZOffset	const double			
→	inZScale	const double	0.000001 - ∞	1.0D	
→	inPointType	PlainType::Type		Int16	Type of single point Z coordinate
→	inMultipointHeight	SurfaceMultipointHeight::Type		Mean	Determines the Z coordinate of a surface point created from more than one point
←	outSurface	Surface&			
←	outMinX	Optional<double&>		NIL	
←	outMinY	Optional<double&>		NIL	
🔍	diagSurfaceValidPointsRegion	Region&			Region of locations where the surface points are valid

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outMinX**, **outMinY**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty point array in ResamplePoint3DGrid.
DomainError	Incorrect coordinate limits in ResamplePoint3DGrid.
DomainError	Surface dimensions too big in ResamplePoint3DGrid.

SkipEmptySurface

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

If the input Surface has at least one point defined, then it is copied to the output; otherwise Nil is returned.

Applications: Secures against domain errors caused by empty surfaces, e.g. just before the FitPlaneToSurface filter is to be invoked.

Syntax

```
void avl::SkipEmptySurface
(
 const avl::Surface& inSurface,
 atl::Conditional<avl::Surface>& outNotEmptySurface,
 bool& outIsNotEmpty
)
```

Parameters

	Name	Type	Default	Description
→	inSurface	const Surface&		Input surface
←	outNotEmptySurface	Conditional<Surface>&		A copy of the input surface, if it is not empty, or Nil otherwise
←	outIsNotEmpty	bool&		Indication if the input surface is not empty

SkipEmptySurfaceRegion

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

If the input surface contains at least one valid point in a given region, then the region is copied to the output; otherwise Nil is returned.

Applications: Secures against domain errors caused by regions with no valid point, e.g. just before the SurfaceMassCenter filter is to be invoked.

Syntax

```
void avl::SkipEmptySurfaceRegion
(
 const avl::Surface& inSurface,
 const avl::Region& inRegion,
 atl::Conditional<avl::Region>& outNotEmptySurfaceRegion,
 bool& outIsNotEmpty
)
```

Parameters

	Name	Type	Default	Description
→	inSurface	const Surface&		
→	inRegion	const Region&		Input region
←	outNotEmptySurfaceRegion	Conditional<Region>&		
←	outIsNotEmpty	bool&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Input region frame exceeds the input surface frame in SkipEmptySurfaceRegion.

SurfaceBoxToImageCoordinates

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Finds the associated image coordinates of surface box.

Applications: Transformation of objects defined in the surface coordinate system to the associated image coordinate system.

Syntax

```
void avl::SurfaceBoxToImageCoordinates
(
 const avl::Box3D& inSurfaceBox,
 const avl::SurfaceCoordinatesFormat& inSurfaceCoordinatesFormat,
 avl::Box& outImageBox,
 atl::Optional<avl::Rectangle2D&> outImageRectangle = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inSurfaceBox	const Box3D&		
→	inSurfaceCoordinatesFormat	const SurfaceCoordinatesFormat&		Offsets and scales on X and Y axes of the surface associated with the image objects were found on
←	outImageBox	Box&		
←	outImageRectangle	Optional<Rectangle2D&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outImageRectangle**.

Read more about Optional Outputs.

SurfaceBoxToRegion

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Finds the surface region covered by surface box.

Applications: Transformation of objects defined in the surface coordinate system to the associated surface region.

Syntax

```
void avl::SurfaceBoxToRegion
(
 const avl::Box3D& inSurfaceBox,
 const avl::SurfaceFormat& inSurfaceFormat,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Default	Description
→	inSurfaceBox	const Box3D&		
→	inSurfaceFormat	const SurfaceFormat&		Surface format for which the output region will be defined
←	outRegion	Region&		Output region

SurfaceCircleToImageCoordinates

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Finds the associated image coordinates of surface circle.

Applications: Transformation of objects defined in the surface coordinate system to the associated image coordinate system.

Syntax

```
void avl::SurfaceCircleToImageCoordinates
(
 const avl::Circle3D& inSurfaceCircle,
 const avl::SurfaceCoordinatesFormat& inSurfaceCoordinatesFormat,
 avl::Circle2D& outImageCircle
)
```

Parameters

	Name	Type	Default	Description
→	inSurfaceCircle	const Circle3D&		
→	inSurfaceCoordinatesFormat	const SurfaceCoordinatesFormat&		Offsets and scales on X and Y axes of the surface associated with the image objects were found on
←	outImageCircle	Circle2D&		

SurfaceCircleToRegion

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Finds the surface region covered by surface circle.

Applications: Transformation of objects defined in the surface coordinate system to the associated surface region.

Syntax

```
void avl::SurfaceCircleToRegion
(
 const avl::Circle3D& inSurfaceCircle,
 const avl::SurfaceFormat& inSurfaceFormat,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Default	Description
→	inSurfaceCircle	const Circle3D&		
→	inSurfaceFormat	const SurfaceFormat&		Surface format for which the output region will be defined
←	outRegion	Region&		Output region

SurfaceCoordinateSystemToImageCoordinates

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Finds the associated image coordinates of surface coordinate system.

Applications: Transformation of objects defined in the surface coordinate system to the associated image coordinate system.

Syntax

```
void avl::SurfaceCoordinateSystemToImageCoordinates
(
 const avl::CoordinateSystem2D& inSurfaceCoordinateSystem,
 const avl::SurfaceCoordinatesFormat& inSurfaceCoordinatesFormat,
 avl::CoordinateSystem2D& outImageCoordinateSystem
)
```

Parameters

	Name	Type	Default	Description
→	inSurfaceCoordinateSystem	const CoordinateSystem2D&		
→	inSurfaceCoordinatesFormat	const SurfaceCoordinatesFormat&		Offsets and scales on X and Y axes of the surface associated with the image objects were found on
←	outImageCoordinateSystem	CoordinateSystem2D&		

SurfaceLineToImageCoordinates

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Finds the associated image coordinates of surface line.

Applications: Transformation of objects defined in the surface coordinate system to the associated image coordinate system.

Syntax

```
void avl::SurfaceLineToImageCoordinates
(
 const avl::Line3D& inSurfaceLine,
 const avl::SurfaceCoordinatesFormat& inSurfaceCoordinatesFormat,
 avl::Line2D& outImageLine
)
```

Parameters

	Name	Type	Default	Description
→	inSurfaceLine	const Line3D&		
→	inSurfaceCoordinatesFormat	const SurfaceCoordinatesFormat&		Offsets and scales on X and Y axes of the surface associated with the image objects were found on
←	outImageLine	Line2D&		

SurfacePathsToImageCoordinates

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Finds the associated image coordinates of surface paths.

Applications: Transformation of objects defined in the surface coordinate system to the associated image coordinate system.

Syntax

```
void avl::SurfacePathsToImageCoordinates
(
 const atl::Array<avl::Path>& inSurfacePaths,
 const avl::SurfaceCoordinatesFormat& inSurfaceCoordinatesFormat,
 atl::Array<avl::Path>& outImagePaths
)
```

Parameters

	Name	Type	Default	Description
→	inSurfacePaths	const Array<Path>&		
→	inSurfaceCoordinatesFormat	const SurfaceCoordinatesFormat&		Offsets and scales on X and Y axes of the surface associated with the image objects were found on
←	outImagePaths	Array<Path>&		

SurfacePathToImageCoordinates

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Finds the associated image coordinates of surface path.

Applications: Transformation of objects defined in the surface coordinate system to the associated image coordinate system.

Syntax

```
void avl::SurfacePathToImageCoordinates
(
 const avl::Path& inSurfacePath,
 const avl::SurfaceCoordinatesFormat& inSurfaceCoordinatesFormat,
 avl::Path& outImagePath
)
```

Parameters

	Name	Type	Default	Description
→	inSurfacePath	const Path&		
→	inSurfaceCoordinatesFormat	const SurfaceCoordinatesFormat&		Offsets and scales on X and Y axes of the surface associated with the image objects were found on
←	outImagePath	Path&		

SurfacePathToRegion

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Finds the surface region covered by surface path.

Applications: Transformation of objects defined in the surface coordinate system to the associated surface region.

Syntax

```
void avl::SurfacePathToRegion
(
 const avl::Path& inSurfacePath,
 const avl::SurfaceFormat& inSurfaceFormat,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Default	Description
→	inSurfacePath	const Path&		
→	inSurfaceFormat	const SurfaceFormat&		Surface format for which the output region will be defined
←	outRegion	Region&		Output region

SurfacePointsToImageCoordinates

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Finds the associated image coordinates of surface points.

Applications: Transformation of objects defined in the surface coordinate system to the associated image coordinate system.

Syntax

```
void avl::SurfacePointsToImageCoordinates
(
 const atl::Array<avl::Point3D>& inSurfacePoints,
 const avl::SurfaceCoordinatesFormat& inSurfaceCoordinatesFormat,
 atl::Array<avl::Point2D>& outImagePoints
)
```

Parameters

	Name	Type	Default	Description
→	inSurfacePoints	const Array<Point3D>&		
→	inSurfaceCoordinatesFormat	const SurfaceCoordinatesFormat&		Offsets and scales on X and Y axes of the surface associated with the image objects were found on
←	outImagePoints	Array<Point2D>&		

SurfacePointToImageCoordinates

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Finds the associated image coordinates of surface point.

Applications: Transformation of objects defined in the surface coordinate system to the associated image coordinate system.

Syntax

```
void avl::SurfacePointToImageCoordinates
(
 const avl::Point3D& inSurfacePoint,
 const avl::SurfaceCoordinatesFormat& inSurfaceCoordinatesFormat,
 avl::Point2D& outImagePoint
)
```

Parameters

	Name	Type	Default	Description
→	inSurfacePoint	const Point3D&		
→	inSurfaceCoordinatesFormat	const SurfaceCoordinatesFormat&		Offsets and scales on X and Y axes of the surface associated with the image objects were found on
←	outImagePoint	Point2D&		

SurfaceSegmentsToImageCoordinates

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Finds the associated image coordinates of surface segments.

Applications: Transformation of objects defined in the surface coordinate system to the associated image coordinate system.

Syntax

```
void avl::SurfaceSegmentsToImageCoordinates
(
 const atl::Array<avl::Segment3D>& inSurfaceSegments,
 const avl::SurfaceCoordinatesFormat& inSurfaceCoordinatesFormat,
 atl::Array<avl::Segment2D>& outImageSegments
)
```

Parameters

	Name	Type	Default	Description
→	inSurfaceSegments	const Array<Segment3D>&		
→	inSurfaceCoordinatesFormat	const SurfaceCoordinatesFormat&		Offsets and scales on X and Y axes of the surface associated with the image objects were found on
←	outImageSegments	Array<Segment2D>&		

SurfaceSegmentToImageCoordinates

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Finds the associated image coordinates of surface segment.

Applications: Transformation of objects defined in the surface coordinate system to the associated image coordinate system.

Syntax

```
void avl::SurfaceSegmentToImageCoordinates
(
 const avl::Segment3D& inSurfaceSegment,
 const avl::SurfaceCoordinatesFormat& inSurfaceCoordinatesFormat,
 avl::Segment2D& outImageSegment
)
```

Parameters

	Name	Type	Default	Description
→	inSurfaceSegment	const Segment3D&		
→	inSurfaceCoordinatesFormat	const SurfaceCoordinatesFormat&		Offsets and scales on X and Y axes of the surface associated with the image objects were found on
←	outImageSegment	Segment2D&		

SurfaceVectorToImageCoordinates

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Finds the associated image coordinates of surface vector.

Applications: Transformation of objects defined in the surface coordinate system to the associated image coordinate system.

Syntax

```
void avl::SurfaceVectorToImageCoordinates
(
 const avl::Vector3D& inSurfaceVector,
 const avl::SurfaceCoordinatesFormat& inSurfaceCoordinatesFormat,
 avl::Vector2D& outImageVector
)
```

Parameters

	Name	Type	Default	Description
→	inSurfaceVector	const Vector3D&		
→	inSurfaceCoordinatesFormat	const SurfaceCoordinatesFormat&		Offsets and scales on X and Y axes of the surface associated with the image objects were found on
←	outImageVector	Vector2D&		

TestSurface

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Returns a sample 3D surface.

Syntax

```
void avl::TestSurface
(
 TestSurfaceState& ioState,
 const avl::PlainType::Type& inPointType,
 const float inDensity,
 const float inScaleX,
 const float inScaleY,
 const float inScaleZ,
 avl::Surface& outSurface
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	TestSurfaceState&			Object used to maintain state of the function.
	inPointType	const PlainType::Type&		Int16	Type of single point Z coordinate
	inDensity	const float	1.0 - 20.0	4.0f	Density of points along each axis over a unit of 10
	inScaleX	const float	0.001 - ∞	1.0f	Scaling of output object on X axis
	inScaleY	const float	0.001 - ∞	1.0f	Scaling of output object on Y axis
	inScaleZ	const float	0.001 - ∞	1.0f	Scaling of output object on Z axis
	outSurface	Surface&			Output object

26. Shape Region Basics

Table of content:

AlignShapeRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Aligns a shape region to a coordinate system.

Syntax

```
void avl::AlignShapeRegion
(
 const avl::ShapeRegion& inShapeRegion,
 const avl::CoordinateSystem2D& inShapeRegionAlignment,
 bool inInverse,
 atl::Optional<int> inFrameWidth,
 atl::Optional<int> inFrameHeight,
 avl::ShapeRegion& outAlignedShapeRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inShapeRegion	const ShapeRegion&			
→	inShapeRegionAlignment	const CoordinateSystem2D&			Coordinate system to align to
→	inInverse	bool			Switches to the inverse transform
→	inFrameWidth	Optional<int>	0 - 65535	NIL	Width of the created region's frame
→	inFrameHeight	Optional<int>	0 - 65535	NIL	Height of the created region's frame
←	outAlignedShapeRegion	ShapeRegion&			

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inShapeRegion** and **outAlignedShapeRegion**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Invalid ShapeRegion in AlignShapeRegion.

CreateShapeRegionRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Creates a region corresponding to the given shape region.

Syntax

```
void avl::CreateShapeRegionRegion
(
 const avl::ShapeRegion& inShapeRegion,
 atl::Optional<const avl::CoordinateSystem2D&> inShapeRegionAlignment,
 int inFrameWidth,
 int inFrameHeight,
 avl::Region& outRegion,
 atl::Optional<avl::ShapeRegion&> outAlignedShapeRegion = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inShapeRegion	const ShapeRegion&			
→	inShapeRegionAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the shape region to the position of the inspected object
→	inFrameWidth	int	0 - 65535		Width of the created region's frame
→	inFrameHeight	int	0 - 65535		Height of the created region's frame
←	outRegion	Region&			Output region
←	outAlignedShapeRegion	Optional<ShapeRegion&>		NIL	Transformed input shape region

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedShapeRegion**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Invalid ShapeRegion in CreateShapeRegionRegion.

PathToShapeRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts a closed path to a shape region.

Syntax

```
void avl::PathToShapeRegion
(
 const avl::Path& inPath,
 avl::ShapeRegion& outShapeRegion
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
←	outShapeRegion	ShapeRegion&		

ShapeRegionToRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts a shape region to a region with an automatically computed frame.

Syntax

```
void avl::ShapeRegionToRegion
(
 const avl::ShapeRegion& inShapeRegion,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Default	Description
→	inShapeRegion	const ShapeRegion&		
←	outRegion	Region&		Output region

Errors

List of possible exceptions:

Error type	Description
DomainError	Invalid ShapeRegion in ShapeRegionToRegion.

27. Geometry 3D Fitting

Table of content:

AnalyzePoint3DGrid

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Analyzes the steps in X and Y directions of a point 3D grid.

Syntax

```
void avl::AnalyzePoint3DGrid
(
 const avl::Point3DGrid& inPointGrid,
 avl::Point3DGridType::Type inPointGridType,
 const avl::ValueLimits& inXLimits,
 const avl::ValueLimits& inYLimits,
 const avl::ValueLimits& inZLimits,
 const float inEpsilonX,
 const float inEpsilonY,
 const float inCoverage,
 double& outScaleX,
 double& outOffsetX,
 double& outScaleY,
 double& outOffsetY
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPointGrid	const Point3DGrid&			
→	inPointGridType	Point3DGridType::Type			
→	inXLimits	const ValueLimits&			Limits for the X coordinate
→	inYLimits	const ValueLimits&			Limits for the Y coordinate
→	inZLimits	const ValueLimits&			Limits for the Z coordinate
→	inEpsilonX	const float	0.0 - ∞	0.0f	Maximum possible error in the X direction
→	inEpsilonY	const float	0.0 - ∞		Maximum possible error in the Y direction
→	inCoverage	const float	0.0 - 1.0	0.3f	Percentage of points that should be taken into account
←	outScaleX	double&			
←	outOffsetX	double&			
←	outScaleY	double&			
←	outOffsetY	double&			

Errors

List of possible exceptions:

Error type	Description
DomainError	Unsupported inPointGridType in AnalyzePoint3DGrid.

FitCircle3DToHole

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Fits a circle to a hole in a plane.

Syntax

```
void avl::FitCircle3DToHole
(
 const avl::Point3DGrid& inPointGrid,
 atl::Optional<const avl::Region&> inRoi,
 avl::MEstimator::Type inPlaneOutlierSuppression,
 float inClippingFactor,
 int inIterationCount,
 atl::Optional<const avl::Plane3D&> inInitialPlane,
 avl::CircleFittingMethod::Type inCircleFittingMethod,
 atl::Optional<avl::MEstimator::Type> inCircleOutlierSuppression,
 atl::Conditional<avl::Circle3D>& outCircle3D,
 avl::Plane3D& outPlane,
 atl::Optional<atl::Array<avl::Point3D>&> outInliers = atl::NIL,
 atl::Optional<atl::Array<float>&> outDistances = atl::NIL,
 atl::Optional<float&> outSignedDistanceSum = atl::NIL,
 atl::Optional<float&> outDistanceSum = atl::NIL,
 atl::Optional<atl::Array<float>&> outSquaredDistances = atl::NIL,
 atl::Optional<float&> outSquaredDistanceSum = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPointGrid	const Point3DGrid&			
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inPlaneOutlierSuppression	MEstimator::Type			
→	inClippingFactor	float	0.675 - 6.0	2.5f	Multitude of standard deviation within which points are considered inliers
→	inIterationCount	int	0 - ∞	5	Number of iterations of outlier suppressing algorithm
→	inInitialPlane	Optional<const Plane3D&>		NIL	Initial approximation of a plane (if available)
→	inCircleFittingMethod	CircleFittingMethod::Type			
→	inCircleOutlierSuppression	Optional<MEstimator::Type>		NIL	
←	outCircle3D	Conditional<Circle3D>&			Circle fitted to a hole
←	outPlane	Plane3D&			
←	outInliers	Optional<Array<Point3D>&>		NIL	Points matching the computed plane
←	outDistances	Optional<Array<float>&>		NIL	Distances of the input points to a resulting plane
←	outSignedDistanceSum	Optional<float&>		NIL	Sum of signed distances of the input points to a resulting plane
←	outDistanceSum	Optional<float&>		NIL	Sum of distances of the input points to a resulting plane
←	outSquaredDistances	Optional<Array<float>&>		NIL	Squared distances of the input points to a resulting plane
←	outSquaredDistanceSum	Optional<float&>		NIL	Sum of squared distances of the input points to a resulting plane

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outInliers**, **outDistances**, **outSignedDistanceSum**, **outDistanceSum**, **outSquaredDistances**, **outSquaredDistanceSum**.

Read more about Optional Outputs.

FitCircleToPoints3D

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Approximates points in 3D with a circle using selected outliers suppression method.

Syntax

```
void avl::FitCircleToPoints3D
(
 const atl::Array<avl::Point3D>& inPoints,
 avl::CircleFittingMethod::Type inFittingMethod,
 atl::Optional<avl::MEstimator::Type> inOutlierSuppression,
 atl::Optional<const avl::Plane3D&> inPlane,
 atl::Conditional<avl::Circle3D>& outCircle,
 atl::Optional<atl::Array<avl::Point3D>&> outInliers = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point3D>&		
→	inFittingMethod	CircleFittingMethod::Type		
→	inOutlierSuppression	Optional<MEstimator::Type>	NIL	
→	inPlane	Optional<const Plane3D&>	NIL	
←	outCircle	Conditional<Circle3D>&		Fitted circle or nothing if method failed to converge,
←	outInliers	Optional<Array<Point3D>&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outInliers**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty point array on input in FitCircleToPoints3D.

FitLineToPoints3D

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Approximates points in 3D with a line using the Least Squares method.

Syntax

```
void avl::FitLineToPoints3D
(
 const atl::Array<avl::Point3D>& inPoints,
 avl::Line3D& outLine,
 atl::Optional<float&> outError = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point3D>&		
←	outLine	Line3D&		
←	outError	Optional<float&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outError**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty point array in FitLineToPoints3D.

FitLineToPoints3D_LTE

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Approximates points in 3D with a line using Least Trimmed Error algorithm.

Syntax

```
void avl::FitLineToPoints3D_LTE
(
 const atl::Array<avl::Point3D>& inPoints,
 int inSeedSubsetSize,
 atl::Optional<int> inEvalSubsetSize,
 avl::Line3D& outLine,
 atl::Optional<atl::Array<avl::Point3D>&> outLTInliers = atl::NIL,
 atl::Optional<float&> outLTError = atl::NIL,
 int& diagIterationCount = atl::Dummy<int>()
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoints	const Array<Point3D>&			
→	inSeedSubsetSize	int	2 - 10	3	Number of points in one combination for getting a sample line
→	inEvalSubsetSize	Optional<int>	3 - ∞	NIL	Number of closest points used for evaluation of a sample line, or Auto if seed points are to be used
←	outLine	Line3D&			Fitted line
←	outLTInliers	Optional<Array<Point3D>&>		NIL	Inlying points of the best LTE line
←	outLTError	Optional<float&>		NIL	The Least Trimmed Error
🔍	diagIterationCount	int&			Number of combinations considered

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outLTInliers**, **outLTError**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty inPoints array in FitLineToPoints3D_LTE.

FitLineToPoints3D_M

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Approximates points in 3D with a line using selected M-estimator for outlier suppression.

Applications: Finding a locally optimal line in 3D. Good enough when the number of outliers is small.

Syntax

```
void avl::FitLineToPoints3D_M
(
 const atl::Array<avl::Point3D>& inPoints,
 avl::MEstimator::Type inOutlierSuppression,
 float inClippingFactor,
 int inIterationCount,
 atl::Optional<const avl::Line3D&> inInitialLine,
 avl::Line3D& outLine,
 atl::Optional<atl::Array<avl::Point3D>&> outInliers = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoints	const Array<Point3D>&			
→	inOutlierSuppression	MEstimator::Type			
→	inClippingFactor	float	0.675 - 6.0	2.5f	Multitude of standard deviation within which points are considered inliers
→	inIterationCount	int	0 - ∞	5	Number of iterations of outlier suppressing algorithm
→	inInitialLine	Optional<const Line3D&>		NIL	Initial approximation (if available)
←	outLine	Line3D&			
←	outInliers	Optional<Array<Point3D>&>		NIL	Points matching the computed line

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outInliers**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty point array on input in FitLineToPoints3D_M

FitLineToPoints3D_RANSAC

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Approximates points in 3D with a line using a RANSAC algorithm.

Syntax

```
void avl::FitLineToPoints3D_RANSAC
(
 const atl::Array<avl::Point3D>& inPoints,
 atl::Optional<int> inMaxOutlierCount,
 const float inMaxInlierDistance,
 atl::Optional<int> inIterationCount,
 atl::Conditional<avl::Line3D>& outLine
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoints	const Array<Point3D>&			
→	inMaxOutlierCount	Optional<int>	0 - ∞	NIL	Determines how many outlier points can be present to end the search
→	inMaxInlierDistance	const float	0.0 - ∞	3.0f	Distance from the output line for a point to be considered an inlier
→	inIterationCount	Optional<int>	1 - ∞	42	Number of iterations; Auto means that all point pairs will be used
←	outLine	Conditional<Line3D>&			

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty inPoints array in FitLineToPoints3D_RANSAC.

FitPlaneToPoints

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Approximates points with a plane using the Least Squares method.

Syntax

```
void avl::FitPlaneToPoints
(
 const atl::Array<avl::Point3D>& inPoints,
 avl::Plane3D& outPlane,
 atl::Optional<atl::Array<float>&> outDistances = atl::NIL,
 atl::Optional<float&> outSignedDistanceSum = atl::NIL,
 atl::Optional<float&> outDistanceSum = atl::NIL,
 atl::Optional<atl::Array<float>&> outSquaredDistances = atl::NIL,
 atl::Optional<float&> outSquaredDistanceSum = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point3D>&		
←	outPlane	Plane3D&		Fitted plane
←	outDistances	Optional<Array<float>&>	NIL	Distances of the input points to a resulting plane
←	outSignedDistanceSum	Optional<float&>	NIL	Sum of signed distances of the input points to a resulting plane
←	outDistanceSum	Optional<float&>	NIL	Sum of distances of the input points to a resulting plane
←	outSquaredDistances	Optional<Array<float>&>	NIL	Squared distances of the input points to a resulting plane
←	outSquaredDistanceSum	Optional<float&>	NIL	Sum of squared distances of the input points to a resulting plane

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outDistances**, **outSignedDistanceSum**, **outDistanceSum**, **outSquaredDistances**, **outSquaredDistanceSum**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty point array on input in FitPlaneToPoints.

FitPlaneToPoints_M

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Approximates points with a plane using selected M-estimator for outlier suppression.

Applications: Finding a locally optimal plane. Good enough when the number of outliers is small.

Syntax

```
void avl::FitPlaneToPoints_M
(
 const atl::Array<avl::Point3D>& inPoints,
 avl::MEstimator::Type inOutlierSuppression,
 float inClippingFactor,
 int inIterationCount,
 atl::Optional<const avl::Plane3D&> inInitialPlane,
 avl::Plane3D& outPlane,
 atl::Optional<atl::Array<avl::Point3D>&> outInliers = atl::NIL,
 atl::Optional<atl::Array<float>&> outDistances = atl::NIL,
 atl::Optional<float&> outSignedDistanceSum = atl::NIL,
 atl::Optional<float&> outDistanceSum = atl::NIL,
 atl::Optional<atl::Array<float>&> outSquaredDistances = atl::NIL,
 atl::Optional<float&> outSquaredDistanceSum = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inPoints	const Array<Point3D>&			
➡	inOutlierSuppression	MEstimator::Type			
➡	inClippingFactor	float	0.675 - 6.0	2.5f	Multitude of standard deviation within which points are considered inliers
➡	inIterationCount	int	0 - ∞	5	Number of iterations of outlier suppressing algorithm
➡	inInitialPlane	Optional<const Plane3D&>		NIL	Initial approximation (if available)
⬅	outPlane	Plane3D&			
⬅	outInliers	Optional<Array<Point3D>&>		NIL	Points matching the computed plane
⬅	outDistances	Optional<Array<float>&>		NIL	Distances of the input points to a resulting plane
⬅	outSignedDistanceSum	Optional<float&>		NIL	Sum of signed distances of the input points to a resulting plane
⬅	outDistanceSum	Optional<float&>		NIL	Sum of distances of the input points to a resulting plane
⬅	outSquaredDistances	Optional<Array<float>&>		NIL	Squared distances of the input points to a resulting plane
⬅	outSquaredDistanceSum	Optional<float&>		NIL	Sum of squared distances of the input points to a resulting plane

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outInliers**, **outDistances**, **outSignedDistanceSum**, **outDistanceSum**, **outSquaredDistances**, **outSquaredDistanceSum**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty point array on input in FitPlaneToPoints_M.

FitSegmentToPoints3D

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Approximates points in 3D with a segment using selected outliers suppression method.

Syntax

```
void avl::FitSegmentToPoints3D
(
 const atl::Array<avl::Point3D>& inPoints,
 atl::Optional<avl::MEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Segment3D>& outSegment,
 atl::Optional<atl::Array<avl::Point3D>&> outInliers = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point3D>&		
→	inOutlierSuppression	Optional<MEstimator::Type>	Tukey	
←	outSegment	Conditional<Segment3D>&		Fitted segment or nothing if method failed to converge
←	outInliers	Optional<Array<Point3D>&>	NIL	Points matching the computed segment

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outInliers**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty point array on input in FitSegmentToPoints3D.

FitSegmentToPoints3D_LTE

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Approximates points in 3D with a segment using a RANSAC algorithm.

Syntax

```
void avl::FitSegmentToPoints3D_LTE
(
 const atl::Array<avl::Point3D>& inPoints,
 int inSeedSubsetSize,
 atl::Optional<int> inEvalSubsetSize,
 avl::Segment3D& outSegment,
 atl::Optional<atl::Array<avl::Point3D>&> outLTInliers = atl::NIL,
 atl::Optional<float&> outLTError = atl::NIL,
 int& diagIterationCount = atl::Dummy<int>()
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoints	const Array<Point3D>&			
→	inSeedSubsetSize	int	2 - 10	3	Number of points in one combination for getting a sample segment
→	inEvalSubsetSize	Optional<int>	3 - ∞	NIL	Number of closest points used for evaluation of a sample segment, or Auto if seed points are to be used
←	outSegment	Segment3D&			Fitted segment
←	outLTInliers	Optional<Array<Point3D>&>		NIL	Inlying points of the best LTE segment
←	outLTError	Optional<float&>		NIL	The Least Trimmed Error
🔍	diagIterationCount	int&			Number of combinations considered

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outLTInliers**, **outLTError**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty inPoints array in FitSegmentToPoints3D_LTE.

FitSegmentToPoints3D_RANSAC

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Approximates points in 3D with a segment using a RANSAC algorithm.

Syntax

```
void avl::FitSegmentToPoints3D_RANSAC
(
 const atl::Array<avl::Point3D>& inPoints,
 atl::Optional<int> inMaxOutlierCount,
 const float inMaxInlierDistance,
 atl::Optional<int> inIterationCount,
 atl::Conditional<avl::Segment3D>& outSegment
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoints	const Array<Point3D>&			
→	inMaxOutlierCount	Optional<int>	0 - ∞	NIL	Determines how many outlier points can be present to end the search
→	inMaxInlierDistance	const float	0.0 - ∞	3.0f	Distance from the output line for a point to be considered an inlier
→	inIterationCount	Optional<int>	1 - ∞	42	Number of iterations; Auto means that all point pairs will be used
←	outSegment	Conditional<Segment3D>&			

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty inPoints array in FitSegmentToPoints3D_RANSAC.

28. Geometry 2D Angle Metrics

Table of content:

AngleBetweenDirections

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Measures the rotation angle from first to second direction.

Syntax

```
void avl::AngleBetweenDirections
(
 float inDirection1,
 float inDirection2,
 atl::Optional<avl::RotationDirection::Type> inRotationDirection,
 avl::AngleRange::Type inAngleRange,
 atl::Optional<float&> outAbsoluteAngle = atl::NIL,
 atl::Optional<float&> outDirectedAngle = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inDirection1	float		Start direction
→	inDirection2	float		Target direction
→	inRotationDirection	Optional<RotationDirection::Type>	NIL	Clockwise, counter-clockwise or automatic (by smaller angle)
→	inAngleRange	AngleRange::Type	_0_180	Switches between ranges <0; 90), <0; 180) and <0; 360)
←	outAbsoluteAngle	Optional<float&>	NIL	Angle value used for measurements <0; 360>
←	outDirectedAngle	Optional<float&>	NIL	Angle value used for clockwise transformations <-360; 360>

Optional Outputs

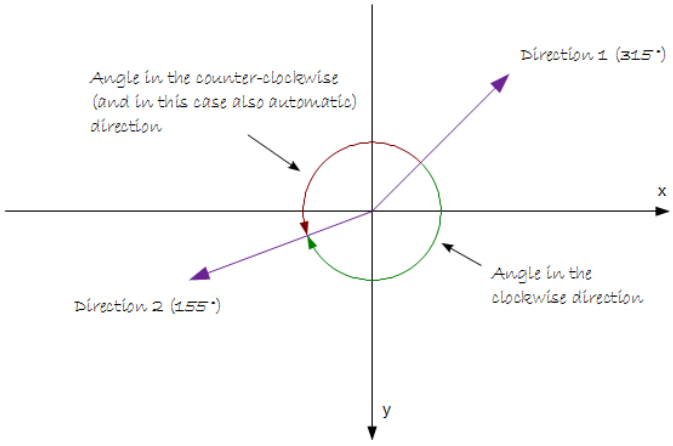
The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAbsoluteAngle**, **outDirectedAngle**.

Read more about Optional Outputs.

Description

Direction is a number in the range of <0, 180) or <0, 360). This is controlled with the **inAngleRange** input. The angle is computed from first to second direction clockwise or counter-clockwise. If the rotation direction is not specified, chosen is the one that produces smaller absolute angle.

Examples



See Also

- AngleBetweenVectors – Measures the angle between two vectors.
- AngleBetweenSegments – Measures the angle between two segments with one of four possible metrics.
- AngleBetweenLines – Measures the smaller and the larger angle between two lines.

AngleBetweenLines

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Measures the smaller and the larger angle between two lines.

Syntax

```
void avl::AngleBetweenLines
(
 const avl::Line2D& inLine1,
 const avl::Line2D& inLine2,
 atl::Optional<float&> outSmallerAngle = atl::NIL,
 atl::Optional<float&> outLargerAngle = atl::NIL
)
```

Parameters

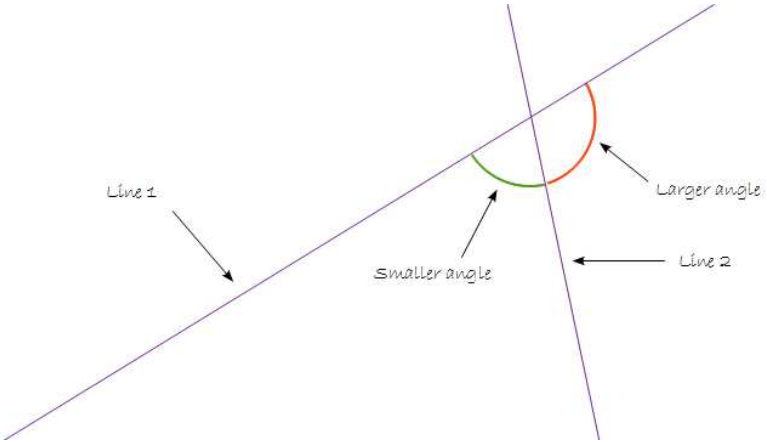
	Name	Type	Default	Description
→	inLine1	const Line2D&		
→	inLine2	const Line2D&		
←	outSmallerAngle	Optional<float&>	NIL	
←	outLargerAngle	Optional<float&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outSmallerAngle**, **outLargerAngle**.

Read more about Optional Outputs.

Examples



Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite line on input in AngleBetweenLines.

See Also

- AngleBetweenSegments – Measures the angle between two segments with one of four possible metrics.

AngleBetweenSegments

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Measures the angle between two segments with one of four possible metrics.

Syntax

```
void avl::AngleBetweenSegments
(
 const avl::Segment2D& inSegment1,
 const avl::Segment2D& inSegment2,
 avl::AngleMetric::Type inAngleMetric,
 bool inAutodetectOrientation,
 atl::Optional<float&> outAbsoluteAngle = atl::NIL,
 atl::Optional<float&> outDirectedAngle = atl::NIL,
 atl::Optional<avl::Arc2D&> outArc = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inSegment1	const Segment2D&		First segment
→	inSegment2	const Segment2D&		Second segment
→	inAngleMetric	AngleMetric::Type		Chooses one of four possible ways of measuring the angle
→	inAutodetectOrientation	bool	True	Autodetects orientation of the segments assuming that these are two consecutive sides of a polygon
←	outAbsoluteAngle	Optional<float&>	NIL	Angle value used for measurements <0; 360>
←	outDirectedAngle	Optional<float&>	NIL	Angle value used for clockwise transformations <-360; 360>
←	outArc	Optional<Arc2D&>	NIL	Angle visualization object

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAbsoluteAngle**, **outDirectedAngle**, **outArc**.

Read more about Optional Outputs.

Description

Angle between segments can be interpreted in two different ways:

1. Segments as vectors - the angle is measured from the first to the second vector clockwise or counter-clockwise.
2. Segments as consecutive sides of a polygon - the angle is measured from the first to the second side clockwise or counter-clockwise.

The method that is appropriate in a particular case can be chosen with the **inAngleMetric** input. Possible values are:

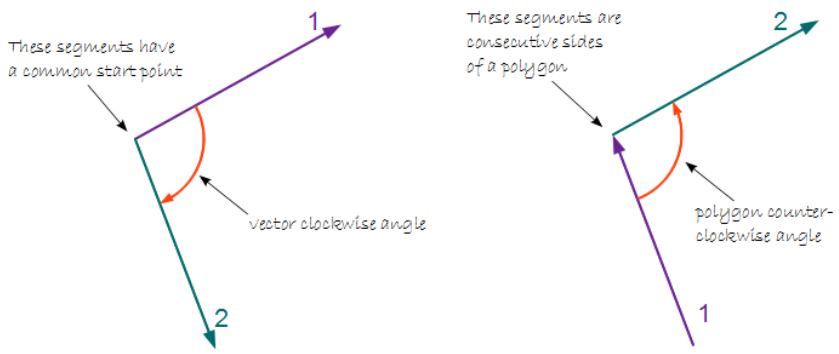
- VectorClockwiseAngle
- VectorCounterClockwiseAngle
- PolygonClockwiseAngle
- PolygonCounterClockwiseAngle

Segment Orientation

We do not usually care about orientation of a segment. However, when measuring angles, this information is important. In the most typical application, which is measuring an angle between two consecutive sides of a polygon, orientation of the segments is used together with **inAngleMetric** to determine the corner point and the side of the polygon (inside or outside). This information can also be obtained automatically by considering the intersection point between lines containing the segments. The **inAutodetectOrientation** parameter is there for that purpose. However, keep in mind that this method may produce unexpected results when the segments are parallel or nearly parallel.

Note that because of inaccuracies of floating-point arithmetic, some geometric operations (including this one) may lead to unpredictable results for degenerated cases. In this filter such a case occurs when an empty segment is given on input.

Examples



Errors

List of possible exceptions:

Error type	Description
DomainError	Unsupported angle metric in AngleBetweenSegments.

See Also

- AngleBetweenThreePoints – Measures the angle defined by three consecutive points.
- AngleBetweenVectors – Measures the angle between two vectors.

AngleBetweenThreePoints

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Measures the angle defined by three consecutive points.

Syntax

```
void avl::AngleBetweenThreePoints
(
 const avl::Point2D& inPoint1,
 const avl::Point2D& inPoint2,
 const avl::Point2D& inPoint3,
 avl::RotationDirection::Type inRotationDirection,
 atl::Optional<float&> outAbsoluteAngle = atl::NIL,
 atl::Optional<float&> outDirectedAngle = atl::NIL,
 atl::Optional<avl::Arc2D&> outArc = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPoint1	const Point2D&		A point on one arm of an angle
→	inPoint2	const Point2D&		The middle point
→	inPoint3	const Point2D&		A point on another arm of the angle
→	inRotationDirection	RotationDirection::Type		Chooses one of two ways of measuring the angle
←	outAbsoluteAngle	Optional<float&>	NIL	Angle value used for measurements <0; 360>
←	outDirectedAngle	Optional<float&>	NIL	Angle value used for clockwise transformations <-360; 360>
←	outArc	Optional<Arc2D&>	NIL	Angle visualization object

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAbsoluteAngle**, **outDirectedAngle**, **outArc**.

Read more about Optional Outputs.

Description

Note that because of inaccuracies of floating-point arithmetic, some geometric operations (including this one) may lead to unpredictable results for degenerated cases. In this filter such a case occurs when some two of the input points are almost equal.

AngleBetweenVectors

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Measures the angle between two vectors.

Syntax

```
void avl::AngleBetweenVectors
(
 const avl::Vector2D& inVector1,
 const avl::Vector2D& inVector2,
 atl::Optional<avl::RotationDirection::Type> inRotationDirection,
 atl::Optional<float&> outAbsoluteAngle = atl::NIL,
 atl::Optional<float&> outDirectedAngle = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inVector1	const Vector2D&		Start vector
→	inVector2	const Vector2D&		Target vector
→	inRotationDirection	Optional<RotationDirection::Type>	NIL	Clockwise, counter-clockwise or automatic (by smaller angle)
←	outAbsoluteAngle	Optional<float&>	NIL	Angle value used for measurements <0; 360>
←	outDirectedAngle	Optional<float&>	NIL	Angle value used for clockwise transformations <-360; 360>

Optional Outputs

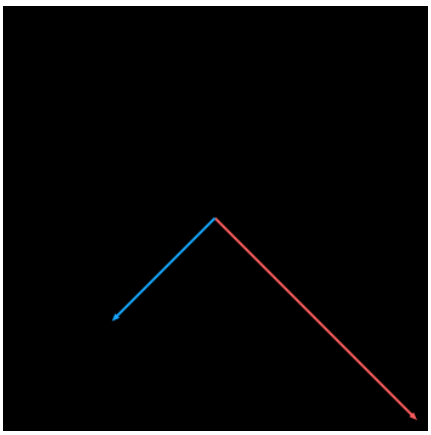
The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAbsoluteAngle**, **outDirectedAngle**.

Read more about Optional Outputs.

Description

Note that because of inaccuracies of floating-point arithmetic, some geometric operations (including this one) may lead to unpredictable results for degenerated cases. In this filter such a case occurs when a zero vector is given on input.

Examples



AngleBetweenVectors *performed on two vectors:* ***inVector1*** *DeltaX = 10, DeltaY = 10,* ***inVector2*** *DeltaX = -5, DeltaY = 5,*
inRotationDirection *= CounterClockwise.*
outAbsoluteAngle *returns* ***270****, and* ***outDirectedAngle*** *returns* ***-270****.*

NormalizeAngle

Header: AVL.h
Namespace: avl
Module: FoundationLite

Normalizes an angle to the range <x, x+90) or <x, x+180) or <x, x+360).

Syntax

```
void avl::NormalizeAngle
(
 float inAngle,
 avl::AngleRange::Type inAngleRange,
 float inBaseAngle,
 float& outNormalizedAngle
)
```

Parameters

	Name	Type	Default	Description
→	inAngle	float		
→	inAngleRange	AngleRange::Type	_0_180	Length of the normalization range: 90, 180 or 360
→	inBaseAngle	float	0.0f	The beginning of the normalization range
←	outNormalizedAngle	float&		

29. Geometry 3D Angle Metrics

Table of content:

AngleBetweenLines3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Measures the smaller and the larger angle between two lines in 3D.

Syntax

```
void avl::AngleBetweenLines3D
(
 const avl::Line3D& inLine1,
 const avl::Line3D& inLine2,
 atl::Optional<float&> outSmallerAngle = atl::NIL,
 atl::Optional<float&> outLargerAngle = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inLine1	const Line3D&		
→	inLine2	const Line3D&		
←	outSmallerAngle	Optional<float&>	NIL	
←	outLargerAngle	Optional<float&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outSmallerAngle**, **outLargerAngle**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite line on input in AngleBetweenLines3D.

AngleBetweenPlanes

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Measures the smaller and the larger angle between two planes in 3D.

Syntax

```
void avl::AngleBetweenPlanes
(
 const avl::Plane3D& inPlane1,
 const avl::Plane3D& inPlane2,
 atl::Optional<float&> outSmallerAngle = atl::NIL,
 atl::Optional<float&> outLargerAngle = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPlane1	const Plane3D&		
→	inPlane2	const Plane3D&		
←	outSmallerAngle	Optional<float&>	NIL	
←	outLargerAngle	Optional<float&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outSmallerAngle**, **outLargerAngle**.

Read more about Optional Outputs.

Description

The operation returns the single measure of the angle between two planes in 3D, which is equal to the angle between their normal vectors.

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite plane on input in AngleBetweenPlanes.

See Also

- AngleBetweenLines3D – Measures the smaller and the larger angle between two lines in 3D.
- AngleBetweenVectors3D – Measures the angle between two vectors in 3D.

AngleBetweenSegments3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Measures the angle between two segments in 3D.

Syntax

```
void avl::AngleBetweenSegments3D
(
 const avl::Segment3D& inSegment1,
 const avl::Segment3D& inSegment2,
 float& outAngle
)
```

Parameters

	Name	Type	Default	Description
→	inSegment1	const Segment3D&		First segment
→	inSegment2	const Segment3D&		Second segment
←	outAngle	float&		Angle value used for measurements <0; 180>

AngleBetweenThreePoints3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Measures the angle defined by three consecutive points in 3D.

Syntax

```
void avl::AngleBetweenThreePoints3D
(
 const avl::Point3D& inPoint1,
 const avl::Point3D& inPoint2,
 const avl::Point3D& inPoint3,
 float& outAngle
)
```

Parameters

	Name	Type	Default	Description
→	inPoint1	const Point3D&		A point on one arm of an angle
→	inPoint2	const Point3D&		The middle point
→	inPoint3	const Point3D&		A point on another arm of the angle
←	outAngle	float&		Angle value used for measurements <0; 180>

AngleBetweenVectors3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Measures the angle between two vectors in 3D.

Syntax

```
void avl::AngleBetweenVectors3D
(
 const avl::Vector3D& inVector1,
 const avl::Vector3D& inVector2,
 float& outAngle
)
```

Parameters

	Name	Type	Default	Description
→	inVector1	const Vector3D&		Start vector
→	inVector2	const Vector3D&		Target vector
←	outAngle	float&		Angle value used for measurements <0; 180>

30. Camera Calibration

Table of content:

- WorldPlaneSegmentToImage
- WorldPlaneStripesToImage
- WorldPlaneStripeToImage

AnnotateGridPoints

Header: AVL.h
Namespace: avl
Module: Calibration

Select a subset of the given points that forms a grid and assign 2D indices to them.

Applications: Recognition of a custom regular grid for the purpose of camera calibration.

Syntax

```
void avl::AnnotateGridPoints
(
 const atl::Array<avl::Point2D>& inPoints,
 atl::Optional<float> inMinDistance,
 atl::Optional<float> inMaxDistance,
 const float inAngleTolerance,
 atl::Array<avl::AnnotatedPoint2D>& outImageGrid,
 atl::Array<avl::Segment2D>& diagValidSubgraph
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoints	const Array<Point2D>&			Points to calculate a grid
→	inMinDistance	Optional<float>	0.0 - ∞	NIL	Minimum distance between two rows or two columns in pixels
→	inMaxDistance	Optional<float>	0.0 - ∞	NIL	Maximum distance between two rows or two columns in pixels
→	inAngleTolerance	const float	0.0 - 90.0	10.0f	Maximum deviation from right angles in the grid
←	outImageGrid	Array<AnnotatedPoint2D>&			Detected grid
🔍	diagValidSubgraph	Array<Segment2D>&			Graph forming a valid grid

AnnotateGridPoints_DeprecatedDeprecated

Header: AVL.h
Namespace: avl
Module: Calibration

Select a subset of the given points that forms a grid and assign world plane coordinates to them.

Applications: Recognition of a custom regular grid, other than the standard chessboard, for the purpose of camera calibration.

Syntax

```
void avl::AnnotateGridPoints_DeprecatedDeprecated
(
 const atl::Array<avl::Point2D>& inPoints,
 const float inScale,
 atl::Optional<float> inMinDistance,
 const float inMaxDistance,
 const float inAngleTolerance,
 atl::Array<avl::Point3D>& outPoints3D,
 atl::Array<avl::Point2D>& outSelectedPoints,
 atl::Array<avl::Segment2D>& diagValidSubgraph
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoints	const Array<Point2D>&			Points to calculate a grid
→	inScale	const float	0.001 - ∞	1.0f	Distance between two rows or two columns of the grid in world units
→	inMinDistance	Optional<float>	0.0 - ∞	NIL	Minimum distance between two rows or two columns in pixels
→	inMaxDistance	const float	0.0 - ∞	10.0f	Maximum distance between two rows or two columns in pixels
→	inAngleTolerance	const float	0.0 - 90.0	10.0f	Maximum deviation from right angles in the grid
←	outPoints3D	Array<Point3D>&			Calculated points in the world units
←	outSelectedPoints	Array<Point2D>&			Which points were selected to calculate the grid
🔍	diagValidSubgraph	Array<Segment2D>&			Graph forming a valid grid

AnnotateGridPoints_Ransac

Header: AVL.h
Namespace: avl
Module: Calibration

Select a subset of the given points that forms a grid and assign 2D indices to them.

Syntax

```
void avl::AnnotateGridPoints_Ransac
(
 const atl::Array<avl::Point2D>& inPoints,
 const atl::Optional<int> inMaxAttempts,
 const atl::Optional<int> inMaxOutlierCount,
 const float inMinLength,
 const atl::Optional<float> inMaxLength,
 const float inMaxLengthRatio,
 const float inAngleRange,
 const float inBaseAspectRatioRange,
 const atl::Optional<float> inBaseAspectRatio,
 const atl::Optional<float> inBaseShear,
 const atl::Optional<float> inBaseOrientation,
 atl::Array<avl::AnnotatedPoint2D>& outPointGrid,
 atl::Array<avl::Segment2D>& diagValidSubgraph,
 float& diagMaxLength
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inPoints	const Array<Point2D>&			
➡	inMaxAttempts	const Optional<int>	1 - ∞	NIL	Maximum number of attempts at finding the grid
➡	inMaxOutlierCount	const Optional<int>	0 - ∞	NIL	Determines how many outlier points can be present to end the search
➡	inMinLength	const float	0.0 - ∞	0.0f	Minimum length of a grid segment
➡	inMaxLength	const Optional<float>	0.0 - ∞	10.0f	Maximum length of a grid segment, if set to Auto it will be approximated
➡	inMaxLengthRatio	const float	1.0 - 2.0	1.3f	Maximum ratio of two consecutive segments in the grid
➡	inAngleRange	const float	0.0 - 45.0	16.0f	Maximum variation of angles between neighbouring grid segments in degrees
➡	inBaseAspectRatioRange	const float	0.0 - 1.0	0.3f	Maximum variation of the base aspect ratio (ignores if base aspect ratio is not given)
➡	inBaseAspectRatio	const Optional<float>	0.3 - 1.0	1.0f	A reference aspect ratio of the grid
➡	inBaseShear	const Optional<float>	0.0 - 60.0	00.0f	A reference shear angle between grid directions
➡	inBaseOrientation	const Optional<float>	0.0 - 360.0	NIL	A reference orientation of one of the grids directions
⬅	outPointGrid	Array<AnnotatedPoint2D>&			Detected grid
🔍	diagValidSubgraph	Array<Segment2D>&			Graph forming a valid grid
🔍	diagMaxLength	float&			Max length computed by the filter (applicable if inMaxLength is set to Auto)

Description

This filter analyzes a 2D point array and finds a grid within it. The grid can have missing points as well as noise (points that are not vertices of the gird). The filter provides various input that allow the user to specify the shape of the grid that they want to be found.

It is expected that the point array given does not have a lot of missing and/or noise points.

The filter uses a RANSAC based algorithm to find grid within the input array and returns the grid that includes the most points from the input array among all grids that were found. If more than one grid include the maximum number of points, the returned one is the one with the smallest angle deviations.

The algorithm starts by finding a grid base, which consists of three points: one that will be considered as the origin of the grid, one that is one unit away from the origin on the grids X axis and one that is one unit away on the grids Y axis. The constraints for these grid bases can be specified by the user with the use of Base parameters:

- **inBaseAspectRatio** defines the length ratio between the two segments that the grid base consists of (one along the X axis and the other along the Y axis). The ratio is calculated by divining the length of the shorter segment by the length of the longer one. If this parameter is set to Auto, the algorithm will check bases with rations in the range 0.3 - 1.0
- **inBaseAspectRatioRange** defines how much the given base aspect ratio can differ from the calculated one.
- **inBaseShear** defines the shear of the grid in degrees. The calculated shear can differ from the one given in this parameter by **inAngleRange** degrees. If this parameter is not given, bases with shears up to 60 degrees will be considered but grids with shears closer to zero will be preferred.
- **inBaseOrientation** defines the orientation of the grids X axis. If this parameters is not given then the positive X direction of the grid will be defined as the direction that gives the biggest positive change in real X coordinated.

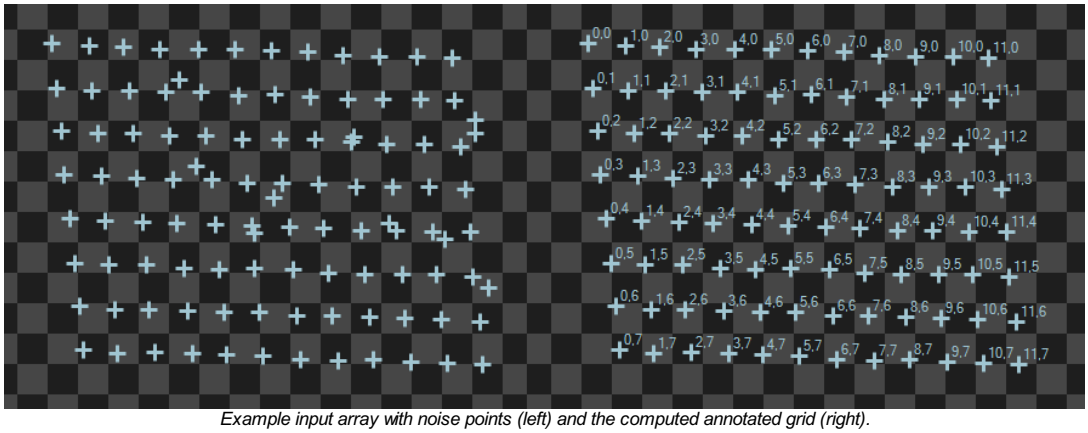
The grid is build from the base with respect to constraint parameters:

- **inMinLength** and **inMaxLength** define the range of possible segment length in the grid (a segment is a single step in either the X or Y direction).
- **inMaxLengthRatio** the maximum length ratio between to neighbouring segments of the grid that are along the same axis (for example: two neighbouring segments along the X axis). The minimum length ratio is calculated as 1 / **inMaxLengthRatio**.
- **inAngleRange** defines how much the angles of neighbouring grid segments can differ.

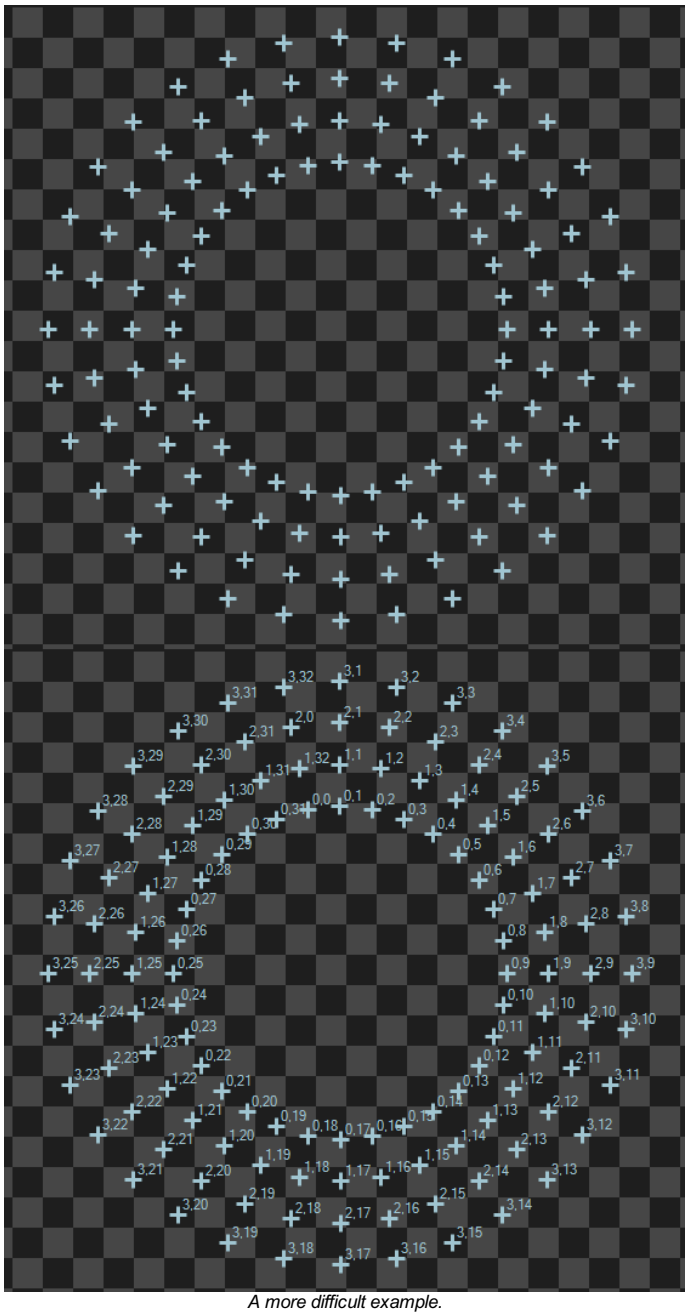
Hints

- The computation time of the filter can be decreased by lowering **inMaxAttempts**.
- The computation time of the filter can be greatly optimized with the **inMinLength** and **inMaxLength** parameters. As they allow the algorithm to filter out a lot of grids early on.
- **inBaseOrientation** can be used to increase the stability of detected grids. This is especially useful for grids the are rotated by 45 degrees, as the X and Y directions of the grid are harder to determine just by looking at positive X changes in the grid.

Examples



Example input array with noise points (left) and the computed annotated grid (right).



A more difficult example.

See Also

- FilterGridPoints – Select a subset of the given points that forms a grid.

CalibrateCamera_LineScan

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the line scan camera intrinsic parameters from calibration grid.

Applications: Computes camera parameters which need to be calculated only once for each camera, and are a prerequisite for other calibration filters.

Syntax

```
void avl::CalibrateCamera_LineScan
(
 const atl::Array<avl::AnnotatedPoint2D>& inImageGrid,
 float inGridSpacing,
 int inImageWidth,
 avl::LensDistortionModelType::Type inDistortionType,
 float inImagePointsStandardDeviation,
 atl::Optional<float> inFocalLength,
 avl::LineScanCameraModel& outCameraModel,
 float& outApproxScaleRatio,
 atl::Optional<float&> outRmsError = atl::NIL,
 atl::Optional<float&> outMaxReprojectionError = atl::NIL,
 atl::Optional<atl::Array<avl::Segment2D>&> outReprojectionErrorSegments = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImageGrid	const Array<AnnotatedPoint2D>&			Annotated calibration grid
→	inGridSpacing	float	0.000001 - ∞		Real-world distance between adjacent grid points.
→	inImageWidth	int	1 - ∞		Image width, used for initial estimation of principal point.
→	inDistortionType	LensDistortionModelType::Type		Polynomial	Lens distortion model
→	inImagePointsStandardDeviation	float	0.0 - ∞	0.1f	Assumed uncertainty of inImagePoints. Used for robust optimization.
→	inFocalLength	Optional<float>		NIL	Specify a fixed focal length, in pixels. In order to calculate the inFocalLength from camera parameters one needs to divide the lens focal length [mm] by sensor pitch [mm/pix].
←	outCameraModel	LineScanCameraModel&			
←	outApproxScaleRatio	float&			Approximate scale ratio between Y and X. Useful for camera/encoder trigger rate configuration. When greater than 1, the image is stretched in Y dimension, when less than 1 it is compressed.
←	outRmsError	Optional<float&>		NIL	Final reprojection RMS error, in pixels.
←	outMaxReprojectionError	Optional<float&>		NIL	Maximum reprojection error among all points.
←	outReprojectionErrorSegments	Optional<Array<Segment2D>&>		NIL	Array of segments connecting input image points to grid reprojections.

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outRmsError**, **outMaxReprojectionError**, **outReprojectionErrorSegments**.

Read more about Optional Outputs.

Description

The filter estimates intrinsic parameters of whole line scan camera system, which consist of a camera and a moving conveyor belt. Such approach, in contrast with area scan camera calibration, is necessary as the moving element of line scan camera system is tightly bound within the image acquisition geometry. This allows for handling distortions caused by:

- intrinsic lens distortions
- image shear caused by deviation from right angle between conveyor belt motion and the camera line
- nonuniform image scaling caused by deviation from ideal encoder trigger period
- 1D perspective effects caused by deviation from right angle between camera optical axis and the imaged plane

Calibration uses a planar calibration grid to perform robust minimization of RMS reprojection error - the square root of averaged squared distances between grid points as observed on the image and their associated grid coordinates projected onto image plane using estimated parameters.

The calibration routine cannot estimate focal length by itself, however it can be set to a fixed value via **inFocalLength**. The **inFocalLength** is measured in pixels, it can be calculated from the sensor and lens parameters:

$$f_{pix} = \frac{f_{lens}}{pp \cdot d}$$

where f_pix - focal length measured in pixels, f_lens - focal length of lens measured in millimeters, pp - sensor pixel pitch measured in millimeters per pixel, d - camera binning or/and image downscaling factor

The **inFocalLength** can also be obtained from angle of view:

$$f_{pix} = \frac{w}{2 \cdot tan(\frac{\alpha}{2})}$$

where f_pix - focal length measured in pixels, w - image width, alpha - angle of view

Only the divisional and polynomial lens distortion models are supported for line scan cameras. The divisional supports most use cases and has predictable behaviour even when calibration data is sparse. Polynomial model may be more accurate but it needs a larger dataset of high quality calibration points across the whole image.

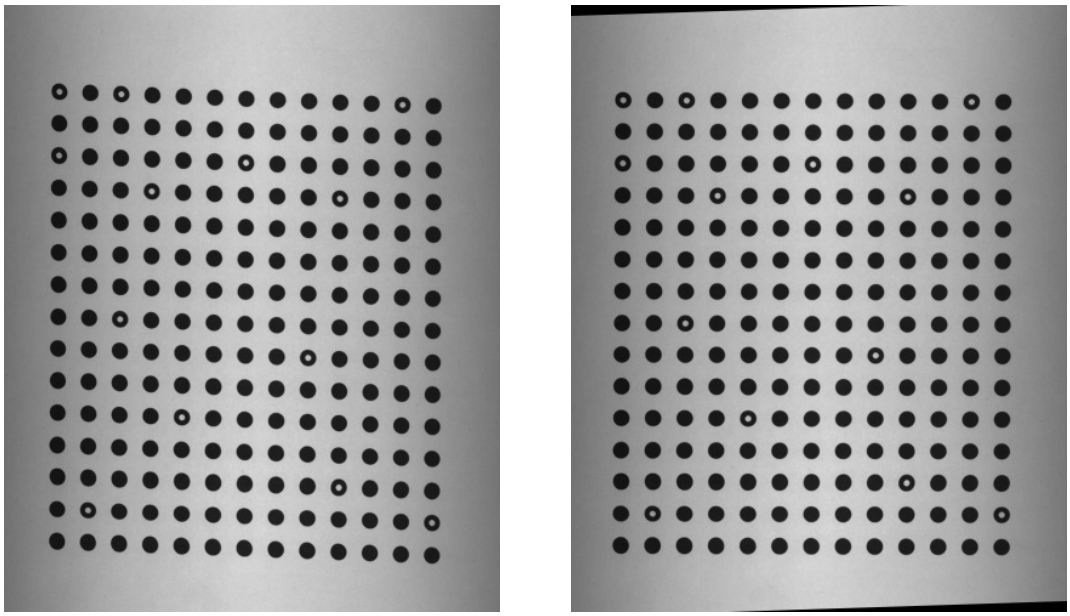
The filter provides a few methods for judging the feasibility of calculated solution.

- The **outRmsError** is the final RMS reprojection error. The main contributor to that value is the random noise in **inImageGrid** points positions. Model mismatch will also result in increased **outRmsError**.
- The **outMaxReprojectionError** is the maximum reprojection error, can be used to judge if there are outliers in the calibration data.
- The **outReprojectionErrorSegments** consists of segments connecting input image points to reprojected world points, and thus it can be readily used for visualization of gross errors. The XY scatter plot of residual vectors (obtained by using SegmentVector on the **outReprojectionErrorSegments**) is a good insight into the residuals distribution, which in ideal case should follow a 2D gaussian distribution centered around point (0,0).

Hints

- High accuracy camera calibration needs a considerable amount of high quality calibration points, especially when using more complicated models. Calibration image should contain hundreds of calibration points spanning the area of interest. The calibration grids should be as flat and stiff as possible (cardboard is not a proper backing material, thick glass is perfect). Take care of proper conditions when taking the calibration images: minimize motion blur by proper lighting, prevent reflections from the calibration surface (ideally use diffusion lighting).
- When the focal length of the camera is not provided via **inFocalLength**, the calculated model will lack that value. Despite that, some operations will still work properly, such as distortion removal, or basic image to world plane calibration (CalibrateWorldPlane_* filters).

Examples



Left: calibration grid as captured by the line scan camera. Right: rectified image with shear distortion eliminated.

Remarks

Note, that the calibration routine assumes that the camera line spans the image along its width (i.e. consecutive image rows correspond to consecutive camera acquisitions).

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty input grid
DomainError	inGridSpacing needs to be positive
DomainError	Thin prism lens distortion is not supported for line scan cameras.

See Also

- CalibrateCamera_Pinhole – Finds the camera intrinsic parameters from calibration grids. Uses pinhole camera model (perspective camera).
- CalibrateCamera_Telecentric – Finds the telecentric camera intrinsic parameters from calibration grids.

CalibrateCamera_Pinhole

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the camera intrinsic parameters from calibration grids. Uses pinhole camera model (perspective camera).

Applications: Computes camera parameters which need to be calculated only once for each camera, and are a prerequisite for other calibration filters.

Syntax

```
void avl::CalibrateCamera_Pinhole
(
 const atl::Array<atl::Array<avl::AnnotatedPoint2D>>& inImageGrids,
 int inImageWidth,
 int inImageHeight,
 avl::LensDistortionModelType::Type inDistortionType,
 float inImagePointsStandardDeviation,
 atl::Optional<float> inFocalLength,
 avl::PinholeCameraModel& outCameraModel,
 atl::Optional<avl::PinholeCameraModel&> outCameraModelStdDev = atl::NIL,
 atl::Optional<float&> outRmsError = atl::NIL,
 atl::Optional<atl::Array<float>&> outMaxReprojectionErrors = atl::NIL,
 atl::Optional<atl::Array<atl::Array<avl::Segment2D>>&> outReprojectionErrorSegments = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImageGrids	const Array<Array<AnnotatedPoint2D>>&			For each view, the annotated calibration grid
→	inImageWidth	int	1 - ∞		Image width, used for initial estimation of principal point.
→	inImageHeight	int	1 - ∞		Image height, used for initial estimation of principal point.
→	inDistortionType	LensDistortionModelType::Type		Polynomial	Lens distortion model
→	inImagePointsStandardDeviation	float	0.0 - ∞	0.1f	Assumed uncertainty of inImagePoints. Used for robust optimization and outCameraModelStdDev estimation.
→	inFocalLength	Optional<float>		NIL	Specify a fixed focal length (do not estimate), in pixels. In order to calculate the inFocalLength from camera parameters one needs to divide the lens focal length [mm] by sensor pitch [mm/pix].
←	outCameraModel	PinholeCameraModel&			
←	outCameraModelStdDev	Optional<PinholeCameraModel&>		NIL	Standard deviations of all model parameters, assuming that inImagePoints positions have a standard deviation equal to inImagePointsStandardDeviation.
←	outRmsError	Optional<float&>		NIL	Final reprojection RMS error, in pixels.
←	outMaxReprojectionErrors	Optional<Array<float>&>		NIL	For each view: the maximum reprojection error among all points.
←	outReprojectionErrorSegments	Optional<Array<Array<Segment2D>>&>		NIL	For each view: array of segments connecting input image points to grid reprojections.

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outCameraModelStdDev**, **outRmsError**, **outMaxReprojectionErrors**, **outReprojectionErrorSegments**.

Read more about Optional Outputs.

Description

The filter estimates intrinsic camera parameters - focal length, principal point location and distortion coefficients from a set of planar calibration grids by means of robust minimization of RMS reprojection error - the square root of averaged squared distances between grid points as observed on the image and their associated grid coordinates projected onto image plane using estimated parameters (i.e. grid poses and camera parameters).

The focal length can be calculated by the filter if at least one calibration grid is not perpendicular to the optical axis of the camera. Alternatively, the focal length can be set to a fixed value via **inFocalLength**. The **inFocalLength** is measured in pixels, it can be calculated from the sensor and lens parameters:

$$f_{pix} = \frac{f_{lens}}{pp \cdot d}$$

where f_pix - focal length measured in pixels, f_lens - focal length of lens measured in millimeters, pp - sensor pixel pitch measured in millimeters per pixel, d - camera binning or/and image downscaling factor

The **inFocalLength** can also be obtained from angle of view, for horizontal case following equation applies:

$$f_{pix} = \frac{w}{2 \cdot tan(\frac{\alpha}{2})}$$

where f_pix - focal length measured in pixels, w - image width, alpha - horizontal angle of view

A few distortion model types are supported. The simplest - divisional - supports most use cases and has predictable behaviour even when calibration data is sparse. Higher order models can be more accurate, however they need a much larger dataset of high quality calibration points, and are usually needed for achieving high levels of positional accuracy across the whole image - order of magnitude below 0.1 pix. Of course this is only a rule of thumb, as each lens is different and there are exceptions.

Distortion model types are compatible with OpenCV, and are expressed with equations using normalized image coordinates:

$$x'' = x' \frac{1}{1 + k_4 \cdot r^2}$$
$$y'' = y' \frac{1}{1 + k_4 \cdot r^2}$$

Divisional distortion model

$$x'' = x'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 x' y' + p_2(r^2 + 2x'^2)$$

$$y'' = y'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + p_1(r^2 + 2y'^2) + 2p_2 x' y'$$

Polynomial distortion model

$$x'' = x'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 x' y' + p_2(r^2 + 2x'^2) + s_1 r^2 + s_2 r^4$$

$$y'' = y'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + p_1(r^2 + 2y'^2) + 2p_2 x' y' + s_3 r^2 + s_4 r^4$$

PolynomialWithThinPrism distortion model

where $r^2 = x'^2 + y'^2$, x' and y' are undistorted, x'' and y'' are distorted normalized image coordinates.

Although every application should be evaluated separately the general guidelines for choosing the right distortion type you will find in table below:

		Required calibration data:		Typical application:	Suggested Lens type:
		Quality:	Quantity:		
Distortion model	Divisional	Average	Average	Robot Guidance	Pinhole
	Polynomial	Very high. Calibration plate should be made of stiff material like metal plate or glass.	Very high evenly distributed across entire ROI. Calculation on areas not covered with calibration plate might be highly inaccurate.	Metrology	Pinhole/Telecentric
	Polynomial with thin prism			Metrology in micrometers	Telecentric

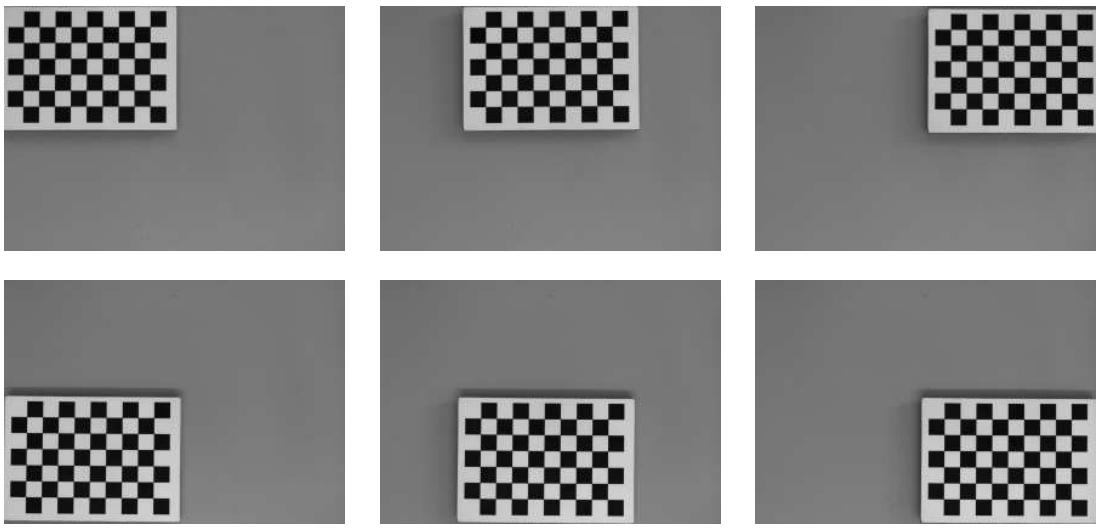
The filter provides a few methods for judging the feasibility of calculated solution.

- The **outRmsError** is the final RMS reprojection error. The main contributor to that value is the random noise in **inImageGrids** points positions. Model mismatch will also result in increased **outRmsError**.
- The **outMaxReprojectionErrors** is an array of maximum reprojection errors, per view. It can be used to find suspicious calibration grids.
- The **outCameraModelStdDev** contains standard deviations of all parameters of estimated model, assuming that **inImageGrids** points positions have a standard deviation equal to **inImagePointsStandardDeviation**. It can be used to verify the stability of estimated parameters. High values may be a result of data deficiency in a sensitive region (e.g. lack of calibration points at the edges of image when high-order distortion model is used).
- The **outReprojectionErrorSegments** consists of segments connecting input image points to reprojected world points, and thus it can be readily used for visualization of gross errors. The XY scatter plot of residual vectors (obtained by using SegmentVector on the **outReprojectionErrorSegments**) is a good insight into the residuals distribution, which in ideal case should follow a 2D gaussian distribution centered around point (0,0).

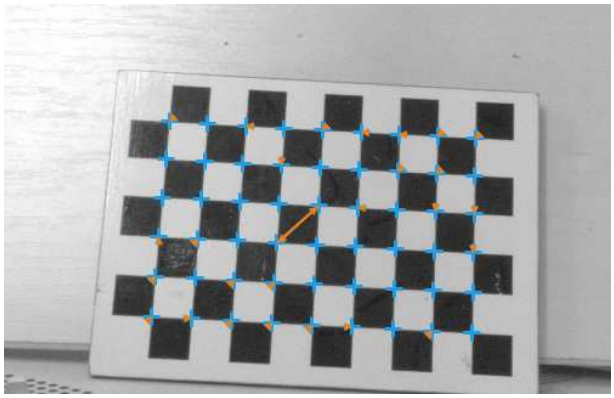
Hints

- High accuracy camera calibration needs a considerable amount of high quality calibration points, especially when using more complicated models. Each calibration image should contain hundreds of calibration points, there should be at least a dozen of calibration images, and all the calibration points should span the area/volume of interest. The calibration grids should be as flat and stiff as possible (cardboard is not a proper backing material, thick glass is perfect). Take care of proper conditions when taking the calibration images: minimize motion blur by proper camera and grid mounts, prevent reflections from the calibration surface (ideally use diffusion lighting).
- There is no need for camera calibration in the production environment, the camera can be calibrated beforehand. As soon as the camera has been assembled with the lens and lens adjustments (zoom/focus/f-stop rings) have been tightly locked, the calibration images can be taken and camera calibration performed. Of course any modifications to the camera-lens setup void the calibration parameters, even apparently minor ones such as removing the lens and putting it back on the camera in seemingly the same position.
- Calibration set containing images of calibration grids that are not perpendicular to the optical axis of the camera allow for automatic focal length estimation. When all calibration grids are perpendicular to the optical axis of the camera and the focal length of the camera is not provided via **inFocalLength**, the calculated model will lack that value. Despite that, some operations will still work properly, such as distortion removal, or basic image to world plane calibration (CalibrateWorldPlane_* filters).
- Note that the pinhole camera calibration doesn't require scale information for its input calibration grids, e.g. when using DetectCalibrationGrid_Chessboard for grid detection, its **inWorldSquareSize** input has no effect on the calibration. Relative scale information between the grid dimensions is still crucial, e.g. when using non-square grid, the aspect ratio of point distances in perpendicular directions must be precisely known.

Examples



A set of grid pictures for basic calibration. Note that high accuracy applications require denser grids and higher amount of pictures.



*Usage of **outReprojectionErrorSegments** for identification of bad association of image points and their grid coordinates in **inImageGrids** - two points are swapped.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty input grid array

See Also

- CalibrateCamera_Telecentric – Finds the telecentric camera intrinsic parameters from calibration grids.

CalibrateCamera_Telecentric

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the telecentric camera intrinsic parameters from calibration grids.

Applications: Computes camera parameters which need to be calculated only once for each camera, and are a prerequisite for other calibration filters.

Syntax

```
void avl::CalibrateCamera_Telecentric
(
 const atl::Array<atl::Array<avl::AnnotatedPoint2D>>& inImageGrids,
 float inGridSpacing,
 int inImageWidth,
 int inImageHeight,
 avl::LensDistortionModelType::Type inDistortionType,
 float inImagePointsStandardDeviation,
 avl::TelecentricCameraModel& outCameraModel,
 atl::Optional<avl::TelecentricCameraModel&> outCameraModelStdDev = atl::NIL,
 atl::Optional<float&> outRmsError = atl::NIL,
 atl::Optional<atl::Array<float>&> outMaxReprojectionErrors = atl::NIL,
 atl::Optional<atl::Array<atl::Array<avl::Segment2D> >&> outReprojectionErrorSegments = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImageGrids	const Array<Array<AnnotatedPoint2D>>&			For each view, the annotated calibration grid
→	inGridSpacing	float	0.000001 - ∞		Real-world distance between adjacent grid points.
→	inImageWidth	int	1 - ∞		Image width, used for initial estimation of principal point.
→	inImageHeight	int	1 - ∞		Image height, used for initial estimation of principal point.
→	inDistortionType	LensDistortionModelType::Type		PolynomialWithThinPrism	Lens distortion model
→	inImagePointsStandardDeviation	float	0.0 - ∞	0.1f	Assumed uncertainty of inImagePoints. Used for robust optimization and outCameraModelStdDev estimation.
←	outCameraModel	TelecentricCameraModel&			
←	outCameraModelStdDev	Optional<TelecentricCameraModel&>		NIL	Standard deviations of all model parameters, assuming that inImagePoints positions are disturbed with gaussian noise with standard deviation equal to inImagePointsStandardDeviation.
←	outRmsError	Optional<float&>		NIL	Final reprojection RMS error, in pixels.
←	outMaxReprojectionErrors	Optional<Array<float>&>		NIL	For each view: the maximum reprojection error among all points.
←	outReprojectionErrorSegments	Optional<Array<Array<Segment2D> >&>		NIL	For each view: array of segments connecting input image points to grid reprojections.

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outCameraModelStdDev**, **outRmsError**, **outMaxReprojectionErrors**, **outReprojectionErrorSegments**.

Read more about Optional Outputs.

Description

The filter estimates intrinsic camera parameters - magnification, principal point location and distortion coefficients from a set of planar calibration grids by means of robust minimization of RMS reprojection error - the square root of averaged squared distances between grid points as observed on the image and their associated grid coordinates projected onto image plane using estimated parameters (i.e. grid poses and camera parameters).

A few distortion model types are supported. The simplest - divisional - supports most use cases and has predictable behaviour even when calibration data is sparse. Higher order models can be more accurate, however they need a much larger dataset of high quality calibration points, and are usually needed for achieving high levels of positional accuracy across the whole image - order of magnitude below 0.1 pix. Of course this is only a rule of thumb, as each lens is different and there are exceptions.

Distortion model types are compatible with OpenCV, and are expressed with equations using normalized image coordinates:

$$x'' = x' \frac{1}{1 + k_4 \cdot r^2}$$
$$y'' = y' \frac{1}{1 + k_4 \cdot r^2}$$

Divisional distortion model

$$x'' = x'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 x' y' + p_2(r^2 + 2x'^2)$$

$$y'' = y'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + p_1(r^2 + 2y'^2) + 2p_2 x' y'$$

Polynomial distortion model

$$x'' = x'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 x' y' + p_2(r^2 + 2x'^2) + s_1 r^2 + s_2 r^4$$

$$y'' = y'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + p_1(r^2 + 2y'^2) + 2p_2 x' y' + s_3 r^2 + s_4 r^4$$

PolynomialWithThinPrism distortion model

where $r^2 = x'^2 + y'^2$, x' and y' are undistorted, x'' and y'' are distorted normalized image coordinates.

Although every application should be evaluated separately the general guidelines for choosing the right distortion type you will find in table below:

		Required calibration data:		Typical application:	Suggested Lens type:
		Quality:	Quantity:		
Distortion model	Divisional	Average	Average	Robot Guidance	Pinhole
	Polynomial	Very high. Calibration plate should be made of stiff material like metal plate or glass.	Very high evenly distributed across entire ROI. Calculation on areas not covered with calibration plate might be highly inaccurate.	Metrology	Pinhole/Telecentric
	Polynomial with thin prism			Metrology in micrometers	Telecentric

The filter provides a few methods for judging the feasibility of calculated solution.

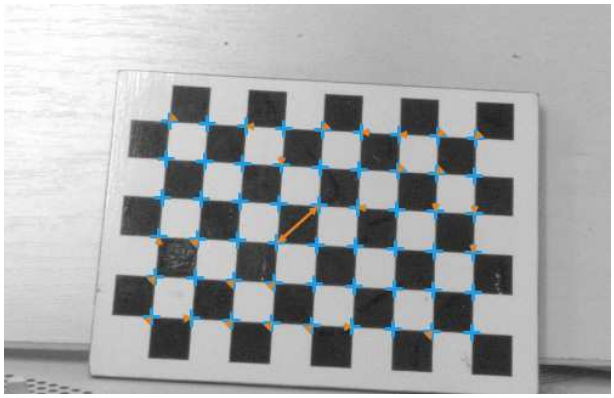
- The **outRmsError** is the final RMS reprojection error. The main contributor to that value is the random noise in **inImageGrids** points positions. Model mismatch will also result in increased **outRmsError**.
- The **outMaxReprojectionErrors** is an array of maximum reprojection errors, per view. It can be used to find suspicious calibration grids.
- The **outCameraModelStdDev** contains standard deviations of all parameters of estimated model, assuming that **inImageGrids** points positions have a standard deviation equal to **inImagePointsStandardDeviation**. It can be used to verify the stability of estimated parameters. High values may be a result of data deficiency in a sensitive region (e.g. lack of calibration points at the edges of image when high-order distortion model is used).
- The **outReprojectionErrorSegments** consists of segments connecting input image points to reprojected world points, and thus it can be readily used for visualization of gross errors. The XY scatter plot of residual vectors (obtained by using SegmentVector on the **outReprojectionErrorSegments**) is a good insight into the residuals distribution, which in ideal case should follow a 2D gaussian distribution centered around point (0,0).

Hints

High accuracy camera calibration needs a considerable amount of high quality calibration points, especially when using more complicated models. Each calibration image should contain hundreds of calibration points, there should be at least a dozen of calibration images, and all the calibration points should span the area/volume of interest. The calibration grids should be as flat and stiff as possible (cardboard is not a proper backing material, thick glass is perfect). Take care of proper conditions when taking the calibration images: minimize motion blur by proper camera and grid mounts, prevent reflections from the calibration surface (ideally use diffusion lighting).

There is no need for camera calibration in the production environment, the camera can be calibrated beforehand. As soon as the camera has been assembled with the lens and lens adjustments (zoom/focus/f-stop rings) have been tightly locked, the calibration images can be taken and camera calibration performed. Of course any modifications to the camera-lens setup void the calibration parameters, even apparently minor ones such as removing the lens and putting it back on the camera in seemingly the same position.

Examples



*Usage of **outReprojectionErrorSegments** for identification of bad association of image points and their grid coordinates in **inImageGrids** - two points are swapped.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty input array
DomainError	inGridSpacing needs to be positive

See Also

- CalibrateCamera_Pinhole – Finds the camera intrinsic parameters from calibration grids. Uses pinhole camera model (perspective camera).

CalibrateWorldPlane_Default

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the image to world plane transformation parameters from world plane calibration grid. World plane origin and axes are unspecified.

Applications: Useful for cases where the position of world plane origin is not important, e.g. distance measurements on a world plane, or image rectification for perspective removal.

Syntax

```
void avl::CalibrateWorldPlane_Default
(
 const atl::Array<avl::AnnotatedPoint2D>& inImageGrid,
 const atl::Optional<const avl::AnyCameraModel&>& inCameraModel,
 float inGridSpacing,
 float inGridThickness,
 avl::RectificationTransform& outTransform,
 atl::Optional<float&> outRmsError = atl::NIL,
 atl::Optional<float&> outMaxReprojectionError = atl::NIL,
 atl::Optional<atl::Array<avl::Segment2D>&> outReprojectionErrorSegments = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImageGrid	const Array<AnnotatedPoint2D>&			Annotated calibration grid
→	inCameraModel	const Optional<const AnyCameraModel&>&		NIL	For undistortion of inImageGrid. If not supplied, the filter will assume that grid came from undistorted image.
→	inGridSpacing	float	0.000001 - ∞	1.0f	Real-world distance between adjacent grid points.
→	inGridThickness	float		0.0f	The world plane will be shifted by given amount in direction perpendicular to the grid to compensate for grid thickness.
←	outTransform	RectificationTransform&			
←	outRmsError	Optional<float&>		NIL	RMS reprojection error, in pixels.
←	outMaxReprojectionError	Optional<float&>		NIL	Maximum reprojection error, in pixels.
←	outReprojectionErrorSegments	Optional<Array<Segment2D>&>		NIL	Array of segments connecting input image points to grid reprojections.

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outRmsError**, **outMaxReprojectionError**, **outReprojectionErrorSegments**.

Read more about Optional Outputs.

Description

The filter estimates the correspondence between the image plane and a "world plane" – a given planar surface in observed space. The image plane, and thus **inImageGrid** points are assumed to be distorted, and to correct for the distortion the **inCameraModel** (calibration data) needs to be provided. The calculated result – **outTransform** contains all the information for transforming the distorted image plane to the world plane.

The **inCameraModel** is also used to define the type of the planar correspondence. For a standard projective camera (pinhole camera), the planar correspondence is a homography. If the **inCameraModel** is a telecentric camera, the planar correspondence is affine (as there are no perspective parameters in orthographic projection). If no **inCameraModel** is provided, the filter defaults to homography.

The homography requires at least four **inImageGrid** points. The affine relation requires at least three.

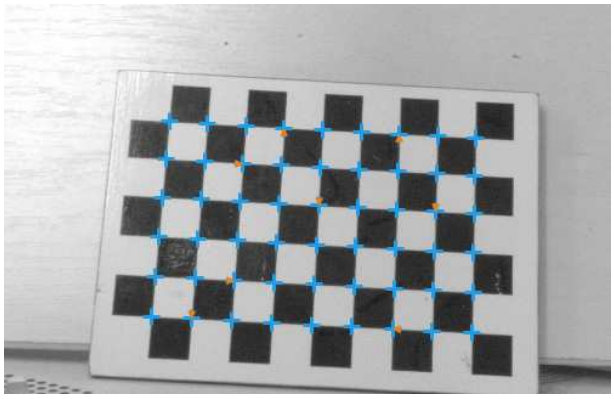
The filter provides a few methods for judging the feasibility of calculated solution.

- The **outRmsError** and **outMaxReprojectionError**. The main contributor to these values is the random noise in **inImageGrid** points positions. Model mismatch (i.e. trying to calibrate a non-planar object, e.g. wavy surface) will also result in increased reprojection errors. Large difference between **outRmsError** and **outMaxReprojectionError** could be a sign of presence of outliers in input data.
- The **outReprojectionErrorSegments** consists of segments connecting input image points to reprojected world points, and thus it can be readily used for visualization of per-point reprojection errors.

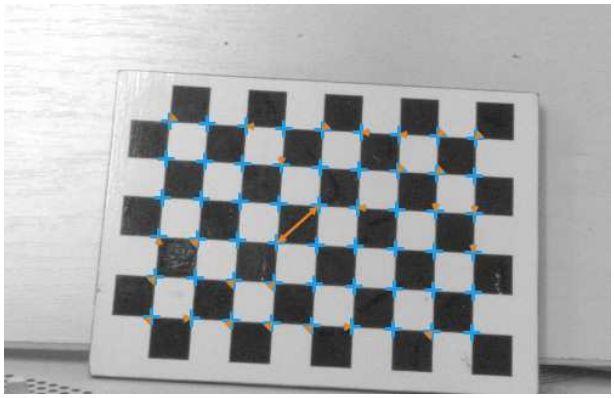
Hints

- Image to world plane coordinate transform can be obtained from as few as 4 point correspondences, however high accuracy calculations need a considerable amount of high quality calibration points. The calibration plane should contain about hundred points, spanning whole area of interest. Take care of proper conditions when taking the calibration images: reduce vibrations that may yield motion blur, prevent reflections from the calibration surface (ideally use diffusion lighting).

Examples



*Good case of image to world plane calibration using high amount of calibration points. Calibration resulted in RMS and maximum reprojection errors less than 1.0, as expected. The **outReprojectionErrorSegments** are not visible at that scale.*



*Example of a gross error. The RMS reprojection error is 17.6, max reprojection error is 91.2, source of these errors is clearly visible thanks to **outReprojectionErrorSegments**: the association of image points and their grid coordinates in **inImageGrid** is wrong.*

Errors

List of possible exceptions:

Error type	Description
DomainError	inGridSpacing needs to be positive

See Also

- CalibrateWorldPlane_OffgridOrigin – Finds the image to world plane transformation matrix, with world origin and axes specified in the image coordinates.
- CalibrateWorldPlane_Labeled – Finds the image to world plane transformation parameters using sparse world coordinate information, i.e. world coordinates are known for only a few points of the grid.
- CalibrateWorldPlane_Multigrid – Finds the image to world plane transformation parameters using multiple grids, using sparse world coordinate information.
- CalibrateWorldPlane_Manual – Finds the image to world plane transformation parameters. Image and their corresponding world points are directly specified (no grid needed).
- CalibrateCamera_Pinhole – Finds the camera intrinsic parameters from calibration grids. Uses pinhole camera model (perspective camera).
- CalibrateCamera_Telecentric – Finds the telecentric camera intrinsic parameters from calibration grids.
- ImagePointsToWorldPlane – Finds the world coordinates of image Points.
- WorldPlanePointsToImage – Finds the image coordinates of world plane Points.
- CreateRectificationMap_PixelUnits – Computes a spatial map for transforming distorted images to rectified images defined in world coordinate plane. Defines the output geometry in pixels.
- CreateRectificationMap_WorldUnits – Computes a spatial map for transforming distorted images to rectified images defined in world coordinate plane. Defines the output geometry in world units.

CalibrateWorldPlane_Labeled

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the image to world plane transformation parameters using sparse world coordinate information, i.e. world coordinates are known for only a few points of the grid.

Applications: Image to world coordinates transformations, also useful for image rectification where exact bounds of output image are important, e.g. stitching.

Syntax

```
void avl::CalibrateWorldPlane_Labeled
(
 const atl::Array<avl::AnnotatedPoint2D>& inImageGrid,
 const atl::Array<avl::AnnotatedPoint2D>& inLabeledWorldPoints,
 const atl::Optional<const avl::AnyCameraModel&>& inCameraModel,
 const atl::Optional<float>& inGridSpacing,
 float inGridThickness,
 bool inInvertedWorldY,
 avl::RectificationTransform& outTransform,
 atl::Optional< atl::Array<avl::AnnotatedPoint2D>& > outGridWorldPoints = atl::NIL,
 atl::Optional< atl::Conditional<float>& > outComputedGridSpacing = atl::NIL,
 atl::Optional< float& > outRmsImageError = atl::NIL,
 atl::Optional< float& > outRmsWorldError = atl::NIL,
 atl::Optional< float& > outMaxReprojectionError = atl::NIL,
 atl::Optional< atl::Array<avl::Segment2D>& > outReprojectionErrorSegments = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImageGrid	const Array<AnnotatedPoint2D>&			Annotated calibration grid.
→	inLabeledWorldPoints	const Array<AnnotatedPoint2D>&			Sparse array of world coordinate points. Annotations need to correspond to those in the inImageGrid input.
→	inCameraModel	const Optional<const AnyCameraModel&>&		NIL	For undistortion of inImageGrid. If not supplied, the filter will assume that grid came from undistorted image.
→	inGridSpacing	const Optional<float>&	0.000001 - ∞	NIL	World distance between grid indices. Used when spacing cannot be computed from supplied inLabeledWorldPoints.
→	inGridThickness	float		0.0f	The world plane will be shifted by given amount in direction perpendicular to the grid to compensate for grid thickness.
→	inInvertedWorldY	bool		False	Set to true if world coordinate system has right-handed orientation, also known as mathematical or standard.
←	outTransform	RectificationTransform&			
←	outGridWorldPoints	Optional< Array<AnnotatedPoint2D>& >		NIL	Array of world coordinates of the calibration grid points.
←	outComputedGridSpacing	Optional< Conditional<float>& >		NIL	World distance between grid indices. NIL when there is no enough information to compute the spacing.
←	outRmsImageError	Optional< float& >		NIL	RMS reprojection error of inImageGrid onto the image plane, in pixels. This is a partial error characterizing inaccuracies in perspective estimation, excluding the influence of world point labeling.
←	outRmsWorldError	Optional< float& >		NIL	RMS reprojection error of inLabeledWorldPoints onto the world plane, in world units. This is a partial error characterizing inaccuracies with labeling of world coordinate system, excluding perspective estimation.
←	outMaxReprojectionError	Optional< float& >		NIL	Maximum reprojection error of inImageGrid onto the image plane, in pixels. This is a partial error characterizing inaccuracies in perspective estimation, excluding the influence of world point labeling.
←	outReprojectionErrorSegments	Optional< Array<Segment2D>& >		NIL	Array of segments connecting inImageGrid points to their reprojections. Note that these segments depict only inaccuracies in perspective estimation - excluding inaccuracies due to the world point labeling.

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outGridWorldPoints**, **outComputedGridSpacing**, **outRmsImageError**, **outRmsWorldError**, **outMaxReprojectionError**, **outReprojectionErrorSegments**.

Read more about Optional Outputs.

Description

The filter estimates the correspondence between the image plane and a "world plane" – a given planar surface in observed space. It is capable of using a sparse world coordinate information - a calibration grid for which only a few world plane coordinates are known.

The image plane, and thus **inImageGrid** points are assumed to be distorted, and to correct for the distortion the **inCameraModel** (calibration data) needs to be provided. The calculated result – **outTransform** contains all the information for transforming the distorted image plane to the world plane.

The **inCameraModel** is also used to define the type of the planar correspondence. For a standard projective camera (pinhole camera), the planar correspondence is a homography. If the **inCameraModel** is a telecentric camera, the planar correspondence is affine (as there are no perspective parameters in orthographic projection). If no **inCameraModel** is provided, the filter defaults to homography.

To uniquely define the **outTransform**, the algorithm requires at least four **inImageGrid** points (the affine relation requires at least three), and at least two **inLabeledWorldPoints**.

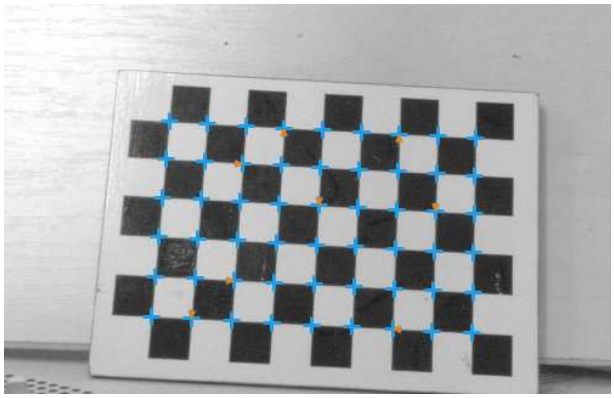
The filter provides a few methods for judging the feasibility of calculated solution.

- The **outRmsImageError** and **outRmsWorldError** could show if the error is due to the **inImageGrid** or **inLabeledWorldPoints** errors. Model mismatch (i.e. trying to calibrate a non-planar object, e.g. wavy surface) will also result in increased reprojection errors. Large difference between **outRmsImageError** and **outMaxReprojectionError** could be a sign of presence of outliers in **inImageGrid** input data.
- The **outReprojectionErrorSegments** consists of segments connecting input image points to reprojected world points, and thus it can be readily used for visualization of per-point reprojection errors (excluding the errors due to incorrect world plane labeling).

Hints

- Image to world plane coordinate transform can be obtained from as few as 4 image points (**inImageGrid** points), however high accuracy calculations need a considerable amount of high quality calibration points. The calibration plane should contain about hundred points, spanning whole area of interest. Take care of proper conditions when taking the calibration images: reduce vibrations that may yield motion blur, prevent reflections from the calibration surface (ideally use diffusion lighting).
- It is often assumed that calibration grid is perfect in terms of 2D world coordinates of its points. If such an assumption is made then 2 world coordinates are enough for labeling of grid world point coordinates (**inLabeledWorldPoints**).
- The output reprojection errors are useful tool for assessing the feasibility of computed solution. Note, that the filter outputs two RMS metrics: **outRmsImageError** and **outRmsWorldError**. The **outRmsImageError** characterizes inaccuracies in perspective estimation, it is not influenced by world coordinate labeling. On the other hand, the **outRmsWorldError** characterizes inaccuracies with labeling of grid 2D world coordinate system, excluding perspective estimation. In order for a solution to be feasible, both these errors must be low.

Examples



*Good case of image to world plane calibration using high amount of calibration points. Calibration resulted in RMS and maximum reprojection errors less than 1.0, as expected. The **outReprojectionErrorSegments** are not visible at that scale.*

Errors

List of possible exceptions:

Error type	Description
DomainError	inGridSpacing needs to be positive

See Also

- CalibrateWorldPlane_Default – Finds the image to world plane transformation parameters from world plane calibration grid. World plane origin and axes are unspecified.
- CalibrateWorldPlane_OffgridOrigin – Finds the image to world plane transformation matrix, with world origin and axes specified in the image coordinates.
- CalibrateWorldPlane_Multigrid – Finds the image to world plane transformation parameters using multiple grids, using sparse world coordinate information.
- CalibrateCamera_Pinhole – Finds the camera intrinsic parameters from calibration grids. Uses pinhole camera model (perspective camera).
- CalibrateCamera_Telecentric – Finds the telecentric camera intrinsic parameters from calibration grids.
- ImagePointsToWorldPlane – Finds the world coordinates of image Points.
- WorldPlanePointsToImage – Finds the image coordinates of world plane Points.
- CreateRectificationMap_PixelUnits – Computes a spatial map for transforming distorted images to rectified images defined in world coordinate plane. Defines the output geometry in pixels.
- CreateRectificationMap_WorldUnits – Computes a spatial map for transforming distorted images to rectified images defined in world coordinate plane. Defines the output geometry in world units.

CalibrateWorldPlane_Manual

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the image to world plane transformation parameters. Image and their corresponding world points are directly specified (no grid needed).

Applications: Prepares transformation parameters for image to world coordinate transformations or image rectification without the need for a world plane calibration grid.

Syntax

```
void avl::CalibrateWorldPlane_Manual
(
 const atl::Array<avl::Point2D>& inImagePoints,
 const atl::Array<avl::Point2D>& inWorldPlanePoints,
 const atl::Optional<const avl::AnyCameraModel&>& inCameraModel,
 float inGridThickness,
 avl::RectificationTransform& outTransform,
 atl::Optional<float&> outRmsError = atl::NIL,
 atl::Optional<float&> outMaxReprojectionError = atl::NIL,
 atl::Optional<atl::Array<avl::Segment2D>&> outReprojectionErrorSegments = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inImagePoints	const Array<Point2D>&		Array of 2D points of the calibration pattern, in the picture.
→	inWorldPlanePoints	const Array<Point2D>&		Array of 2D points of the calibration pattern, in a given world coordinate plane.
→	inCameraModel	const Optional<const AnyCameraModel&>&	NIL	For undistortion of inImageGrid. If not supplied, the filter will assume that grid came from undistorted image.
→	inGridThickness	float	0.0f	The world plane will be shifted by given amount in direction perpendicular to the grid to compensate for grid thickness.
←	outTransform	RectificationTransform&		
←	outRmsError	Optional<float&>	NIL	RMS reprojection error, in pixels.
←	outMaxReprojectionError	Optional<float&>	NIL	Maximum reprojection error, in pixels.
←	outReprojectionErrorSegments	Optional<Array<Segment2D>&>	NIL	Array of segments connecting input image points to reprojected world points.

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outRmsError**, **outMaxReprojectionError**, **outReprojectionErrorSegments**.

Read more about Optional Outputs.

Description

The filter estimates the correspondence between the image plane and a "world plane" – a given planar surface in observed space. The image plane, and thus **inImagePoints** are assumed to be distorted, and to correct for the distortion the **inCameraModel** (calibration data) needs to be provided. The calculated result – **outTransform** contains all the information for transforming the distorted image plane to the world plane.

The **inCameraModel** is also used to define the type of the planar correspondence. For a standard projective camera (pinhole camera), the planar correspondence is a homography. If the **inCameraModel** is a telecentric camera, the planar correspondence is affine (as there are no perspective parameters in orthographic projection). If no **inCameraModel** is provided, the filter defaults to homography.

The homography requires at least four **inImagePoints** (and their **inWorldPlanePoints** correspondences). The affine relation requires at least three such pairs.

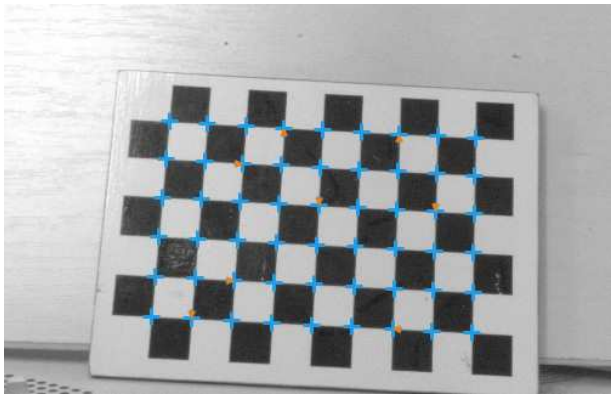
The filter provides a few methods for judging the feasibility of calculated solution.

- The **outRmsError** and **outMaxReprojectionError**. The main contributor to these values is the random noise in **inImagePoints** positions. Model mismatch (i.e. trying to calibrate a non-planar object, e.g. wavy surface) will also result in increased reprojection errors. Large difference between **outRmsError** and **outMaxReprojectionError** could be a sign of presence of outliers in input data.
- The **outReprojectionErrorSegments** consists of segments connecting input image points to reprojected world points, and thus it can be readily used for visualization of per-point reprojection errors.

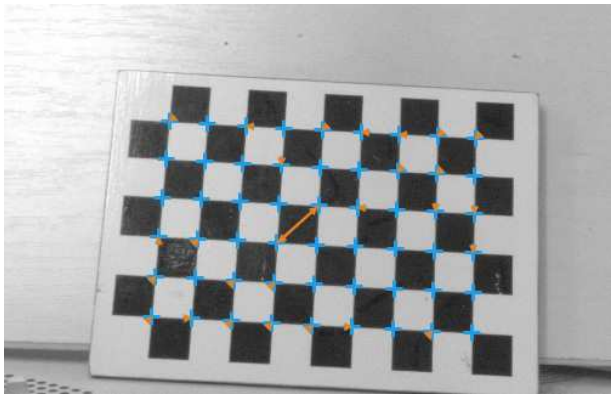
Hints

- Image to world plane coordinate transform can be obtained from as few as 4 point correspondences, however high accuracy calculations need a considerable amount of high quality calibration points. The calibration plane should contain about hundred points, spanning whole area of interest. Take care of proper conditions when taking the calibration images: reduce vibrations that may yield motion blur, prevent reflections from the calibration surface (ideally use diffusion lighting).

Examples



*Good case of image to world plane calibration using high amount of calibration points. Calibration resulted in RMS and maximum reprojection errors less than 1.0, as expected. The **outReprojectionErrorSegments** are not visible at that scale.*



*Example of a gross error. The RMS reprojection error is 17.6, max reprojection error is 91.2, source of these errors is clearly visible thanks to **outReprojectionErrorSegments**: the association of **inImagePoints** and **inWorldPlanePoints** is wrong.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Array inImagePoints and inWorldPlanePoints sizes differ

See Also

- CalibrateWorldPlane_Default – Finds the image to world plane transformation parameters from world plane calibration grid. World plane origin and axes are unspecified.
- CalibrateWorldPlane_OffgridOrigin – Finds the image to world plane transformation matrix, with world origin and axes specified in the image coordinates.
- CalibrateWorldPlane_Labeled – Finds the image to world plane transformation parameters using sparse world coordinate information, i.e. world coordinates are known for only a few points of the grid.
- CalibrateWorldPlane_Multigrid – Finds the image to world plane transformation parameters using multiple grids, using sparse world coordinate information.
- CalibrateCamera_Pinhole – Finds the camera intrinsic parameters from calibration grids. Uses pinhole camera model (perspective camera).
- CalibrateCamera_Telecentric – Finds the telecentric camera intrinsic parameters from calibration grids.
- ImagePointsToWorldPlane – Finds the world coordinates of image Points.
- WorldPlanePointsToImage – Finds the image coordinates of world plane Points.
- CreateRectificationMap_PixelUnits – Computes a spatial map for transforming distorted images to rectified images defined in world coordinate plane. Defines the output geometry in pixels.
- CreateRectificationMap_WorldUnits – Computes a spatial map for transforming distorted images to rectified images defined in world coordinate plane. Defines the output geometry in world units.

CalibrateWorldPlane_Multigrid

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the image to world plane transformation parameters using multiple grids, using sparse world coordinate information.

Applications: Covers cases where a single grid spanning whole working area on a world plane is not feasible. Also useful when high accuracy is desired.

Syntax

```
void avl::CalibrateWorldPlane_Multigrid
(
 const atl::Array<atl::Array<avl::AnnotatedPoint2D> >& inImageGrids,
 const atl::Array<atl::Array<avl::AnnotatedPoint2D> >& inLabeledWorldPoints,
 const atl::Optional<const avl::AnyCameraModel&>& inCameraModel,
 const atl::Optional<float>& inGridSpacing,
 float inGridThickness,
 bool inInvertedWorldY,
 avl::RectificationTransform& outTransform,
 atl::Optional< atl::Array<atl::Array<avl::AnnotatedPoint2D> >& > outGridWorldPoints = atl::NIL,
 atl::Optional< atl::Conditional<float>& > outComputedGridSpacing = atl::NIL,
 atl::Optional< float& > outRmsImageError = atl::NIL,
 atl::Optional< float& > outRmsWorldError = atl::NIL,
 atl::Optional< atl::Array<float>& > outMaxReprojectionErrors = atl::NIL,
 atl::Optional< atl::Array<atl::Array<avl::Segment2D> >& > outReprojectionErrorSegments = atl::NIL,
 atl::Optional< atl::Array<float>& > outGridRotations = atl::NIL,
 atl::Optional< atl::Array<avl::Vector2D>& > outGridTranslations = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImageGrids	const Array<Array<AnnotatedPoint2D> >&			Array of annotated calibration grids.
→	inLabeledWorldPoints	const Array<Array<AnnotatedPoint2D> >&			Sparse array of world coordinate points. Annotations need to correspond to those in the inImageGrid input.
→	inCameraModel	const Optional<const AnyCameraModel&>&		NIL	For undistortion of inGridImagePoints. If not supplied, the filter will assume that grids came from undistorted images.
→	inGridSpacing	const Optional<float>&	0.000001 - ∞	NIL	World distance between grid indices. Used when spacing cannot be computed from supplied inLabeledWorldPoints.
→	inGridThickness	float		0.0f	The world plane will be shifted by given amount in direction perpendicular to the grid to compensate for grid thickness.
→	inInvertedWorldY	bool		False	Set to true if world coordinate system has right-handed orientation, also known as mathematical or standard.
←	outTransform	RectificationTransform&			
←	outGridWorldPoints	Optional< Array<Array<AnnotatedPoint2D> >& >		NIL	For each grid: Array of world coordinates of the calibration grid points.
←	outComputedGridSpacing	Optional< Conditional<float>& >		NIL	World distance between grid indices. NIL when there is no enough information to compute the spacing.
←	outRmsImageError	Optional< float& >		NIL	RMS reprojection error of inImageGrids onto the image plane, in pixels. This is a partial error characterizing inaccuracies in perspective estimation, excluding the influence of world point labeling.
←	outRmsWorldError	Optional< float& >		NIL	RMS reprojection error of inLabeledGridIds onto the world plane, in world units. This is a partial error characterizing inaccuracies with labeling of world coordinate system, excluding perspective estimation.
←	outMaxReprojectionErrors	Optional< Array<float>& >		NIL	For each grid: Maximum reprojection error of inImageGrids onto the image plane, in pixels. This is a partial error characterizing inaccuracies in perspective estimation, excluding the influence of world point labeling.
←	outReprojectionErrorSegments	Optional< Array<Array<Segment2D> >& >		NIL	For each grid: Array of segments connecting inImageGrid points to their reprojections. Note that these segments depict only inaccuracies in perspective estimation, excluding inaccuracies due to the world point labeling.
←	outGridRotations	Optional< Array<float>& >		NIL	Grids' rotations on the world plane.
←	outGridTranslations	Optional< Array<Vector2D>& >		NIL	Grids' positions on the world plane.

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outGridWorldPoints**, **outComputedGridSpacing**, **outRmsImageError**, **outRmsWorldError**, **outMaxReprojectionErrors**, **outReprojectionErrorSegments**, **outGridRotations**, **outGridTranslations**.

Read more about Optional Outputs.

Description

The filter estimates the correspondence between the image plane and a "world plane" – a given planar surface in observed space. It is capable of using a multiple grids with sparse world coordinate information - i.e. grids for which only a few world plane coordinates are known.

The image plane, and thus **inImageGrids** points are assumed to be distorted, and to correct for the distortion the **inCameraModel** (calibration data) needs to be provided. The calculated result – **outTransform** contains all the information for transforming the distorted image plane to the world plane.

Currently, only pinhole camera model is supported, so the planar correspondence used is a homography.

For each grid, the algorithm requires at least four **inImageGrids** points. At least two **inLabeledWorldPoints** (in total, among all grids) are required to uniquely determine the **outTransform**.

The filter provides a few methods for judging the feasibility of calculated solution.

- The **outRmsImageError** and **outRmsWorldError** could show if the error is due to the **inImageGrids** or **inLabeledWorldPoints** errors. Model mismatch (i.e. trying to calibrate a non-planar object, e.g. wavy surface) will also result in increased reprojection errors. Large difference between **outRmsImageError** and **outMaxReprojectionErrors** could be a sign of presence of outliers in **inImageGrids** input data.
- The **outReprojectionErrorSegments** consists of segments connecting input image points to reprojected world points, and thus it can be readily used for visualization of per-point reprojection errors (excluding the errors due to incorrect world plane labeling).

Hints

- Image to world plane coordinate transform can be obtained from as few as 4 image points (**inImageGrids** points), however high accuracy calculations need a considerable amount of high quality calibration points. The calibration plane should contain about hundred points, spanning whole area of interest. Take care of proper conditions when taking the calibration images: reduce vibrations that may yield motion blur, prevent reflections from the calibration surface (ideally use diffusion lighting).
- The output reprojection errors are useful tool for assessing the feasibility of computed solution. Note, that the filter outputs two RMS metrics: **outRmsImageError** and **outRmsWorldError**. The **outRmsImageError** characterizes inaccuracies in perspective estimation, it is not influenced by world coordinate labeling. On the other hand, the **outRmsWorldError** characterizes inaccuracies with labeling of grid 2D world coordinate system, excluding perspective estimation. In order for a solution to be feasible, both these errors must be low.

Errors

List of possible exceptions:

Error type	Description
DomainError	Array inImageGrids and inLabeledWorldPoints sizes differ
DomainError	inGridSpacing needs to be positive

See Also

- CalibrateWorldPlane_Default – Finds the image to world plane transformation parameters from world plane calibration grid. World plane origin and axes are unspecified.
- CalibrateWorldPlane_OffgridOrigin – Finds the image to world plane transformation matrix, with world origin and axes specified in the image coordinates.
- CalibrateWorldPlane_Labeled – Finds the image to world plane transformation parameters using sparse world coordinate information, i.e. world coordinates are known for only a few points of the grid.
- CalibrateWorldPlane_Manual – Finds the image to world plane transformation parameters. Image and their corresponding world points are directly specified (no grid needed).
- CalibrateCamera_Pinhole – Finds the camera intrinsic parameters from calibration grids. Uses pinhole camera model (perspective camera).
- CalibrateCamera_Telecentric – Finds the telecentric camera intrinsic parameters from calibration grids.
- ImagePointsToWorldPlane – Finds the world coordinates of image Points.
- WorldPlanePointsToImage – Finds the image coordinates of world plane Points.
- CreateRectificationMap_PixelUnits – Computes a spatial map for transforming distorted images to rectified images defined in world coordinate plane. Defines the output geometry in pixels.
- CreateRectificationMap_WorldUnits – Computes a spatial map for transforming distorted images to rectified images defined in world coordinate plane. Defines the output geometry in world units.

CalibrateWorldPlane_OffgridOrigin

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the image to world plane transformation matrix, with world origin and axes specified in the image coordinates.

Applications: Rather rarely seen special cases where absolute world plane coordinates cannot be specified in terms of calibration grid, but the location of world plane origin is still required. When in doubt, use the CalibrateWorldPlane_Labeled filter.

Syntax

```
void avl::CalibrateWorldPlane_OffgridOrigin
(
 const atl::Array<avl::AnnotatedPoint2D>& inImageGrid,
 const atl::Optional<const avl::AnyCameraModel&>& inCameraModel,
 float inGridSpacing,
 float inGridThickness,
 const atl::Optional<avl::Point2D>& inWorldPlaneOrigin,
 const atl::Optional<avl::Point2D>& inWorldPlaneXAxis,
 avl::RectificationTransform& outTransform,
 atl::Optional<float&> outRmsError = atl::NIL,
 atl::Optional<float&> outMaxReprojectionError = atl::NIL,
 atl::Optional<atl::Array<avl::Segment2D>&> outReprojectionErrorSegments = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImageGrid	const Array<AnnotatedPoint2D>&			Annotated calibration grid
→	inCameraModel	const Optional<const AnyCameraModel&>&		NIL	For undistortion of inImageGrid. If not supplied, the filter will assume that grid came from undistorted image.
→	inGridSpacing	float	0.000001 - ∞	1.0f	Real-world distance between adjacent grid points.
→	inGridThickness	float		0.0f	The world plane will be shifted by given amount in direction perpendicular to the grid to compensate for grid thickness. Note, that inWorldPlaneOrigin and inWorldPlaneXAxis won't be compensated, so they need to be selected on the target world plane, and not on the grid.
→	inWorldPlaneOrigin	const Optional<Point2D>&		NIL	Sets world plane origin (as the image point). The world plane origin will lie at the specified image point. Note, that this point won't be shifted by inGridThickness parameter.
→	inWorldPlaneXAxis	const Optional<Point2D>&		NIL	Sets world plane x axis direction (as the image point). The world plane x axis will lie at the specified image point. Note, that this point won't be shifted by inGridThickness parameter.
←	outTransform	RectificationTransform&			
←	outRmsError	Optional<float&>		NIL	RMS reprojection error, in pixels.
←	outMaxReprojectionError	Optional<float&>		NIL	Maximum reprojection error, in pixels.
←	outReprojectionErrorSegments	Optional<Array<Segment2D>&>		NIL	Array of segments connecting input image points to grid reprojections.

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outRmsError**, **outMaxReprojectionError**, **outReprojectionErrorSegments**.

Read more about Optional Outputs.

Description

The filter estimates the correspondence between the image plane and a "world plane" – a given planar surface in observed space. The image plane, and thus **inImageGrid** points are assumed to be distorted, and to correct for the distortion the **inCameraModel** (calibration data) needs to be provided. The calculated result – **outTransform** contains all the information for transforming the distorted image plane to the world plane.

The **inCameraModel** is also used to define the type of the planar correspondence. For a standard projective camera (pinhole camera), the planar correspondence is a homography. If the **inCameraModel** is a telecentric camera, the planar correspondence is affine (as there are no perspective parameters in orthographic projection). If no **inCameraModel** is provided, the filter defaults to homography.

The homography requires at least four **inImageGrid** points. The affine relation requires at least three.

This filter allows for setting the world plane origin and x-axis direction in terms of input image points - see the **inWorldPlaneOrigin** and **inWorldPlaneXAxis**. Note that the origin and the axis direction does not need to lie on a grid point, it can be arbitrarily chosen.

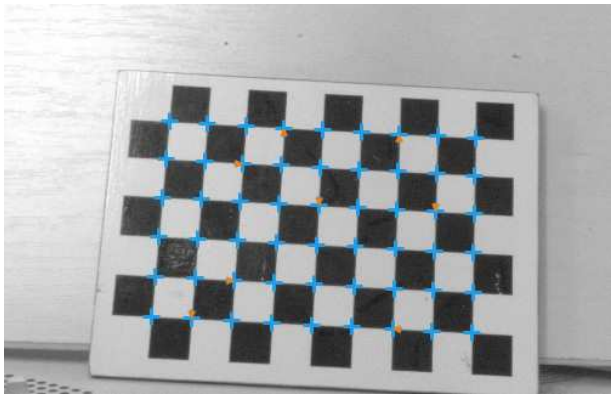
The filter provides a few methods for judging the feasibility of calculated solution.

- The **outRmsError** and **outMaxReprojectionError**. The main contributor to these values is the random noise in **inImageGrid** point positions. Model mismatch (i.e. trying to calibrate a non-planar object, e.g. wavy surface) will also result in increased reprojection errors. Large difference between **outRmsError** and **outMaxReprojectionError** could be a sign of presence of outliers in input data.
- The **outReprojectionErrorSegments** consists of segments connecting input image points to reprojected world points, and thus it can be readily used for visualization of per-point reprojection errors.

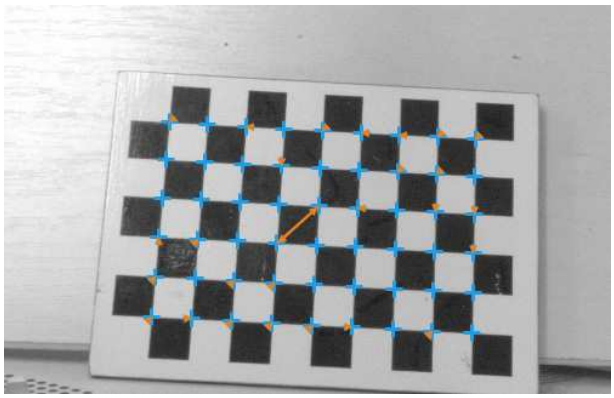
Hints

- Image to world plane coordinate transform can be obtained from as few as 4 point correspondences, however high accuracy calculations need a considerable amount of high quality calibration points. The calibration plane should contain about hundred points, spanning whole area of interest. Take care of proper conditions when taking the calibration images: reduce vibrations that may yield motion blur, prevent reflections from the calibration surface (ideally use diffusion lighting).

Examples



*Good case of image to world plane calibration using high amount of calibration points. Calibration resulted in RMS and maximum reprojection errors less than 1.0, as expected. The **outReprojectionErrorSegments** are not visible at that scale.*



*Example of a gross error. The RMS reprojection error is 17.6, max reprojection error is 91.2, source of these errors is clearly visible thanks to **outReprojectionErrorSegments**: the association of image points and their grid coordinates in **inImageGrid** is wrong.*

Errors

List of possible exceptions:

Error type	Description
DomainError	inGridSpacing needs to be positive

See Also

- CalibrateWorldPlane_Default – Finds the image to world plane transformation parameters from world plane calibration grid. World plane origin and axes are unspecified.
- CalibrateWorldPlane_Labeled – Finds the image to world plane transformation parameters using sparse world coordinate information, i.e. world coordinates are known for only a few points of the grid.
- CalibrateWorldPlane_Multigrid – Finds the image to world plane transformation parameters using multiple grids, using sparse world coordinate information.
- CalibrateWorldPlane_Manual – Finds the image to world plane transformation parameters. Image and their corresponding world points are directly specified (no grid needed).
- CalibrateCamera_Pinhole – Finds the camera intrinsic parameters from calibration grids. Uses pinhole camera model (perspective camera).
- CalibrateCamera_Telecentric – Finds the telecentric camera intrinsic parameters from calibration grids.
- ImagePointsToWorldPlane – Finds the world coordinates of image Points.
- WorldPlanePointsToImage – Finds the image coordinates of world plane Points.
- CreateRectificationMap_PixelUnits – Computes a spatial map for transforming distorted images to rectified images defined in world coordinate plane. Defines the output geometry in pixels.
- CreateRectificationMap_WorldUnits – Computes a spatial map for transforming distorted images to rectified images defined in world coordinate plane. Defines the output geometry in world units.

ConvertRectificationMap

Header: AVL.h
Namespace: avl
Module: Calibration

Converts a rectification map to a different format.

Syntax

```
void avl::ConvertRectificationMap
(
 const avl::RectificationMap& inRectificationMap,
 atl::Optional<avl::PlainType::Type> inNewPixelType,
 atl::Optional<int> inNewDepth,
 const int inNewPitchAlignment,
 avl::RectificationMap& outRectificationMap
)
```

Parameters

	Name	Type	Range	Default	Description
→	inRectificationMap	const RectificationMap&			
→	inNewPixelType	Optional<PlainType::Type>		NIL	
→	inNewDepth	Optional<int>	1 - 4	NIL	
→	inNewPitchAlignment	const int	0 - +∞	16	
←	outRectificationMap	RectificationMap&			

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect inSpatialMap map in ConvertRectificationMap.

CreateRectificationMap_Advanced

Header: AVL.h
Namespace: avl
Module: Calibration

Computes a spatial map for transforming distorted images to rectified images defined in world coordinate plane. Allows for flexible definition of output geometry.

Applications: Rarely seen special edge cases. Usually it is sufficient to use CreateRectificationMap_PixelUnits or CreateRectificationMap_WorldUnits filters.

Syntax

```
void avl::CreateRectificationMap_Advanced
(
 const avl::ImageFormat& inImageFormat,
 const avl::CreateSpatialMapUnit::Type& inCenterPointType,
 const atl::Optional<avl::Point2D>& inCenterPoint,
 const avl::CreateSpatialMapUnit::Type& inOutputSizeType,
 const atl::Optional<float>& inOutputWidth,
 const atl::Optional<float>& inOutputHeight,
 const atl::Optional<float>& inWorldScale,
 bool inInvertedWorldY,
 avl::InterpolationMethod::Type inInterpolationMethod,
 const avl::RectificationTransform& inTransform,
 avl::RectificationMap& outRectificationMap
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImageFormat	const ImageFormat&			Input image format.
→	inCenterPointType	const CreateSpatialMapUnit::Type&		Pixels	Specifies units of measurement of inCenterPoint.
→	inCenterPoint	const Optional<Point2D>&		NIL	Specifies a point which will be the center of output image. Depending on inCenterPointType it can be either defined in pixels of input image or in world units. Defaults to the center of input image.
→	inOutputSizeType	const CreateSpatialMapUnit::Type&		Pixels	Specifies units of measurement of inOutputWidth and inOutputHeight.
→	inOutputWidth	const Optional<float>&	0.001 - ∞	NIL	Specifies the size of output image. Depending on inOutputSizeType it can be either defined in pixels or world units. Defaults to the size of input image.
→	inOutputHeight	const Optional<float>&	0.001 - ∞	NIL	Specifies the size of output image. Depending on inOutputSizeType it can be either defined in pixels or world units. Defaults to the size of input image.
→	inWorldScale	const Optional<float>&	0.001 - ∞	NIL	[pix / world unit] Specifies the scale for output image. By default scale is calculated such that there will be no rescaling at the inCenterPoint.
→	inInvertedWorldY	bool		False	Set to true if world coordinate system has right-handed orientation, also known as mathematical or standard. This effectively mirrors the rectified image vertically.
→	inInterpolationMethod	InterpolationMethod::Type		Bilinear	
→	inTransform	const RectificationTransform&			Transform's camera model is needed for undistortion of image, when not supplied the generated map will assume undistorted image on input. Transform's homography is needed for transforming to given world plane, when not supplied the generated map will only remove distortion of image.
←	outRectificationMap	RectificationMap&			

Description

Creates a RectificationMap, which is used by RectifyImage filter for image *rectification* onto a defined *world plane*. Point locations on rectified images are related to the world plane (defined by **inTransform**) only by translation and scaling.

This filter allows for the most flexible definition of output canvas geometry, however for most cases filters CreateRectificationMap_PixelUnits or CreateRectificationMap_WorldUnits suffice.

The filter may be used to generate a map which only removes a lens distortion in the image – to achieve this behaviour attach a camera model directly to **inTransform** (i.e. do not provide a homography matrix).

The **inTransform** may not contain a camera model – in such case the the generated map will assume undistorted image on input.

The output image dimensions, resolution and cropping are specified by:

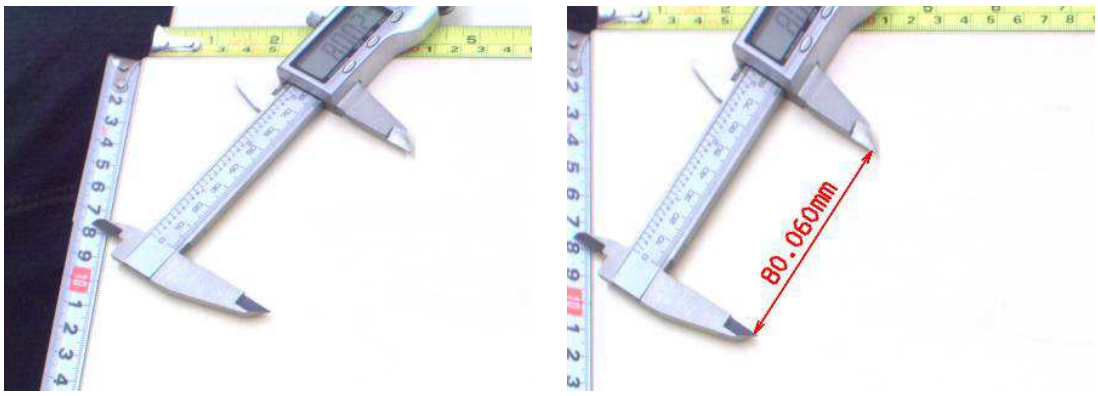
1. **inCenterPoint** - the point which will be the center of output image. Can be selected on the input image or it can be defined in world units - see **inCenterPointType**
2. **inOutputWidth**,**inOutputHeight** - the size of output image, can be defined in output pixels or in world units - see **inOutputSizeType**
3. **inWorldScale** - the scale for output image, defined in pixels per world unit.

For example, if one is interested in area spanning from (0,0) to (10,6) world coordinate, with resolution 100 pixels / world unit, then inputs would be set as follows:

- **inCenterPoint** = (5,3)
- **inCenterPointType** = WorldUnits
- **inOutputWidth** = 10
- **inOutputHeight** = 6
- **inOutputSizeType** = WorldUnits
- **inWorldScale** = 100

Auxiliary data contained inside the RectificationMap describes the relation of rectified image to the world plane.

Examples



Left: original image, as captured by a camera, with mild lens distortion present. Right: rectified image with annotated length measurement.

CreateRectificationMap_Basic

Header: AVL.h
Namespace: avl
Module: Calibration

Computes a spatial map for removing lens and perspective distortion directly from an image containing circles calibration pattern. Internally uses a pinhole camera model with polynomial lens distortion.

Applications: Quick and easy one-step calibration for basic removal of lens and perspective distortions.

Syntax

```
void avl::CreateRectificationMap_Basic
(
 const avl::Image& inImage,
 float inCircleRadius,
 float inCircleDetectionThreshold,
 atl::Array<avl::AnnotatedPoint2D>& outImageGrid,
 avl::RectificationMap& outRectificationMap
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inCircleRadius	float	1.0 - ∞		Circle radius measured in input image pixels.
→	inCircleDetectionThreshold	float	0.0 - ∞	20.0f	Detection threshold (relative to local image patch).
←	outImageGrid	Array<AnnotatedPoint2D>&			Detected grid, upon which the calibration will be based.
←	outRectificationMap	RectificationMap&			

Errors

List of possible exceptions:

Error type	Description
DomainError	No grid detected in the input image

CreateRectificationMap_PixelUnits

Header: AVL.h
Namespace: avl
Module: Calibration

Computes a spatial map for transforming distorted images to rectified images defined in world coordinate plane. Defines the output geometry in pixels.

Applications: Specification of fixed output geometry allows for constant environment even when recalibration is performed.

Syntax

```
void avl::CreateRectificationMap_PixelUnits
(
 const avl::ImageFormat& inImageFormat,
 const atl::Optional<avl::Point2D>& inCenterPoint,
 const atl::Optional<int>& inOutputWidth,
 const atl::Optional<int>& inOutputHeight,
 const atl::Optional<float>& inWorldScale,
 bool inInvertedWorldY,
 avl::InterpolationMethod::Type inInterpolationMethod,
 const avl::RectificationTransform& inTransform,
 avl::RectificationMap& outRectificationMap
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImageFormat	const ImageFormat&			Input image format.
→	inCenterPoint	const Optional<Point2D>&		NIL	Specifies a point which will be the center of output image. Defaults to the center of input image.
→	inOutputWidth	const Optional<int>&	1 - ∞	NIL	Specifies the pixel size of output image. Defaults to the size of input image.
→	inOutputHeight	const Optional<int>&	1 - ∞	NIL	Specifies the pixel size of output image. Defaults to the size of input image.
→	inWorldScale	const Optional<float>&	0.001 - ∞	NIL	[pix / world unit] Specifies the scale for output image. By default scale is calculated such that there will be no rescaling at the inCenterPoint.
→	inInvertedWorldY	bool		False	Set to true if world coordinate system has right-handed orientation, also known as mathematical or standard. This effectively mirrors the rectified image vertically.
→	inInterpolationMethod	InterpolationMethod::Type		Bilinear	
→	inTransform	const RectificationTransform&			Transform's camera model is needed for undistortion of image, when not supplied the generated map will assume undistorted image on input. Transform's homography is needed for transforming to given world plane, when not supplied the generated map will only remove distortion of image.
←	outRectificationMap	RectificationMap&			

Description

Creates a RectificationMap, which is used by RectifyImage filter for image *rectification* onto a defined *world plane*. Point locations on rectified images are related to the world plane (defined by **inTransform**) only by translation and scaling.

The filter may be used to generate a map which only removes a lens distortion in the image – to achieve this behaviour attach a camera model directly to **inTransform** (i.e. do not provide a homography matrix).

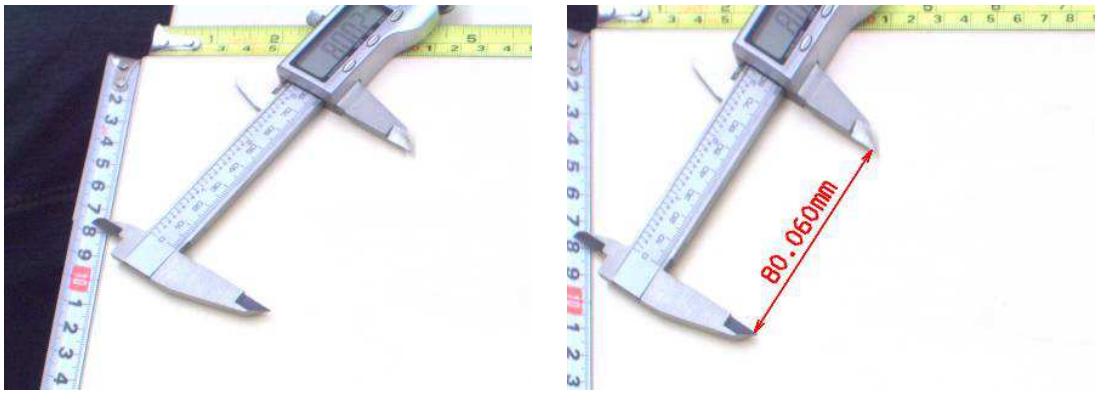
The **inTransform** may not contain a camera model – in such case the the generated map will assume undistorted image on input.

The output image dimensions, resolution and cropping are specified by:

1. **inCenterPoint** - the point on the input image, which will be the center of output image.
2. **inOutputWidth**,**inOutputHeight** - size of output image.
3. **inWorldScale** - the scale for output image, defined in pixels per world unit.

Auxiliary data contained inside the RectificationMap describes the relation of rectified image to the world plane.

Examples



Left: original image, as captured by a camera, with mild lens distortion present. Right: rectified image with annotated length measurement.

CreateRectificationMap_WorldUnits

Header: AVL.h
Namespace: avl
Module: Calibration

Computes a spatial map for transforming distorted images to rectified images defined in world coordinate plane. Defines the output geometry in world units.

Applications: Specification of output geometry in world units is especially useful in image stitching.

Syntax

```
void avl::CreateRectificationMap_WorldUnits
(
 const avl::ImageFormat& inImageFormat,
 float inLeftBound,
 float inRightBound,
 float inTopBound,
 float inBottomBound,
 const atl::Optional<float>& inWorldScale,
 bool inInvertedWorldY,
 avl::InterpolationMethod::Type inInterpolationMethod,
 const avl::RectificationTransform& inTransform,
 avl::RectificationMap& outRectificationMap
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImageFormat	const ImageFormat&			Input image format.
→	inLeftBound	float			World X coordinate of left side of generated output image.
→	inRightBound	float			World X coordinate of right side of generated output image.
→	inTopBound	float			World Y coordinate of top side of generated output image.
→	inBottomBound	float			World Y coordinate of bottom side of generated output image.
→	inWorldScale	const Optional<float>&	0.001 - ∞	NIL	[pix / world unit] Specifies the scale for output image. By default scale is calculated such that there will be no rescaling at the center of output image.
→	inInvertedWorldY	bool		False	Set to true if world coordinate system has right-handed orientation, also known as mathematical or standard. This effectively mirrors the rectified image vertically.
→	inInterpolationMethod	InterpolationMethod::Type		Bilinear	
→	inTransform	const RectificationTransform&			Transform's camera model is needed for undistortion of image, when not supplied the generated map will assume undistorted image on input. Transform's homography is needed for transforming to given world plane, when not supplied the generated map will only remove distortion of image.
←	outRectificationMap	RectificationMap&			

Description

Creates a RectificationMap, which is used by RectifyImage filter for image *rectification* onto a defined *world plane*. Point locations on rectified images are related to the world plane (defined by **inTransform**) only by translation and scaling.

The **inTransform** may not contain a camera model – in such case the the generated map will assume undistorted image on input.

The output image dimensions, resolution and cropping are specified by:

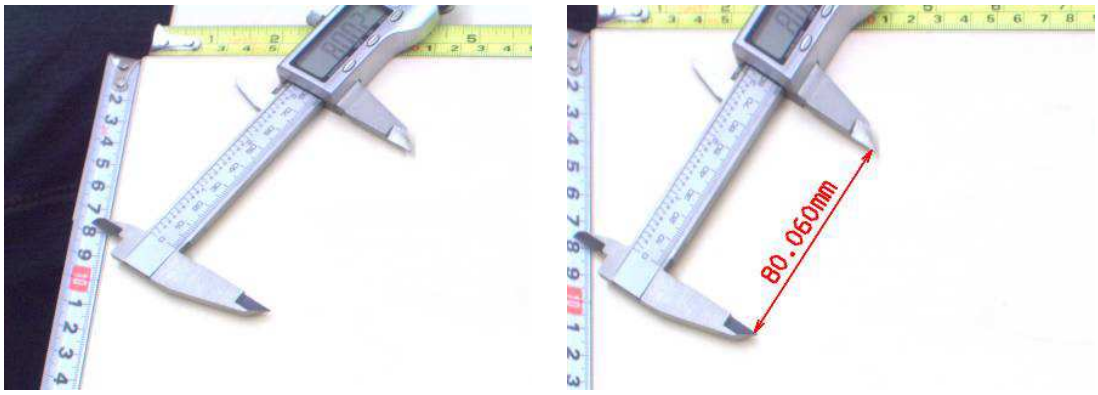
1. **inLeftBound**,**inRightBound**,**inTopBound**,**inBottomBound** - real world coordinate boundaries of the area which output image will cover.
2. **inWorldScale** - the scale for output image, defined in pixels per world unit.

For example, if one is interested in area spanning from (0,0) to (10,6) world coordinate, with resolution 100 pixels / world unit, then inputs would be set as follows:

- **inLeftBound** = 0
- **inRightBound** = 10
- **inTopBound** = 0
- **inBottomBound** = 6
- **inWorldScale** = 100

Auxiliary data contained inside the RectificationMap describes the relation of rectified image to the world plane.

Examples



Left: original image, as captured by a camera, with mild lens distortion present. Right: rectified image with annotated length measurement.

Errors

List of possible exceptions:

Error type	Description
DomainError	inLeftBound should be less than inRightBound
DomainError	inTopBound should be less than inBottomBound

DetectCalibrationGrid_Chessboard

Header: AVL.h
Namespace: avl
Module: Calibration

Detects a chessboard calibration grid on the image, and returns calibration points where 4 chessboard squares meet.

Syntax

```
void avl::DetectCalibrationGrid_Chessboard
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 const avl::Size& inBoardSize,
 bool inFastApproximate,
 atl::Array<avl::AnnotatedPoint2D>& outImageGrid
)
```

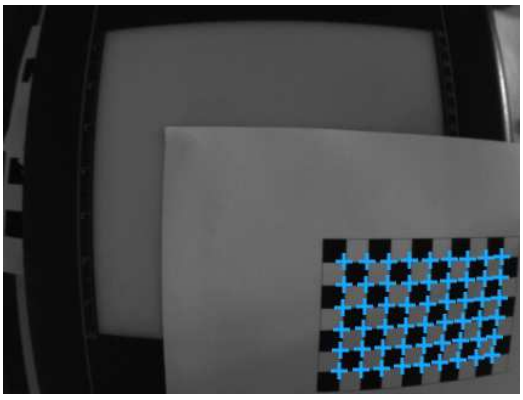
Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inBoardSize	const Size&	(Width: 0, Height: 0)	Number of checkerboard squares in X and Y dimensions.
→	inFastApproximate	bool	False	Fast filter execution, but result is approximate.
←	outImageGrid	Array<AnnotatedPoint2D>&		Detected grid

Hints

Make sure that the whole calibration grid is visible in the image. Otherwise, it will not be detected because the detection algorithm requires a few pixels wide quiet zone around the chessboard. Pay attention to the number of columns and rows, as providing misleading data may make the algorithm work incorrectly or not work at all.

Examples



*DetectCalibrationGrid_Chessboard executed with **inBoardSize** = {10,7}*

Errors

List of possible exceptions:

Error type	Description
DomainError	Input image is too small
DomainError	Input image must have pixels of UInt8 type
DomainError	Region of interest exceeds an input image.
DomainError	Specified board is too small, minimum size is 4x4

See Also

- DetectCalibrationGrid_Circles – Detects an arbitrary size symmetric circle pattern on the image.

DetectCalibrationGrid_Circles

Header: AVL.h
Namespace: avl
Module: Calibration

Detects an arbitrary size symmetric circle pattern on the image.

Syntax

```
void avl::DetectCalibrationGrid_Circles
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 float inCircleRadius,
 float inCircleDetectionThreshold,
 avl::Polarity::Type inCirclePolarity,
 atl::Array<avl::AnnotatedPoint2D>& outImageGrid,
 avl::Region& diagCirclesRegion,
 atl::Array<avl::Point2D>& diagCircleCandidates
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inCircleRadius	float	1.0 - ∞		Circle radius measured in input image pixels.
→	inCircleDetectionThreshold	float	0.0 - ∞	20.0f	Detection threshold (relative to local image patch).
→	inCirclePolarity	Polarity::Type		Any	Circle intensity with respect to background.
←	outImageGrid	Array<AnnotatedPoint2D>&			Detected grid
🔍	diagCirclesRegion	Region&			Image after thresholding, this is the circle detector input.
🔍	diagCircleCandidates	Array<Point2D>&			Detected circle centers, before the grid construction step.

Requirements

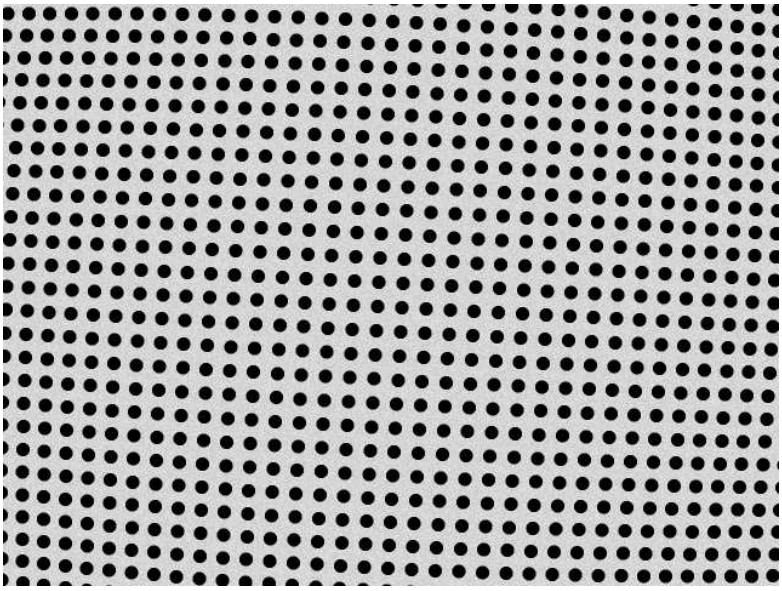
For input **inImage** only pixel formats are supported: uint8.

Read more about pixel formats in Image documentation.

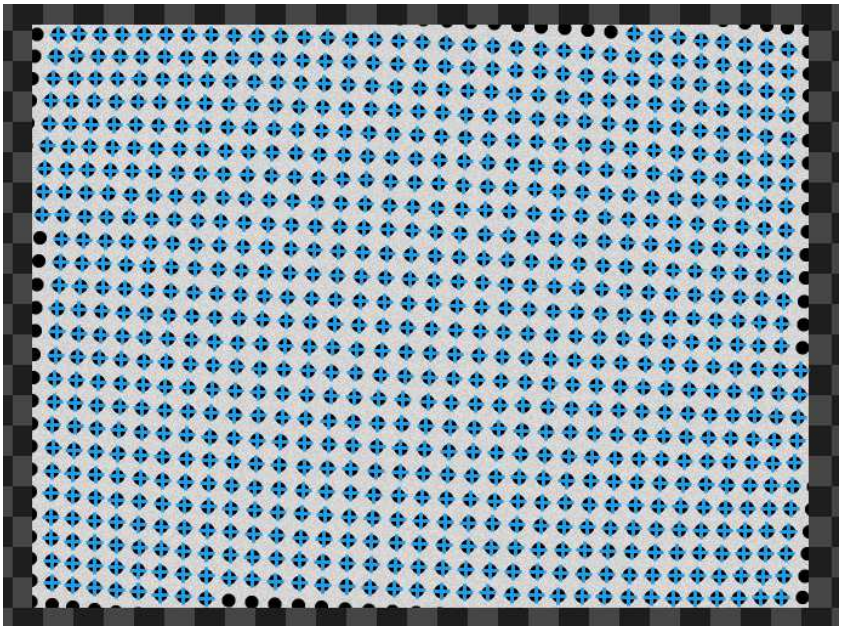
Hints

Circles may have holes. This can utilized, for example, for designating some specific coordinates on the calibration board. Note that the hole diameter must be less than a half of the circle diameter.

Examples



*Input image for DetectCalibrationGrid_Circles executed with **inCircleRadius** = 5*



Detected calibration points

Remarks

The circle pattern must be a rectangular grid, with equal spacing in both dimensions.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported inImage pixel format.
DomainError	Region of interest exceeds an input image.
DomainError	Not supported inImage pixel format in DetectCalibrationGrid_Circles. Supported formats: UInt8.

See Also

- DetectCalibrationGrid_Chessboard – Detects a chessboard calibration grid on the image, and returns calibration points where 4 chessboard squares meet.

DetectCalibrationGrid_Circles_DeprecatedDeprecated

Header: AVL.h
Namespace: avl
Module: Calibration

Detects an arbitrary size symmetric circle pattern on the image.

Applications: Camera calibration, image to world coordinates transformations.

Syntax

```
void avl::DetectCalibrationGrid_Circles_DeprecatedDeprecated
(
 const avl::Image& inImage,
 float inCircleRadius,
 float inWorldCircleSpacing,
 float inCircleDetectionMinScore,
 atl::Array<avl::Point2D>& outImagePoints,
 atl::Array<avl::Point2D>& outWorldPlanePoints,
 atl::Array<avl::Circle2D>& diagCircleCandidates
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inCircleRadius	float	1.0 - ∞		Circle radius measured in input image pixels.
→	inWorldCircleSpacing	float	0.0 - ∞	1.0f	Real-world distance between adjacent circles centers.
→	inCircleDetectionMinScore	float	0.0 - ∞	20.0f	Minimum matching score for circle detector.
←	outImagePoints	Array<Point2D>&			Image coordinates of detected calibration points.
←	outWorldPlanePoints	Array<Point2D>&			World plane coordinates of detected calibration points.
	diagCircleCandidates	Array<Circle2D>&			Detected circles, before the grid construction step

FilterGridPoints

Header: AVL.h
Namespace: avl
Module: Calibration

Select a subset of the given points that forms a grid.

Syntax

```
void avl::FilterGridPoints
(
 const atl::Array<avl::Point2D>& inPoints,
 const atl::Optional<int> inMaxAttempts,
 const atl::Optional<int> inMaxOutlierCount,
 const float inMinLength,
 const atl::Optional<float> inMaxLength,
 const float inMaxLengthRatio,
 const float inAngleRange,
 const float inBaseAspectRatioRange,
 const atl::Optional<float> inBaseAspectRatio,
 const atl::Optional<float> inBaseShear,
 const atl::Optional<float> inBaseOrientation,
 atl::Array<avl::Point2D>& outPointGrid,
 atl::Array<avl::Segment2D>& diagValidSubgraph,
 float& diagMaxLength
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoints	const Array<Point2D>&			
→	inMaxAttempts	const Optional<int>	1 - ∞	NIL	Maximum number of attempts at finding the grid
→	inMaxOutlierCount	const Optional<int>	0 - ∞	NIL	Determines how many outlier points can be present to end the search
→	inMinLength	const float	0.0 - ∞	0.0f	Minimum length of a grid segment
→	inMaxLength	const Optional<float>	0.0 - ∞	10.0f	Maximum length of a grid segment, if set to Auto it will be approximated
→	inMaxLengthRatio	const float	1.0 - 2.0	1.3f	Maximum ratio of two consecutive segments in the grid
→	inAngleRange	const float	0.0 - 45.0	16.0f	Maximum variation of angles between neighbouring grid segments in degrees
→	inBaseAspectRatioRange	const float	0.0 - 1.0	0.3f	Maximum variation of the base aspect ratio (ignores if base aspect ratio is not given)
→	inBaseAspectRatio	const Optional<float>	0.3 - 1.0	1.0f	A reference aspect ratio of the grid
→	inBaseShear	const Optional<float>	0.0 - 60.0	0.0f	A reference shear angle between grid directions that uses inMaxAngleRange as the maximum error range
→	inBaseOrientation	const Optional<float>	0.0 - 360.0	NIL	A reference orientation of one of the grids directions
←	outPointGrid	Array<Point2D>&			Detected grid
🔍	diagValidSubgraph	Array<Segment2D>&			Graph forming a valid grid
🔍	diagMaxLength	float&			Max length computed by the filter (applicable if inMaxLength is set to Auto)

Description

This is a simplified version of AnnotateGridPoints_Ransac. See the documentation of that filter for more detail.

See Also

- AnnotateGridPoints_Ransac – Select a subset of the given points that forms a grid and assign 2D indices to them.

GenerateCalibrationPoints

Header: AVL.h
Namespace: avl
Module: Calibration

Generates artificial points for camera calibration. Doesn't support distortion.

Applications: Calibration testing.

Syntax

```
void avl::GenerateCalibrationPoints
(
 int inPointsX,
 int inPointsY,
 float inWorldPointSpacing,
 const avl::Vector3D& inPlaneRotationAxis,
 float inPlaneRotationAngle,
 const avl::Vector3D& inPlaneTranslation,
 avl::CameraModelType::Type inCameraType,
 float inFocalLengthOrMagnificationX,
 atl::Optional<float> inFocalLengthOrMagnificationY,
 atl::Optional<avl::Point2D> inPrincipalPoint,
 const avl::Size& inImageSize,
 atl::Array<avl::AnnotatedPoint2D>& outImageGrid,
 atl::Array<avl::Point2D>& outWorldPoints
)
```

Parameters

	Name	Type	Default	Description
→	inPointsX	int	10	Number of points in X direction on the calibration plane.
→	inPointsY	int	10	Number of points in Y direction on the calibration plane.
→	inWorldPointSpacing	float	1.0f	Distances between consecutive points on the calibration plane.
→	inPlaneRotationAxis	const Vector3D&	(DeltaX: 0.0, DeltaY: 0.0, DeltaZ: 1.0)	Calibration plane rotation axis.
→	inPlaneRotationAngle	float	0.0f	Calibration plane rotation angle.
→	inPlaneTranslation	const Vector3D&	(DeltaX: 0.0, DeltaY: 0.0, DeltaZ: 10.0)	Calibration plane translation (applied after rotation).
→	inCameraType	CameraModelType::Type	PinholeCameraModel	Calibration camera model (pinhole, telecentric or line scan)
→	inFocalLengthOrMagnificationX	float	100.0f	
→	inFocalLengthOrMagnificationY	Optional<float>	NIL	
→	inPrincipalPoint	Optional<Point2D>	NIL	By default is on the center of output image
→	inImageSize	const Size&	(Width: 640, Height: 480)	The size of image that artificial camera captures
←	outImageGrid	Array<AnnotatedPoint2D>&		
←	outWorldPoints	Array<Point2D>&		

ImageEdgesToWorldPlane

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the world coordinates of image Edges.

Syntax

```
void avl::ImageEdgesToWorldPlane
(
 const atl::Array<avl::Edge1D>& inImageEdges,
 const avl::RectificationTransform& inTransform,
 atl::Array<avl::Edge1D>& outWorldEdges
)
```

Parameters

	Name	Type	Default	Description
→	inImageEdges	const Array<Edge1D>&		
→	inTransform	const RectificationTransform&		Transform's camera model is needed for undistortion, when not supplied the filter assumes undistorted input. Transform's homography is needed for transforming coordinates to given world plane, when not supplied the filter only removes distortion of input.
←	outWorldEdges	Array<Edge1D>&		

ImageEdgeToWorldPlane

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the world coordinates of image Edge.

Syntax

```
void avl::ImageEdgeToWorldPlane
(
 const avl::Edge1D& inImageEdge,
 const avl::RectificationTransform& inTransform,
 avl::Edge1D& outWorldEdge
)
```

Parameters

	Name	Type	Default	Description
→	inImageEdge	const Edge1D&		
→	inTransform	const RectificationTransform&		Transform's camera model is needed for undistortion, when not supplied the filter assumes undistorted input. Transform's homography is needed for transforming coordinates to given world plane, when not supplied the filter only removes distortion of input.
←	outWorldEdge	Edge1D&		

ImageGapsToWorldPlane

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the world coordinates of image Gaps.

Syntax

```
void avl::ImageGapsToWorldPlane
(
 const atl::Array<avl::Gap1D>& inImageGaps,
 const avl::RectificationTransform& inTransform,
 atl::Array<avl::Gap1D>& outWorldGaps
)
```

Parameters

	Name	Type	Default	Description
→	inImageGaps	const Array<Gap1D>&		
→	inTransform	const RectificationTransform&		Transform's camera model is needed for undistortion, when not supplied the filter assumes undistorted input. Transform's homography is needed for transforming coordinates to given world plane, when not supplied the filter only removes distortion of input.
←	outWorldGaps	Array<Gap1D>&		

ImageGapToWorldPlane

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the world coordinates of image Gap.

Syntax

```
void avl::ImageGapToWorldPlane
(
 const avl::Gap1D& inImageGap,
 const avl::RectificationTransform& inTransform,
 avl::Gap1D& outWorldGap
)
```

Parameters

	Name	Type	Default	Description
→	inImageGap	const Gap1D&		
→	inTransform	const RectificationTransform&		Transform's camera model is needed for undistortion, when not supplied the filter assumes undistorted input. Transform's homography is needed for transforming coordinates to given world plane, when not supplied the filter only removes distortion of input.
←	outWorldGap	Gap1D&		

ImagePathsToWorldPlane

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the world coordinates of image Paths.

Syntax

```
void avl::ImagePathsToWorldPlane
(
 const atl::Array<avl::Path>& inImagePaths,
 const avl::RectificationTransform& inTransform,
 atl::Array<avl::Path>& outWorldPaths
)
```

Parameters

	Name	Type	Default	Description
→	inImagePaths	const Array<Path>&		
→	inTransform	const RectificationTransform&		Transform's camera model is needed for undistortion, when not supplied the filter assumes undistorted input. Transform's homography is needed for transforming coordinates to given world plane, when not supplied the filter only removes distortion of input.
←	outWorldPaths	Array<Path>&		

ImagePathToWorldPlane

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the world coordinates of image Path.

Syntax

```
void avl::ImagePathToWorldPlane
(
 const avl::Path& inImagePath,
 const avl::RectificationTransform& inTransform,
 avl::Path& outWorldPath
)
```

Parameters

	Name	Type	Default	Description
→	inImagePath	const Path&		
→	inTransform	const RectificationTransform&		Transform's camera model is needed for undistortion, when not supplied the filter assumes undistorted input. Transform's homography is needed for transforming coordinates to given world plane, when not supplied the filter only removes distortion of input.
←	outWorldPath	Path&		

ImagePointsToWorldPlane

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the world coordinates of image Points.

Syntax

```
void avl::ImagePointsToWorldPlane
(
 const atl::Array<avl::Point2D>& inImagePoints,
 const avl::RectificationTransform& inTransform,
 atl::Array<avl::Point2D>& outWorldPoints
)
```

Parameters

	Name	Type	Default	Description
→	inImagePoints	const Array<Point2D>&		
→	inTransform	const RectificationTransform&		Transform's camera model is needed for undistortion, when not supplied the filter assumes undistorted input. Transform's homography is needed for transforming coordinates to given world plane, when not supplied the filter only removes distortion of input.
←	outWorldPoints	Array<Point2D>&		

ImagePointToWorldPlane

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the world coordinates of image Point.

Syntax

```
void avl::ImagePointToWorldPlane
(
 const avl::Point2D& inImagePoint,
 const avl::RectificationTransform& inTransform,
 avl::Point2D& outWorldPoint
)
```

Parameters

	Name	Type	Default	Description
→	inImagePoint	const Point2D&		
→	inTransform	const RectificationTransform&		Transform's camera model is needed for undistortion, when not supplied the filter assumes undistorted input. Transform's homography is needed for transforming coordinates to given world plane, when not supplied the filter only removes distortion of input.
←	outWorldPoint	Point2D&		

ImageRidgesToWorldPlane

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the world coordinates of image Ridges.

Syntax

```
void avl::ImageRidgesToWorldPlane
(
 const atl::Array<avl::Ridge1D>& inImageRidges,
 const avl::RectificationTransform& inTransform,
 atl::Array<avl::Ridge1D>& outWorldRidges
)
```

Parameters

	Name	Type	Default	Description
→	inImageRidges	const Array<Ridge1D>&		
→	inTransform	const RectificationTransform&		Transform's camera model is needed for undistortion, when not supplied the filter assumes undistorted input. Transform's homography is needed for transforming coordinates to given world plane, when not supplied the filter only removes distortion of input.
←	outWorldRidges	Array<Ridge1D>&		

ImageRidgeToWorldPlane

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the world coordinates of image Ridge.

Syntax

```
void avl::ImageRidgeToWorldPlane
(
 const avl::Ridge1D& inImageRidge,
 const avl::RectificationTransform& inTransform,
 avl::Ridge1D& outWorldRidge
)
```

Parameters

	Name	Type	Default	Description
→	inImageRidge	const Ridge1D&		
→	inTransform	const RectificationTransform&		Transform's camera model is needed for undistortion, when not supplied the filter assumes undistorted input. Transform's homography is needed for transforming coordinates to given world plane, when not supplied the filter only removes distortion of input.
←	outWorldRidge	Ridge1D&		

ImageSegmentsToWorldPlane

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the world coordinates of image Segments.

Syntax

```
void avl::ImageSegmentsToWorldPlane
(
 const atl::Array<avl::Segment2D>& inImageSegments,
 const avl::RectificationTransform& inTransform,
 atl::Array<avl::Segment2D>& outWorldSegments
)
```

Parameters

	Name	Type	Default	Description
→	inImageSegments	const Array<Segment2D>&		
→	inTransform	const RectificationTransform&		Transform's camera model is needed for undistortion, when not supplied the filter assumes undistorted input. Transform's homography is needed for transforming coordinates to given world plane, when not supplied the filter only removes distortion of input.
←	outWorldSegments	Array<Segment2D>&		

ImageSegmentToWorldPlane

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the world coordinates of image Segment.

Syntax

```
void avl::ImageSegmentToWorldPlane
(
 const avl::Segment2D& inImageSegment,
 const avl::RectificationTransform& inTransform,
 avl::Segment2D& outWorldSegment
)
```

Parameters

	Name	Type	Default	Description
→	inImageSegment	const Segment2D&		
→	inTransform	const RectificationTransform&		Transform's camera model is needed for undistortion, when not supplied the filter assumes undistorted input. Transform's homography is needed for transforming coordinates to given world plane, when not supplied the filter only removes distortion of input.
←	outWorldSegment	Segment2D&		

ImageStripesToWorldPlane

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the world coordinates of image Stripes.

Syntax

```
void avl::ImageStripesToWorldPlane
(
 const atl::Array<avl::Stripe1D>& inImageStripes,
 const avl::RectificationTransform& inTransform,
 atl::Array<avl::Stripe1D>& outWorldStripes
)
```

Parameters

	Name	Type	Default	Description
→	inImageStripes	const Array<Stripe1D>&		
→	inTransform	const RectificationTransform&		Transform's camera model is needed for undistortion, when not supplied the filter assumes undistorted input. Transform's homography is needed for transforming coordinates to given world plane, when not supplied the filter only removes distortion of input.
←	outWorldStripes	Array<Stripe1D>&		

ImageStripeToWorldPlane

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the world coordinates of image Stripe.

Syntax

```
void avl::ImageStripeToWorldPlane
(
 const avl::Stripe1D& inImageStripe,
 const avl::RectificationTransform& inTransform,
 avl::Stripe1D& outWorldStripe
)
```

Parameters

	Name	Type	Default	Description
→	inImageStripe	const Stripe1D&		
→	inTransform	const RectificationTransform&		Transform's camera model is needed for undistortion, when not supplied the filter assumes undistorted input. Transform's homography is needed for transforming coordinates to given world plane, when not supplied the filter only removes distortion of input.
←	outWorldStripe	Stripe1D&		

RectifyImage

Header: AVL.h
Namespace: avl
Module: Calibration

Applies a spatial map to distorted image transforming it to rectified image defined in world coordinates.

Syntax

```
void avl::RectifyImage
(
 const avl::Image& inImage,
 const avl::RectificationMap& inRectificationMap,
 avl::Image& outImage,
 atl::Optional< avl::Matrix& > outRectifiedTransform = atl::NIL,
 atl::Optional< avl::Point2D& > outWorldOrigin = atl::NIL,
 atl::Optional< float& > outWorldScale = atl::NIL,
 atl::Optional< float& > outWorldScaleInv = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRectificationMap	const RectificationMap&		The spatial map with associated data for world coordinates calculation on rectified image. Designed to be set with calibration GUI.
←	outImage	Image&		Remapped image.
←	outRectifiedTransform	Optional< Matrix& >	NIL	For convenient calculation of world coordinates on rectified image. Connects directly to Image...ToWorldPlane and WorldPlane...ToImage filters. The transformation is only translation + scaling.
←	outWorldOrigin	Optional< Point2D& >	NIL	Position of world origin on the rectified image.
←	outWorldScale	Optional< float& >	NIL	[pix / world unit] World scale of the rectified image.
←	outWorldScaleInv	Optional< float& >	NIL	[world unit / pix] Inverse of outWorldScale. Connects directly to inResolution input of filters such as PointToPointDistance.

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outRectifiedTransform**, **outWorldOrigin**, **outWorldScale**, **outWorldScaleInv**.

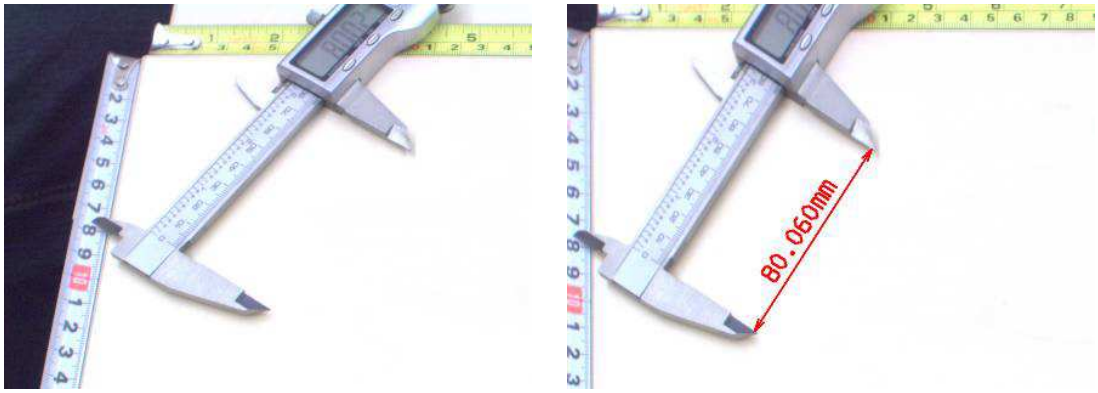
Read more about Optional Outputs.

Description

Uses a RectificationMap computed by CreateRectificationMap* filters (such as CreateRectificationMap_PixelUnits or CreateRectificationMap_WorldUnits) for image *rectification* onto a defined *world plane*. Point locations on rectified images are related to the world plane only by translation and scaling.

Auxiliary outputs **outWorldOrigin** and **outWorldScale** fully describe the relation of rectified image to world plane. For convenience the **outRectifiedTransform** output is provided, which describes the same relation, however it can be directly connected to the ImagePointsToWorldPlane filter family to obtain world coordinates of points detected on the remapped (rectified) image.

Examples



Left: original image, as captured by a camera, with mild lens distortion present. Right: rectified image with annotated length measurement.

See Also

- RemapImage – Applies a precomputed image transform, defined by a spatial map object.

ShiftWorldPlane

Header: AVL.h
Namespace: avl
Module: Calibration

Shifts world plane along its normal vector.

Applications: Correction of already calibrated world plane. Useful for avoiding recalibration when, for example, the product thickness on a conveyor belt changes by a known amount.

Syntax

```
void avl::ShiftWorldPlane
(
 const avl::RectificationTransform& inTransform,
 double inDeltaZ,
 avl::RectificationTransform& outTransform
)
```

Parameters

	Name	Type	Default	Description
→	inTransform	const RectificationTransform&		
→	inDeltaZ	double	0.0D	The world plane will be shifted by given amount in direction perpendicular to the grid. Positive translations move plane away from camera.
←	outTransform	RectificationTransform&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Camera model in inTransform is not defined.
DomainError	Homography matrix in inTransform is not defined.
DomainError	Shifting world plane with non pinhole camera models (e.g. telecentric) is not supported yet.

WorldPlaneEdgesToImage

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the image coordinates of world plane Edges.

Syntax

```
void avl::WorldPlaneEdgesToImage
(
 const atl::Array<avl::Edge1D>& inWorldEdges,
 const avl::RectificationTransform& inTransform,
 atl::Array<avl::Edge1D>& outImageEdges
)
```

Parameters

	Name	Type	Default	Description
→	inWorldEdges	const Array<Edge1D>&		
→	inTransform	const RectificationTransform&		Transform's camera model is needed for applying distortion back. Transform's homography is needed for transforming coordinates back from a given world plane.
←	outImageEdges	Array<Edge1D>&		

WorldPlaneEdgeToImage

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the image coordinates of world plane Edge.

Syntax

```
void avl::WorldPlaneEdgeToImage
(
 const avl::Edge1D& inWorldEdge,
 const avl::RectificationTransform& inTransform,
 avl::Edge1D& outImageEdge
)
```

Parameters

	Name	Type	Default	Description
→	inWorldEdge	const Edge1D&		
→	inTransform	const RectificationTransform&		Transform's camera model is needed for applying distortion back. Transform's homography is needed for transforming coordinates back from a given world plane.
←	outImageEdge	Edge1D&		

WorldPlaneGapsToImage

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the image coordinates of world plane Gaps.

Syntax

```
void avl::WorldPlaneGapsToImage
(
 const atl::Array<avl::Gap1D>& inWorldGaps,
 const avl::RectificationTransform& inTransform,
 atl::Array<avl::Gap1D>& outImageGaps
)
```

Parameters

	Name	Type	Default	Description
→	inWorldGaps	const Array<Gap1D>&		
→	inTransform	const RectificationTransform&		Transform's camera model is needed for applying distortion back. Transform's homography is needed for transforming coordinates back from a given world plane.
←	outImageGaps	Array<Gap1D>&		

WorldPlaneGapToImage

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the image coordinates of world plane Gap.

Syntax

```
void avl::WorldPlaneGapToImage
(
 const avl::Gap1D& inWorldGap,
 const avl::RectificationTransform& inTransform,
 avl::Gap1D& outImageGap
)
```

Parameters

	Name	Type	Default	Description
→	inWorldGap	const Gap1D&		
→	inTransform	const RectificationTransform&		Transform's camera model is needed for applying distortion back. Transform's homography is needed for transforming coordinates back from a given world plane.
←	outImageGap	Gap1D&		

WorldPlanePathsToImage

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the image coordinates of world plane Paths.

Syntax

```
void avl::WorldPlanePathsToImage
(
 const atl::Array<avl::Path>& inWorldPaths,
 const avl::RectificationTransform& inTransform,
 atl::Array<avl::Path>& outImagePaths
)
```

Parameters

	Name	Type	Default	Description
→	inWorldPaths	const Array<Path>&		
→	inTransform	const RectificationTransform&		Transform's camera model is needed for applying distortion back. Transform's homography is needed for transforming coordinates back from a given world plane.
←	outImagePaths	Array<Path>&		

WorldPlanePathToImage

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the image coordinates of world plane Path.

Syntax

```
void avl::WorldPlanePathToImage
(
 const avl::Path& inWorldPath,
 const avl::RectificationTransform& inTransform,
 avl::Path& outImagePath
)
```

Parameters

	Name	Type	Default	Description
→	inWorldPath	const Path&		
→	inTransform	const RectificationTransform&		Transform's camera model is needed for applying distortion back. Transform's homography is needed for transforming coordinates back from a given world plane.
←	outImagePath	Path&		

WorldPlanePointsToImage

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the image coordinates of world plane Points.

Syntax

```
void avl::WorldPlanePointsToImage
(
 const atl::Array<avl::Point2D>& inWorldPoints,
 const avl::RectificationTransform& inTransform,
 atl::Array<avl::Point2D>& outImagePoints
)
```

Parameters

	Name	Type	Default	Description
→	inWorldPoints	const Array<Point2D>&		
→	inTransform	const RectificationTransform&		Transform's camera model is needed for applying distortion back. Transform's homography is needed for transforming coordinates back from a given world plane.
←	outImagePoints	Array<Point2D>&		

WorldPlanePointToImage

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the image coordinates of world plane Point.

Syntax

```
void avl::WorldPlanePointToImage
(
 const avl::Point2D& inWorldPoint,
 const avl::RectificationTransform& inTransform,
 avl::Point2D& outImagePoint
)
```

Parameters

	Name	Type	Default	Description
→	inWorldPoint	const Point2D&		
→	inTransform	const RectificationTransform&		Transform's camera model is needed for applying distortion back. Transform's homography is needed for transforming coordinates back from a given world plane.
←	outImagePoint	Point2D&		

WorldPlaneRidgesToImage

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the image coordinates of world plane Ridges.

Syntax

```
void avl::WorldPlaneRidgesToImage
(
 const atl::Array<avl::Ridge1D>& inWorldRidges,
 const avl::RectificationTransform& inTransform,
 atl::Array<avl::Ridge1D>& outImageRidges
)
```

Parameters

	Name	Type	Default	Description
→	inWorldRidges	const Array<Ridge1D>&		
→	inTransform	const RectificationTransform&		Transform's camera model is needed for applying distortion back. Transform's homography is needed for transforming coordinates back from a given world plane.
←	outImageRidges	Array<Ridge1D>&		

WorldPlaneRidgeToImage

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the image coordinates of world plane Ridge.

Syntax

```
void avl::WorldPlaneRidgeToImage
(
 const avl::Ridge1D& inWorldRidge,
 const avl::RectificationTransform& inTransform,
 avl::Ridge1D& outImageRidge
)
```

Parameters

	Name	Type	Default	Description
→	inWorldRidge	const Ridge1D&		
→	inTransform	const RectificationTransform&		Transform's camera model is needed for applying distortion back. Transform's homography is needed for transforming coordinates back from a given world plane.
←	outImageRidge	Ridge1D&		

WorldPlaneSegmentsToImage

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the image coordinates of world plane Segments.

Syntax

```
void avl::WorldPlaneSegmentsToImage
(
 const atl::Array<avl::Segment2D>& inWorldSegments,
 const avl::RectificationTransform& inTransform,
 atl::Array<avl::Segment2D>& outImageSegments
)
```

Parameters

	Name	Type	Default	Description
→	inWorldSegments	const Array<Segment2D>&		
→	inTransform	const RectificationTransform&		Transform's camera model is needed for applying distortion back. Transform's homography is needed for transforming coordinates back from a given world plane.
←	outImageSegments	Array<Segment2D>&		

WorldPlaneSegmentToImage

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the image coordinates of world plane Segment.

Syntax

```
void avl::WorldPlaneSegmentToImage
(
 const avl::Segment2D& inWorldSegment,
 const avl::RectificationTransform& inTransform,
 avl::Segment2D& outImageSegment
)
```

Parameters

	Name	Type	Default	Description
→	inWorldSegment	const Segment2D&		
→	inTransform	const RectificationTransform&		Transform's camera model is needed for applying distortion back. Transform's homography is needed for transforming coordinates back from a given world plane.
←	outImageSegment	Segment2D&		

WorldPlaneStripesToImage

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the image coordinates of world plane Stripes.

Syntax

```
void avl::WorldPlaneStripesToImage
(
 const atl::Array<avl::Stripe1D>& inWorldStripes,
 const avl::RectificationTransform& inTransform,
 atl::Array<avl::Stripe1D>& outImageStripes
)
```

Parameters

	Name	Type	Default	Description
→	inWorldStripes	const Array<Stripe1D>&		
→	inTransform	const RectificationTransform&		Transform's camera model is needed for applying distortion back. Transform's homography is needed for transforming coordinates back from a given world plane.
←	outImageStripes	Array<Stripe1D>&		

WorldPlaneStripeToImage

Header: AVL.h
Namespace: avl
Module: Calibration

Finds the image coordinates of world plane Stripe.

Syntax

```
void avl::WorldPlaneStripeToImage
(
 const avl::Stripe1D& inWorldStripe,
 const avl::RectificationTransform& inTransform,
 avl::Stripe1D& outImageStripe
)
```

Parameters

	Name	Type	Default	Description
→	inWorldStripe	const Stripe1D&		
→	inTransform	const RectificationTransform&		Transform's camera model is needed for applying distortion back. Transform's homography is needed for transforming coordinates back from a given world plane.
←	outImageStripe	Stripe1D&		

31. Conversions

Table of content:

- Point3DArrayToPoint2DArray
- Point3DGridToPoint3DArray
- Point3DToPoint2D
- ProfileToPoint2DArray
- ProfileToRealArray
- RealArrayToPixelArray
- RealArrayToProfile
- RealToPixel
- Rectangle2DToShapeRegion
- Rectangle2DToShapeRegionDeprecated
- RegionToImage
- RegionToShapeRegion
- Ring2DToShapeRegion
- Segment2DToLine2D
- Segment2DToPath
- Segment2DToSegment3D
- Segment3DToLine3D
- Segment3DToPath
- Segment3DToSegment2D
- StringLabelToString
- StringToClassifyObjectModelDirectory
- StringToDetectAnomalies1ModelDirectory
- StringToDetectAnomalies2ModelDirectory
- StringToDetectFeaturesModelDirectory
- StringToLocatePointsModelDirectory
- StringToSegmentInstancesModelDirectory
- Stripe1DToSegment2D
- SurfaceFormatToSurfaceCoordinatesFormat
- SurfaceToImage
- SurfaceToPoint3DArray
- SurfaceToPoint3DGrid
- SurfaceToSurfaceCoordinatesFormat
- SurfaceToSurfaceFormat
- TelecentricCameraModelToAnyCameraModel
- TelecentricCameraModelToRectificationTransform
- ValueLimitsToValueLimits_f64
- ValueLimits_f64ToValueLimits
- Vector2DToVector3D
- Vector3DToVector2D
- BoolToXSightIOLevel
- XSightIOLevelToBool

AnyCameraModelToRectificationTransform

Header: AVL.h
Namespace: avl
Module: Calibration

Conversion

Syntax

```
void avl::AnyCameraModelToRectificationTransform
(
 const avl::AnyCameraModel& inModel,
 avl::RectificationTransform& outTransform
)
```

Parameters

	Name	Type	Default	Description
→	inModel	const AnyCameraModel&		
←	outTransform	RectificationTransform&		

Also in AVL Lite

BoolToHash

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a Bool to a Hash.

Syntax

```
void avl::BoolToHash
(
 const bool inBool,
 avl::Hash& outHash
)
```

Parameters

	Name	Type	Default	Description
→	inBool	const bool		
←	outHash	Hash&		

Also in AVL Lite

BoolToPassFailStatus

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a bool value to a pass/fail status.

Syntax

```
void avl::BoolToPassFailStatus
(
 const bool inBool,
 avl::PassFailStatus& outPassFailStatus
)
```

Parameters

	Name	Type	Default	Description
→	inBool	const bool		
←	outPassFailStatus	PassFailStatus&		

Box3DToBox

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts a box in 3D to a box in 2D by ignoring its Z coordinates.

Syntax

```
void avl::Box3DToBox
(
 const avl::Box3D& inBox3D,
 avl::Box& outBox2D
)
```

Parameters

	Name	Type	Default	Description
→	inBox3D	const Box3D&		
←	outBox2D	Box&		

BoxToBox3D

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts a box in 2D to a box in 3D.

Syntax

```
void avl::BoxToBox3D
(
 const avl::Box& inBox2D,
 avl::Box3D& outBox3D
)
```

Parameters

	Name	Type	Default	Description
→	inBox2D	const Box&		
←	outBox3D	Box3D&		

Also in AVL Lite

BoxToRectangle2D

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a box to a rectangle.

Syntax

```
void avl::BoxToRectangle2D
(
 const avl::Box& inBox,
 avl::Rectangle2D& outRectangle
)
```

Parameters

	Name	Type	Default	Description
→	inBox	const Box&		
←	outRectangle	Rectangle2D&		

BoxToShapeRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a box to shape region.

Syntax

```
void avl::BoxToShapeRegion
(
 const avl::Box& inBox,
 avl::ShapeRegion& outShapeRegion
)
```

Parameters

	Name	Type	Default	Description
→	inBox	const Box&		
←	outShapeRegion	ShapeRegion&		

BoxToShapeRegionDeprecated

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a box to a shape region.

Syntax

```
void avl::BoxToShapeRegionDeprecated
(
 const avl::Box& inBox,
 avl::ShapeRegionDeprecated& outShapeRegion
)
```

Parameters

	Name	Type	Default	Description
→	inBox	const Box&		
←	outShapeRegion	ShapeRegionDeprecated&		

Circle2DToCircle3D

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts a circle in 2D to a circle in 3D.

Syntax

```
void avl::Circle2DToCircle3D
(
 const avl::Circle2D& inCircle2D,
 avl::Circle3D& outCircle3D
)
```

Parameters

	Name	Type	Default	Description
→	inCircle2D	const Circle2D&		
←	outCircle3D	Circle3D&		

Also in AVL Lite

Circle2DToEllipse2D

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a circle to an ellipse.

Syntax

```
void avl::Circle2DToEllipse2D
(
 const avl::Circle2D& inCircle,
 avl::Ellipse2D& outEllipse
)
```

Parameters

	Name	Type	Default	Description
→	inCircle	const Circle2D&		
←	outEllipse	Ellipse2D&		

Also in AVL Lite

Circle2DToRing2D

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a circle to a full circle ring.

Syntax

```
void avl::Circle2DToRing2D
(
 const avl::Circle2D& inCircle,
 avl::Ring2D& outRing
)
```

Parameters

	Name	Type	Default	Description
→	inCircle	const Circle2D&		
←	outRing	Ring2D&		

Also in AVL Lite

Circle2DToShapeRegion

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a circle to region of interest.

Syntax

```
void avl::Circle2DToShapeRegion
(
 const avl::Circle2D& inCircle,
 avl::ShapeRegion& outShapeRegion
)
```

Parameters

	Name	Type	Default	Description
→	inCircle	const Circle2D&		
←	outShapeRegion	ShapeRegion&		

Also in AVL Lite

Circle2DToShapeRegionDeprecated

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a circle to a shape region.

Syntax

```
void avl::Circle2DToShapeRegionDeprecated
(
 const avl::Circle2D& inCircle,
 avl::ShapeRegionDeprecated& outShapeRegion
)
```

Parameters

	Name	Type	Default	Description
→	inCircle	const Circle2D&		
←	outShapeRegion	ShapeRegionDeprecated&		

Circle3DToCircle2D

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts a circle in 3D to a circle in 2D by ignoring its Z coordinates and tilt.

Syntax

```
void avl::Circle3DToCircle2D
(
 const avl::Circle3D& inCircle3D,
 avl::Circle2D& outCircle2D
)
```

Parameters

	Name	Type	Default	Description
→	inCircle3D	const Circle3D&		
←	outCircle2D	Circle2D&		

DeepModelToClassifyObjectModelDirectory

Header: AVL.h
Namespace: avl
Module: DL_Common

Conversion between DeepModel and ClassifyObjectModelDirectory

Syntax

```
void avl::DeepModelToClassifyObjectModelDirectory
(
 const avl::DeepModel& inDeepModel,
 avl::ClassifyObjectModelDirectory& outModelDirectory
)
```

Parameters

	Name	Type	Default	Description
→	inDeepModel	const DeepModel&		
←	outModelDirectory	ClassifyObjectModelDirectory&		

DeepModelToDetectAnomalies1ModelDirectory

Header: AVL.h
Namespace: avl
Module: DL_Common

Conversion between DeepModel and DetectAnomalies1ModelDirectory

Syntax

```
void avl::DeepModelToDetectAnomalies1ModelDirectory
(
 const avl::DeepModel& inDeepModel,
 avl::DetectAnomalies1ModelDirectory& outModelDirectory
)
```

Parameters

	Name	Type	Default	Description
→	inDeepModel	const DeepModel&		
←	outModelDirectory	DetectAnomalies1ModelDirectory&		

DeepModelToDetectAnomalies2ModelDirectory

Header: AVL.h
Namespace: avl
Module: DL_Common

Conversion between DeepModel and DetectAnomalies2ModelDirectory

Syntax

```
void avl::DeepModelToDetectAnomalies2ModelDirectory
(
 const avl::DeepModel& inDeepModel,
 avl::DetectAnomalies2ModelDirectory& outModelDirectory
)
```

Parameters

	Name	Type	Default	Description
→	inDeepModel	const DeepModel&		
←	outModelDirectory	DetectAnomalies2ModelDirectory&		

DeepModelToDetectFeaturesModelDirectory

Header: AVL.h
Namespace: avl
Module: DL_Common

Conversion between DeepModel and DetectFeaturesModelDirectory

Syntax

```
void avl::DeepModelToDetectFeaturesModelDirectory
(
 const avl::DeepModel& inDeepModel,
 avl::DetectFeaturesModelDirectory& outModelDirectory
)
```

Parameters

	Name	Type	Default	Description
→	inDeepModel	const DeepModel&		
←	outModelDirectory	DetectFeaturesModelDirectory&		

DeepModelToLocatePointsModelDirectory

Header: AVL.h
Namespace: avl
Module: DL_Common

Conversion between DeepModel and LocatePointsModelDirectory

Syntax

```
void avl::DeepModelToLocatePointsModelDirectory
(
 const avl::DeepModel& inDeepModel,
 avl::LocatePointsModelDirectory& outModelDirectory
)
```

Parameters

	Name	Type	Default	Description
→	inDeepModel	const DeepModel&		
←	outModelDirectory	LocatePointsModelDirectory&		

DeepModelToSegmentInstancesModelDirectory

Header: AVL.h
Namespace: avl
Module: DL_Common

Conversion between DeepModel and SegmentInstancesModelDirectory

Syntax

```
void avl::DeepModelToSegmentInstancesModelDirectory
(
 const avl::DeepModel& inDeepModel,
 avl::SegmentInstancesModelDirectory& outModelDirectory
)
```

Parameters

	Name	Type	Default	Description
→	inDeepModel	const DeepModel&		
←	outModelDirectory	SegmentInstancesModelDirectory&		

DirectoryToClassifyObjectModelDirectory

Header: AVL.h
Namespace: avl
Module: DL_Common

Conversion between Directory and ClassifyObjectModelDirectory

Syntax

```
void avl::DirectoryToClassifyObjectModelDirectory
(
 const atl::Directory& inDirectory,
 avl::ClassifyObjectModelDirectory& outModelDirectory
)
```

Parameters

	Name	Type	Default	Description
→	inDirectory	const Directory&		
←	outModelDirectory	ClassifyObjectModelDirectory&		

DirectoryToDetectAnomalies1ModelDirectory

Header: AVL.h
Namespace: avl
Module: DL_Common

Conversion between Directory and DetectAnomalies1ModelDirectory

Syntax

```
void avl::DirectoryToDetectAnomalies1ModelDirectory
(
 const atl::Directory& inDirectory,
 avl::DetectAnomalies1ModelDirectory& outModelDirectory
)
```

Parameters

	Name	Type	Default	Description
→	inDirectory	const Directory&		
←	outModelDirectory	DetectAnomalies1ModelDirectory&		

DirectoryToDetectAnomalies2ModelDirectory

Header: AVL.h
Namespace: avl
Module: DL_Common

Conversion between Directory and DetectAnomalies2ModelDirectory

Syntax

```
void avl::DirectoryToDetectAnomalies2ModelDirectory
(
 const atl::Directory& inDirectory,
 avl::DetectAnomalies2ModelDirectory& outModelDirectory
)
```

Parameters

	Name	Type	Default	Description
→	inDirectory	const Directory&		
←	outModelDirectory	DetectAnomalies2ModelDirectory&		

DirectoryToDetectFeaturesModelDirectory

Header: AVL.h
Namespace: avl
Module: DL_Common

Conversion between Directory and DetectFeaturesModelDirectory

Syntax

```
void avl::DirectoryToDetectFeaturesModelDirectory
(
 const atl::Directory& inDirectory,
 avl::DetectFeaturesModelDirectory& outModelDirectory
)
```

Parameters

	Name	Type	Default	Description
→	inDirectory	const Directory&		
←	outModelDirectory	DetectFeaturesModelDirectory&		

DirectoryToLocatePointsModelDirectory

Header: AVL.h
Namespace: avl
Module: DL_Common

Conversion between Directory and LocatePointsModelDirectory

Syntax

```
void avl::DirectoryToLocatePointsModelDirectory
(
 const atl::Directory& inDirectory,
 avl::LocatePointsModelDirectory& outModelDirectory
)
```

Parameters

	Name	Type	Default	Description
→	inDirectory	const Directory&		
←	outModelDirectory	LocatePointsModelDirectory&		

DirectoryToSegmentInstancesModelDirectory

Header: AVL.h
Namespace: avl
Module: DL_Common

Conversion between Directory and SegmentInstancesModelDirectory

Syntax

```
void avl::DirectoryToSegmentInstancesModelDirectory
(
 const atl::Directory& inDirectory,
 avl::SegmentInstancesModelDirectory& outModelDirectory
)
```

Parameters

	Name	Type	Default	Description
→	inDirectory	const Directory&		
←	outModelDirectory	SegmentInstancesModelDirectory&		

Also in AVL Lite

Ellipse2DToShapeRegion

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts an ellipse to region of interest.

Syntax

```
void avl::Ellipse2DToShapeRegion
(
 const avl::Ellipse2D& inEllipse,
 avl::ShapeRegion& outShapeRegion
)
```

Parameters

	Name	Type	Default	Description
→	inEllipse	const Ellipse2D&		
←	outShapeRegion	ShapeRegion&		

Also in AVL Lite

Gap1DToSegment2D

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a gap to a segment.

Syntax

```
void avl::Gap1DToSegment2D
(
 const avl::Gap1D& inGap,
 avl::Segment2D& outSegment
)
```

Parameters

	Name	Type	Default	Description
→	inGap	const Gap1D&		
←	outSegment	Segment2D&		

Also in AVL Lite

HeatmapToImage

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a heatmap to an image.

Syntax

```
void avl::HeatmapToImage
(
 const avl::Heatmap& inHeatmap,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inHeatmap	const Heatmap&		
←	outImage	Image&		Output image

Also in AVL Lite

ImageToImageFormat

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts an image to an image format.

Syntax

```
void avl::ImageToImageFormat
(
 const avl::Image& inImage,
 avl::ImageFormat& outImageFormat
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
←	outImageFormat	ImageFormat&		

ImageToSize

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts an image to a size.

Syntax

```
void avl::ImageToSize
(
 const avl::Image& inImage,
 avl::Size& outSize
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
←	outSize	Size&		

IntegerToHash

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts an Integer to a Hash.

Syntax

```
void avl::IntegerToHash
(
 const int inInteger,
 avl::Hash& outHash
)
```

Parameters

	Name	Type	Default	Description
→	inInteger	const int		
←	outHash	Hash&		

IntegerToPixel

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts an integer value to pixel.

Syntax

```
void avl::IntegerToPixel
(
 const int inValue,
 avl::Pixel& outPixel
)
```

Parameters

	Name	Type	Default	Description
→	inValue	const int		
←	outPixel	Pixel&		

Line2DToLine3D

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts a line in 2D to a line in 3D.

Syntax

```
void avl::Line2DToLine3D
(
 const avl::Line2D& inLine2D,
 avl::Line3D& outLine3D
)
```

Parameters

	Name	Type	Default	Description
→	inLine2D	const Line2D&		
←	outLine3D	Line3D&		

Line3DToLine2D

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts a line in 3D to a line in 2D by ignoring its Z coordinates.

Syntax

```
void avl::Line3DToLine2D
(
 const avl::Line3D& inLine3D,
 avl::Line2D& outLine2D
)
```

Parameters

	Name	Type	Default	Description
→	inLine3D	const Line3D&		
←	outLine2D	Line2D&		

LineScanCameraModelToAnyCameraModel

Header: AVL.h
Namespace: avl
Module: Calibration

Conversion

Syntax

```
void avl::LineScanCameraModelToAnyCameraModel
(
 const avl::LineScanCameraModel& inModel,
 avl::AnyCameraModel& outModel
)
```

Parameters

	Name	Type	Default	Description
→	inModel	const LineScanCameraModel&		
←	outModel	AnyCameraModel&		

LineScanCameraModelToRectificationTransform

Header: AVL.h
Namespace: avl
Module: Calibration

Conversion

Syntax

```
void avl::LineScanCameraModelToRectificationTransform
(
 const avl::LineScanCameraModel& inModel,
 avl::RectificationTransform& outTransform
)
```

Parameters

	Name	Type	Default	Description
→	inModel	const LineScanCameraModel&		
←	outTransform	RectificationTransform&		

Also in AVL Lite

LocationArrayToPoint2DArray

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts an array of locations to an array of points.

Syntax

```
void avl::LocationArrayToPoint2DArray
(
 const atl::Array<avl::Location>& inLocationArray,
 atl::Array<avl::Point2D>& outPointArray
)
```

Parameters

	Name	Type	Default	Description
→	inLocationArray	const Array<Location>&		
←	outPointArray	Array<Point2D>&		

Also in AVL Lite

LocationToPoint2D

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a location to its center point.

Syntax

```
void avl::LocationToPoint2D
(
 const avl::Location& inLocation,
 avl::Point2D& outPoint
)
```

Parameters

	Name	Type	Default	Description
→	inLocation	const Location&		
←	outPoint	Point2D&		

MatrixToRectificationTransform

Header: AVL.h
Namespace: avl
Module: Calibration

Conversion

Syntax

```
void avl::MatrixToRectificationTransform
(
 const avl::Matrix& inHomography,
 avl::RectificationTransform& outTransform
)
```

Parameters

	Name	Type	Default	Description
→	inHomography	const Matrix&		
←	outTransform	RectificationTransform&		

Errors

List of possible exceptions:

Error type	Description
DomainError	inHomography matrix must be a 3x3 matrix

Also in AVL Lite

PassFailStatusToBool

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a pass/fail status to a bool value.

Syntax

```
void avl::PassFailStatusToBool
(
 const avl::PassFailStatus& inPassFailStatus,
 bool& outBool
)
```

Parameters

	Name	Type	Default	Description
→	inPassFailStatus	const PassFailStatus&		
←	outBool	bool&		

Also in AVL Lite

PassFailStatusToHash

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a PassFailStatus object to a Hash.

Syntax

```
void avl::PassFailStatusToHash
(
 const avl::PassFailStatus& inPassFailStatus,
 avl::Hash& outHash
)
```

Parameters

	Name	Type	Default	Description
→	inPassFailStatus	const PassFailStatus&		
←	outHash	Hash&		

Also in AVL Lite

PathToPoint2DArray

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a path to an array of points.

Syntax

```
void avl::PathToPoint2DArray
(
 const avl::Path& inPath,
 atl::Array<avl::Point2D >& outPoints
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
←	outPoints	Array<Point2D >&		

PinholeCameraModelToAnyCameraModel

Header: AVL.h
Namespace: avl
Module: Calibration

Conversion

Syntax

```
void avl::PinholeCameraModelToAnyCameraModel
(
 const avl::PinholeCameraModel& inModel,
 avl::AnyCameraModel& outModel
)
```

Parameters

	Name	Type	Default	Description
→	inModel	const PinholeCameraModel&		
←	outModel	AnyCameraModel&		

PinholeCameraModelToRectificationTransform

Header: AVL.h
Namespace: avl
Module: Calibration

Conversion

Syntax

```
void avl::PinholeCameraModelToRectificationTransform
(
 const avl::PinholeCameraModel& inModel,
 avl::RectificationTransform& outTransform
)
```

Parameters

	Name	Type	Default	Description
→	inModel	const PinholeCameraModel&		
←	outTransform	RectificationTransform&		

Also in AVL Lite

Point2DArrayToLocationArray

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts an array of points to an array of locations.

Syntax

```
void avl::Point2DArrayToLocationArray
(
 const atl::Array<avl::Point2D>& inPointArray,
 atl::Array<avl::Location>& outLocationArray
)
```

Parameters

	Name	Type	Default	Description
→	inPointArray	const Array<Point2D>&		
←	outLocationArray	Array<Location>&		

Also in AVL Lite

Point2DToLocation

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a point to a location its contained within.

Syntax

```
void avl::Point2DToLocation
(
 const avl::Point2D& inPoint,
 avl::Location& outLocation
)
```

Parameters

	Name	Type	Default	Description
→	inPoint	const Point2D&		
←	outLocation	Location&		

Point2DToPoint3D

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts a point in 2D to a point in 3D.

Syntax

```
void avl::Point2DToPoint3D
(
 const avl::Point2D& inPoint2D,
 avl::Point3D& outPoint3D
)
```

Parameters

	Name	Type	Default	Description
→	inPoint2D	const Point2D&		
←	outPoint3D	Point3D&		

Point3DArrayToPoint2DArray

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts an array of points in 3D to an array of points in 2D by ignoring their Z coordinates.

Syntax

```
void avl::Point3DArrayToPoint2DArray
(
 const atl::Array<avl::Point3D>& inPoint3DArray,
 atl::Array<avl::Point2D>& outPoint2DArray
)
```

Parameters

	Name	Type	Default	Description
→	inPoint3DArray	const Array<Point3D>&		
←	outPoint2DArray	Array<Point2D>&		

Point3DGridToPoint3DArray

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Converts a grid of points to an array of points removing nonexistent ones.

Syntax

```
void avl::Point3DGridToPoint3DArray
(
 const avl::Point3DGrid& inPoint3DGrid,
 atl::Array<avl::Point3D>& outPoint3DArray
)
```

Parameters

	Name	Type	Default	Description
→	inPoint3DGrid	const Point3DGrid&		
←	outPoint3DArray	Array<Point3D>&		

Point3DToPoint2D

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts a point in 3D to a point in 2D by ignoring its Z coordinate.

Syntax

```
void avl::Point3DToPoint2D
(
 const avl::Point3D& inPoint3D,
 avl::Point2D& outPoint2D
)
```

Parameters

	Name	Type	Default	Description
→	inPoint3D	const Point3D&		
←	outPoint2D	Point2D&		

Also in AVL Lite

ProfileToPoint2DArray

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a profile to an array of points.

Syntax

```
void avl::ProfileToPoint2DArray
(
 const avl::Profile& inProfile,
 atl::Array<avl::Point2D>& outPoints
)
```

Parameters

	Name	Type	Default	Description
→	inProfile	const Profile&		Input profile
←	outPoints	Array<Point2D>&		

Also in AVL Lite

ProfileToRealArray

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a profile to an array of real numbers.

Syntax

```
void avl::ProfileToRealArray
(
 const avl::Profile& inProfile,
 atl::Array<float>& outArray
)
```

Parameters

	Name	Type	Default	Description
→	inProfile	const Profile&		Input profile
←	outArray	Array<float>&		

Also in AVL Lite

RealArrayToPixelArray

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts an array of real values to an array of pixels.

Syntax

```
void avl::RealArrayToPixelArray
(
 const atl::Array<float>& inValues,
 atl::Array<avl::Pixel>& outPixels
)
```

Parameters

	Name	Type	Default	Description
→	inValues	const Array<float>&		
←	outPixels	Array<Pixel>&		

Also in AVL Lite

RealArrayToProfile

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts an array of real numbers to a profile.

Syntax

```
void avl::RealArrayToProfile
(
 const atl::Array<float>& inArray,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Default	Description
→	inArray	const Array<float>&		
←	outProfile	Profile&		Output profile

Also in AVL Lite

RealToPixel

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a real value to pixel.

Syntax

```
void avl::RealToPixel
(
 const float inValue,
 avl::Pixel& outPixel
)
```

Parameters

	Name	Type	Default	Description
→	inValue	const float		
←	outPixel	Pixel&		

Also in AVL Lite

Rectangle2DToShapeRegion

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a rectangle to region of interest.

Syntax

```
void avl::Rectangle2DToShapeRegion
(
 const avl::Rectangle2D& inRectangle,
 avl::ShapeRegion& outShapeRegion
)
```

Parameters

	Name	Type	Default	Description
→	inRectangle	const Rectangle2D&		
←	outShapeRegion	ShapeRegion&		

Rectangle2DToShapeRegionDeprecated

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a rectangle to a shape region.

Syntax

```
void avl::Rectangle2DToShapeRegionDeprecated
(
 const avl::Rectangle2D& inRectangle,
 avl::ShapeRegionDeprecated& outShapeRegion
)
```

Parameters

	Name	Type	Default	Description
→	inRectangle	const Rectangle2D&		
←	outShapeRegion	ShapeRegionDeprecated&		

RegionToImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a region to an image.

Syntax

```
void avl::RegionToImage
(
 const avl::Region& inRegion,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outImage	Image&		Output image

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

RegionToShapeRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a region to region of interest.

Syntax

```
void avl::RegionToShapeRegion
(
 const avl::Region& inRegion,
 avl::ShapeRegion& outShapeRegion
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outShapeRegion	ShapeRegion&		

Also in AVL Lite

Ring2DToShapeRegion

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a ring to region of interest.

Syntax

```
void avl::Ring2DToShapeRegion
(
 const avl::Ring2D& inRing,
 avl::ShapeRegion& outShapeRegion
)
```

Parameters

	Name	Type	Default	Description
→	inRing	const Ring2D&		
←	outShapeRegion	ShapeRegion&		

Also in AVL Lite

Segment2DToLine2D

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a segment to a line.

Syntax

```
void avl::Segment2DToLine2D
(
 const avl::Segment2D& inSegment,
 avl::Line2D& outLine
)
```

Parameters

	Name	Type	Default	Description
→	inSegment	const Segment2D&		
←	outLine	Line2D&		

Also in AVL Lite

Segment2DToPath

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a segment to a path.

Syntax

```
void avl::Segment2DToPath
(
 const avl::Segment2D& inSegment,
 avl::Path& outPath
)
```

Parameters

	Name	Type	Default	Description
→	inSegment	const Segment2D&		
←	outPath	Path&		Output path

Segment2DToSegment3D

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts a segment in 2D to a segment in 3D.

Syntax

```
void avl::Segment2DToSegment3D
(
 const avl::Segment2D& inSegment2D,
 avl::Segment3D& outSegment3D
)
```

Parameters

	Name	Type	Default	Description
→	inSegment2D	const Segment2D&		
←	outSegment3D	Segment3D&		

Segment3DToLine3D

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts a segment in 3D to a line in 3D.

Syntax

```
void avl::Segment3DToLine3D
(
 const avl::Segment3D& inSegment,
 avl::Line3D& outLine
)
```

Parameters

	Name	Type	Default	Description
→	inSegment	const Segment3D&		
←	outLine	Line3D&		

Segment3DToPath

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts a segment in 3D to a path in 2D by ignoring its Z coordinates.

Syntax

```
void avl::Segment3DToPath
(
 const avl::Segment3D& inSegment3D,
 avl::Path& outPath
)
```

Parameters

	Name	Type	Default	Description
→	inSegment3D	const Segment3D&		
←	outPath	Path&		Output path

Segment3DToSegment2D

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts a segment in 3D to a segment in 2D by ignoring its Z coordinates.

Syntax

```
void avl::Segment3DToSegment2D
(
 const avl::Segment3D& inSegment3D,
 avl::Segment2D& outSegment2D
)
```

Parameters

	Name	Type	Default	Description
→	inSegment3D	const Segment3D&		
←	outSegment2D	Segment2D&		

Also in AVL Lite

StringLabelToString

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a StringLabel object to a String.

Syntax

```
void avl::StringLabelToString
(
 const avl::StringLabel& inStringLabel,
 atl::String& outString
)
```

Parameters

	Name	Type	Default	Description
→	inStringLabel	const StringLabel&		
←	outString	String&		

StringToClassifyObjectModelDirectory

Header: AVL.h
Namespace: avl
Module: DL_Common

Conversion between Directory and ClassifyObjectModelDirectory

Syntax

```
void avl::StringToClassifyObjectModelDirectory
(
 const atl::String& inString,
 avl::ClassifyObjectModelDirectory& outModelDirectory
)
```

Parameters

	Name	Type	Default	Description
→	inString	const String&		
←	outModelDirectory	ClassifyObjectModelDirectory&		

StringToDetectAnomalies1ModelDirectory

Header: AVL.h
Namespace: avl
Module: DL_Common

Conversion between String and DetectAnomalies1ModelDirectory

Syntax

```
void avl::StringToDetectAnomalies1ModelDirectory
(
 const atl::String& inString,
 avl::DetectAnomalies1ModelDirectory& outModelDirectory
)
```

Parameters

	Name	Type	Default	Description
→	inString	const String&		
←	outModelDirectory	DetectAnomalies1ModelDirectory&		

StringToDetectAnomalies2ModelDirectory

Header: AVL.h
Namespace: avl
Module: DL_Common

Conversion between String and DetectAnomalies2ModelDirectory

Syntax

```
void avl::StringToDetectAnomalies2ModelDirectory
(
 const atl::String& inString,
 avl::DetectAnomalies2ModelDirectory& outModelDirectory
)
```

Parameters

	Name	Type	Default	Description
→	inString	const String&		
←	outModelDirectory	DetectAnomalies2ModelDirectory&		

StringToDetectFeaturesModelDirectory

Header: AVL.h
Namespace: avl
Module: DL_Common

Conversion between String and DetectFeaturesModelDirectory

Syntax

```
void avl::StringToDetectFeaturesModelDirectory
(
 const atl::String& inString,
 avl::DetectFeaturesModelDirectory& outModelDirectory
)
```

Parameters

	Name	Type	Default	Description
→	inString	const String&		
←	outModelDirectory	DetectFeaturesModelDirectory&		

StringToLocatePointsModelDirectory

Header: AVL.h
Namespace: avl
Module: DL_Common

Conversion between String and LocatePointsModelDirectory

Syntax

```
void avl::StringToLocatePointsModelDirectory
(
 const atl::String& inString,
 avl::LocatePointsModelDirectory& outModelDirectory
)
```

Parameters

	Name	Type	Default	Description
→	inString	const String&		
←	outModelDirectory	LocatePointsModelDirectory&		

StringToSegmentInstancesModelDirectory

Header: AVL.h
Namespace: avl
Module: DL_Common

Conversion between String and SegmentInstancesModelDirectory

Syntax

```
void avl::StringToSegmentInstancesModelDirectory
(
 const atl::String& inString,
 avl::SegmentInstancesModelDirectory& outModelDirectory
)
```

Parameters

	Name	Type	Default	Description
→	inString	const String&		
←	outModelDirectory	SegmentInstancesModelDirectory&		

Also in AVL Lite

Stripe1DToSegment2D

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a stripe to a segment.

Syntax

```
void avl::Stripe1DToSegment2D
(
 const avl::Stripe1D& inStripe,
 avl::Segment2D& outSegment
)
```

Parameters

	Name	Type	Default	Description
→	inStripe	const Stripe1D&		
←	outSegment	Segment2D&		

SurfaceFormatToSurfaceCoordinatesFormat

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Converts a surface format to a surface coordinates format.

Syntax

```
void avl::SurfaceFormatToSurfaceCoordinatesFormat
(
 const avl::SurfaceFormat& inSurfaceFormat,
 avl::SurfaceCoordinatesFormat& outSurfaceCoordinatesFormat
)
```

Parameters

	Name	Type	Default	Description
→	inSurfaceFormat	const SurfaceFormat&		
←	outSurfaceCoordinatesFormat	SurfaceCoordinatesFormat&		

SurfaceToImage

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Converts a surface object to an image.

Syntax

```
void avl::SurfaceToImage
(
 const avl::Surface& inSurface,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inSurface	const Surface&		Input surface
←	outImage	Image&		Output image

SurfaceToPoint3DArray

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Converts a surface object to an array of points removing nonexistent ones.

Syntax

```
void avl::SurfaceToPoint3DArray
(
 const avl::Surface& inSurface,
 atl::Array<avl::Point3D>& outPoint3DArray
)
```

Parameters

	Name	Type	Default	Description
→	inSurface	const Surface&		Input surface
←	outPoint3DArray	Array<Point3D>&		Output point array

SurfaceToPoint3DGrid

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Converts a surface object to a grid of points.

Syntax

```
void avl::SurfaceToPoint3DGrid
(
 const avl::Surface& inSurface,
 avl::Point3DGrid& outPoint3DGrid
)
```

Parameters

	Name	Type	Default	Description
→	inSurface	const Surface&		Input surface
←	outPoint3DGrid	Point3DGrid&		Output point grid

SurfaceToSurfaceCoordinatesFormat

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Converts a surface to a surface coordinates format.

Syntax

```
void avl::SurfaceToSurfaceCoordinatesFormat
(
 const avl::Surface& inSurface,
 avl::SurfaceCoordinatesFormat& outSurfaceCoordinatesFormat
)
```

Parameters

	Name	Type	Default	Description
→	inSurface	const Surface&		
←	outSurfaceCoordinatesFormat	SurfaceCoordinatesFormat&		

SurfaceToSurfaceFormat

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Converts a surface to a surface format.

Syntax

```
void avl::SurfaceToSurfaceFormat
(
 const avl::Surface& inSurface,
 avl::SurfaceFormat& outSurfaceFormat
)
```

Parameters

	Name	Type	Default	Description
→	inSurface	const Surface&		
←	outSurfaceFormat	SurfaceFormat&		

TelecentricCameraModelToAnyCameraModel

Header: AVL.h
Namespace: avl
Module: Calibration

Conversion

Syntax

```
void avl::TelecentricCameraModelToAnyCameraModel
(
 const avl::TelecentricCameraModel& inModel,
 avl::AnyCameraModel& outModel
)
```

Parameters

	Name	Type	Default	Description
→	inModel	const TelecentricCameraModel&		
←	outModel	AnyCameraModel&		

TelecentricCameraModelToRectificationTransform

Header: AVL.h
Namespace: avl
Module: Calibration

Conversion

Syntax

```
void avl::TelecentricCameraModelToRectificationTransform
(
 const avl::TelecentricCameraModel& inModel,
 avl::RectificationTransform& outTransform
)
```

Parameters

	Name	Type	Default	Description
→	inModel	const TelecentricCameraModel&		
←	outTransform	RectificationTransform&		

Also in AVL Lite

ValueLimitsToValueLimits_f64

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts value limits of different types.

Syntax

```
void avl::ValueLimitsToValueLimits_f64
(
 const avl::ValueLimits& inValueLimits,
 avl::ValueLimits_f64& outValueLimits_f64
)
```

Parameters

	Name	Type	Default	Description
→	inValueLimits	const ValueLimits&		
←	outValueLimits_f64	ValueLimits_f64&		

ValueLimits_f64ToValueLimits

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts value limits of different types.

Syntax

```
void avl::ValueLimits_f64ToValueLimits
(
 const avl::ValueLimits_f64& inValueLimits_f64,
 avl::ValueLimits& outValueLimits
)
```

Parameters

	Name	Type	Default	Description
→	inValueLimits_f64	const ValueLimits_f64&		
←	outValueLimits	ValueLimits&		

Vector2DToVector3D

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts a vector in 2D to a vector in 3D.

Syntax

```
void avl::Vector2DToVector3D
(
 const avl::Vector2D& inVector2D,
 avl::Vector3D& outVector3D
)
```

Parameters

	Name	Type	Default	Description
→	inVector2D	const Vector2D&		
←	outVector3D	Vector3D&		

Vector3DToVector2D

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts a vector in 3D to a vector in 2D by ignoring its Z coordinates.

Syntax

```
void avl::Vector3DToVector2D
(
 const avl::Vector3D& inVector3D,
 avl::Vector2D& outVector2D
)
```

Parameters

	Name	Type	Default	Description
→	inVector3D	const Vector3D&		
←	outVector2D	Vector2D&		

BoolToXSightIOLevel

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Converts port level to a bool.

Syntax

```
void avl::BoolToXSightIOLevel
(
 bool inBool,
 avl::XSightIOLevel::Type& outIOLevel
)
```

Parameters

	Name	Type	Default	Description
→	inBool	bool		
←	outIOLevel	XSightIOLevel::Type&		

XSightIOLevelToBool

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Converts port level to a bool.

Syntax

```
void avl::XSightIOLevelToBool
(
 avl::XSightIOLevel::Type inIOLevel,
 bool& outBool
)
```

Parameters

	Name	Type	Default	Description
→	inIOLevel	XSightIOLevel::Type		
←	outBool	bool&		

32. Path Basics

Table of content:

AppendPointToPath

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Appends a point to a path.

Syntax

```
void avl::AppendPointToPath
(
 avl::Path& ioPath,
 const avl::Point2D& inPoint
)
```

Parameters

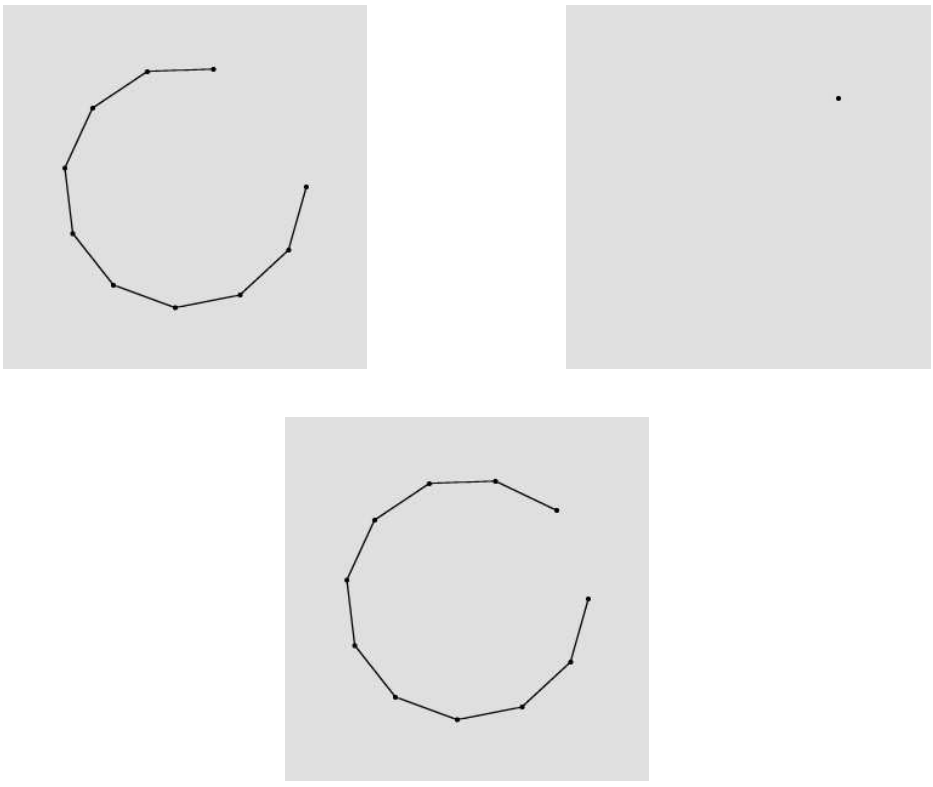
	Name	Type	Default	Description
	ioPath	Path&		
	inPoint	const Point2D&		Input point

Description

The operation extends an open path by adding a single point at its end (the point becomes new end point of a path).

When a closed path is passed to the filter, an error with appropriate description occurs.

Examples



***AppendPointToPath** run on sample path and point.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Closed path on input in AppendPointToPath.

See Also

- ConcatenatePaths – Joins up to four open paths.

ClosePath

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Adds the segment connecting the last point with the first one in a path.

Syntax

```
void avl::ClosePath
(
 avl::Path& ioPath
)
```

Parameters

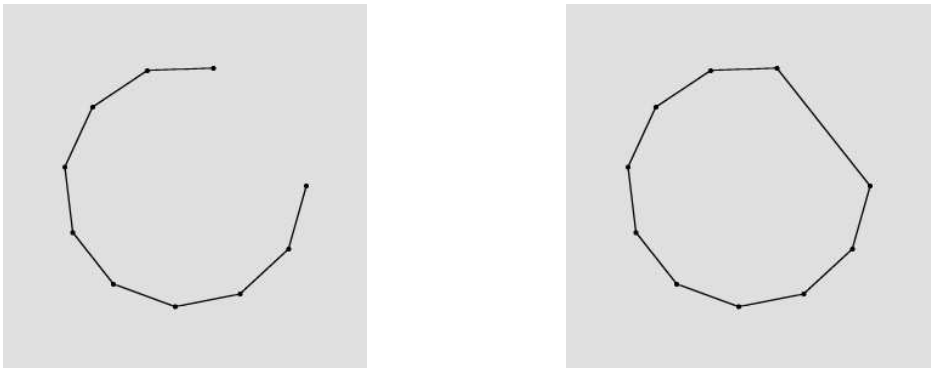
	Name	Type	Default	Description
	ioPath	Path&		

Description

The operation extends an open path by adding a segment between its first point and the last one, therefore producing a closed path.

If a closed path is passed, the filter passes it onto output without any interference.

Examples



***ClosePath** run on a sample path.*

See Also

- OpenPath – Removes the segment connecting the last point with the first one in a path.

CreateArcPath

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates an open path containing cocircular, equidistant points.

Syntax

```
void avl::CreateArcPath
(
 const avl::Arc2D& inArc,
 int inPointCount,
 avl::Path& outPath
)
```

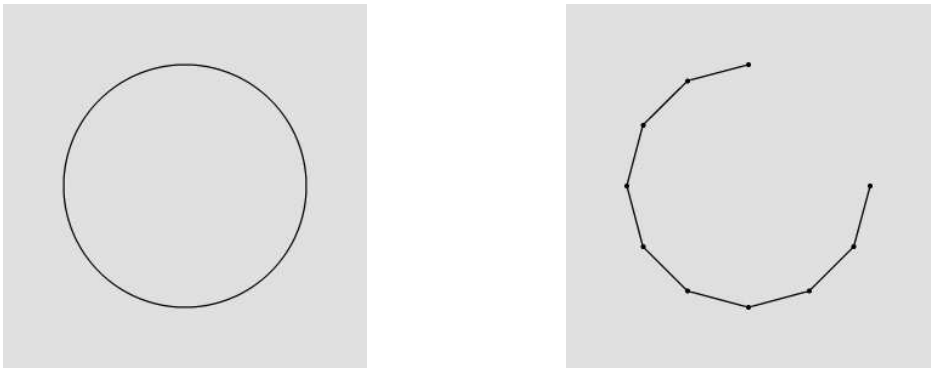
Parameters

	Name	Type	Range	Default	Description
➡	inArc	const Arc2D&			
➡	inPointCount	int	2 - ∞	8	Number of points in the resulting path
⬅	outPath	Path&			Output path

Description

The operation produces an open path that consists of **inPointCount** equidistant points selected along the **inArc** arc.

Examples



CreateArcPath *run using a sample arc with* ***inPointCount*** *= 10.*

See Also

- CreateSegmentPath – Creates an open path containing collinear, equidistant points.
- CreateCirclePath – Creates a closed path containing cocircular, equidistant points.

CreateCirclePath

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a closed path containing cocircular, equidistant points.

Syntax

```
void avl::CreateCirclePath
(
 const avl::Circle2D& inCircle,
 int inPointCount,
 avl::Path& outPath
)
```

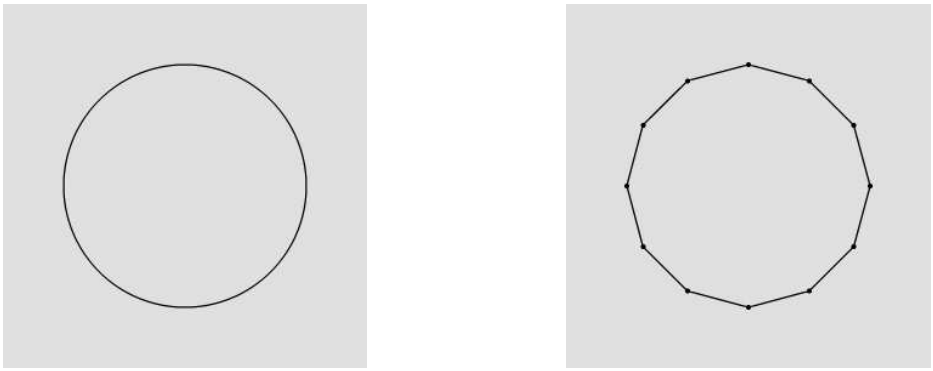
Parameters

	Name	Type	Range	Default	Description
➡	inCircle	const Circle2D&			
➡	inPointCount	int	2 - ∞	8	Number of points in the resulting path
⬅	outPath	Path&			Output path

Description

The operation produces a closed path that consist of **inPointCount** equidistant points selected along the **inCircle** circumference.

Examples



CreateCirclePath *run using a sample circle with* ***inPointCount*** *= 12.*

See Also

- CreateSegmentPath – Creates an open path containing collinear, equidistant points.
- CreateArcPath – Creates an open path containing cocircular, equidistant points.

CreateEllipsePath

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a closed path containing elliptical, equidistant points.

Syntax

```
void avl::CreateEllipsePath
(
 const avl::Point2D& inCenter,
 float inRadiusX,
 float inRadiusY,
 int inPointCount,
 avl::Path& outPath
)
```

Parameters

	Name	Type	Range	Default	Description
→	inCenter	const Point2D&			
→	inRadiusX	float	0.0 - ∞		
→	inRadiusY	float	0.0 - ∞		
→	inPointCount	int	2 - ∞	8	Number of points in the resulting path
←	outPath	Path&			Output path

CreateRectanglePath

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a closed path containing four vertices of rectangle.

Syntax

```
void avl::CreateRectanglePath
(
 const avl::Rectangle2D& inRectangle,
 avl::Path& outPath
)
```

Parameters

	Name	Type	Default	Description
→	inRectangle	const Rectangle2D&		
←	outPath	Path&		Output path

CreateSegmentPath

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates an open path containing collinear, equidistant points.

Syntax

```
void avl::CreateSegmentPath
(
 const avl::Point2D& inBegin,
 const avl::Point2D& inEnd,
 int inPointCount,
 avl::Path& outPath
)
```

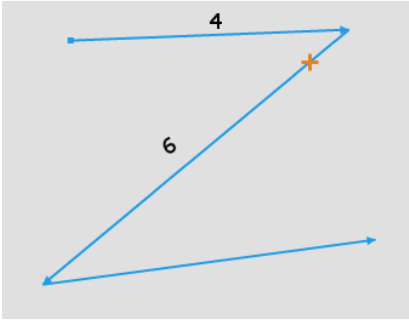
Parameters

	Name	Type	Range	Default	Description
→	inBegin	const Point2D&			First point of the path
→	inEnd	const Point2D&			Last point of the path
→	inPointCount	int	2 - ∞	2	Number of points in the created path
←	outPath	Path&			Output path

Description

The operation produces an open path that consist of **inPointCount** equidistant points selected along the segment between **inBegin** and **inEnd**.

Examples



CreateSegmentPath *run with* ***inPointCount*** *= 5.*

See Also

- CreateArcPath – Creates an open path containing cocircular, equidistant points.
- CreateCirclePath – Creates a closed path containing cocircular, equidistant points.

GetPathCharacteristicPoint

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns the selected characteristic point of a path.

Syntax

```
void avl::GetPathCharacteristicPoint
(
 const avl::Path& inPath,
 int inIndex,
 const bool inInverse,
 avl::Point2D& outPoint
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPath	const Path&			Input path
→	inIndex	int	0 - ∞		Index of a point of the input path
→	inInverse	const bool			Reversed order of points
←	outPoint	Point2D&			

Errors

List of possible exceptions:

Error type	Description
DomainError	Index out of range in GetPathCharacteristicPoint.

GetPathCharacteristicPoint_Interpolated

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns linear interpolation between two consecutive characteristic points of a path.

Syntax

```
void avl::GetPathCharacteristicPoint_Interpolated
(
 const avl::Path& inPath,
 float inIndex,
 avl::Point2D& outPoint
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPath	const Path&			Input path
→	inIndex	float	0.0 - ∞		Real-valued point index; fractional values result in interpolation
←	outPoint	Point2D&			

Errors

List of possible exceptions:

Error type	Description
DomainError	Index out of range in GetPathCharacteristicPoint_Interpolated.

GetPathSegment

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns the selected segment of a path.

Syntax

```
void avl::GetPathSegment
(
 const avl::Path& inPath,
 int inIndex,
 const bool inInverse,
 avl::Segment2D& outSegment
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPath	const Path&			Input path
→	inIndex	int	0 - ∞		
→	inInverse	const bool			Reversed order of segments
←	outSegment	Segment2D&			

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty path in GetPathSegment.
DomainError	Incorrect segment index in GetPathSegment.

GetPointOnPath

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns a path point of desired distance (measured along path) from the first point of the path.

Syntax

```
void avl::GetPointOnPath
(
 const avl::Path& inPath,
 const float inDistanceAlongPath,
 avl::Point2D& outPoint
)
```

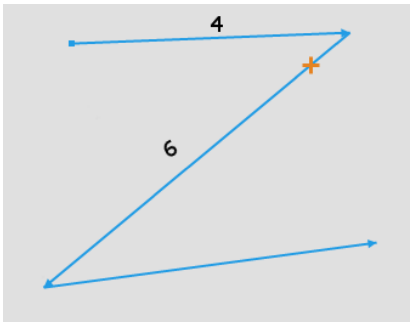
Parameters

	Name	Type	Range	Default	Description
→	inPath	const Path&			Input path
→	inDistanceAlongPath	const float	0.0 - ∞		Distance along path from the first characteristic point to the desired point
←	outPoint	Point2D&			

Description

Returns a path point of desired distance (measured along path) from the first point of the path.

Examples



Point with a distance of 5 from path origin.

Errors

List of possible exceptions:

Error type	Description
DomainError	Desired distance along path exceeds the path length in GetPointOnPath.

InsertToPath

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Inserts a new point to a path at a specified location.

Syntax

```
void avl::InsertToPath
(
 avl::Path& ioPath,
 const avl::Point2D& inPoint,
 int inIndex,
 bool inInverse
)
```

Parameters

	Name	Type	Range	Default	Description
	ioPath	Path&			
	inPoint	const Point2D&			Input point to be inserted.
	inIndex	int	0 - ∞		Input index within the ioPath at which the inValue will be placed.
	inInverse	bool		False	Determines if the indices are counted from beginning or from end of the input path.

Errors

List of possible exceptions:

Error type	Description
DomainError	Index out of range in InsertToPath.

OpenPath

Header: AVL.h
Namespace: avl
Module: FoundationLite

Removes the segment connecting the last point with the first one in a path.

Syntax

```
void avl::OpenPath
(
 avl::Path& ioPath
)
```

Parameters

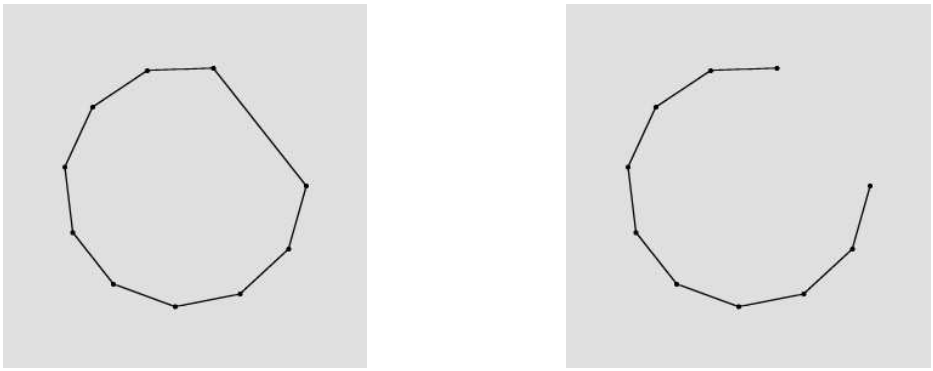
	Name	Type	Default	Description
	ioPath	Path&		

Description

The operation reduces a closed path by removing a segment between its first point and the last one, therefore producing an open path.

If an open path is passed, the filter passes it onto output without any interference.

Examples



OpenPath *run on a sample path.*

See Also

- ClosePath – Adds the segment connecting the last point with the first one in a path.

RemovePointFromPath

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Removes a point from a path.

Syntax

```
void avl::RemovePointFromPath
(
 avl::Path& ioPath,
 int inIndex,
 bool inInverse
)
```

Parameters

	Name	Type	Default	Description
	ioPath	Path&		
	inIndex	int		
	inInverse	bool		

Errors

List of possible exceptions:

Error type	Description
DomainError	Index out of range in RemovePointFromPath.

SetPathCharacteristicPoint

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Modifies the value of a selected characteristic point.

Syntax

```
void avl::SetPathCharacteristicPoint
(
 avl::Path& ioPath,
 int inIndex,
 const bool inInverse,
 const avl::Point2D& inPoint
)
```

Parameters

	Name	Type	Range	Default	Description
	ioPath	Path&			
	inIndex	int	0 - ∞		Index of a point of the input path
	inInverse	const bool			Reversed order of points
	inPoint	const Point2D&			

Errors

List of possible exceptions:

Error type	Description
DomainError	Index out of range in SetPathCharacteristicPoint.

SkipEmptyPath

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

If the input path has at least one point, then it is copied to the output; otherwise Nil is returned.

Applications: Secures against domain errors caused by empty paths, e.g. just before the PathBoundingBox filter is to be invoked.

Syntax

```
void avl::SkipEmptyPath
(
 const avl::Path& inPath,
 atl::Conditional<avl::Path>& outNotEmptyPath,
 bool& outIsNotEmpty
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
←	outNotEmptyPath	Conditional<Path>&		A copy of the input path, if it is not empty, or Nil otherwise
←	outIsNotEmpty	bool&		Indication if the input path is not empty

SkipNotPolygon

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

If the input path is closed and has no self-intersections, then it is copied to the output; otherwise Nil is returned.

Syntax

```
void avl::SkipNotPolygon
(
 const avl::Path& inPath,
 atl::Conditional<avl::Path>& outPolygon,
 bool& outIsPolygon
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
←	outPolygon	Conditional<Path>&		
←	outIsPolygon	bool&		

Subpath

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates an open path from a continuous subsequence of the input path's points.

Syntax

```
void avl::Subpath
(
 const avl::Path& inPath,
 const int inStart,
 const int inPointCount,
 avl::Path& outPath
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPath	const Path&			Input path
→	inStart	const int	0 - ∞		Index of the first point of the subpath
→	inPointCount	const int	0 - ∞	1	Number of points in the subpath
←	outPath	Path&			The resulting subpath

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect index range on input in Subpath.

33. Principal Component Analysis

Table of content:

ApplyPCATransform

Header: AVL.h
Namespace: avl
Module: FoundationPro

Applies previously obtained Principal Component Analysis (PCA) transformation coefficients to new data.

Syntax

```
void avl::ApplyPCATransform
(
 const avl::Matrix& inMatrix,
 const avl::PCAModel& inPCAModel,
 avl::Matrix& outTransformedMatrix
)
```

Parameters

	Name	Type	Default	Description
→	inMatrix	const Matrix&		Input data with variables in columns and examples in rows.
→	inPCAModel	const PCAModel&		Previously created PCA model to apply to data provided in inMatrix.
←	outTransformedMatrix	Matrix&		Transformed inMatrix.

Errors

List of possible exceptions:

Error type	Description
DomainError	Malformed inPCAModel - MeanVector and StandardDeviationVector are not row-vectors in ApplyPCATransform.
DomainError	Malformed inPCAModel - MeanVector and StandardDeviationVector have to have the same length in ApplyPCATransform.
DomainError	PCAModel does not match - inMatrix column count does not match in ApplyPCATransform.
DomainError	PCAModel does not match - PCAFeatureVector dimensions does not correspond to inMatrix dimensions in ApplyPCATransform.
DomainError	PCAModel does not match - StandardDeviationVector length is different then inMatrix column count in ApplyPCATransform.

CreatePCATransform

Header: AVL.h
Namespace: avl
Module: FoundationPro

Performs the Principal Component Analysis (PCA) on provided data, creates the feature vector and normalization coefficients (mean and standard deviation of variables).

Syntax

```
void avl::CreatePCATransform
(
 const avl::Matrix& inMatrix,
 const int inDimensions,
 atl::Optional<float> inVarianceToLeave,
 avl::PCAModel& outPCAModel,
 avl::Matrix& outTransformedMatrix,
 avl::Matrix& diagCovarianceMatrix,
 avl::Matrix& diagNormalizedData,
 atl::Array<int>& diagUsedFeatureIndices
)
```

Parameters

	Name	Type	Range	Default	Description
→	inMatrix	const Matrix&			Input data, where variables are in column, and examples are in rows.
→	inDimensions	const int	1 - ∞		How many data dimensions (variables) to be left in transformed data.
→	inVarianceToLeave	Optional<float>	0.0 - 1.0	0.95f	How many of input data variance should be left in transformed data; overrides inDimensions input.
←	outPCAModel	PCAModel&			Resulting PCA model.
←	outTransformedMatrix	Matrix&			Transformed inMatrix with reduced dimensionality.
🔍	diagCovarianceMatrix	Matrix&			Covariance matrix of input data.
🔍	diagNormalizedData	Matrix&			Input data after normalization: scaling and centering.
🔍	diagUsedFeatureIndices	Array<int>&			Indices of columns in inMatrix, which were used as Principal Components.

Errors

List of possible exceptions:

Error type	Description
DomainError	Cannot conduct PCA on empty matrix in CreatePCATransform.
DomainError	Cannot conduct principal component analysis for 1-row data set in CreatePCATransform.
DomainError	Cannot reduce data to less than 1 dimension in CreatePCATransform.
DomainError	Could not compute eigenvalues and/or eigenvectors in CreatePCATransform.
DomainError	inDimensions has to be lesser then inMatrix column count in PCA filter in CreatePCATransform.
DomainError	The process did not converge in CreatePCATransform.
DomainError	The provided data did not satisfy the prerequisites in CreatePCATransform.

MatrixDeterminant

Header: AVL.h
Namespace: avl
Module: FoundationPro

Find the determinant of a square matrix.

Syntax

```
void avl::MatrixDeterminant
(
 const avl::Matrix& inMatrix,
 float& outDeterminant
)
```

Parameters

	Name	Type	Default	Description
→	inMatrix	const Matrix&		
←	outDeterminant	float&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty matrix in MatrixDeterminant.
DomainError	Matrix dimensions incompatible in MatrixDeterminant.

MatrixPseudoEigenvectors

Header: AVL.h
Namespace: avl
Module: FoundationPro

Find the pseudo-eigenvalues and pseudo-eigenvectors of a symmetrical square matrix.

Syntax

```
void avl::MatrixPseudoEigenvectors
(
 const avl::Matrix& inMatrix,
 avl::Matrix& outEigenvectors,
 atl::Array<float>& outEigenvalues
)
```

Parameters

	Name	Type	Default	Description
→	inMatrix	const Matrix&		
←	outEigenvectors	Matrix&		Row matrix of eigenvectors
←	outEigenvalues	Array<float>&		Array of eigenvalues

Errors

List of possible exceptions:

Error type	Description
DomainError	Matrix empty on input in MatrixPseudoEigenvectors!
DomainError	Matrix with non-matching dimensions on input in MatrixPseudoEigenvectors!

NormalizeMatrixData

Header: AVL.h
Namespace: avl
Module: FoundationPro

Treats Matrix as a data frame, where examples are in rows while columns represent features, and normalizes the data by subtracting mean from each column and dividing it by its standard deviation.

Syntax

```
void avl::NormalizeMatrixData
(
 const avl::Matrix& inMatrix,
 atl::Optional<const avl::Matrix&> inMeansVector,
 atl::Optional<const avl::Matrix&> inStandardDeviationsVector,
 avl::Matrix& outNormalizedMatrix,
 avl::Matrix& outMeansVector,
 avl::Matrix& outStandardDeviationsVector
)
```

Parameters

	Name	Type	Default	Description
→	inMatrix	const Matrix&		Input data frame.
→	inMeansVector	Optional<const Matrix&>	NIL	If provided, will be used in normalization of inMatrix.
→	inStandardDeviationsVector	Optional<const Matrix&>	NIL	If provided, will be used in normalization of inMatrix.
←	outNormalizedMatrix	Matrix&		Resulting normalized matrix.
←	outMeansVector	Matrix&		Resulting Means vector - copy of inMeansVector, or calculated Means, if inMeansVector was set NIL.
←	outStandardDeviationsVector	Matrix&		Resulting StdDevs vector - copy of inStandardDeviationsVector, or calculated Means, if inStandardDeviationsVector was set NIL.

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect matrix dimensions in NormalizeData.
DomainError	One can provide both Means and StdDevs vector or none of them.

ReversePCATransform

Header: AVL.h
Namespace: avl
Module: FoundationPro

Reverses Principal Component Analysis (PCA) process. Can be used to transform data back to original feature space.

Syntax

```
void avl::ReversePCATransform
(
 const avl::Matrix& inTransformedMatrix,
 const avl::PCAModel& inPCAModel,
 avl::Matrix& outMatrix
)
```

Parameters

	Name	Type	Default	Description
→	inTransformedMatrix	const Matrix&		Data that was transformed earlier.
→	inPCAModel	const PCAModel&		PCA model used to create inTransformedMatrix.
←	outMatrix	Matrix&		inTransformedMatrix transformed back to its original feature space.

Errors

List of possible exceptions:

Error type	Description
DomainError	Feature vector of inPCAModel and Transformed Data matrices dimensions do not correspond to each other in ReversePCATransform.
DomainError	inMeanVector and inStandardDeviationVector have incorrect size for inTransformedMatrix provided in ReversePCATransform.
DomainError	Malformed inPCAModel - StandardDeviationVector is not row-vector in ReversePCATransform.
DomainError	Malformed inPCAModel - MeanVector is not row-vector in ReversePCATransform.
DomainError	Malformed inPCAModel - uneven vector sizes in ReversePCATransform.

34. Image Look Up Tables

Table of content:

ApplyPixelLut

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Applies previously created Look Up transformation to provided image.

Syntax

```
void avl::ApplyPixelLut
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 const avl::PixelLut& inLut,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Image to which LUT transformation will be applied
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inLut	const PixelLut&		LUT object, which defines transformation
←	outImage	Image&		Transformed image

Requirements

For input **inImage** only pixel formats are supported: 1□uint8, 1□int8, 1□uint16, 1□int16, 2□uint8, 2□int8, 2□uint16, 2□int16, 3□uint8, 3□int8, 3□uint16, 3□int16, 4□uint8, 4□int8, 4□uint16, 4□int16.

Read more about pixel formats in Image documentation.

Description

This filter will apply previously created (by i.e. CreatePowerLut) LookUp Table transformation to **inImage**. Type and depth of resulting **outImage** are determined by properties of **inLut**, which were fixed during creation of LUT.

Generally speaking, LookUp transform should only be applied to monochromatic images. However, ApplyPixelLut allows for transformation of color images, under special conditions:

- **inLut** has to be created for single channel output
- **inImage** has to be multichannel (its depth has to be greater than 1)

If those conditions are met, processing steps are as follows: input image is being split into separate channels, every channel is processed by **inLut**, resulting processed channels are being merged into **outImage**.

It also possible to apply multichannel LUT to multichannel image, however number of channels (depth) must match, i.e. one can apply LUT of depth 3 to color, 3-channel image.

ApplyPixelLut can work with images of type: INT8, UINT8, INT16, UINT16, INT32, REAL. Operation time can be longer with INT32 and REAL, because LUT is not cached in this case - all calculations are performed during each execution, as opposed to cached execution, when only copying of appropriate values is conducted.

Remarks

Please note, that one cannot apply custom LUT created for INT32 and REAL types. This would result in storing enormous array of data in-memory and would not be feasible.

Standard operations like PowerImage, CorrectGamma and LogarithmImage for images of type Int32 and Real are available in Image Point Transforms category.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in ApplyPixelLut.
DomainError	Not supported inImage pixel format in ApplyPixelLut. Supported formats: 1xUInt8, 1xInt8, 1xUInt16, 1xInt16, 2xUInt8, 2xInt8, 2xUInt16, 2xInt16, 3xUInt8, 3xInt8, 3xUInt16, 3xInt16, 4xUInt8, 4xInt8, 4xUInt16, 4xInt16.

See Also

- ApplyPixelLut – Applies previously created Look Up transformation to provided image.
- CreatePowerLut – Creates Look Up Table for power operation on image pixels.
- CreateGammaCorrectionLut – Creates Look Up Table for gamma correction operation on image pixels.
- CreateLogarithmLut – Creates Look Up Table for logarithm operation on image pixels.

CreateCustomLut

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Creates Look Up Table for user provided operation on image pixels.

Syntax

```
void avl::CreateCustomLut
(
 const LutOperatorType& inLutFunction,
 const avl::PlainType::Type& inOutputType,
 atl::Optional<void*> inUserData,
 avl::PixelLut& outLut
)
```

Parameters

	Name	Type	Default	Description
→	inLutFunction	const LutOperatorType&		
→	inOutputType	const PlainType::Type&		
→	inUserData	Optional<void*>	NIL	Data passed as the second parameter to inLutFunction
←	outLut	PixelLut&		

Description

This function creates custom Look Up Table from function provided by user in form of callable.

Function to use can be passed as pointer to function, callable object (object with operator() defined) or lambda expression. Only requirement for such callable is for it to accept single parameter of type float and return value of type float.

Value passed to provided function is value of pixel. Value returned from provided function is value demanded from LUT being created.

CreateCustomLut will apply provided function on all possible values of **inOutputType** type and cache them in internal structure. Such PixelLut object can be then passed to ApplyPixelLut.

CreateCustomLut can create Look Up Tables for the following types: Int8, UInt8, Int16, UInt16.

Remarks

Standard operations like PowerImage, CorrectGamma and LogarithmImage for images of type Int32 and Real are available in Image Point Transforms category.

Errors

List of possible exceptions:

Error type	Description
DomainError	Cannot create LUT with empty operator in CreateCustomLut.

See Also

- ApplyPixelLut – Applies previously created Look Up transformation to provided image.
- CreatePowerLut – Creates Look Up Table for power operation on image pixels.
- CreateGammaCorrectionLut – Creates Look Up Table for gamma correction operation on image pixels.
- CreateLogarithmLut – Creates Look Up Table for logarithm operation on image pixels.

CreateGammaCorrectionLut

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Creates Look Up Table for gamma correction operation on image pixels.

Syntax

```
void avl::CreateGammaCorrectionLut
(
 const float inValue,
 const avl::PlainType::Type& inOutputType,
 avl::PixelLut& outLut
)
```

Parameters

	Name	Type	Default	Description
→	inValue	const float	2.0f	
→	inOutputType	const PlainType::Type&		
←	outLut	PixelLut&		

Description

This operation will create LookUp Table with precalculated values of gamma correction on all pixel values of type defined in **inOutputType**. Such LUT can be reused across multiple ApplyPixelLut usages. This filter can create LUTs only for following types: INT8, UINT8, INT16, UINT16.

Remarks

Standard operations like PowerImage, CorrectGamma and LogarithmImage for images of type Int32 and Real are available in Image Point Transforms category.

See Also

- ApplyPixelLut – Applies previously created Look Up transformation to provided image.
- CreatePowerLut – Creates Look Up Table for power operation on image pixels.
- CreateGammaCorrectionLut – Creates Look Up Table for gamma correction operation on image pixels.
- CreateLogarithmLut – Creates Look Up Table for logarithm operation on image pixels.
- CorrectGamma – Performs gamma correction.

CreateLogarithmLut

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Creates Look Up Table for logarithm operation on image pixels.

Syntax

```
void avl::CreateLogarithmLut
(
 atl::Optional<const float&> inScale,
 float inOffset,
 bool inNormalizeZero,
 const avl::PlainType::Type& inOutputType,
 avl::PixelLut& outLut
)
```

Parameters

	Name	Type	Default	Description
→	inScale	Optional<const float&>	NIL	Scaling factor (default: 255)
→	inOffset	float		Offset factor
→	inNormalizeZero	bool		Specifies whether the output range should be rescaled to start from 0
→	inOutputType	const PlainType::Type&		
←	outLut	PixelLut&		

Description

This operation will create LookUp Table with precalculated values of logarithm operation on all pixel values of type defined in **inOutputType**. Such LUT can be reused across multiple ApplyPixelLut usages. This filter can create LUTs only for following types: INT8, UINT8, INT16, UINT16.

Remarks

Standard operations like PowerImage, CorrectGamma and LogarithmImage for images of type Int32 and Real are available in Image Point Transforms category.

See Also

- ApplyPixelLut – Applies previously created Look Up transformation to provided image.
- CreatePowerLut – Creates Look Up Table for power operation on image pixels.
- CreateGammaCorrectionLut – Creates Look Up Table for gamma correction operation on image pixels.
- CreateLogarithmLut – Creates Look Up Table for logarithm operation on image pixels.
- LogarithmImage – Computes a natural logarithm of each pixel.

CreatePowerLut

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Creates Look Up Table for power operation on image pixels.

Syntax

```
void avl::CreatePowerLut
(
 float inValue,
 const avl::PlainType::Type& inOutputType,
 avl::PixelLut& outLut
)
```

Parameters

	Name	Type	Default	Description
→	inValue	float	2.0f	
→	inOutputType	const PlainType::Type&		
←	outLut	PixelLut&		

Description

This operation will create LookUp Table with precalculated values of power operation on all pixel values of type defined in **inOutputType**. Such LUT can be reused across multiple ApplyPixelLut usages. This filter can create LUTs only for following types: INT8, UINT8, INT16, UINT16.

Remarks

Standard operations like PowerImage, CorrectGamma and LogarithmImage for images of type Int32 and Real are available in Image Point Transforms category.

See Also

- ApplyPixelLut – Applies previously created Look Up transformation to provided image.
- CreatePowerLut – Creates Look Up Table for power operation on image pixels.
- CreateGammaCorrectionLut – Creates Look Up Table for gamma correction operation on image pixels.
- CreateLogarithmLut – Creates Look Up Table for logarithm operation on image pixels.
- PowerImage – Exponentiates each pixel to the given power.

35. Geometry 2D Features

Table of content:

- SegmentVector
- VectorDirection
- VectorLength
- VectorsMedian

ArcCircle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the circle containing an arc.

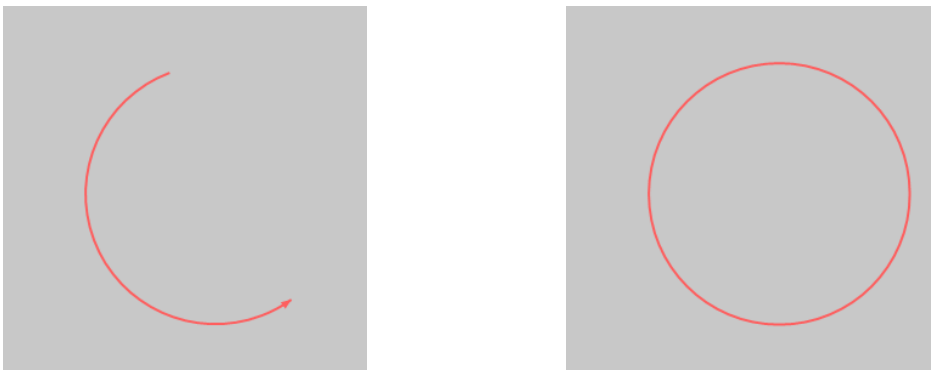
Syntax

```
void avl::ArcCircle
(
 const avl::Arc2D& inArc,
 avl::Circle2D& outCircle
)
```

Parameters

	Name	Type	Default	Description
→	inArc	const Arc2D&		Input arc
←	outCircle	Circle2D&		Circle containing the arc

Examples



***ArcCircle** performed on the sample arc.*

ArcEndpoints

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes two endpoints of an arc.

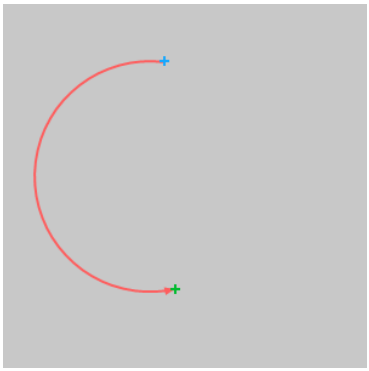
Syntax

```
void avl::ArcEndpoints
(
 const avl::Arc2D& inArc,
 avl::Point2D& outPoint1,
 avl::Point2D& outPoint2
)
```

Parameters

	Name	Type	Default	Description
→	inArc	const Arc2D&		
←	outPoint1	Point2D&		First endpoint
←	outPoint2	Point2D&		Second endpoint

Examples



*The resulting **outPoint1**(blue) and **outPoint2**(green) drawn with the input arc.*

ArcLength

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the length of an arc.

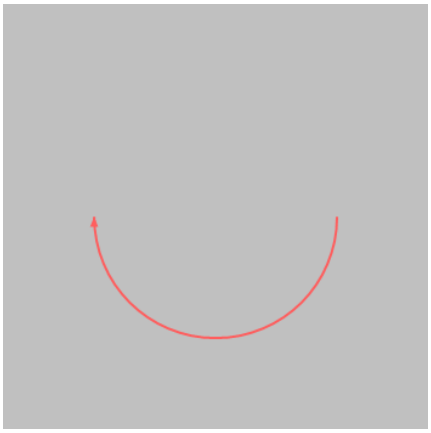
Syntax

```
void avl::ArcLength
(
 const avl::Arc2D& inArc,
 float& outLength
)
```

Parameters

	Name	Type	Default	Description
→	inArc	const Arc2D&		
←	outLength	float&		

Examples



ArcLength *performed on an arc with parameters:* ***X*** *= 175,* ***Y*** *= 175,* ***Radius*** *= 100,* ***StartAngle*** *= 0,* ***SweepAngle*** *= 180.*
The ***outLength*** *returns a value of* ***314,159****.*

ArcMidpoint

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the midpoint of an arc.

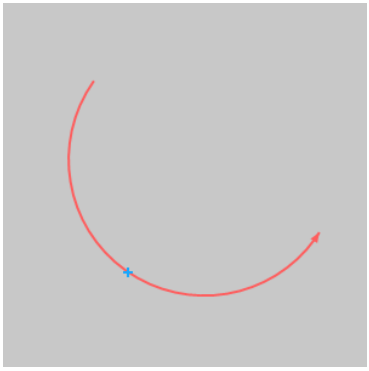
Syntax

```
void avl::ArcMidpoint
(
 const avl::Arc2D& inArc,
 avl::Point2D& outMidpoint
)
```

Parameters

	Name	Type	Default	Description
→	inArc	const Arc2D&		
←	outMidpoint	Point2D&		

Examples



*The resulting **outMidpoint** drawn with the input arc.*

CircleArea

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the area of a circle.

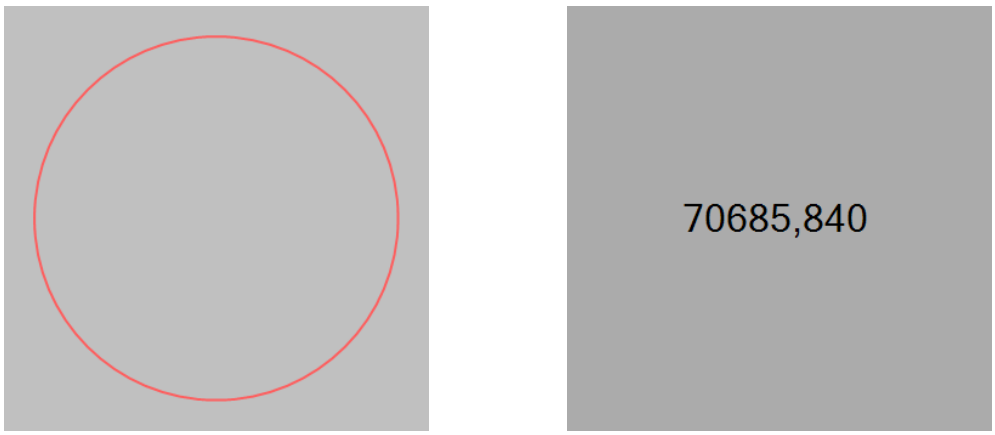
Syntax

```
void avl::CircleArea
(
 const avl::Circle2D& inCircle,
 float& outArea
)
```

Parameters

	Name	Type	Default	Description
→	inCircle	const Circle2D&		
←	outArea	float&		

Examples



***CircleArea** performed on a circle with parameters: **X** = 175, **Y** = 175, **Radius** = 150. The **outArea** returns a value of **70685,84**.*

CircleBoundingBox

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the smallest box containing a circle.

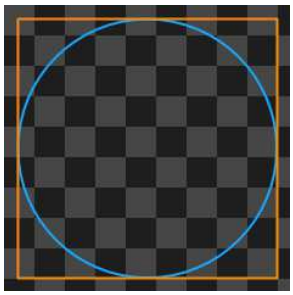
Syntax

```
void avl::CircleBoundingBox
(
 const avl::Circle2D& inCircle,
 avl::Box& outBoundingBox
)
```

Parameters

	Name	Type	Default	Description
→	inCircle	const Circle2D&		
←	outBoundingBox	Box&		

Examples



CircleBoundingBox performed on a sample circle.

CircleBoundingRectangle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the smallest rectangle containing a circle.

Syntax

```
void avl::CircleBoundingRectangle
(
 const avl::Circle2D& inCircle,
 atl::Optional<float> inAngle,
 avl::Rectangle2D& outBoundingRectangle,
 atl::Optional<avl::Point2D&> outCenter = atl::NIL,
 atl::Optional<float&> outLongSide = atl::NIL,
 atl::Optional<float&> outShortSide = atl::NIL
)
```

Parameters

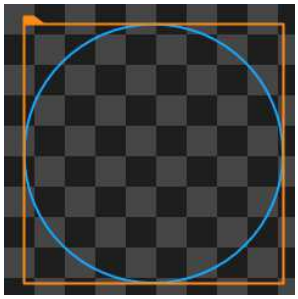
	Name	Type	Default	Description
→	inCircle	const Circle2D&		Input circle
→	inAngle	Optional<float>	NIL	Expected angle of the resulting rectangle
←	outBoundingRectangle	Rectangle2D&		Smallest bounding rectangle of the input circle
←	outCenter	Optional<Point2D&>	NIL	Center of the bounding rectangle
←	outLongSide	Optional<float&>	NIL	Length of the bounding rectangle long side
←	outShortSide	Optional<float&>	NIL	Length of the bounding rectangle short side

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outCenter**, **outLongSide**, **outShortSide**.

Read more about Optional Outputs.

Examples



CircleBoundingRectangle performed on a sample circle.

CircleCharacteristicPoint

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns a characteristic point (e.g. the top-left) of a box containing the input circle.

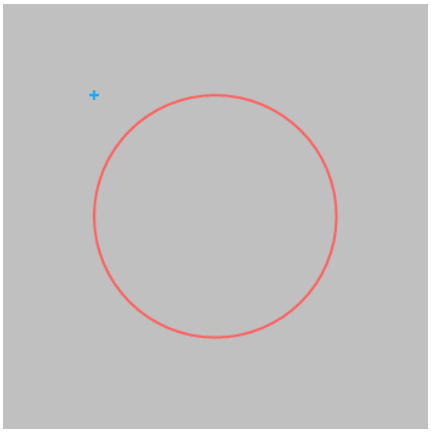
Syntax

```
void avl::CircleCharacteristicPoint
(
 const avl::Circle2D& inCircle,
 const avl::Anchor2D::Type inPointAnchor,
 avl::Point2D& outPoint
)
```

Parameters

	Name	Type	Default	Description
→	inCircle	const Circle2D&		
→	inPointAnchor	const Anchor2D::Type	TopLeft	
←	outPoint	Point2D&		

Examples



CircleCharacteristicPoint *performed on a circle with* ***inPointAnchor*** *= TopLeft.*

CirclePerimeterLength

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the length of a circle perimeter.

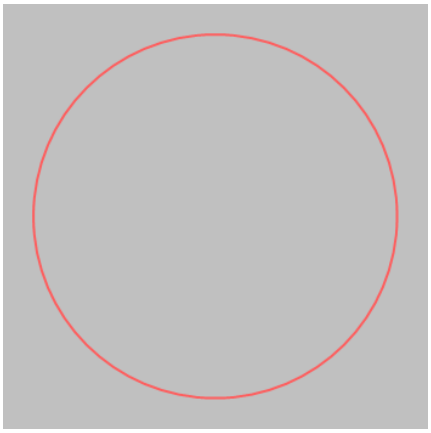
Syntax

```
void avl::CirclePerimeterLength
(
 const avl::Circle2D& inCircle,
 float& outPerimeterLength
)
```

Parameters

	Name	Type	Default	Description
→	inCircle	const Circle2D&		
←	outPerimeterLength	float&		

Examples



CirclePerimeterLength *performed on a circle with parameters:* ***X*** *= 175,* ***Y*** *= 175,* ***Radius*** *= 150.*
The ***outPerimeterLength*** *returns a value of* ***942,478****.*

CircleSection

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes an arciform section of the circle perimeter.

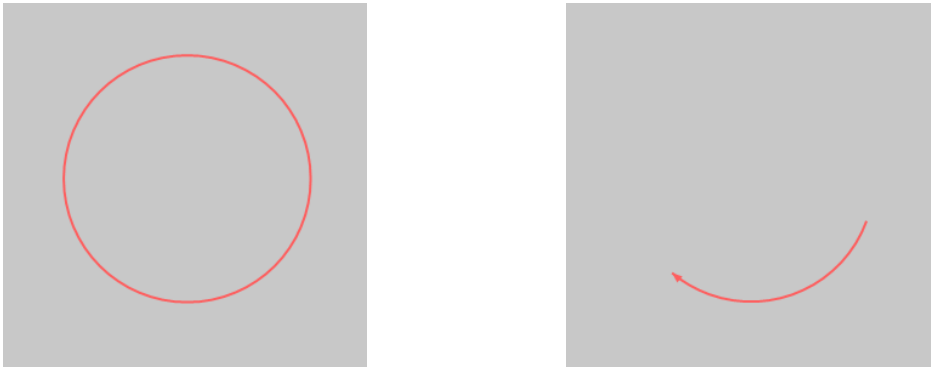
Syntax

```
void avl::CircleSection
(
 const avl::Circle2D& inCircle,
 float inStartAngle,
 float inSweepAngle,
 avl::Arc2D& outArc
)
```

Parameters

	Name	Type	Default	Description
→	inCircle	const Circle2D&		Input circle
→	inStartAngle	float		Direction at which the arc begins
→	inSweepAngle	float		Angular length of the arc (may be negative)
←	outArc	Arc2D&		

Examples



***CircleSection** performed on the sample circle, **inStartAngle** = 20.0 and **inSweepAngle** = 110.0.*

EllipseArea

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the area of an ellipse.

Syntax

```
void avl::EllipseArea
(
 const avl::Ellipse2D& inEllipse,
 float& outArea
)
```

Parameters

	Name	Type	Default	Description
→	inEllipse	const Ellipse2D&		
←	outArea	float&		

EllipseBoundingBox

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the smallest box containing an ellipse.

Syntax

```
void avl::EllipseBoundingBox
(
 const avl::Ellipse2D& inEllipse,
 avl::Box& outBoundingBox
)
```

Parameters

	Name	Type	Default	Description
→	inEllipse	const Ellipse2D&		
←	outBoundingBox	Box&		

EllipseBoundingRectangle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the smallest rectangle containing an ellipse.

Syntax

```
void avl::EllipseBoundingRectangle
(
 const avl::Ellipse2D& inEllipse,
 avl::Rectangle2D& outBoundingRectangle,
 atl::Optional<avl::Point2D&> outCenter = atl::NIL,
 atl::Optional<float&> outLongSide = atl::NIL,
 atl::Optional<float&> outShortSide = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inEllipse	const Ellipse2D&		
←	outBoundingRectangle	Rectangle2D&		Smallest bounding rectangle of the input points
←	outCenter	Optional<Point2D&>	NIL	Center of the bounding rectangle
←	outLongSide	Optional<float&>	NIL	Length of the bounding rectangle long side
←	outShortSide	Optional<float&>	NIL	Length of the bounding rectangle short side

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outCenter**, **outLongSide**, **outShortSide**.

Read more about Optional Outputs.

EllipseBoundingRectangle_FixedAngle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the smallest rectangle with the given orientation angle containing an ellipse.

Syntax

```
void avl::EllipseBoundingRectangle_FixedAngle
(
 const avl::Ellipse2D& inEllipse,
 float inAngle,
 avl::Rectangle2D& outBoundingRectangle,
 atl::Optional<avl::Point2D&> outCenter = atl::NIL,
 atl::Optional<float&> outLongSide = atl::NIL,
 atl::Optional<float&> outShortSide = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inEllipse	const Ellipse2D&		
→	inAngle	float		Expected angle of the resulting rectangle
←	outBoundingRectangle	Rectangle2D&		Smallest bounding rectangle of the input ellipse
←	outCenter	Optional<Point2D&>	NIL	Center of the bounding rectangle
←	outLongSide	Optional<float&>	NIL	Length of the bounding rectangle long side
←	outShortSide	Optional<float&>	NIL	Length of the bounding rectangle short side

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outCenter**, **outLongSide**, **outShortSide**.

Read more about Optional Outputs.

LineNormalVector

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the normal vector of a line.

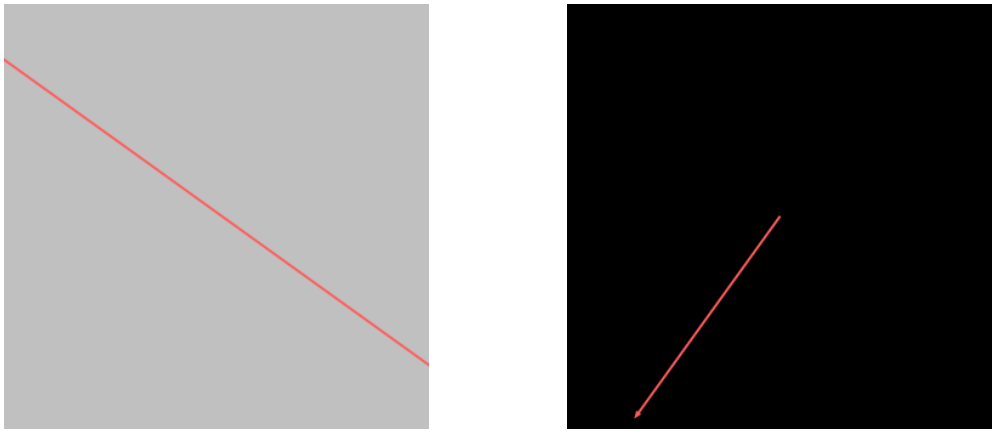
Syntax

```
void avl::LineNormalVector
(
 const avl::Line2D& inLine,
 bool inReverse,
 avl::Vector2D& outVector
)
```

Parameters

	Name	Type	Default	Description
→	inLine	const Line2D&		
→	inReverse	bool	True	
←	outVector	Vector2D&		

Examples



LineNormalVector *performed on a line (left image) returns* ***outVector*** *(right image).*

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite line on input in LineNormalVector.

LineOrientation

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the orientation angle of a line as angle in the range from 0 to 180.

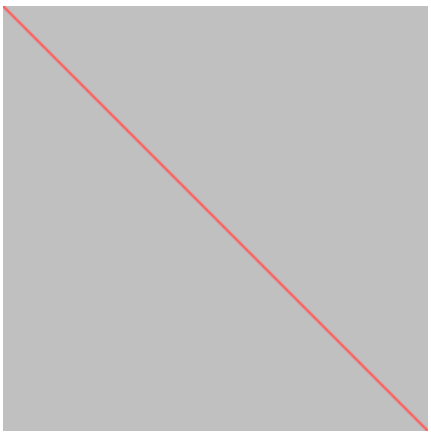
Syntax

```
void avl::LineOrientation
(
 const avl::Line2D& inLine,
 float& outOrientationAngle
)
```

Parameters

	Name	Type	Default	Description
→	inLine	const Line2D&		
←	outOrientationAngle	float&		

Examples



LineOrientation *performed on a line.*
The ***outOrientationAngle*** *in this case returns a value of* ***45****.*

PointsBoundingBox

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the smallest box containing an array of points.

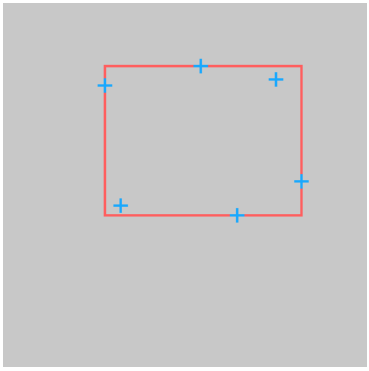
Syntax

```
void avl::PointsBoundingBox
(
 const atl::Array<avl::Point2D>& inPoints,
 avl::Box& outBoundingBox
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		
←	outBoundingBox	Box&		

Examples



*The resulting **outBoundingBox** drawn with the input points.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array on input in PointsBoundingBox.

PointsBoundingBox_OrNil

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the smallest box containing an array of points; returns NIL if the array is empty.

Syntax

```
void avl::PointsBoundingBox_OrNil
(
 const atl::Array<avl::Point2D>& inPoints,
 atl::Conditional<avl::Box>& outBoundingBox
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		
←	outBoundingBox	Conditional<Box>&		

PointsBoundingCircle

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the smallest circle containing an array of points.

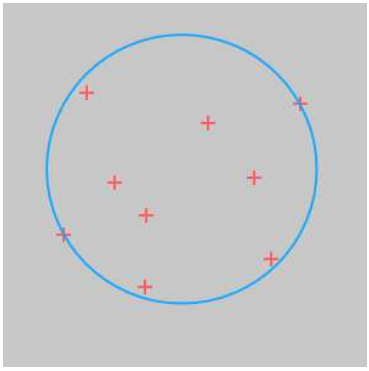
Syntax

```
void avl::PointsBoundingCircle
(
 const atl::Array<avl::Point2D>& inPoints,
 avl::Circle2D& outBoundingCircle
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		
←	outBoundingCircle	Circle2D&		

Examples



*The resulting **outBoundingCircle** drawn with the input points.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array on input in PointsBoundingCircle.

PointsBoundingCircle_OrNil

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the smallest circle containing an array of points; returns NIL if the array is empty.

Syntax

```
void avl::PointsBoundingCircle_OrNil
(
 const atl::Array<avl::Point2D>& inPoints,
 atl::Conditional<avl::Circle2D>& outBoundingCircle
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		
←	outBoundingCircle	Conditional<Circle2D>&		

PointsBoundingEllipse

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the smallest ellipse containing an array of points if such ellipse exists.

Syntax

```
void avl::PointsBoundingEllipse
(
 const atl::Array<avl::Point2D>& inPoints,
 avl::Ellipse2D& outBoundingEllipse,
 atl::Array<avl::Point2D>& outBoundaryPoints
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		
←	outBoundingEllipse	Ellipse2D&		
←	outBoundaryPoints	Array<Point2D>&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array on input in PointsBoundingEllipse.
DomainError	Failed to find proper ellipse PointsBoundingEllipse.

PointsBoundingParallelogram

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes the smallest parallelogram containing an array of points.

Syntax

```
void avl::PointsBoundingParallelogram
(
 const atl::Array<avl::Point2D>& inPoints,
 avl::BoundingRectangleFeature::Type inBoundingParallelogramFeature,
 avl::Path& outBoundingParallelogram,
 atl::Optional<avl::Point2D&> outCenter = atl::NIL,
 atl::Optional<float&> outLongSide = atl::NIL,
 atl::Optional<float&> outShortSide = atl::NIL,
 atl::Optional<float&> outAngle = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		Input array of points
→	inBoundingParallelogramFeature	BoundingRectangleFeature::Type	MinimalArea	Determines what kind of bounding parallelogram will be computed
←	outBoundingParallelogram	Path&		Smallest bounding parallelogram of the input points
←	outCenter	Optional<Point2D&>	NIL	Center of the bounding parallelogram
←	outLongSide	Optional<float&>	NIL	Length of the bounding parallelogram long side
←	outShortSide	Optional<float&>	NIL	Length of the bounding parallelogram short side
←	outAngle	Optional<float&>	NIL	Angle of the bounding parallelogram

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outCenter**, **outLongSide**, **outShortSide**, **outAngle**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	No points on input in PointsBoundingParallelogram.
DomainError	Unknown parallelogram feature in PointsBoundingParallelogram.

PointsBoundingRectangle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the smallest rectangle containing an array of points.

Syntax

```
void avl::PointsBoundingRectangle
(
 const atl::Array<avl::Point2D>& inPoints,
 avl::BoundingRectangleFeature::Type inBoundingRectangleFeature,
 float inReferenceAngle,
 avl::RectangleOrientation::Type inRectangleOrientation,
 avl::Rectangle2D& outBoundingRectangle,
 atl::Optional<avl::Point2D&> outCenter = atl::NIL,
 atl::Optional<float&> outLongSide = atl::NIL,
 atl::Optional<float&> outShortSide = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		Input array of points
→	inBoundingRectangleFeature	BoundingRectangleFeature::Type	MinimalArea	Determines what kind of bounding rectangle will be computed
→	inReferenceAngle	float	0.0f	The middle angle of the valid range of the output rectangle's angle
→	inRectangleOrientation	RectangleOrientation::Type	Horizontal	Orientation of the output rectangle
←	outBoundingRectangle	Rectangle2D&		Smallest bounding rectangle of the input points
←	outCenter	Optional<Point2D&>	NIL	Center of the bounding rectangle
←	outLongSide	Optional<float&>	NIL	Length of the bounding rectangle long side
←	outShortSide	Optional<float&>	NIL	Length of the bounding rectangle short side

Optional Outputs

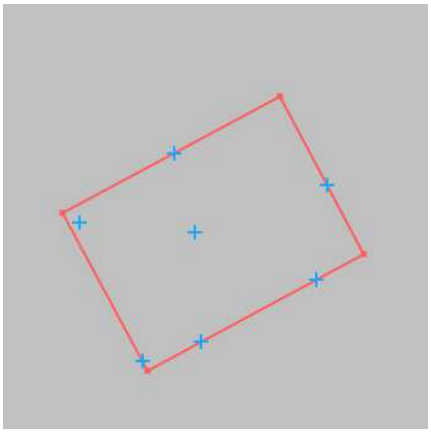
The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outCenter**, **outLongSide**, **outShortSide**.

Read more about Optional Outputs.

Description

The filter computes a rectangle with the smallest possible selected feature that contains all given points. The angle of the resulting rectangle is then normalized as in the NormalizeRectangleOrientation filter.

Examples



*The resulting **outBoundingRectangle** drawn with the input points and with **inRectangleOrientation** set on **Horizontal***

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect BoundingRectangleFeature in PointsBoundingRectangle.
DomainError	No points on input in PointsBoundingRectangle.

See Also

- NormalizeRectangleOrientation – Changes orientation of the given rectangle according to parameters.

PointsBoundingRectangle_FixedAngle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the smallest rectangle with the given orientation angle containing an array of points.

Syntax

```
void avl::PointsBoundingRectangle_FixedAngle
(
 const atl::Array<avl::Point2D>& inPoints,
 float inAngle,
 avl::Rectangle2D& outBoundingRectangle,
 atl::Optional<avl::Point2D&> outCenter = atl::NIL,
 atl::Optional<float&> outLongSide = atl::NIL,
 atl::Optional<float&> outShortSide = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		Input array of points
→	inAngle	float		Expected angle of the resulting rectangle
←	outBoundingRectangle	Rectangle2D&		Smallest bounding rectangle of the input points
←	outCenter	Optional<Point2D&>	NIL	Center of the bounding rectangle
←	outLongSide	Optional<float&>	NIL	Length of the bounding rectangle long side
←	outShortSide	Optional<float&>	NIL	Length of the bounding rectangle short side

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outCenter**, **outLongSide**, **outShortSide**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array on input in PointsBoundingRectangle_FixedAngle.

PointsBoundingRectangle_FixedAngle_OrNil

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the smallest rectangle with the given orientation angle containing an array of points; returns NIL when the array is empty.

Syntax

```
void avl::PointsBoundingRectangle_FixedAngle_OrNil
(
 const atl::Array<avl::Point2D>& inPoints,
 float inAngle,
 atl::Conditional<avl::Rectangle2D>& outBoundingRectangle,
 atl::Conditional<avl::Point2D>& outCenter,
 atl::Conditional<float>& outLongSide,
 atl::Conditional<float>& outShortSide
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		Input array of points
→	inAngle	float		Expected angle of the resulting rectangle
←	outBoundingRectangle	Conditional<Rectangle2D>&		Smallest bounding rectangle of the input points
←	outCenter	Conditional<Point2D>&		Center of the bounding rectangle
←	outLongSide	Conditional<float>&		Length of the bounding rectangle long side
←	outShortSide	Conditional<float>&		Length of the bounding rectangle short side

PointsBoundingRectangle_OrNil

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the smallest rectangle containing an array of points; returns NIL when the array is empty.

Syntax

```
void avl::PointsBoundingRectangle_OrNil
(
 const atl::Array<avl::Point2D>& inPoints,
 avl::BoundingRectangleFeature::Type inBoundingRectangleFeature,
 float inReferenceAngle,
 avl::RectangleOrientation::Type inRectangleOrientation,
 atl::Conditional<avl::Rectangle2D>& outBoundingRectangle,
 atl::Conditional<avl::Point2D>& outCenter,
 atl::Conditional<float>& outLongSide,
 atl::Conditional<float>& outShortSide
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		Input array of points
→	inBoundingRectangleFeature	BoundingRectangleFeature::Type	MinimalArea	Determines what kind of bounding rectangle will be computed
→	inReferenceAngle	float	0.0f	The middle angle of the valid range of the output rectangle's angle
→	inRectangleOrientation	RectangleOrientation::Type	Horizontal	Orientation of the output rectangle
←	outBoundingRectangle	Conditional<Rectangle2D>&		Smallest bounding rectangle of the input points
←	outCenter	Conditional<Point2D>&		Center of the bounding rectangle
←	outLongSide	Conditional<float>&		Length of the bounding rectangle long side
←	outShortSide	Conditional<float>&		Length of the bounding rectangle short side

PointsCaliperDiameter

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the longest and the shortest width of the input points measured as distance between parallel lines containing all of them.

Syntax

```
void avl::PointsCaliperDiameter
(
 const atl::Array<avl::Point2D>& inPoints,
 atl::Optional<avl::Segment2D&> outMinDiameter = atl::NIL,
 atl::Optional<float&> outMinDiameterLength = atl::NIL,
 atl::Optional<avl::Segment2D&> outMaxDiameter = atl::NIL,
 atl::Optional<float&> outMaxDiameterLength = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		
←	outMinDiameter	Optional<Segment2D&>	NIL	
←	outMinDiameterLength	Optional<float&>	NIL	
←	outMaxDiameter	Optional<Segment2D&>	NIL	
←	outMaxDiameterLength	Optional<float&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outMinDiameter**, **outMinDiameterLength**, **outMaxDiameter**, **outMaxDiameterLength**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty point array on input in PointsCaliperDiameter.

PointsConvexHull

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the smallest convex shape that contains the given array of points.

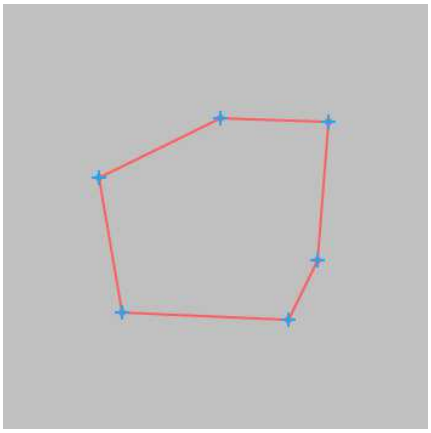
Syntax

```
void avl::PointsConvexHull
(
 const atl::Array<avl::Point2D>& inPoints,
 avl::Path& outConvexHull
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		
←	outConvexHull	Path&		A closed path representing the computed convex hull

Examples



*The resulting **outConvexHull** drawn with the input points.*

PointsDiameter

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Finds the longest segment connecting two points from a given array.

Syntax

```
void avl::PointsDiameter
(
 const atl::Array<avl::Point2D>& inPoints,
 atl::Optional<avl::Segment2D&> outDiameter = atl::NIL,
 atl::Optional<float&> outDiameterLength = atl::NIL
)
```

Parameters

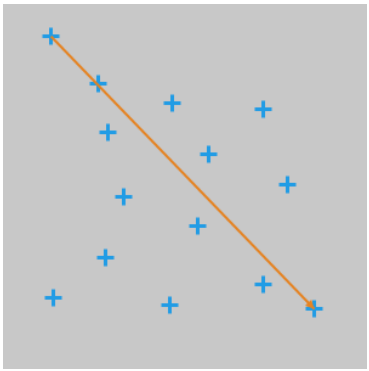
	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		Input array of points
←	outDiameter	Optional<Segment2D&>	NIL	Longest segment found
←	outDiameterLength	Optional<float&>	NIL	Length of longest segment found

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outDiameter**, **outDiameterLength**.

Read more about Optional Outputs.

Examples



PointsDiameter *performed on an array of input points. The orange line is the result.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array of points on input in PointsDiameter.

PointsMassCenter

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the central point of the input points.

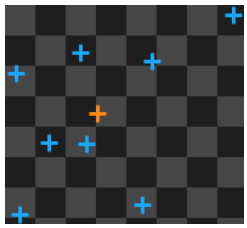
Syntax

```
void avl::PointsMassCenter
(
 const atl::Array<avl::Point2D>& inPoints,
 avl::Point2D& outMassCenter
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		
←	outMassCenter	Point2D&		

Examples



PointsMassCenter *performed on a set of 8 input points. The orange point is the result.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Input point array is empty in PointsMassCenter.

PointsMassCenter_OrNil

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the central point of the input points; returns NIL if the array is empty.

Syntax

```
void avl::PointsMassCenter_OrNil
(
 const atl::Array<avl::Point2D>& inPoints,
 atl::Conditional<avl::Point2D>& outMassCenter
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		
←	outMassCenter	Conditional<Point2D>&		

PointsMedian

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the geometric median of the input points.

Syntax

```
void avl::PointsMedian
(
 const atl::Array<avl::Point2D>& inPoints,
 atl::Optional<const atl::Array<float>&> inWeights,
 const int inMaxIterationCount,
 avl::Point2D& outGeometricMedian,
 atl::Optional<float&> outDistanceSum = atl::NIL,
 atl::Array<avl::Point2D>& diagApproximationSteps = atl::Dummy<atl::Array<Point2D> >()
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoints	const Array<Point2D>&			Input points
→	inWeights	Optional<const Array<float>&>		NIL	Optional input weights
→	inMaxIterationCount	const int	1 - ∞	10	Maximum number of iterations
←	outGeometricMedian	Point2D&			Geometric median
←	outDistanceSum	Optional<float&>		NIL	Sum of distances from input points to geometric median
🔍	diagApproximationSteps	Array<Point2D>&			Approximate geometric medians calculated during subsequent iterations

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outDistanceSum**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Input arrays must be of the same size in PointsMedian.
DomainError	Input point array is empty in PointsMedian.

PointsOrientation

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the orientation of a set of 2D points.

Syntax

```
void avl::PointsOrientation
(
 const atl::Array<avl::Point2D>& inPoints,
 float& outOrientation
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		
←	outOrientation	float&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array of points on input in PointsOrientation.

RectangleArea

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the area of a rectangle.

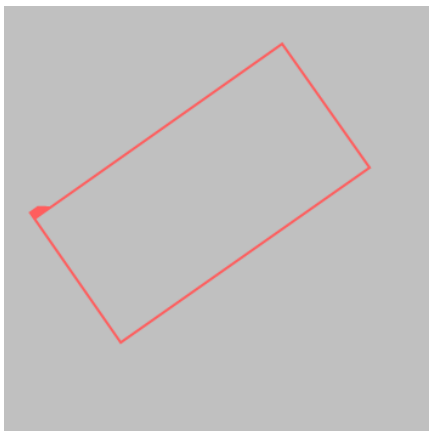
Syntax

```
void avl::RectangleArea
(
 const avl::Rectangle2D& inRectangle,
 float& outArea
)
```

Parameters

	Name	Type	Default	Description
→	inRectangle	const Rectangle2D&		
←	outArea	float&		

Examples



__RectangleArea__ performed on a rectangle with parameters: __X__ = 25, __Y__ = 175, __Angle__ = 325, __Width__ = 250, __Height__ = 125. The __outArea__ returns a value of __31250__.

RectangleBoundingBox

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the smallest box containing a rectangle.

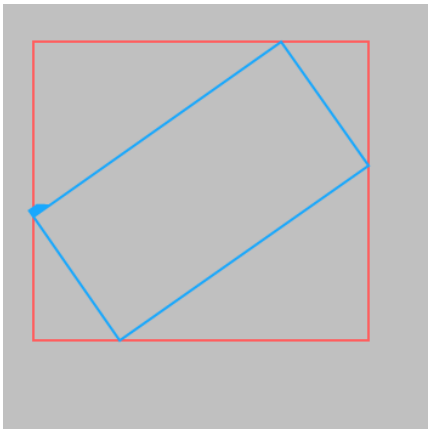
Syntax

```
void avl::RectangleBoundingBox
(
 const avl::Rectangle2D& inRectangle,
 avl::Box& outBoundingBox
)
```

Parameters

	Name	Type	Default	Description
→	inRectangle	const Rectangle2D&		
←	outBoundingBox	Box&		

Examples



RectangleBoundingBox *performed on a rectangle (blue) with parameters:* ***X*** *= 25,* ***Y*** *= 175,* ***Angle*** *= 325,* ***Width*** *= 250,* ***Height*** *= 125. The result (red box) is returned in* ***outBoundingBox****.*

RectangleBoundingCircle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the smallest circle containing a rectangle.

Syntax

```
void avl::RectangleBoundingCircle
(
 const avl::Rectangle2D& inRectangle,
 avl::Circle2D& outBoundingCircle
)
```

Parameters

	Name	Type	Default	Description
→	inRectangle	const Rectangle2D&		
←	outBoundingCircle	Circle2D&		

RectangleCenter

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the center point of a rectangle.

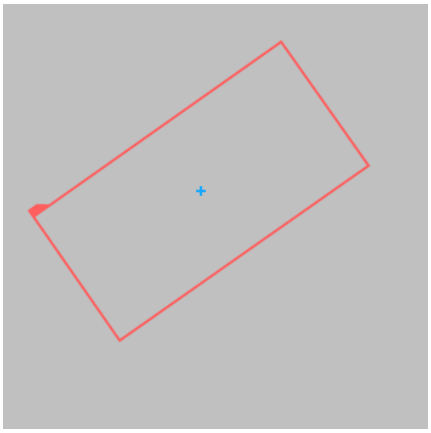
Syntax

```
void avl::RectangleCenter
(
 const avl::Rectangle2D& inRectangle,
 avl::Point2D& outCenterPoint
)
```

Parameters

	Name	Type	Default	Description
→	inRectangle	const Rectangle2D&		
←	outCenterPoint	Point2D&		

Examples



RectangleCenter *performed on a rectangle with parameters:* ***X*** *= 25,* ***Y*** *= 175,* ***Angle*** *= 325,* ***Width*** *= 250,* ***Height*** *= 125.*
The center point is returned in ***outCenterPoint****.*

RectangleCharacteristicPoint

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns a characteristic point (e.g. the top-left) of the input rectangle.

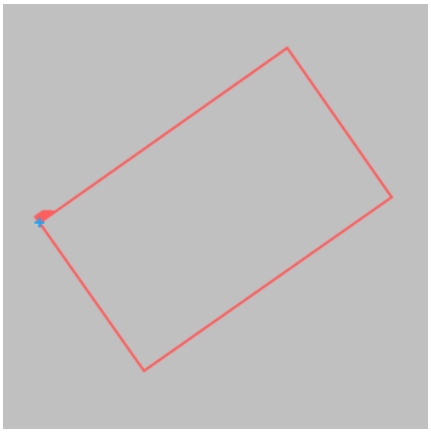
Syntax

```
void avl::RectangleCharacteristicPoint
(
 const avl::Rectangle2D& inRectangle,
 const avl::Anchor2D::Type inPointAnchor,
 avl::Point2D& outPoint
)
```

Parameters

	Name	Type	Default	Description
→	inRectangle	const Rectangle2D&		
→	inPointAnchor	const Anchor2D::Type	TopLeft	
←	outPoint	Point2D&		

Examples



RectangleCharacteristicPoint *performed on a rectangle with* ***inPointAnchor*** *= TopLeft.*

RectangleCharacteristicPoints

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes rectangle's characteristic points.

Syntax

```
void avl::RectangleCharacteristicPoints
(
 const avl::Rectangle2D& inRectangle,
 atl::Optional<avl::Point2D&> outTopLeft = atl::NIL,
 atl::Optional<avl::Point2D&> outTopCenter = atl::NIL,
 atl::Optional<avl::Point2D&> outTopRight = atl::NIL,
 atl::Optional<avl::Point2D&> outMiddleLeft = atl::NIL,
 atl::Optional<avl::Point2D&> outMiddleCenter = atl::NIL,
 atl::Optional<avl::Point2D&> outMiddleRight = atl::NIL,
 atl::Optional<avl::Point2D&> outBottomLeft = atl::NIL,
 atl::Optional<avl::Point2D&> outBottomCenter = atl::NIL,
 atl::Optional<avl::Point2D&> outBottomRight = atl::NIL,
 atl::Optional<atl::Array<avl::Point2D>&> outCorners = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inRectangle	const Rectangle2D&		
←	outTopLeft	Optional<Point2D&>	NIL	
←	outTopCenter	Optional<Point2D&>	NIL	
←	outTopRight	Optional<Point2D&>	NIL	
←	outMiddleLeft	Optional<Point2D&>	NIL	
←	outMiddleCenter	Optional<Point2D&>	NIL	
←	outMiddleRight	Optional<Point2D&>	NIL	
←	outBottomLeft	Optional<Point2D&>	NIL	
←	outBottomCenter	Optional<Point2D&>	NIL	
←	outBottomRight	Optional<Point2D&>	NIL	
←	outCorners	Optional<Array<Point2D>&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outTopLeft**, **outTopCenter**, **outTopRight**, **outMiddleLeft**, **outMiddleCenter**, **outMiddleRight**, **outBottomLeft**, **outBottomCenter**, **outBottomRight**, **outCorners**.

Read more about Optional Outputs.

RectangleCorners

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the four corner points of a rectangle.

Syntax

```
void avl::RectangleCorners
(
 const avl::Rectangle2D& inRectangle,
 avl::Point2D& outTopLeft,
 avl::Point2D& outTopRight,
 avl::Point2D& outBottomRight,
 avl::Point2D& outBottomLeft
)
```

Parameters

	Name	Type	Default	Description
→	inRectangle	const Rectangle2D&		
←	outTopLeft	Point2D&		
←	outTopRight	Point2D&		
←	outBottomRight	Point2D&		
←	outBottomLeft	Point2D&		

RectanglePerimeterLength

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the perimeter length of a rectangle.

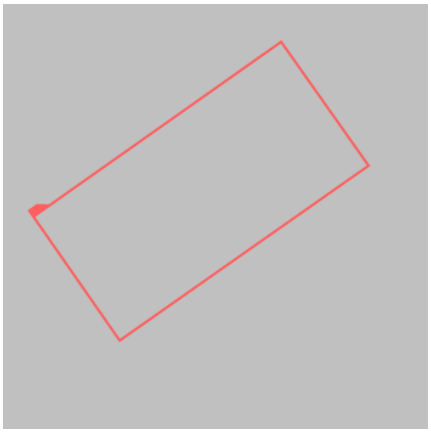
Syntax

```
void avl::RectanglePerimeterLength
(
 const avl::Rectangle2D& inRectangle,
 float& outPerimeterLength
)
```

Parameters

	Name	Type	Default	Description
→	inRectangle	const Rectangle2D&		
←	outPerimeterLength	float&		

Examples



RectanglePerimeterLength *performed on a rectangle with parameters:* ***X*** *= 25,* ***Y*** *= 175,* ***Angle*** *= 325,* ***Width*** *= 250,* ***Height*** *= 125.*
The ***outPerimeterLength*** *returns a value of* ***750****.*

RectangleSides

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns segments representing the sides of the input rectangle.

Syntax

```
void avl::RectangleSides
(
 const avl::Rectangle2D& inRectangle,
 atl::Optional<avl::Segment2D&> outTopSide = atl::NIL,
 atl::Optional<avl::Segment2D&> outRightSide = atl::NIL,
 atl::Optional<avl::Segment2D&> outBottomSide = atl::NIL,
 atl::Optional<avl::Segment2D&> outLeftSide = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inRectangle	const Rectangle2D&		
←	outTopSide	Optional<Segment2D&>	NIL	
←	outRightSide	Optional<Segment2D&>	NIL	
←	outBottomSide	Optional<Segment2D&>	NIL	
←	outLeftSide	Optional<Segment2D&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outTopSide**, **outRightSide**, **outBottomSide**, **outLeftSide**.

Read more about Optional Outputs.

SegmentBisector

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes a line passing through the center of a segment at a right angle.

Syntax

```
void avl::SegmentBisector
(
 const avl::Segment2D& inSegment,
 avl::Line2D& outBisector
)
```

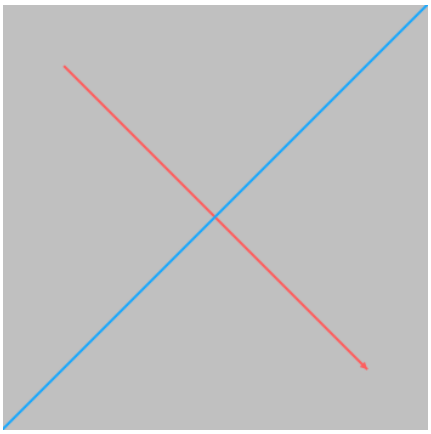
Parameters

	Name	Type	Default	Description
→	inSegment	const Segment2D&		
←	outBisector	Line2D&		

Description

Note that because of inaccuracies of floating-point arithmetic, some geometric operations (including this one) may lead to unpredictable results for degenerated cases. In this filter such a case occurs when an empty segment is given on input.

Examples



SegmentBisector *performed on a segment with parameters:* ***X1*** *= 50,* ***Y1*** *= 50,* ***X2*** *= 300,* ***Y2*** *= 300.*

SegmentCenter

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the center point of a segment.

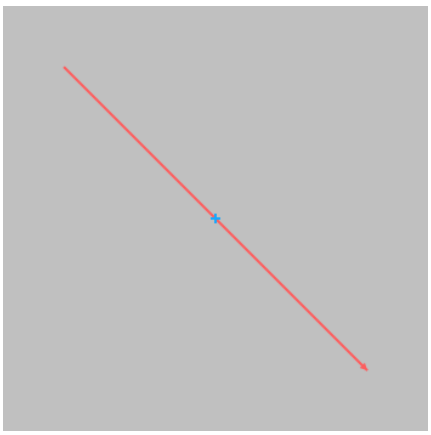
Syntax

```
void avl::SegmentCenter
(
 const avl::Segment2D& inSegment,
 avl::Point2D& outCenterPoint
)
```

Parameters

	Name	Type	Default	Description
→	inSegment	const Segment2D&		
←	outCenterPoint	Point2D&		

Examples



SegmentCenter *performed on a segment with parameters:* ***X1*** *= 50,* ***Y1*** *= 50,* ***X2*** *= 300,* ***Y2*** *= 300.*

SegmentLength

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the length of a segment.

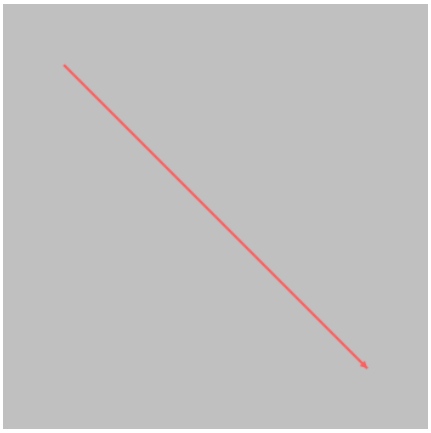
Syntax

```
void avl::SegmentLength
(
 const avl::Segment2D& inSegment,
 float& outLength
)
```

Parameters

	Name	Type	Default	Description
→	inSegment	const Segment2D&		
←	outLength	float&		

Examples



SegmentLength *performed on a segment with parameters:* ***X1*** *= 50,* ***Y1*** *= 50,* ***X2*** *= 300,* ***Y2*** *= 300.*
outLength *returns* ***353,553****.*

SegmentLine

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the line passing through a segment.

Syntax

```
void avl::SegmentLine
(
 const avl::Segment2D& inSegment,
 avl::Line2D& outLine
)
```

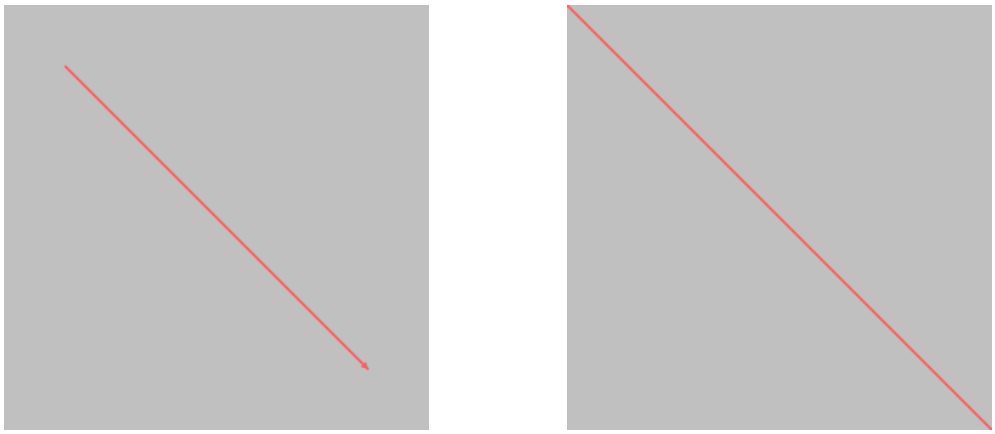
Parameters

	Name	Type	Default	Description
→	inSegment	const Segment2D&		Input segment
←	outLine	Line2D&		The resulting line

Description

Note that because of inaccuracies of floating-point arithmetic, some geometric operations (including this one) may lead to unpredictable results for degenerated cases. In this filter such a case occurs when an empty segment is given on input.

Examples



SegmentLine *performed on a segment with parameters:* ***X1*** *= 50,* ***Y1*** *= 50,* ***X2*** *= 300,* ***Y2*** *= 300.*
outLine *returns line (picture on right).*

SegmentNormalVector

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes an unitary vector perpendicular to the segment.

Syntax

```
void avl::SegmentNormalVector
(
 const avl::Segment2D& inSegment,
 bool inReverse,
 avl::Vector2D& outNormalVector
)
```

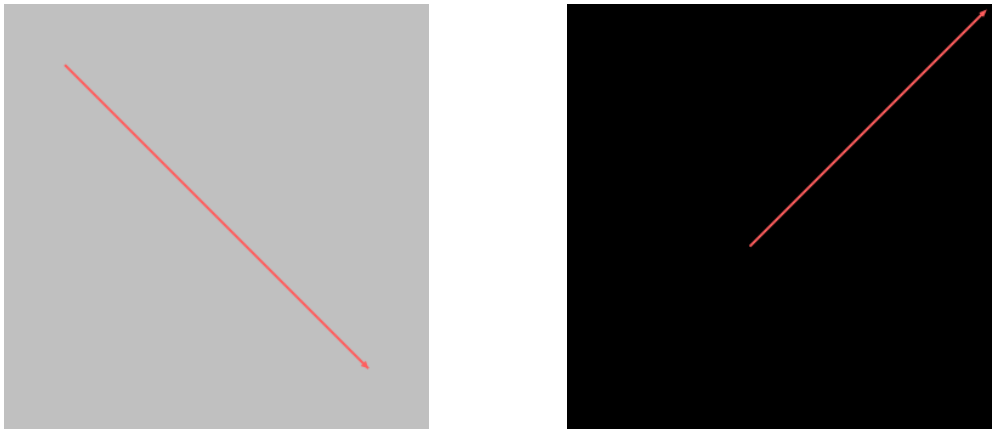
Parameters

	Name	Type	Default	Description
→	inSegment	const Segment2D&		
→	inReverse	bool	True	
←	outNormalVector	Vector2D&		

Description

Note that because of inaccuracies of floating-point arithmetic, some geometric operations (including this one) may lead to unpredictable results for degenerated cases. In this filter such a case occurs when an empty segment is given on input.

Examples



SegmentNormalVector *performed on a segment with parameters:* ***X1*** *= 50,* ***Y1*** *= 50,* ***X2*** *= 300,* ***Y2*** *= 300.*
outNormalVector *result shown on the right picture.*

SegmentOrientation

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the orientation of a segment.

Syntax

```
void avl::SegmentOrientation
(
 const avl::Segment2D& inSegment,
 avl::AngleRange::Type inAngleRange,
 float& outOrientationAngle
)
```

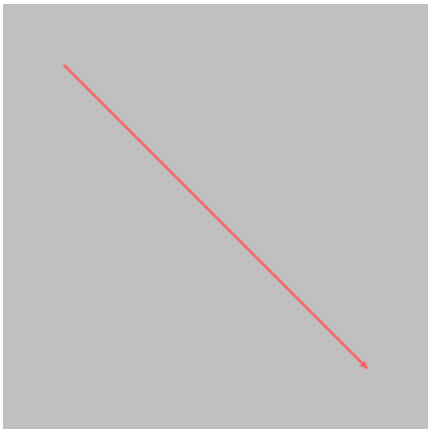
Parameters

	Name	Type	Default	Description
→	inSegment	const Segment2D&		
→	inAngleRange	AngleRange::Type	_0_180	Switches between 0-90, 0-180 or 0-360 degrees
←	outOrientationAngle	float&		

Description

Note that because of inaccuracies of floating-point arithmetic, some geometric operations (including this one) may lead to unpredictable results for degenerated cases. In this filter such a case occurs when an empty segment is given on input.

Examples



SegmentOrientation *performed on a segment with parameters:* ***X1*** *= 50,* ***Y1*** *= 50,* ***X2*** *= 300,* ***Y2*** *= 300 and* ***inAngleRange*** *set to _0_180.* ***outOrientationAngle*** *returns* ***45****.*

SegmentVector

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns the vector [x2 - x1, y2 - y1].

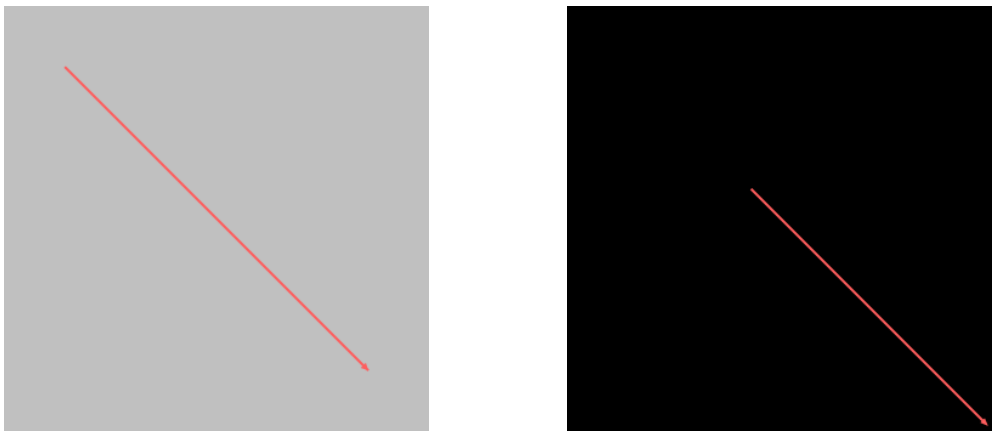
Syntax

```
void avl::SegmentVector
(
 const avl::Segment2D& inSegment,
 avl::Vector2D& outVector
)
```

Parameters

	Name	Type	Default	Description
→	inSegment	const Segment2D&		
←	outVector	Vector2D&		

Examples



SegmentVector *performed on a segment with parameters:* ***X1*** *= 50,* ***Y1*** *= 50,* ***X2*** *= 300,* ***Y2*** *= 300.*
outVector *result shown on the right picture.*

VectorDirection

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the direction angle of a vector as an angle in range the from 0 to 360.

Syntax

```
void avl::VectorDirection
(
 const avl::Vector2D& inVector,
 float& outDirection
)
```

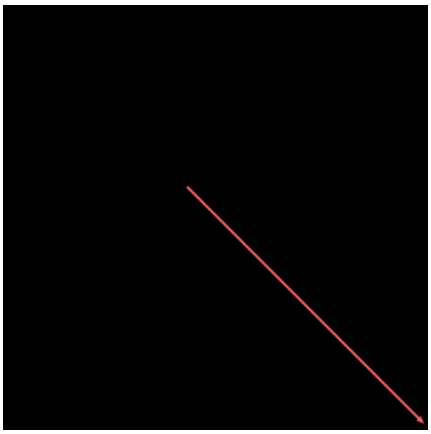
Parameters

	Name	Type	Default	Description
→	inVector	const Vector2D&		
←	outDirection	float&		

Description

Note that because of inaccuracies of floating-point arithmetic, some geometric operations (including this one) may lead to unpredictable results for degenerated cases. In this filter such a case occurs when a zero vector is given on input.

Examples



VectorDirection *performed on a vector with parameters:* ***DeltaX*** *= 10,* ***DeltaY*** *= 10.*
outDirection *returns* ***45****.*

VectorLength

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the length of a vector.

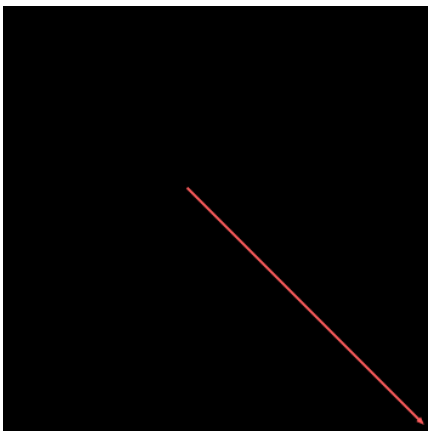
Syntax

```
void avl::VectorLength
(
 const avl::Vector2D& inVector,
 float& outLength
)
```

Parameters

	Name	Type	Default	Description
→	inVector	const Vector2D&		
←	outLength	float&		

Examples



VectorLength *performed on a vector with parameters:* ***DeltaX*** *= 10,* ***DeltaY*** *= 10.*
outLength *returns* ***14,142****.*

VectorsMedian

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the geometric median of the input vectors.

Syntax

```
void avl::VectorsMedian
(
 const atl::Array<avl::Vector2D>& inVectors,
 atl::Optional<const atl::Array<float>&> inWeights,
 const int inMaxIterationCount,
 avl::Vector2D& outGeometricMedian,
 atl::Optional<float&> outDistanceSum = atl::NIL,
 atl::Array<avl::Vector2D>& diagApproximationSteps = atl::Dummy<atl::Array<Vector2D> >()
)
```

Parameters

	Name	Type	Range	Default	Description
→	inVectors	const Array<Vector2D>&			Input vectors
→	inWeights	Optional<const Array<float>&>		NIL	Optional input weights
→	inMaxIterationCount	const int	1 - ∞	10	Maximum number of iterations
←	outGeometricMedian	Vector2D&			Geometric median
←	outDistanceSum	Optional<float&>		NIL	Sum of distances from input vectors to geometric median
🔍	diagApproximationSteps	Array<Vector2D>&			Approximate geometric medians calculated during subsequent iterations

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outDistanceSum**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Input arrays must be of the same size in VectorsMedian.
DomainError	Input vector array is empty in VectorsMedian.

36. Assertions

Table of content:

AssertImageEqualTo

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Asserts that two images are equal or differ by a small given margin.

Syntax

```
void avl::AssertImageEqualTo
(
 const avl::Image& inImage,
 const avl::Image& inExpectedImage,
 const float inMaxDifference,
 const atl::String& inDescription,
 const float inEpsilon = 1.0f,
 const int inEpsilonScale = -6
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inImage	const Image&			Input image
➡	inExpectedImage	const Image&			
➡	inMaxDifference	const float	0.0 - ∞	0.0f	Maximum absolute difference between a specific pixel of the two images
➡	inDescription	const String&			
➡	inEpsilon	const float	0.0 - ∞	1.0f	Maximum allowed difference, significant. InEpsilon will be multiplied by 10^inEpsilonScale before checking the difference
➡	inEpsilonScale	const int		-6	

Errors

List of possible exceptions:

Error type	Description
DomainError	Input images have different pixel formats in AssertImageEqualTo.
DomainError	Input images have different sizes in AssertImageEqualTo.

37. Image Enhancement

Table of content:

AutoAdjustColors

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Adjusts image colors by stretching each channel separately.

Syntax

```
void avl::AutoAdjustColors
(
 const avl::Image& inImage,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
←	outImage	Image&		Output image

Requirements

For input **inImage** only pixel formats are supported: 3□uint8, 3□int8, 3□uint16, 3□int16, 3□int32, 3□real.

Read more about pixel formats in Image documentation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported inImage pixel format in AutoAdjustColors. Supported formats: 3xUInt8, 3xInt8, 3xUInt16, 3xInt16, 3xInt32, 3xReal.

EqualizeImageHistogram

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Maps image pixels to new values to achieve uniform distribution of intensities in the range (0, 255).

Syntax

```
void avl::EqualizeImageHistogram
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 float inSaturateBrightestFraction,
 float inSaturateDarkestFraction,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inSaturateBrightestFraction	float	0.0 - 1.0	0.0f	Fraction of the brightest pixels skipped during normalization
→	inSaturateDarkestFraction	float	0.0 - 1.0	0.0f	Fraction of the darkest pixels skipped during normalization
←	outImage	Image&			Output image

Requirements

For input **inImage** only pixel formats are supported: 1□uint8.

Read more about pixel formats in Image documentation.

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

Read more about In-place Computation.

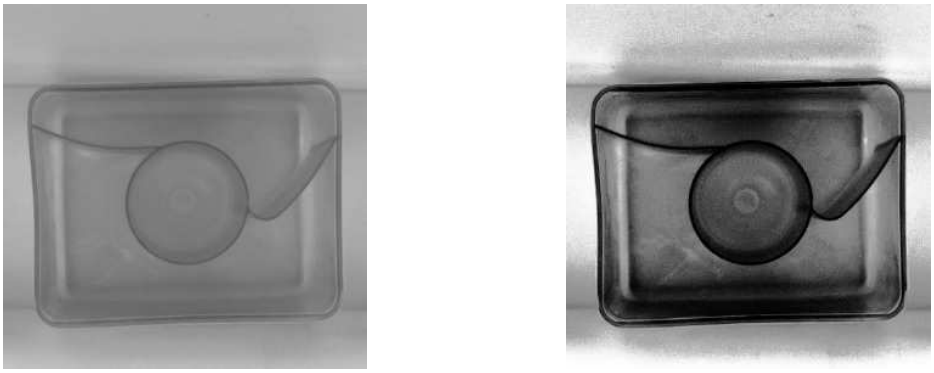
Description

The filter applies non-linear mapping to image pixel values so that pixel intensities of the resulting image are evenly distributed in range from 0 to 255.

The operation computes the cumulative histogram c of **inImage** and the image size N. Then the result is computed as follows:

$$outImage[i,j] = C[inImage[i,j]] \times \frac{255}{N}$$

Examples



EqualizeImageHistogram *run on example image.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in EqualizeImageHistogram.
DomainError	The sum of inSaturateBrightestFraction and inSaturateDarkestFraction can't be greater than 1 in EqualizeImageHistogram.
DomainError	Not supported inImage pixel format in EqualizeImageHistogram. Supported formats: 1xUInt8.

See Also

- ImageHistogram – Computes the histogram of the image pixel values.
- ConvertToCumulativeHistogram – Computes the cumulative histogram of input histogram.

ExpaintImage_Bornemann

Header: AVL.h
Namespace: avl
Module: FoundationPro

Speculatively sets pixel values outside of a region using the fast marching method and coherence analysis.

Syntax

```
void avl::ExpaintImage_Bornemann
(
 const avl::Image& inImage,
 const avl::Region& inRegionToExpaint,
 const int inExpaintingRadius,
 const int inRange,
 const float inPreSmoothing,
 const float inPostSmoothing,
 const float inSharpness,
 avl::LuminanceMode::Type inLuminanceMode,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRegionToExpaint	const Region&			Part of the image to be expainted
→	inExpaintingRadius	const int	1 - +∞	8	How far to expaint from the region
→	inRange	const int	1 - +∞	6	Defines how far a pixel can be from one currently being inpainted to be considered in calculations
→	inPreSmoothing	const float	0.0 - ∞	2.0f	Standard deviation of a gaussian kernel used before inpainting calculations
→	inPostSmoothing	const float	0.0 - ∞	3.0f	Standard deviation of a gaussian kernel used after initial inpainting calculations
→	inSharpness	const float	0.0 - ∞	35.0f	Desired sharpness of edges inside of the inpainted region (higher = sharper)
→	inLuminanceMode	LuminanceMode::Type		YUV	Determines how the luminance of the input image will be computed
←	outImage	Image&			Output image

Description

This filter expaints a region of an image by **inExpaintingRadius** pixels using the fast marching method and coherence flow analysis.

A detailed description of this method can be found here: InpaintImage_Bornemann.

Examples

Input image

*From the left: output with **inRange** = 4, output with **inRange** = 12*

*From the left: output with **inPreSmoothing** = 1, output with **inPreSmoothing** = 3*

*From the left: output with **inPostSmoothing** = 1, output with **inPostSmoothing** = 6*

*From the left: output with **inSharpness** = 25, output with **inSharpness** = 100*

Errors

List of possible exceptions:

Error type	Description
DomainError	Expainting region (inRegionToExpaint) exceeds the image in ExpaintImage_Bornemann.
DomainError	No pixels available at the edge of inRegionToExpaint in ExpaintImage_Bornemann.

See Also

- InpaintImage – Fills in a region of an image with pixel values interpolated from the borders of the area.
- InpaintImage_Telea – Fills in a region of an image with pixel values interpolated from the borders of the area; uses fast marching method.
- InpaintImage_Bornemann – Fills in a region of an image with pixel values interpolated from the borders of the area; uses fast marching method and coherence analysis.
- ExpaintImage_Telea – Speculatively sets pixel values outside of a region using the fast marching method.

ExpaintImage_Telea

Header: AVL.h
Namespace: avl
Module: FoundationPro

Speculatively sets pixel values outside of a region using the fast marching method.

Syntax

```
void avl::ExpaintImage_Telea
(
 const avl::Image& inImage,
 const avl::Region& inRegionToExpaint,
 const int inExpaintingRadius,
 const int inRange,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRegionToExpaint	const Region&			Part of the image to be expainted
→	inExpaintingRadius	const int	1 - +∞	8	How far to expaint from the region
→	inRange	const int	1 - +∞	6	Defines how far a pixel can be from one currently being inpainted to be considered in calculations
←	outImage	Image&			Output image

Description

This filter expaints a region of an image by **inExpaintingRadius** pixels using the fast marching method. It is a simper version of ExpaintImage_Bornemann.

Examples

*Comparison between low and high values of **inRange**. From the left: input, output with **inRange** = 4, output with **inRange** = 12*

Errors

List of possible exceptions:

Error type	Description
DomainError	Expainting region (inRegionToExpaint) exceeds the image in ExpaintImage_Telea.
DomainError	No pixels available at the edge of inRegionToExpaint in ExpaintImage_Telea.

See Also

- InpaintImage – Fills in a region of an image with pixel values interpolated from the borders of the area.
- InpaintImage_Telea – Fills in a region of an image with pixel values interpolated from the borders of the area; uses fast marching method.
- InpaintImage_Bornemann – Fills in a region of an image with pixel values interpolated from the borders of the area; uses fast marching method and coherence analysis.
- ExpaintImage_Bornemann – Speculatively sets pixel values outside of a region using the fast marching method and coherence analysis.

InpaintImage

Header: AVL.h
Namespace: avl
Module: FoundationPro

Fills in a region of an image with pixel values interpolated from the borders of the area.

Applications: Speculative setting of unknown (or unwanted) pixels. Usually used for preparing incomplete images for further processing.

Syntax

```
void avl::InpaintImage
(
 avl::Image& ioImage,
 float inLambda,
 const avl::ShapeRegion& inRegionToInpaint,
 atl::Optional<const avl::CoordinateSystem2D&> inRoiAlignment
)
```

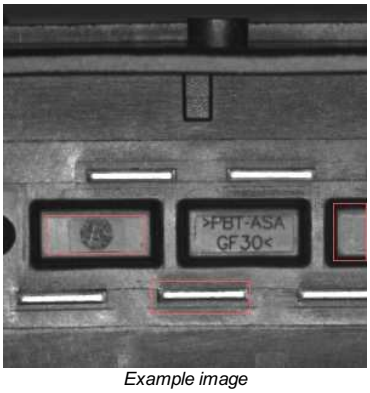
Parameters

	Name	Type	Range	Default	Description
	ioImage	Image&			
	inLambda	float	0.0 - 1.0	0.5f	Ratio between vertical and horizontal interpolation
	inRegionToInpaint	const ShapeRegion&			Region to be inpainted
	inRoiAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the region

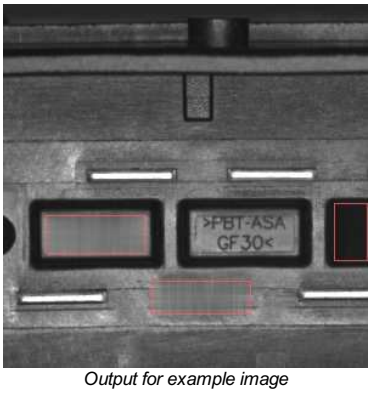
Description

The filter fills in the region (**inRegionToInpaint**) within the image (**inImage**) using the color of the surrounding (external border) of the region. The value of the pixel is a weighted average of horizontal and vertical linear interpolation, where the weight is defined by the parameter **inLambda**. Thus it is a weighted average of four pixels: the closest pixel of the remaining image in four directions, where the weight is reversed distance between pixels times the parameter (which is *inLambda* for the upper and lower and *1.0-inLambda* for the left and right pixels). In case one of those points doesn't exist, it's just omitted in averaging. If none of the point exist, the color is zero.

Examples



Example image



Output for example image

Remarks

It works well when the image around the region is "uniform" in the sense that the (external) border of the region is smooth. If not, the interpolated surface is striped/chequered.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in InpaintImage.

InpaintImage_Bornemann

Header: AVL.h
Namespace: avl
Module: FoundationPro

Fills in a region of an image with pixel values interpolated from the borders of the area; uses fast marching method and coherence analysis.

Applications: Speculative setting of unknown (or unwanted) pixels. This version is giving the best results among reasonably fast traditional methods.

Syntax

```
void avl::InpaintImage_Bornemann
(
 const avl::Image& inImage,
 const avl::Region& inRegionToInpaint,
 const int inRange,
 const float inPreSmoothing,
 const float inPostSmoothing,
 const float inSharpness,
 avl::LuminanceMode::Type inLuminanceMode,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRegionToInpaint	const Region&			
→	inRange	const int	1 - +∞	6	Defines how far a pixel can be from one currently being inpainted to be considered in calculations
→	inPreSmoothing	const float	0.0 - ∞	2.0f	Standard deviation of a gaussian kernel used before inpainting calculations
→	inPostSmoothing	const float	0.0 - ∞	3.0f	Standard deviation of a gaussian kernel used after initial inpainting calculations
→	inSharpness	const float	0.0 - ∞	35.0f	Desired sharpness of edges inside of the inpainted region (higher = sharper)
→	inLuminanceMode	LuminanceMode::Type		YUV	Determines how the luminance of the input image will be computed
←	outImage	Image&			Output image

Description

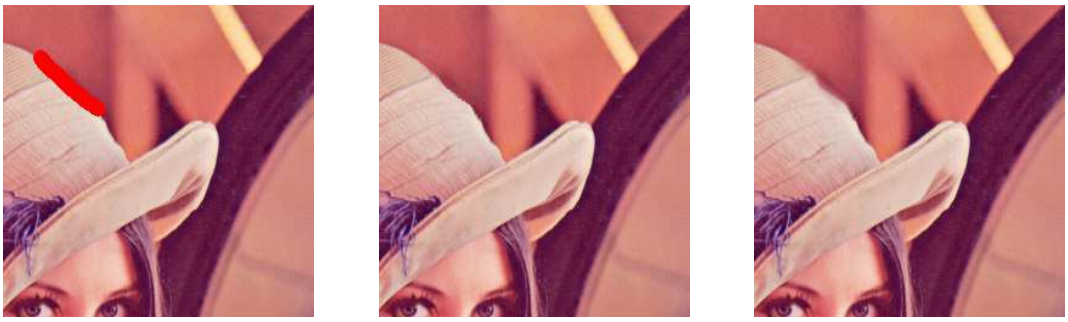
This filter inpaint a region of an image using the fast marching method and coherence flow analysis.

Each inpainted pixel is filled with a color calculated as a weighted average of all already known (i.e. those outside of the inpainting range or those that were already) pixels withing a range specified by the parameter **inRange**. Using a higher range will result in smoother edges withing the inpainted region. The calculation of the weights is done by analyzing the coherence flow of the input image which is previously smoothed with a gaussian kernel with the standard deviation given in **inPreSmoothing**. This means that lower values of this parameter will result in more detail being preserved but it may also introduce unwanted artifacts. After computing the coherence direction and strength, those values are also smoothed to allow for better preservation of angled edges. This smoothing is done using a kernel with the standard deviation given in **inPostSmoothing**. Higher values of this parameter higher quality estimation of angles within the inpainted region but it will also increase the work time of this filter. Additionally one can specify how important the coherence flow information is by using the **inSharpness** parameter (Setting **inSharpness** to zero will mean that coherence information is to be ignored). Higher values will stress the importance of preserving the structure of the inpainted image and will result in sharper edges.

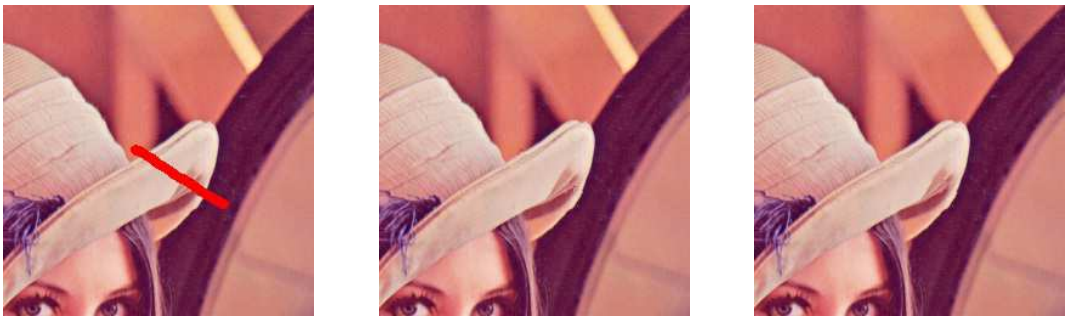
Hints

- To preserve color information of 3-channel images in a way that is more suited for a human eye, set **inLuminanceMode** to YUV.
- Sometimes higher values of **inPreSmoothing** are needed to filter out noise that might interfere with coherence direction calculations.
- Increasing **inSharpness** can help prevent blurred edges withing the inpainting region.
- If round edges are expected withing the inpainting region, high values of **inPostSmoothing** are recommended.

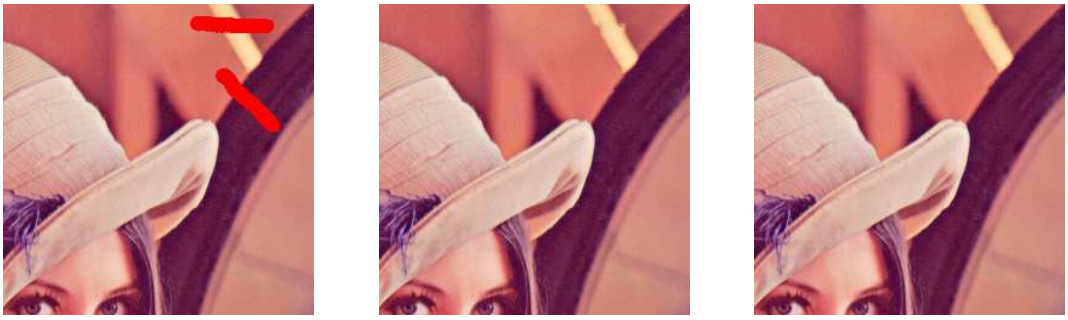
Examples



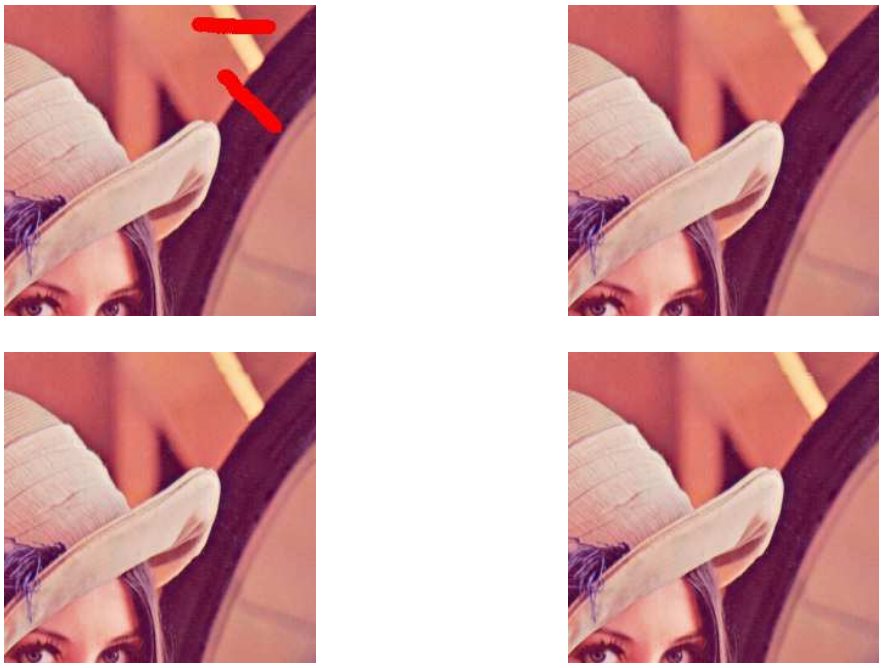
*Comparison between low and high values of **inRange**. From the left: input, output with **inRange** = 4, output with **inRange** = 12*



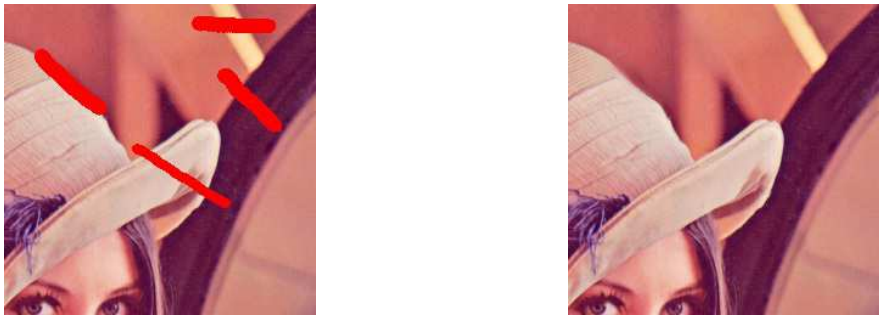
*Comparison between low and high values of **inPreSmoothing**. From the left: input, output with **inPreSmoothing** = 1, output with **inPreSmoothing** = 3*



*Comparison between low and high values of **inPostSmoothing**. From the left: input, output with **inPostSmoothing** = 1, output with **inPostSmoothing** = 6*



*Comparison between different values of **inSharpness**. From top left: input, output with **inSharpness** = 1, output with **inSharpness** = 25, output with **inSharpness** = 100*



*An example inpaint done with **inRange** = 6, **inPreSmoothing** = 1.5, **inPostSmoothing** = 6 and **inSharpness** = 35*

Errors

List of possible exceptions:

Error type	Description
DomainError	Expainting region (inRegionToExpaint) exceeds the image in InpaintImage_Bornemann.
DomainError	No pixels available at the edge of inRegionToExpaint in InpaintImage_Bornemann.

See Also

- InpaintImage – Fills in a region of an image with pixel values interpolated from the borders of the area.
- InpaintImage_Telea – Fills in a region of an image with pixel values interpolated from the borders of the area; uses fast marching method.
- ExpaintImage_Telea – Speculatively sets pixel values outside of a region using the fast marching method.
- ExpaintImage_Bornemann – Speculatively sets pixel values outside of a region using the fast marching method and coherence analysis.

InpaintImage_Telea

Header: AVL.h
Namespace: avl
Module: FoundationPro

Fills in a region of an image with pixel values interpolated from the borders of the area; uses fast marching method.

Applications: Speculative setting of unknown (or unwanted) pixels. This version is trying to do this better than with simple interpolation.

Syntax

```
void avl::InpaintImage_Telea
(
 const avl::Image& inImage,
 const avl::Region& inRegionToInpaint,
 const int inRange,
 avl::Image& outImage
)
```

Parameters

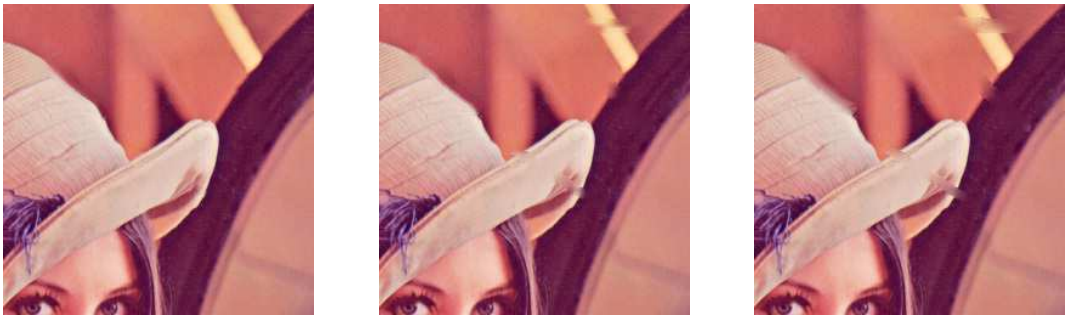
	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRegionToInpaint	const Region&			
→	inRange	const int	1 - +∞	6	Defines how far a pixel can be from one currently being inpainted to be considered in calculations
←	outImage	Image&			Output image

Description

This filter inpaint a region of an image using the fast marching method. It is a simper version of InpaintImage_Bornemann in that it only uses the shape of the inpainting region and doesn't use structural information of the image.

Each inpainted pixel is filled with a color calculated as a weighted average of all already known (i.e. those outside of the inpainting range or those that were already) pixels withing a range specified by the parameter **inRange**. Using a higher range will result in smoother edges withing the inpainted region.

Examples



*Comparison between low and high values of **inRange**. From the left: input, output with **inRange** = 4, output with **inRange** = 12*

Errors

List of possible exceptions:

Error type	Description
DomainError	Expainting region (inRegionToExpaint) exceeds the image in InpaintImage_Telea.
DomainError	No pixels available at the edge of inRegionToInpaint in InpaintImage_Telea.

See Also

- InpaintImage – Fills in a region of an image with pixel values interpolated from the borders of the area.
- InpaintImage_Bornemann – Fills in a region of an image with pixel values interpolated from the borders of the area; uses fast marching method and coherence analysis.
- ExpaintImage_Telea – Speculatively sets pixel values outside of a region using the fast marching method.
- ExpaintImage_Bornemann – Speculatively sets pixel values outside of a region using the fast marching method and coherence analysis.

NormalizeImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Rescales an image linearly, so that its minimum becomes inNewMinimum and the maximum of the remaining pixels becomes inNewMaximum.

Applications: Aims at better using the image's dynamic range to represent an interesting subset of pixel values.

Syntax

```
void avl::NormalizeImage
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 float inNewMinimum,
 float inNewMaximum,
 float inSaturateBrightestFraction,
 float inSaturateDarkestFraction,
 atl::Optional<float> inMinValue,
 atl::Optional<float> inMaxValue,
 avl::Image& outImage,
 float& outA,
 float& outB,
 avl::Region& diagLinearNormalizedRegion
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inImage	const Image&			Input image
➡	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
➡	inNewMinimum	float		0.0f	Desired minimum value of the resulting image
➡	inNewMaximum	float		255.0f	Desired maximum value of the resulting image
➡	inSaturateBrightestFraction	float	0.0 - 1.0	0.0f	Fraction of the brightest pixels skipped during normalization
➡	inSaturateDarkestFraction	float	0.0 - 1.0	0.0f	Fraction of the darkest pixels skipped during normalization
➡	inMinValue	Optional<float>		NIL	Pixels darker than that value will be skipped during normalization
➡	inMaxValue	Optional<float>		NIL	Pixels brighter than that value will be skipped during normalization
⬅	outImage	Image&			Rescaled image
⬅	outA	float&			Multiplicative parameter of the applied linear transformation of pixel values
⬅	outB	float&			Additive parameter of the applied linear transformation of pixel values
🔍	diagLinearNormalizedRegion	Region&			Region of image that has been linearly normalized

Description

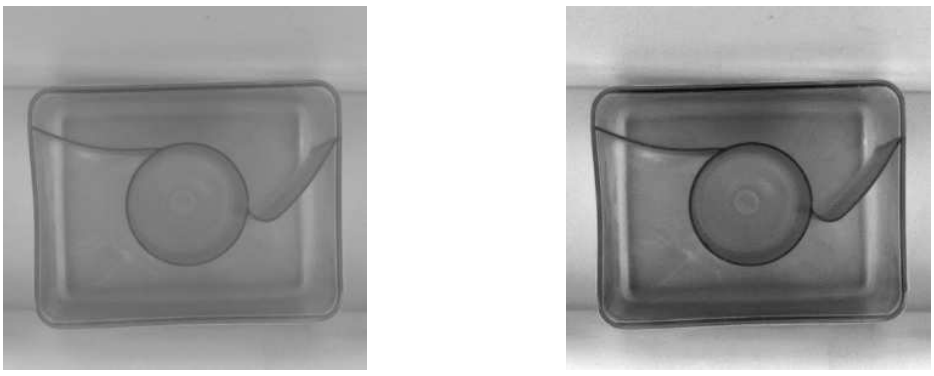
This filter linearly scales the pixel values of an image in order to make the histogram span the desired range of values.

The operation computes the parameters A, B of the linear transform that scales the image values to the desired range and applies the transform computing the results as follows:

$$\begin{aligned} outImage[i,j] &= inImage[i,j] \times A + B \\ outA &= A \\ outB &= B \end{aligned}$$

The **inSaturateBrightestFraction** and **inSaturateDarkestFraction** parameters can be used to make image normalization independent from salt and pepper noise. The normalization skips a chosen fraction of the brightest and the darkest pixels during counting A and B. The brightest and darkest pixels are set to **inNewMaximum** and **inNewMinimum** respectively, while the rest of the image is linearly scaled. A region of the linear scaling is available on **diagLinearNormalizedRegion** diagnostic output. For example, setting **inSaturateBrightestFraction** to 0.01 causes skipping 1 percent of the brightest pixels during counting A and B. Please note that the **inSaturateBrightestFraction** and **inSaturateDarkestFraction** parameters can be non-zero only for images with UInt8, Int8, UInt16 or Int16 pixel type.

Examples



NormalizeImage *run on example image.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty image on input in NormalizeImage.
DomainError	Region exceeds an input image in NormalizeImage.
DomainError	The inSaturationFraction inputs cannot be used for images with this pixel type in NormalizeImage.
DomainError	The sum of inSaturateBrightestFraction and inSaturateDarkestFraction can't be greater than 1 in NormalizeImage.

See Also

- RescalePixels – Applies linear transformation to pixel values.
- ResaturateImage – Sets pixels below the low value to minimum, above the high value to maximum, and interpolates the rest.
- AddToImage – Adds a scalar value to each pixel.

NormalizeLocalBrightness_Gauss

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Normalizes local brightness of the image. Internally uses Gauss smoothing.

Applications: Compensates uneven illumination.

Syntax

```
void avl::NormalizeLocalBrightness_Gauss
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 float inStdDevX,
 atl::Optional<float> inStdDevY,
 float inTargetMean,
 float inGammaValue,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inStdDevX	float	0.0 - ∞	5.0f	Horizontal smoothing standard deviation
→	inStdDevY	Optional<float>	0.0 - ∞	NIL	Vertical smoothing standard deviation
→	inTargetMean	float		128.0f	Target mean brightness
→	inGammaValue	float	0.01 - 8.0	1.0f	Gamma coefficient, where 1.0 is neutral
←	outImage	Image&			Output image

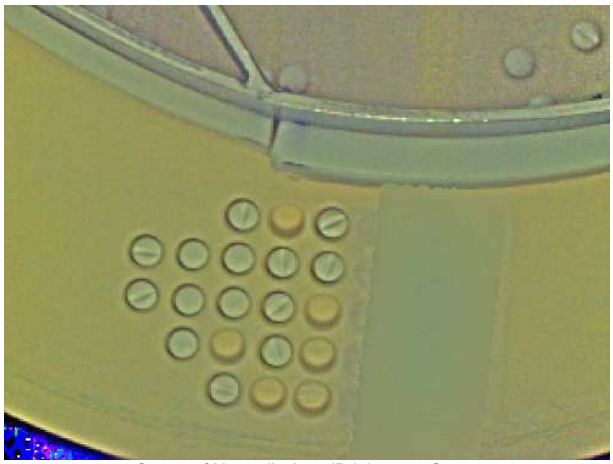
Description

This filter normalizes every pixels brightness to **inTargetMean** based on a local average calculated using **SmoothImage_Gauss** passing to it parameters **inStdDevX** and **inStdDevY**.

Gamma correction can be performed on the image before normalization through the **inGammaValue** parameter.

Examples

Example image



Output of NormalizeLocalBrightness_Gauss

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- NormalizeImage – Rescales an image linearly, so that its minimum becomes inNewMinimum and the maximum of the remaining pixels becomes inNewMaximum.
- NormalizeLocalContrast – Normalizes local contrast of the image using Wallis filter.
- NormalizeLocalBrightness_Mean – Normalizes local brightness of the image. Internally uses Mean smoothing.
- SmoothImage_Gauss – Smooths an image using a gaussian kernel.

NormalizeLocalBrightness_Mean

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Normalizes local brightness of the image. Internally uses Mean smoothing.

Applications: Compensates uneven illumination.

Syntax

```
void avl::NormalizeLocalBrightness_Mean
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 int inRadiusX,
 atl::Optional<int> inRadiusY,
 float inTargetMean,
 float inGammaValue,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inRadiusX	int	0 - ∞	10	Nearly half of the kernel's width (2*R+1)
→	inRadiusY	Optional<int>	0 - ∞	NIL	Nearly half of the kernel's height (2*R+1), or same as inRadiusX
→	inTargetMean	float		128.0f	Target mean brightness
→	inGammaValue	float	0.01 - 8.0	1.0f	Gamma coefficient, where 1.0 is neutral
←	outImage	Image&			Output image

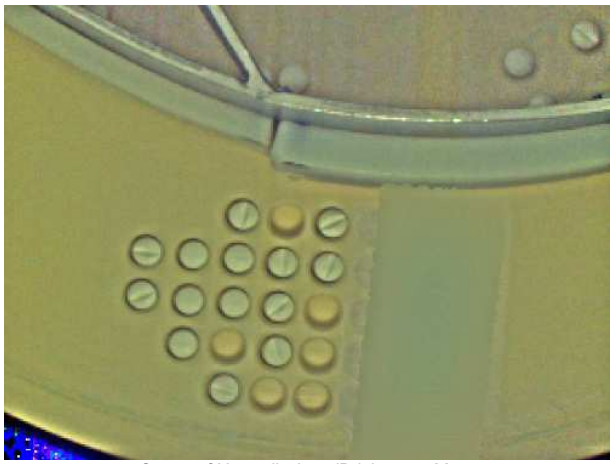
Description

This filter normalizes every pixels brightness to **inTargetMean** based on a local average calculated using **SmoothImage_Mean** passing to it parameters **inRadiusX** and **inRadiusY**.

Gamma correction can be performed on the image before normalization through the **inGammaValue** parameter.

Examples

Example image



Output of NormalizeLocalBrightness_Mean

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- NormalizeImage – Rescales an image linearly, so that its minimum becomes inNewMinimum and the maximum of the remaining pixels becomes inNewMaximum.
- NormalizeLocalContrast – Normalizes local contrast of the image using Wallis filter.
- NormalizeLocalBrightness_Gauss – Normalizes local brightness of the image. Internally uses Gauss smoothing.
- SmoothImage_Mean – Smooths an image by averaging pixels within a rectangular kernel.

NormalizeLocalContrast

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Normalizes local contrast of the image using Wallis filter.

Syntax

```
void avl::NormalizeLocalContrast
(
 const avl::Image& inMonoImage,
 atl::Optional<const avl::Region&> inRoi,
 const float inTargetMean,
 const float inTargetVariance,
 const int inUniformnessScale,
 const float inBrightnessPreserveRatio,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inMonoImage	const Image&			Monochromatic input image
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inTargetMean	const float		128.0f	Target local mean
→	inTargetVariance	const float	0.1 - ∞	90.0f	Target local variance
→	inUniformnessScale	const int	1 - +∞	10	Radius of neighbourhood to uniformize
→	inBrightnessPreserveRatio	const float	0.0 - 1.0	0.1f	How much of original brightness to be kept
←	outImage	Image&			Output image

Requirements

For input **inMonoImage** only pixel formats are supported: 1□uint8, 1□int8, 1□uint16, 1□int16, 1□int32, 1□real.

Read more about pixel formats in Image documentation.

Description

This filter resaturates every pixel of monochromatic image **inMonoImage** so that its neighbourhood matches approximately average of **inTargetMean** and variance of **inTargetVariance** using Wallis algorithm.

Parameter **inUniformnessScale** determines the radius of the neighbourhood.

Parameter **inBrightnessPreserveRatio** determines how much of the original brightness is kept on the resulting image.

New value of pixel is given by:

$$P_{new} = A\,P_{old} + B$$

where

$$A = \frac{Var_{new}}{\frac{90}{CFC} + Var_{loc}}$$

$$B = Mean_{new}(1 - BPR) + Mean_{loc}(BPR - A)$$

and

- Dev_{loc} and $Mean_{loc}$ are local deviation and mean respectively, taken from the square neighbourhood of radius **inUniformnessScale**
- Dev_{new} and $Mean_{new}$ are target deviation and mean respectively, **inTargetVariance** and **inTargetMean**
- CFC is Contrast Force Constant, set to 0.9
- BPR is **inBrightnessPreserveRatio**

Examples

Example image

Output of NormalizeLocalContrast filter

Previous image with NormalizeImage filter applied to it

Remarks

After the filter is run on the image, the neighbourhoods of pixels **do not** have desired variance and average. First reason for that is presence of constants BPR and CFC. Second reason is that forcing matching given average would require solving system of as many linear equations as there are pixels on the image.

Errors

List of possible exceptions:

Error type	Description
DomainError	Mono image expected on input in NormalizeLocalContrast.
DomainError	Region exceeds an input image in NormalizeLocalContrast.
DomainError	Not supported inMonoImage pixel format in NormalizeLocalContrast. Supported formats: 1xUInt8, 1xInt8, 1xUInt16, 1xInt16, 1xInt32, 1xReal.

See Also

- RescalePixels – Applies linear transformation to pixel values.
- ResaturateImage – Sets pixels below the low value to minimum, above the high value to maximum, and interpolates the rest.
- AddToImage – Adds a scalar value to each pixel.
- NormalizeImage – Rescales an image linearly, so that its minimum becomes inNewMinimum and the maximum of the remaining pixels becomes inNewMaximum.

SharpenImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Enhances contrast of an image so that it appears sharper.

Applications: Use this filter only to improve human perception, not to preprocess an image for analysis.

Syntax

```
void avl::SharpenImage
(
 const avl::Image& inImage,
 float inContrastFactor,
 avl::KernelShape::Type inKernel,
 int inRadiusX,
 atl::Optional<int> inRadiusY,
 avl::Image& outImage
)
```

Parameters

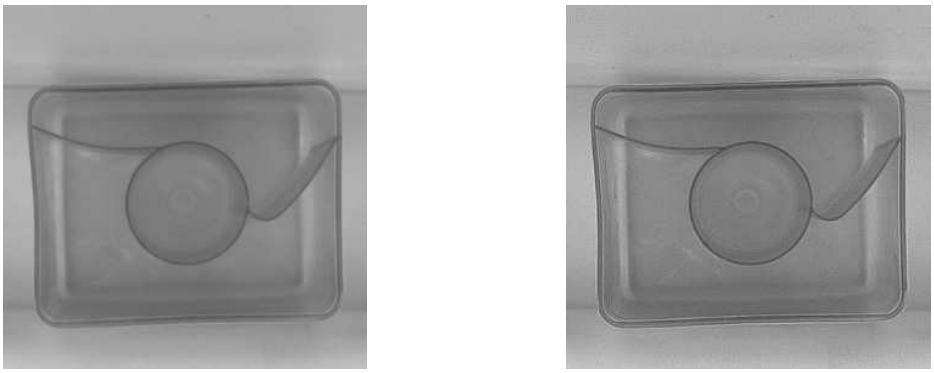
	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inContrastFactor	float	0.0 - ∞	1.0f	Value representing the strength of the contrast enhancement
→	inKernel	KernelShape::Type			Kernel shape
→	inRadiusX	int	0 - ∞	1	Nearly half of the kernel's width (2*R+1)
→	inRadiusY	Optional<int>	0 - ∞	NIL	Nearly half of the kernel's height (2*R+1), or same as inRadiusX
←	outImage	Image&			Output image

Description

This filter increases the contrast of the image by multiplying by **inContrastFactor** the difference between a pixel value and its corresponding pixel value in the smoothed input image:

$$outImage[i,j] = inImage[i,j] + inContrastFactor \times (inImage[i,j] - smoothedImage[i,j])$$

Examples



SharpenImage *run on example image with* ***inContrastFactor*** *= 2.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

This operation is optimized for SSE41 technology for pixels of type: UINT8.

This operation is optimized for AVX2 technology for pixels of type: UINT8.

This operation is optimized for NEON technology for pixels of type: UINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

38. Geometry 2D Interpolations

Table of content:

AverageAngle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the average of the input angles using points on a unit circle.

Syntax

```
void avl::AverageAngle
(
 const atl::Array<float>& inAngles,
 avl::AngleRange::Type inAngleRange,
 float& outAverageAngle
)
```

Parameters

	Name	Type	Default	Description
→	inAngles	const Array<float>&		
→	inAngleRange	AngleRange::Type	_0_180	
←	outAverageAngle	float&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array on input in AverageAngle.

AveragePoint

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the middle point of two input points.

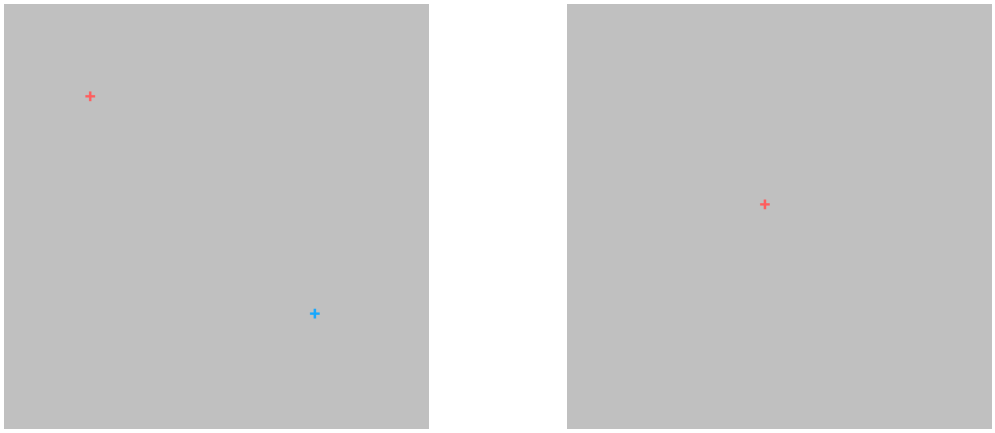
Syntax

```
void avl::AveragePoint
(
 const avl::Point2D& inPoint1,
 const avl::Point2D& inPoint2,
 avl::Point2D& outPoint
)
```

Parameters

	Name	Type	Default	Description
→	inPoint1	const Point2D&		
→	inPoint2	const Point2D&		
←	outPoint	Point2D&		

Examples



***AveragePoint** performed on two points.*

LerpAngles

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Linearly interpolates between two angles in the direction of minimum turn.

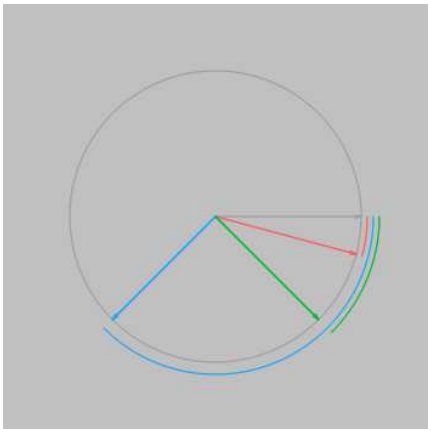
Syntax

```
void avl::LerpAngles
(
 float inAngle0,
 float inAngle1,
 atl::Optional<avl::RotationDirection::Type> inRotationDirection,
 avl::AngleRange::Type inAngleRange,
 float inLambda,
 bool inInverse,
 float& outAngle
)
```

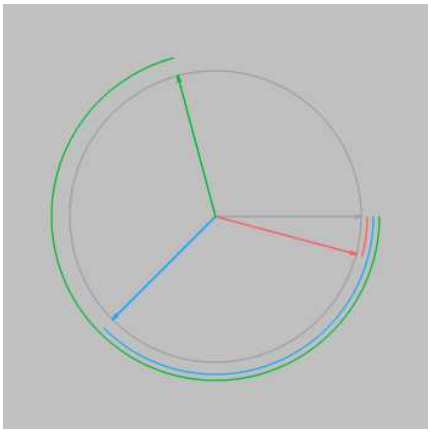
Parameters

	Name	Type	Default	Description
→	inAngle0	float		
→	inAngle1	float		
→	inRotationDirection	Optional<RotationDirection::Type>	NIL	Clockwise, counter-clockwise or auto
→	inAngleRange	AngleRange::Type	_0_180	
→	inLambda	float		Interpolation between the input angles where 0.0 value is equal to inAngle0 and 1.0 to inAngle1
→	inInverse	bool		
←	outAngle	float&		

Examples



LerpAngles *performed on red* ***inAngle0*** *= 15 and blue* ***inAngle1*** *= 135 with* ***inLambda*** *= 0,25 and* ***inAngleRange*** *= 0-360. Green is the resulting* ***outAngle****.*



LerpAngles *performed on red* ***inAngle0*** *= 15 and blue* ***inAngle1*** *= 135 with* ***inLambda*** *= -1,0 and* ***inAngleRange*** *= 0-360. Green is the resulting* ***outAngle****.*

Remarks

Please note that:

- interpolation begins at **inAngle0**,
- for positive **inLambda** values interpolation is performed in the direction given in the **inRotationDirection** parameter while for negative - in the inverted direction.

Hence when **inLambda** = 0,0, **outAngle** is equal to **inAngle0**, while for **inLambda** = 1,0 it's the same as **inAngle1** and for **inLambda** = -1,0 - as (**inAngle0** - **inAngle1**). Other **inLambda** values interpolate between the input angles and beyond.

LerpPaths

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Linearly interpolates between two paths.

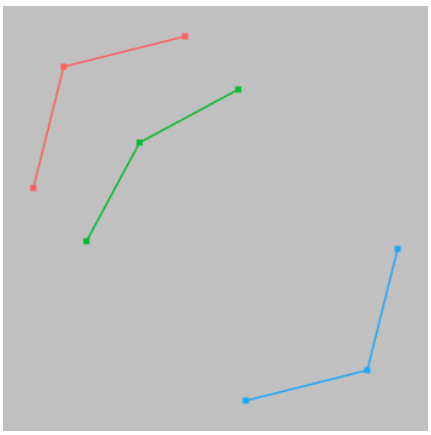
Syntax

```
void avl::LerpPaths
(
 const avl::Path& inPath0,
 const avl::Path& inPath1,
 const float inLambda,
 avl::Path& outPath
)
```

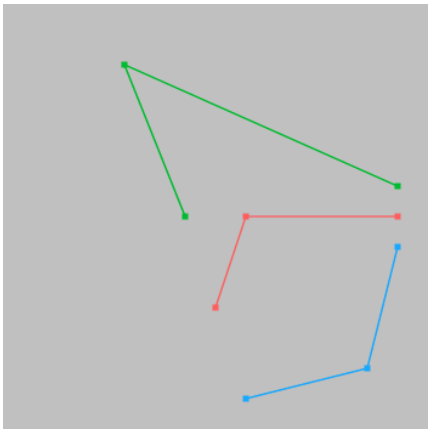
Parameters

	Name	Type	Range	Default	Description
→	inPath0	const Path&			
→	inPath1	const Path&			First input path
→	inLambda	const float	-∞ - ∞	0.5f	Interpolation between the input paths where 0.0 value is equal to inPath0 and 1.0 to inPath1
←	outPath	Path&			Output path

Examples



***LerpPaths** performed on red **inPath0** and blue **inPath1** with **inLambda** = 0,25. Green is the resulting **outPath**.*



***LerpPaths** performed on red **inPath0** and blue **inPath1** with **inLambda** = -1,0. Green is the resulting **outPath**.*

Remarks

Please note that:

- interpolation begins at **inPath0**,
- for positive **inLambda** values interpolation is performed in the direction of **inPath1** while for negative - in the direction of a path which is the result of the following operation: (**inPath0** - **inPath1**), in example a path consisting of **inPath1** points mirrored by the corresponding **inPath0** points.

Hence when **inLambda** = 0,0, **outPath** is equal to **inPath0**, while for **inLambda** = 1,0 it's the same as **inPath1** and for **inLambda** = -1,0 - as (**inPath0** - **inPath1**). Other **inLambda** values interpolate between the input paths and beyond.

Errors

List of possible exceptions:

Error type	Description
DomainError	Numbers of points in paths are not the same in LerpPaths.
DomainError	Open and closed paths on input in LerpPaths.

LerpPoints

Header: AVL.h
Namespace: avl
Module: FoundationLite

Linearly interpolates between two points.

Syntax

```
void avl::LerpPoints
(
 const avl::Point2D& inPoint0,
 const avl::Point2D& inPoint1,
 float inLambda,
 avl::Point2D& outPoint
)
```

Parameters

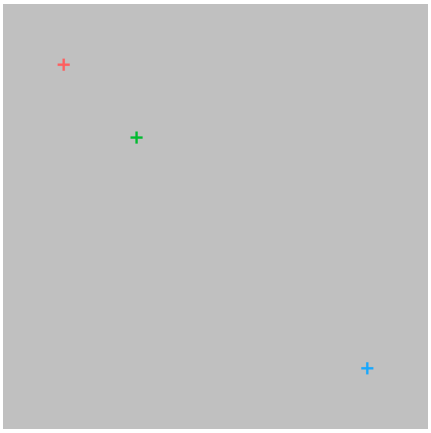
	Name	Type	Range	Default	Description
→	inPoint0	const Point2D&			
→	inPoint1	const Point2D&			
→	inLambda	float	-∞ -∞	0.5f	Interpolation between the input points where 0.0 value is equal to inPoint0 and 1.0 to inPoint1
←	outPoint	Point2D&			

In-place Processing

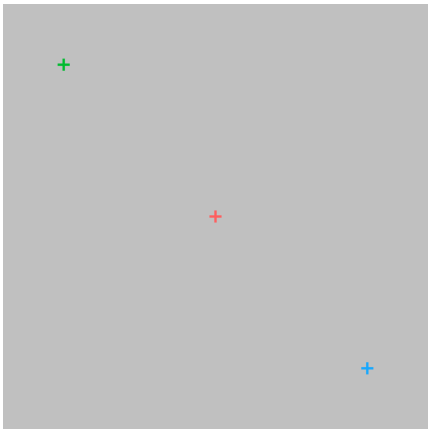
This function supports in-place data processing - you can pass the same reference to **inPoint1** and **outPoint**

Read more about In-place Computation.

Examples



***LerpPoints** performed on red **inPoint0** and blue **inPoint1** with **inLambda** = 0,25. Green is the resulting **outPoint**.*



***LerpPoints** performed on red **inPoint0** and blue **inPoint1** with **inLambda** = -1,0. Green is the resulting **outPoint**.*

Remarks

Please note that:

- interpolation begins at **inPoint0**,
- for positive **inLambda** values interpolation is performed in the direction of **inPoint1** while for negative - in the direction of a point which is the result of the following operation: (**inPoint0** - **inPoint1**), in example a point acquired by mirroring **inPoint1** by **inPoint0**.

Hence when **inLambda** = 0,0, **outPoint** is equal to **inPoint0**, while for **inLambda** = 1,0 it's the same as **inPoint1** and for **inLambda** = -1,0 - as (**inPoint0** - **inPoint1**). Other **inLambda** values interpolate between the input points and beyond.

Also in **AVL Lite**

LerpSegments

Header: AVL.h
Namespace: avl
Module: FoundationLite

Linearly interpolates between two segments.

Syntax

```
void avl::LerpSegments
(
 const avl::Segment2D& inSegment0,
 const avl::Segment2D& inSegment1,
 float inLambda,
 bool inIgnoreOrientation,
 avl::Segment2D& outSegment
)
```

Parameters

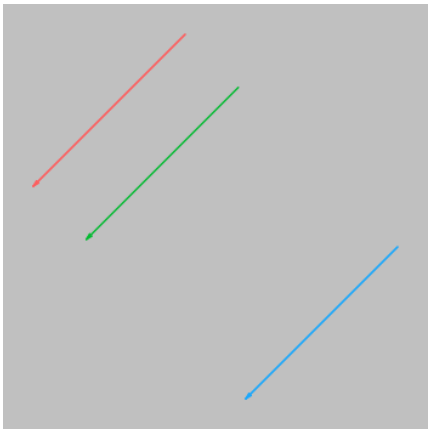
	Name	Type	Range	Default	Description
→	inSegment0	const Segment2D&			
→	inSegment1	const Segment2D&			
→	inLambda	float	-∞ - ∞	0.5f	Interpolation between the input segments where 0.0 value is equal to inSegment0 and 1.0 to inSegment1
→	inIgnoreOrientation	bool			
←	outSegment	Segment2D&			

In-place Processing

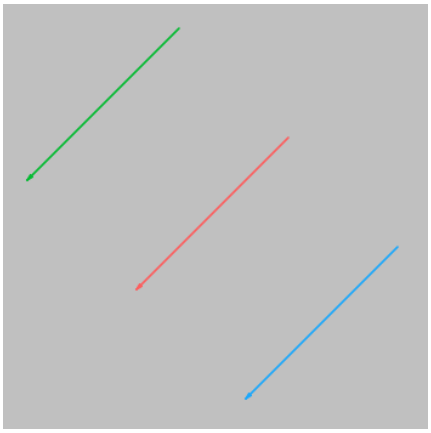
This function supports in-place data processing - you can pass the same reference to **inSegment1** and **outSegment**

Read more about In-place Computation.

Examples



***LerpSegments** performed on red **inSegment0** and blue **inSegment1** with **inLambda** = 0,25. Green is the resulting **outSegment**.*



***LerpSegments** performed on red **inSegment0** and blue **inSegment1** with **inLambda** = -1,0. Green is the resulting **outSegment**.*

Remarks

Please note that:

- interpolation begins at **inSegment0**,
- for positive **inLambda** values interpolation is performed in the direction of **inSegment1** while for negative - in the direction of a segment which is the result of the following operation: (**inSegment0** - **inSegment1**), in example a segment consisting of **inSegment1** points mirrored by the corresponding **inSegment0** points.

Hence when **inLambda** = 0,0, **outSegment** is equal to **inSegment0**, while for **inLambda** = 1,0 it's the same as **inSegment1** and for **inLambda** = -1,0 - as (**inSegment0** - **inSegment1**). Other **inLambda** values interpolate between the input segments and beyond.

Also in AVL Lite

LerpVectors

Header: AVL.h
Namespace: avl
Module: FoundationLite

Linearly interpolates between two vectors.

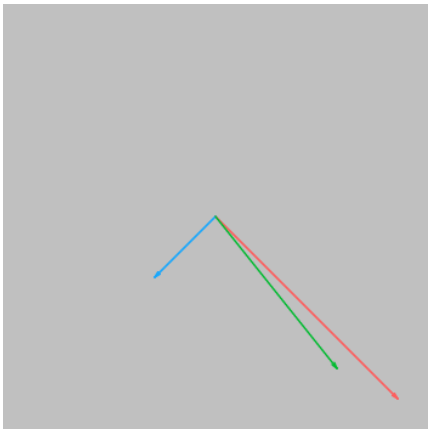
Syntax

```
void avl::LerpVectors
(
 const avl::Vector2D& inVector0,
 const avl::Vector2D& inVector1,
 const float inLambda,
 avl::Vector2D& outVector
)
```

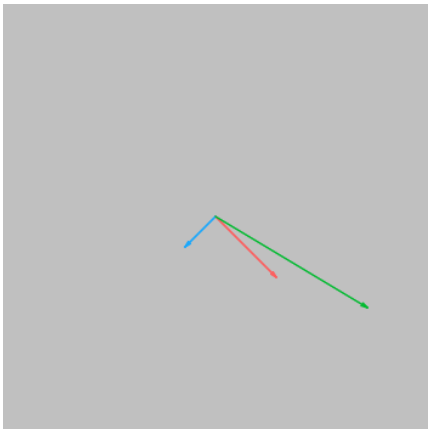
Parameters

	Name	Type	Range	Default	Description
→	inVector0	const Vector2D&			
→	inVector1	const Vector2D&			
→	inLambda	const float	-∞ - ∞	0.5f	Interpolation between the input vectors where 0.0 value is equal to inVector0 and 1.0 to inVector1
←	outVector	Vector2D&			

Examples



LerpVectors *performed on red* ***inVector0*** *= (150, 150) and blue* ***inVector1*** *= (-50, 50) with* ***inLambda*** *= 0,25. Green is the resulting* ***outVector****.*



LerpVectors *performed on red* ***inVector0*** *= (50, 50) and blue* ***inVector1*** *= (-25, 25) with* ***inLambda*** *= -1,0. Green is the resulting* ***outVector****.*

Remarks

Please note that:

- interpolation begins at **inVector0**,
- for positive **inLambda** values interpolation is performed in the direction of **inVector1** while for negative - in the direction of a vector which, when visualized with the same base point, ends in the point acquired by mirroring the head of **inVector0** by a line perpendicular to the vector (**inVector0** - **inVector1**).

Hence when **inLambda** = 0,0, **outVector** is equal to **inVector0**, while for **inLambda** = 1,0 it's the same as **inVector1** and for **inLambda** = -1,0 - as the previously described vector. Other **inLambda** values interpolate between the input vectors and beyond.

Also in AVL Lite

LerpVectors_Radial

Header: AVL.h
Namespace: avl
Module: FoundationLite

Linearly interpolates between two radial vectors.

Syntax

```
void avl::LerpVectors_Radial
(
 const avl::Vector2D& inVector0,
 const avl::Vector2D& inVector1,
 atl::Optional<avl::RotationDirection::Type> inRotationDirection,
 const float inLambda,
 avl::Vector2D& outVector
)
```

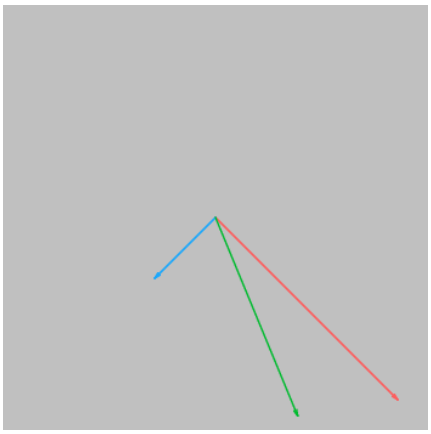
Parameters

	Name	Type	Range	Default	Description
→	inVector0	const Vector2D&			
→	inVector1	const Vector2D&			
→	inRotationDirection	Optional<RotationDirection::Type>		NIL	Clockwise, counter-clockwise or auto
→	inLambda	const float	-∞ - ∞	0.5f	Interpolation between the input vectors where 0.0 value is equal to inVector0 and 1.0 to inVector1
←	outVector	Vector2D&			

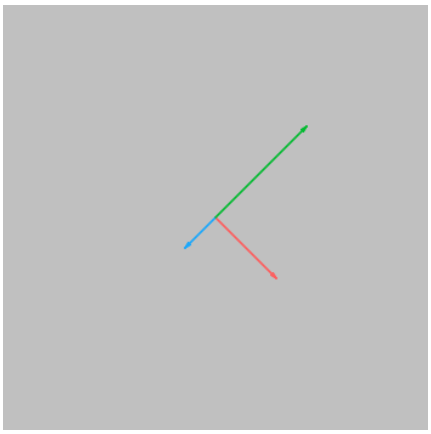
Description

Note that because of inaccuracies of floating-point arithmetic, some geometric operations (including this one) may lead to unpredictable results for degenerated cases. In this filter such a case occurs when a zero vector is given on input.

Examples



***LerpVectors** performed on red **inVector0** = (150, 150) and blue **inVector1** = (-50, 50) with **inLambda** = 0,25 and **inRotationDirection** = Clockwise. Green is the resulting **outVector**.*



***LerpVectors** performed on red **inVector0** = (50, 50) and blue **inVector1** = (-25, 25) with **inLambda** = -1,0 and **inRotationDirection** = Clockwise. Green is the resulting **outVector**.*

Remarks

Please note that:

- interpolation begins at **inVector0**,
- for positive **inLambda** values interpolation is performed in the direction of **inVector1** while for negative - in the direction of a vector acquired by mirroring **inVector1** by a line determined by **inVector0** and its length properly adjusted.

Hence when **inLambda** = 0,0, **outVector** is equal to **inVector0**, while for **inLambda** = 1,0 it's the same as **inVector1** and for **inLambda** = -1,0 - as the previously described vector. Other **inLambda** values interpolate between the input vectors and beyond.

MedianAngle

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the median angle of the input angles.

Applications: This is a substitute for arithmetic average which cannot be done well for angles. For example the average of 1 and 359 is incorrectly 180 (we think 0).

Syntax

```
void avl::MedianAngle
(
 const atl::Array<float>& inAngles,
 avl::AngleRange::Type inAngleRange,
 float& outMedianAngle
)
```

Parameters

	Name	Type	Default	Description
→	inAngles	const Array<float>&		
→	inAngleRange	AngleRange::Type	_0_180	
←	outMedianAngle	float&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array on input in MedianAngle.

39. Path Combinators

Table of content:

AveragePath

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the average of two paths (of equal size and type) point by point.

Syntax

```
void avl::AveragePath
(
 const avl::Path& inPath1,
 const avl::Path& inPath2,
 avl::Path& outPath
)
```

Parameters

	Name	Type	Default	Description
→	inPath1	const Path&		Input path
→	inPath2	const Path&		Input path
←	outPath	Path&		Output path

In-place Processing

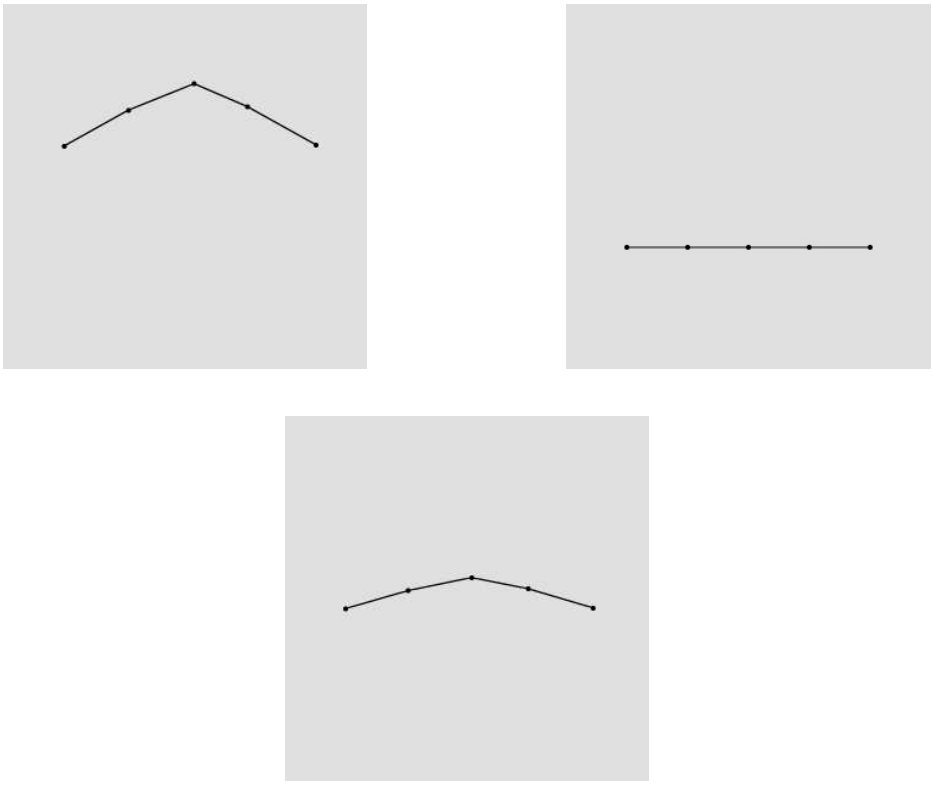
This function supports in-place data processing - you can pass the same reference to **inPath1** and **outPath**, **inPath2** and **outPath**

Read more about In-place Computation.

Description

The operation computes the average of two paths of equal **size** and **type** (open/closed). Each point of the resulting path is an average of the corresponding points in the input paths.

Examples



AveragePath *run on the sample paths.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Input paths have unequal size in AveragePath.
DomainError	Open/closed paths mismatch in AveragePath.

ConcatenatePaths

Header: AVL.h
Namespace: avl
Module: FoundationLite

Joins up to four open paths.

Syntax

```
void avl::ConcatenatePaths
(
 const avl::Path& inPath1,
 const avl::Path& inPath2,
 const avl::Path& inPath3,
 const avl::Path& inPath4,
 bool inClose,
 avl::Path& outPath
)
```

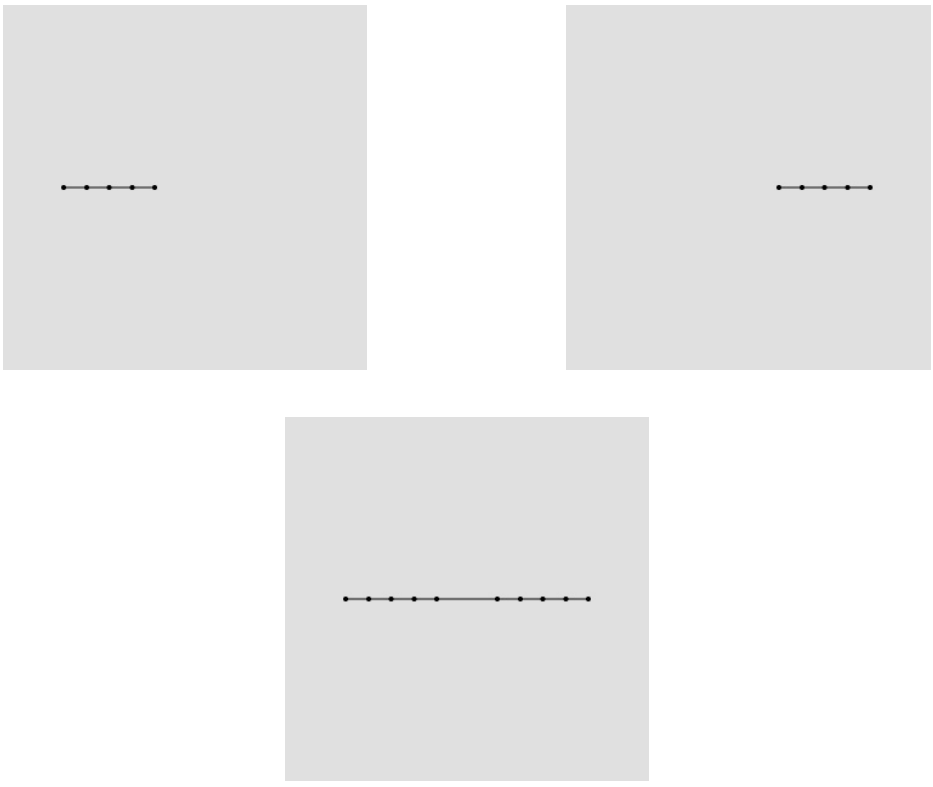
Parameters

	Name	Type	Default	Description
→	inPath1	const Path&		Input path
→	inPath2	const Path&		Input path
→	inPath3	const Path&		Input path
→	inPath4	const Path&		Input path
→	inClose	bool		If set to true adding in result a segment between its first point and the last one, therefore producing a closed path
←	outPath	Path&		Output path

Description

The operations joins two open paths adding a segment connecting the end of the **inPath1** to the begin of **inPath2**.

Examples



ConcatenatePaths *run on the sample paths.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Closed path on input in ConcatenatePaths.

ConcatenatePaths_OfArray

Header: AVL.h
Namespace: avl
Module: FoundationLite

Joins open paths of an array.

Syntax

```
void avl::ConcatenatePaths_OfArray
(
 const atl::Array<avl::Path>& inPathArray,
 bool inClose,
 avl::Path& outPath
)
```

Parameters

	Name	Type	Default	Description
→	inPathArray	const Array<Path>&		Input paths
→	inClose	bool		If set to true adding in result a segment between its first point and the last one, therefore producing a closed path
←	outPath	Path&		Output path

Description

Array version of ConcatenatePaths.

Errors

List of possible exceptions:

Error type	Description
DomainError	Closed path on input in ConcatenatePaths_OfArray.

See Also

- ConcatenatePaths – Joins up to four open paths.

ConcatenatePaths_OfLoop

Header: AVL.h
Namespace: avl
Module: FoundationLite

Joins open paths appearing in consecutive iterations.

Syntax

```
void avl::ConcatenatePaths_OfLoop
(
 ConcatenatePaths_OfLoopState& ioState,
 const avl::Path& inPath,
 bool inClose,
 avl::Path& outPath
)
```

Parameters

	Name	Type	Default	Description
	ioState	ConcatenatePaths_OfLoopState&		Object used to maintain state of the function.
	inPath	const Path&		Input path
	inClose	bool		If set to true adding in result a segment between its first point and the last one, therefore producing a closed path
	outPath	Path&		Output path

Description

Loop version of ConcatenatePaths.

Errors

List of possible exceptions:

Error type	Description
DomainError	Closed path on input in ConcatenatePaths_OfLoop.

See Also

- ConcatenatePaths – Joins up to four open paths.

JoinAdjacentPaths

Header: AVL.h
Namespace: avl
Module: FoundationPro

Joins those paths of an array which endpoints lie near enough.

Syntax

```
void avl::JoinAdjacentPaths
(
 const atl::Array<avl::Path>& inPaths,
 float inMaxDistance,
 float inMaxAngle,
 atl::Optional<float> inMaxDeviation,
 float inExcessTrim,
 atl::Optional<float> inEndingLength,
 avl::PathJoiningMethod::Type inJoiningMethod,
 avl::PathJoiningAngleMeasure::Type inAngleMeasure,
 bool inIgnorePathEndsOrder,
 bool inAllowCycles,
 float inMinPathLength,
 atl::Array<avl::Path>& outPaths,
 atl::Optional<atl::Array<avl::Path>&> outMatchedPieces = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPaths	const Array<Path>&			
→	inMaxDistance	float	0.0 - ∞	10.0f	Maximal distance between paths that can be joined
→	inMaxAngle	float	0.0 - 180.0	30.0f	Maximal allowed angle between paths being joined
→	inMaxDeviation	Optional<float>	0.0 - ∞	NIL	Maximal allowed thickness of a minimal stripe containing both paths being joined
→	inExcessTrim	float	0.0 - ∞		Length of the part of each path to be removed from its both sides
→	inEndingLength	Optional<float>	0.0 - ∞	0.0f	Determines the length of the path end used for path angle computing
→	inJoiningMethod	PathJoiningMethod::Type		AddBridge	Determines a method to join two paths in one
→	inAngleMeasure	PathJoiningAngleMeasure::Type		InputPathOrientation	Determines a method to measure the angle between two input paths
→	inIgnorePathEndsOrder	bool		True	If set to false, only the end of a path can be joined with the begin of another path
→	inAllowCycles	bool		True	Determines if cycles can be created during joining process
→	inMinPathLength	float	0.0 - ∞	0.0f	Minimal length of a path
←	outPaths	Array<Path>&			
←	outMatchedPieces	Optional<Array<Path>&>		NIL	Input paths that have been joined with another path

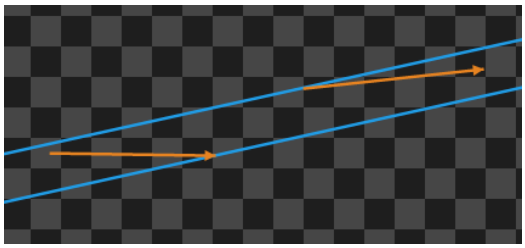
Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outMatchedPieces**.

Read more about Optional Outputs.

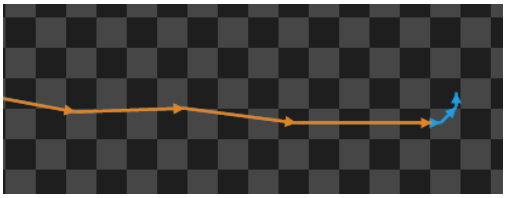
Description

The operation repeatedly connects the pair of endpoints of open paths of an array until no such pair can be connected. If an endpoint can be joined with more than one another, the endpoint which generates smallest joining value is chosen. The joining value is computed by combining distance between endpoints being joined, their angle difference and their deviation value. An endpoint can be joined with another one in its vicinity only if the distance between them is not greater than **inMaxDistance**. Their angle difference, which is computed in one of available ways depending on **inAngleMeasure** value, cannot be greater than **inMaxAngle**. Their deviation value, which is the thickness of a minimal stripe containing both path endings being joined, cannot exceed **inMaxDeviation**.



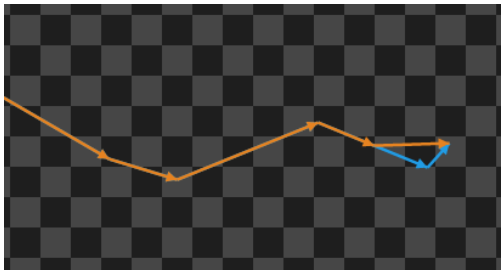
The deviation value for two orange segments is the distance between two parallel blue lines.

Before joining process begins, the input paths are shortened by **inExcessTrim** on both their ends. The parameter makes it possible to use the filter for paths that can be a bit curly on their ends.



*Only orange part of the path takes part in joining process. The curly blue path ending is removed by **inExcessTrim** parameter.*

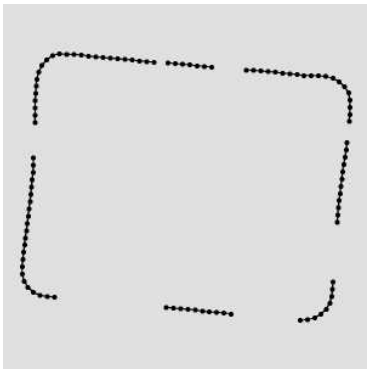
While joining takes place, the **inEndingLength** parameter determines what part of the path ending is used to determine the angle between two paths. If it is set to Nil, the segment connecting path begin and path end is decisive.

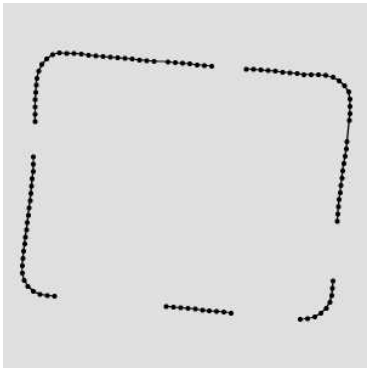


*The last orange segment is taken into account while determining angle between two paths. The omitted blue path ending has length **inEndingLength**.*

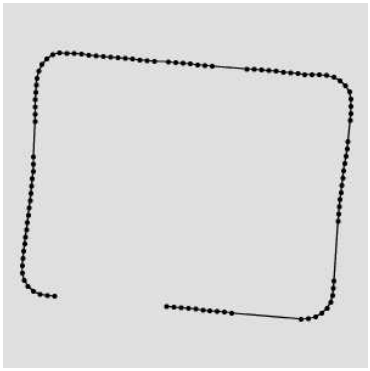
There are also two flags controlling the joining process. If the **inIgnorePathEndsOrder** is set, a path begin can also be joined with another path begin and a path end with another path end. Otherwise a path begin can be joined with a path end only. The **inAllowCycles** parameter determines if a cycle is allowed to emerge during joining. Finally, after the joining phase paths that are shorter than **inMinPathLength** are removed from the final results.

Examples





JoinAdjacentPaths *run on the sample path array with* ***inMaxDistance*** *= 25.*



JoinAdjacentPaths *run on the sample path array with* ***inMaxDistance*** *= 1000.*

See Also

- ClosePath – Adds the segment connecting the last point with the first one in a path.

PathLineIntersections

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes the common points of a path and a line.

Syntax

```
void avl::PathLineIntersections
(
 const avl::Path& inPath,
 const avl::Line2D& inLine,
 atl::Array<avl::Point2D>& outIntersectionPoints,
 atl::Optional<atl::Array<int>&> outSegmentIndices = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
→	inLine	const Line2D&		Input line
←	outIntersectionPoints	Array<Point2D>&		Intersections between the path and the line
←	outSegmentIndices	Optional<Array<int>&>	NIL	Indices of the segments of the path which generate found intersection points

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outSegmentIndices**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite line on input in PathLineIntersections.

PathPathIntersections

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes the common points of two paths.

Syntax

```
void avl::PathPathIntersections
(
 const avl::Path& inPath1,
 const avl::Path& inPath2,
 atl::Array<avl::Point2D>& outIntersectionPoints,
 atl::Optional<atl::Array<int>&> outSegmentIndices1 = atl::NIL,
 atl::Optional<atl::Array<int>&> outSegmentIndices2 = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPath1	const Path&		Input path
→	inPath2	const Path&		Input path
←	outIntersectionPoints	Array<Point2D>&		Intersections between the input paths
←	outSegmentIndices1	Optional<Array<int>&>	NIL	Indices of the segments of the first path which generate found intersection points
←	outSegmentIndices2	Optional<Array<int>&>	NIL	Indices of the segments of the second path which generate found intersection points

Optional Outputs

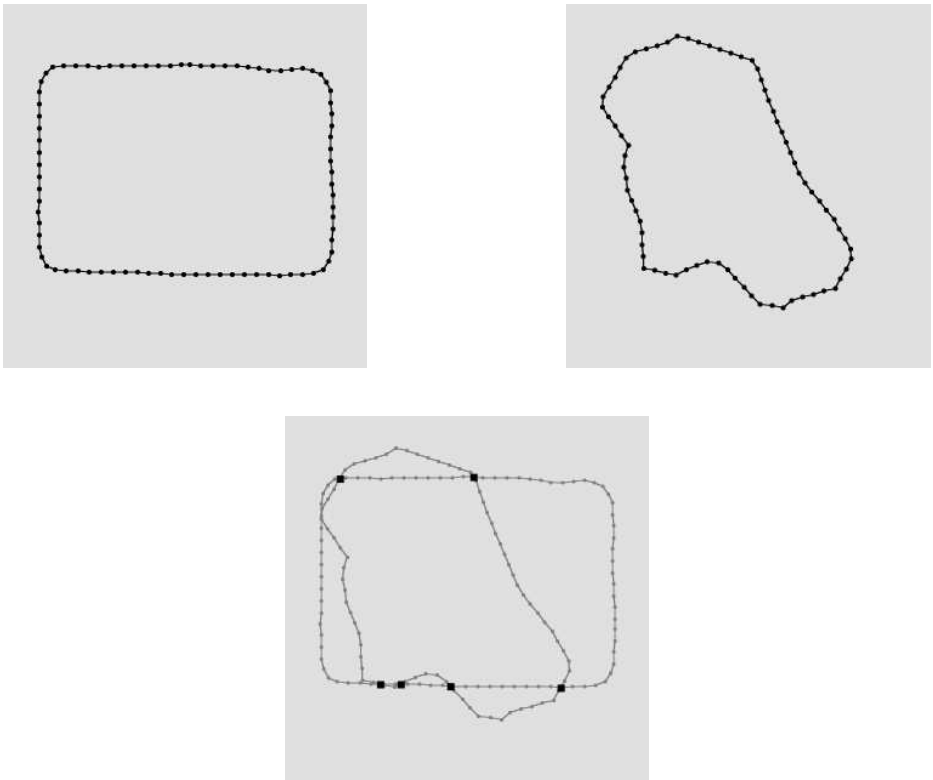
The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outSegmentIndices1**, **outSegmentIndices2**.

Read more about Optional Outputs.

Description

The operation computes an array of the intersection points of two given paths.

Examples



PathPathIntersections *run on the sample paths. The resulting* ***outIntersectionPoints*** *were drawn onto the input paths.*

PathSegmentIntersections

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes the common points of a path and a segment.

Syntax

```
void avl::PathSegmentIntersections
(
 const avl::Path& inPath,
 const avl::Segment2D& inSegment,
 atl::Array<avl::Point2D>& outIntersectionPoints,
 atl::Optional<atl::Array<int>&> outSegmentIndices = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
→	inSegment	const Segment2D&		Input segment
←	outIntersectionPoints	Array<Point2D>&		Intersections between the path and the segment
←	outSegmentIndices	Optional<Array<int>&>	NIL	Indices of the segments of the path which generate found intersection points

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outSegmentIndices**.

Read more about Optional Outputs.

SplitPathByLine

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Splits path in common points of a path and a line.

Syntax

```
void avl::SplitPathByLine
(
 const avl::Path& inPath,
 const avl::Line2D& inLine,
 atl::Array<avl::Path>& outPaths
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Path being split
→	inLine	const Line2D&		Line used for splitting
←	outPaths	Array<Path>&		Paths arisen from splitting the initial path

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite line on input in SplitPathByLine.

SplitPathByPath

Header: AVL.h
Namespace: avl
Module: FoundationPro

Splits path in common points of two paths.

Syntax

```
void avl::SplitPathByPath
(
 const avl::Path& inPath1,
 const avl::Path& inPath2,
 atl::Array<avl::Path>& outPaths
)
```

Parameters

	Name	Type	Default	Description
→	inPath1	const Path&		Path being split
→	inPath2	const Path&		Path used for splitting
←	outPaths	Array<Path>&		Paths arisen from splitting the initial path

SplitPathBySegment

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Splits path in common points of a path and a segment.

Syntax

```
void avl::SplitPathBySegment
(
 const avl::Path& inPath,
 const avl::Segment2D& inSegment,
 atl::Array<avl::Path>& outPaths
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Path being split
→	inSegment	const Segment2D&		Segment used for splitting
←	outPaths	Array<Path>&		Paths arisen from splitting the initial path

40. Geometry 2D Basics

Table of content:

CreateArc

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates an arc from an aligned point, radius, and angle range.

Syntax

```
void avl::CreateArc
(
 const avl::Point2D& inPoint,
 avl::Anchor2D::Type inPointAnchor,
 float inRadius,
 float inStartAngle,
 float inSweepAngle,
 avl::Arc2D& outArc
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoint	const Point2D&			
→	inPointAnchor	Anchor2D::Type		MiddleCenter	Alignment of the point relatively to the box of the circle
→	inRadius	float	0.0 - ∞		Circle radius
→	inStartAngle	float			Direction at which the arc begins
→	inSweepAngle	float			Length of the arc (may be negative)
←	outArc	Arc2D&			

CreateCircle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a circle from an aligned point and radius.

Syntax

```
void avl::CreateCircle
(
 const avl::Point2D& inPoint,
 avl::Anchor2D::Type inPointAnchor,
 float inRadius,
 avl::Circle2D& outCircle
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoint	const Point2D&			
→	inPointAnchor	Anchor2D::Type		MiddleCenter	Alignment of the point relatively to the box of the circle
→	inRadius	float	0.0 - ∞		Circle radius
←	outCircle	Circle2D&			

CreatePointGrid

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a grid of points.

Syntax

```
void avl::CreatePointGrid
(
 const avl::Point2D& inPoint,
 avl::Anchor2D::Type inAnchor,
 const int inRowCount,
 const int inColumnCount,
 const float inRowStep,
 const float inColumnStep,
 atl::Array<avl::Point2D>& outPointGrid
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoint	const Point2D&			A starting point of the created grid
→	inAnchor	Anchor2D::Type		TopLeft	Anchor of the reference point
→	inRowCount	const int	0 - +∞		Number of rows the grid will have
→	inColumnCount	const int	0 - +∞		Number of columns the grid will have
→	inRowStep	const float	0.0 - ∞	1.0f	Distance between consecutive rows of the created grid
→	inColumnStep	const float	0.0 - ∞	1.0f	Distance between consecutive columns of the created grid
←	outPointGrid	Array<Point2D>&			Created point grid

CreateRandomPointArray

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates array of random points inside given box.

Syntax

```
void avl::CreateRandomPointArray
(
 const int inLength,
 const avl::Box& inBox,
 const float inStep,
 atl::Optional<int> inSeed,
 atl::Array< avl::Point2D >& outArray
)
```

Parameters

	Name	Type	Range	Default	Description
→	inLength	const int	1 - +∞	10	Length of output array
→	inBox	const Box&			Bounding box of generated point
→	inStep	const float	0.0001 - ∞	1.0f	Minimal difference between two generated values on each coordinate
→	inSeed	Optional<int>		NIL	Random seed
←	outArray	Array< Point2D >&			

Errors

List of possible exceptions:

Error type	Description
DomainError	Value of inStep is greater than height of box in CreateRandomPointArray.
DomainError	Value of inStep is greater than width of box in CreateRandomPointArray.

CreateRectangle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a rectangle from an aligned point.

Syntax

```
void avl::CreateRectangle
(
 const avl::Point2D& inPoint,
 avl::Anchor2D::Type inPointAnchor,
 float inAngle,
 float inWidth,
 float inHeight,
 avl::Rectangle2D& outRectangle
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoint	const Point2D&			
→	inPointAnchor	Anchor2D::Type		TopLeft	Alignment of the input point relatively to the box of the position
→	inAngle	float			Clock-wise orientation angle
→	inWidth	float	0.0 - ∞		Width of the created rectangle
→	inHeight	float	0.0 - ∞		Height of the created rectangle
←	outRectangle	Rectangle2D&			

CreateRectangularPointGrid

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a grid of points in a described rectangle.

Syntax

```
void avl::CreateRectangularPointGrid
(
 const avl::Rectangle2D& inRectangle,
 const avl::CoordinateSystem2D& inAlignment,
 const int inRowCount,
 const int inColumnCount,
 const float inMarginWidth,
 const float inMarginHeight,
 atl::Array<avl::Point2D>& outPointGrid,
 avl::Rectangle2D& diagAlignedRectangle
)
```

Parameters

	Name	Type	Range	Default	Description
→	inRectangle	const Rectangle2D&			Where points should be inscribed
→	inAlignment	const CoordinateSystem2D&			
→	inRowCount	const int	2 - + ∞		Number of rows the grid will have
→	inColumnCount	const int	2 - + ∞		Number of columns the grid will have
→	inMarginWidth	const float	0.0 - ∞		
→	inMarginHeight	const float	0.0 - ∞		
←	outPointGrid	Array<Point2D>&			Output points
🔍	diagAlignedRectangle	Rectangle2D&			

CreateRing

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a ring from an aligned point, inner and outer radius, and angle range.

Syntax

```
void avl::CreateRing
(
 const avl::Point2D& inPoint,
 avl::Anchor2D::Type inPointAnchor,
 float inInnerRadius,
 float inOuterRadius,
 avl::Ring2D& outRing
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoint	const Point2D&			
→	inPointAnchor	Anchor2D::Type		MiddleCenter	Alignment of the point relatively to the box of the ring outer circle
→	inInnerRadius	float	0.0 - ∞		Inner ring radius
→	inOuterRadius	float	0.0 - ∞		Outer ring radius
←	outRing	Ring2D&			

CreateSegment

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Produces a segment of given parameters.

Syntax

```
void avl::CreateSegment
(
 const avl::Point2D& inPoint,
 float inPointAnchor,
 float inDirection,
 float inLength,
 avl::Segment2D& outSegment
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoint	const Point2D&			A point on the created segment
→	inPointAnchor	float	0.0 - 1.0		Anchor point, 0 means the beginning, 1 means the end of the segment
→	inDirection	float			Desired direction of the segment in degrees
→	inLength	float			Desired length of the segment in pixels
←	outSegment	Segment2D&			The resulting segment

CreateVector

Header: AVL.h
Namespace: avl
Module: FoundationLite

Produces a vector of given length and direction.

Syntax

```
void avl::CreateVector
(
 float inLength,
 float inDirection,
 avl::Vector2D& outVector
)
```

Parameters

	Name	Type	Default	Description
→	inLength	float		Desired length of the vector in pixels
→	inDirection	float		Desired direction of the vector in degrees
←	outVector	Vector2D&		The resulting vector

Also in AVL Lite

RandomPoint

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates random point inside given box.

Syntax

```
void avl::RandomPoint
(
 RandomState& ioState,
 const avl::Box& inBox,
 atl::Optional<int> inSeed,
 avl::Point2D& outPoint
)
```

Parameters

	Name	Type	Default	Description
↔	ioState	RandomState&		Object used to maintain state of the function.
→	inBox	const Box&		Bounding box of generated point
→	inSeed	Optional<int>	NIL	Random seed
←	outPoint	Point2D&		

Also in AVL Lite

WKT_LoadLineString

Header: AVL.h
Namespace: avl
Module: FoundationLite

Loads path from WKT text format.

Syntax

```
void avl::WKT_LoadLineString
(
 const atl::String& inText,
 avl::Path& outPath
)
```

Parameters

	Name	Type	Default	Description
→	inText	const String&		
←	outPath	Path&		Output path

WKT_LoadMultiLineString

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Loads multiple paths from WKT text format.

Syntax

```
void avl::WKT_LoadMultiLineString
(
 const atl::String& inText,
 atl::Array<avl::Path>& outPaths
)
```

Parameters

	Name	Type	Default	Description
→	inText	const String&		
←	outPaths	Array<Path>&		

WKT_LoadMultiPoint

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Loads multiple points from WKT text format.

Syntax

```
void avl::WKT_LoadMultiPoint
(
 const atl::String& inText,
 atl::Array<avl::Point2D>& outPoints
)
```

Parameters

	Name	Type	Default	Description
→	inText	const String&		
←	outPoints	Array<Point2D>&		

WKT_LoadMultiPolygon

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Loads multiple polygons from WKT text format.

Syntax

```
void avl::WKT_LoadMultiPolygon
(
 const atl::String& inText,
 atl::Array<avl::Path>& outOuterPolygons,
 atl::Array<atl::Array<avl::Path>>& outInnerPolygons
)
```

Parameters

	Name	Type	Default	Description
→	inText	const String&		
←	outOuterPolygons	Array<Path>&		
←	outInnerPolygons	Array<Array<Path>>&		

WKT_LoadMultiPolygon_AsRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Loads multiple polygons from WKT text format as region.

Syntax

```
void avl::WKT_LoadMultiPolygon_AsRegion
(
 const atl::String& inText,
 int inFrameWidth,
 int inFrameHeight,
 atl::Array<avl::Region>& outRegions
)
```

Parameters

	Name	Type	Range	Default	Description
→	inText	const String&			
→	inFrameWidth	int	0 - 65535		
→	inFrameHeight	int	0 - 65535		
←	outRegions	Array<Region>&			

WKT_LoadPoint

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Loads point from WKT text format.

Syntax

```
void avl::WKT_LoadPoint
(
 const atl::String& inText,
 avl::Point2D& outPoint
)
```

Parameters

	Name	Type	Default	Description
→	inText	const String&		
←	outPoint	Point2D&		

WKT_LoadPolygon

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Loads polygon from WKT text format.

Syntax

```
void avl::WKT_LoadPolygon
(
 const atl::String& inText,
 avl::Path& outOuterPolygon,
 atl::Array<avl::Path>& outInnerPolygons
)
```

Parameters

	Name	Type	Default	Description
→	inText	const String&		
←	outOuterPolygon	Path&		
←	outInnerPolygons	Array<Path>&		

WKT_LoadPolygon_AsRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Loads polygon from WKT text format as region.

Syntax

```
void avl::WKT_LoadPolygon_AsRegion
(
 const atl::String& inText,
 int inFrameWidth,
 int inFrameHeight,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inText	const String&			
→	inFrameWidth	int	0 - 65535		
→	inFrameHeight	int	0 - 65535		
←	outRegion	Region&			Output region

41. Box

Table of content:

BoxCenter

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns center of a box.

Syntax

```
void avl::BoxCenter
(
 const avl::Box& inBox,
 avl::Point2D& outCenter
)
```

Parameters

	Name	Type	Default	Description
→	inBox	const Box&		
←	outCenter	Point2D&		

BoxCharacteristicPoint

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns a characteristic point (e.g. the top-left) of a box.

Syntax

```
void avl::BoxCharacteristicPoint
(
 const avl::Box& inBox,
 const avl::Anchor2D::Type inPointAnchor,
 avl::Point2D& outPoint
)
```

Parameters

	Name	Type	Default	Description
→	inBox	const Box&		
→	inPointAnchor	const Anchor2D::Type	TopLeft	Selecting one of the 9 characteristic points
←	outPoint	Point2D&		

BoxCharacteristicPoints

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the characteristic points of the box.

Syntax

```
void avl::BoxCharacteristicPoints
(
 const avl::Box& inBox,
 atl::Optional<avl::Point2D&> outTopLeft = atl::NIL,
 atl::Optional<avl::Point2D&> outTopCenter = atl::NIL,
 atl::Optional<avl::Point2D&> outTopRight = atl::NIL,
 atl::Optional<avl::Point2D&> outMiddleLeft = atl::NIL,
 atl::Optional<avl::Point2D&> outMiddleCenter = atl::NIL,
 atl::Optional<avl::Point2D&> outMiddleRight = atl::NIL,
 atl::Optional<avl::Point2D&> outBottomLeft = atl::NIL,
 atl::Optional<avl::Point2D&> outBottomCenter = atl::NIL,
 atl::Optional<avl::Point2D&> outBottomRight = atl::NIL,
 atl::Optional<atl::Array<avl::Point2D>&> outCorners = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inBox	const Box&		
←	outTopLeft	Optional<Point2D&>	NIL	
←	outTopCenter	Optional<Point2D&>	NIL	
←	outTopRight	Optional<Point2D&>	NIL	
←	outMiddleLeft	Optional<Point2D&>	NIL	
←	outMiddleCenter	Optional<Point2D&>	NIL	
←	outMiddleRight	Optional<Point2D&>	NIL	
←	outBottomLeft	Optional<Point2D&>	NIL	
←	outBottomCenter	Optional<Point2D&>	NIL	
←	outBottomRight	Optional<Point2D&>	NIL	
←	outCorners	Optional<Array<Point2D>&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outTopLeft**, **outTopCenter**, **outTopRight**, **outMiddleLeft**, **outMiddleCenter**, **outMiddleRight**, **outBottomLeft**, **outBottomCenter**, **outBottomRight**, **outCorners**.

Read more about Optional Outputs.

BoxesBoundingBox

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the bounding box of given boxes.

Syntax

```
void avl::BoxesBoundingBox
(
 const atl::Array<avl::Box>& inBoxes,
 avl::Box& outBoundingBox
)
```

Parameters

	Name	Type	Default	Description
→	inBoxes	const Array<Box>&		
←	outBoundingBox	Box&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array on input in BoxesBoundingBox.

BoxesBoundingBox_OrNil

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the bounding box of given boxes; returns NIL if the array is empty.

Syntax

```
void avl::BoxesBoundingBox_OrNil
(
 const atl::Array<avl::Box>& inBoxes,
 atl::Conditional<avl::Box>& outBoundingBox
)
```

Parameters

	Name	Type	Default	Description
→	inBoxes	const Array<Box>&		
←	outBoundingBox	Conditional<Box>&		

BoxIntersection

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the common part of two boxes.

Syntax

```
void avl::BoxIntersection
(
 const avl::Box& inBox1,
 const avl::Box& inBox2,
 avl::Box& outBox
)
```

Parameters

	Name	Type	Default	Description
→	inBox1	const Box&		
→	inBox2	const Box&		
←	outBox	Box&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inBox1** and **outBox**, **inBox2** and **outBox**

Read more about In-place Computation.

BoxToBoxDistance

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes minimal distance between one of the points of the first box with one of the points of the second box.

Syntax

```
void avl::BoxToBoxDistance
(
 const avl::Box& inBox1,
 const avl::Box& inBox2,
 float inResolution,
 float& outDistance,
 atl::Optional<avl::Segment2D&> outConnectingSegment = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inBox1	const Box&			
→	inBox2	const Box&			
→	inResolution	float	0.0 - ∞	1.0f	
←	outDistance	float&			
←	outConnectingSegment	Optional<Segment2D&>		NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outConnectingSegment**.

Read more about Optional Outputs.

CreateBox

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a box.

Syntax

```
void avl::CreateBox
(
 const avl::Location& inLocation,
 avl::Anchor2D::Type inLocationAnchor,
 int inWidth,
 int inHeight,
 avl::Box& outBox
)
```

Parameters

	Name	Type	Range	Default	Description
→	inLocation	const Location&			
→	inLocationAnchor	Anchor2D::Type			
→	inWidth	int	0 - ∞		
→	inHeight	int	0 - ∞		
←	outBox	Box&			

DilateBox

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Performs a morphological dilation on a box using box kernel.

Syntax

```
void avl::DilateBox
(
 const avl::Box& inBox,
 const int inRadiusX,
 atl::Optional<int> inRadiusY,
 avl::Box& outBox
)
```

Parameters

	Name	Type	Range	Default	Description
→	inBox	const Box&			Input box
→	inRadiusX	const int	0 - ∞	1	Horizontal radius of the box kernel
→	inRadiusY	Optional<int>	0 - ∞	NIL	Vertical radius of the box kernel
←	outBox	Box&			Dilated box

In-place Processing

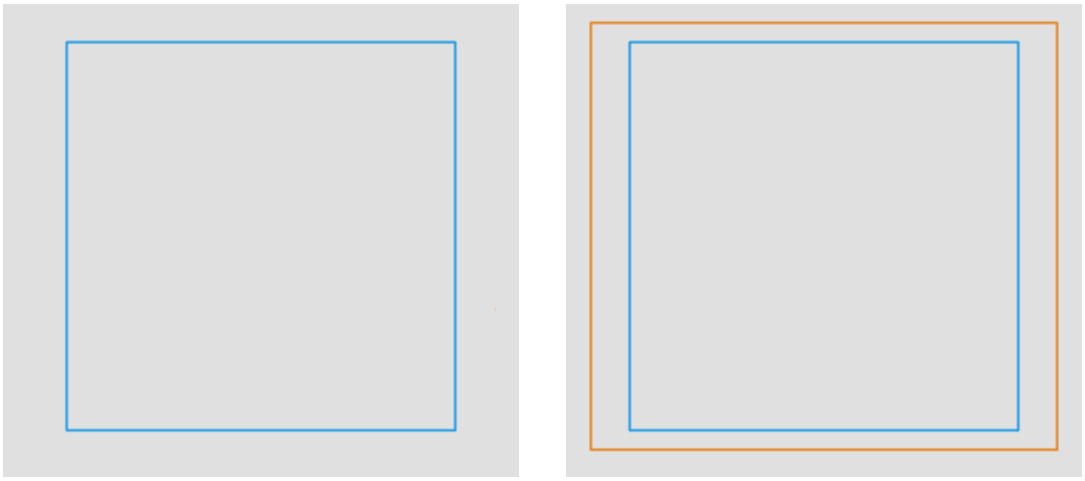
This function supports in-place data processing - you can pass the same reference to **inBox** and **outBox**

Read more about In-place Computation.

Description

Performs a morphological dilation on a box using box kernel.

Examples



***DilateBox** performed with kernel size 2x1.*

See Also

- ResizeBox_Relative – Resizes a box to relatively defined dimensions.
- ResizeRegion – Enlarges or shrinks a region to new dimensions.

ErodeBox

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Performs a morphological erosion on a box using box kernel.

Syntax

```
void avl::ErodeBox
(
 const avl::Box& inBox,
 const int inRadiusX,
 atl::Optional<int> inRadiusY,
 avl::Box& outBox
)
```

Parameters

	Name	Type	Range	Default	Description
→	inBox	const Box&			Input box
→	inRadiusX	const int	0 - ∞	1	Horizontal radius of the box kernel
→	inRadiusY	Optional<int>	0 - ∞	NIL	Vertical radius of the box kernel
←	outBox	Box&			Eroded box

In-place Processing

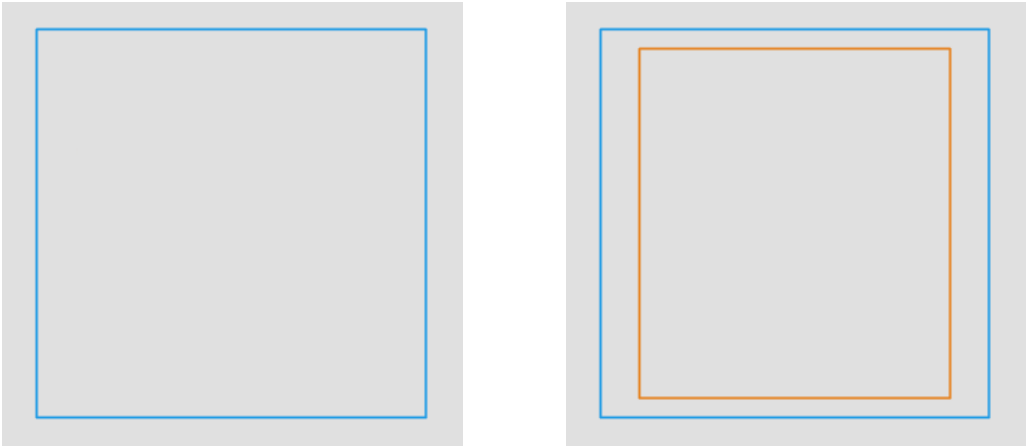
This function supports in-place data processing - you can pass the same reference to **inBox** and **outBox**

Read more about In-place Computation.

Description

Performs a morphological erosion on a box using box kernel.

Examples



ErodeBox *performed with kernel size 2x1.*

See Also

- ResizeBox_Relative – Resizes a box to relatively defined dimensions.
- ResizeRegion – Enlarges or shrinks a region to new dimensions.

RemoveEmptyBoxes

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

The input boxes which have both dimensions different from zero are copied to the output.

Applications: Secures against domain errors caused by empty boxes.

Syntax

```
void avl::RemoveEmptyBoxes
(
 const atl::Array<avl::Box>& inBoxes,
 atl::Array<avl::Box>& outNotEmptyBoxes,
 atl::Array<bool>& outIsNotEmpty
)
```

Parameters

	Name	Type	Default	Description
→	inBoxes	const Array<Box>&		
←	outNotEmptyBoxes	Array<Box>&		
←	outIsNotEmpty	Array<bool>&		

ResizeBox

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Changes the dimensions of a box.

Syntax

```
void avl::ResizeBox
(
 const avl::Box& inBox,
 avl::Anchor2D::Type inAnchor,
 atl::Optional<int> inNewWidth,
 atl::Optional<int> inNewHeight,
 avl::Box& outBox
)
```

Parameters

	Name	Type	Range	Default	Description
→	inBox	const Box&			Input box
→	inAnchor	Anchor2D::Type		TopLeft	Point of the box which position will not change
→	inNewWidth	Optional<int>	0 - ∞	NIL	Target width of the box
→	inNewHeight	Optional<int>	0 - ∞	NIL	Target height of the box
←	outBox	Box&			Resized box

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inBox** and **outBox**

Read more about In-place Computation.

Description

The operation changes dimensions of input Box to (**inNewWidth**, **inNewHeight**).

See Also

- ResizeBox_Relative – Resizes a box to relatively defined dimensions.
- ResizeRegion – Enlarges or shrinks a region to new dimensions.

ResizeBox_Delta

Header: AVL.h
Namespace: avl
Module: FoundationLite

Changes the dimensions of a box by adding some values.

Syntax

```
void avl::ResizeBox_Delta
(
 const avl::Box& inBox,
 avl::Anchor2D::Type inAnchor,
 int inWidthDelta,
 int inHeightDelta,
 avl::Box& outBox
)
```

Parameters

	Name	Type	Default	Description
→	inBox	const Box&		Input box
→	inAnchor	Anchor2D::Type	TopLeft	Point of the box which position will not change
→	inWidthDelta	int	0	Value added to width of the box
→	inHeightDelta	int	0	Value added to height of the box
←	outBox	Box&		Resized box

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inBox** and **outBox**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Negative box dimensions in ResizeBox_Delta.

ResizeBox_Relative

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Resizes a box to relatively defined dimensions.

Syntax

```
void avl::ResizeBox_Relative
(
 const avl::Box& inBox,
 avl::Anchor2D::Type inAnchor,
 float inHorizontalScale,
 float inVerticalScale,
 avl::Box& outBox
)
```

Parameters

	Name	Type	Range	Default	Description
→	inBox	const Box&			Input box
→	inAnchor	Anchor2D::Type		TopLeft	Point of the box which position will not change
→	inHorizontalScale	float	0.0 - ∞	1.0f	Scale factor of the horizontal resize
→	inVerticalScale	float	0.0 - ∞	1.0f	Scale factor of the vertical resize
←	outBox	Box&			Resized box

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inBox** and **outBox**

Read more about In-place Computation.

Description

The operation resizes an input box multiplying its width by **inHorizontalScale** and height by **inVerticalScale**.

See Also

- ResizeBox – Changes the dimensions of a box.
- ResizeRegion – Enlarges or shrinks a region to new dimensions.

SkipEmptyBox

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

If the input box has both dimensions different from zero, then it is copied to the output; otherwise Nil is returned.

Applications: Secures against domain errors caused by empty boxes.

Syntax

```
void avl::SkipEmptyBox
(
 const avl::Box& inBox,
 atl::Conditional<avl::Box>& outNotEmptyBox,
 bool& outIsNotEmpty
)
```

Parameters

	Name	Type	Default	Description
→	inBox	const Box&		
←	outNotEmptyBox	Conditional<Box>&		
←	outIsNotEmpty	bool&		

SplitBox

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Splits a box into two along a direction.

Syntax

```
void avl::SplitBox
(
 const avl::Box& inBox,
 const avl::SplitDirection::Type inSplitDirection,
 avl::Box& outBox1,
 avl::Box& outBox2
)
```

Parameters

	Name	Type	Default	Description
➡	inBox	const Box&		
➡	inSplitDirection	const SplitDirection::Type		
⬅	outBox1	Box&		
⬅	outBox2	Box&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inBox** and **outBox1**, **inBox** and **outBox2**

Read more about In-place Computation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Unknown split direction in SplitBox.

TestBoxEmpty

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Tests whether the box is empty.

Syntax

```
void avl::TestBoxEmpty
(
 const avl::Box& inBox,
 bool& outIsEmpty
)
```

Parameters

	Name	Type	Default	Description
➡	inBox	const Box&		
⬅	outIsEmpty	bool&		

TestBoxEqualTo

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Tests whether given boxes are equal.

Syntax

```
void avl::TestBoxEqualTo
(
 const avl::Box& inBox,
 const avl::Box& inReferenceBox,
 bool& outIsEqual
)
```

Parameters

	Name	Type	Default	Description
→	inBox	const Box&		
→	inReferenceBox	const Box&		
←	outIsEqual	bool&		

TestBoxInBox

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Tests whether a box is contained in another one.

Syntax

```
void avl::TestBoxInBox
(
 const avl::Box& inSubBox,
 const avl::Box& inBox,
 bool& outIsContained
)
```

Parameters

	Name	Type	Default	Description
→	inSubBox	const Box&		
→	inBox	const Box&		
←	outIsContained	bool&		

TestBoxIntersectsWith

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Tests whether two boxes have non-empty intersection.

Syntax

```
void avl::TestBoxIntersectsWith
(
 const avl::Box& inBox,
 const avl::Box& inReferenceBox,
 bool& outBoxesIntersect
)
```

Parameters

	Name	Type	Default	Description
→	inBox	const Box&		
→	inReferenceBox	const Box&		
←	outBoxesIntersect	bool&		

TestBoxNotEmpty

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Tests whether the box is not empty.

Syntax

```
void avl::TestBoxNotEmpty
(
 const avl::Box& inBox,
 bool& outIsNotEmpty
)
```

Parameters

	Name	Type	Default	Description
→	inBox	const Box&		
←	outIsNotEmpty	bool&		

TranslateBox

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Shifts a box by a discreet vector.

Syntax

```
void avl::TranslateBox
(
 const avl::Box& inBox,
 int inDeltaX,
 int inDeltaY,
 bool inInverse,
 avl::Box& outBox
)
```

Parameters

	Name	Type	Default	Description
→	inBox	const Box&		Input box
→	inDeltaX	int		Shift along the x axis
→	inDeltaY	int		Shift along the y axis
→	inInverse	bool		Switches to inverse operation
←	outBox	Box&		Shifted box

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inBox** and **outBox**

Read more about In-place Computation.

42. Geometry 3D Basics

Table of content:

CreateBox3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Creates a box in 3D.

Syntax

```
void avl::CreateBox3D
(
 const avl::Point3D& inPoint3D,
 const avl::Anchor3D& inPoint3DAnchor3D,
 float inXLength,
 float inYLength,
 float inZLength,
 avl::Box3D& outBox3D
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoint3D	const Point3D&			
→	inPoint3DAnchor3D	const Anchor3D&			
→	inXLength	float	0.0 - ∞		
→	inYLength	float	0.0 - ∞		
→	inZLength	float	0.0 - ∞		
←	outBox3D	Box3D&			

Errors

List of possible exceptions:

Error type	Description
DomainError	Invalid Anchor3D in CreateBox3D.

CreateSegment3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Produces a segment in 3D of given parameters.

Syntax

```
void avl::CreateSegment3D
(
 const avl::Point3D& inPoint3D,
 float inPointAnchor,
 const avl::Vector3D& inDirectionVector,
 float inLength,
 avl::Segment3D& outSegment3D
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoint3D	const Point3D&			A point on the created segment
→	inPointAnchor	float	0.0 - 1.0		Anchor point, 0 means the beginning, 1 means the end of the segment
→	inDirectionVector	const Vector3D&			Desired direction of the segment
→	inLength	float			Desired length of the segment in pixels
←	outSegment3D	Segment3D&			The resulting segment

LoadPoint3DArrayFromTextFile

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Loads arrays of 3D points from text file.

Syntax

```
void avl::LoadPoint3DArrayFromTextFile
(
 const atl::File& inFile,
 const avl::Grid3DFileFormat::Type inFileFormat,
 const atl::String& inSplitters,
 const int inLineSkip,
 atl::Array<avl::Point3D>& outPoints,
 atl::String& diagTextLine,
 avl::Point3D& diagExtractedValues
)
```

Parameters

	Name	Type	Default	Description
→	inFile	const File&		
→	inFileFormat	const Grid3DFileFormat::Type		
→	inSplitters	const String&		Characters ignored between numbers, space is always splitter and ignored in file sequence
→	inLineSkip	const int	0	Number of first lines to skip
←	outPoints	Array<Point3D>&		
🔍	diagTextLine	String&		First valid parse line
🔍	diagExtractedValues	Point3D&		Example of expected line

Errors

List of possible exceptions:

Error type	Description
DomainError	Splitters should do not contains any spaces in Load3DPointArrayFromTextFile.
IoError	Unable to open file in Load3DPointArrayFromTextFile.

RandomPoint3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Creates random 3D point inside given box.

Syntax

```
void avl::RandomPoint3D
(
 RandomState& ioState,
 const avl::Box3D& inBox3D,
 atl::Optional<int> inSeed,
 avl::Point3D& outPoint3D
)
```

Parameters

	Name	Type	Default	Description
↔	ioState	RandomState&		Object used to maintain state of the function.
→	inBox3D	const Box3D&		Bounding box of generated point
→	inSeed	Optional<int>	NIL	Random seed
←	outPoint3D	Point3D&		

SkipEmptyBox3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

If the input box in 3D has all dimensions different from zero, then it is copied to the output; otherwise Nil is returned.

Applications: Secures against domain errors caused by empty boxes.

Syntax

```
void avl::SkipEmptyBox3D
(
 const avl::Box3D& inBox3D,
 atl::Conditional<avl::Box3D>& outNotEmptyBox3D,
 bool& outIsNotEmpty
)
```

Parameters

	Name	Type	Default	Description
→	inBox3D	const Box3D&		
←	outNotEmptyBox3D	Conditional<Box3D>&		
←	outIsNotEmpty	bool&		

TestBox3DEmpty

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Tests whether the box in 3D is empty.

Syntax

```
void avl::TestBox3DEmpty
(
 const avl::Box3D& inBox3D,
 bool& outIsEmpty
)
```

Parameters

	Name	Type	Default	Description
→	inBox3D	const Box3D&		
←	outIsEmpty	bool&		

TestBox3DInBox3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Tests whether a box in 3D is contained in another one.

Syntax

```
void avl::TestBox3DInBox3D
(
 const avl::Box3D& inSubBox3D,
 const avl::Box3D& inBox3D,
 bool& outIsContained
)
```

Parameters

	Name	Type	Default	Description
→	inSubBox3D	const Box3D&		
→	inBox3D	const Box3D&		
←	outIsContained	bool&		

TestBox3DIntersectsWith

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Tests whether two boxes in 3D have non-empty intersection.

Syntax

```
void avl::TestBox3DIntersectsWith
(
 const avl::Box3D& inBox3D,
 const avl::Box3D& inReferenceBox3D,
 bool& outBoxes3DIntersect
)
```

Parameters

	Name	Type	Default	Description
→	inBox3D	const Box3D&		
→	inReferenceBox3D	const Box3D&		
←	outBoxes3DIntersect	bool&		

TestBox3DNotEmpty

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Tests whether the box in 3D is not empty.

Syntax

```
void avl::TestBox3DNotEmpty
(
 const avl::Box3D& inBox3D,
 bool& outIsNotEmpty
)
```

Parameters

	Name	Type	Default	Description
→	inBox3D	const Box3D&		
←	outIsNotEmpty	bool&		

43. Histogram Basics

Table of content:

ConvertToCumulativeHistogram

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the cumulative histogram of input histogram.

Syntax

```
void avl::ConvertToCumulativeHistogram
(
 const avl::Histogram& inHistogram,
 avl::Histogram& outHistogram
)
```

Parameters

	Name	Type	Default	Description
→	inHistogram	const Histogram&		Input histogram
←	outHistogram	Histogram&		Output histogram

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inHistogram** and **outHistogram**

Read more about In-place Computation.

CreateHistogram

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Creates the histogram of the array of real numbers.

Syntax

```
void avl::CreateHistogram
(
 const atl::Array<float>& inArray,
 atl::Optional<const atl::Array<double>&> inWeights,
 atl::Optional<float> inDomainStart,
 const float inBinSize,
 atl::Optional<int> inBinCount,
 const bool inCyclic,
 avl::Histogram& outHistogram
)
```

Parameters

	Name	Type	Range	Default	Description
→	inArray	const Array<float>&			Array from which histogram will be generated
→	inWeights	Optional<const Array<double>&>		NIL	Weights corresponding to the elements of 'inArray'
→	inDomainStart	Optional<float>		NIL	Input domain begin
→	inBinSize	const float	0.0001 - ∞	1.0f	Input bin size
→	inBinCount	Optional<int>	0 - + ∞	NIL	Input domain length
→	inCyclic	const bool		False	Determines if input data is cyclic
←	outHistogram	Histogram&			Output histogram

Description

The operation creates a histogram of **inArray** values.

Errors

List of possible exceptions:

Error type	Description
DomainError	Inconsistent array lengths at inArray and inWeights in CreateHistogram.
DomainError	Input bin width is incorrect in CreateHistogram.
DomainError	Input boundaries are incorrect in CreateHistogram.
DomainError	Negative weight encountered in CreateHistogram.
DomainError	Output histogram size too large in CreateHistogram.

See Also

- ImageHistogram – Computes the histogram of the image pixel values.

CreateUniformHistogram

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Creates a uniform histogram of desired parameters with common value of all bins.

Syntax

```
void avl::CreateUniformHistogram
(
 double inValue,
 float inDomainStart,
 float inBinSize,
 int inBinCount,
 avl::Histogram& outHistogram
)
```

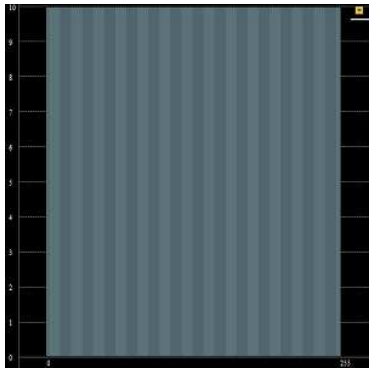
Parameters

	Name	Type	Range	Default	Description
→	inValue	double			Common value for all bins of the constructed histogram
→	inDomainStart	float		0.0f	First value of the domain represented by the histogram
→	inBinSize	float	0.0 - ∞	1.0f	Length of the domain section represented by each bin
→	inBinCount	int	1 - + ∞	1	Length of the domain represented by the histogram
←	outHistogram	Histogram&			The resulting histogram

Description

The operation creates a histogram composed from equally valued bins representing the given domain.

Examples



CreateUniformHistogram *run with* ***inValue*** *= 10.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Non-positive bin width in CreateUniformHistogram.

See Also

- ImageHistogram – Computes the histogram of the image pixel values.
- MakeHistogram – Creates a histogram out of an array of bin values.

GetHistogramBin

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Returns the value of a single histogram bin.

Syntax

```
void avl::GetHistogramBin
(
 const avl::Histogram& inHistogram,
 int inIndex,
 const bool inCyclic,
 const bool inInverse,
 double& outValue,
 atl::Optional<float&> outBinStart = atl::NIL,
 atl::Optional<float&> outBinEnd = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inHistogram	const Histogram&		Input histogram
→	inIndex	int		Input bin index
→	inCyclic	const bool	False	Whether to wrap the index around or not
→	inInverse	const bool		Reversed order of bins
←	outValue	double&		Output value of the bin
←	outBinStart	Optional<float&>	NIL	Lower limit of the bin
←	outBinEnd	Optional<float&>	NIL	Upper limit of the bin

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outBinStart**, **outBinEnd**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect index on input in GetHistogramBin.

GetHistogramCorrespondingBin

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Selects a bin that the given value in the histogram domain falls into and returns the value of this bin.

Syntax

```
void avl::GetHistogramCorrespondingBin
(
 const avl::Histogram& inHistogram,
 float inBinSelectionValue,
 const bool inCyclic,
 double& outValue
)
```

Parameters

	Name	Type	Default	Description
→	inHistogram	const Histogram&		Input histogram
→	inBinSelectionValue	float		Input value in the histogram domain that will be used to select the bin
→	inCyclic	const bool	False	Whether to wrap the index around or not
←	outValue	double&		Output value of the bin

Errors

List of possible exceptions:

Error type	Description
DomainError	inBinSelectionValue exceeds the histogram domain in GetHistogramCorrespondingBin.

HistogramIndices

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Returns an array of histogram elements' indices.

Syntax

```
void avl::HistogramIndices
(
 const avl::Histogram& inHistogram,
 atl::Array<int>& outIndices
)
```

Parameters

	Name	Type	Default	Description
→	inHistogram	const Histogram&		Input histogram
←	outIndices	Array<int>&		

SetHistogramBin

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Sets the value of a single histogram bin.

Syntax

```
void avl::SetHistogramBin
(
 avl::Histogram& ioHistogram,
 int inIndex,
 const bool inCyclic,
 const bool inInverse,
 double inNewValue
)
```

Parameters

	Name	Type	Default	Description
↔	ioHistogram	Histogram&		
→	inIndex	int		Input bin index
→	inCyclic	const bool	False	Whether to wrap the index around or not
→	inInverse	const bool		Reversed order of bins
→	inNewValue	double		Input new value to be set

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect index on input in SetHistogramBin.

SetHistogramCorrespondingBin

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Selects a bin that the given value in the histogram domain falls into and sets the value of this bin.

Syntax

```
void avl::SetHistogramCorrespondingBin
(
 avl::Histogram& ioHistogram,
 float inBinSelectionValue,
 const bool inCyclic,
 double inNewValue
)
```

Parameters

	Name	Type	Default	Description
	ioHistogram	Histogram&		
	inBinSelectionValue	float		Input value in the histogram domain that will be used to select the bin
	inCyclic	const bool	False	Whether to wrap the index around or not
	inNewValue	double		Input new value to be set

Errors

List of possible exceptions:

Error type	Description
DomainError	inBinSelectionValue exceeds the histogram domain in SetHistogramCorrespondingBin.

SkipEmptyDataHistogram

Header: AVL.h
Namespace: avl
Module: FoundationBasic

If the input histogram contains any non-zero bin, then the histogram is copied to the output; otherwise Nil is returned.

Applications: Secures against domain errors caused by histograms representing empty data sets, e.g. just before the HistogramDataAverage filter is to be invoked.

Syntax

```
void avl::SkipEmptyDataHistogram
(
 const avl::Histogram& inHistogram,
 atl::Conditional<avl::Histogram>& outNotEmptyDataHistogram,
 bool& outIsNotEmpty
)
```

Parameters

	Name	Type	Default	Description
	inHistogram	const Histogram&		Input histogram
	outNotEmptyDataHistogram	Conditional<Histogram>&		
	outIsNotEmpty	bool&		

SkipEmptyHistogram

Header: AVL.h
Namespace: avl
Module: FoundationBasic

If the input histogram contains at least one bin, then it is copied to the output; otherwise Nil is returned.

Applications: Secures against domain errors caused by empty histograms, e.g. just before the HistogramAverage filter is to be invoked.

Syntax

```
void avl::SkipEmptyHistogram
(
 const avl::Histogram& inHistogram,
 atl::Conditional<avl::Histogram>& outNotEmptyHistogram,
 bool& outIsNotEmpty
)
```

Parameters

	Name	Type	Default	Description
→	inHistogram	const Histogram&		Input histogram
←	outNotEmptyHistogram	Conditional<Histogram>&		The histogram, if it is not empty
←	outIsNotEmpty	bool&		Indication if the input histogram is not empty

44. Image Basics

Table of content:

ClearImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Sets image pixels in ROI to the specified value.

Syntax

```
void avl::ClearImage
(
 avl::Image& ioImage,
 atl::Optional<const avl::Region&> inRegion,
 const avl::Pixel& inValue
)
```

Parameters

	Name	Type	Default	Description
	ioImage	Image&		
	inRegion	Optional<const Region&>	NIL	Input region
	inValue	const Pixel&		The new background color

See Also

- DrawRegion – Draws a region on an image.
- DrawRegions_SingleColor – Draws regions on an image with a single color.
- FillImage – Fills the input image with one color.

CopyImageData

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates an image copy with the same size and pixel format.

Syntax

```
void avl::CopyImageData
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
	inImage	const Image&		Input image
	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
	outImage	Image&		Output image

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in CopyImageData.

CopyPixels

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Copies pixels from one image to another.

Syntax

```
void avl::CopyPixels
(
 avl::Image& ioImage,
 const avl::Region& inRoi,
 const avl::Image& inSourceImage
)
```

Parameters

	Name	Type	Default	Description
↔	ioImage	Image&		
→	inRoi	const Region&		Range of pixels to be processed
→	inSourceImage	const Image&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Different image formats in CopyPixels.
DomainError	Different image sizes in CopyPixels.

CreateImageFromPoint3DGrid

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Creates a depth image from the Z-values of the input point grid.

Applications: Allows for performing 2D operations on 3D data.

Syntax

```
void avl::CreateImageFromPoint3DGrid
(
 const avl::Point3DGrid& inPoint3DGrid,
 const float inMissingPointValue,
 const float inPixelOffset,
 const float inPixelScale,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoint3DGrid	const Point3DGrid&			
→	inMissingPointValue	const float			Value assigned to pixels where point is undefined
→	inPixelOffset	const float		0.0f	Value added to every pixel
→	inPixelScale	const float	-∞ -∞	1.0f	Value every pixel is multiplied by
←	outImage	Image&			Output image

CreateImageFromSurface

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Creates a depth image from the Z-values of the input surface.

Applications: Allows for performing 2D operations on 3D data.

Syntax

```
void avl::CreateImageFromSurface
(
 const avl::Surface& inSurface,
 avl::PlainType::Type inPixelType,
 avl::ResampleSurfaceMode::Type inResampleSurfaceMode,
 atl::Optional<double> inXOffset,
 atl::Optional<double> inYOffset,
 atl::Optional<double> inPixelOffset,
 atl::Optional<double> inPixelScale,
 atl::Optional<float> inMissingPointValue,
 avl::Image& outImage,
 atl::Optional<avl::SurfaceFormat&> outSurfaceFormat = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inPixelType	PlainType::Type		Real	Output image pixel type
→	inResampleSurfaceMode	ResampleSurfaceMode::Type			
→	inXOffset	Optional<double>		NIL	Offset for the X axis of the output image; if set to Nil, surface X offset is chosen
→	inYOffset	Optional<double>		NIL	Offset for the Y axis of the output image; if set to Nil, surface Y offset is chosen
→	inPixelOffset	Optional<double>		NIL	Offset that the input surface values have in the output image; if set to Nil, surface Z offset is chosen
→	inPixelScale	Optional<double>	0.0 - ∞	1.0D	Scale that the input surface values have in the output image; if set to Nil, surface Z scale is chosen
→	inMissingPointValue	Optional<float>		0.0f	Value assigned to pixels where point is undefined
←	outImage	Image&			Input surface depth image
←	outSurfaceFormat	Optional<SurfaceFormat&>		NIL	Format of the surface that the output image represents

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outSurfaceFormat**.

Read more about Optional Outputs.

CreateImageFromSurface_AnyScales

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Creates a depth image from the Z-values of the input surface.

Applications: Allows for performing 2D operations on 3D data.

Syntax

```
void avl::CreateImageFromSurface_AnyScales
(
 const avl::Surface& inSurface,
 avl::PlainType::Type inPixelType,
 atl::Optional<double> inXOffset,
 atl::Optional<double> inXScale,
 atl::Optional<double> inYOffset,
 atl::Optional<double> inYScale,
 atl::Optional<double> inPixelOffset,
 atl::Optional<double> inPixelScale,
 atl::Optional<float> inMissingPointValue,
 avl::Image& outImage,
 atl::Optional<avl::SurfaceFormat&> outSurfaceFormat = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inPixelType	PlainType::Type		Real	Output image pixel type
→	inXOffset	Optional<double>		NIL	Offset for the X axis of the output image; if set to Nil, surface X offset is chosen
→	inXScale	Optional<double>	0.000001 - ∞	NIL	Scale for the X axis of the output image; if set to Nil, surface X scale is chosen
→	inYOffset	Optional<double>		NIL	Offset for the Y axis of the output image; if set to Nil, surface Y offset is chosen
→	inYScale	Optional<double>	0.000001 - ∞	NIL	Scale for the Y axis of the output image; if set to Nil, surface Y scale is chosen
→	inPixelOffset	Optional<double>		NIL	Offset that the input surface values have in the output image; if set to Nil, surface Z offset is chosen
→	inPixelScale	Optional<double>	0.0 - ∞	1.0D	Scale that the input surface values have in the output image; if set to Nil, surface Z scale is chosen
→	inMissingPointValue	Optional<float>		0.0f	Value assigned to pixels where point is undefined
←	outImage	Image&			Input surface depth image
←	outSurfaceFormat	Optional<SurfaceFormat&>		NIL	Format of the surface that the output image represents

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outSurfaceFormat**.

Read more about Optional Outputs.

EmptyImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates an image filled with one color.

Applications: Most typically used to prepare a background for image drawing tools.

Syntax

```
void avl::EmptyImage
(
 const int inWidth,
 const int inHeight,
 const avl::Pixel& inColor,
 const int inChannels,
 const avl::PlainType::Type& inPixelType,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inWidth	const int	1 - 65535	320	Width of the created image
→	inHeight	const int	1 - 65535	240	Height of the created image
→	inColor	const Pixel&			Color of all pixels of the created image
→	inChannels	const int	1 - 4	3	Number of channels
→	inPixelType	const PlainType::Type&		UInt8	
←	outImage	Image&			Output image

Description

The operation produces an empty image of desired dimensions and color.

Examples

***EmptyImage** run with default parameters produces 320x240 black image.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Also in AVL Lite

FillImage

Header: AVL.h
Namespace: avl
Module: FoundationLite

Fills the input image with one color.

Syntax

```
void avl::FillImage
(
 avl::Image& ioImage,
 const avl::Pixel& inColor
)
```

Parameters

	Name	Type	Default	Description
	ioImage	Image&		
	inColor	const Pixel&		Color of all pixels of the created image

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- ClearImage – Sets image pixels in ROI to the specified value.

Also in AVL Lite

GetImageColumn

Header: AVL.h
Namespace: avl
Module: FoundationLite

Extracts an array of pixel values from a single column of an image.

Syntax

```
void avl::GetImageColumn
(
 const avl::Image& inImage,
 atl::Optional<int> inChannelIndex,
 const int inColumnIndex,
 atl::Array<float>& outValues,
 atl::Array<avl::Pixel>& outPixels
)
```

Parameters

	Name	Type	Range	Default	Description
	inImage	const Image&			Input image
	inChannelIndex	Optional<int>	0 - 3	NIL	Selects a channel of the input image
	inColumnIndex	const int	0 - 65535		Selects a column of the input image
	outValues	Array<float>&			Output pixel values of the column
	outPixels	Array<Pixel>&			Output pixels of the column

Errors

List of possible exceptions:

Error type	Description
DomainError	Channel index out of range in GetImageColumn.
DomainError	Column index out of range in GetImageColumn.

GetImageData

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Extracts the image content (raw pixel data) as a binary buffer.

Syntax

```
void avl::GetImageData
(
 const avl::Image& inImage,
 atl::Optional<int> inPitch,
 avl::ByteBuffer& outBuffer
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inPitch	Optional<int>	0 - ∞	NIL	Length, in bytes, between starts of consecutive image rows for the resulting data
←	outBuffer	ByteBuffer&			

Errors

List of possible exceptions:

Error type	Description
DomainError	Pitch value is too small for the source image width and pixel format.
RuntimeError	Invalid image format in GetImageData.
RuntimeError	Resulting byte buffer size is too large.

GetImagePixel

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns a single pixel of an image.

Syntax

```
void avl::GetImagePixel
(
 const avl::Image& inImage,
 const avl::Location& inLocation,
 avl::Pixel& outPixel,
 float& outValue
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inLocation	const Location&		Location of the pixel to be accessed
←	outPixel	Pixel&		Output pixel
←	outValue	float&		Average pixel value

Errors

List of possible exceptions:

Error type	Description
DomainError	Input location exceeds dimensions of an input image in GetImagePixel.

GetImagePixel_Interpolated

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns an interpolated single pixel of an image.

Applications: Sub-pixel sampling of an image.

Syntax

```
void avl::GetImagePixel_Interpolated
(
 const avl::Image& inImage,
 const avl::Point2D& inPoint,
 avl::InterpolationMethod::Type inInterpolation,
 avl::Pixel& outPixel,
 float& outValue
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inPoint	const Point2D&		Point on the input image to be accessed
→	inInterpolation	InterpolationMethod::Type	Bilinear	
←	outPixel	Pixel&		Output pixel
←	outValue	float&		Average pixel value

Errors

List of possible exceptions:

Error type	Description
DomainError	Input point exceeds dimensions of an input image in GetImagePixel_Interpolated.
DomainError	Unknown interpolation method in GetImagePixel_Interpolated.

GetImageRow

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Extracts an array of pixel values from a single row of an image.

Syntax

```
void avl::GetImageRow
(
 const avl::Image& inImage,
 atl::Optional<int> inChannelIndex,
 const int inRowIndex,
 atl::Array<float>& outValues,
 atl::Array<avl::Pixel>& outPixels
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inChannelIndex	Optional<int>	0 - 3	NIL	Selects a channel of the input image
→	inRowIndex	const int	0 - 65535		Selects a row of the input image
←	outValues	Array<float>&			Output pixel values of the row
←	outPixels	Array<Pixel>&			Output pixels of the row

Errors

List of possible exceptions:

Error type	Description
DomainError	Channel index out of range in GetImageRow.
DomainError	Row index out of range in GetImageRow.

Also in AVL Lite

GetMultipleImagePixelValues_Safe

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns an array of pixel values at specified locations; the image range is checked.

Syntax

```
void avl::GetMultipleImagePixelValues_Safe
(
 const avl::Image& inImage,
 const atl::Array<avl::Point2D>& inPoints,
 const avl::InterpolationMethod::Type inInterpolation,
 float inDefault,
 atl::Array<float>& outValues
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inPoints	const Array<Point2D>&		
→	inInterpolation	const InterpolationMethod::Type	Bilinear	
→	inDefault	float	0.0f	
←	outValues	Array<float>&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Unknown interpolation method in GetMultipleImagePixelValues_Safe.

Also in AVL Lite

ImageBox

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns a box corresponding to the dimensions of an image.

Syntax

```
void avl::ImageBox
(
 const avl::Image& inImage,
 avl::Box& outBox
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
←	outBox	Box&		Output box

ImageCharacteristicPoint

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns one of the 9 characteristic points of an image (corners or mid-points).

Syntax

```
void avl::ImageCharacteristicPoint
(
 const avl::Image& inImage,
 const avl::Anchor2D::Type inPointAnchor,
 avl::Point2D& outPoint
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inPointAnchor	const Anchor2D::Type	TopLeft	Selecting one of the 9 characteristic points
←	outPoint	Point2D&		

ImageToMatrix

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a single-channel image to a matrix.

Syntax

```
void avl::ImageToMatrix
(
 const avl::Image& inMonoImage,
 avl::Matrix& outMatrix
)
```

Parameters

	Name	Type	Default	Description
→	inMonoImage	const Image&		
←	outMatrix	Matrix&		

Requirements

For input **inMonoImage** only pixel formats are supported: 1□uint8, 1□int8, 1□uint16, 1□int16, 1□int32, 1□real.

Read more about pixel formats in Image documentation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported inMonoImage pixel format in ImageToMatrix. Supported formats: 1xUInt8, 1xInt8, 1xUInt16, 1xInt16, 1xInt32, 1xReal.

JoinProfilesIntoImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates an image by joining an array of 1D profiles into consecutive image rows.

Applications: Usually used for merging 3D profiles into a depth image.

Syntax

```
void avl::JoinProfilesIntoImage
(
 const atl::Array<avl::Profile>& inProfiles,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inProfiles	const Array<Profile>&		
←	outImage	Image&		Output image

Errors

List of possible exceptions:

Error type	Description
DomainError	Input profiles do not have the same sizes in JoinProfilesIntoImage.
DomainError	Input profiles have different X coordinates in JoinProfilesIntoImage.

JoinProfilesIntoImage_OfSeries

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates an image by joining 1D profiles, which appear in consecutive iterations.

Applications: Usually used for merging 3D profiles into a depth image.

Syntax

```
void avl::JoinProfilesIntoImage_OfSeries
(
 JoinProfilesIntoImage_OfSeriesState& ioState,
 const avl::Profile& inProfile,
 const int inSeriesSize,
 atl::Conditional<avl::Image>& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	JoinProfilesIntoImage_OfSeriesState&			Object used to maintain state of the function.
→	inProfile	const Profile&			Input profile
→	inSeriesSize	const int	1 - ∞		Number of profiles that constitute a single image
←	outImage	Conditional<Image>&			A depth image every inSeriesSize iterations, or Nil in all other iterations

Errors

List of possible exceptions:

Error type	Description
DomainError	Input profiles do not have the same sizes in JoinProfilesIntoImage_OfSeries.

LoadImageObject

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Loads serialized Image object from AVDATA file.

Syntax

```
void avl::LoadImageObject
(
 const atl::File& inFilename,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inFilename	const File&		Name of the source file
←	outImage	Image&		Deserialized output Image

MatrixToImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a matrix to a single-channel real image.

Syntax

```
void avl::MatrixToImage
(
 const avl::Matrix& inMatrix,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inMatrix	const Matrix&		
←	outImage	Image&		Output image

RealignImagePitch

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a new image with a standard pitch alignment.

Applications: Can be used to normalize an image created with a non-standard third-party camera or framegrabber.

Syntax

```
void avl::RealignImagePitch
(
 const avl::Image& inImage,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
←	outImage	Image&		Output image

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty image on input in RealignImagePitch.

SaveImageObject

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Saves serialized Image object as AVDATA file.

Syntax

```
void avl::SaveImageObject
(
 const avl::Image& inImage,
 const atl::File& inFilename
)
```

Parameters

	Name	Type	Default	Description
➡	inImage	const Image&		Image to be serialized
➡	inFilename	const File&		Name of the target file

SetImageColumn

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Sets pixel values in a single entire column of an image.

Applications: Allows for creating images from calculated real values. It is significantly slower than SetImageRow.

Syntax

```
void avl::SetImageColumn
(
 avl::Image& ioImage,
 atl::Optional<int> inChannelIndex,
 const int inColumnIndex,
 const atl::Array<float>& inValues
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioImage	Image&			
➡	inChannelIndex	Optional<int>	0 - 3	NIL	Selects a channel of the input image
➡	inColumnIndex	const int	0 - 65535		Selects a column of the input image
➡	inValues	const Array<float>&			New values for specified column

Errors

List of possible exceptions:

Error type	Description
DomainError	Channel index out of range in SetImageColumn.
DomainError	Column index out of range in SetImageColumn.
DomainError	Size of inValues array and image height are not equal in SetImageColumn.

SetImagePixel

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Sets a pixel of an image to the selected value.

Syntax

```
void avl::SetImagePixel
(
 avl::Image& ioImage,
 const avl::Location& inLocation,
 const avl::Pixel& inPixel
)
```

Parameters

	Name	Type	Default	Description
	ioImage	Image&		
	inLocation	const Location&		Location of the pixel to be set
	inPixel	const Pixel&		New pixel value

Errors

List of possible exceptions:

Error type	Description
DomainError	Input location exceeds dimensions of an input image in SetImagePixel.

SetImagePixels

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Sets multiple pixels of an image to the selected value (all the pixels get the same value).

Syntax

```
void avl::SetImagePixels
(
 avl::Image& ioImage,
 const atl::Array<avl::Location>& inLocations,
 const avl::Pixel& inPixel
)
```

Parameters

	Name	Type	Default	Description
	ioImage	Image&		
	inLocations	const Array<Location>&		Locations of the pixels to be set
	inPixel	const Pixel&		New pixel value

Errors

List of possible exceptions:

Error type	Description
DomainError	Input location exceeds dimensions of an input image in SetImagePixels.

SetImageRow

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Sets pixel values in a single entire row of an image.

Applications: Allows for creating images from calculated real values.

Syntax

```
void avl::SetImageRow
(
 avl::Image& ioImage,
 atl::Optional<int> inChannelIndex,
 const int inRowIndex,
 const atl::Array<float>& inValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioImage	Image&			
	inChannelIndex	Optional<int>	0 - 3	NIL	Selects a channel of the input image
	inRowIndex	const int	0 - 65535		Selects a row of the input image
	inValues	const Array<float>&			New values for specified row

Errors

List of possible exceptions:

Error type	Description
DomainError	Channel index out of range in SetImageRow.
DomainError	Row index out of range in SetImageRow.
DomainError	Size of inValues array and image width are not equal in SetImageRow.

SetMultipleImagePixels

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Sets pixels of an image to the selected values (each pixel gets another value).

Syntax

```
void avl::SetMultipleImagePixels
(
 avl::Image& ioImage,
 const atl::Array<avl::Location>& inLocations,
 const atl::Array<avl::Pixel>& inPixels
)
```

Parameters

	Name	Type	Default	Description
	ioImage	Image&		
	inLocations	const Array<Location>&		Locations of the pixels to be set
	inPixels	const Array<Pixel>&		New pixel values

Errors

List of possible exceptions:

Error type	Description
DomainError	Input location exceeds dimensions of an input image in SetMultipleImagePixels.

SkipEmptyImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

If the input image contains at least one pixel, then it is copied to the output; otherwise Nil is returned.

Applications: Secures against domain errors caused by empty images.

Syntax

```
void avl::SkipEmptyImage
(
 const avl::Image& inImage,
 atl::Conditional<avl::Image>& outNotEmptyImage,
 bool& outIsNotEmpty
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
←	outNotEmptyImage	Conditional<Image>&		
←	outIsNotEmpty	bool&		

TestImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns a sample image.

Applications: Makes it possible to quickly present results of various image processing filters.

Syntax

```
void avl::TestImage
(
 avl::TestImageId::Type inImageId,
 atl::Optional<avl::Image&> outRgbImage = atl::NIL,
 atl::Optional<avl::Image&> outMonoImage = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inImageId	TestImageId::Type		ID of test image
←	outRgbImage	Optional<Image&>	NIL	Output color image
←	outMonoImage	Optional<Image&>	NIL	Output mono image

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outRgbImage**, **outMonoImage**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	This image is not available in your region.

45. Location

Table of content:

Also in AVL Lite

LocationCenter

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns the center point of a pixel indicated by the provided location.

Syntax

```
void avl::LocationCenter
(
 const avl::Location& inLocation,
 avl::Point2D& outCenter
)
```

Parameters

	Name	Type	Default	Description
→	inLocation	const Location&		Pixel location.
←	outCenter	Point2D&		Center point of the provided pixel.

Also in AVL Lite

TranslateLocation

Header: AVL.h
Namespace: avl
Module: FoundationLite

Shifts a Location by a given number of pixels along each axis.

Syntax

```
void avl::TranslateLocation
(
 const avl::Location& inLocation,
 int inDeltaX,
 int inDeltaY,
 bool inInverse,
 avl::Location& outLocation
)
```

Parameters

	Name	Type	Default	Description
→	inLocation	const Location&		Input location
→	inDeltaX	int		Shift along the x axis
→	inDeltaY	int		Shift along the y axis
→	inInverse	bool		Switches to inverse operation
←	outLocation	Location&		Shifted location

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inLocation** and **outLocation**

Read more about In-place Computation.

46. Point3DGrid Basics

Table of content:

CreatePoint3DGridFromImage

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Creates a Point3DGrid structure from coordinates encoded in pixels of image.

Applications: Creating a Point3DGrid structure out of an image obtained from a 3D camera or other external sources that encodes point cloud XYZ coordinates as pixel components of 2D image.

Syntax

```
void avl::CreatePoint3DGridFromImage
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 const avl::PointCloudCoordinateTransform& inXCoordinateTransform,
 const avl::PointCloudCoordinateTransform& inYCoordinateTransform,
 const avl::PointCloudCoordinateTransform& inZCoordinateTransform,
 avl::Point3DGrid& outPoint3DGrid
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Source image with per pixel encoded XYZ coordinates
→	inRoi	Optional<const Region&>	NIL	Region determining valid points in resulting point grid
→	inXCoordinateTransform	const PointCloudCoordinateTransform&	PointCloudCoordinateTransform (Offset: 0.000000D ValueCoordinateTransform: ImageValueCoordinateTransform(ChannelIndex: 0 Scale: 1.0D InvalidValues: Nil) LocationCoordinateTransform: Nil Limits: ValueLimits_f64(MinValue: Nil, MaxValue: Nil))	Description of the creation of the X coordinate
→	inYCoordinateTransform	const PointCloudCoordinateTransform&	PointCloudCoordinateTransform (Offset: 0.000000D ValueCoordinateTransform: ImageValueCoordinateTransform(ChannelIndex: 1 Scale: 1.0D InvalidValues: Nil) LocationCoordinateTransform: Nil Limits: ValueLimits_f64(MinValue: Nil, MaxValue: Nil))	Description of the creation of the Y coordinate
→	inZCoordinateTransform	const PointCloudCoordinateTransform&	PointCloudCoordinateTransform (Offset: 0.000000D ValueCoordinateTransform: ImageValueCoordinateTransform(ChannelIndex: 2 Scale: 1.0D InvalidValues: Nil) LocationCoordinateTransform: Nil Limits: ValueLimits_f64(MinValue: Nil, MaxValue: Nil))	Description of the creation of the Z coordinate
←	outPoint3DGrid	Point3DGrid&		

Description

The operation creates a point cloud object based on the input point image. Coordinates of points included in the cloud are created based on **inXCoordinateTransform**, **inYCoordinateTransform** and **inYCoordinateTransform**. The listed structures describe the transformation of pixel values and their indexes to a given coordinate. At least one of the transforms based on pixel location or value must be present in PointCloudCoordinateTransform.

Example for the coordination of x. We assume that inXCoordinateTransform.locationCoordinateTransform and inXCoordinateTransform.valueCoordinateTransform are not empty. Empty transformation will be skipped.

New value of pixel is given by:

$$P_x = scale_{value} * value_{channel} + scale_{row} * i + scale_{column} * j + offset$$

where

- $scale_{value}$ is inXCoordinateTransform.valueCoordinateTransform.scale
- $value_{channel}$ is image pixel channel value, when channel is inXCoordinateTransform.valueCoordinateTransform.channelIndex
- $scale_{row}$ is inXCoordinateTransform.locationCoordinateTransform.rowScale
- $scale_{column}$ is inXCoordinateTransform.locationCoordinateTransform.columnScale
- i is pixel row index
- j is pixel column index
- $offset$ is inXCoordinateTransform.offset

InvalidValues are checked based on direct pixel value.

Limits are applied to the final, transformed 3D point.

Errors

List of possible exceptions:

Error type	Description
DomainError	Channel index from inXCoordinateTransform is bigger than image channel count in CreatePoint3DGridFromImage.
DomainError	Channel index from inYCoordinateTransform is bigger than image channel count in CreatePoint3DGridFromImage.
DomainError	Channel index from inZCoordinateTransform is bigger than image channel count in CreatePoint3DGridFromImage.
DomainError	PointCloudCoordinateTransform requires at least one non-empty X coordinate transform in CreatePoint3DGridFromImage.
DomainError	PointCloudCoordinateTransform requires at least one non-empty Y coordinate transform in CreatePoint3DGridFromImage.
DomainError	PointCloudCoordinateTransform requires at least one non-empty Z coordinate transform in CreatePoint3DGridFromImage.

See Also

- ArrangePoint3DArray – Creates a surface structure from Point3D array taking into account X and Y coordinates.
- CreateSurfaceFromImage – Creates a Surface structure from coordinates encoded in pixels of an image.

GetPoint3DGridPoint_Interpolated

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Returns an interpolated single point of a point 3D grid.

Syntax

```
void avl::GetPoint3DGridPoint_Interpolated
(
 const avl::Point3DGrid& inPoint3DGrid,
 float inOffsetX,
 float inOffsetY,
 float inStepX,
 float inStepY,
 float inX,
 float inY,
 int inInterpolationRadius,
 avl::Point3D& outPoint
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoint3DGrid	const Point3DGrid&			Input point 3D grid
→	inOffsetX	float			Defines offset of the input grid along X axis
→	inOffsetY	float			Defines offset of the input grid along Y axis
→	inStepX	float	0.01 - ∞	1.0f	Defines step of the input grid along X axis
→	inStepY	float	0.01 - ∞	1.0f	Defines step of the input grid along Y axis
→	inX	float			X coordinate of the input point
→	inY	float			Y coordinate of the input point
→	inInterpolationRadius	int	0 - 65535	1	Radius of vicinity taking into account to interpolate not existing point
←	outPoint	Point3D&			Output point

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect input coordinates in GetPoint3DGridPoint_Interpolated.

SkipEmptyPoint3DGrid

Header: AVL.h
Namespace: avl
Module: Vision3DLite

If the input Point3DGrid has at least one point defined, then it is copied to the output; otherwise Nil is returned.

Applications: Secures against domain errors caused by empty grids, e.g. just before the FitPlaneToPoints filter is to be invoked.

Syntax

```
void avl::SkipEmptyPoint3DGrid
(
 const avl::Point3DGrid& inPoint3DGrid,
 atl::Conditional<avl::Point3DGrid>& outNotEmptyPoint3DGrid,
 bool& outIsNotEmpty
)
```

Parameters

	Name	Type	Default	Description
→	inPoint3DGrid	const Point3DGrid&		
←	outNotEmptyPoint3DGrid	Conditional<Point3DGrid>&		A copy of the input grid, if it is not empty, or Nil otherwise
←	outIsNotEmpty	bool&		Indication if the input grid is not empty

TestPoint3DGrid

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns a sample 3D object.

Syntax

```
void avl::TestPoint3DGrid
(
 TestPoint3DGridState& ioState,
 const float inDensity,
 const float inScaleX,
 const float inScaleY,
 const float inScaleZ,
 avl::Point3DGrid& outPoint3DGrid
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	TestPoint3DGridState&			Object used to maintain state of the function.
→	inDensity	const float	1.0 - 20.0	4.0f	Density of points in output object
→	inScaleX	const float	0.001 - ∞	1.0f	Scaling of output object on X axis
→	inScaleY	const float	0.001 - ∞	1.0f	Scaling of output object on Y axis
→	inScaleZ	const float	0.001 - ∞	1.0f	Scaling of output object on Z axis
←	outPoint3DGrid	Point3DGrid&			Output object

47. Profile Basics

Table of content:

CreateUniformProfile

Header: AVL.h
Namespace: avl
Module: FoundationPro

Creates a profile with all its elements set to the same value.

Syntax

```
void avl::CreateUniformProfile
(
 float inValue,
 int inSize,
 float inXOffset,
 float inXScale,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Range	Default	Description
→	inValue	float			Input common value of all elements
→	inSize	int	0 - +∞	1	Input the number of elements
→	inXOffset	float		0.0f	X offset of the output profile
→	inXScale	float	0.001 - ∞	1.0f	X scale of the output profile
←	outProfile	Profile&			Output profile

GetProfileElement

Header: AVL.h
Namespace: avl
Module: FoundationPro

Gets a single value from a profile, located at the specified index.

Syntax

```
void avl::GetProfileElement
(
 const avl::Profile& inProfile,
 const int inIndex,
 const bool inInverse,
 float& outValue
)
```

Parameters

	Name	Type	Range	Default	Description
→	inProfile	const Profile&			Input profile
→	inIndex	const int	0 - ∞		
→	inInverse	const bool			Reversed order of elements
←	outValue	float&			

Errors

List of possible exceptions:

Error type	Description
DomainError	inIndex is out of profile range in GetProfileElement.

GetProfileElement_Interpolated

Header: AVL.h
Namespace: avl
Module: FoundationPro

Gets a single value from a profile, interpolated at any point.

Syntax

```
void avl::GetProfileElement_Interpolated
(
 const avl::Profile& inProfile,
 bool inCyclic,
 bool inSmooth,
 float inIndex,
 float& outValue
)
```

Parameters

	Name	Type	Range	Default	Description
→	inProfile	const Profile&			Input profile
→	inCyclic	bool			
→	inSmooth	bool			Determines whether an interpolation between two profile elements should be smooth
→	inIndex	float	-∞ - ∞	0.0f	
←	outValue	float&			

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty profile on input in GetProfileElement_Interpolated.
DomainError	Index out of range in GetProfileElement_Interpolated.

GetProfileValue

Header: AVL.h
Namespace: avl
Module: FoundationPro

Gets a single value from a profile, located at the specified X (real-world) coordinate.

Syntax

```
void avl::GetProfileValue
(
 const avl::Profile& inProfile,
 float inX,
 bool inSmooth,
 float& outValue
)
```

Parameters

	Name	Type	Default	Description
→	inProfile	const Profile&		Input profile
→	inX	float		X coordinate at which the profile value is calculated
→	inSmooth	bool		Determines whether an interpolation between two profile elements should be smooth
←	outValue	float&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty profile on input in GetProfileValue.
DomainError	X coordinate out of range in GetProfileValue.

ProfileCoordinates

Header: AVL.h
Namespace: avl
Module: FoundationPro

Returns an array of all X and Y coordinates of the input profile.

Syntax

```
void avl::ProfileCoordinates
(
 const avl::Profile& inProfile,
 atl::Optional<const avl::Range&> inRange,
 atl::Array<float>& outXCoordinates,
 atl::Array<float>& outYCoordinates
)
```

Parameters

	Name	Type	Default	Description
→	inProfile	const Profile&		Input profile
→	inRange	Optional<const Range&>	NIL	
←	outXCoordinates	Array<float>&		
←	outYCoordinates	Array<float>&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Range exceeds the input profile in ProfileCoordinates.

ProfileIndices

Header: AVL.h
Namespace: avl
Module: FoundationPro

Returns an array of all indices of the input profile (0, 1, 2, ...).

Syntax

```
void avl::ProfileIndices
(
 const avl::Profile& inProfile,
 atl::Array<int>& outIndices
)
```

Parameters

	Name	Type	Default	Description
→	inProfile	const Profile&		Input profile
←	outIndices	Array<int>&		

SetProfileElement

Header: AVL.h
Namespace: avl
Module: FoundationPro

Sets a single element in a profile, located at the specified index.

Syntax

```
void avl::SetProfileElement
(
 avl::Profile& ioProfile,
 int inIndex,
 bool inInverse,
 float inValue
)
```

Parameters

	Name	Type	Range	Default	Description
	ioProfile	Profile&			
	inIndex	int	0 - ∞		
	inInverse	bool			Reversed order of elements
	inValue	float			

Errors

List of possible exceptions:

Error type	Description
DomainError	inIndex is out of profile range in SetProfileElement.

SetProfileXTransform

Header: AVL.h
Namespace: avl
Module: FoundationPro

Sets the offset and the scale of a profile in the X axis.

Applications: Makes it possible to use real-world X coordinates in a profile.

Syntax

```
void avl::SetProfileXTransform
(
 avl::Profile& ioProfile,
 float inXOffset,
 float inXScale
)
```

Parameters

	Name	Type	Range	Default	Description
	ioProfile	Profile&			
	inXOffset	float		0.0f	
	inXScale	float	0.001 - ∞	1.0f	

SkipEmptyProfile

Header: AVL.h
Namespace: avl
Module: FoundationPro

If the input profile contains at least one element, then it is copied to the output; otherwise Nil is returned.

Applications: Secures against domain errors caused by empty profiles, e.g. just before the ProfileAverage filter is to be invoked.

Syntax

```
void avl::SkipEmptyProfile
(
 const avl::Profile& inProfile,
 atl::Conditional<avl::Profile>& outNotEmptyProfile,
 bool& outIsNotEmpty
)
```

Parameters

	Name	Type	Default	Description
→	inProfile	const Profile&		Input profile
←	outNotEmptyProfile	Conditional<Profile>&		
←	outIsNotEmpty	bool&		

48. Deep Learning

Table of content:

MergeCharactersIntoLines

Header: AVL.h
Namespace: avl
Module: DL_OCR

Converts a output of DL_ReadCharacters to lines of text.

Syntax

```
void avl::MergeCharactersIntoLines
(
 const atl::Array<avl::OcrResult>& inCharacters,
 float inMaxGap,
 float inMaxShift,
 float inMargin,
 int inMinLength,
 bool inFlatten,
 atl::Array<avl::Rectangle2D >& outLines,
 atl::Array <atl::String >& outStrings,
 atl::Array< atl::Conditional<int> >& outMapping
)
```

Parameters

	Name	Type	Range	Default	Description
→	inCharacters	const Array<OcrResult>&			Output of DL_ReadCharacters
→	inMaxGap	float	0.0 - 10.0	0.25f	Maximum horizontal gap between joint characters' boxes, denoted as % of 'A' char height
→	inMaxShift	float	0.0 - 1.0	0.25f	Maximum vertical misalignment between joint character's boxes, denoted as % of 'A' char height
→	inMargin	float	0.0 - 10.0		Additional margin added to result, denoted as % of 'A' char height
→	inMinLength	int	1 - 200	1	Minimal number of chars to create line
→	inFlatten	bool		False	If True, it concatenates the words on the line into a single result string, otherwise each word is a separate result string
←	outLines	Array<Rectangle2D >&			Minimal Box which cover all selected character boxes
←	outStrings	Array <String >&			Text of merged characters
←	outMapping	Array< Conditional<int> >&			Mapping between input characters and output lines, outMapping[i] stores the index line to which inCharacters[i] belongs. If outMapping[i] is NIL it means that inCharacters[i] has not been added to any line

49. Configuration

Table of content:

ChargeImageMemoryPools

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Preallocates memory buffers for images.

Applications: This may improve performance when many images are created and destroyed in an application.

Syntax

```
void avl::ChargeImageMemoryPools
(
 const atl::Array<int>& inRequestedPoolSizes,
 bool inTouch
)
```

Parameters

	Name	Type	Default	Description
→	inRequestedPoolSizes	const Array<int>&		Copy a value from InspectImageMemoryPools.outPoolSizes
→	inTouch	bool		Forces getting the new memory pools to the cache memory

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect size of 'inRequestedPoolSizes' array.

CheckLicense

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Check license for a specified module.

Syntax

```
void avl::CheckLicense
(
 avl::ProductType::Type inProductType,
 bool& outIsValid
)
```

Parameters

	Name	Type	Default	Description
→	inProductType	ProductType::Type		
←	outIsValid	bool&		

ControlAVX2

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Enables or disables AVX2 cpu extension usage by other filters.

Syntax

```
void avl::ControlAVX2
(
 bool inIsEnabled,
 bool& outIsAvailable
)
```

Parameters

	Name	Type	Default	Description
→	inIsEnabled	bool	True	When 'true' AVX2 instructions are used
←	outIsAvailable	bool&		

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

ControlImageMemoryPools

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Enables or disables deterministic image memory allocator.

Applications: This may improve performance when many images are created and destroyed in an application.

Syntax

```
void avl::ControlImageMemoryPools
(
 bool inIsEnabled
)
```

Parameters

	Name	Type	Default	Description
→	inIsEnabled	bool	True	Specifies whether memory pools should be used or not

ControlParallelComputing

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Enables or disables filters multithreading parallelization.

Syntax

```
void avl::ControlParallelComputing
(
 bool inIsEnabled,
 atl::Optional<int> inThreadCount,
 atl::Optional<int> inBlockTime,
 int& outThreadCount
)
```

Parameters

	Name	Type	Range	Default	Description
→	inIsEnabled	bool		True	When 'true' parallel computing is used.
→	inThreadCount	Optional<int>	1 - 32	NIL	Requested number of threads.
→	inBlockTime	Optional<int>		NIL	Sets the number of milliseconds that a thread should wait, after completing the execution of a parallel region, before sleeping.
←	outThreadCount	int&			Actually available number of threads.

ControlSIMD

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Enables or disables SIMD cpu extension (e.g. SSE, AVX2, NEON) usage by other filters. This filter does not affect third party components e.g. camera software, OpenCV.

Syntax

```
void avl::ControlSIMD
(
 atl::Optional<avl::SimdLevel::Type> inLevel,
 avl::SimdLevel::Type& outLevel
)
```

Parameters

	Name	Type	Default	Description
→	inLevel	Optional<SimdLevel::Type>	NIL	Maximum simd level to be enabled
←	outLevel	SimdLevel::Type&		Actually adjusted simd level

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

ControlSSE

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Enables or disables SSE cpu extension usage by other filters.

Syntax

```
void avl::ControlSSE
(
 bool inIsEnabled,
 bool& outIsAvailable
)
```

Parameters

	Name	Type	Default	Description
→	inIsEnabled	bool	True	When 'true' SSE instructions are used
←	outIsAvailable	bool&		

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

DischargeImageMemoryPools

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Releases preallocated image memory buffers.

Syntax

```
void avl::DischargeImageMemoryPools
(
)
```

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

GetAvailableLicenses

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Reads available licenses in the system.

Syntax

```
void avl::GetAvailableLicenses
(
 atl::Array<avl::License>& outLicenses
)
```

Parameters

	Name	Type	Default	Description
⬅	outLicenses	Array<License>&		

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

GetComputerID

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Gets current Computer ID.

Applications: Can be used for creating custom license protection for integrator's applications.

Syntax

```
void avl::GetComputerID
(
 atl::String& outComputerID
)
```

Parameters

	Name	Type	Default	Description
⬅	outComputerID	String&		

GetDongleSerialNumber

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Gets the current usb dongle's serial number.

Syntax

```
void avl::GetDongleSerialNumber
(
 atl::String& outDongleSerialNumber
)
```

Parameters

	Name	Type	Default	Description
⬅	outDongleSerialNumber	String&		

GetLibraryVersion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Gets current library version.

Syntax

```
void avl::GetLibraryVersion
(
 int& outMajor,
 int& outMinor,
 int& outBuild,
 int& outRevision
)
```

Parameters

	Name	Type	Default	Description
⬅	outMajor	int&		
⬅	outMinor	int&		
⬅	outBuild	int&		
⬅	outRevision	int&		

GetThreadLimitInfo

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns how many threads are possible and a list of threads currently being used.

Applications: Makes it easier to diagnose licensing problems related to exceeding the limit of threads.

Syntax

```
void avl::GetThreadLimitInfo
(
 atl::Conditional<int>& outThreadLimit,
 atl::Array<atl::String>& outActiveThreadIds
)
```

Parameters

	Name	Type	Default	Description
⬅	outThreadLimit	Conditional<int>&		
⬅	outActiveThreadIds	Array<String>&		

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

InitLibrary

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Initializes library internals. Should be called before any other function from AVL.

Applications: If you do not call this first, initialization will be done in the first iteration, possibly affecting system's performance.

Syntax

```
void avl::InitLibrary
(
)
```

InspectImageMemoryPools

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns information about allocated image memory buffers. This information can be used for preallocation.

Syntax

```
void avl::InspectImageMemoryPools
(
 bool& outIsEnabled,
 atl::Array<int>& outPoolSizes,
 atl::Array<int>& outBlockLengths,
 int& outTotalUsage
)
```

Parameters

	Name	Type	Default	Description
⬅	outIsEnabled	bool&		Specifies whether memory pools are used or not
⬅	outPoolSizes	Array<int>&		Count of blocks in consecutive pools
⬅	outBlockLengths	Array<int>&		Length of blocks in consecutive pools
⬅	outTotalUsage	int&		Total image memory allocated [B]

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VerifyProjectID

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Verifies a project ID.

Applications: This tools is useful for system integrators who need verification if a system is used with a license coming from a legitimate source.

Syntax

```
void avl::VerifyProjectID
(
 int inProjectID,
 bool inThrowError,
 bool& outIsLicenseValid
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inProjectID	int	0 - ∞		Project ID assigned to your license
➡	inThrowError	bool			
⬅	outIsLicenseValid	bool&			

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

WriteLog

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Write log message with selected log level.

Syntax

```
void avl::WriteLog
(
 const atl::String& inMessage,
 const avl::LogsLevel::Type& inLogLevel
)
```

Parameters

	Name	Type	Default	Description
➡	inMessage	const String&		
➡	inLogLevel	const LogsLevel::Type&		

50. Shape Fitting 3D

Table of content:

CreateCircleFittingMap3D

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Precomputes a data object that is required for fast circle fitting on surfaces.

Applications: Used together with circle fitting, but can be moved before the loop.

Syntax

```
void avl::CreateCircleFittingMap3D
(
 const avl::SurfaceFormat& inSurfaceFormat,
 const avl::CircleFittingField& inFittingField,
 atl::Optional<const avl::CoordinateSystem2D&> inFittingFieldAlignment,
 const int inScanCount,
 atl::Optional<float> inSamplingStep,
 int inScanWidth,
 avl::InterpolationMethod::Type inSurfaceInterpolation,
 avl::CircleFittingMap& outFittingMap,
 atl::Array<avl::Segment2D>& diagScanSegments,
 atl::Array<avl::Rectangle2D>& diagSamplingAreas
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurfaceFormat	const SurfaceFormat&			Dimensions, depth image pixel type, coordinate offsets and scales of a surface on which circle fitting will be performed
→	inFittingField	const CircleFittingField&			Defines a ring in which scan segments will be created
→	inFittingFieldAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the fitting field to the position of the inspected object
→	inScanCount	const int	3 - ∞	10	The number of points that will be searched to estimate the position of the circle
→	inSamplingStep	Optional<float>		NIL	Desired distance between consecutive sampling points on the scan segments; if Nil, the bigger of surface X and Y scales is chosen
→	inScanWidth	int	1 - ∞	5	The width of each scan field in pixels of the surface depth image
→	inSurfaceInterpolation	InterpolationMethod::Type		InterpolationMethod::NearestNeighbour	Interpolation method used for extraction of depth image pixel values
←	outFittingMap	CircleFittingMap&			Optimized data required for circle fitting
🔍	diagScanSegments	Array<Segment2D>&			Segments along which the scans will be run
🔍	diagSamplingAreas	Array<Rectangle2D>&			Scan fields created for point detection

CreatePathFittingMap3D

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Precomputes a data object that is required for fast path fitting on surfaces.

Applications: Used together with path fitting, but can be moved before the loop.

Syntax

```
void avl::CreatePathFittingMap3D
(
 const avl::SurfaceFormat& inSurfaceFormat,
 const avl::PathFittingField& inFittingField,
 atl::Optional<const avl::CoordinateSystem2D&> inFittingFieldAlignment,
 atl::Optional<float> inScanStep,
 atl::Optional<float> inSamplingStep,
 int inScanWidth,
 avl::InterpolationMethod::Type inSurfaceInterpolation,
 avl::PathFittingMap& outFittingMap,
 atl::Array<avl::Segment2D>& diagScanSegments,
 atl::Array<avl::Rectangle2D>& diagSamplingAreas
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurfaceFormat	const SurfaceFormat&			Dimensions, depth image pixel type, coordinate offsets and scales of a surface on which path fitting will be performed
→	inFittingField	const PathFittingField&			Defines a stripe in which scan segments will be created
→	inFittingFieldAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the fitting field to the position of the inspected object
→	inScanStep	Optional<float>	0.0 - ∞	NIL	Optional implicit conversion of the input path to an equidistant one
→	inSamplingStep	Optional<float>		NIL	Desired distance between consecutive sampling points on the scan segments; if Nil, the bigger of surface X and Y scales is chosen
→	inScanWidth	int	1 - ∞	5	The width of each scan field in pixels of the surface depth image
→	inSurfaceInterpolation	InterpolationMethod::Type		InterpolationMethod::NearestNeighbour	Interpolation method used for extraction of depth image pixel values
←	outFittingMap	PathFittingMap&			Optimized data required for path fitting
🔍	diagScanSegments	Array<Segment2D>&			Segments along which the scans will be run
🔍	diagSamplingAreas	Array<Rectangle2D>&			Scan fields created for point detection

CreateSegmentFittingMap3D

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Precomputes a data object that is required for fast segment fitting on surfaces.

Applications: Used together with segment fitting, but can be moved before the loop.

Syntax

```
void avl::CreateSegmentFittingMap3D
(
 const avl::SurfaceFormat& inSurfaceFormat,
 const avl::SegmentFittingField& inFittingField,
 atl::Optional<const avl::CoordinateSystem2D&> inFittingFieldAlignment,
 const int inScanCount,
 atl::Optional<float> inSamplingStep,
 int inScanWidth,
 avl::InterpolationMethod::Type inSurfaceInterpolation,
 avl::SegmentFittingMap& outFittingMap,
 atl::Array<avl::Segment2D>& diagScanSegments,
 atl::Array<avl::Rectangle2D>& diagSamplingAreas
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurfaceFormat	const SurfaceFormat&			Dimensions, depth image pixel type, coordinate offsets and scales of a surface on which segment fitting will be performed
→	inFittingField	const SegmentFittingField&			Defines a rectangle in which scan segments will be created
→	inFittingFieldAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the fitting field to the position of the inspected object
→	inScanCount	const int	3 - ∞	10	The number of points that will be searched to estimate the position of the segment
→	inSamplingStep	Optional<float>		NIL	Desired distance between consecutive sampling points on the scan segments; if Nil, the bigger of surface X and Y scales is chosen
→	inScanWidth	int	1 - ∞	5	The width of each scan field in pixels of the surface depth image
→	inSurfaceInterpolation	InterpolationMethod::Type		InterpolationMethod::NearestNeighbour	Interpolation method used for extraction of depth image pixel values
←	outFittingMap	SegmentFittingMap&			Optimized data required for segment fitting
🔍	diagScanSegments	Array<Segment2D>&			Segments along which the scans will be run
🔍	diagSamplingAreas	Array<Rectangle2D>&			Scan fields created for point detection

CreateSurfaceMeasurementMap

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

(Pre)computes surface sampling locations used by MeasureObjectWidth3D function.

Syntax

```
void avl::CreateSurfaceMeasurementMap
(
 const avl::SurfaceFormat& inSurfaceFormat,
 const avl::SegmentScanField& inScanField,
 atl::Optional<const avl::CoordinateSystem2D&> inScanFieldAlignment,
 int inScanCount,
 atl::Optional<float> inSamplingStep,
 int inScanWidth,
 avl::InterpolationMethod::Type inSurfaceInterpolation,
 atl::Array<avl::ScanMap>& outMeasurementMap,
 atl::Optional<avl::SegmentScanField&> outAlignedScanField = atl::NIL,
 atl::Array<avl::Segment2D>& diagScanSegments
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurfaceFormat	const SurfaceFormat&			Information about dimensions, depth and pixel type of the surface
→	inScanField	const SegmentScanField&			Field in which scans will be performed
→	inScanFieldAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the scan field to the position of the inspected object
→	inScanCount	int	2 - ∞	5	Number of scans to be performed
→	inSamplingStep	Optional<float>		NIL	Desired distance between consecutive sampling points on the scan segments; if Nil, the bigger of surface X and Y scales is chosen
→	inScanWidth	int	1 - ∞	5	Width of the scan area
→	inSurfaceInterpolation	InterpolationMethod::Type		Bilinear	Interpolation method used in extraction of image pixel values
←	outMeasurementMap	Array<ScanMap>&			
←	outAlignedScanField	Optional<SegmentScanField&>		NIL	Field in which the scans will be performed
🔍	diagScanSegments	Array<Segment2D>&			Array of scan segments

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedScanField**.

Read more about Optional Outputs.

FitCircleToEdges3D

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Performs a series of 1D edge detections and finds a circle that best matches the detected points.

Applications: Precise detection of a circular object or hole, whose rough location is known beforehand.

Syntax

```
void avl::FitCircleToEdges3D
(
 const avl::Surface& inSurface,
 const CircleFittingMap& inFittingMap,
 const EdgeScanParams3D& inEdgeScanParams,
 avl::Selection::Type inEdgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 float inMaxIncompleteness,
 avl::CircleFittingMethod::Type inFittingMethod,
 atl::Optional<avl::MEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Circle3D>& outCircle,
 atl::Optional<atl::Array<atl::Conditional<avl::SurfaceEdge1D>>&> outEdges = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<atl::Array<avl::Point3D>&> outInliers = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outHeightProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Surface to fit the circle to
→	inFittingMap	const CircleFittingMap&			Input fitting map
→	inEdgeScanParams	const EdgeScanParams3D&			Parameters controlling the edge extraction process
→	inEdgeSelection	Selection::Type		SelectionBest	Selection mode of edges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of edge points not found
→	inFittingMethod	CircleFittingMethod::Type		AlgebraicTaubin	Method used to fit a circle
→	inOutlierSuppression	Optional<MEstimator::Type>		NIL	Selects a method for ignoring incorrectly detected points
←	outCircle	Conditional<Circle3D>&			Fitted circle or nothing if the fitting fails
←	outEdges	Optional<Array<Conditional<SurfaceEdge1D>>&>		NIL	Found edges
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual circle points and the corresponding reference circle points
←	outInliers	Optional<Array<Point3D>&>		NIL	Points matching the fitting Circle
←	outHeightProfiles	Optional<Array<Profile>&>		NIL	Extracted surface height profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the edge (derivative) operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outEdges**, **outDeviationProfile**, **outInliers**, **outHeightProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

FitCircleToEdges3D_Direct

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Performs a series of 1D edge detections in 3D and finds a circle that best matches the detected points.

Applications: Precise detection of a circular object or hole, whose rough location is known beforehand.

Syntax

```
void avl::FitCircleToEdges3D_Direct
(
 const avl::Surface& inSurface,
 const avl::CircleFittingField& inFittingField,
 atl::Optional<const avl::CoordinateSystem2D&> inFittingFieldAlignment,
 int inScanCount,
 atl::Optional<float> inSamplingStep,
 int inScanWidth,
 const avl::InterpolationMethod::Type inSurfaceInterpolation,
 const avl::EdgeScanParams3D& inEdgeScanParams,
 avl::Selection::Type inEdgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 float inMaxIncompleteness,
 avl::CircleFittingMethod::Type inFittingMethod,
 atl::Optional<avl::MEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Circle3D>& outCircle,
 atl::Optional<atl::Array<atl::Conditional<avl::SurfaceEdge1D>>&> outEdges = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<avl::CircleFittingField&> outAlignedFittingField = atl::NIL,
 atl::Optional<atl::Array<avl::Point3D>&> outInliers = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outHeightProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL,
 atl::Array<avl::Segment2D>& diagScanSegments,
 atl::Array<avl::Rectangle2D>& diagSamplingAreas
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Surface to fit the circle to
→	inFittingField	const CircleFittingField&			Circle fitting field
→	inFittingFieldAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the fitting field to the position of the inspected object
→	inScanCount	int	3 - ∞	10	The number of points that will be searched to estimate the position of the circle
→	inSamplingStep	Optional<float>	0.0 - ∞	NIL	Desired distance between consecutive sampling points on the scan segments; if Nil, the bigger of surface X and Y scales is chosen
→	inScanWidth	int	1 - ∞	5	The width of each scan field (in pixels)
→	inSurfaceInterpolation	const InterpolationMethod::Type		Bilinear	Interpolation method used for extraction of surface points
→	inEdgeScanParams	const EdgeScanParams3D&		EdgeScanParams3D (ProfileInterpolation: Quadratic4 SmoothingStdDev: 1.0f MinMagnitude: 5.0f EdgeTransition: LowToHigh)	Parameters controlling the edge extraction process
→	inEdgeSelection	Selection::Type			Selection mode of edges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of edge points not found
→	inFittingMethod	CircleFittingMethod::Type		AlgebraicTaubin	Method used to fit a circle
→	inOutlierSuppression	Optional<MEstimator::Type>		NIL	Selects a method for ignoring incorrectly detected points
←	outCircle	Conditional<Circle3D>&			Fitted circle or nothing if the fitting fails
←	outEdges	Optional<Array<Conditional<SurfaceEdge1D>>&>		NIL	Found edges
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual circle points and the corresponding reference circle points
←	outAlignedFittingField	Optional<CircleFittingField&>		NIL	Fitting field used; in the image coordinate system
←	outInliers	Optional<Array<Point3D>&>		NIL	Points matching the fitting Circle
←	outHeightProfiles	Optional<Array<Profile>&>		NIL	Extracted surface height profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the edge (derivative) operator response
🔍	diagScanSegments	Array<Segment2D>&			Segments along which the scans were run
🔍	diagSamplingAreas	Array<Rectangle2D>&			Areas from which the input image is sampled

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outEdges**, **outDeviationProfile**, **outAlignedFittingField**, **outInliers**, **outHeightProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Sampling step set to zero in FitCircleToEdges3D.

FitCircleToRidges3D

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Performs a series of 1D ridge detections and finds a circle that best matches the detected points.

Applications: Precise detection of a circular object or hole, whose rough location is known beforehand.

Syntax

```
void avl::FitCircleToRidges3D
(
 const avl::Surface& inSurface,
 const CircleFittingMap& inFittingMap,
 const RidgeScanParams3D& inRidgeScanParams,
 avl::Selection::Type inRidgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 float inMaxIncompleteness,
 avl::CircleFittingMethod::Type inFittingMethod,
 atl::Optional<avl::MEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Circle3D>& outCircle,
 atl::Optional<atl::Array<atl::Conditional<avl::SurfaceRidge1D>>&> outRidges = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<atl::Array<avl::Point3D>&> outInliers = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outHeightProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Surface to fit the circle to
→	inFittingMap	const CircleFittingMap&			Input fitting map
→	inRidgeScanParams	const RidgeScanParams3D&			Parameters controlling the ridge extraction process
→	inRidgeSelection	Selection::Type		SelectionBest	Selection mode of ridges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker ridges can be detected in the vicinity of stronger ridges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of ridge points not found
→	inFittingMethod	CircleFittingMethod::Type		AlgebraicTaubin	Method used to fit a circle
→	inOutlierSuppression	Optional<MEstimator::Type>		NIL	Selects a method for ignoring incorrectly detected points
←	outCircle	Conditional<Circle3D>&			Fitted circle or nothing if the fitting fails
←	outRidges	Optional<Array<Conditional<SurfaceRidge1D>>&>		NIL	Found ridges
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual circle points and the corresponding reference circle points
←	outInliers	Optional<Array<Point3D>&>		NIL	Points matching the fitting Circle
←	outHeightProfiles	Optional<Array<Profile>&>		NIL	Extracted surface height profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the ridge operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outRidges**, **outDeviationProfile**, **outInliers**, **outHeightProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Ridge operator parameters are too low in surface ridges detector in FitCircleToRidges3D.

FitCircleToRidges3D_Direct

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Performs a series of 1D ridge detections in 3D and finds a circle that best matches the detected points.

Applications: Precise detection of a circular object or hole, whose rough location is known beforehand.

Syntax

```
void avl::FitCircleToRidges3D_Direct
(
 const avl::Surface& inSurface,
 const avl::CircleFittingField& inFittingField,
 atl::Optional<const avl::CoordinateSystem2D&> inFittingFieldAlignment,
 int inScanCount,
 atl::Optional<float> inSamplingStep,
 int inScanWidth,
 const avl::InterpolationMethod::Type inSurfaceInterpolation,
 const avl::RidgeScanParams3D& inRidgeScanParams,
 avl::Selection::Type inRidgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 float inMaxIncompleteness,
 avl::CircleFittingMethod::Type inFittingMethod,
 atl::Optional<avl::MEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Circle3D>& outCircle,
 atl::Optional<atl::Array<atl::Conditional<avl::SurfaceRidge1D>>&> outRidges = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<avl::CircleFittingField&> outAlignedFittingField = atl::NIL,
 atl::Optional<atl::Array<avl::Point3D>&> outInliers = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outHeightProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL,
 atl::Array<avl::Segment2D>& diagScanSegments,
 atl::Array<avl::Rectangle2D>& diagSamplingAreas
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Surface to fit the circle to
→	inFittingField	const CircleFittingField&			Circle fitting field
→	inFittingFieldAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the fitting field to the position of the inspected object
→	inScanCount	int	3 - ∞	10	The number of points that will be searched to estimate the position of the circle
→	inSamplingStep	Optional<float>	0.0 - ∞	NIL	Desired distance between consecutive sampling points on the scan segments; if Nil, the bigger of surface X and Y scales is chosen
→	inScanWidth	int	1 - ∞	5	The width of each scan field (in pixels)
→	inSurfaceInterpolation	const InterpolationMethod::Type		Bilinear	Interpolation method used for extraction of surface points
→	inRidgeScanParams	const RidgeScanParams3D&		RidgeScanParams3D (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f RidgeWidth: 5.0f RidgeMargin: 2.0f RidgeOperator: Minimum MinMagnitude: 5.0f RidgePolarity: Low)	Parameters controlling the ridge extraction process
→	inRidgeSelection	Selection::Type			Selection mode of ridges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker ridges can be detected in the vicinity of stronger ridges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of ridge points not found
→	inFittingMethod	CircleFittingMethod::Type		AlgebraicTaubin	Method used to fit a circle
→	inOutlierSuppression	Optional<MEstimator::Type>		NIL	Selects a method for ignoring incorrectly detected points
←	outCircle	Conditional<Circle3D>&			Fitted circle or nothing if the fitting fails
←	outRidges	Optional<Array<Conditional<SurfaceRidge1D>>&>		NIL	Found ridges
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual circle points and the corresponding reference circle points
←	outAlignedFittingField	Optional<CircleFittingField&>		NIL	Fitting field used; in the image coordinate system
←	outInliers	Optional<Array<Point3D>&>		NIL	Points matching the fitting Circle
←	outHeightProfiles	Optional<Array<Profile>&>		NIL	Extracted surface height profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the ridge operator response
🔍	diagScanSegments	Array<Segment2D>&			Segments along which the scans were run
🔍	diagSamplingAreas	Array<Rectangle2D>&			Areas from which the input image is sampled

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outRidges**, **outDeviationProfile**, **outAlignedFittingField**, **outInliers**, **outHeightProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Ridge operator parameters are too low in surface ridges detector in FitCircleToRidges3D.
DomainError	Sampling step set to zero in FitCircleToRidges3D.

FitCircleToStripe3D

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Performs a series of 1D edge detections and finds a circle that best matches the detected points.

Applications: Precise detection of a circular object or hole, whose rough location is known beforehand.

Syntax

```
void avl::FitCircleToStripe3D
(
 const avl::Surface& inSurface,
 const CircleFittingMap& inFittingMap,
 const StripeScanParams3D& inStripeScanParams,
 avl::Selection::Type inStripeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 float inMaxIncompleteness,
 avl::CircleFittingMethod::Type inFittingMethod,
 atl::Optional<avl::MEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Circle3D>& outCircle,
 atl::Conditional<avl::Circle3D>& outInnerCircle,
 atl::Conditional<avl::Circle3D>& outOuterCircle,
 atl::Optional<atl::Array<atl::Conditional<avl::SurfaceStripe1D>>&> outStripes = atl::NIL,
 atl::Optional<atl::Array<avl::Point3D>&> outStripePoints = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outHeightProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Surface to fit the circle to
→	inFittingMap	const CircleFittingMap&			Input fitting map
→	inStripeScanParams	const StripeScanParams3D&			Parameters controlling the stripe extraction process
→	inStripeSelection	Selection::Type		Selection::Best	Selection mode of stripe
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of stripe points not found
→	inFittingMethod	CircleFittingMethod::Type		AlgebraicTaubin	Method used to fit a circle
→	inOutlierSuppression	Optional<MEstimator::Type>		NIL	Selects a method for ignoring incorrectly detected points
←	outCircle	Conditional<Circle3D>&			Fitted circle in the middle of found stripe or nothing if the fitting fails
←	outInnerCircle	Conditional<Circle3D>&			Fitted inner circle
←	outOuterCircle	Conditional<Circle3D>&			Fitted outer circle
←	outStripes	Optional<Array<Conditional<SurfaceStripe1D>>&>		NIL	Found stripes
←	outStripePoints	Optional<Array<Point3D>&>		NIL	Extracted points of middle circle of a surface stripe
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual circle points and the corresponding reference circle points
←	outHeightProfiles	Optional<Array<Profile>&>		NIL	Extracted surface height profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the edge (derivative) operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outStripes**, **outStripePoints**, **outDeviationProfile**, **outHeightProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

FitCircleToStripe3D_Direct

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Performs a series of 1D edge detections in 3D and finds a circle that best matches the detected points.

Applications: Precise detection of a circular object or hole, whose rough location is known beforehand.

Syntax

```
void avl::FitCircleToStripe3D_Direct
(
 const avl::Surface& inSurface,
 const avl::CircleFittingField& inFittingField,
 atl::Optional<const avl::CoordinateSystem2D&> inFittingFieldAlignment,
 int inScanCount,
 atl::Optional<float> inSamplingStep,
 int inScanWidth,
 const avl::InterpolationMethod::Type inSurfaceInterpolation,
 const avl::StripeScanParams3D& inStripeScanParams,
 avl::Selection::Type inStripeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 float inMaxIncompleteness,
 avl::CircleFittingMethod::Type inFittingMethod,
 atl::Optional<avl::MEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Circle3D>& outCircle,
 atl::Conditional<avl::Circle3D>& outInnerCircle,
 atl::Conditional<avl::Circle3D>& outOuterCircle,
 atl::Optional<atl::Array<atl::Conditional<avl::SurfaceStripe1D>>&> outStripes = atl::NIL,
 atl::Optional<atl::Array<avl::Point3D>&> outStripePoints = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<avl::CircleFittingField&> outAlignedFittingField = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outHeightProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL,
 atl::Array<avl::Segment2D>& diagScanSegments,
 atl::Array<avl::Rectangle2D>& diagSamplingAreas
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Surface to fit the circle to
→	inFittingField	const CircleFittingField&			Circle fitting field
→	inFittingFieldAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the fitting field to the position of the inspected object
→	inScanCount	int	3 - ∞	10	The number of points that will be searched to estimate the position of the circle
→	inSamplingStep	Optional<float>	0.0 - ∞	NIL	Desired distance between consecutive sampling points on the scan segments; if Nil, the bigger of surface X and Y scales is chosen
→	inScanWidth	int	1 - ∞	5	The width of each scan field (in pixels)
→	inSurfaceInterpolation	const InterpolationMethod::Type		Bilinear	Interpolation method used for extraction of surface points
→	inStripeScanParams	const StripeScanParams3D&		StripeScanParams3D (ProfileInterpolation: Quadratic4 SmoothingStdDev: 1.0f MinMagnitude: 5.0f MaxInnerEdgeMagnitude: Nil MinStripeWidth: 0.0f MaxStripeWidth: Nil StripePolarity: High)	Parameters controlling the stripe extraction process
→	inStripeSelection	Selection::Type			Selection mode of stripe
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of stripe points not found
→	inFittingMethod	CircleFittingMethod::Type		AlgebraicTaubin	Method used to fit a circle
→	inOutlierSuppression	Optional<MEstimator::Type>		NIL	Selects a method for ignoring incorrectly detected points
←	outCircle	Conditional<Circle3D>&			Fitted circle in the middle of found stripe or nothing if the fitting fails
←	outInnerCircle	Conditional<Circle3D>&			Fitted inner circle
←	outOuterCircle	Conditional<Circle3D>&			Fitted outer circle
←	outStripes	Optional<Array<Conditional<SurfaceStripe1D>>&>		NIL	Found stripes
←	outStripePoints	Optional<Array<Point3D>&>		NIL	Extracted points of middle circle of a surface stripe
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual circle points and the corresponding reference circle points
←	outAlignedFittingField	Optional<CircleFittingField&>		NIL	Fitting field used; in the image coordinate system
←	outHeightProfiles	Optional<Array<Profile>&>		NIL	Extracted surface height profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the edge (derivative) operator response
🔍	diagScanSegments	Array<Segment2D>&			Segments along which the scans were run
🔍	diagSamplingAreas	Array<Rectangle2D>&			Areas from which the input image is sampled

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outStripes**, **outStripePoints**, **outDeviationProfile**, **outAlignedFittingField**, **outHeightProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Sampling step set to zero in FitCircleToStripe3D.

FitPathToEdges3D

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Performs a series of 1D edge detections and creates a path from the detected points.

Applications: Tracing of an object contour, whose rough location and shape is known beforehand.

Syntax

```
void avl::FitPathToEdges3D
(
 const avl::Surface& inSurface,
 const avl::PathFittingMap& inFittingMap,
 const EdgeScanParams3D& inEdgeScanParams,
 avl::Selection::Type inEdgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 atl::Optional<int> inMaxPathInterpolationLength,
 atl::Optional<float> inMaxDeviationDelta,
 float inMaxIncompleteness,
 atl::Conditional<atl::Array<avl::Point3D> >& outPath,
 atl::Optional<atl::Array<atl::Conditional<avl::SurfaceEdge1D>>&> outEdges = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outHeightProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL,
 atl::Optional<atl::Conditional<atl::Array<avl::Segment3D>>&> outPathSegments = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Surface to fit the path to
→	inFittingMap	const PathFittingMap&			Input fitting map
→	inEdgeScanParams	const EdgeScanParams3D&			Parameters controlling the edge extraction process
→	inEdgeSelection	Selection::Type		Selection::Best	Selection mode of edges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
→	inMaxPathInterpolationLength	Optional<int>		NIL	Maximal number of consecutive points not found
→	inMaxDeviationDelta	Optional<float>	0.0 - ∞	NIL	Maximal difference between deviations of consecutive path points
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of edge points not found
←	outPath	Conditional<Array<Point3D> >&			Fitted path or nothing if the fitting failed
←	outEdges	Optional<Array<Conditional<SurfaceEdge1D>>&>		NIL	Found edges
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual path points and the corresponding reference path points
←	outHeightProfiles	Optional<Array<Profile>&>		NIL	Extracted surface height profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the edge (derivative) operator response
←	outPathSegments	Optional<Conditional<Array<Segment3D>>&>		NIL	Segments of the fitted path or nothing if the fitting failed

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outEdges**, **outDeviationProfile**, **outHeightProfiles**, **outResponseProfiles**, **outPathSegments**.

Read more about Optional Outputs.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

FitPathToEdges3D_Direct

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Performs a series of 1D edge detections in 3D and creates a path from the detected points.

Applications: Tracing of an object contour, whose rough location and shape is known beforehand.

Syntax

```
void avl::FitPathToEdges3D_Direct
(
 const avl::Surface& inSurface,
 const avl::PathFittingField& inFittingField,
 atl::Optional<const avl::CoordinateSystem2D&> inFittingFieldAlignment,
 atl::Optional<float> inScanStep,
 atl::Optional<float> inSamplingStep,
 int inScanWidth,
 const avl::InterpolationMethod::Type inSurfaceInterpolation,
 const avl::EdgeScanParams3D& inEdgeScanParams,
 avl::Selection::Type inEdgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 atl::Optional<int> inMaxPathInterpolationLength,
 atl::Optional<float> inMaxDeviationDelta,
 float inMaxIncompleteness,
 atl::Conditional<atl::Array<avl::Point3D> >& outPath,
 atl::Optional<atl::Array<atl::Conditional<avl::SurfaceEdge1D>>&> outEdges = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<avl::PathFittingField&> outAlignedFittingField = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outHeightProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL,
 atl::Optional<atl::Conditional<atl::Array<avl::Segment3D> >&> outPathSegments = atl::NIL,
 atl::Array<avl::Segment2D>& diagScanSegments,
 atl::Array<avl::Rectangle2D>& diagSamplingAreas
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Surface to fit path to
→	inFittingField	const PathFittingField&			Path fitting field
→	inFittingFieldAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the fitting field to the position of the inspected object
→	inScanStep	Optional<float>	0.0 - ∞	5.0f	Optional implicit conversion of the input path to an equidistant one
→	inSamplingStep	Optional<float>	0.0 - ∞	NIL	Desired distance between consecutive sampling points on the scan segments; if Nil, the bigger of surface X and Y scales is chosen
→	inScanWidth	int	1 - ∞	5	The width of each scan field (in pixels)
→	inSurfaceInterpolation	const InterpolationMethod::Type		Bilinear	Interpolation method used for extraction of surface points
→	inEdgeScanParams	const EdgeScanParams3D&		EdgeScanParams3D (ProfileInterpolation: Quadratic4 SmoothingStdDev: 1.0f MinMagnitude: 5.0f EdgeTransition: LowToHigh)	Parameters controlling the edge extraction process
→	inEdgeSelection	Selection::Type			Selection mode of edges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
→	inMaxPathInterpolationLength	Optional<int>	0 - ∞	1	Maximal number of consecutive points not found
→	inMaxDeviationDelta	Optional<float>	0.0 - ∞	NIL	Maximal difference between deviations of consecutive path points
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of edge points not found
←	outPath	Conditional<Array<Point3D> >&			Fitted path or nothing if the fitting failed
←	outEdges	Optional<Array<Conditional<SurfaceEdge1D>>&>		NIL	Found edges
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual path points and the corresponding reference path points
←	outAlignedFittingField	Optional<PathFittingField&>		NIL	Fitting field used; in the image coordinate system
←	outHeightProfiles	Optional<Array<Profile>&>		NIL	Extracted surface height profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the edge (derivative) operator response
←	outPathSegments	Optional<Conditional<Array<Segment3D> >&>		NIL	Segments of the fitted path or nothing if the fitting failed
🔍	diagScanSegments	Array<Segment2D>&			Segments along which the scans were run
🔍	diagSamplingAreas	Array<Rectangle2D>&			Areas from which the input image is sampled

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outEdges**, **outDeviationProfile**, **outAlignedFittingField**, **outHeightProfiles**, **outResponseProfiles**, **outPathSegments**.

Read more about Optional Outputs.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Sampling step set to zero in FitPathToEdges3D.

FitPathToRidges3D

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Performs a series of 1D ridge detections and creates a path from the detected points.

Applications: Tracing of a thin line, whose rough location is known beforehand.

Syntax

```
void avl::FitPathToRidges3D
(
 const avl::Surface& inSurface,
 const PathFittingMap& inFittingMap,
 const RidgeScanParams3D& inRidgeScanParams,
 avl::Selection::Type inRidgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 atl::Optional<int> inMaxPathInterpolationLength,
 atl::Optional<float> inMaxDeviationDelta,
 float inMaxIncompleteness,
 atl::Conditional<atl::Array<avl::Point3D> >& outPath,
 atl::Optional<atl::Array<atl::Conditional<avl::SurfaceRidge1D> >&> outRidges = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outHeightProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL,
 atl::Optional<atl::Conditional<atl::Array<avl::Segment3D>>&> outPathSegments = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Surface to fit the path to
→	inFittingMap	const PathFittingMap&			Input fitting map
→	inRidgeScanParams	const RidgeScanParams3D&			Parameters controlling the ridge extraction process
→	inRidgeSelection	Selection::Type		SelectionBest	Selection mode of ridges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker ridges can be detected in the vicinity of stronger ridges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
→	inMaxPathInterpolationLength	Optional<int>		NIL	Maximal number of consecutive points not found
→	inMaxDeviationDelta	Optional<float>	0.0 - ∞	NIL	Maximal difference between deviations of consecutive path points
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of ridge points not found
←	outPath	Conditional<Array<Point3D> >&			Fitted path or nothing if the fitting failed
←	outRidges	Optional<Array<Conditional<SurfaceRidge1D> >&>		NIL	Found ridges
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual path points and the corresponding reference path points
←	outHeightProfiles	Optional<Array<Profile>&>		NIL	Extracted surface height profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the ridge operator response
←	outPathSegments	Optional<Conditional<Array<Segment3D>>&>		NIL	Segments of the fitted path or nothing if the fitting failed

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outRidges**, **outDeviationProfile**, **outHeightProfiles**, **outResponseProfiles**, **outPathSegments**.

Read more about Optional Outputs.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Ridge operator parameters are too low in surface ridges detector in FitPathToRidges3D.

FitPathToRidges3D_Direct

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Performs a series of 1D ridge detections in 3D and creates a path from the detected points.

Applications: Tracing of a thin line, whose rough location and shape is known beforehand.

Syntax

```
void avl::FitPathToRidges3D_Direct
(
 const avl::Surface& inSurface,
 const avl::PathFittingField& inFittingField,
 atl::Optional<const avl::CoordinateSystem2D&> inFittingFieldAlignment,
 atl::Optional<float> inScanStep,
 atl::Optional<float> inSamplingStep,
 int inScanWidth,
 const avl::InterpolationMethod::Type inSurfaceInterpolation,
 const avl::RidgeScanParams3D& inRidgeScanParams,
 avl::Selection::Type inRidgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 atl::Optional<int> inMaxPathInterpolationLength,
 atl::Optional<float> inMaxDeviationDelta,
 float inMaxIncompleteness,
 atl::Conditional<atl::Array<avl::Point3D> >& outPath,
 atl::Optional<atl::Array<atl::Conditional<avl::SurfaceRidge1D>>&> outRidges = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<avl::PathFittingField&> outAlignedFittingField = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outHeightProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL,
 atl::Optional<atl::Conditional<atl::Array<avl::Segment3D> >&> outPathSegments = atl::NIL,
 atl::Array<avl::Segment2D>& diagScanSegments,
 atl::Array<avl::Rectangle2D>& diagSamplingAreas
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Surface to fit the path to
→	inFittingField	const PathFittingField&			Path fitting field
→	inFittingFieldAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the fitting field to the position of the inspected object
→	inScanStep	Optional<float>	0.0 - ∞	5.0f	Optional implicit conversion of the input path to an equidistant one
→	inSamplingStep	Optional<float>	0.0 - ∞	NIL	Desired distance between consecutive sampling points on the scan segments; if Nil, the bigger of surface X and Y scales is chosen
→	inScanWidth	int	1 - ∞	5	The width of each scan field (in pixels)
→	inSurfaceInterpolation	const InterpolationMethod::Type		Bilinear	Interpolation method used for extraction of surface points
→	inRidgeScanParams	const RidgeScanParams3D&		RidgeScanParams3D (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f RidgeWidth: 5.0f RidgeMargin: 2.0f RidgeOperator: Minimum MinMagnitude: 5.0f RidgePolarity: Low)	Parameters controlling the ridge extraction process
→	inRidgeSelection	Selection::Type			Selection mode of ridges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker ridges can be detected in the vicinity of stronger ridges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
→	inMaxPathInterpolationLength	Optional<int>	0 - ∞	1	Maximal number of consecutive points not found
→	inMaxDeviationDelta	Optional<float>	0.0 - ∞	NIL	Maximal difference between deviations of consecutive path points
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of ridge points not found
←	outPath	Conditional<Array<Point3D> >&			Fitted path or nothing if the fitting failed
←	outRidges	Optional<Array<Conditional<SurfaceRidge1D>>&>		NIL	Found ridges
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual path points and the corresponding reference path points
←	outAlignedFittingField	Optional<PathFittingField&>		NIL	Fitting field used; in the image coordinate system
←	outHeightProfiles	Optional<Array<Profile>&>		NIL	Extracted surface height profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the ridge operator response
←	outPathSegments	Optional<Conditional<Array<Segment3D> >&>		NIL	Segments of the fitted path or nothing if the fitting failed
↔	diagScanSegments	Array<Segment2D>&			Segments along which the scans were run
↔	diagSamplingAreas	Array<Rectangle2D>&			Areas from which the input image is sampled

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outRidges**, **outDeviationProfile**, **outAlignedFittingField**, **outHeightProfiles**, **outResponseProfiles**, **outPathSegments**.

Read more about Optional Outputs.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Ridge operator parameters are too low in surface ridges detector in FitPathToRidges3D.
DomainError	Sampling step set to zero in FitPathToRidges3D.

FitPathToStripe3D

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Performs a series of 1D stripe detections and creates a path from the detected points.

Applications: Tracing of a stripe, whose rough location and shape is known beforehand.

Syntax

```
void avl::FitPathToStripe3D
(
 const avl::Surface& inSurface,
 const avl::PathFittingMap& inFittingMap,
 const StripeScanParams3D& inStripeScanParams,
 avl::Selection::Type inStripeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 atl::Optional<int> inMaxPathInterpolationLength,
 atl::Optional<float> inMaxDeviationDelta,
 float inMaxIncompleteness,
 atl::Conditional<atl::Array<avl::Point3D> >& outPath,
 atl::Conditional<atl::Array<avl::Point3D> >& outLeftPath,
 atl::Conditional<atl::Array<avl::Point3D> >& outRightPath,
 atl::Optional<atl::Array<atl::Conditional<avl::SurfaceStripe1D> >&> outStripes = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outHeightProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL,
 atl::Optional<atl::Conditional<atl::Array<avl::Segment3D>>&> outPathSegments = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Surface to fit the path to
→	inFittingMap	const PathFittingMap&			Input fitting map
→	inStripeScanParams	const StripeScanParams3D&			Parameters controlling the stripe extraction process
→	inStripeSelection	Selection::Type		Selection::Best	Selection mode of stripe
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
→	inMaxPathInterpolationLength	Optional<int>		NIL	Maximal number of consecutive points not found
→	inMaxDeviationDelta	Optional<float>	0.0 - ∞	NIL	Maximal difference between deviations of consecutive path points
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of stripe points not found
←	outPath	Conditional<Array<Point3D> >&			Fitted path or nothing if the fitting failed
←	outLeftPath	Conditional<Array<Point3D> >&			Fitted left path
←	outRightPath	Conditional<Array<Point3D> >&			Fitted right path
←	outStripes	Optional<Array<Conditional<SurfaceStripe1D> >&>		NIL	Found stripes
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual path points and the corresponding reference path points
←	outHeightProfiles	Optional<Array<Profile>&>		NIL	Extracted surface height profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the edge (derivative) operator response
←	outPathSegments	Optional<Conditional<Array<Segment3D>>&>		NIL	Segments of the fitted path or nothing if the fitting failed

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outStripes**, **outDeviationProfile**, **outHeightProfiles**, **outResponseProfiles**, **outPathSegments**.

Read more about Optional Outputs.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

FitPathToStripe3D_Direct

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Performs a series of 1D stripe detections in 3D and creates a path from the detected points.

Applications: Tracing of a stripe, whose rough location and shape is known beforehand.

Syntax

```
void avl::FitPathToStripe3D_Direct
(
 const avl::Surface& inSurface,
 const avl::PathFittingField& inFittingField,
 atl::Optional<const avl::CoordinateSystem2D&> inFittingFieldAlignment,
 atl::Optional<float> inScanStep,
 atl::Optional<float> inSamplingStep,
 int inScanWidth,
 const avl::InterpolationMethod::Type inSurfaceInterpolation,
 const avl::StripeScanParams3D& inStripeScanParams,
 avl::Selection::Type inStripeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 atl::Optional<int> inMaxPathInterpolationLength,
 atl::Optional<float> inMaxDeviationDelta,
 float inMaxIncompleteness,
 atl::Conditional<atl::Array<avl::Point3D> >& outPath,
 atl::Conditional<atl::Array<avl::Point3D> >& outLeftPath,
 atl::Conditional<atl::Array<avl::Point3D> >& outRightPath,
 atl::Optional<atl::Array<atl::Conditional<avl::SurfaceStripe1D>>&> outStripes = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<avl::PathFittingField&> outAlignedFittingField = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outHeightProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL,
 atl::Optional<atl::Conditional<atl::Array<avl::Segment3D> >&> outPathSegments = atl::NIL,
 atl::Array<avl::Segment2D>& diagScanSegments,
 atl::Array<avl::Rectangle2D>& diagSamplingAreas
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Surface to fit path to
→	inFittingField	const PathFittingField&			Path fitting field
→	inFittingFieldAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the fitting field to the position of the inspected object
→	inScanStep	Optional<float>	0.0 - ∞	5.0f	Optional implicit conversion of the input path to an equidistant one
→	inSamplingStep	Optional<float>	0.0 - ∞	NIL	Desired distance between consecutive sampling points on the scan segments; if Nil, the bigger of surface X and Y scales is chosen
→	inScanWidth	int	1 - ∞	5	The width of each scan field (in pixels)
→	inSurfaceInterpolation	const InterpolationMethod::Type		Bilinear	Interpolation method used for extraction of surface points
→	inStripeScanParams	const StripeScanParams3D&		StripeScanParams3D (ProfileInterpolation: Quadratic4 SmoothingStdDev: 1.0f MinMagnitude: 5.0f MaxInnerEdgeMagnitude: Nil MinStripeWidth: 0.0f MaxStripeWidth: Nil StripePolarity: High)	Parameters controlling the stripe extraction process
→	inStripeSelection	Selection::Type			Selection mode of stripes
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
→	inMaxPathInterpolationLength	Optional<int>	0 - ∞	1	Maximal number of consecutive points not found
→	inMaxDeviationDelta	Optional<float>	0.0 - ∞	NIL	Maximal difference between deviations of consecutive path points
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of stripe points not found
←	outPath	Conditional<Array<Point3D> >&			Fitted path or nothing if the fitting failed
←	outLeftPath	Conditional<Array<Point3D> >&			Fitted left path
←	outRightPath	Conditional<Array<Point3D> >&			Fitted right path
←	outStripes	Optional<Array<Conditional<SurfaceStripe1D>>&>		NIL	Found stripes
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual path points and the corresponding reference path points
←	outAlignedFittingField	Optional<PathFittingField&>		NIL	Fitting field used; in the image coordinate system
←	outHeightProfiles	Optional<Array<Profile>&>		NIL	Extracted surface height profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the edge (derivative) operator response
←	outPathSegments	Optional<Conditional<Array<Segment3D> >&>		NIL	Segments of the fitted path or nothing if the fitting failed
🔍	diagScanSegments	Array<Segment2D>&			Segments along which the scans were run
🔍	diagSamplingAreas	Array<Rectangle2D>&			Areas from which the input image is sampled

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outStripes**, **outDeviationProfile**, **outAlignedFittingField**, **outHeightProfiles**, **outResponseProfiles**, **outPathSegments**.

Read more about Optional Outputs.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Sampling step set to zero in FitPathToStripe3D.

FitSegmentToEdges3D

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Performs a series of 1D edge detections and finds a segment that best matches the detected points.

Applications: Precise detection of a straight edge, whose rough location is known beforehand.

Syntax

```
void avl::FitSegmentToEdges3D
(
 const avl::Surface& inSurface,
 const avl::SegmentFittingMap& inFittingMap,
 const EdgeScanParams3D& inEdgeScanParams,
 avl::Selection::Type inEdgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 float inMaxIncompleteness,
 atl::Optional<avl::MEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Segment3D>& outSegment,
 atl::Optional<atl::Array<atl::Conditional<avl::SurfaceEdge1D>>&> outEdges = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<atl::Array<avl::Point3D>&> outInliers = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outHeightProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Surface to fit segment to
→	inFittingMap	const SegmentFittingMap&			Input fitting map
→	inEdgeScanParams	const EdgeScanParams3D&			Parameters controlling the edge extraction process
→	inEdgeSelection	Selection::Type		Selection::Best	Selection mode of edges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of edge points not found
→	inOutlierSuppression	Optional<MEstimator::Type>		NIL	Selects a method for ignoring incorrectly detected points
←	outSegment	Conditional<Segment3D>&			Fitted segment or nothing if the fitting fails
←	outEdges	Optional<Array<Conditional<SurfaceEdge1D>>&>		NIL	Found edges
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual segment points and the corresponding reference segment points
←	outInliers	Optional<Array<Point3D>&>		NIL	Points matching the fitting segment
←	outHeightProfiles	Optional<Array<Profile>&>		NIL	Extracted surface height profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the edge (derivative) operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outEdges**, **outDeviationProfile**, **outInliers**, **outHeightProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

FitSegmentToEdges3D_Direct

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Performs a series of 1D edge detections in 3D and finds a segment that best matches the detected points.

Applications: Precise detection of a straight edge, whose rough location is known beforehand.

Syntax

```
void avl::FitSegmentToEdges3D_Direct
(
 const avl::Surface& inSurface,
 const avl::SegmentFittingField& inFittingField,
 atl::Optional<const avl::CoordinateSystem2D&> inFittingFieldAlignment,
 int inScanCount,
 atl::Optional<float> inSamplingStep,
 int inScanWidth,
 const avl::InterpolationMethod::Type inSurfaceInterpolation,
 const avl::EdgeScanParams3D& inEdgeScanParams,
 avl::Selection::Type inEdgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 float inMaxIncompleteness,
 atl::Optional<avl::MEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Segment3D>& outSegment,
 atl::Optional<atl::Array<atl::Conditional<avl::SurfaceEdge1D>>&> outEdges = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<avl::SegmentFittingField&> outAlignedFittingField = atl::NIL,
 atl::Optional<atl::Array<avl::Point3D>&> outInliers = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outHeightProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL,
 atl::Array<avl::Segment2D>& diagScanSegments,
 atl::Array<avl::Rectangle2D>& diagSamplingAreas
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Surface to fit segment to
→	inFittingField	const SegmentFittingField&			Segment fitting field
→	inFittingFieldAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the fitting field to the position of the inspected object
→	inScanCount	int	3 - ∞	10	The number of points that will be searched to estimate the position of the segment
→	inSamplingStep	Optional<float>	0.0 - ∞	NIL	Desired distance between consecutive sampling points on the scan segments; if Nil, the bigger of surface X and Y scales is chosen
→	inScanWidth	int	1 - ∞	5	The width of each scan field (in pixels)
→	inSurfaceInterpolation	const InterpolationMethod::Type		Bilinear	Interpolation method used for extraction of surface points
→	inEdgeScanParams	const EdgeScanParams3D&		EdgeScanParams3D (ProfileInterpolation: Quadratic4 SmoothingStdDev: 1.0f MinMagnitude: 5.0f EdgeTransition: LowToHigh)	Parameters controlling the edge extraction process
→	inEdgeSelection	Selection::Type			Selection mode of edges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of edge points not found
→	inOutlierSuppression	Optional<MEstimator::Type>		NIL	Selects a method for ignoring incorrectly detected points
←	outSegment	Conditional<Segment3D>&			Fitted segment or nothing if the fitting fails
←	outEdges	Optional<Array<Conditional<SurfaceEdge1D>>&>		NIL	Found edges
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual segment points and the corresponding reference segment points
←	outAlignedFittingField	Optional<SegmentFittingField&>		NIL	Fitting field used; in the image coordinate system
←	outInliers	Optional<Array<Point3D>&>		NIL	Points matching the fitting segment
←	outHeightProfiles	Optional<Array<Profile>&>		NIL	Extracted surface height profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the edge (derivative) operator response
🔍	diagScanSegments	Array<Segment2D>&			Segments along which the scans were run
🔍	diagSamplingAreas	Array<Rectangle2D>&			Areas from which the input image is sampled

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outEdges**, **outDeviationProfile**, **outAlignedFittingField**, **outInliers**, **outHeightProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Sampling step set to zero in FitSegmentToEdges3D.

FitSegmentToRidges3D

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Performs a series of 1D ridge detections and finds a segment that best matches the detected points.

Applications: Precise detection of a thin straight line, whose rough location is known beforehand.

Syntax

```
void avl::FitSegmentToRidges3D
(
 const avl::Surface& inSurface,
 const SegmentFittingMap& inFittingMap,
 const RidgeScanParams3D& inRidgeScanParams,
 avl::Selection::Type inRidgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 float inMaxIncompleteness,
 atl::Optional<avl::MEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Segment3D>& outSegment,
 atl::Optional<atl::Array<atl::Conditional<avl::SurfaceRidge1D>>&> outRidges = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<atl::Array<avl::Point3D>&> outInliers = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outHeightProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Surface to fit the segment to
→	inFittingMap	const SegmentFittingMap&			Input fitting map
→	inRidgeScanParams	const RidgeScanParams3D&			Parameters controlling the ridge extraction process
→	inRidgeSelection	Selection::Type		SelectionBest	Selection mode of ridges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker ridges can be detected in the vicinity of stronger ridges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of ridge points not found
→	inOutlierSuppression	Optional<MEstimator::Type>		NIL	Selects a method for ignoring incorrectly detected points
←	outSegment	Conditional<Segment3D>&			Fitted segment or nothing if the fitting fails
←	outRidges	Optional<Array<Conditional<SurfaceRidge1D>>&>		NIL	Found ridges
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual segment points and the corresponding reference segment points
←	outInliers	Optional<Array<Point3D>&>		NIL	Points matching the fitting segment
←	outHeightProfiles	Optional<Array<Profile>&>		NIL	Extracted surface height profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the ridge operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outRidges**, **outDeviationProfile**, **outInliers**, **outHeightProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Ridge operator parameters are too low in surface ridges detector in FitSegmentToRidges3D.

FitSegmentToRidges3D_Direct

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Performs a series of 1D ridge detections in 3D and finds a segment that best matches the detected points.

Applications: Precise detection of a thin straight line, whose rough location is known beforehand.

Syntax

```
void avl::FitSegmentToRidges3D_Direct
(
 const avl::Surface& inSurface,
 const avl::SegmentFittingField& inFittingField,
 atl::Optional<const avl::CoordinateSystem2D&> inFittingFieldAlignment,
 int inScanCount,
 atl::Optional<float> inSamplingStep,
 int inScanWidth,
 const avl::InterpolationMethod::Type inSurfaceInterpolation,
 const avl::RidgeScanParams3D& inRidgeScanParams,
 avl::Selection::Type inRidgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 float inMaxIncompleteness,
 atl::Optional<avl::MEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Segment3D>& outSegment,
 atl::Optional<atl::Array<atl::Conditional<avl::SurfaceRidge1D>>&> outRidges = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<avl::SegmentFittingField&> outAlignedFittingField = atl::NIL,
 atl::Optional<atl::Array<avl::Point3D>&> outInliers = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outHeightProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL,
 atl::Array<avl::Segment2D>& diagScanSegments,
 atl::Array<avl::Rectangle2D>& diagSamplingAreas
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Surface to fit the segment to
→	inFittingField	const SegmentFittingField&			Segment fitting field
→	inFittingFieldAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the fitting field to the position of the inspected object
→	inScanCount	int	3 - ∞	10	The number of points that will be searched to estimate the position of the segment
→	inSamplingStep	Optional<float>	0.0 - ∞	NIL	Desired distance between consecutive sampling points on the scan segments; if Nil, the bigger of surface X and Y scales is chosen
→	inScanWidth	int	1 - ∞	5	The width of each scan field (in pixels)
→	inSurfaceInterpolation	const InterpolationMethod::Type		Bilinear	Interpolation method used for extraction of surface points
→	inRidgeScanParams	const RidgeScanParams3D&		RidgeScanParams3D (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f RidgeWidth: 5.0f RidgeMargin: 2.0f RidgeOperator: Minimum MinMagnitude: 5.0f RidgePolarity: Low)	Parameters controlling the ridge extraction process
→	inRidgeSelection	Selection::Type			Selection mode of ridges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker ridges can be detected in the vicinity of stronger ridges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of ridge points not found
→	inOutlierSuppression	Optional<MEstimator::Type>		NIL	Selects a method for ignoring incorrectly detected points
←	outSegment	Conditional<Segment3D>&			Fitted segment or nothing if the fitting fails
←	outRidges	Optional<Array<Conditional<SurfaceRidge1D>>&>		NIL	Found ridges
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual segment points and the corresponding reference segment points
←	outAlignedFittingField	Optional<SegmentFittingField&>		NIL	Fitting field used; in the image coordinate system
←	outInliers	Optional<Array<Point3D>&>		NIL	Points matching the fitting segment
←	outHeightProfiles	Optional<Array<Profile>&>		NIL	Extracted surface height profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the ridge operator response
🔍	diagScanSegments	Array<Segment2D>&			Segments along which the scans were run
🔍	diagSamplingAreas	Array<Rectangle2D>&			Areas from which the input image is sampled

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outRidges**, **outDeviationProfile**, **outAlignedFittingField**, **outInliers**, **outHeightProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Ridge operator parameters are too low in surface ridges detector in FitSegmentToRidges3D.
DomainError	Sampling step set to zero in FitSegmentToRidges3D.

FitSegmentToStripe3D

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Performs a series of 1D edge detections and finds a segment that best matches the detected points.

Applications: Precise detection of a straight edge, whose rough location is known beforehand.

Syntax

```
void avl::FitSegmentToStripe3D
(
 const avl::Surface& inSurface,
 const avl::SegmentFittingMap& inFittingMap,
 const StripeScanParams3D& inStripeScanParams,
 avl::Selection::Type inStripeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 float inMaxIncompleteness,
 atl::Optional<avl::MEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Segment3D>& outSegment,
 atl::Conditional<avl::Segment3D>& outLeftSegment,
 atl::Conditional<avl::Segment3D>& outRightSegment,
 atl::Optional<atl::Array<atl::Conditional<avl::SurfaceStripe1D>>&> outStripes = atl::NIL,
 atl::Optional<atl::Array<avl::Point3D>&> outStripePoints = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outHeightProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Surface to fit segment to
→	inFittingMap	const SegmentFittingMap&			Input fitting map
→	inStripeScanParams	const StripeScanParams3D&			Parameters controlling the stripe extraction process
→	inStripeSelection	Selection::Type		Selection::Best	Selection mode of stripe
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of stripe points not found
→	inOutlierSuppression	Optional<MEstimator::Type>		NIL	Selects a method for ignoring incorrectly detected points
←	outSegment	Conditional<Segment3D>&			Fitted segment or nothing if the fitting fails
←	outLeftSegment	Conditional<Segment3D>&			Fitted left segment
←	outRightSegment	Conditional<Segment3D>&			Fitted right segment
←	outStripes	Optional<Array<Conditional<SurfaceStripe1D>>&>		NIL	Found stripes
←	outStripePoints	Optional<Array<Point3D>&>		NIL	Extracted points of middle segment of a surface stripe
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual segment points and the corresponding reference segment points
←	outHeightProfiles	Optional<Array<Profile>&>		NIL	Extracted surface height profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the edge (derivative) operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outStripes**, **outStripePoints**, **outDeviationProfile**, **outHeightProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

FitSegmentToStripe3D_Direct

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Performs a series of 1D edge detections in 3D and finds a segment that best matches the detected points.

Applications: Precise detection of a straight edge, whose rough location is known beforehand.

Syntax

```
void avl::FitSegmentToStripe3D_Direct
(
 const avl::Surface& inSurface,
 const avl::SegmentFittingField& inFittingField,
 atl::Optional<const avl::CoordinateSystem2D&> inFittingFieldAlignment,
 int inScanCount,
 atl::Optional<float> inSamplingStep,
 int inScanWidth,
 const avl::InterpolationMethod::Type inSurfaceInterpolation,
 const avl::StripeScanParams3D& inStripeScanParams,
 avl::Selection::Type inStripeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 float inMaxIncompleteness,
 atl::Optional<avl::MEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Segment3D>& outSegment,
 atl::Conditional<avl::Segment3D>& outLeftSegment,
 atl::Conditional<avl::Segment3D>& outRightSegment,
 atl::Optional<atl::Array<atl::Conditional<avl::SurfaceStripe1D>>&> outStripes = atl::NIL,
 atl::Optional<atl::Array<avl::Point3D>&> outStripePoints = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<avl::SegmentFittingField&> outAlignedFittingField = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outHeightProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL,
 atl::Array<avl::Segment2D>& diagScanSegments,
 atl::Array<avl::Rectangle2D>& diagSamplingAreas
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Surface to fit segment to
→	inFittingField	const SegmentFittingField&			Segment fitting field
→	inFittingFieldAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the fitting field to the position of the inspected object
→	inScanCount	int	3 - ∞	10	The number of points that will be searched to estimate the position of the segment
→	inSamplingStep	Optional<float>	0.0 - ∞	NIL	Desired distance between consecutive sampling points on the scan segments; if Nil, the bigger of surface X and Y scales is chosen
→	inScanWidth	int	1 - ∞	5	The width of each scan field (in pixels)
→	inSurfaceInterpolation	const InterpolationMethod::Type		Bilinear	Interpolation method used for extraction of surface points
→	inStripeScanParams	const StripeScanParams3D&		StripeScanParams3D (ProfileInterpolation: Quadratic4 SmoothingStdDev: 1.0f MinMagnitude: 5.0f MaxInnerEdgeMagnitude: Nil MinStripeWidth: 0.0f MaxStripeWidth: Nil StripePolarity: High)	Parameters controlling the stripe extraction process
→	inStripeSelection	Selection::Type			Selection mode of stripe
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of stripe points not found
→	inOutlierSuppression	Optional<MEstimator::Type>		NIL	Selects a method for ignoring incorrectly detected points
←	outSegment	Conditional<Segment3D>&			Fitted segment or nothing if the fitting fails
←	outLeftSegment	Conditional<Segment3D>&			Fitted left segment
←	outRightSegment	Conditional<Segment3D>&			Fitted right segment
←	outStripes	Optional<Array<Conditional<SurfaceStripe1D>>&>		NIL	Found stripes
←	outStripePoints	Optional<Array<Point3D>&>		NIL	Extracted points of middle segment of a surface stripe
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual segment points and the corresponding reference segment points
←	outAlignedFittingField	Optional<SegmentFittingField&>		NIL	Fitting field used; in the image coordinate system
←	outHeightProfiles	Optional<Array<Profile>&>		NIL	Extracted surface height profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the edge (derivative) operator response
🔍	diagScanSegments	Array<Segment2D>&			Segments along which the scans were run
🔍	diagSamplingAreas	Array<Rectangle2D>&			Areas from which the input image is sampled

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outStripes**, **outStripePoints**, **outDeviationProfile**, **outAlignedFittingField**, **outHeightProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Sampling step set to zero in FitSegmentToStripe3D.

MeasureObjectWidth3D

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Measures the width of an object using stripe detection.

Applications: Easy and precise measurement of distances between to straight parallel edges.

Syntax

```
void avl::MeasureObjectWidth3D
(
 const avl::Surface& inSurface,
 const atl::Array<avl::ScanMap>& inMeasurementMap,
 const avl::StripeScanParams3D& inStripeScanParams,
 avl::MeasureObjectMethod::Type inMeasureMethod,
 avl::Selection::Type inStripeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 atl::Optional<avl::MEstimator::Type> inOutlierSuppression,
 int inOutlierCount,
 atl::Conditional<float>& outObjectWidth,
 atl::Conditional<avl::Segment2D>& outSegment1,
 atl::Conditional<avl::Segment2D>& outSegment2,
 atl::Optional<atl::Array<atl::Conditional<avl::SurfaceStripe1D>>&> outStripes = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outHeightProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inMeasurementMap	const Array<ScanMap>&			Input measurement map
→	inStripeScanParams	const StripeScanParams3D&			Parameters controlling the object stripe extraction process
→	inMeasureMethod	MeasureObjectMethod::Type			Method used to measure the object
→	inStripeSelection	Selection::Type		Selection::Best	Selection mode of edges of the object
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
→	inOutlierSuppression	Optional<MEstimator::Type>		NIL	Selects a method for ignoring incorrectly detected points
→	inOutlierCount	int	0 - ∞		Determines how many outlying points are rejected before the width is measured
←	outObjectWidth	Conditional<float>&			Width of the object
←	outSegment1	Conditional<Segment2D>&			First edge of the object
←	outSegment2	Conditional<Segment2D>&			Second edge of the object
←	outStripes	Optional<Array<Conditional<SurfaceStripe1D>>&>		NIL	Detected stripes
←	outHeightProfiles	Optional<Array<Profile>&>		NIL	Extracted surface height profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the edge (derivative) operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outStripes**, **outHeightProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Invalid MeasurementMap in MeasureObjectWidth3D function. Use CreateSurfaceMeasurementMap function to create it properly.

51. Image Color Spaces

Table of content:

BayerToRgb

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts a Bayer-encoded color image into RGB color space.

Applications: Use this filter only if the conversion functionality is not provided by the camera (check appropriate pixel types).

Syntax

```
void avl::BayerToRgb
(
 const avl::Image& inMonoImage,
 avl::DebayeringMethod::Type inDebayeringMethod,
 avl::BayerType::Type inBayerType,
 avl::Image& outRgbImage
)
```

Parameters

	Name	Type	Default	Description
→	inMonoImage	const Image&		
→	inDebayeringMethod	DebayeringMethod::Type		
→	inBayerType	BayerType::Type		
←	outRgbImage	Image&		

Requirements

For input **inMonoImage** only pixel formats are supported: 1□uint8, 1□int8, 1□uint16, 1□int16, 1□int32, 1□real.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

This operation is optimized for SSSE3 technology for pixels of types: UINT8(for inDebayeringMethod=Bilinear).

This operation is optimized for NEON technology for pixels of types: UINT8(for inDebayeringMethod=Bilinear).

Errors

List of possible exceptions:

Error type	Description
DomainError	Not a 1-channel image at input of BayerToRgb.
DomainError	Unsupported inBayerAlgorithm in BayerToRgb.
DomainError	Not supported inMonoImage pixel format in BayerToRgb. Supported formats: 1xUInt8, 1xInt8, 1xUInt16, 1xInt16, 1xInt32, 1xReal.

CmykToRgb

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts color space from Cyan-Magenta-Yellow-Key into Red-Green-Blue.

Applications: CMYK color space is designed for printing - not recommended for machine vision.

Syntax

```
void avl::CmykToRgb
(
 const avl::Image& inCmykImage,
 avl::Image& outRgbImage
)
```

Parameters

	Name	Type	Default	Description
→	inCmykImage	const Image&		
←	outRgbImage	Image&		

Requirements

For input **inCmykImage** only pixel formats are supported: 4□uint8, 4□int8, 4□uint16, 4□int16, 4□int32, 4□real.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not a 4-channel image in CmykToRgb.
DomainError	Not supported inCmykImage pixel format in CmykToRgb. Supported formats: 4xUInt8, 4xInt8, 4xUInt16, 4xInt16, 4xInt32, 4xReal.

HsiToRgb

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts color space from Hue-Saturation-Intensity into Red-Green-Blue.

Syntax

```
void avl::HsiToRgb
(
 const avl::Image& inHsiImage,
 avl::Image& outRgbImage
)
```

Parameters

	Name	Type	Default	Description
→	inHsiImage	const Image&		
←	outRgbImage	Image&		

Requirements

For input **inHsiImage** only pixel formats are supported: 3□uint8.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

This operation supports processing on OpenCL compatible device.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not a 3-channel and 8-bit image in HsiToRgb.
DomainError	Not supported inHsiImage pixel format in HsiToRgb. Supported formats: 3xUInt8.

HslToRgb

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts color space from Hue-Saturation-Luminance to Red-Green-Blue.

Syntax

```
void avl::HslToRgb
(
 const avl::Image& inHslImage,
 avl::Image& outRgbImage
)
```

Parameters

	Name	Type	Default	Description
→	inHslImage	const Image&		
←	outRgbImage	Image&		

Requirements

For input **inHslImage** only pixel formats are supported: 3□uint8.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

This operation supports processing on OpenCL compatible device.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not a 3-channel and 8-bit image in HslToRgb.
DomainError	Not supported inHslImage pixel format in HslToRgb. Supported formats: 3xUInt8.

HsvToRgb

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts color space from Hue-Saturation-Value to Red-Green-Blue.

Syntax

```
void avl::HsvToRgb
(
 const avl::Image& inHsvImage,
 avl::Image& outRgbImage
)
```

Parameters

	Name	Type	Default	Description
→	inHsvImage	const Image&		
←	outRgbImage	Image&		

Requirements

For input **inHsvImage** only pixel formats are supported: 3□uint8.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

This operation supports processing on OpenCL compatible device.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not a 3-channel and 8-bit image in HsvToRgb.
DomainError	Not supported inHsvImage pixel format in HsvToRgb. Supported formats: 3xUInt8.

LabToRgb

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts color space from L*a*b* to Red-Green-Blue.

Syntax

```
void avl::LabToRgb
(
 const avl::Image& inLabImage,
 avl::Image& outRgbImage
)
```

Parameters

	Name	Type	Default	Description
→	inLabImage	const Image&		
←	outRgbImage	Image&		

Requirements

For input **inLabImage** only pixel formats are supported: 3□real.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not a 3-channel image in LabToRgb.
DomainError	Not supported inLabImage pixel format in LabToRgb. Supported formats: 3xReal.

Rgb555ToRgb888

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts a RGB555 image into a RGB888 one.

Applications: Decompression of pixel depth from 2 to 3 bytes.

Syntax

```
void avl::Rgb555ToRgb888
(
 const avl::Image& inRgb555Image,
 avl::Image& outRgb888Image
)
```

Parameters

	Name	Type	Default	Description
→	inRgb555Image	const Image&		
←	outRgb888Image	Image&		

Requirements

For input **inRgb555Image** only pixel formats are supported: 1□uint16.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not a 1-channel and 16-bit image in Rgb555ToRgb888.
DomainError	Not supported inRgb555Image pixel format in Rgb555ToRgb888. Supported formats: 1xUInt16.

Rgb565ToRgb888

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts a RGB565 image into a RGB888 one.

Applications: Decompression of pixel depth from 2 to 3 bytes.

Syntax

```
void avl::Rgb565ToRgb888
(
 const avl::Image& inRgb565Image,
 avl::Image& outRgb888Image
)
```

Parameters

	Name	Type	Default	Description
→	inRgb565Image	const Image&		
←	outRgb888Image	Image&		

Requirements

For input **inRgb565Image** only pixel formats are supported: 1□uint16.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not a 1-channel and 16-bit image in Rgb565ToRgb888.
DomainError	Not supported inRgb565Image pixel format in Rgb565ToRgb888. Supported formats: 1xUInt16.

Rgb888ToRgb555

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts a RGB888 image into a RGB555 one.

Applications: Compression of pixel depth from 3 to 2 bytes.

Syntax

```
void avl::Rgb888ToRgb555
(
 const avl::Image& inRgb888Image,
 avl::Image& outRgb555Image
)
```

Parameters

	Name	Type	Default	Description
→	inRgb888Image	const Image&		
←	outRgb555Image	Image&		

Requirements

For input **inRgb888Image** only pixel formats are supported: 3□uint8.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not a 3-channel and 8-bit image in Rgb888ToRgb555.
DomainError	Not supported inRgb888Image pixel format in Rgb888ToRgb555. Supported formats: 3xUInt8.

Rgb888ToRgb565

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts a RGB888 image into a RGB565 one.

Applications: Compression of pixel depth from 3 to 2 bytes.

Syntax

```
void avl::Rgb888ToRgb565
(
 const avl::Image& inRgb888Image,
 avl::Image& outRgb565Image
)
```

Parameters

	Name	Type	Default	Description
→	inRgb888Image	const Image&		
←	outRgb565Image	Image&		

Requirements

For input **inRgb888Image** only pixel formats are supported: 3□uint8.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not a 3-channel and 8-bit image in Rgb888ToRgb565.
DomainError	Not supported inRgb888Image pixel format in Rgb888ToRgb565. Supported formats: 3xUInt8.

RgbToBayer

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts a RGB color image into Bayer-encoded color image.

Syntax

```
void avl::RgbToBayer
(
 const avl::Image& inRgbImage,
 avl::BayerType::Type inBayerType,
 avl::Image& outBayerImage
)
```

Parameters

	Name	Type	Default	Description
→	inRgbImage	const Image&		
→	inBayerType	BayerType::Type		
←	outBayerImage	Image&		

Requirements

For input **inRgbImage** only pixel formats are supported: 3□uint8, 3□int8, 3□uint16, 3□int16, 3□int32, 3□real.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not a 3-channel image at input of RgbToBayer.
DomainError	Not supported bayer type value in RgbToBayer.
DomainError	Not supported inRgbImage pixel format in RgbToBayer. Supported formats: 3xUInt8, 3xInt8, 3xUInt16, 3xInt16, 3xInt32, 3xReal.

RgbToCmyk

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts color space from Red-Green-Blue into Cyan-Magenta-Yellow-Key.

Applications: CMYK color space is designed for printing - not recommended for machine vision.

Syntax

```
void avl::RgbToCmyk
(
 const avl::Image& inRgbImage,
 avl::Image& outCmykImage
)
```

Parameters

	Name	Type	Default	Description
→	inRgbImage	const Image&		
←	outCmykImage	Image&		

Requirements

For input **inRgbImage** only pixel formats are supported: 3□uint8, 3□int8, 3□uint16, 3□int16, 3□int32, 3□real.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not a 3-channel image in RgbToCmyk.
DomainError	Not supported inRgbImage pixel format in RgbToCmyk. Supported formats: 3xUInt8, 3xInt8, 3xUInt16, 3xInt16, 3xInt32, 3xReal.

RgbToHsi

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts color space from Red-Green-Blue into Hue-Saturation-Intensity.

Applications: Color analysis is easier in the HSI color space than in RGB.

Syntax

```
void avl::RgbToHsi
(
 const avl::Image& inRgbImage,
 avl::Image& outHsiImage
)
```

Parameters

	Name	Type	Default	Description
→	inRgbImage	const Image&		
←	outHsiImage	Image&		

Requirements

For input **inRgbImage** only pixel formats are supported: 3□uint8.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

This operation supports processing on OpenCL compatible device.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not a 3-channel and 8-bit image in RgbToHsi.
DomainError	Not supported inRgbImage pixel format in RgbToHsi. Supported formats: 3xUInt8.

RgbToHsl

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts color space from Red-Green-Blue to Hue-Saturation-Luminance.

Applications: Color analysis is easier in the HSL color space than in RGB.

Syntax

```
void avl::RgbToHsl
(
 const avl::Image& inRgbImage,
 avl::Image& outHslImage
)
```

Parameters

	Name	Type	Default	Description
→	inRgbImage	const Image&		
←	outHslImage	Image&		

Requirements

For input **inRgbImage** only pixel formats are supported: 3□uint8.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

This operation supports processing on OpenCL compatible device.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not a 3-channel and 8-bit image in RgbToHsl.
DomainError	Not supported inRgbImage pixel format in RgbToHsl. Supported formats: 3xUInt8.

RgbToHsv

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts color space from Red-Green-Blue to Hue-Saturation-Value.

Applications: Color analysis is easier in the HSV color space than in RGB.

Syntax

```
void avl::RgbToHsv
(
 const avl::Image& inRgbImage,
 avl::Image& outHsvImage
)
```

Parameters

	Name	Type	Default	Description
→	inRgbImage	const Image&		
←	outHsvImage	Image&		

Requirements

For input **inRgbImage** only pixel formats are supported: 3□uint8.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation is optimized for AVX2 technology for pixels of type: UINT8.

This operation is optimized for SSE4 technology for pixels of type: UINT8.

This operation is optimized for NEON technology for pixels of type: UINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

This operation supports processing on OpenCL compatible device.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not a 3-channel and 8-bit image in RgbToHsv.
DomainError	Not supported inRgbImage pixel format in RgbToHsv. Supported formats: 3xUInt8.

RgbToLab

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts color space from Red-Green-Blue to L*a*b*.

Syntax

```
void avl::RgbToLab
(
 const avl::Image& inRgbImage,
 avl::Image& outLabImage
)
```

Parameters

	Name	Type	Default	Description
→	inRgbImage	const Image&		
←	outLabImage	Image&		

Requirements

For input **inRgbImage** only pixel formats are supported: 3□uint8, 3□int8, 3□uint16, 3□int16, 3□int32, 3□real.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not a 3-channel image in RgbToLab.
DomainError	Not supported inRgbImage pixel format in RgbToLab. Supported formats: 3xUInt8, 3xInt8, 3xUInt16, 3xInt16, 3xInt32, 3xReal.

RgbToXyz

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts color space from Red-Green-Blue to XYZ.

Applications: XYZ color space is designed for human perception - not recommended for machine vision.

Syntax

```
void avl::RgbToXyz
(
 const avl::Image& inRgbImage,
 avl::Image& outXyzImage
)
```

Parameters

	Name	Type	Default	Description
→	inRgbImage	const Image&		
←	outXyzImage	Image&		

Requirements

For input **inRgbImage** only pixel formats are supported: 3□uint8, 3□int8, 3□uint16, 3□int16, 3□int32, 3□real.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

This operation supports processing on OpenCL compatible device (when pixel type is uint8).

Errors

List of possible exceptions:

Error type	Description
DomainError	Not a 3-channel image in RgbToXyz.
DomainError	Not supported inRgbImage pixel format in RgbToXyz. Supported formats: 3xUInt8, 3xInt8, 3xUInt16, 3xInt16, 3xInt32, 3xReal.

RgbToYCoCg

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts color space from Red-Green-Blue into pseudointensitY-Orange-Green.

Syntax

```
void avl::RgbToYCoCg
(
 const avl::Image& inRgbImage,
 avl::Image& outYCoCgImage
)
```

Parameters

	Name	Type	Default	Description
→	inRgbImage	const Image&		
←	outYCoCgImage	Image&		

Requirements

For input **inRgbImage** only pixel formats are supported: 3□uint8, 3□int8, 3□uint16, 3□int16, 3□int32, 3□real.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not a 3-channel image in RgbToYCoCg.
DomainError	Not supported inRgbImage pixel format in RgbToYCoCg. Supported formats: 3xUInt8, 3xInt8, 3xUInt16, 3xInt16, 3xInt32, 3xReal.

RgbToYuv

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts color space from Red-Green-Blue into YUV.

Syntax

```
void avl::RgbToYuv
(
 const avl::Image& inRgbImage,
 avl::Image& outYuvImage
)
```

Parameters

	Name	Type	Default	Description
→	inRgbImage	const Image&		
←	outYuvImage	Image&		

Requirements

For input **inRgbImage** only pixel formats are supported: 3□uint8.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not a 3-channel and 8 bit image in RgbToYuv.
DomainError	Not supported inRgbImage pixel format in RgbToYuv. Supported formats: 3xUInt8.

SplitBayerImage

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Creates several monochromatic images from individual pixels of the input Bayer pattern image.

Syntax

```
void avl::SplitBayerImage
(
 const avl::Image& inBayerImage,
 avl::Image& outImage1,
 avl::Image& outImage2,
 avl::Image& outImage3,
 avl::Image& outImage4
)
```

Parameters

	Name	Type	Default	Description
→	inBayerImage	const Image&		
←	outImage1	Image&		First output image
←	outImage2	Image&		Second output image
←	outImage3	Image&		Third output image
←	outImage4	Image&		Fourth output image

Requirements

For input **inBayerImage** only pixel formats are supported: 1□uint8, 1□int8, 1□uint16, 1□int16, 1□int32, 1□real.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Input image must have even dimensions in SplitBayerImage.
DomainError	Only one channel input image is allowed in SplitBayerImage.
DomainError	Not supported inBayerImage pixel format in SplitBayerImage. Supported formats: 1xUInt8, 1xInt8, 1xUInt16, 1xInt16, 1xInt32, 1xReal.

XyzToRgb

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts color space from XYZ to Red-Green-Blue.

Applications: XYZ color space is designed for human perception - not recommended for machine vision.

Syntax

```
void avl::XyzToRgb
(
 const avl::Image& inXyzImage,
 avl::Image& outRgbImage
)
```

Parameters

	Name	Type	Default	Description
→	inXyzImage	const Image&		
←	outRgbImage	Image&		

Requirements

For input **inXyzImage** only pixel formats are supported: 3□uint8, 3□int8, 3□uint16, 3□int16, 3□int32, 3□real.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

This operation supports processing on OpenCL compatible device (when pixel type is uint8).

Errors

List of possible exceptions:

Error type	Description
DomainError	Not a 3-channel image in XyzToRgb.
DomainError	Not supported inXyzImage pixel format in XyzToRgb. Supported formats: 3xUInt8, 3xInt8, 3xUInt16, 3xInt16, 3xInt32, 3xReal.

YCoCgToRgb

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts color space from pseudointensitY-Orange-Green into Red-Green-Blue.

Syntax

```
void avl::YCoCgToRgb
(
 const avl::Image& inYCoCgImage,
 avl::Image& outRgbImage
)
```

Parameters

	Name	Type	Default	Description
→	inYCoCgImage	const Image&		
←	outRgbImage	Image&		

Requirements

For input **inYCoCgImage** only pixel formats are supported: 3□uint8, 3□int8, 3□uint16, 3□int16, 3□int32, 3□real.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not a 3-channel image in YCoCgToRgb.
DomainError	Not supported inYCoCgImage pixel format in YCoCgToRgb. Supported formats: 3xUInt8, 3xInt8, 3xUInt16, 3xInt16, 3xInt32, 3xReal.

Yuv442ToRgb

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts a YUV-encoded image into RGB color space.

Syntax

```
void avl::Yuv442ToRgb
(
 const avl::Image& inYuvImage,
 avl::Image& outRgbImage
)
```

Parameters

	Name	Type	Default	Description
→	inYuvImage	const Image&		
←	outRgbImage	Image&		

Requirements

For input **inYuvImage** only pixel formats are supported: 3□uint8.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not a 3-channel and 8-bit image in Yuv442ToRgb.
DomainError	Not an even width of a YUV image in Yuv442ToRgb.
DomainError	Not supported inYuvImage pixel format in Yuv442ToRgb. Supported formats: 3xUInt8.

YuvToRgb

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts color space from YUV into Red-Green-Blue.

Syntax

```
void avl::YuvToRgb
(
 const avl::Image& inYuvImage,
 avl::Image& outRgbImage
)
```

Parameters

	Name	Type	Default	Description
→	inYuvImage	const Image&		
←	outRgbImage	Image&		

Requirements

For input **inYuvImage** only pixel formats are supported: 3□uint8.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not a 3-channel and 8 bit image in YuvToRgb.
DomainError	Not supported inYuvImage pixel format in YuvToRgb. Supported formats: 3xUInt8.

52. Image Local Transforms

Table of content:

BottomHatImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Performs a morphological black top hat (bottom hat) operation on a image using a predefined kernel.

Syntax

```
void avl::BottomHatImage
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<const avl::Region&> inSourceRoi,
 atl::Optional<avl::Pixel> inBorderColor,
 avl::KernelShape::Type inKernel,
 int inRadiusX,
 atl::Optional<int> inRadiusY,
 avl::Image& outImage,
 avl::Region& diagKernel
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Range of outImage pixels to be computed
→	inSourceRoi	Optional<const Region&>		NIL	Range of inImage pixels to be considered in computations
→	inBorderColor	Optional<Pixel>		NIL	Color of the imaginary pixels outside the image boundaries
→	inKernel	KernelShape::Type			Selects kernel shape
→	inRadiusX	int	0 - ∞	1	Nearly half of the kernel's width (2*R+1)
→	inRadiusY	Optional<int>	0 - ∞	NIL	Nearly half of the kernel's height
←	outImage	Image&			Output image
🔍	diagKernel	Region&			Kernel shape

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

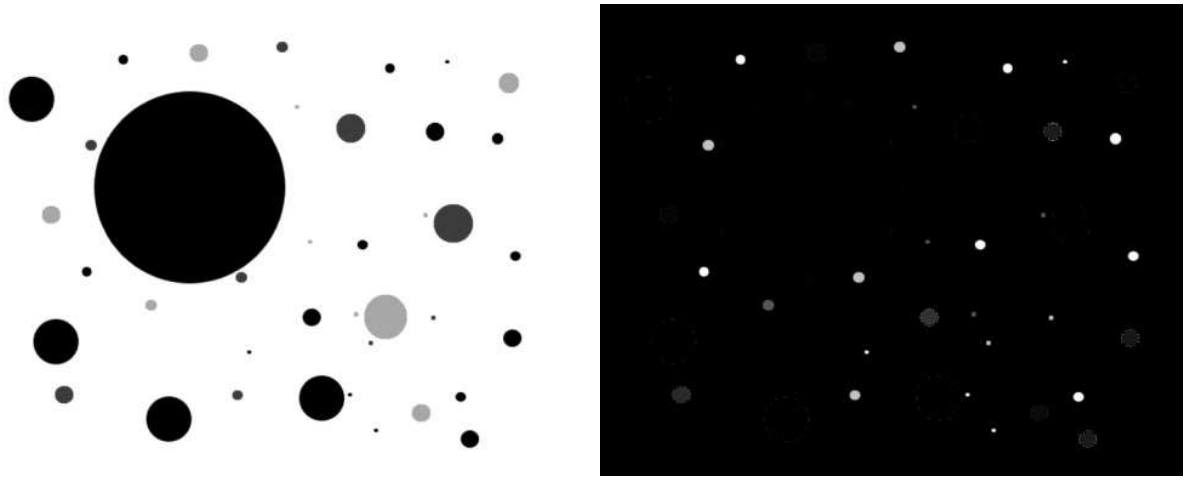
Read more about In-place Computation.

Description

Extracts from image small objects that are darker than surroundings.

Is performed by running consecutively two filters. CloseImage to get the image without small objects and SubtractFromImage to remove everything but them.

Examples



*Bottom Hat used to remove bigger objects. Used parameters **inKernel**=Ellipse and **inRadiusX**=6. Source image on the left and result on the right.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation is optimized for NEON technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in BottomHatImage.

BottomHatImage_AnyKernel

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Performs a morphological black top hat (bottom hat) operation on a image using an arbitrary kernel.

Syntax

```
void avl::BottomHatImage_AnyKernel
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<const avl::Region&> inSourceRoi,
 atl::Optional<avl::Pixel> inBorderColor,
 const avl::Region& inKernel,
 atl::Optional<const avl::Location&> inKernelAnchor,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of outImage pixels to be computed
→	inSourceRoi	Optional<const Region&>	NIL	Range of inImage pixels to be considered in computations
→	inBorderColor	Optional<Pixel>	NIL	Color of the imaginary pixels outside the image boundaries
→	inKernel	const Region&		Selects kernel shape
→	inKernelAnchor	Optional<const Location&>	NIL	A location within inKernel, defining its center
←	outImage	Image&		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

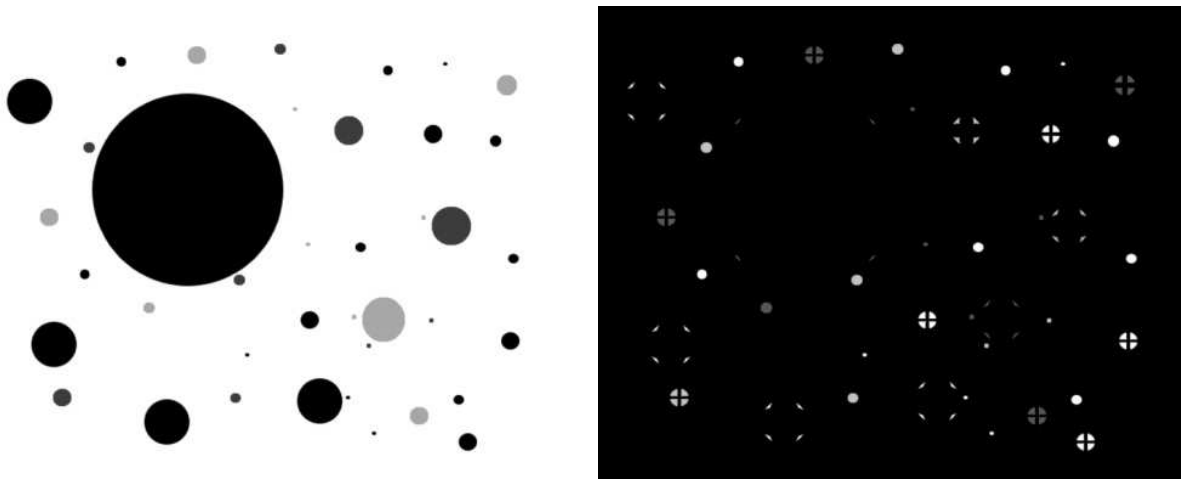
Read more about In-place Computation.

Description

Extracts from image small objects that are darker than surroundings. Uses user-defined kernel.

Is performed by running consecutively two filters. CloseImage_AnyKernel to get the image without small objects and SubtractFromImage to remove everything but them.

Examples



Bottom Hat used to remove bigger objects. Source image on the left and result on the right.

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation is optimized for NEON technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in BottomHatImage_AnyKernel.

BottomHatImage_Mask

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Performs a morphological black top hat (bottom hat) operation on a image using a predefined mask.

Syntax

```
void avl::BottomHatImage_Mask
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<avl::Pixel> inBorderColor,
 avl::MorphologyKernel::Type inKernel,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of outImage pixels to be computed
→	inBorderColor	Optional<Pixel>	NIL	Color of the imaginary pixels outside the image boundaries
→	inKernel	MorphologyKernel::Type		Selects kernel shape
←	outImage	Image&		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

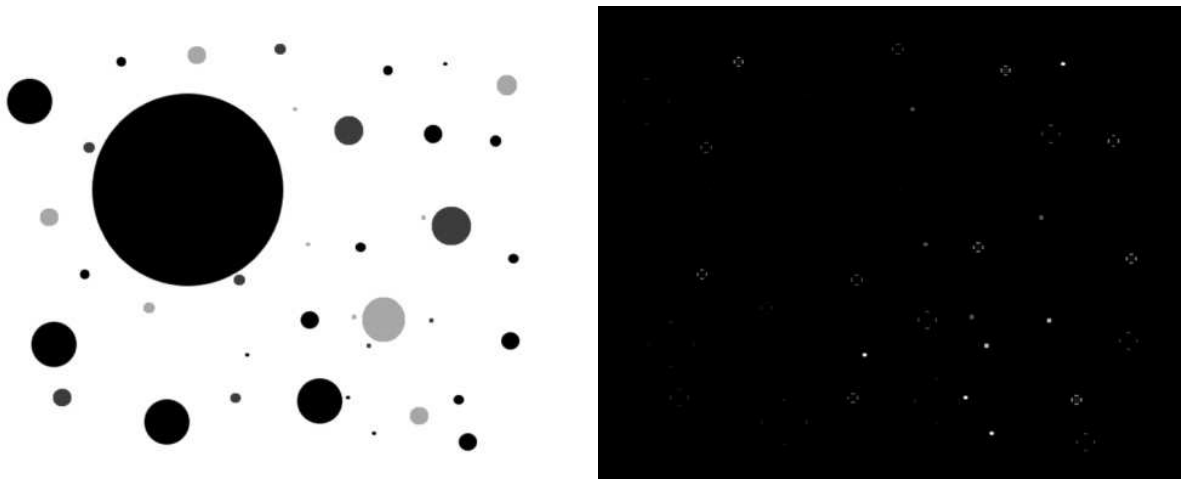
Read more about In-place Computation.

Description

Extracts from image small objects that are darker than surroundings. Uses predefined mask.

Is performed by running consecutively two filters. CloseImage_Mask to get the image without small objects and SubtractFromImage to remove everything but them.

Examples



*Bottom Hat used to remove bigger objects. Used parameters **inKernel**=Box5x5. Source image on the left and result on the right.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation is optimized for NEON technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in BottomHatImage_Mask.

CloseImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Removes small dark structures from an image (or fills in bright ones) by applying consecutive dilation and erosion.

Applications: E.g. removal of the "pepper" component of salt-and-pepper noise.

Syntax

```
void avl::CloseImage
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<const avl::Region&> inSourceRoi,
 atl::Optional<avl::Pixel> inBorderColor,
 avl::KernelShape::Type inKernel,
 int inRadiusX,
 atl::Optional<int> inRadiusY,
 avl::Image& outImage,
 avl::Region& diagKernel
)
```

Parameters

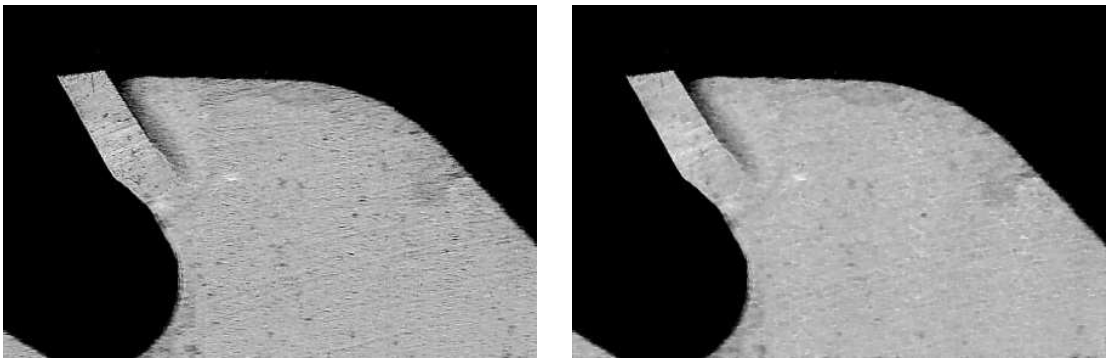
	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Range of outImage pixels to be computed
→	inSourceRoi	Optional<const Region&>		NIL	Range of inImage pixels to be considered in computations
→	inBorderColor	Optional<Pixel>		NIL	Color of the imaginary pixels outside the image boundaries
→	inKernel	KernelShape::Type			Selects kernel shape
→	inRadiusX	int	0 - ∞	1	Nearly half of the kernel's width (2*R+1)
→	inRadiusY	Optional<int>	0 - ∞	NIL	Nearly half of the kernel's height
←	outImage	Image&			Output image
🔍	diagKernel	Region&			Kernel shape

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

Read more about In-place Computation.

Examples



CloseImage *used to remove dark scratches from an image.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation is optimized for NEON technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in CloseImage.

CloseImage_AnyKernel

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Removes small dark structures from an image (or fills in bright ones) by applying consecutive dilation and erosion.

Applications: E.g. removal of the "pepper" component of salt-and-pepper noise.

Syntax

```
void avl::CloseImage_AnyKernel
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<const avl::Region&> inSourceRoi,
 atl::Optional<avl::Pixel> inBorderColor,
 const avl::Region& inKernel,
 atl::Optional<const avl::Location&> inKernelAnchor,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of outImage pixels to be computed
→	inSourceRoi	Optional<const Region&>	NIL	Range of inImage pixels to be considered in computations
→	inBorderColor	Optional<Pixel>	NIL	Color of the imaginary pixels outside the image boundaries
→	inKernel	const Region&		Kernel shape
→	inKernelAnchor	Optional<const Location&>	NIL	A location within inKernel, defining its center
←	outImage	Image&		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

Read more about In-place Computation.

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation is optimized for NEON technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in CloseImage_AnyKernel.

CloseImage_Mask

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Removes small dark structures from an image (or fills in bright ones) by applying consecutive dilation and erosion.

Applications: E.g. removal of the "pepper" component of salt-and-pepper noise.

Syntax

```
void avl::CloseImage_Mask
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<avl::Pixel> inBorderColor,
 avl::MorphologyKernel::Type inKernel,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of outImage pixels to be computed
→	inBorderColor	Optional<Pixel>	NIL	Color of the imaginary pixels outside the image boundaries
→	inKernel	MorphologyKernel::Type		Selects kernel shape
←	outImage	Image&		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

Read more about In-place Computation.

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation is optimized for NEON technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in CloseImage_Mask.
DomainError	Unsupported kernel shape in CloseImage_Mask.

ConvolveImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes a convolution of the input image with a user-specified mask.

Applications: Non-standard local transforms defined by the user.

Syntax

```
void avl::ConvolveImage
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 const avl::Matrix& inMask,
 bool inNormalizeMaskValues,
 atl::Optional<const avl::Location&> inMaskOrigin,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of outImage pixels to be computed
→	inMask	const Matrix&		Image convolution kernel
→	inNormalizeMaskValues	bool	False	Normalize sum of weights in mask to one
→	inMaskOrigin	Optional<const Location&>	NIL	Relative location of point to its mask
←	outImage	Image&		Output image

Description

The operation computes new value of pixel as a convolution of **inImage** pixels and the **inMask** values. Pixels which mask exceeds **inRoi** region are set to black. Input and output pixel have the same type. If result value does not fit into pixel type you should use ConvertPixelType.

Examples

ConvolveImage *performed on the sample image with* ***inMask*** $\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

This operation supports processing on OpenCL compatible device (when inRoi=NIL).

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty convolution mask on input in ConvolveImage.
DomainError	Mask origin is outside the dimensions of the mask in ConvolveImage.
DomainError	Region exceeds an input image in ConvolveImage.
DomainError	Sum of weights in Mask is equal zero. Cannot normalize values in ConvolveImage.

DifferenceOfGaussians

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Applies difference of Gaussians on an image, i.e. computes difference of two Gaussian smoothed images.

Applications: Emphasizes high-frequency image features such as lines or patches / dots.

Syntax

```
void avl::DifferenceOfGaussians
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 float inStdDev,
 float inStdDevRatio,
 float inKernelRelativeSize,
 float inScale,
 avl::Image& outImage,
 avl::Profile& diagKernelShape
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Range of output pixels to be computed
→	inStdDev	float	0.0 - ∞	3.0f	Smoothing standard deviation for the smaller kernel
→	inStdDevRatio	float	1.0 - ∞	1.6f	Defines how many times larger is the second kernel
→	inKernelRelativeSize	float	0.0 - 3.0	3.0f	A multiple of the standard deviation determining the size of the kernel
→	inScale	float		1.0f	Output image scaling factor
←	outImage	Image&			Output image
🔍	diagKernelShape	Profile&			Middle row of the kernel being used

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT16, REAL.

This operation is optimized for SSE41 technology for pixels of type: UINT16.

This operation is optimized for AVX2 technology for pixels of types: UINT8, SINT16, UINT16, REAL.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in DifferenceOfGaussians.

DilateAndErodeImage

Header: AVL.h
Namespace: avl
Module: FoundationPro

Calculates dilation and erosion simultaneously for faster execution.

Syntax

```
void avl::DilateAndErodeImage
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<const avl::Region&> inSourceRoi,
 atl::Optional<avl::Pixel> inBorderColor,
 avl::KernelShape::Type inKernel,
 int inRadiusX,
 atl::Optional<int> inRadiusY,
 avl::Image& outDilated,
 avl::Image& outEroded
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Range of outImage pixels to be computed
→	inSourceRoi	Optional<const Region&>		NIL	Range of inImage pixels to be read
→	inBorderColor	Optional<Pixel>		NIL	Color of the imaginary pixels outside the image boundaries
→	inKernel	KernelShape::Type			Kernel shape
→	inRadiusX	int	0 - ∞	1	Nearly half of the kernel's width (2*R+1)
→	inRadiusY	Optional<int>	0 - ∞	NIL	Nearly half of the kernel's height (2*R+1), or same as inRadiusX
←	outDilated	Image&			
←	outEroded	Image&			

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation is optimized for NEON technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	inBorderColor is relevant only when inSourceRoi is set to Auto (NIL) in DilateAndErodeImage.
DomainError	Not supported kernel on input in DilateAndErodeImage.
DomainError	Region exceeds an input image in DilateAndErodeImage.

Also in AVL Lite

DilateImage

Header: AVL.h
Namespace: avl
Module: FoundationLite

Replaces each pixel with the maximum of pixels within a kernel.

Applications: Thickens bright features in an images and thins dark ones.

Syntax

```
void avl::DilateImage
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<const avl::Region&> inSourceRoi,
 atl::Optional<avl::Pixel> inBorderColor,
 avl::KernelShape::Type inKernel,
 int inRadiusX,
 atl::Optional<int> inRadiusY,
 avl::Image& outImage,
 avl::Region& diagKernel
)
```

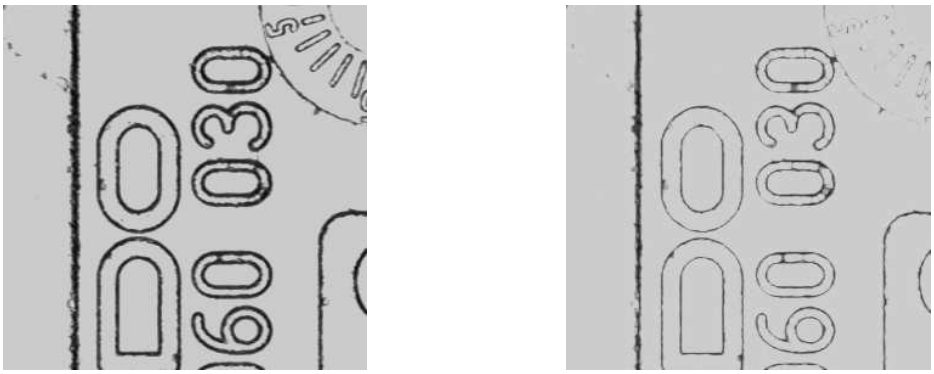
Parameters

	Name	Type	Range	Default	Description
➡	inImage	const Image&			Input image
➡	inRoi	Optional<const Region&>		NIL	Range of outImage pixels to be computed
➡	inSourceRoi	Optional<const Region&>		NIL	Range of inImage pixels to be considered in computations
➡	inBorderColor	Optional<Pixel>		NIL	Color of the imaginary pixels outside the image boundaries
➡	inKernel	KernelShape::Type			Kernel shape
➡	inRadiusX	int	0 - ∞	1	Nearly half of the kernel's width (2*R+1)
➡	inRadiusY	Optional<int>	0 - ∞	NIL	Nearly half of the kernel's height (2*R+1), or same as inRadiusX
⬅	outImage	Image&			Output image
🔍	diagKernel	Region&			Kernel shape

Description

The operation replaces each pixel with the brightest pixel in its neighbourhood, thus shrinking dark areas in **inImage** and expanding the bright ones.

Examples



DilateImage *performed on the sample image with* ***inKernel*** *= Box,* ***inRadiusX*** *= 1,* ***inRadiusY*** *= 1.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation is optimized for NEON technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	inBorderColor is relevant only when inSourceRoi is set to Auto (NIL) in DilateImage.
DomainError	Not supported kernel on input in DilateImage.
DomainError	Region exceeds an input image in DilateImage.

See Also

- ErodeImage – Replaces each pixel with the minimum of pixels within a kernel.

DilateImage_AnyKernel

Header: AVL.h
Namespace: avl
Module: FoundationPro

Replaces each pixel with the maximum of pixels within an arbitrary kernel.

Syntax

```
void avl::DilateImage_AnyKernel
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<const avl::Region&> inSourceRoi,
 atl::Optional<avl::Pixel> inBorderColor,
 const avl::Region& inKernel,
 atl::Optional<const avl::Location&> inKernelAnchor,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of outImage pixels to be computed
→	inSourceRoi	Optional<const Region&>	NIL	Range of inImage pixels to be considered in computations
→	inBorderColor	Optional<Pixel>	NIL	Color of the imaginary pixels outside the image boundaries
→	inKernel	const Region&		Kernel shape (any)
→	inKernelAnchor	Optional<const Location&>	NIL	A location within inKernel, defining its center
←	outImage	Image&		Output image

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation is optimized for NEON technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation supports automatic parallelization for multicore and multiprocessor systems.

This operation supports processing on OpenCL compatible device (when inRoi=NIL, inSourceRoi=NIL, inBorderColor=NIL).

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty kernel on input in DilateImage_AnyKernel.
DomainError	Region exceeds an input image in DilateImage_AnyKernel.

DilateImage_Mask

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Replaces each pixel with the maximum of pixels within a small rectangular kernel.

Syntax

```
void avl::DilateImage_Mask
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::MorphologyKernel::Type inKernel,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of outImage pixels to be computed
→	inKernel	MorphologyKernel::Type		Selects a predefined kernel
←	outImage	Image&		Output image

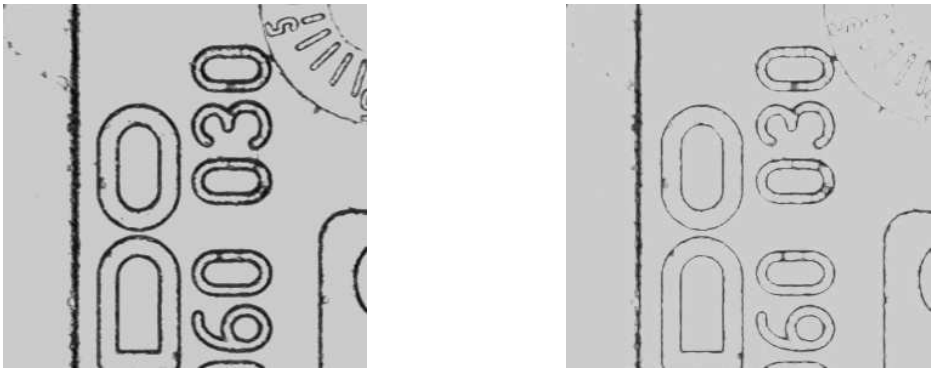
Description

The operation replaces each pixel with the brightest pixel in its neighbourhood, thus shrinking dark areas in **inImage** and expanding the bright ones.

This filter is a simplified version of DilateImage. It is limited to only a few symmetric kernels and supports only basic ROI handling.

Please note that on some machines the filter execution time can be greater than execution time of the corresponding DilateImage filter.

Examples



DilateImage_Mask *performed on the sample image with* ***inKernel*** *= Box3x3.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT16, UINT16.

This operation is optimized for AVX2 technology for pixels of types: UINT8, SINT16, UINT16.

This operation is optimized for NEON technology for pixels of types: UINT8, SINT8, SINT16, UINT16, SINT32, REAL.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in DilateImage_Mask.

See Also

- DilateImage – Replaces each pixel with the maximum of pixels within a kernel.
- DilateImage_AnyKernel – Replaces each pixel with the maximum of pixels within an arbitrary kernel.

ErodeImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Replaces each pixel with the minimum of pixels within a kernel.

Applications: Thins bright features in an image and thickens dark ones.

Syntax

```
void avl::ErodeImage
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<const avl::Region&> inSourceRoi,
 atl::Optional<avl::Pixel> inBorderColor,
 avl::KernelShape::Type inKernel,
 int inRadiusX,
 atl::Optional<int> inRadiusY,
 avl::Image& outImage,
 avl::Region& diagKernel
)
```

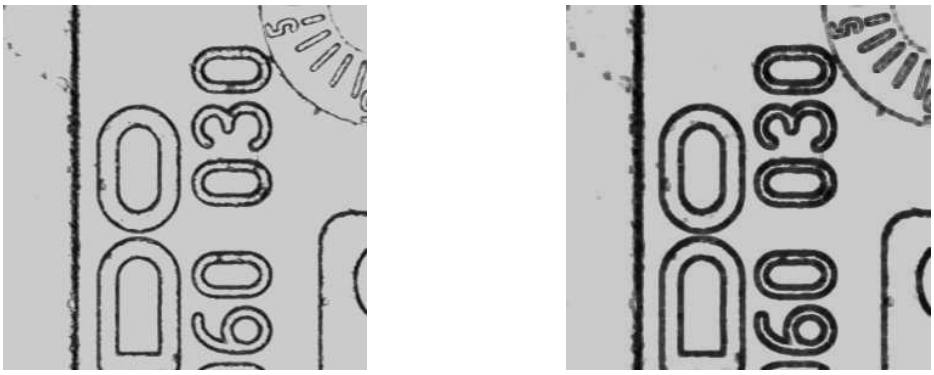
Parameters

	Name	Type	Range	Default	Description
➡	inImage	const Image&			Input image
➡	inRoi	Optional<const Region&>		NIL	Range of outImage pixels to be computed
➡	inSourceRoi	Optional<const Region&>		NIL	Range of inImage pixels to be considered in computations
➡	inBorderColor	Optional<Pixel>		NIL	Color of the imaginary pixels outside the image boundaries
➡	inKernel	KernelShape::Type			Kernel shape
➡	inRadiusX	int	0 - ∞	1	Nearly half of the kernel's width (2*R+1)
➡	inRadiusY	Optional<int>	0 - ∞	NIL	Nearly half of the kernel's height (2*R+1), or same as inRadiusX
⬅	outImage	Image&			Output image
🔍	diagKernel	Region&			Kernel shape

Description

The operation replaces each pixel with the darkest pixel in its neighbourhood, thus shrinking bright areas in **inImage** and expanding the dark ones.

Examples



ErodeImage *performed on the sample image with* ***inKernel*** *= Box,* ***inRadiusX*** *= 1,* ***inRadiusY*** *= 1.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation is optimized for NEON technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	inBorderColor is relevant only when inSourceRoi is set to Auto (NIL) in ErodeImage.
DomainError	Not supported kernel on input in ErodeImage.
DomainError	Region exceeds an input image in ErodeImage.

See Also

- DilateImage – Replaces each pixel with the maximum of pixels within a kernel.

ErodeImage_AnyKernel

Header: AVL.h
Namespace: avl
Module: FoundationPro

Replaces each pixel with the minimum of pixels within an arbitrary kernel.

Syntax

```
void avl::ErodeImage_AnyKernel
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<const avl::Region&> inSourceRoi,
 atl::Optional<avl::Pixel> inBorderColor,
 const avl::Region& inKernel,
 atl::Optional<const avl::Location&> inKernelAnchor,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of outImage pixels to be computed
→	inSourceRoi	Optional<const Region&>	NIL	Range of inImage pixels to be considered in computations
→	inBorderColor	Optional<Pixel>	NIL	Color of the imaginary pixels outside the image boundaries
→	inKernel	const Region&		Kernel shape (any)
→	inKernelAnchor	Optional<const Location&>	NIL	A location within inKernel, defining its center
←	outImage	Image&		Output image

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation is optimized for NEON technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation supports automatic parallelization for multicore and multiprocessor systems.

This operation supports processing on OpenCL compatible device (when inRoi=NIL, inSourceRoi=NIL, inBorderColor=NIL).

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty kernel on input in ErodeImage_AnyKernel.
DomainError	Region exceeds an input image in ErodeImage_AnyKernel.

ErodeImage_Mask

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Replaces each pixel with the minimum of pixels within a small rectangular kernel.

Syntax

```
void avl::ErodeImage_Mask
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::MorphologyKernel::Type inKernel,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of outImage pixels to be computed
→	inKernel	MorphologyKernel::Type		Selects a predefined kernel
←	outImage	Image&		Output image

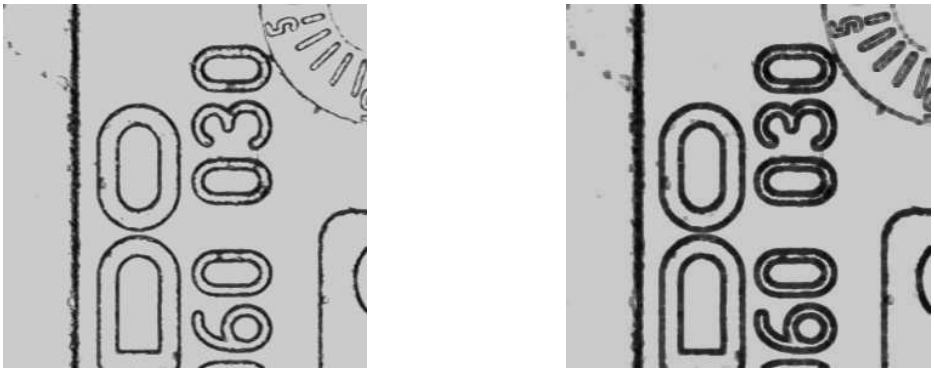
Description

The operation replaces each pixel with the darkest pixel in its neighbourhood, thus shrinking bright areas in **inImage** and expanding the dark ones.

This filter is a simplified version of ErodeImage. It is limited to only a few symmetric kernels and supports only basic ROI handling.

Please note that on some machines the filter execution time can be greater than execution time of the corresponding ErodeImage filter.

Examples



ErodeImage_Mask *performed on the sample image with* ***inKernel*** *= Box3x3.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT16, UINT16.

This operation is optimized for AVX2 technology for pixels of types: UINT8, SINT16, UINT16.

This operation is optimized for NEON technology for pixels of types: UINT8, SINT8, SINT16, UINT16, SINT32, REAL.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in ErodeImage_Mask.

See Also

- ErodeImage – Replaces each pixel with the minimum of pixels within a kernel.
- ErodeImage_AnyKernel – Replaces each pixel with the minimum of pixels within an arbitrary kernel.

GradientDirAndPresenceImage

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes an image of gradient directions mapped to the range from 1 to 255. Zero means "no edge".

Applications: For highly optimized analysis of gradient directions.

Syntax

```
void avl::GradientDirAndPresenceImage
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::GradientMaskOperator::Type inOperator,
 float inEdgeThreshold,
 avl::AngleRange::Type inAngleRange,
 avl::Image& outDirectionsImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Range of output pixels to be computed
→	inOperator	GradientMaskOperator::Type			Defines how the gradient is computed
→	inEdgeThreshold	float	0.01 - ∞	10.0f	Minimum edge magnitude (other pixels will be set to 0)
→	inAngleRange	AngleRange::Type		_0_360	Range of output angles
←	outDirectionsImage	Image&			

Description

The operation computes the angle of the intensity change direction at each pixel of the **inImage**. Firstly the selected **inOperator** is used to obtain two-dimensional gradient vector at each pixel. When this vector length exceeds **inEdgeThreshold** the angle of the vector is calculated, scaled and stored in **outDirectionsImage** pixel.

This operation always generates a single-channel image with the uint8 pixel type on the output, regardless of the input image format, with following pixel values:

- When gradient length does not exceed threshold, the value of 0 is stored in pixel.
- When gradient length exceeds threshold, its angle is scaled to range 0..255 and stored in pixel value.

When a pixel value is non-zero, one can restore the original angle with the formula:

```
double angle = value * 360.0 / 255.0;   // if inAngleRange == _0_360
double angle = value * 180.0 / 255.0;   // if inAngleRange == _0_180
double angle = value * 90.0 / 255.0;    // if inAngleRange == _0_90
```

To measure the angular distance between two directions described by two non-zero pixel values, use the following formulas:

```
   int valueDif = abs(inValue1 - inValue2);
double angleDif = (valueDif < 128 ? (double)valueDif : (255.0 - valueDif)) * 360.0 / 255.0;  // if inAngleRange ==
_0_360
double angleDif = ((double)valueDif) * 180.0 / 255.0;                                        // if inAngleRange ==
_0_180
double angleDif = ((double)valueDif) * 90.0 / 255.0;                                         // if inAngleRange ==
_0_90
```

Hardware Acceleration

This operation is optimized for SSE41 technology for pixels of type: UINT8.

This operation is optimized for AVX2 technology for pixels of type: UINT8.

This operation is optimized for NEON technology for pixels of type: UINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in GradientDirAndPresenceImage.

See Also

- GradientImage_Mask – Computes a gradient image with a Sobel or Prewitt operator.
- GradientImage – Computes a gradient image with smoothing operator of any size. The output pixels are signed.
- GradientMagnitudeImage – Measures the strength of gradient at each pixel location.

GradientImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes a gradient image with smoothing operator of any size. The output pixels are signed.

Syntax

```
void avl::GradientImage
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::GradientOperator::Type inOperator,
 const float inStdDevX,
 atl::Optional<float> inStdDevY,
 avl::Image& outGradientImage,
 avl::Image& diagGradientDirections
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Range of outGradientImage pixels to be computed
→	inOperator	GradientOperator::Type			Defines how the gradient is computed
→	inStdDevX	const float	0.0 - ∞	2.0f	Horizontal smoothing standard deviation
→	inStdDevY	Optional<float>	0.0 - ∞	NIL	Vertical smoothing standard deviation
←	outGradientImage	Image&			Gradients of the image
🔍	diagGradientDirections	Image&			Gradient directions presented in a human readable format

Remarks

Please note that about a half of the pixels will be negative and in the preview windows they will be displayed as black.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in GradientImage.

GradientImage_Mask

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes a gradient image with a Sobel or Prewitt operator.

Syntax

```
void avl::GradientImage_Mask
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::GradientMaskOperator::Type inOperator,
 const int inScale,
 avl::Image& outGradientImage,
 avl::Image& diagGradientDirections
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Range of outGradientImage pixels to be computed
→	inOperator	GradientMaskOperator::Type			Defines how the gradient is computed
→	inScale	const int	1 - 16	1	Scales the resulting gradients
←	outGradientImage	Image&			Gradients of the image
🔍	diagGradientDirections	Image&			Gradient directions presented in a human readable format

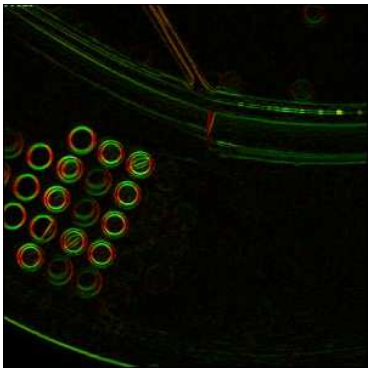
Description

The operation computes the image gradient, which is directional change of the image intensity at each pixel. Each pixel of the resulting two-channel **outGradientImage** represents the gradient at the corresponding **inImage** pixel as a two-dimensional vector (each dimension on a separate, signed image channel). The length and direction of the vector represents the strength and direction of the intensity change.

Note that the format of the resulting image is not perfectly legible for a human eye. Firstly, the resulting image contains negative values. Secondly, directions of the gradients are represented indirectly.

- To extract information about the sole **strength** of the intensity change one could use GradientMagnitudeImage filter.

Examples



GradientImage_Mask *performed on the sample image with* ***inOperator*** *= Sobel. Image on the right is actually an absolute value of the resulting* ***outGradientImage****, so negative pixel values are also visible.*

Hardware Acceleration

This operation is optimized for SSSE3 technology for pixels of type: UINT8.

This operation is optimized for AVX2 technology for pixels of type: UINT8.

This operation is optimized for NEON technology for pixels of type: UINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in GradientImage_Mask.

See Also

- GradientImage – Computes a gradient image with smoothing operator of any size. The output pixels are signed.
- GradientMagnitudeImage – Measures the strength of gradient at each pixel location.

GradientMagnitudeImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Measures the strength of gradient at each pixel location.

Syntax

```
void avl::GradientMagnitudeImage
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::GradientMaskOperator::Type inOperator,
 avl::MagnitudeMeasure::Type inMeasure,
 const int inScale,
 avl::Image& outValueImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Range of output pixels to be computed
→	inOperator	GradientMaskOperator::Type			Defines how the gradient is computed
→	inMeasure	MagnitudeMeasure::Type		Hypot	Defines how the gradient magnitude is computed
→	inScale	const int	1 - 16	1	Scales the resulting gradient magnitudes
←	outValueImage	Image&			Gradient magnitudes of the image

Description

The operation computes the magnitude of the intensity change at each pixel of the **inImage**. Firstly the selected **inOperator** is used to obtain two-dimensional gradient vector at each pixel. Then the magnitudes of the vectors are estimated using the **inMeasure** method.

Specified by **inMeasure** method computes magnitude (A) from horizontal gradient component (x) and vertical gradient component (y) using one of following formulas:

$$A_{Horizontal} = |x|$$

$$A_{Vertical} = |y|$$

$$A_{Average} = \frac{|x| + |y|}{2}$$

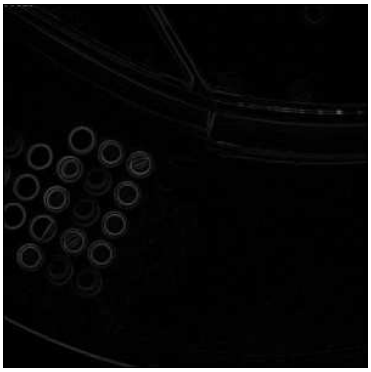
$$A_{Sum} = |x| + |y|$$

$$A_{Maximum} = Max(|x|, |y|)$$

$$A_{Hypot} = \sqrt{x^2 + y^2}$$

The magnitudes are multiplied by **inScale** factor and saturated if they exceed the greatest value of their type.

Examples



GradientMagnitudeImage *performed on the sample image with* ***inOperator*** *= Sobel,* ***inMeasure*** *= Hypot.*

Hardware Acceleration

This operation is optimized for SSSE3 technology for pixels of type: UINT8.

This operation is optimized for AVX2 technology for pixels of type: UINT8.

This operation is optimized for NEON technology for pixels of type: UINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in GradientMagnitudeImage.

See Also

- GradientImage_Mask – Computes a gradient image with a Sobel or Prewitt operator.
- GradientImage – Computes a gradient image with smoothing operator of any size. The output pixels are signed.

GradientMagnitudeImage_Signed

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes an image of gradient for only selected direction.

Applications: For highly optimized analysis of gradient directions.

Syntax

```
void avl::GradientMagnitudeImage_Signed
(
 const avl::Image& inImage,
 avl::EdgeTransition::Type inEdgeTransition,
 avl::DifferentiationMethod::Type inDiffMethod,
 avl::GradientOrientation::Type inGradientOrientation,
 bool inSigned,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inEdgeTransition	EdgeTransition::Type		Defines what is considered as an edge in the image.
→	inDiffMethod	DifferentiationMethod::Type		Defines which finite difference approach to use
→	inGradientOrientation	GradientOrientation::Type		Defines which gradient orientation to compute
→	inSigned	bool	False	Defines whether or not to output a signed result (true) or just a positive result (false)
←	outImage	Image&		Output image

Requirements

For input **inImage** only pixel formats are supported: 1□uint8, 1□int8, 1□uint16, 1□int16, 1□int32, 1□real.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation is optimized for SSE41 technology for pixels of type: UINT8.

This operation is optimized for AVX2 technology for pixels of type: UINT8.

This operation is optimized for NEON technology for pixels of type: UINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported inImage pixel format in GradientMagnitudeImage_Signed. Supported formats: 1xUInt8, 1xInt8, 1xUInt16, 1xInt16, 1xInt32, 1xReal.

OpenImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Removes small bright structures from an image (or fills in dark ones) by applying consecutive erosion and dilation.

Applications: E.g. removal of the "salt" component of salt-and-pepper noise.

Syntax

```
void avl::OpenImage
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<const avl::Region&> inSourceRoi,
 atl::Optional<avl::Pixel> inBorderColor,
 avl::KernelShape::Type inKernel,
 int inRadiusX,
 atl::Optional<int> inRadiusY,
 avl::Image& outImage,
 avl::Region& diagKernel
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Range of outImage pixels to be computed
→	inSourceRoi	Optional<const Region&>		NIL	Range of inImage pixels to be considered in computations
→	inBorderColor	Optional<Pixel>		NIL	Color of the imaginary pixels outside the image boundaries
→	inKernel	KernelShape::Type			Kernel shape
→	inRadiusX	int	0 - ∞	1	Nearly half of the kernel's width (2*R+1)
→	inRadiusY	Optional<int>	0 - ∞	NIL	Nearly half of the kernel's height (2*R+1), or same as inRadiusX
←	outImage	Image&			Output image
🔍	diagKernel	Region&			Kernel shape

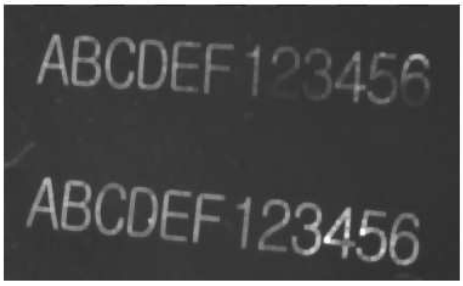
In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

Read more about In-place Computation.

Examples

Input image with salt-and-pepper noise.



Result of applying OpenImage.

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation is optimized for NEON technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in OpenImage.

OpenImage_AnyKernel

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Removes small bright structures from an image (or fills in dark ones) by applying consecutive erosion and dilation.

Applications: E.g. removal of the "salt" component of salt-and-pepper noise.

Syntax

```
void avl::OpenImage_AnyKernel
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<const avl::Region&> inSourceRoi,
 atl::Optional<avl::Pixel> inBorderColor,
 const avl::Region& inKernel,
 atl::Optional<const avl::Location&> inKernelAnchor,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of outImage pixels to be computed
→	inSourceRoi	Optional<const Region&>	NIL	Range of inImage pixels to be considered in computations
→	inBorderColor	Optional<Pixel>	NIL	Color of the imaginary pixels outside the image boundaries
→	inKernel	const Region&		Kernel shape
→	inKernelAnchor	Optional<const Location&>	NIL	A location within inKernel, defining its center
←	outImage	Image&		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

Read more about In-place Computation.

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation is optimized for NEON technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in OpenImage_AnyKernel.

OpenImage_Mask

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Removes small bright structures from an image (or fills in dark ones) by applying consecutive erosion and dilation.

Applications: E.g. removal of the "salt" component of salt-and-pepper noise.

Syntax

```
void avl::OpenImage_Mask
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<avl::Pixel> inBorderColor,
 avl::MorphologyKernel::Type inKernel,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of outImage pixels to be computed
→	inBorderColor	Optional<Pixel>	NIL	Color of the imaginary pixels outside the image boundaries
→	inKernel	MorphologyKernel::Type		Kernel shape
←	outImage	Image&		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

Read more about In-place Computation.

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation is optimized for NEON technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in OpenImage_Mask.
DomainError	Unsupported kernel shape in OpenImage_Mask.

SmoothImage_Bilateral

Header: AVL.h
Namespace: avl
Module: FoundationPro

Smooths an image while preserving sharp edges.

Syntax

```
void avl::SmoothImage_Bilateral
(
 const avl::Image& inImage,
 atl::Optional<const avl::Image&> inEdgesImage,
 atl::Optional<const avl::Region&> inRoi,
 const float inDistanceSigma,
 const float inColorSigma,
 avl::BilateralSamplingMethod::Type inSamplingMethod,
 const int inIterationCount,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inEdgesImage	Optional<const Image&>		NIL	Image to be used for edge detection
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inDistanceSigma	const float	0.0 - 128.0	5.0f	Sigma used when calculating the gaussian difference two pixel positions
→	inColorSigma	const float	0.0 - ∞	35.0f	Sigma used when calculating the gaussian difference between two colors
→	inSamplingMethod	BilateralSamplingMethod::Type		Poisson	Whether to use a naive (full) or a subsampling method
→	inIterationCount	const int	1 - ∞	1	Determines how many times the bilateral filter will be run
←	outImage	Image&			Output image

Description

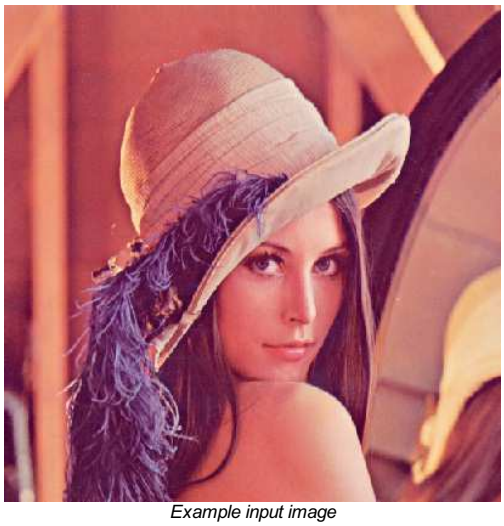
The bilateral filter is a low pass filtering algorithm weighting pixels not just in the spatial domain (which is what the gaussian filter does) but also in the color domain. This allows the filter to consider only the pixels that are both nearby and similar in color. The resulting image is smoothed but sharp edges are maintained.

A different image can be used for color distance calculations if the **inEdgesImage** parameter is set. This allows to transfer edges from other images or preserve edges only for part of an image.

The filter can be run in a iterative mode. When **inIterationCount** is set to a value greater than 1 the filter runs **inIterationCount** times using the result from the previous iteration as the edges image in the next.

The algorithm is computationally complex so, if high accuracy is not needed, Poisson Disk subsampling can be used to significantly speed up the algorithms execution time with minimal looses in quality.

Examples



Example input image

Output image with distance sigma set to 5 and color sigma set to 35

Output image with distance sigma set to 10 and color sigma set to 35

Edges image that is grey everywhere outside of a circle and the resulting output image.

Edges image which is different from the input image and the resulting output image.

Output image with distance sigma set to 5 and color sigma set to 35 after one iteration

Output image with distance sigma set to 5 and color sigma set to 35 after three iterations

Remarks

The **inDistanceSigma** parameter behaves similarly to standard deviation in the gaussian filter.

Higher values of **inColorSigma** allow the filter to blend pixel that differ more in terms of color.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Image and edges image must be of the same size in SmoothImage_Bilateral.
DomainError	Image and edges image must be of the same type in SmoothImage_Bilateral.
DomainError	Image and edges image must have the same depth in SmoothImage_Bilateral.
DomainError	Region exceeds an input image in SmoothImage_Bilateral.

See Also

- SmoothImage_Gauss – Smooths an image using a gaussian kernel.

SmoothImage_Deriche

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Smooths an image using Deriche filter.

Applications: Approximation of the gaussian filter, which can be faster for large kernels.

Syntax

```
void avl::SmoothImage_Deriche
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 const float inAlphaX,
 atl::Optional<float> inAlphaY,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Range of output pixels to be computed
→	inAlphaX	const float	0.001 - ∞	0.5f	Horizontal coefficient
→	inAlphaY	Optional<float>	0.001 - ∞	NIL	Vertical coefficient
←	outImage	Image&			Output image

Hints

- To make smoothing stronger, decrease the **inAlphaX** and - optionally - **inAlphaY**.
- If the smoothing kernel is not very big, it is recommended to use SmoothImage_Gauss instead.

Examples

SmoothImage_Deriche *performed on a sample image with* ***inAlphaX*** *= 0.5.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in SmoothImage_Deriche.

SmoothImage_DirAndPresence

Header: AVL.h
Namespace: avl
Module: FoundationPro

Smooths the result of GradientDirAndPresenceImage.

Syntax

```
void avl::SmoothImage_DirAndPresence
(
 const avl::Image& inDirectionsImage,
 const int inRadiusX,
 atl::Optional<int> inRadiusY,
 const int inQuantizationLevel,
 const int inMinSampleCount,
 avl::Image& outDirectionsImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inDirectionsImage	const Image&			
→	inRadiusX	const int	0 - ∞	1	Horizontal radius of the kernel (width is 2r+1)
→	inRadiusY	Optional<int>	0 - ∞	NIL	Vertical radius of the kernel (height is 2r+1)
→	inQuantizationLevel	const int	1 - 5	5	Determines angular quantization level; the lower the better but slower
→	inMinSampleCount	const int	1 - ∞	1	A window is evaluated to 0 if it contains less than this number of non-zero pixels
←	outDirectionsImage	Image&			

Errors

List of possible exceptions:

Error type	Description
DomainError	The result of GradientDirAndPresenceImage (of type 1xUInt8) should be provided in inGradientDirAndPresenceImage in SmoothImage_DirAndPresence.

SmoothImage_Gauss

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Smooths an image using a gaussian kernel.

Applications: Removal of gaussian noise from images.

Syntax

```
void avl::SmoothImage_Gauss
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 float inStdDevX,
 atl::Optional<float> inStdDevY,
 const float inKernelRelativeSize,
 avl::Image& outImage,
 int& diagKernelRadiusX,
 int& diagKernelRadiusY
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Range of output pixels to be computed
→	inStdDevX	float	0.0 - ∞	1.0f	Horizontal smoothing standard deviation
→	inStdDevY	Optional<float>	0.0 - ∞	NIL	Vertical smoothing standard deviation
→	inKernelRelativeSize	const float	0.0 - 3.0	2.0f	A multiple of the standard deviation determining the size of the kernel
←	outImage	Image&			Output image
🔍	diagKernelRadiusX	int&			Horizontal radius of Gaussian kernel being used
🔍	diagKernelRadiusY	int&			Vertical radius of Gaussian kernel being used

Hints

- To make smoothing stronger, increase the **inStdDevX** and - optionally - **inStdDevY**.
- Increase **inKernelRelativeSize** to achieve better quality at the cost of a bit longer execution time.
- For small kernels consider switching to SmoothImage_Gauss_Mask to achieve the highest performance.

Examples

SmoothImage_Gauss *performed on a sample image with* ***inStdDevX*** *= 3.0.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT16, REAL.

This operation is optimized for SSE41 technology for pixels of type: UINT16.

This operation is optimized for AVX2 technology for pixels of types: UINT8, SINT16, UINT16, REAL.

This operation is optimized for NEON technology for pixels of types: UINT8, SINT16, UINT16, REAL.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in SmoothImage_Gauss.

SmoothImage_Gauss_Mask

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Smooths an image using a predefined gaussian kernel.

Applications: Removal of gaussian noise from images (fast).

Syntax

```
void avl::SmoothImage_Gauss_Mask
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::GaussKernel::Type inKernel,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of outImage pixels to be computed
→	inKernel	GaussKernel::Type	_3x3	Predefined Gauss kernel
←	outImage	Image&		Output image

Description

This operation is a simplified, fast in computation, version of SmoothImage_Gauss, with predefined kernel and simplified ROI handling.

Kernel used in operation can be chosen by **inKernel** parameter:

- **Box_2x2**: 3 by 3 pixels kernel similar to 2 by 2 gaussian kernel rotated by 45 degrees:

$$\frac{1}{8}\begin{bmatrix}0 & 1 & 0\\ 1 & (4) & 1\\ 0 & 1 & 0\end{bmatrix}$$

- **Box_3x3**: 3 by 3 pixels kernel with StdDev ≈ 0.85 of following form:

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & (4) & 2\\ 1 & 2 & 1\end{bmatrix}$$

- **Box_5x5**: 5 by 5 pixels kernel with StdDev ≈ 1.1 of following form:

$$\frac{1}{109}\begin{bmatrix}1 & 3 & 5 & 3 & 1\\ 3 & 9 & 15 & 9 & 3\\ 5 & 15 & (25) & 15 & 5\\ 3 & 9 & 15 & 9 & 3\\ 1 & 3 & 5 & 3 & 1\end{bmatrix}$$

- **Box_7x7**: 7 by 7 pixels kernel with StdDev ≈ 1.7 of following form:

$$\frac{1}{256}\begin{bmatrix}1 & 2 & 3 & 4 & 3 & 2 & 1\\ 2 & 4 & 6 & 8 & 6 & 4 & 2\\ 3 & 6 & 9 & 12 & 9 & 6 & 3\\ 4 & 8 & 12 & (16) & 12 & 8 & 4\\ 3 & 6 & 9 & 12 & 9 & 6 & 3\\ 2 & 4 & 6 & 8 & 6 & 4 & 2\\ 1 & 2 & 3 & 4 & 3 & 2 & 1\end{bmatrix}$$

- **Box_9x9**: 9 by 9 pixels kernel with StdDev ≈ 2.0 of following form:

$$\frac{1}{1089}\begin{bmatrix}1 & 2 & 4 & 6 & 7 & 6 & 4 & 2 & 1\\ 2 & 4 & 8 & 12 & 14 & 12 & 8 & 4 & 2\\ 4 & 8 & 16 & 24 & 28 & 24 & 16 & 8 & 4\\ 6 & 12 & 24 & 36 & 42 & 36 & 24 & 12 & 6\\ 7 & 14 & 28 & 42 & (49) & 42 & 28 & 14 & 7\\ 6 & 12 & 24 & 36 & 42 & 36 & 24 & 12 & 6\\ 4 & 8 & 16 & 24 & 28 & 24 & 16 & 8 & 4\\ 2 & 4 & 8 & 12 & 14 & 12 & 8 & 4 & 2\\ 1 & 2 & 4 & 6 & 7 & 6 & 4 & 2 & 1\end{bmatrix}$$

- **Box_11x11**: 11 by 11 pixels kernel with StdDev ≈ 2.2 of following form:

$$\frac{1}{4096}\begin{bmatrix}1 & 2 & 4 & 8 & 11 & 12 & 11 & 8 & 4 & 2 & 1\\ 2 & 4 & 8 & 16 & 22 & 24 & 22 & 16 & 8 & 4 & 2\\ 4 & 8 & 16 & 32 & 44 & 48 & 44 & 32 & 16 & 8 & 4\\ 8 & 16 & 32 & 64 & 88 & 96 & 88 & 64 & 32 & 16 & 8\\ 11 & 22 & 44 & 88 & 121 & 132 & 121 & 88 & 44 & 22 & 11\\ 12 & 24 & 48 & 96 & 132 & (144) & 132 & 96 & 48 & 24 & 12\\ 11 & 22 & 44 & 88 & 121 & 132 & 121 & 88 & 44 & 22 & 11\\ 8 & 16 & 32 & 64 & 88 & 96 & 88 & 64 & 32 & 16 & 8\\ 4 & 8 & 16 & 32 & 44 & 48 & 44 & 32 & 16 & 8 & 4\\ 2 & 4 & 8 & 16 & 22 & 24 & 22 & 16 & 8 & 4 & 2\\ 1 & 2 & 4 & 8 & 11 & 12 & 11 & 8 & 4 & 2 & 1\end{bmatrix}$$

Hints

- Select kernel size by setting the **inKernel** input.

Examples

SmoothImage_Gauss_Mask *performed on a sample image with* ***inKernel*** *= _11x11.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT16.

This operation is optimized for SSE4.1 technology for pixels of types: Kernel 11x11 UINT8.

This operation is optimized for AVX2 technology for pixels of type: UINT8.

This operation is optimized for NEON technology for pixels of type: UINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in SmoothImage_Gauss_Mask.

See Also

- SmoothImage_Mean_Mask – Smooths an image by averaging pixels within a small rectangular kernel.
- SmoothImage_Gauss – Smooths an image using a gaussian kernel.

SmoothImage_Mean

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Smooths an image by averaging pixels within a rectangular kernel.

Applications: Usually used for computing features related to local image "windows". Can be also used for noise removal, but Gauss is superior here.

Syntax

```
void avl::SmoothImage_Mean
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<const avl::Region&> inSourceRoi,
 atl::Optional<avl::Pixel> inBorderColor,
 avl::KernelShape::Type inKernel,
 int inRadiusX,
 atl::Optional<int> inRadiusY,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Range of outImage pixels to be computed
→	inSourceRoi	Optional<const Region&>		NIL	Range of inImage pixels to be read
→	inBorderColor	Optional<Pixel>		NIL	Color of the imaginary pixels outside the image boundaries
→	inKernel	KernelShape::Type			Kernel shape
→	inRadiusX	int	0 - ∞	1	Nearly half of the kernel's width (2*R+1)
→	inRadiusY	Optional<int>	0 - ∞	NIL	Nearly half of the kernel's height (2*R+1), or same as inRadiusX
←	outImage	Image&			Output image

Description

Replaces each pixel with the average of all pixels contained in a rectangular kernel. The width of the kernel is **2 * inRadiusX + 1**, the height is **2 * inRadiusY + 1**. When **inRadiusY** is set to **Auto**, then its value is implicitly copied from **inRadiusX**.

Hints

- Define the size of the kernel by setting **inRadiusX** and - optionally - **inRadiusY**.
- Highest performance will be achieved with **inKernel** = Box. Other kernel shapes will result is longer execution time.
- Define **inSourceRoi** is some pixels of the input images should be ignored (advanced).

Examples

SmoothImage_Mean *performed on a sample image with* ***inRadiusX*** *= 4.*

Hardware Acceleration

This operation is optimized for PARALLEL SSE2 technology for pixels of types: UINT8, SINT8, SINT16, SINT32, REAL.

This operation is optimized for SSE41 technology for pixels of type: UINT16.

This operation is optimized for AVX2 technology for pixels of types: UINT8, SINT8, SINT16, SINT32, REAL, UINT16.

This operation is optimized for NEON technology for pixels of types: UINT8, UINT16.

Errors

List of possible exceptions:

Error type	Description
DomainError	inBorderColor is relevant only when inSourceRoi is set to Auto (NIL) in SmoothImage_Mean.
DomainError	Not supported kernel on input in SmoothImage_Mean.
DomainError	Region exceeds an input image in SmoothImage_Mean.

SmoothImage_Mean_AnyKernel

Header: AVL.h
Namespace: avl
Module: FoundationPro

Smooths an image by averaging pixels within an arbitrary kernel.

Applications: Usually used for computing features related to local image "windows" having non-standard shape.

Syntax

```
void avl::SmoothImage_Mean_AnyKernel
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<const avl::Region&> inSourceRoi,
 atl::Optional<avl::Pixel> inBorderColor,
 const avl::Region& inKernel,
 atl::Optional<const avl::Location&> inKernelAnchor,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of outImage pixels to be computed
→	inSourceRoi	Optional<const Region&>	NIL	Range of inImage pixels to be considered in computations
→	inBorderColor	Optional<Pixel>	NIL	Color of the imaginary pixels outside the image boundaries
→	inKernel	const Region&		Kernel shape (any)
→	inKernelAnchor	Optional<const Location&>	NIL	A location within inKernel, defining its center
←	outImage	Image&		Output image

Hints

- Define the kernel shape by setting the **inKernel** input.
- Define the kernel center point by setting the **inKernelAnchor** input. If you leave it **NIL**, the mass center will be used.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty kernel on input in SmoothImage_Mean_AnyKernel.
DomainError	Region exceeds an input image in SmoothImage_Mean_AnyKernel.

See Also

- SmoothImage_Mean – Smooths an image by averaging pixels within a rectangular kernel.

SmoothImage_Mean_Mask

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Smooths an image by averaging pixels within a small rectangular kernel.

Applications: This is a faster alternative to SmoothImage_Mean when the kernel is very small.

Syntax

```
void avl::SmoothImage_Mean_Mask
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::MeanKernel::Type inKernel,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of outImage pixels to be computed
→	inKernel	MeanKernel::Type		Selects a predefined kernel
←	outImage	Image&		Output image

Description

This operation is a simplified, fast in computation, version of SmoothImage_Mean, with predefined kernel and simplified ROI handling.

Kernel used in operation can be chosen by **inKernel** parameter:

- **Box_3x3**: 3 by 3 pixels, the closest equivalent is SmoothImage_Mean call with **inKernel** = Box, **inRadiusX** = 1, **inRadiusY** = 1, **inIterationsCount** = 1
- **Box_5x5**: 5 by 5 pixels, the closest equivalent is SmoothImage_Mean call with **inKernel** = Box, **inRadiusX** = 2, **inRadiusY** = 2, **inIterationsCount** = 1

Although above calls are described as closest equivalent their results may vary.

Hints

- Choose kernel size by setting **inKernel** input.
- If a larger kernel is required, switch to SmoothImage_Mean.
- If you want to remove noises from an image, consider using SmoothImage_Gauss_Mask instead.

Examples

SmoothImage_Mean_Mask *performed on a sample image with* ***inKernel*** *= Box5x5.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT16.

This operation is optimized for AVX2 technology for pixels of types: SSE2: UINT8, SINT16.

This operation is optimized for NEON technology for pixels of types: UINT8, SINT8, UINT16, SINT16.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in SmoothImage_Mean_Mask.

See Also

- SmoothImage_Gauss_Mask – Smooths an image using a predefined gaussian kernel.
- SmoothImage_Mean – Smooths an image by averaging pixels within a rectangular kernel.

SmoothImage_Median

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Replaces each pixel with the median of pixels within a kernel.

Applications: Edge-preserving noise removal (but slow).

Syntax

```
void avl::SmoothImage_Median
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<const avl::Region&> inSourceRoi,
 avl::SmoothImageMedianKernel::Type inKernel,
 int inRadiusX,
 atl::Optional<int> inRadiusY,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Range of outImage pixels being written
→	inSourceRoi	Optional<const Region&>		NIL	Range of inImage pixels being read
→	inKernel	SmoothImageMedianKernel::Type			
→	inRadiusX	int	0 - ∞	1	Nearly half of the kernel's width (2*R+1)
→	inRadiusY	Optional<int>	0 - ∞	NIL	Nearly half of the kernel's height (2*R+1), or same as inRadiusX
←	outImage	Image&			Output image

Requirements

For input **inImage** only pixel formats are supported: uint8.

Read more about pixel formats in Image documentation.

Hints

- Define the kernel size by setting the **inRadiusX** and **inRadiusY** inputs.
- For small kernels consider switching to SmoothImage_Median_Mask to achieve higher performance.

Examples

SmoothImage_Median *performed on a sample image with* ***inRadiusX*** *= 4,* ***inRadiusY*** *= Nil.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported kernel in SmoothImage_Median.
DomainError	Not supported pixel format in SmoothImage_Median.
DomainError	Region exceeds an input image in SmoothImage_Median.
DomainError	Source roi exceeds an input image in SmoothImage_Median.
DomainError	Not supported inImage pixel format in SmoothImage_Median. Supported formats: UInt8.

SmoothImage_Median_Mask

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Replaces each pixel with the median of pixels within a 3x3 rectangular kernel (faster).

Syntax

```
void avl::SmoothImage_Median_Mask
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of outImage pixels to be computed
←	outImage	Image&		Output image

Requirements

For input **inImage** only pixel formats are supported: uint8.

Read more about pixel formats in Image documentation.

Examples

SmoothImage_Median_Mask *performed on a sample image.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in SmoothImage_Median_Mask.
DomainError	Not supported inImage pixel format in SmoothImage_Median_Mask. Supported formats: UInt8.

SmoothImage_Middle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Replaces each pixel with the average of maximum and minimum calculated within a kernel.

Applications: Useful for calculating per-pixel threshold values for image binarization.

Syntax

```
void avl::SmoothImage_Middle
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<const avl::Region&> inSourceRoi,
 atl::Optional<avl::Pixel> inBorderColor,
 avl::KernelShape::Type inKernel,
 int inRadiusX,
 atl::Optional<int> inRadiusY,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Range of outImage pixels to be computed
→	inSourceRoi	Optional<const Region&>		NIL	Range of inImage pixels to be read
→	inBorderColor	Optional<Pixel>		NIL	Color of the imaginary pixels outside the image boundaries
→	inKernel	KernelShape::Type			Kernel shape
→	inRadiusX	int	0 - ∞	1	Nearly half of the kernel's width (2*R+1)
→	inRadiusY	Optional<int>	0 - ∞	NIL	Nearly half of the kernel's height (2*R+1), or same as inRadiusX
←	outImage	Image&			Output image

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation is optimized for technology.

This operation is optimized for NEON technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	inBorderColor is relevant only when inSourceRoi is set to Auto (NIL) in SmoothImage_Middle.
DomainError	Not supported kernel on input in SmoothImage_Middle.
DomainError	Region exceeds an input image in SmoothImage_Middle.

SmoothImage_Quantile

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Replaces each pixel with a quantile of pixels within a kernel.

Applications: Edge-preserving noise removal (but slow).

Syntax

```
void avl::SmoothImage_Quantile
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<const avl::Region&> inSourceRoi,
 float inQuantile,
 avl::SmoothImageMedianKernel::Type inKernel,
 int inRadiusX,
 atl::Optional<int> inRadiusY,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Range of outImage pixels being written
→	inSourceRoi	Optional<const Region&>		NIL	Range of inImage pixels being read
→	inQuantile	float	0.0 - 1.0	0.8f	The quantile to be calculated for a neighbourhood of each pixel
→	inKernel	SmoothImageMedianKernel::Type			
→	inRadiusX	int	0 - ∞	1	Nearly half of the kernel's width (2*R+1)
→	inRadiusY	Optional<int>	0 - ∞	NIL	Nearly half of the kernel's height (2*R+1), or same as inRadiusX
←	outImage	Image&			Output image

Requirements

For input **inImage** only pixel formats are supported: uint8.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported kernel in SmoothImage_Quantile.
DomainError	Not supported pixel format in SmoothImage_Quantile.
DomainError	Region exceeds an input image in SmoothImage_Quantile.
DomainError	Source roi exceeds an input image in SmoothImage_Quantile.
DomainError	Not supported inImage pixel format in SmoothImage_Quantile. Supported formats: UInt8.

SmoothRegion_Mean

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Smooths an region by averaging pixels within a rectangular kernel.

Applications: Usually used for computing features related to local image "windows". Can be also used for noise removal, but Gauss is superior here.

Syntax

```
void avl::SmoothRegion_Mean
(
 const avl::Region& inRegion,
 int inRadiusX,
 atl::Optional<int> inRadiusY,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inRegion	const Region&			Input region
→	inRadiusX	int	0 - ∞	1	Nearly half of the kernel's width (2*R+1)
→	inRadiusY	Optional<int>	0 - ∞	NIL	Nearly half of the kernel's height (2*R+1), or same as inRadiusX
←	outImage	Image&			Output image

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

StandardDeviationImage

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Creates image of pixels' local standard deviations.

Syntax

```
void avl::StandardDeviationImage
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<const avl::Region&> inSourceRoi,
 const int inKernelRadius,
 avl::Image& outStdDevImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image.
→	inRoi	Optional<const Region&>		NIL	Region of Interest.
→	inSourceRoi	Optional<const Region&>		NIL	Range of inImage pixels to be read
→	inKernelRadius	const int	1 - 40	4	Radius of square kernel (width and height are 2r+1).
←	outStdDevImage	Image&			Resulting image.

Requirements

For input **inImage** only pixel formats are supported: uint8, int8, uint16, int16, real.

Read more about pixel formats in Image documentation.

Examples

Input image

Output image (with additional postprocessing: normalization)

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in StandardDeviationImage.
DomainError	Not supported inImage pixel format in StandardDeviationImage. Supported formats: UInt8, Int8, UInt16, Int16, Real.

TopHatImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Performs a morphological white top hat operation on a image using a predefined kernel.

Syntax

```
void avl::TopHatImage
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<const avl::Region&> inSourceRoi,
 atl::Optional<avl::Pixel> inBorderColor,
 avl::KernelShape::Type inKernel,
 int inRadiusX,
 atl::Optional<int> inRadiusY,
 avl::Image& outImage,
 avl::Region& diagKernel
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Range of outImage pixels to be computed
→	inSourceRoi	Optional<const Region&>		NIL	Range of inImage pixels to be considered in computations
→	inBorderColor	Optional<Pixel>		NIL	Color of the imaginary pixels outside the image boundaries
→	inKernel	KernelShape::Type			Kernel shape
→	inRadiusX	int	0 - ∞	1	Nearly half of the kernel's width (2*R+1)
→	inRadiusY	Optional<int>	0 - ∞	NIL	Nearly half of the kernel's height (2*R+1), or same as inRadiusX
←	outImage	Image&			Output image
	diagKernel	Region&			Kernel shape

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

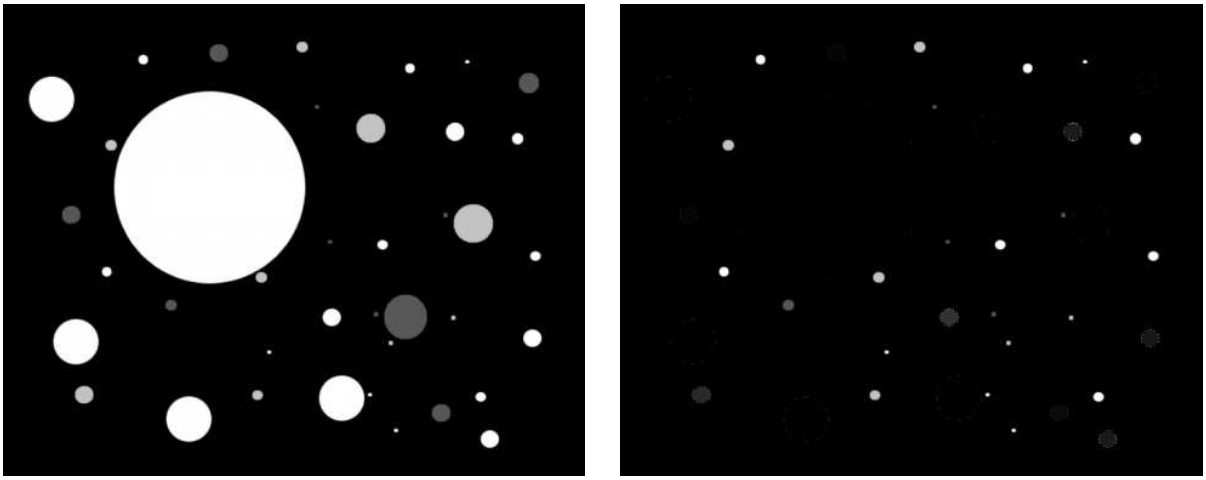
Read more about In-place Computation.

Description

Extracts from image small objects that are brighter than surroundings.

Is performed by running consecutively two filters. OpenImage to get the image without small objects and SubtractFromImage to remove everything but them.

Examples



*Top Hat used to remove bigger objects. Used parameters **inKernel**=Ellipse and **inRadiusX**=6. Source image on the left and result on the right.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation is optimized for NEON technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in TopHatImage.

TopHatImage_AnyKernel

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Performs a morphological white top hat operation on a image using an arbitrary kernel.

Syntax

```
void avl::TopHatImage_AnyKernel
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<const avl::Region&> inSourceRoi,
 atl::Optional<avl::Pixel> inBorderColor,
 const avl::Region& inKernel,
 atl::Optional<const avl::Location&> inKernelAnchor,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of outImage pixels to be computed
→	inSourceRoi	Optional<const Region&>	NIL	Range of inImage pixels to be considered in computations
→	inBorderColor	Optional<Pixel>	NIL	Color of the imaginary pixels outside the image boundaries
→	inKernel	const Region&		Kernel shape
→	inKernelAnchor	Optional<const Location&>	NIL	A location within inKernel, defining its center
←	outImage	Image&		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

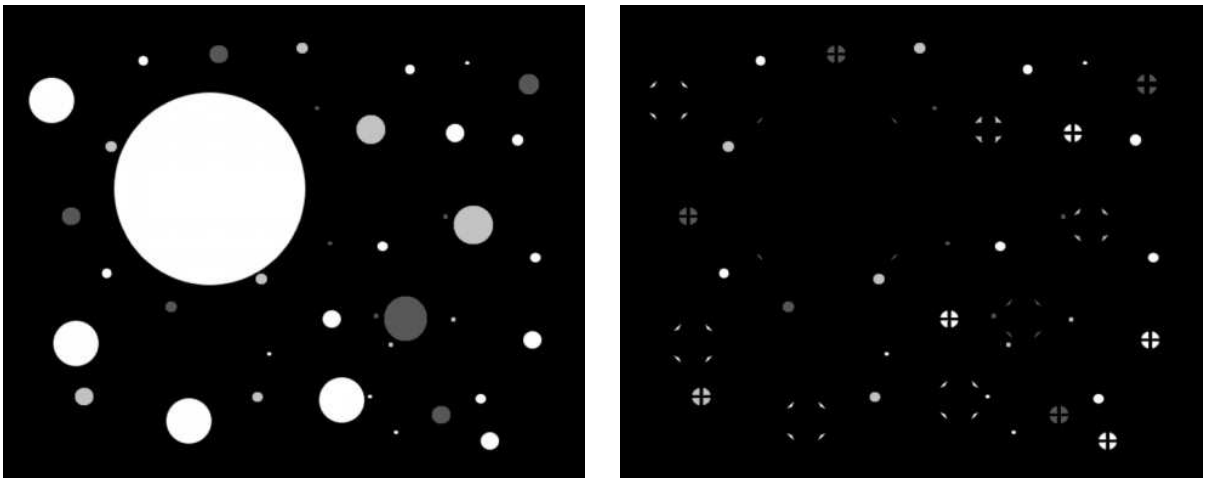
Read more about In-place Computation.

Description

Extracts from image small objects that are brighter than surroundings. Uses user-defined kernel.

Is performed by running consecutively two filters. OpenImage_AnyKernel to get the image without small objects and SubtractFromImage to remove everything but them.

Examples



Top Hat used to remove bigger objects. Source image on the left and result on the right.

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation is optimized for NEON technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in TopHatImage_AnyKernel.

TopHatImage_Mask

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Performs a morphological white top hat operation on a image using a predefined mask.

Syntax

```
void avl::TopHatImage_Mask
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<avl::Pixel> inBorderColor,
 avl::MorphologyKernel::Type inKernel,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of outImage pixels to be computed
→	inBorderColor	Optional<Pixel>	NIL	Color of the imaginary pixels outside the image boundaries
→	inKernel	MorphologyKernel::Type		Kernel shape
←	outImage	Image&		Output image

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outImage**

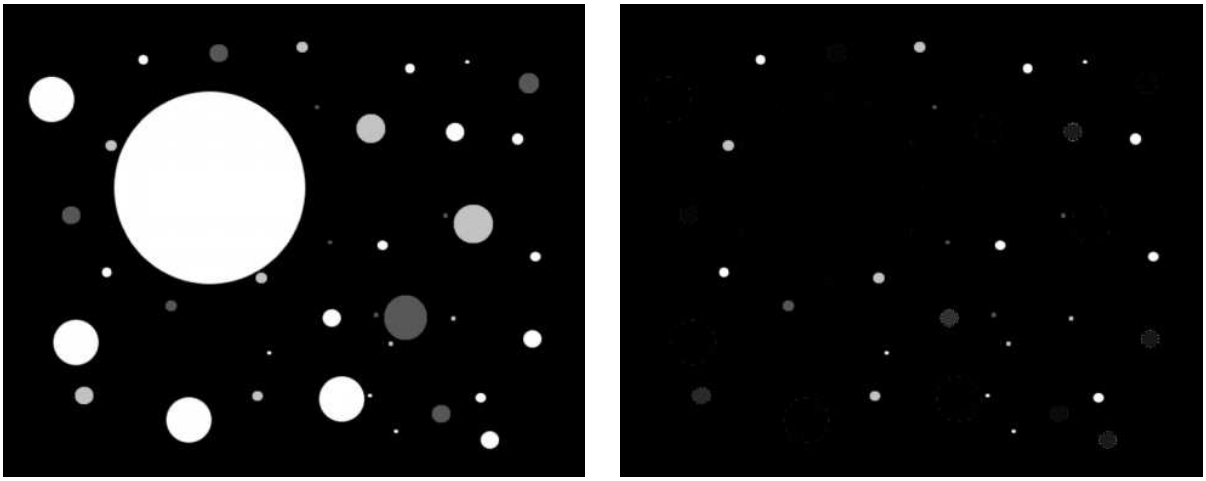
Read more about In-place Computation.

Description

Extracts from image small objects that are brighter than surroundings. Uses predefined mask.

Is performed by running consecutively two filters. OpenImage_Mask to get the image without small objects and SubtractFromImage to remove everything but them.

Examples



*Top Hat used to remove bigger objects. Used parameters **inKernel**=Box5x5. Source image on the left and result on the right.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation is optimized for NEON technology for pixels of types: all formats (when inSourceRoi = NIL and inBorderColor = NIL).

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in TopHatImage_Mask.

53. Region Morphology

Table of content:

BottomHatRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Performs a morphological black top hat (bottom hat) operation on a region using a predefined kernel.

Syntax

```
void avl::BottomHatRegion
(
 const avl::Region& inRegion,
 avl::KernelShape::Type inKernel,
 int inRadiusX,
 atl::Optional<int> inRadiusY,
 avl::Region& outRegion
)
```

Parameters

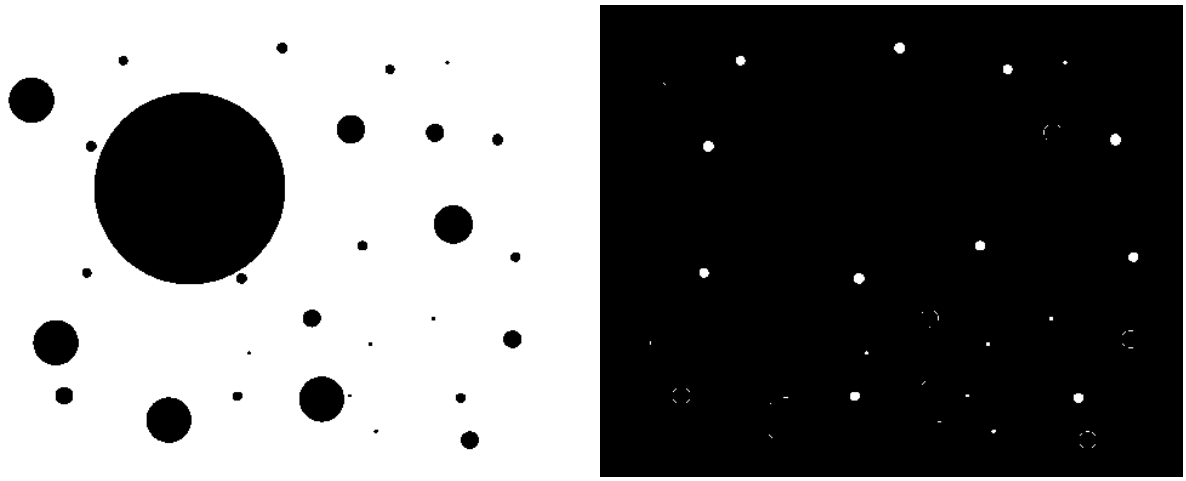
	Name	Type	Range	Default	Description
➡	inRegion	const Region&			Input region
➡	inKernel	KernelShape::Type			Kernel shape (predefined)
➡	inRadiusX	int	0 - ∞	1	Nearly half of the kernel's width (2*R+1)
➡	inRadiusY	Optional<int>	0 - ∞	NIL	Nearly half of the kernel's height (2*R+1), or same as inRadiusX
⬅	outRegion	Region&			Output region

Description

Extracts small parts which do not belong to region. Uses predefined kernels.

Is performed by running consecutively two filters. CloseRegion to remove small holes from region and RegionDifference to remove rest of the region.

Examples



*Top Hat used to remove bigger parts of which do not belong to region. Used parameters **inKernel**=Ellipse and **inRadiusX**=5. Region marked white. Source image on the left and result on the right.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Unsupported kernel in BottomHatRegion.

BottomHatRegion_AnyKernel

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Performs a morphological black top hat (bottom hat) operation on a region using an arbitrary kernel.

Syntax

```
void avl::BottomHatRegion_AnyKernel
(
 const avl::Region& inRegion,
 const avl::Region& inKernel,
 avl::Region& outRegion
)
```

Parameters

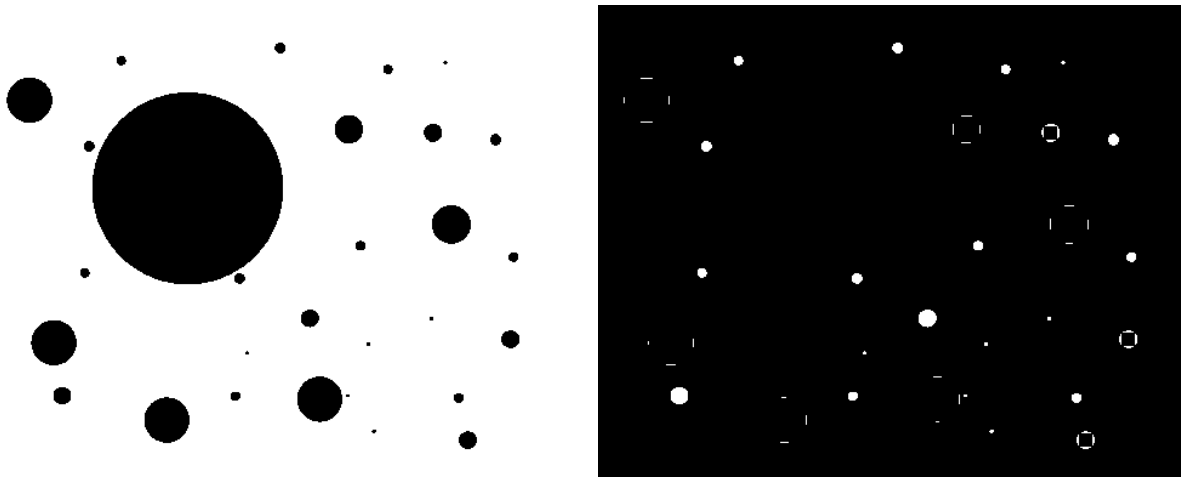
	Name	Type	Default	Description
➡	inRegion	const Region&		Input region
➡	inKernel	const Region&		Kernel shape (any)
⬅	outRegion	Region&		Output region

Description

Extracts small parts which do not belong to region. Uses user-defined kernels.

Is performed by running consecutively two filters. CloseRegion to remove small holes from region and RegionDifference to remove rest of the region.

Examples



Top Hat used to remove bigger parts of which do not belong to region. Region marked white. Source image on the left and result on the right.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty kernel on input in BottomHatRegion_AnyKernel.

CloseRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Performs a morphological closing on a region using selected predefined kernel.

Applications: Filling-in small gaps in a region without making it thicker.

Syntax

```
void avl::CloseRegion
(
 const avl::Region& inRegion,
 avl::KernelShape::Type inKernel,
 int inRadiusX,
 atl::Optional<int> inRadiusY,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inRegion	const Region&			Input region
→	inKernel	KernelShape::Type			Kernel shape (predefined)
→	inRadiusX	int	0 - ∞	1	Nearly half of the kernel's width (2*R+1)
→	inRadiusY	Optional<int>	0 - ∞	NIL	Nearly half of the kernel's height (2*R+1), or same as inRadiusX
←	outRegion	Region&			Output region

Description

The operation performs a morphological closing, which is a tool used for filling gaps in a region. The operation is a convolution of two basic morphological operations:

- Firstly, the input region is dilated using DilateRegion operation.
- Then, the resulting region is eroded using ErodeRegion operation.

During the dilation gaps that are small enough are actually closed, while further erosion assures that the width of region limbs is preserved.

Both of the component operations are conducted using the same **inKernel**, **inRadiusX** and **inRadiusY** parameters.

Hints

- Increase **inRadiusX** to fill-in bigger gaps in the region.
- Change **inKernel** to Ellipse to make the filter work equally strongly in each direction (execution will be slower).

Examples

*CloseRegion run with **inKernel** = Ellipse of dimensions **inRadiusX** = 5, **inRadiusY** = 5.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Unsupported kernel in CloseRegion.

See Also

- DilateRegion – Performs a morphological dilation on a region using a predefined kernel.
- CloseRegion_AnyKernel – Performs a morphological closing on a region using an arbitrary kernel.
- OpenRegion – Performs a morphological opening on a region using a predefined kernel.

CloseRegion_AnyKernel

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Performs a morphological closing on a region using an arbitrary kernel.

Applications: Filling-in gaps of a particular shape without making the region thicker.

Syntax

```
void avl::CloseRegion_AnyKernel
(
 const avl::Region& inRegion,
 const avl::Region& inKernel,
 avl::Region& outRegion
)
```

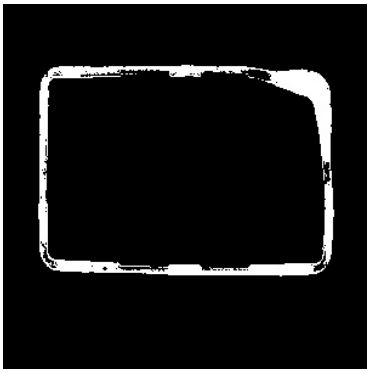
Parameters

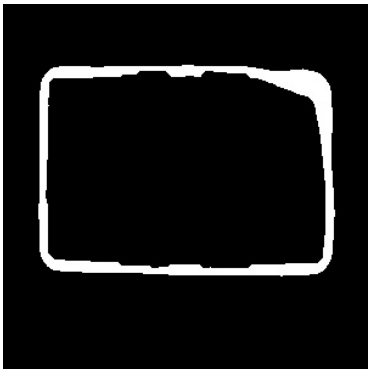
	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inKernel	const Region&		Kernel shape (any)
←	outRegion	Region&		Output region

Description

The operation is a cousin of the CloseRegion filter, yet it uses any proper region selected by user as a structuring element. The center of a kernel is assumed to be at location (width/2,height/2). Refer to the CloseRegion article for further information regarding the morphological closing itself.

Examples





Errors

List of possible exceptions:

Error type	Description
DomainError	Empty kernel on input in CloseRegion_AnyKernel.

See Also

- CloseRegion – Performs a morphological closing on a region using selected predefined kernel.

DemarcateRegions

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Splits common pixels of the input regions among these regions.

Applications: Use this filter to make sure that the regions of an array do not intersect.

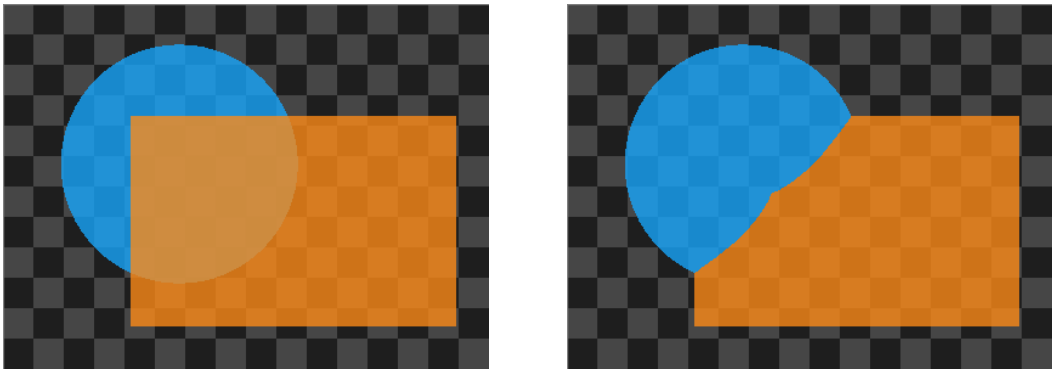
Syntax

```
void avl::DemarcateRegions
(
 const atl::Array<avl::Region>& inRegions,
 atl::Optional<avl::Metric::Type> inMetric,
 atl::Array<avl::Region>& outRegions
)
```

Parameters

	Name	Type	Default	Description
→	inRegions	const Array<Region>&		
→	inMetric	Optional<Metric::Type>	NIL	Metric used for deciding which region owns a pixel; if set to NIL, region with lowest index in the input array is chosen
←	outRegions	Array<Region>&		

Examples



*DemarcateRegions performed with **inMetric** = Euclidean.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Dimensions of two or more regions differ in DemarcateRegions.

DilateRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Performs a morphological dilation on a region using a predefined kernel.

Applications: Making the region thicker or filling-in small holes within it.

Syntax

```
void avl::DilateRegion
(
 const avl::Region& inRegion,
 avl::KernelShape::Type inKernel,
 int inRadiusX,
 atl::Optional<int> inRadiusY,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inRegion	const Region&			Input region
➡	inKernel	KernelShape::Type			Kernel shape (predefined)
➡	inRadiusX	int	0 - ∞	1	Nearly half of the kernel's width (2*R+1)
➡	inRadiusY	Optional<int>	0 - ∞	NIL	Nearly half of the kernel's height (2*R+1), or same as inRadiusX
⬅	outRegion	Region&			Output region

Description

The operation performs a morphological dilation, which is a basic tool used for region expansion. Similarly to other region morphology operations, dilation is conducted using a shape called kernel (or structuring element). The kernel is repeatedly centered at each location within the dimensions of the input region. Then, pixel location **L** is added to the resulting region if and only if at least one of the input region pixels lies inside the kernel centered at this (**L**) location.

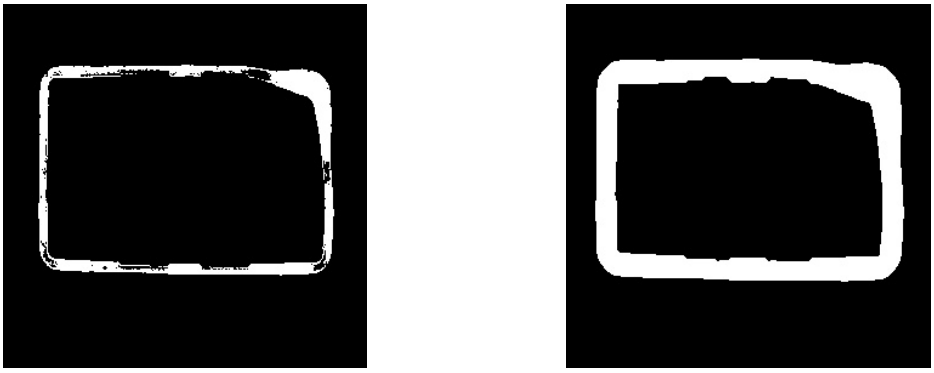
The input parameter **inKernel** allows to choose the shape of a kernel, while parameters **inRadiusX**, **inRadiusY** allow to determine its dimensions. For instance:

- **inKernel** = Ellipse together with **inRadiusX** = **inRadiusY** results in expansion performed equally in all directions.
- Setting **inRadiusX** to higher value than **inRadiusY** results in expansion performed more significantly along horizontal axis.

Hints

- Increase **inRadiusX** to make the output region thicker.
- Change **inKernel** to Ellipse to make the filter work equally strongly in each direction (execution will be slower).

Examples



*DilateRegion run with **inKernel** = Ellipse of dimensions **inRadiusX** = 5, **inRadiusY** = 5.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Unsupported kernel in DilateRegion.

See Also

- CloseRegion – Performs a morphological closing on a region using selected predefined kernel.
- DilateRegion_AnyKernel – Performs a morphological dilation on a region using an arbitrary kernel.
- ErodeRegion – Performs a morphological erosion on a region using a predefined kernel.

DilateRegion_AnyKernel

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Performs a morphological dilation on a region using an arbitrary kernel.

Applications: Making the region thicker in a non-symmetrical way or filling-in gaps of a particular shape.

Syntax

```
void avl::DilateRegion_AnyKernel
(
 const avl::Region& inRegion,
 const avl::Region& inKernel,
 atl::Optional<const avl::Location&> inKernelAnchor,
 avl::Region& outRegion
)
```

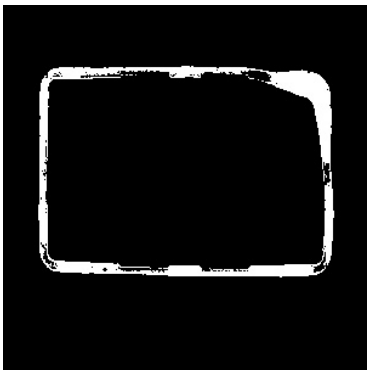
Parameters

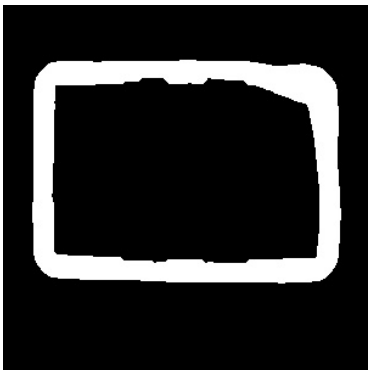
	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inKernel	const Region&		Kernel shape (any)
→	inKernelAnchor	Optional<const Location&>	NIL	Location within inKernel, defining its center
←	outRegion	Region&		Output region

Description

The operation is a cousin of the DilateRegion filter, yet it uses any proper region selected by user as a structuring element. The center of a kernel is assumed to be at location (width/2,height/2). Refer to the DilateRegion article for further information regarding the morphological dilation itself.

Examples





Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty kernel on input in DilateRegion_AnyKernel.

See Also

- DilateRegion – Performs a morphological dilation on a region using a predefined kernel.

ErodeRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Performs a morphological erosion on a region using a predefined kernel.

Applications: Making the region thinner or removing small parts.

Syntax

```
void avl::ErodeRegion
(
 const avl::Region& inRegion,
 avl::KernelShape::Type inKernel,
 int inRadiusX,
 atl::Optional<int> inRadiusY,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inRegion	const Region&			Input region
➡	inKernel	KernelShape::Type			Kernel shape (predefined)
➡	inRadiusX	int	0 - ∞	1	Nearly half of the kernel's width (2*R+1)
➡	inRadiusY	Optional<int>	0 - ∞	NIL	Nearly half of the kernel's height (2*R+1), or same as inRadiusX
⬅	outRegion	Region&			Output region

Description

The operation performs a morphological erosion, which is a basic tool used for region shrinking. Similarly to other region morphology operations, erosion is conducted using a shape called kernel (or structuring element). The kernel is repeatedly centered at each location within the dimensions of the input region. Then, pixel location **L** is added to the resulting region if and only if all of the input region pixels lies inside the kernel centered at this (**L**) location.

The input parameter **inKernel** allows to choose the shape of a kernel, while parameters **inRadiusX**, **inRadiusY** allow to determine its dimensions. For instance:

- **inKernel** = Ellipse together with **inRadiusX** = **inRadiusY** results in shrinking performed equally in all directions.
- Setting **inRadiusX** to higher value than **inRadiusY** results in shrinking performed more significantly along horizontal axis.

Hints

- Increase **inRadiusX** to make the output region thinner.
- Change **inKernel** to Ellipse to make the filter work equally strongly in each direction (execution will be slower).

Examples

*ErodeRegion run with **inKernel** = Ellipse of dimensions **inRadiusX** = 3, **inRadiusY** = 3.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Unsupported kernel in ErodeRegion.

See Also

- OpenRegion – Performs a morphological opening on a region using a predefined kernel.
- ErodeRegion_AnyKernel – Performs a morphological erosion on a region using an arbitrary kernel.
- DilateRegion – Performs a morphological dilation on a region using a predefined kernel.

ErodeRegion_AnyKernel

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Performs a morphological erosion on a region using an arbitrary kernel.

Applications: Making the region thinner in a non-symmetrical way or removing small parts of a particular shape.

Syntax

```
void avl::ErodeRegion_AnyKernel
(
 const avl::Region& inRegion,
 const avl::Region& inKernel,
 atl::Optional<const avl::Location&> inKernelAnchor,
 avl::Region& outRegion
)
```

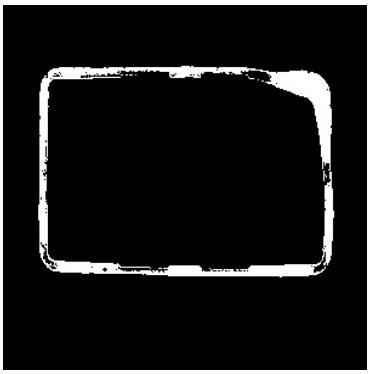
Parameters

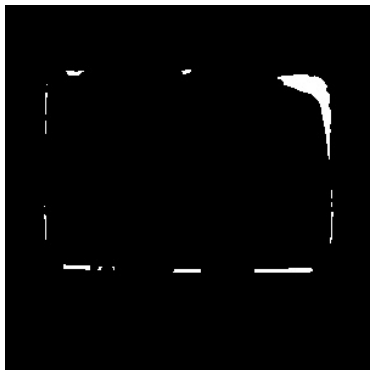
	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inKernel	const Region&		Kernel shape (any)
→	inKernelAnchor	Optional<const Location&>	NIL	Location within inKernel, defining its center
←	outRegion	Region&		Output region

Description

The operation is a cousin of the ErodeRegion filter, yet it uses any proper region selected by user as a structuring element. The center of a kernel is assumed to be at location (width/2,height/2). Refer to the ErodeRegion article for further information regarding the morphological erosion itself.

Examples





Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty kernel on input in ErodeRegion_AnyKernel.

See Also

- ErodeRegion – Performs a morphological erosion on a region using a predefined kernel.

ErodeRegion_Threshold

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Erodes a region with inThresholdValue pixels of inKernel.

Syntax

```
void avl::ErodeRegion_Threshold
(
 const avl::Region& inRegion,
 const avl::Region& inKernel,
 atl::Optional<const avl::Location&> inKernelAnchor,
 const int inThresholdValue,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inRegion	const Region&			Input region
→	inKernel	const Region&			
→	inKernelAnchor	Optional<const Location&>		NIL	
→	inThresholdValue	const int	0 - ∞		
←	outRegion	Region&			Output region

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty kernel on input in ErodeRegion_Threshold.

ExpandRegions

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Splits pixels of the input regions and their complement among these regions.

Applications: Use this filter to make sure that every pixel belongs to a region.

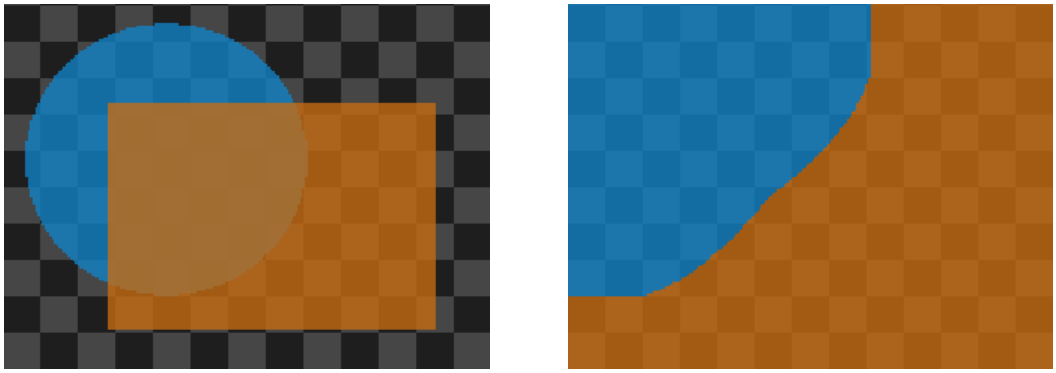
Syntax

```
void avl::ExpandRegions
(
 const atl::Array<avl::Region>& inRegions,
 atl::Optional<avl::Metric::Type> inMetric,
 atl::Array<avl::Region>& outRegions
)
```

Parameters

	Name	Type	Default	Description
➡	inRegions	const Array<Region>&		
➡	inMetric	Optional<Metric::Type>	NIL	Metric used for deciding which region receives a pixel; if set to NIL, Taxi metric is used after demarcating with NIL metric
⬅	outRegions	Array<Region>&		

Examples



*ExpandRegions performed with **inMetric** = Taxi.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Dimensions of two or more regions differ in ExpandRegions.
DomainError	Input and output arrays are not distinct in ExpandRegions.

OpenRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Performs a morphological opening on a region using a predefined kernel.

Applications: Removing small parts from a region without making it thinner.

Syntax

```
void avl::OpenRegion
(
 const avl::Region& inRegion,
 avl::KernelShape::Type inKernel,
 int inRadiusX,
 atl::Optional<int> inRadiusY,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inRegion	const Region&			Input region
➡	inKernel	KernelShape::Type			Kernel shape (predefined)
➡	inRadiusX	int	0 - ∞	1	Nearly half of the kernel's width (2*R+1)
➡	inRadiusY	Optional<int>	0 - ∞	NIL	Nearly half of the kernel's height (2*R+1), or same as inRadiusX
⬅	outRegion	Region&			Output region

Description

The operation performs a morphological opening, which is a tool used for removing thin parts from a region. The operation is a convolution of two basic morphological operations:

- Firstly, the input region is eroded using ErodeRegion operation.
- Then, the resulting region is dilated using DilateRegion operation.

During the erosion thin parts of a region are eliminated, while further dilation assures that the width of region limbs is preserved.

Both of the component operations are conducted using the same **inKernel**, **inRadiusX** and **inRadiusY** parameters.

Hints

- Increase **inRadiusX** to remove more small parts from the region.
- Change **inKernel** to Ellipse to make the filter work equally strongly in each direction (execution will be slower).

Examples

*OpenRegion run with **inKernel** = Ellipse of dimensions **inRadiusX** = 3, **inRadiusY** = 3.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Unsupported kernel in OpenRegion.

See Also

- ErodeRegion – Performs a morphological erosion on a region using a predefined kernel.
- OpenRegion_AnyKernel – Performs a morphological opening on a region using an arbitrary kernel.
- CloseRegion – Performs a morphological closing on a region using selected predefined kernel.

OpenRegion_AnyKernel

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Performs a morphological opening on a region using an arbitrary kernel.

Applications: Removing small parts of a particular shape without making the region thinner.

Syntax

```
void avl::OpenRegion_AnyKernel
(
 const avl::Region& inRegion,
 const avl::Region& inKernel,
 avl::Region& outRegion
)
```

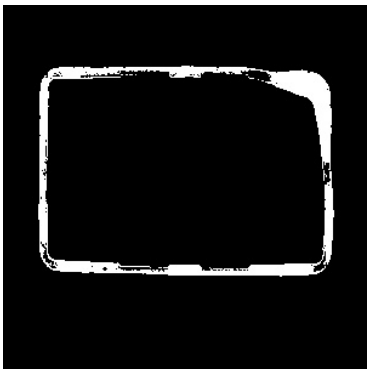
Parameters

	Name	Type	Default	Description
➡	inRegion	const Region&		Input region
➡	inKernel	const Region&		Kernel shape (any)
⬅	outRegion	Region&		Output region

Description

The operation is a cousin of the OpenRegion filter, yet it uses any proper region selected by user as a structuring element. The center of a kernel is assumed to be at location (width/2,height/2). Refer to the OpenRegion article for further information regarding the morphological opening itself.

Examples



Errors

List of possible exceptions:

Error type	Description
DomainError	Empty kernel on input in OpenRegion_AnyKernel.

See Also

- OpenRegion – Performs a morphological opening on a region using a predefined kernel.

PruneRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Removes one pixel wide branches from a region.

Syntax

```
void avl::PruneRegion
(
 const avl::Region& inRegion,
 const int inMaxLength,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inRegion	const Region&			Input region
→	inMaxLength	const int	0 - ∞	3	Maximal length of a branch of the input region to be pruned
←	outRegion	Region&			Output region

Description

The operation removes all branches with length at most **inMaxLength** from the input region. The branches being removed have to be at most one pixel wide, so the filter should be invoked on thinned regions such as the results of SkeletonizeRegion operation only.

Examples

PruneRegion *run on a sample one pixel wide region with* ***inMaxLength*** *= 58.*

See Also

- SkeletonizeRegion – Thins a region to its skeleton.

RegionHitAndMissTransform

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Performs a hit-and-miss transformation on a region using arbitrary kernels.

Syntax

```
void avl::RegionHitAndMissTransform
(
 const avl::Region& inRegion,
 const avl::Region& inHitKernel,
 atl::Optional<const avl::Location&> inHitKernelAnchor,
 const avl::Region& inMissKernel,
 atl::Optional<const avl::Location&> inMissKernelAnchor,
 avl::Region& outRegion
)
```

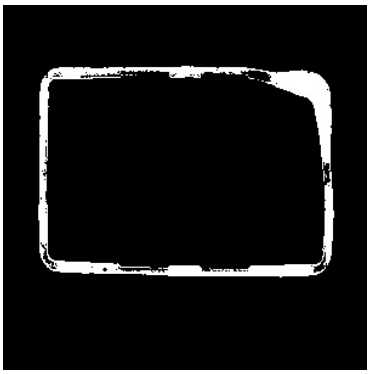
Parameters

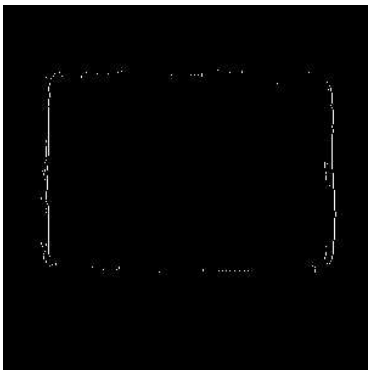
	Name	Type	Default	Description
➡	inRegion	const Region&		Input region
➡	inHitKernel	const Region&		The kernel of pixels that should be present
➡	inHitKernelAnchor	Optional<const Location&>	NIL	Location withing inHitKernel, defining its center
➡	inMssKernel	const Region&		The kernel of pixels that should be missing
➡	inMssKernelAnchor	Optional<const Location&>	NIL	Location withing inMssKernel, defining its center
⬅	outRegion	Region&		Output region

Description

The operation performs a hit-and-miss transformation. Similarly to other region morphology operations, hit-and-miss is conducted using a shape called kernel (or structuring element). In this particular case, there are two kernels: **inHitKernel** and **inMissKernel**. Both of them are repeatedly centered at each location within the dimensions of the input region. Then, pixel location **L** is added to the resulting region if and only if all of the **inHitKernel** pixels centered at **L** lie inside and all of the **inMissKernel** pixels centered at **L** lie outside the input region.

Examples





RegionHitAndMissTransform *run with kernels* $\begin{bmatrix} 1 & x & x \\ 1 & 0 & x \\ 1 & x & x \end{bmatrix}$ *where* `1`*'s represent pixels from* ***inHitKernel****,* `0`*'s from* ***inMissKernel*** *and* `x`*'s the other ones.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Hit kernel empty in RegionHitAndMssTransform.
DomainError	Inconsistent region dimensions in RegionHitAndMssTransform.
DomainError	Mss kernel empty in RegionHitAndMssTransform.

See Also

- ErodeRegion_AnyKernel – Performs a morphological erosion on a region using an arbitrary kernel.
- DilateRegion_AnyKernel – Performs a morphological dilation on a region using an arbitrary kernel.

SkeletonizeRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Thins a region to its skeleton.

Syntax

```
void avl::SkeletonizeRegion
(
 const avl::Region& inRegion,
 avl::RegionSkeletonMethod::Type inRegionSkeletonMethod,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inRegionSkeletonMethod	RegionSkeletonMethod::Type	TwelveConnected	
←	outRegion	Region&		Output region

Description

The operation performs skeletonization presenting result as a region.

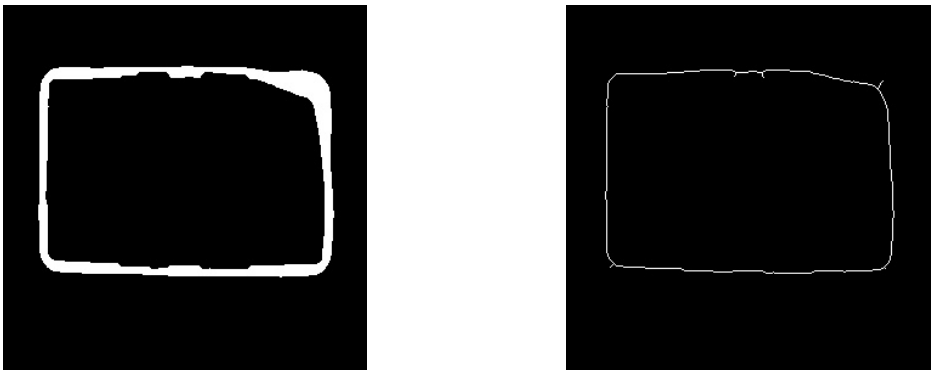
Skeleton of a region is a connected set of medial axis of its limbs. It is a useful tool when one is interested only in general structure of a shape and wants to disregard width of its limbs. Two methods are available, depending on the value of **inRegionSkeletonMethod** being chosen:

- **EightConnected**: the input region is thinned as long as the thinning procedure affects it; the structuring elements come from the well known Golay alphabet
- **TwelveConnected**: the input region is thinned also, but the structuring elements being used are slightly larger than in previous method (they consider 12-neighborhood of the given pixel instead of 8-neighborhood). The method comes from the paper of U. Eckhardt and G. Maderlechner "Invariant thinning"

The second method is slower than the first one, but its results look in most cases much better than the first one's results.

This filter is a cousin of RegionMedialAxis which represents the results as an array of paths instead of a region.

Examples



SkeletonizeRegion *run on a sample region with* ***inRegionSkeletonMethod*** *=* ***TwelveConnected****.*

See Also

- RegionMedialAxis – Computes an array of paths corresponding to the skeleton of the input region.

ThickenRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Performs a morphological thickening on a region using predefined kernels.

Syntax

```
void avl::ThickenRegion
(
 const avl::Region& inRegion,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outRegion	Region&		Output region

Description

The operation performs a morphological thickening. Similarly to other region morphology operations, thickening is conducted using a shape called kernel (or structuring element). There are two kernels: hit kernel and miss kernel, which are passed as parameters to RegionHitAndMissTransform. This operation's result is then added to the input region giving the output region. The filter uses internally predefined kernels, which are dual to those from ThinRegion filter.

See Also

- ThinRegion – Performs a morphological thinning on a region using predefined kernels.
- RegionHitAndMissTransform – Performs a hit-and-miss transformation on a region using arbitrary kernels.
- DilateRegion_AnyKernel – Performs a morphological dilation on a region using an arbitrary kernel.

ThinRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Performs a morphological thinning on a region using predefined kernels.

Syntax

```
void avl::ThinRegion
(
 const avl::Region& inRegion,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outRegion	Region&		Output region

Description

The operation performs a morphological thinning. Similarly to other region morphology operations, thinning is conducted using a shape called kernel (or structuring element). There are two kernels: hit kernel and miss kernel, which are passed as parameters to RegionHitAndMissTransform. This operation's result is then subtracted from the input region giving the output region. The filter uses internally predefined kernels, which are the same as in SkeletonizeRegion filter.

See Also

- SkeletonizeRegion – Thins a region to its skeleton.
- ThickenRegion – Performs a morphological thickening on a region using predefined kernels.
- RegionHitAndMissTransform – Performs a hit-and-miss transformation on a region using arbitrary kernels.
- ErodeRegion_AnyKernel – Performs a morphological erosion on a region using an arbitrary kernel.

ThresholdSmoothedRegion_Mean

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Returns region of pixels witch collect minAmount of pixels in rectangle in input region.

Syntax

```
void avl::ThresholdSmoothedRegion_Mean
(
 const avl::Region& inRegion,
 int inRadiusX,
 atl::Optional<int> inRadiusY,
 float inMinCount,
 int& outMinAmount,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inRegion	const Region&			Input region
→	inRadiusX	int			Nearly half of the kernel's width (2*R+1)
→	inRadiusY	Optional<int>		NIL	Nearly half of the kernel's height (2*R+1), or same as inRadiusX
→	inMinCount	float	0.0 - 1.0		Minimal part of pixels in rectangle to be in region
←	outMinAmount	int&			
←	outRegion	Region&			Output region

TopHatRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Performs a morphological white top hat operation on a region using a predefined kernel.

Syntax

```
void avl::TopHatRegion
(
 const avl::Region& inRegion,
 avl::KernelShape::Type inKernel,
 int inRadiusX,
 atl::Optional<int> inRadiusY,
 avl::Region& outRegion
)
```

Parameters

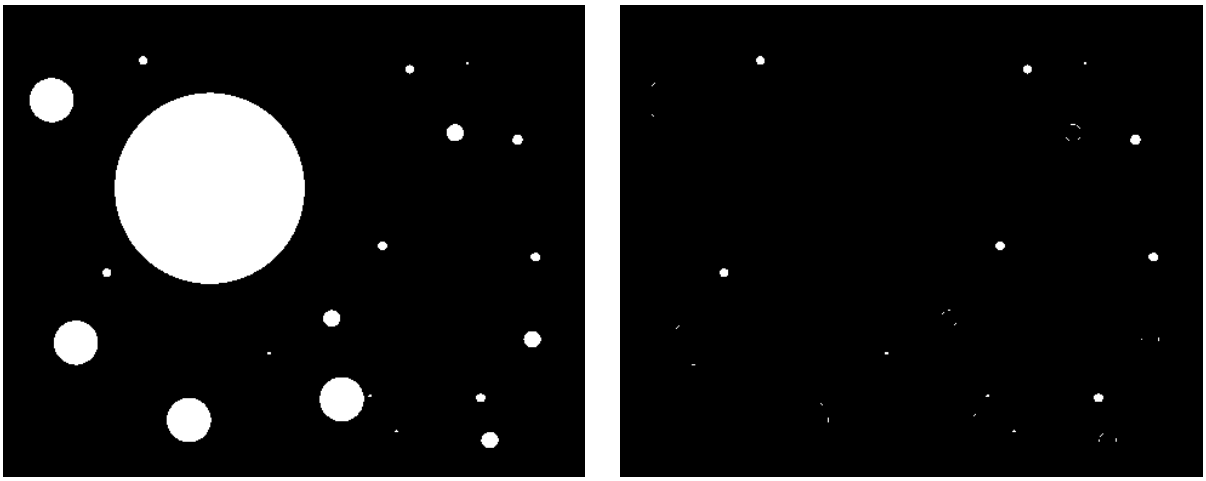
	Name	Type	Range	Default	Description
➡	inRegion	const Region&			Input region
➡	inKernel	KernelShape::Type			Kernel shape (predefined)
➡	inRadiusX	int	0 - ∞	1	Nearly half of the kernel's width (2*R+1)
➡	inRadiusY	Optional<int>	0 - ∞	NIL	Nearly half of the kernel's height (2*R+1), or same as inRadiusX
⬅	outRegion	Region&			Output region

Description

Extracts from region small parts. Uses predefined kernels.

Is performed by running consecutively two filters. OpenRegion to remove small parts from region and RegionDifference to remove rest of the region.

Examples



*Top Hat used to remove bigger parts of region. Used parameters **inKernel**=Ellipse and **inRadiusX**=5. Region marked white. Source image on the left and result on the right.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Unsupported kernel in TopHatRegion.

TopHatRegion_AnyKernel

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Performs a morphological white top hat operation on a region using an arbitrary kernel.

Syntax

```
void avl::TopHatRegion_AnyKernel
(
 const avl::Region& inRegion,
 const avl::Region& inKernel,
 avl::Region& outRegion
)
```

Parameters

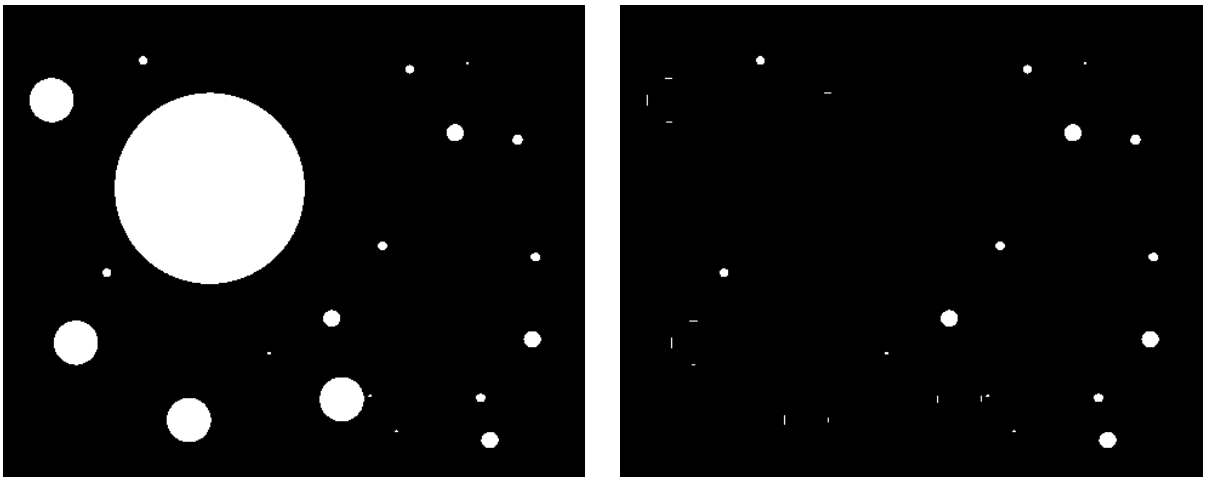
	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inKernel	const Region&		Kernel shape (any)
←	outRegion	Region&		Output region

Description

Extracts from region small parts. Uses user-defined kernels.

Is performed by running consecutively two filters. OpenRegion to remove small parts from region and RegionDifference to remove rest of the region.

Examples



Top Hat used to remove bigger parts of region. Region marked white. Source image on the left and result on the right.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty kernel on input in TopHatRegion_AnyKernel.

54. Geometry 3D Features

Table of content:

Box3DCenter

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Returns the center of the input box in 3D.

Syntax

```
void avl::Box3DCenter
(
 const avl::Box3D& inBox3D,
 avl::Point3D& outCenter
)
```

Parameters

	Name	Type	Default	Description
→	inBox3D	const Box3D&		
←	outCenter	Point3D&		

Box3DCharacteristicPoint

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Returns a characteristic point of a box in 3D.

Syntax

```
void avl::Box3DCharacteristicPoint
(
 const avl::Box3D& inBox3D,
 const avl::Anchor3D& inPoint3DAnchor,
 avl::Point3D& outPoint3D
)
```

Parameters

	Name	Type	Default	Description
→	inBox3D	const Box3D&		
→	inPoint3DAnchor	const Anchor3D&		Selecting one of the 27 characteristic points
←	outPoint3D	Point3D&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Invalid Anchor3D in Box3DCharacteristicPoint.

Box3DSurfaceArea

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Returns the surface area of the input box in 3D.

Syntax

```
void avl::Box3DSurfaceArea
(
 const avl::Box3D& inBox3D,
 float& outSurfaceArea
)
```

Parameters

	Name	Type	Default	Description
→	inBox3D	const Box3D&		
←	outSurfaceArea	float&		

Box3DVolume

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Returns the volume of the input box in 3D.

Syntax

```
void avl::Box3DVolume
(
 const avl::Box3D& inBox3D,
 float& outVolume
)
```

Parameters

	Name	Type	Default	Description
➡	inBox3D	const Box3D&		
⬅	outVolume	float&		

Boxes3DBoundingBox3D_OrNil

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes the bounding box 3D of given boxes in 3D; returns NIL if the array is empty.

Syntax

```
void avl::Boxes3DBoundingBox3D_OrNil
(
 const atl::Array<avl::Box3D>& inBoxes3D,
 atl::Conditional<avl::Box3D>& outBoundingBox3D
)
```

Parameters

	Name	Type	Default	Description
➡	inBoxes3D	const Array<Box3D>&		
⬅	outBoundingBox3D	Conditional<Box3D>&		

BoxesBoundingBox3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes the bounding box 3D of given boxes in 3D.

Syntax

```
void avl::BoxesBoundingBox3D
(
 const atl::Array<avl::Box3D>& inBoxes3D,
 avl::Box3D& outBoundingBox3D
)
```

Parameters

	Name	Type	Default	Description
➡	inBoxes3D	const Array<Box3D>&		
⬅	outBoundingBox3D	Box3D&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array on input in BoxesBoundingBox3D.

Circle3DArea

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes the area of a circle in 3D.

Syntax

```
void avl::Circle3DArea
(
 const avl::Circle3D& inCircle3D,
 float& outArea
)
```

Parameters

	Name	Type	Default	Description
→	inCircle3D	const Circle3D&		
←	outArea	float&		

Circle3DBoundingBox

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes the smallest box in 3D containing a circle in 3D.

Syntax

```
void avl::Circle3DBoundingBox
(
 const avl::Circle3D& inCircle3D,
 avl::Box3D& outBoundingBox3D
)
```

Parameters

	Name	Type	Default	Description
→	inCircle3D	const Circle3D&		
←	outBoundingBox3D	Box3D&		

Circle3DPerimeterLength

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes the length of a circle in 3D perimeter.

Syntax

```
void avl::Circle3DPerimeterLength
(
 const avl::Circle3D& inCircle3D,
 float& outPerimeterLength
)
```

Parameters

	Name	Type	Default	Description
→	inCircle3D	const Circle3D&		
←	outPerimeterLength	float&		

Plane3DNormalVector

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes the normal vector of a line.

Syntax

```
void avl::Plane3DNormalVector
(
 const avl::Plane3D& inPlane,
 avl::Vector3D& outNormalVector
)
```

Parameters

	Name	Type	Default	Description
→	inPlane	const Plane3D&		
←	outNormalVector	Vector3D&		

Plane3DOrientation

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes the orientation of a plane as angles in the range from 0 to 180 between the plane and coordinate planes.

Syntax

```
void avl::Plane3DOrientation
(
 const avl::Plane3D& inPlane,
 float& outAngleXY,
 float& outAngleXZ,
 float& outAngleYZ,
 atl::Optional<avl::Point3D&> outPointOnPlane = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPlane	const Plane3D&		
←	outAngleXY	float&		
←	outAngleXZ	float&		
←	outAngleYZ	float&		
←	outPointOnPlane	Optional<Point3D&>	NIL	Projection of the beginning of the coordinate axes on the input plane

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outPointOnPlane**.

Read more about Optional Outputs.

Points3DMedian

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes the geometric median of the input points.

Syntax

```
void avl::Points3DMedian
(
 const atl::Array<avl::Point3D>& inPoints,
 atl::Optional<const atl::Array<float>&> inWeights,
 const int inMaxIterationCount,
 avl::Point3D& outGeometricMedian,
 atl::Optional<float&> outDistanceSum = atl::NIL,
 atl::Array<avl::Point3D>& diagApproximationSteps = atl::Dummy<atl::Array<Point3D>>()
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoints	const Array<Point3D>&			Input points
→	inWeights	Optional<const Array<float>&>		NIL	Optional input weights
→	inMaxIterationCount	const int	1 - ∞	10	Maximum number of iterations
←	outGeometricMedian	Point3D&			Geometric median
←	outDistanceSum	Optional<float&>		NIL	Sum of distances from input points to geometric median
🔍	diagApproximationSteps	Array<Point3D>&			Approximate geometric medians calculated during subsequent iterations

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outDistanceSum**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Input arrays must be of the same size in Points3DMedian.
DomainError	Input point array is empty in Points3DMedian.

PointsBoundingBox3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes the smallest box in 3D containing an array of points in 3D.

Syntax

```
void avl::PointsBoundingBox3D
(
 const atl::Array<avl::Point3D>& inPoints,
 avl::Box3D& outBoundingBox3D
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point3D>&		
←	outBoundingBox3D	Box3D&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array on input in PointsBoundingBox3D.

PointsBoundingBox3D_OrNil

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes the smallest box in 3D containing an array of points in 3D; returns NIL if the array is empty.

Syntax

```
void avl::PointsBoundingBox3D_OrNil
(
 const atl::Array<avl::Point3D>& inPoints,
 atl::Conditional<avl::Box3D>& outBoundingBox3D
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point3D>&		
←	outBoundingBox3D	Conditional<Box3D>&		

PointsMassCenter3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes the central point of the input points.

Syntax

```
void avl::PointsMassCenter3D
(
 const atl::Array<avl::Point3D>& inPoints,
 avl::Point3D& outMassCenter
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point3D>&		Input array of points
←	outMassCenter	Point3D&		Central point of input points

Errors

List of possible exceptions:

Error type	Description
DomainError	Input point array is empty in PointsMassCenter3D.

Segment3DBisector

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes a plane passing through the center of a segment in 3D at a right angle.

Syntax

```
void avl::Segment3DBisector
(
 const avl::Segment3D& inSegment3D,
 avl::Plane3D& outBisector
)
```

Parameters

	Name	Type	Default	Description
→	inSegment3D	const Segment3D&		
←	outBisector	Plane3D&		

Segment3DBoundingBox

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes the smallest box in 3D containing a segment in 3D.

Syntax

```
void avl::Segment3DBoundingBox
(
 const avl::Segment3D& inSegment3D,
 avl::Box3D& outBoundingBox3D
)
```

Parameters

	Name	Type	Default	Description
→	inSegment3D	const Segment3D&		
←	outBoundingBox3D	Box3D&		

Segment3DCenter

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes the center point of a segment in 3D.

Syntax

```
void avl::Segment3DCenter
(
 const avl::Segment3D& inSegment3D,
 avl::Point3D& outCenterPoint
)
```

Parameters

	Name	Type	Default	Description
→	inSegment3D	const Segment3D&		
←	outCenterPoint	Point3D&		

Segment3DLength

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes the length of a segment in 3D.

Syntax

```
void avl::Segment3DLength
(
 const avl::Segment3D& inSegment3D,
 float& outLength
)
```

Parameters

	Name	Type	Default	Description
→	inSegment3D	const Segment3D&		
←	outLength	float&		

Segment3DLine

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes the line in 3D passing through a segment.

Syntax

```
void avl::Segment3DLine
(
 const avl::Segment3D& inSegment3D,
 avl::Line3D& outLine3D
)
```

Parameters

	Name	Type	Default	Description
→	inSegment3D	const Segment3D&		Input segment in 3D
←	outLine3D	Line3D&		The resulting line

Segment3DVector

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Returns the vector [x2 - x1, y2 - y1, z2 - z1].

Syntax

```
void avl::Segment3DVector
(
 const avl::Segment3D& inSegment3D,
 avl::Vector3D& outVector3D
)
```

Parameters

	Name	Type	Default	Description
→	inSegment3D	const Segment3D&		
←	outVector3D	Vector3D&		

Vector3DAzimuth

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes the angle between a 3D vector's projection to the XY plane and the X axis measured toward Y axis, as an angle in the range from 0 to 360.

Syntax

```
void avl::Vector3DAzimuth
(
 const avl::Vector3D& inVector3D,
 float& outAzimuth
)
```

Parameters

	Name	Type	Default	Description
→	inVector3D	const Vector3D&		
←	outAzimuth	float&		

Vector3DElevation

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes the angle between a 3D vector's projection to the XY plane and itself measured toward Z axis, as an angle in the range from -90 to 90.

Syntax

```
void avl::Vector3DElevation
(
 const avl::Vector3D& inVector3D,
 float& outElevation
)
```

Parameters

	Name	Type	Default	Description
→	inVector3D	const Vector3D&		
←	outElevation	float&		

Vector3DLength

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes the length of a 3D vector.

Syntax

```
void avl::Vector3DLength
(
 const avl::Vector3D& inVector3D,
 float& outLength
)
```

Parameters

	Name	Type	Default	Description
→	inVector3D	const Vector3D&		
←	outLength	float&		

Vectors3DMedian

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes the geometric median of the input vectors.

Syntax

```
void avl::Vectors3DMedian
(
 const atl::Array<avl::Vector3D>& inVectors,
 atl::Optional<const atl::Array<float>&> inWeights,
 const int inMaxIterationCount,
 avl::Vector3D& outGeometricMedian,
 atl::Optional<float&> outDistanceSum = atl::NIL,
 atl::Array<avl::Vector3D>& diagApproximationSteps = atl::Dummy<atl::Array<Vector3D>>()
)
```

Parameters

	Name	Type	Range	Default	Description
→	inVectors	const Array<Vector3D>&			Input vectors
→	inWeights	Optional<const Array<float>&>		NIL	Optional input weights
→	inMaxIterationCount	const int	1 - ∞	10	Maximum number of iterations
←	outGeometricMedian	Vector3D&			Geometric median
←	outDistanceSum	Optional<float&>		NIL	Sum of distances from input vectors to geometric median
🔍	diagApproximationSteps	Array<Vector3D>&			Approximate geometric medians calculated during subsequent iterations

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outDistanceSum**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Input arrays must be of the same size in Vectors3DMedian.
DomainError	Input vector array is empty in Vectors3DMedian.

55. Geometry 3D Intersections

Table of content:

BoxIntersection3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes the common part of two boxes in 3D.

Syntax

```
void avl::BoxIntersection3D
(
 const avl::Box3D& inBox3D1,
 const avl::Box3D& inBox3D2,
 avl::Box3D& outBox3D
)
```

Parameters

	Name	Type	Default	Description
→	inBox3D1	const Box3D&		
→	inBox3D2	const Box3D&		
←	outBox3D	Box3D&		

CirclePlaneIntersection3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes common points of a circle in 3D and a plane.

Syntax

```
void avl::CirclePlaneIntersection3D
(
 const avl::Circle3D& inCircle3D,
 const avl::Plane3D& inPlane,
 atl::Conditional<avl::Point3D>& outIntersectionPoint1,
 atl::Conditional<avl::Point3D>& outIntersectionPoint2
)
```

Parameters

	Name	Type	Default	Description
→	inCircle3D	const Circle3D&		
→	inPlane	const Plane3D&		
←	outIntersectionPoint1	Conditional<Point3D>&		
←	outIntersectionPoint2	Conditional<Point3D>&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite circle on input in CirclePlaneIntersection3D.
DomainError	Indefinite plane on input in CirclePlaneIntersection3D.

LineLineIntersection3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes a common point of two lines in 3D.

Syntax

```
void avl::LineLineIntersection3D
(
 const avl::Line3D& inLine1,
 const avl::Line3D& inLine2,
 atl::Conditional<avl::Point3D>& outIntersectionPoint
)
```

Parameters

	Name	Type	Default	Description
➡	inLine1	const Line3D&		
➡	inLine2	const Line3D&		
⬅	outIntersectionPoint	Conditional<Point3D>&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite line on input in LineLineIntersection3D.

LinePlaneIntersection3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes the intersection point of a line in 3D and a plane.

Syntax

```
void avl::LinePlaneIntersection3D
(
 const avl::Line3D& inLine3D,
 const avl::Plane3D& inPlane,
 atl::Conditional<avl::Point3D>& outIntersectionPoint
)
```

Parameters

	Name	Type	Default	Description
➡	inLine3D	const Line3D&		
➡	inPlane	const Plane3D&		
⬅	outIntersectionPoint	Conditional<Point3D>&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite line on input in LinePlaneIntersection3D.
DomainError	Indefinite plane on input in LinePlaneIntersection3D.

PlanePlaneIntersection3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes a common line in 3D of two planes.

Syntax

```
void avl::PlanePlaneIntersection3D
(
 const avl::Plane3D& inPlane1,
 const avl::Plane3D& inPlane2,
 atl::Conditional<avl::Line3D>& outIntersectionLine
)
```

Parameters

	Name	Type	Default	Description
➡	inPlane1	const Plane3D&		
➡	inPlane2	const Plane3D&		
⬅	outIntersectionLine	Conditional<Line3D>&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite plane on input in PlanePlaneIntersection3D.

SegmentPlaneIntersection3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes the intersection point of a segment in 3D and a plane.

Syntax

```
void avl::SegmentPlaneIntersection3D
(
 const avl::Segment3D& inSegment3D,
 const avl::Plane3D& inPlane,
 atl::Conditional<avl::Point3D>& outIntersectionPoint
)
```

Parameters

	Name	Type	Default	Description
➡	inSegment3D	const Segment3D&		
➡	inPlane	const Plane3D&		
⬅	outIntersectionPoint	Conditional<Point3D>&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite plane on input in SegmentPlaneIntersection3D.

56. Image Thresholding

Table of content:

SelectThresholdValue

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Selects best threshold value using the image histogram.

Applications: Usually used before a thresholding filter when the image brightness is variable. Use with care.

Syntax

```
void avl::SelectThresholdValue
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::ThresholdSelectionMethod::Type inMethod,
 float& outThresholdValue,
 atl::Array<float>& diagThresholdRatings,
 atl::Array<float>& diagBackgroundPixelsFraction,
 atl::Array<float>& diagForegroundPixelsFraction
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inMethod	ThresholdSelectionMethod::Type		Method used to select the best threshold
←	outThresholdValue	float&		Best threshold separating background pixels from foreground pixels
🔍	diagThresholdRatings	Array<float>&		Contains ratings gained if that threshold would be chosen
🔍	diagBackgroundPixelsFraction	Array<float>&		Fraction of pixels that are darker than the index value
🔍	diagForegroundPixelsFraction	Array<float>&		Fraction of pixels that are brighter than the index value

Requirements

For input **inImage** only pixel formats are supported: uint8.

Read more about pixel formats in Image documentation.

Description

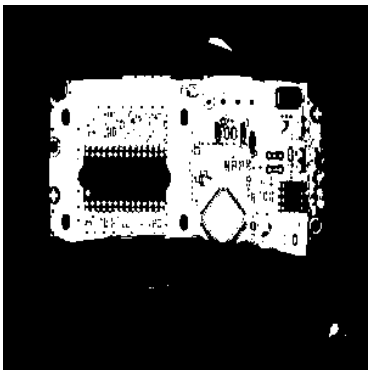
The operation estimates an intensity threshold value that discriminates foreground pixels from background pixels in **inImage**.

Note that to provide meaningful results it is required that the image in fact consists of foreground and background components that strongly differ in brightness.

Hints

- Choose **inMethod** that works best in your application. None of the methods is perfect, but **ClusteringOtsu** is the most universal one.

Examples



SelectThresholdValue *performed on the sample image (on the left) with* ***inMethod*** *= ClusteringKittler yields* ***outThresholdValue*** *= 189.5. Pixels of the input image brighter than 189.5 are presented on the right.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in SelectThresholdValue.
DomainError	Not supported inImage pixel format in SelectThresholdValue. Supported formats: UInt8.

See Also

- ThresholdImage – Transforms each pixel value to maximum or minimum depending on whether they belong to the specified range.

ThresholdImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Transforms each pixel value to maximum or minimum depending on whether they belong to the specified range.

Applications: Image binarization when the illumination is constant and uniform.

Syntax

```
void avl::ThresholdImage
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<float> inMinValue,
 atl::Optional<float> inMaxValue,
 float inFuzziness,
 avl::Image& outMonoImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Region of interest
→	inMinValue	Optional<float>		128.0f	Minimum value of a pixel that is considered foreground (Auto = -INF)
→	inMaxValue	Optional<float>		NIL	Maximum value of a pixel that is considered foreground (Auto = +INF)
→	inFuzziness	float	0.0 - ∞	0.0f	A tolerance for inMin/MaxValue that results in intermediate output values
←	outMonoImage	Image&			

Description

The operation transforms each pixel value to the maximum or minimum level thus creating binary image. The result of the transformation depends on the pixel intensity:

- Pixel values in range (**inMinValue**, **inMaxValue**) are transformed to the maximum level.
- Other pixel values are transformed to the minimum level.

If any of the parameters **inMinValue**, **inMaxValue** is not set, it is assumed to be, accordingly, *-infinity* or *infinity*.

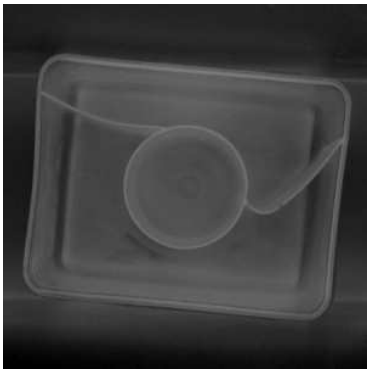
Parameter **inFuzziness** (set to 0 by default) allows to perform fuzzy thresholding which linearly interpolates those pixel values that differ by at most **inFuzziness** from the values **inMinValue**, **inMaxValue**; thus creating smooth transition between minimum and maximum values in the resulting image (see **Remarks** section).

In the multichannel images the operation uses an average of channel values in each pixel, thus the resulting image is always monochromatic.

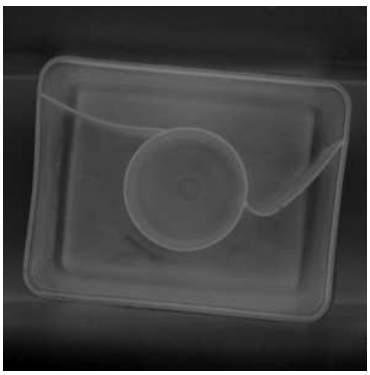
Hints

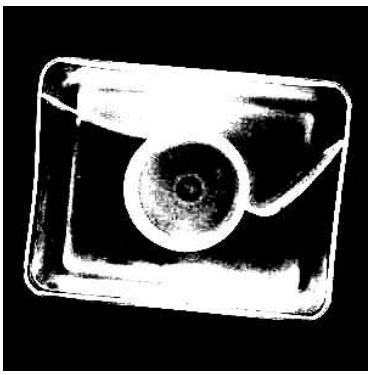
- Define **inMinValue** to obtain white pixels for bright objects (brighter than the specified value).
- Define **inMaxValue** to obtain white pixels for dark objects (darker than the specified value).
- Use **inFuzziness** to add some smooth transitions between black and white pixels in the result.

Examples



ThresholdImage *performed on the sample image with* ***inMinValue*** *= 80.0,* ***inMaxValue*** *= auto,* ***inFuzziness*** *= 0.0.*





ThresholdImage *performed on the sample image with* ***inMinValue*** *= 80.0,* ***inMaxValue*** *= auto,* ***inFuzziness*** *= 2.5.*

Remarks

When image pixel type is Real, parameter **inFuzziness** has no influence on the output (is ignored). The reason is strictly mathematical. Namely, there does not exist a smooth transition between *-infinity* and *+infinity* values, which are, correspondingly, minimum and maximum levels of pixel brightness.

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: 1xUINT8 (for inFuzziness = 0), 3xUINT8 (for inFuzziness = 0).

This operation is optimized for AVX2 technology for pixels of types: 1xUINT8 (for inFuzziness = 0), 3xUINT8 (for inFuzziness = 0).

This operation is optimized for NEON technology for pixels of types: 1xUINT8 (for inFuzziness = 0), 3xUINT8 (for inFuzziness = 0).

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in ThresholdImage.

See Also

- ThresholdToRegion – Creates a region containing image pixels with values within the specified range.

ThresholdImage_Color

Header: AVL.h
Namespace: avl
Module: FoundationPro

Transforms each pixel value to maximum or minimum depending on the distance from a given color.

Applications: Color analysis with a given reference color.

Syntax

```
void avl::ThresholdImage_Color
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 const avl::Pixel& inRgbColor,
 float inChromaAmount,
 float inMaxDifference,
 float inFuzziness,
 avl::Image& outMonoImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Region of interest
→	inRgbColor	const Pixel&			Color to compare the image to
→	inChromaAmount	float	0.0 - 1.0	0.7f	Proportion of chromatic information in distance computation
→	inMaxDifference	float	0.0 - ∞	5.0f	Maximum difference between image pixel and model color
→	inFuzziness	float	0.0 - ∞	0.0f	A tolerance for computed difference that results in intermediate output values
←	outMonoImage	Image&			

Requirements

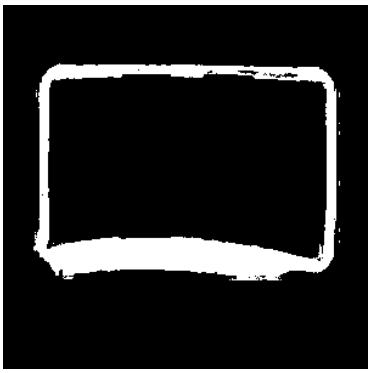
For input **inImage** only pixel formats are supported: 1□uint8, 2□uint8, 3□uint8, 4□uint8.

Read more about pixel formats in Image documentation.

Hints

- Define the reference color by setting the **inRgbColor** input.
- Increase **inChromaAmount** to make the function less sensitive to changes in brightness. Decrease it to make brightness more important.
- Set **inMaxDifference** experimentally to a value that best separates the foreground and background pixels.
- Use **inFuzziness** to add some smooth transitions between black and white pixels in the result.

Examples



ThresholdImage_Color *performed on a sample image with* ***inRgbColor*** *= (192, 34, 22),* ***inChromaAmount*** *= 1.0,* ***inMaxDifference*** *= 48.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: 1xUINT8 (for inFuzziness = 0), 3xUINT8 (for inFuzziness = 0).

This operation is optimized for NEON technology for pixels of types: 1xUINT8 (for inFuzziness = 0), 3xUINT8 (for inFuzziness = 0).

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in ThresholdImage_Color.
DomainError	Not supported inImage pixel format in ThresholdImage_Color. Supported formats: 1xUInt8, 2xUInt8, 3xUInt8, 4xUInt8.

See Also

- ColorDistanceImage – Compares each pixel with the specified color using chromatic and non-chromatic information.

ThresholdImage_Dynamic

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Thresholds an image relatively to some value calculated in a local rectangular neighbourhood.

Applications: Image binarization when the illumination is uneven.

Syntax

```
void avl::ThresholdImage_Dynamic
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<const avl::Region&> inSourceRoi,
 avl::ThresholdDynamicReferenceMethod::Type inReferenceMethod,
 avl::KernelShape::Type inKernel,
 int inRadiusX,
 atl::Optional<int> inRadiusY,
 atl::Optional<float> inMinRelativeValue,
 atl::Optional<float> inMaxRelativeValue,
 float inFuzziness,
 avl::Image& outMonoImage,
 avl::Image& diagBaseImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Region in which pixels are written
→	inSourceRoi	Optional<const Region&>		NIL	Region from which pixels are read
→	inReferenceMethod	ThresholdDynamicReferenceMethod::Type		Mean	Specifies how the local threshold value will be calculated (see SmoothImage filter family)
→	inKernel	KernelShape::Type		Box	Kernel shape.
→	inRadiusX	int	0 - 65535	5	Horizontal radius of local neighbourhood.
→	inRadiusY	Optional<int>	0 - 65535	NIL	Vertical radius of local neighbourhood (Auto = inRadiusX)
→	inMinRelativeValue	Optional<float>		0.0f	Minimum relative value of a pixel that is considered foreground (Auto = -INF)
→	inMaxRelativeValue	Optional<float>		NIL	Maximum relative value of a pixel that is considered foreground (Auto = +INF)
→	inFuzziness	float	0.0 - ∞	0.0f	A tolerance for inMin/MaxRelativeValue that results in intermediate output values. Clamped on half of pixel max value (e.g. max fuzziness for uint8 image is 128).
←	outMonoImage	Image&			
🔍	diagBaseImage	Image&			Diagnostic threshold values.

Description

The operation transforms each pixel value to the maximum or minimum level thus creating binary image. The result of the transformation depends on the relative pixel intensity:

- Pixel values that are brighter than local average of the pixel neighbourhood by at least **inMinRelativeValue** and at most **inMaxRelativeValue** are transformed to the maximum level.
- Other pixel values are transformed to the minimum level.

If any of the parameters **inMinRelativeValue**, **inMaxRelativeValue** is not set, it is assumed to be, accordingly, *-infinity* or *infinity*.

Pixel neighbourhood used to compute the local average is a rectangle of dimensions $(2 \cdot \mathbf{inRadiusX} + 1) \times (2 \cdot \mathbf{inRadiusY} + 1)$ centered at the pixel being processed.

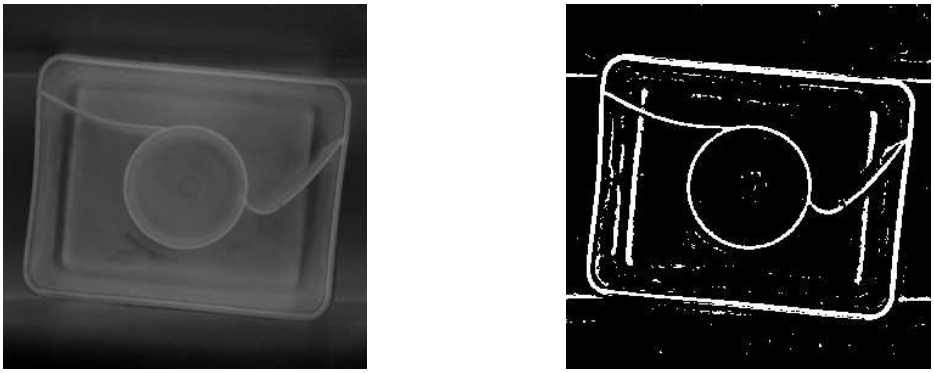
Parameter **inFuzziness** (set to 0 by default) allows to perform fuzzy thresholding which linearly interpolates those pixel values that differ by at most **inFuzziness** from the border intensities; thus creating smooth transition between minimum and maximum values in the resulting image.

In the multichannel images the operation uses an average of channel values in each pixel, thus the resulting image is always monochromatic.

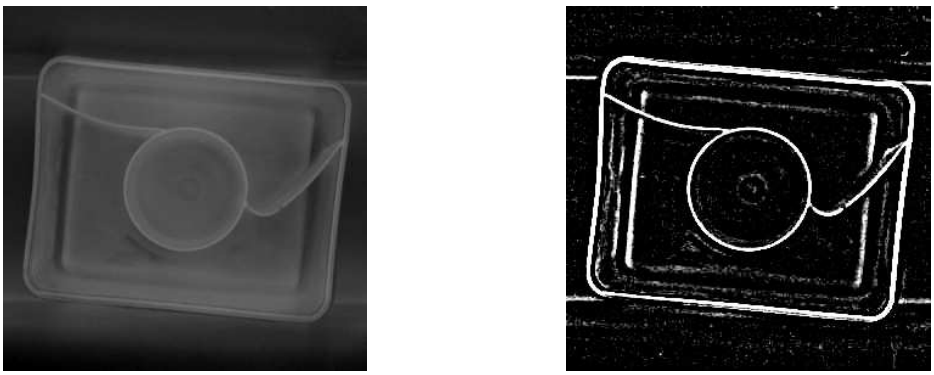
Hints

- Define **inMinRelativeValue** to obtain white pixels for objects which are brighter than the neighborhood.
- Define **inMaxRelativeValue** to obtain white pixels for objects which are darker than the neighborhood.
- Increase **inRadiusX** (and optionally **inRadiusY**) to define a bigger neighborhood.
- Use **inFuzziness** to add some smooth transitions between black and white pixels in the result.

Examples



ThresholdImage_Dynamic *performed on the sample image with* ***inMinRelativeValue*** *= 5.0,* ***inMaxRelativeValue*** *= auto,* ***inFuzziness*** *= 0.0.*



ThresholdImage_Dynamic *performed on the sample image with* ***inMinRelativeValue*** *= 5.0,* ***inMaxRelativeValue*** *= auto,* ***inFuzziness*** *= 10.0.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of type: 1xUINT8.

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect fuzziness value in ThresholdImage_Dynamic.
DomainError	Roi exceeds image dimensions in ThresholdImage_Dynamic.
DomainError	Source roi exceeds image dimensions in ThresholdImage_Dynamic.

See Also

- ThresholdToRegion_Dynamic – Thresholds an image relatively to the average pixel value in a local rectangular neighborhood.

ThresholdImage_HSx

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Transforms each pixel value to minimum or maximum depending on whether it belongs to specified region in the HSV, HSL or HSI color space.

Applications: Color analysis.

Syntax

```
void avl::ThresholdImage_HSx
(
 const avl::Image& inRgbImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::HSxColorModel::Type inColorModel,
 int inBeginHue,
 int inEndHue,
 atl::Optional<int> inMinSaturation,
 atl::Optional<int> inMaxSaturation,
 atl::Optional<int> inMinBrightness,
 atl::Optional<int> inMaxBrightness,
 float inFuzziness,
 avl::Image& outMonoImage,
 avl::Image& diagHSxImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inRgbImage	const Image&			Input image in the RGB color space
→	inRoi	Optional<const Region&>		NIL	Region of interest
→	inColorModel	HSxColorModel::Type			Selected color model
→	inBeginHue	int	0 - 255	0	Lowest acceptable Hue; if higher than inEndHue, then range wrapping is used
→	inEndHue	int	0 - 255	255	Highest acceptable Hue, if lower than inBeginHue, then range wrapping is used
→	inMinSaturation	Optional<int>	0 - 255	128	
→	inMaxSaturation	Optional<int>	0 - 255	NIL	
→	inMinBrightness	Optional<int>	0 - 255	128	Minimum brightness; denotes V, L or I, depending on inColorModel
→	inMaxBrightness	Optional<int>	0 - 255	NIL	Maximum brightness; denotes V, L or I, depending on inColorModel
→	inFuzziness	float	0.0 - ∞		Tolerance for value ranges that results in intermediate output values
←	outMonoImage	Image&			
	diagHSxImage	Image&			Image in HSx color space

Requirements

For input **inRgbImage** only pixel formats are supported: 3□uint8.

Read more about pixel formats in Image documentation.

Description

The operation transforms each pixel of three-channel **inRgbImage** to the maximum or minimum level thus creating binary image. The input image is considered to be encoded using RGB color representation. Each of the image pixel is internally converted to HSx (HSV, HSL or HSI) color representation and then examined.

- Pixels meeting all of the following conditions are transformed to the maximum level:
  - Value of the Hue parameter is in cyclic range (**inBeginHue**, **inEndHue**).
  - Value of the Saturation parameter is in range (**inMinSaturation**, **inMaxSaturation**).
  - Value of the Brightness parameter is in range (**inMinBrightness**, **inMaxBrightness**).
- Other pixels are transformed to the minimum level.

If any of the parameters **inMinSaturation**, **inMinBrightness** is not set, it is assumed to be *-infinity*.

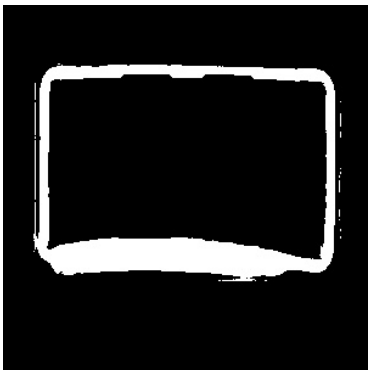
If any of the parameters **inMaxSaturation**, **inMaxBrightness** is not set, it is assumed to be *infinity*.

Parameter **inFuzziness** (set to 0 by default) allows to perform fuzzy thresholding which linearly interpolates those pixel values that differ by at most **inFuzziness** from the border parameter values; thus creating smooth transition between minimum and maximum values in the resulting image.

Hints

- Choose **inColorModel** taking into account accuracy and execution speed (HSV is the fastest, HSI is usually the most accurate).
- Define the range of HSV/HSL/HSI values that best separates the foreground and background pixels. Analyze the results on the **outMonoImage** output.
- Use **inFuzziness** to add some smooth transitions between black and white pixels in the result.

Examples



ThresholdImage_HSx *performed on the sample image with* ***inColorModel*** *= HSV,* ***inBeginHue*** *= 0.0,* ***inEndHue*** *= 10.0,* ***inMinSaturation*** *= 120.0,* ***inMinBrightness*** *= 70.0.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not a 3-channel and 8-bit image in ThresholdImage_HSx.
DomainError	Region exceeds an input image in ThresholdImage_HSx.
DomainError	Not supported inRgbImage pixel format in ThresholdImage_HSx. Supported formats: 3xUInt8.

See Also

- ThresholdToRegion_HSx – Creates a region containing image pixels which belongs to specified region in HSV, HSL or HSI space.

ThresholdImage_Hysteresis

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Thresholds an image with a hysteresis, i.e. with a lower threshold for neighboring pixels.

Syntax

```
void avl::ThresholdImage_Hysteresis
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<float> inMinValue,
 atl::Optional<float> inMaxValue,
 float inHysteresis,
 avl::Image& outMonoImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Region of interest
→	inMinValue	Optional<float>		128.0f	Minimum value of a pixel that is considered foreground (Auto = -INF)
→	inMaxValue	Optional<float>		NIL	Maximum value of a pixel that is considered foreground (Auto = +INF)
→	inHysteresis	float	0.0 - ∞	16.0f	Defines how much the threshold criteria are lowered for pixels neighboring with other foreground pixels
←	outMonoImage	Image&			

Description

The operation transforms each pixel value to the maximum or minimum level thus creating binary image. The result of the transformation depends on the pixel intensity:

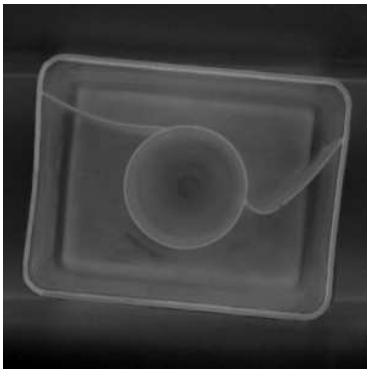
- Pixel values in range [**inMinValue**, **inMaxValue**] are transformed to the maximum level.
- Pixel values lower than **inMinValue**-**inHysteresis** or higher than **inMaxValue**+**inHysteresis** are transformed to the minimum level.
- Pixel values in range [**inMinValue**-**inHysteresis**, **inMinValue**) or in range (**inMaxValue**, **inMaxValue**+**inHysteresis**] are transformed to the maximum if (and only if) in the processed image there is a path of consecutive pixels of value in range [**inMinValue**-**inHysteresis**, **inMaxValue**+**inHysteresis**] that connects the pixel being considered and any pixel with value in range [**inMinValue**, **inMaxValue**].

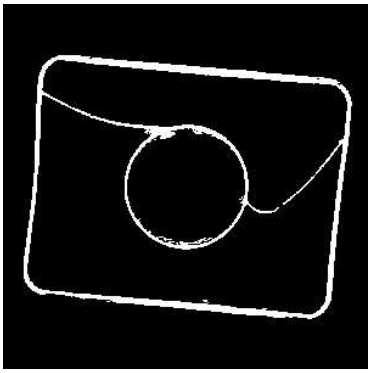
In the multichannel images the operation uses an average of channel values in each pixel, thus the resulting image is always monochromatic.

Hints

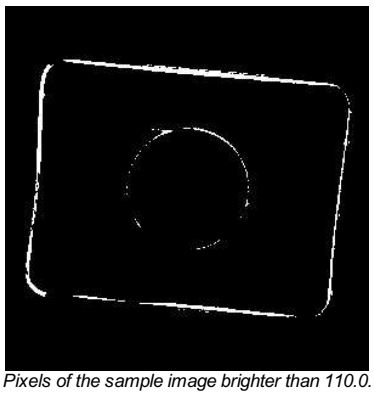
- Define **inMinValue** to extract bright objects (brighter than the specified value).
- Define **inMaxValue** to extract dark objects (darker than the specified value).
- Use **inHysteresis** to allow weaker threshold for pixels neighboring with already extracted ones.

Examples

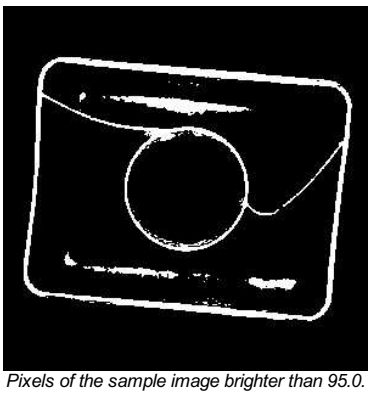




ThresholdImage_Hysteresis *performed on the sample image with* ***inMinValue*** *= 110.0,* ***inMaxValue*** *= Nil,* ***inHysteresis*** *= 15.0.*



Pixels of the sample image brighter than 110.0.



Pixels of the sample image brighter than 95.0.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in ThresholdImage_Hysteresis.

See Also

- ThresholdToRegion – Creates a region containing image pixels with values within the specified range.

ThresholdImage_Multirange

Header: AVL.h
Namespace: avl
Module: FoundationPro

Transforms each pixel value to maximum or minimum depending on whether they belong to the specified range.

Applications: Image binarization when the illumination is constant and uniform.

Syntax

```
void avl::ThresholdImage_Multirange
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<float> inMinValue1,
 atl::Optional<float> inMaxValue1,
 atl::Optional<float> inMinValue2,
 atl::Optional<float> inMaxValue2,
 atl::Optional<float> inMinValue3,
 atl::Optional<float> inMaxValue3,
 atl::Optional<float> inMinValue4,
 atl::Optional<float> inMaxValue4,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Region of interest
→	inMinValue1	Optional<float>	1.0f	Minimum value of a pixel that is considered foreground (Auto = -INF)
→	inMaxValue1	Optional<float>	NIL	Maximum value of a pixel that is considered foreground (Auto = +INF)
→	inMinValue2	Optional<float>	64.0f	Minimum value of a pixel that is considered foreground (Auto = -INF)
→	inMaxValue2	Optional<float>	NIL	Maximum value of a pixel that is considered foreground (Auto = +INF)
→	inMinValue3	Optional<float>	128.0f	Minimum value of a pixel that is considered foreground (Auto = -INF)
→	inMaxValue3	Optional<float>	NIL	Maximum value of a pixel that is considered foreground (Auto = +INF)
→	inMinValue4	Optional<float>	192.0f	Minimum value of a pixel that is considered foreground (Auto = -INF)
→	inMaxValue4	Optional<float>	NIL	Maximum value of a pixel that is considered foreground (Auto = +INF)
←	outImage	Image&		Output image

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: 1xUINT8 (for inFuzziness = 0).

This operation is optimized for AVX2 technology for pixels of types: 1xUINT8 (for inFuzziness = 0) PARALLEL.

ThresholdImage_Relative

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Thresholds an image with a different threshold value for each pixel (inBaseImage(x, y) + inValue).

Syntax

```
void avl::ThresholdImage_Relative
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 const avl::Image& inBaseImage,
 atl::Optional<float> inMinRelativeValue,
 atl::Optional<float> inMaxRelativeValue,
 float inFuzziness,
 avl::Image& outMonoImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Region of interest
→	inBaseImage	const Image&			Pixels of this image are subtracted from inImage before thresholding
→	inMinRelativeValue	Optional<float>		128.0f	Minimum relative value of a pixel that is considered foreground (Auto = -INF)
→	inMaxRelativeValue	Optional<float>		NIL	Maximum relative value of a pixel that is considered foreground (Auto = +INF)
→	inFuzziness	float	0.0 - ∞	0.0f	A tolerance for inMin/MaxRelativeValue that results in intermediate output values
←	outMonoImage	Image&			

Description

The operation transforms each pixel value of **inImage** to the maximum or minimum level thus creating binary image. The result of the transformation depends on the pixel intensity compared with the intensity of the **inBaseImage** pixels:

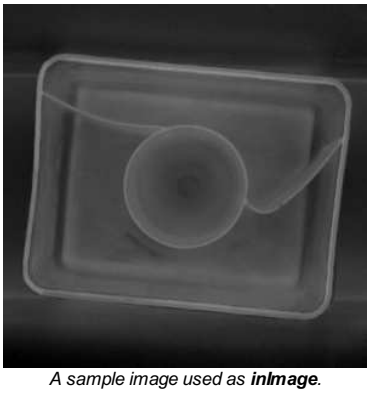
- Pixels that are brighter than corresponding pixels of **inBaseImage** by at least **inMinRelativeValue** and at most **inMaxRelativeValue** are transformed to the maximum level.
- Other pixel values are transformed to the minimum level.

If any of the parameters **inMinRelativeValue**, **inMaxRelativeValue** is not set, it is assumed to be, accordingly, *-infinity* or *infinity*.

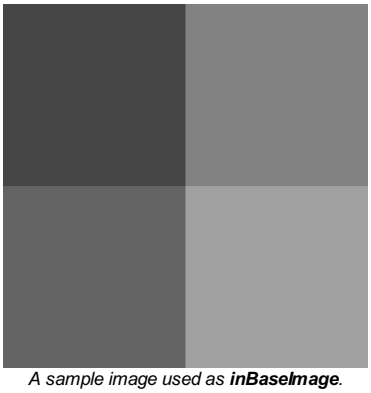
Parameter **inFuzziness** (set to 0 by default) allows to perform fuzzy thresholding which linearly interpolates those pixel values that differ by at most **inFuzziness** from the border intensities; thus creating smooth transition between minimum and maximum values in the resulting image (see **Remarks** section).

In the multichannel images the operation uses an average of channel values in each pixel, thus the resulting image is always monochromatic.

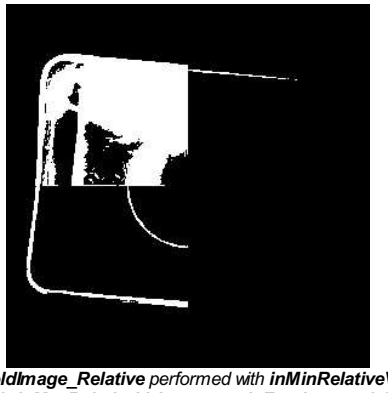
Examples



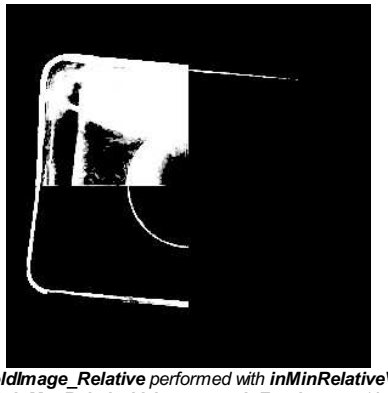
*A sample image used as **inImage**.*



*A sample image used as **inBaseImage**.*



***ThresholdImage_Relative** performed with **inMinRelativeValue** = 0.0, **inMaxRelativeValue** = auto, **inFuzziness** = 0.0.*



***ThresholdImage_Relative** performed with **inMinRelativeValue** = 0.0, **inMaxRelativeValue** = auto, **inFuzziness** = 10.0.*

Remarks

When image pixel type is Real, parameter **inFuzziness** has no influence on the output (input is ignored). The reason is strictly mathematical. Namely, there does not exist a smooth transition between *-infinity* and *infinity* values, which are, correspondingly, minimum and maximum levels of pixel brightness.

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: 1xUINT8, 3xUINT8.

This operation is optimized for AVX2 technology for pixels of types: 1xUINT8, 3xUINT8.

This operation is optimized for NEON technology for pixels of types: 1xUINT8, 3xUINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Image formats are not the same in ThresholdImage_Relative.
DomainError	Image sizes are not equal in ThresholdImage_Relative.
DomainError	Region exceeds an input image in ThresholdImage_Relative.

See Also

- ThresholdToRegion_Relative – Thresholds an image with a different threshold value for each pixel (inBaseImage(x, y) + inValue).

ThresholdImage_RGB

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Transforms each pixel value to minimum or maximum depending on whether it belongs to the specified range for each individual pixel component.

Applications: Multi-channel thresholding.

Syntax

```
void avl::ThresholdImage_RGB
(
 const avl::Image& inRgbImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<int> inMinRed,
 atl::Optional<int> inMaxRed,
 atl::Optional<int> inMinGreen,
 atl::Optional<int> inMaxGreen,
 atl::Optional<int> inMinBlue,
 atl::Optional<int> inMaxBlue,
 atl::Optional<int> inMinAlpha,
 atl::Optional<int> inMaxAlpha,
 float inFuzziness,
 avl::Image& outMonoImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inRgbImage	const Image&			Input image, usually in the RGB color space
→	inRoi	Optional<const Region&>		NIL	Region of interest
→	inMinRed	Optional<int>	0 - 255	128	Minimum for the first pixel component, usually Red (Auto = -INF)
→	inMaxRed	Optional<int>	0 - 255	NIL	Maximum for the first pixel component, usually Red (Auto = +INF)
→	inMinGreen	Optional<int>	0 - 255	128	Minimum for the second pixel component, usually Green (Auto = -INF)
→	inMaxGreen	Optional<int>	0 - 255	NIL	Maximum for the second pixel component, usually Green (Auto = +INF)
→	inMinBlue	Optional<int>	0 - 255	128	Minimum for the third pixel component, usually Blue (Auto = -INF)
→	inMaxBlue	Optional<int>	0 - 255	NIL	Maximum for the third pixel component, usually Blue (Auto = +INF)
→	inMinAlpha	Optional<int>	0 - 255	NIL	Minimum for the fourth pixel component, usually Blue (Auto = -INF)
→	inMaxAlpha	Optional<int>	0 - 255	NIL	Maximum for the fourth pixel component, usually Blue (Auto = +INF)
→	inFuzziness	float	0.0 - ∞		Tolerance for the ranges that results in intermediate output values
←	outMonoImage	Image&			

Requirements

For input **inRgbImage** only pixel formats are supported: 3□uint8, 4□uint8.

Read more about pixel formats in Image documentation.

Description

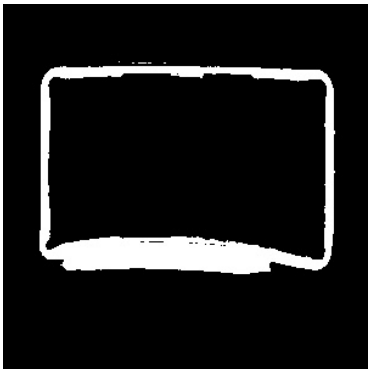
The operation transforms each pixel of three-channel **inRgbImage** to the maximum or minimum level thus creating binary image. The input image is considered to be encoded using RGB color representation.

- Pixels meeting all of the following conditions are transformed to the maximum level:
  - Intensity of the "red" channel is in range (**inMinRed**, **inMaxRed**).
  - Intensity of the "green" channel is in range (**inMinGreen**, **inMaxGreen**).
  - Intensity of the "blue" channel is in range (**inMinBlue**, **inMaxBlue**).
- Other pixels are transformed to the maximum level.

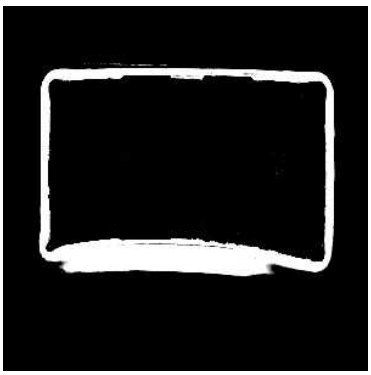
If any of the parameters **inMinRed**, **inMinGreen**, **inMinBlue** is not set, it is assumed to be *-infinity*. If any of the parameters **inMaxRed**, **inMaxGreen**, **inMaxBlue** is not set, it is assumed to be *infinity*.

Parameter **inFuzziness** (set to 0 by default) allows to perform fuzzy thresholding which linearly interpolates those pixel values that differ by at most **inFuzziness** from the border channel intensities; thus creating smooth transition between minimum and maximum values in the resulting image.

Examples

ThresholdImage_RGB *performed on the sample image with* ***inMinRed*** *= 120.0,* ***inMaxGreen*** *= 100.0,* ***inMaxBlue*** *= 100.0,* ***inFuzziness*** *= 0.0.*

ThresholdImage_RGB *performed on the sample image with* ***inMinRed*** *= 120.0,* ***inMaxGreen*** *= 100.0,* ***inMaxBlue*** *= 100.0,* ***inFuzziness*** *= 10.0.*

Hardware Acceleration

This operation is optimized for SSSE4 (for inFuzziness = 0) technology.

This operation is optimized for AVX2 (for inFuzziness = 0) technology.

This operation is optimized for NEON technology for pixels of types: 1xUINT8 (for inFuzziness = 0), 3xUINT8 (for inFuzziness = 0).

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not a 3-channel and 8-bit image in ThresholdImage_RGB.
DomainError	Region exceeds an input image in ThresholdImage_RGB.
DomainError	Not supported inRgbImage pixel format in ThresholdImage_RGB. Supported formats: 3xUInt8, 4xUInt8.

See Also

- ThresholdToRegion_RGB – Creates a region containing image pixels which belongs to the specified range for each individual pixel component.

ThresholdToRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Creates a region containing image pixels with values within the specified range.

Applications: Extraction of a region of objects that can be defined by a salient brightness.

Syntax

```
void avl::ThresholdToRegion
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<float> inMinValue,
 atl::Optional<float> inMaxValue,
 float inHysteresis,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Region of interest
→	inMinValue	Optional<float>		128.0f	Minimum value of a pixel that is considered foreground (Auto = -INF)
→	inMaxValue	Optional<float>		NIL	Maximum value of a pixel that is considered foreground (Auto = +INF)
→	inHysteresis	float	0.0 - ∞	0.0f	Defines how much the threshold criteria are lowered for pixels neighboring with other foreground pixels
←	outRegion	Region&			Output region

Description

The operation is a cousin of ThresholdImage yet computes a region instead of an image. The resulting region contains those pixels of the input image that meet one of the following conditions:

- Pixel value is in range [**inMinValue**, **inMaxValue**].
- Pixel value is in range [**inMinValue**-**inHysteresis**, **inMinValue**) or in range (**inMaxValue**, **inMaxValue**+**inHysteresis**] and in the processed image there is a path of consecutive pixels of value in range [**inMinValue**-**inHysteresis**, **inMaxValue**+**inHysteresis**] that connects the pixel being considered and any pixel with value in range [**inMinValue**, **inMaxValue**].

If any of the parameters **inMinValue**, **inMaxValue** is not set, it is assumed to be, accordingly, *-infinity* or *infinity*.

In the multichannel images the operation uses an average of channel values in each pixel.

Hints

- Define **inMinValue** to extract bright objects (brighter than the specified value).
- Define **inMaxValue** to extract dark objects (darker than the specified value).
- Use **inHysteresis** to allow weaker threshold for pixels neighboring with already extracted ones.

Examples

ThresholdToRegion *performed on the sample image with* ***inMinValue*** *= 80.0,* ***inMaxValue*** *= auto.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: 1xUINT8, 3xUINT8, 1xINT16, 1xUINT16.

This operation is optimized for NEON technology for pixels of types: 1xUINT8, 3xUINT8, 1xINT16.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in ThresholdToRegion.

See Also

- ThresholdImage – Transforms each pixel value to maximum or minimum depending on whether they belong to the specified range.

ThresholdToRegion_Color

Header: AVL.h
Namespace: avl
Module: FoundationPro

Creates a region containing image pixels with values close to the given color.

Applications: Color analysis with a given reference color.

Syntax

```
void avl::ThresholdToRegion_Color
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 const avl::Pixel& inRgbColor,
 float inChromaAmount,
 float inMaxDifference,
 float inHysteresis,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Region of interest
→	inRgbColor	const Pixel&			Color to compare the image to
→	inChromaAmount	float	0.0 - 1.0	0.7f	Proportion of chromatic information in distance computation
→	inMaxDifference	float	0.0 - ∞	5.0f	Maximum difference between image pixel and model color
→	inHysteresis	float	0.0 - ∞	0.0f	Defines how much the difference criterium is lowered for pixels neighboring with other foreground pixels
←	outRegion	Region&			Output region

Requirements

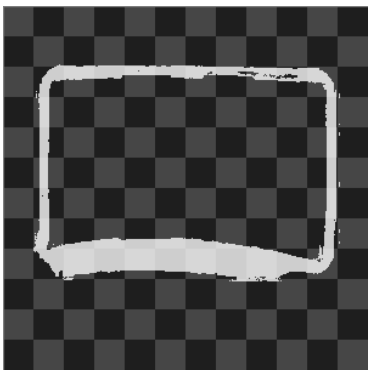
For input **inImage** only pixel formats are supported: 1□uint8, 2□uint8, 3□uint8, 4□uint8.

Read more about pixel formats in Image documentation.

Hints

- Define the reference color by setting the **inRgbColor** input.
- Increase **inChromaAmount** to make the filter less sensitive to changes in brightness. Decrease it to make brightness more important.
- Set **inMaxDifference** experimentally to a value that best separates the foreground and background pixels.

Examples



ThresholdToRegion_Color *performed on a sample image with* ***inRgbColor*** *= (192, 34, 22),* ***inChromaAmount*** *= 1.0,* ***inMaxDifference*** *= 48.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: 1xUINT8, 3xUINT8.

This operation is optimized for NEON technology for pixels of types: 1xUINT8, 3xUINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in ThresholdToRegion_Color.
DomainError	Not supported inImage pixel format in ThresholdToRegion_Color. Supported formats: 1xUInt8, 2xUInt8, 3xUInt8, 4xUInt8.

See Also

- ColorDistanceImage – Compares each pixel with the specified color using chromatic and non-chromatic information.

ThresholdToRegion_Dynamic

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Thresholds an image relatively to the average pixel value in a local rectangular neighborhood.

Applications: Useful in case of uneven illumination.

Syntax

```
void avl::ThresholdToRegion_Dynamic
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<const avl::Region&> inSourceRoi,
 int inRadiusX,
 atl::Optional<int> inRadiusY,
 atl::Optional<float> inMinRelativeValue,
 atl::Optional<float> inMaxRelativeValue,
 float inHysteresis,
 avl::Region& outRegion,
 avl::Image& diagBaseImage
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inImage	const Image&			Input image
➡	inRoi	Optional<const Region&>		NIL	Region in which pixels are written
➡	inSourceRoi	Optional<const Region&>		NIL	Region from which pixels are read
➡	inRadiusX	int	0 - 65535	5	Horizontal radius of internal mean blur
➡	inRadiusY	Optional<int>	0 - 65535	NIL	Vertical radius of internal mean blur (Auto = inRadiusX)
➡	inMinRelativeValue	Optional<float>		5.0f	Minimum relative value of a pixel that is considered foreground (Auto = -INF)
➡	inMaxRelativeValue	Optional<float>		NIL	Maximum relative value of a pixel that is considered foreground (Auto = +INF)
➡	inHysteresis	float	0.0 - ∞	0.0f	Defines how much the threshold criteria are lowered for pixels neighboring with other foreground pixels
⬅	outRegion	Region&			Output region
🔍	diagBaseImage	Image&			Diagnostic blurred image.

Description

The operation is a cousin of ThresholdImage_Dynamic yet computes a region instead of an image. The resulting region contains only those pixels of the input image, which are brighter at least by **inMinRelativeValue** and at most by **inMaxRelativeValue** than the local average. If any of the parameters **inMinRelativeValue**, **inMaxRelativeValue** is not set, it is assumed to be, accordingly, -infinity or infinity.

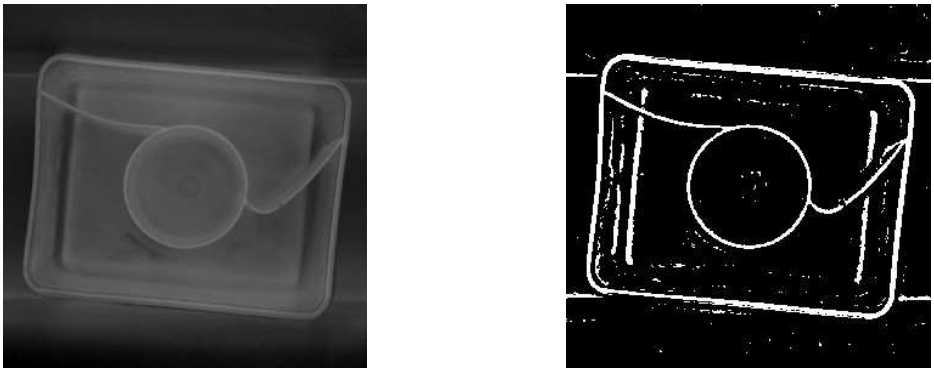
Pixel neighbourhood used to compute the local average is a rectangle of dimensions $(2 \cdot \mathbf{inRadiusX} + 1) \times (2 \cdot \mathbf{inRadiusY} + 1)$ centered at the pixel being processed.

In the multichannel images the operation uses an average of channel values in each pixel.

Hints

- Define **inMinRelativeValue** to extract objects which are brighter than the neighborhood.
- Define **inMaxRelativeValue** to extract objects which are darker than the neighborhood.
- Increase **inRadiusX** (and optionally **inRadiusY**) to define a bigger neighborhood.

Examples



ThresholdToRegion_Dynamic *performed on the sample image with* ***inMinRelativeValue*** *= 5.0,* ***inMaxRelativeValue*** *= auto.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in ThresholdToRegion_Dynamic.
DomainError	Roi exceeds image dimensions in ThresholdToRegion_Dynamic.
DomainError	Source roi exceeds image dimensions in ThresholdToRegion_Dynamic.

See Also

- ThresholdImage_Dynamic – Thresholds an image relatively to some value calculated in a local rectangular neighbourhood.

ThresholdToRegion_HSx

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Creates a region containing image pixels which belongs to specified region in HSV, HSL or HSI space.

Applications: Extraction of a region characterized with a specific color.

Syntax

```
void avl::ThresholdToRegion_HSx
(
 const avl::Image& inRgbImage,
 atl::Optional<const avl::Region&> inRoi,
 const avl::HSxColorModel::Type inColorModel,
 int inBeginHue,
 int inEndHue,
 atl::Optional<int> inMinSaturation,
 atl::Optional<int> inMaxSaturation,
 atl::Optional<int> inMinBrightness,
 atl::Optional<int> inMaxBrightness,
 avl::Region& outRegion,
 avl::Image& diagHSxImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inRgbImage	const Image&			
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inColorModel	const HSxColorModel::Type			Selected color model
→	inBeginHue	int	0 - 255	0	Lowest acceptable Hue; if higher than inEndHue, then range wrapping is used
→	inEndHue	int	0 - 255	255	Highest acceptable Hue, if lower than inBeginHue, then range wrapping is used
→	inMinSaturation	Optional<int>	0 - 255	128	
→	inMaxSaturation	Optional<int>	0 - 255	NIL	
→	inMinBrightness	Optional<int>	0 - 255	128	Minimum brightness; denotes V, L or I, depending on inColorModel
→	inMaxBrightness	Optional<int>	0 - 255	NIL	Maximum brightness; denotes V, L or I, depending on inColorModel
←	outRegion	Region&			Output region
	diagHSxImage	Image&			Diagnostic image in HSx color space

Requirements

For input **inRgbImage** only pixel formats are supported: 3□uint8.

Read more about pixel formats in Image documentation.

Description

The operation is a cousin of ThresholdImage_HSx yet it computes a region instead of an image. The three-channel **inRgbImage** is considered to be encoded using RGB color representation. Each of the image pixel is internally converted to HSx (HSV, HSL or HSI) color representation and then examined. The resulting region contains only those pixels of the input image, which meets all of the following conditions:

- Value of the Hue parameter is in cyclic range (**inBeginHue**, **inEndHue**).
- Value of the Saturation parameter is in range (**inMinSaturation**, **inMaxSaturation**).
- Value of the Value parameter is in range (**inMinBrightness**, **inMaxBrightness**).

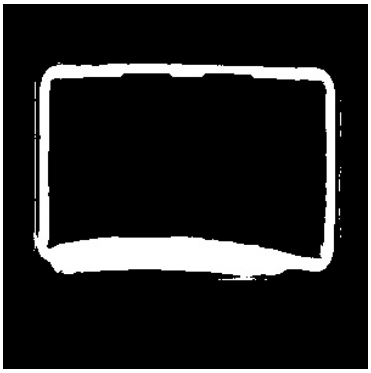
If any of the parameters **inMinSaturation**, **inMinBrightness** is not set, it is assumed to be -infinity.

If any of the parameters **inMaxSaturation**, **inMaxBrightness** is not set, it is assumed to be infinity.

Hints

- Choose **inColorModel** taking into account accuracy and execution speed (HSV is the fastest, HSI is usually the most accurate).
- Define the range of HSV/HSL/HSI values that best separates the foreground and background pixels. Analyze the results on the **outRegion** output.

Examples



ThresholdToRegion_HSx *performed on the sample image with* ***inColorModel*** *= HSV,* ***inBeginHue*** *= 0.0,* ***inEndHue*** *= 10.0,* ***inMinSaturation*** *= 120.0,* ***inMinBrightness*** *= 70.0.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not a 3-channel and 8-bit image in ThresholdToRegion_HSx.
DomainError	Region exceeds an input image in ThresholdToRegion_HSx.
DomainError	Not supported inRgbImage pixel format in ThresholdToRegion_HSx. Supported formats: 3xUInt8.

See Also

- ThresholdImage_HSx – Transforms each pixel value to minimum or maximum depending on whether it belongs to specified region in the HSV, HSL or HSI color space.

ThresholdToRegion_Relative

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Thresholds an image with a different threshold value for each pixel (inBaseImage(x, y) + inValue).

Syntax

```
void avl::ThresholdToRegion_Relative
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 const avl::Image& inBaseImage,
 atl::Optional<float> inMinRelativeValue,
 atl::Optional<float> inMaxRelativeValue,
 float inHysteresis,
 avl::Region& outRegion
)
```

Parameters

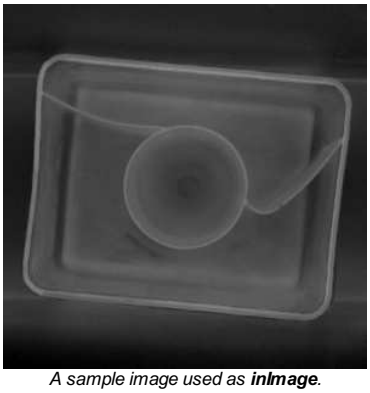
	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Region of interest
→	inBaseImage	const Image&			Pixels of this image are subtracted from inImage before thresholding
→	inMinRelativeValue	Optional<float>		128.0f	Minimum relative value of a pixel that is considered foreground (Auto = -INF)
→	inMaxRelativeValue	Optional<float>		NIL	Maximum relative value of a pixel that is considered foreground (Auto = +INF)
→	inHysteresis	float	0.0 - ∞	0.0f	Defines how much the threshold criteria are lowered for pixels neighboring with other foreground pixels
←	outRegion	Region&			Output region

Description

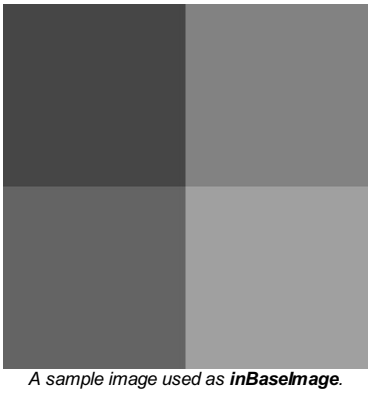
The operation is a cousin of ThresholdImage_Relative yet computes a region instead of an image. The resulting region contains only those pixels of the input image, which are brighter at least by **inMinRelativeValue** and at most by **inMaxRelativeValue** than the corresponding pixel of **inBaseImage**. If any of the parameters **inMinRelativeValue**, **inMaxRelativeValue** is not set, it is assumed to be, accordingly, -infinity or infinity.

In the multichannel images the operation uses an average of channel values in each pixel.

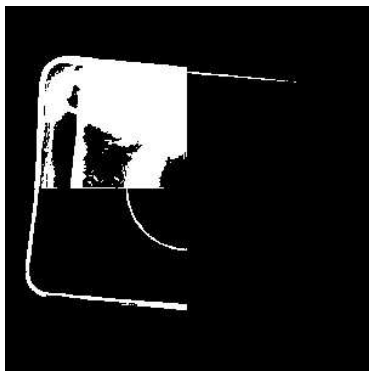
Examples



*A sample image used as **inImage**.*



*A sample image used as **inBaseImage**.*



ThresholdToRegion_Relative performed with **inMinRelativeValue** = 0.0, **inMaxRelativeValue** = auto.

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: 1xUINT8, 3xUINT8.

This operation is optimized for NEON technology for pixels of types: 1xUINT8, 3xUINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Image formats are not the same in ThresholdToRegion_Relative.
DomainError	Image sizes are not equal in ThresholdToRegion_Relative.
DomainError	Region exceeds an input image in ThresholdToRegion_Relative.

See Also

- ThresholdImage_Relative – Thresholds an image with a different threshold value for each pixel (inBaseImage(x, y) + inValue).

ThresholdToRegion_Relative_DarkBright

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Thresholds an image using two relative thresholds and produces two regions (Dark and Bright) in a single image pass.

Applications: This filter can replace two instances of ThresholdToRegion_Relative which may slightly improve performance.

Syntax

```
void avl::ThresholdToRegion_Relative_DarkBright
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 const avl::Image& inBaseImage,
 float inDarkThreshold,
 float inBrightThreshold,
 float inHysteresis,
 avl::Region& outDarkRegion,
 avl::Region& outBrightRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Region of interest
→	inBaseImage	const Image&			Pixels of this image are subtracted from inImage before thresholding
→	inDarkThreshold	float		64.0f	Maximum relative value of a pixel that is considered DarkRegion
→	inBrightThreshold	float		192.0f	Minimum relative value of a pixel that is considered BrightRegion
→	inHysteresis	float	0.0 - ∞	0.0f	Defines how much the threshold criteria are lowered for pixels neighboring with other foreground pixels
←	outDarkRegion	Region&			Region of pixels equal or lower than inDarkThreshold
←	outBrightRegion	Region&			Region of pixels equal or greater than inBrightThreshold

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: 1xUINT8, 3xUINT8.

This operation is optimized for NEON technology for pixels of types: 1xUINT8, 3xUINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Image formats are not the same in ThresholdToRegion_Relative_DarkBright.
DomainError	Image sizes are not equal in ThresholdToRegion_Relative_DarkBright.
DomainError	Region exceeds an input image in ThresholdToRegion_Relative_DarkBright.

ThresholdToRegion_RGB

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Creates a region containing image pixels which belongs to the specified range for each individual pixel component.

Applications: Multi-channel thresholding.

Syntax

```
void avl::ThresholdToRegion_RGB
(
 const avl::Image& inRgbImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<int> inMinRed,
 atl::Optional<int> inMaxRed,
 atl::Optional<int> inMinGreen,
 atl::Optional<int> inMaxGreen,
 atl::Optional<int> inMinBlue,
 atl::Optional<int> inMaxBlue,
 atl::Optional<int> inMinAlpha,
 atl::Optional<int> inMaxAlpha,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inRgbImage	const Image&			Input image, usually in the RGB color space
→	inRoi	Optional<const Region&>		NIL	Region of interest
→	inMinRed	Optional<int>	0 - 255	128	Minimum for the first pixel component, usually Red (Auto = -INF)
→	inMaxRed	Optional<int>	0 - 255	NIL	Maximum for the first pixel component, usually Red (Auto = +INF)
→	inMinGreen	Optional<int>	0 - 255	128	Minimum for the second pixel component, usually Green (Auto = -INF)
→	inMaxGreen	Optional<int>	0 - 255	NIL	Maximum for the second pixel component, usually Green (Auto = +INF)
→	inMinBlue	Optional<int>	0 - 255	128	Minimum for the third pixel component, usually Blue (Auto = -INF)
→	inMaxBlue	Optional<int>	0 - 255	NIL	Maximum for the third pixel component, usually Blue (Auto = +INF)
→	inMinAlpha	Optional<int>	0 - 255	NIL	Minimum for the fourth pixel component, usually Blue (Auto = -INF)
→	inMaxAlpha	Optional<int>	0 - 255	NIL	Maximum for the fourth pixel component, usually Blue (Auto = +INF)
←	outRegion	Region&			Output region

Requirements

For input **inRgbImage** only pixel formats are supported: 3□uint8, 4□uint8.

Read more about pixel formats in Image documentation.

Description

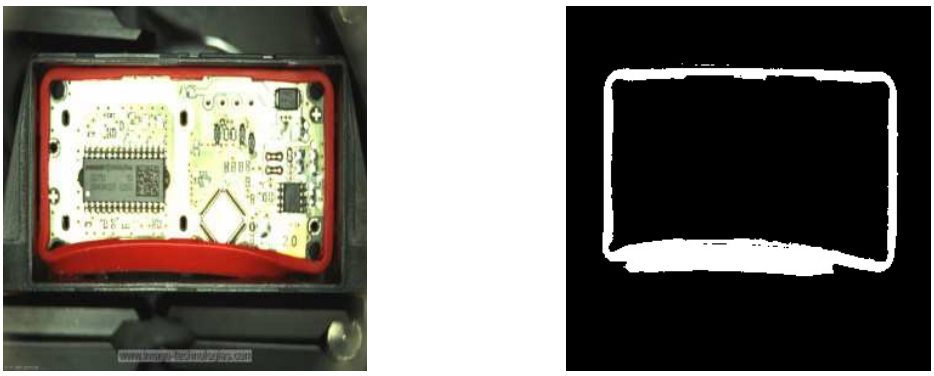
The operation is a cousin of ThresholdImage_RGB yet it computes a region instead of an image. The three-channel **inRgbImage** is considered to be encoded using RGB color representation. The resulting region contains only those pixels of the input image, which meets all of the following conditions:

- Intensity of the "red" channel is in range (**inMinRed**, **inMaxRed**).
- Intensity of the "green" channel is in range (**inMinGreen**, **inMaxGreen**).
- Intensity of the "blue" channel is in range (**inMinBlue**, **inMaxBlue**).

If any of the parameters **inMinRed**, **inMinGreen**, **inMinBlue** is not set, it is assumed to be -infinity.

If any of the parameters **inMaxRed**, **inMaxGreen**, **inMaxBlue** is not set, it is assumed to be infinity.

Examples



***ThresholdToRegion_RGB** performed on the sample image with **inMinRed** = 120.0, **inMaxGreen** = 100.0, **inMaxBlue** = 100.0.*

Hardware Acceleration

This operation is optimized for SSE41 technology for pixels of type: UINT8.

This operation is optimized for NEON technology for pixels of type: UINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not an 8-bit image in ThresholdToRegion_RGB.
DomainError	Region exceeds an input image in ThresholdToRegion_RGB.
DomainError	Unsupported image depth in ThresholdToRegion_RGB.
DomainError	Not supported inRgbImage pixel format in ThresholdToRegion_RGB. Supported formats: 3xUInt8, 4xUInt8.

See Also

- ThresholdImage_RGB – Transforms each pixel value to minimum or maximum depending on whether it belongs to the specified range for each individual pixel component.

57. HandEyeCalibration Calibration

Table of content:

CalibrateEyeInHand

Header: AVL.h
Namespace: avl
Module: Calibration

Computes Hand-Eye calibration matrices.

Syntax

```
void avl::CalibrateEyeInHand
(
 const atl::Array<avl::Matrix>& inRotationGripperToBase,
 const atl::Array<avl::Matrix>& inTranslationGripperToBase,
 const atl::Array<avl::Matrix>& inRotationTargetToCam,
 const atl::Array<avl::Matrix>& inTranslationTargetToCam,
 const avl::HandEyeCalibrationMethod::Type inCalibrationMethod,
 avl::Matrix& outRotationCamToGripper,
 avl::Matrix& outTranslationCamToGripper
)
```

Parameters

	Name	Type	Default	Description
→	inRotationGripperToBase	const Array<Matrix>&		Rotation part extracted from the homogeneous matrix that transforms a point expressed in the gripper frame to the robot base frame.
→	inTranslationGripperToBase	const Array<Matrix>&		Translation part extracted from the homogeneous matrix that transforms a point expressed in the gripper frame to the robot base frame.
→	inRotationTargetToCam	const Array<Matrix>&		Rotation part extracted from the homogeneous matrix that transforms a point expressed in the target frame to the camera frame.
→	inTranslationTargetToCam	const Array<Matrix>&		Translation part extracted from the homogeneous matrix that transforms a point expressed in the target frame to the camera frame
→	inCalibrationMethod	const HandEyeCalibrationMethod::Type		Calibration method.
←	outRotationCamToGripper	Matrix&		Estimated rotation part extracted from the homogeneous matrix that transforms a point expressed in the camera frame to the gripper frame.
←	outTranslationCamToGripper	Matrix&		Estimated translation part extracted from the homogeneous matrix that transforms a point expressed in the camera frame to the gripper frame.

Errors

List of possible exceptions:

Error type	Description
DomainError	All of the arrays should have the same number of measurements.
DomainError	At least 3 measurements are needed.

EstimateAffine3DTransform

Header: AVL.h
Namespace: avl
Module: Calibration

Computes optimal affine transformation between two 3D point sets.

Syntax

```
void avl::EstimateAffine3DTransform
(
 const atl::Array<avl::Point3D>& inSrcpt,
 const atl::Array<avl::Point3D>& inDstpt,
 float inRansacThreshold,
 float inConfidence,
 avl::Matrix& outTransform,
 atl::Array<int>& outOutliers,
 int& outValue
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSrcpt	const Array<Point3D>&			List of 3D points in the source coordinate system.
→	inDstpt	const Array<Point3D>&			List of 3D points in the destination coordinate system.
→	inRansacThreshold	float		3.0f	Maximum reprojection error in the RANSAC algorithm to consider a point as an inlier.
→	inConfidence	float	0.0 - 1.0	0.99f	Confidence level, between 0 and 1, for the estimated transformation. Anything between 0.95 and 0.99 is usually good enough. Values too close to 1 can slow down the estimation significantly. Values lower than 0.8-0.9 can result in an incorrectly estimated transformation.
←	outTransform	Matrix&			Estimated affine transformation matrix.
←	outOutliers	Array<int>&			Output vector indicating which points are outliers (1-inlier, 0-outlier).
←	outValue	int&			

Errors

List of possible exceptions:

Error type	Description
DomainError	EstimateAffine3DTransform failed to find a solution.
DomainError	Input arrays must have at least 4 points in EstimateAffine3DTransform.
DomainError	Input arrays must have the same number of points in EstimateAffine3DTransform.

58. Geometry 2D Intersections

Table of content:

CircleCircleIntersection

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes common points of two circles.

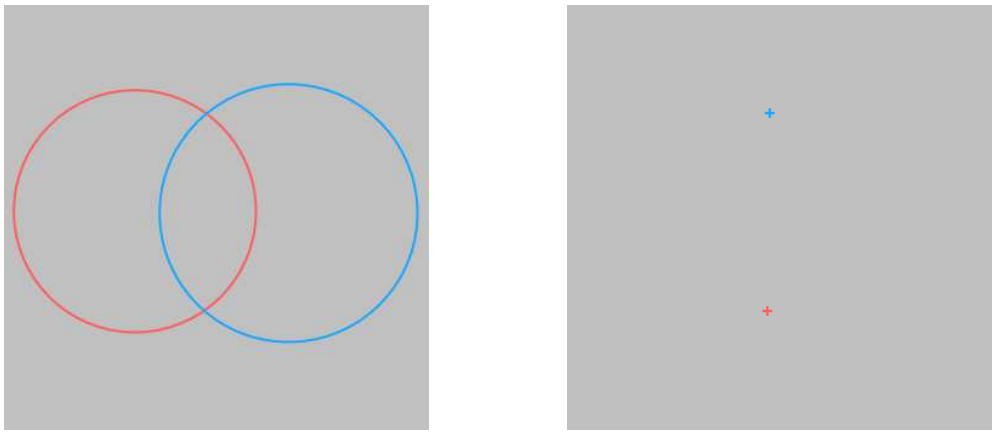
Syntax

```
void avl::CircleCircleIntersection
(
 const avl::Circle2D& inCircle1,
 const avl::Circle2D& inCircle2,
 atl::Conditional<avl::Point2D>& outIntersectionPoint1,
 atl::Conditional<avl::Point2D>& outIntersectionPoint2
)
```

Parameters

	Name	Type	Default	Description
→	inCircle1	const Circle2D&		
→	inCircle2	const Circle2D&		
←	outIntersectionPoint1	Conditional<Point2D>&		
←	outIntersectionPoint2	Conditional<Point2D>&		

Examples



CircleCircleIntersection *performed on two circles.*

LineArcIntersection

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the common points of an arc and a line.

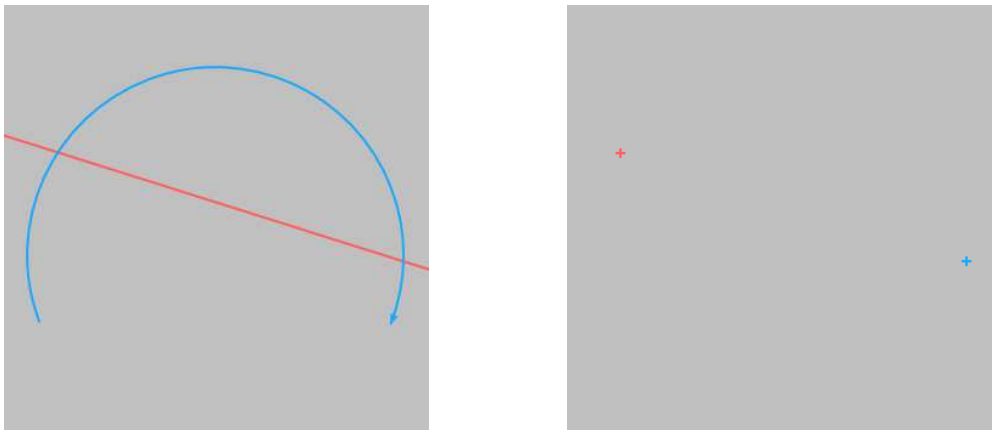
Syntax

```
void avl::LineArcIntersection
(
 const avl::Line2D& inLine,
 const avl::Arc2D& inArc,
 atl::Conditional<avl::Point2D>& outIntersectionPoint1,
 atl::Conditional<avl::Point2D>& outIntersectionPoint2
)
```

Parameters

	Name	Type	Default	Description
→	inLine	const Line2D&		
→	inArc	const Arc2D&		
←	outIntersectionPoint1	Conditional<Point2D>&		
←	outIntersectionPoint2	Conditional<Point2D>&		

Examples



LineArcIntersection *performed on line and arc.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite line on input in LineArcIntersection.

LineCircleIntersection

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes common points of a circle and a line.

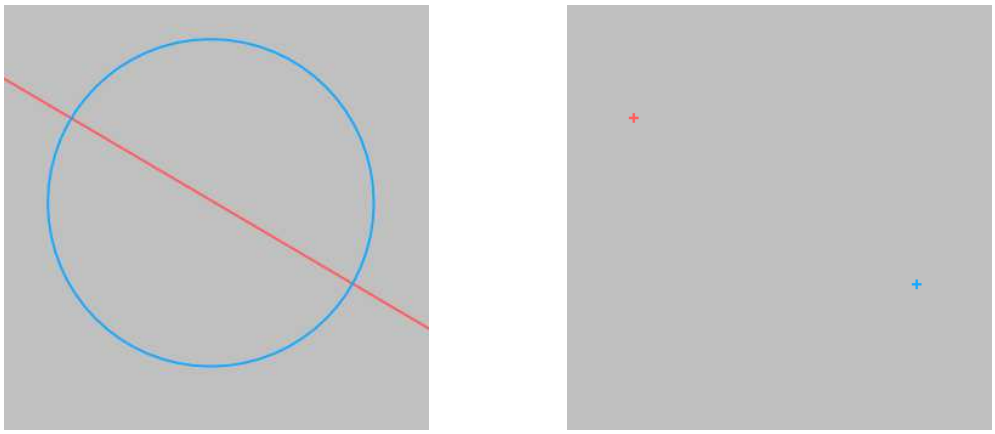
Syntax

```
void avl::LineCircleIntersection
(
 const avl::Line2D& inLine,
 const avl::Circle2D& inCircle,
 atl::Conditional<avl::Point2D>& outIntersectionPoint1,
 atl::Conditional<avl::Point2D>& outIntersectionPoint2
)
```

Parameters

	Name	Type	Default	Description
→	inLine	const Line2D&		
→	inCircle	const Circle2D&		
←	outIntersectionPoint1	Conditional<Point2D>&		
←	outIntersectionPoint2	Conditional<Point2D>&		

Examples



LineCircleIntersection *performed on line and circle.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite line on input in LineCircleIntersection.

LineLineIntersection

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes a common point of two lines.

Syntax

```
void avl::LineLineIntersection
(
 const avl::Line2D& inLine1,
 const avl::Line2D& inLine2,
 atl::Conditional<avl::Point2D>& outIntersectionPoint
)
```

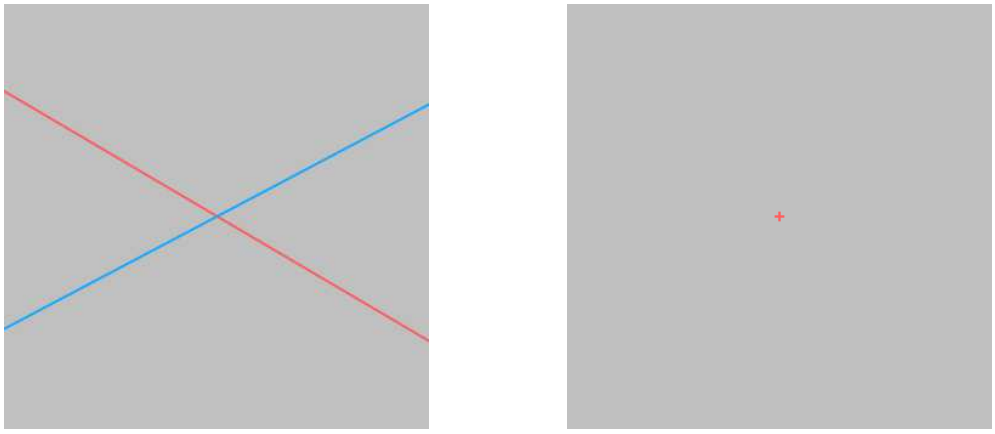
Parameters

	Name	Type	Default	Description
→	inLine1	const Line2D&		
→	inLine2	const Line2D&		
←	outIntersectionPoint	Conditional<Point2D>&		

Hints

- Whenever possible, prefer LineSegmentIntersection instead as it is more numerically stable for near-parallel lines.

Examples



***LineLineIntersection** performed on two lines.*

See Also

- LineSegmentIntersection – Computes a common point of a line and a segment.
- SegmentSegmentIntersection – Computes a common point of two segments.

LineSegmentIntersection

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes a common point of a line and a segment.

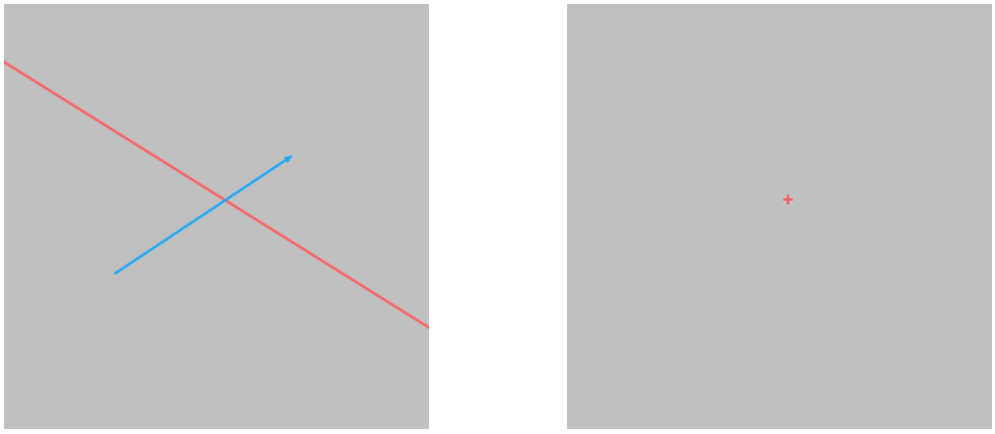
Syntax

```
void avl::LineSegmentIntersection
(
 const avl::Line2D& inLine,
 const avl::Segment2D& inSegment,
 atl::Conditional<avl::Point2D>& outIntersectionPoint
)
```

Parameters

	Name	Type	Default	Description
→	inLine	const Line2D&		
→	inSegment	const Segment2D&		
←	outIntersectionPoint	Conditional<Point2D>&		

Examples



LineSegmentIntersection *performed on line and segment.*

See Also

- SegmentSegmentIntersection – Computes a common point of two segments.

SegmentArcIntersection

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the common points of an arc and a segment.

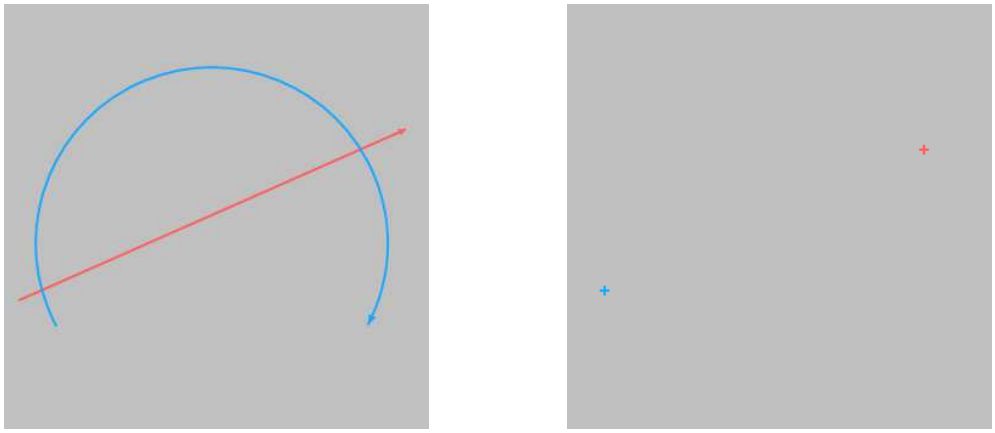
Syntax

```
void avl::SegmentArcIntersection
(
 const avl::Segment2D& inSegment,
 const avl::Arc2D& inArc,
 atl::Conditional<avl::Point2D>& outIntersectionPoint1,
 atl::Conditional<avl::Point2D>& outIntersectionPoint2
)
```

Parameters

	Name	Type	Default	Description
→	inSegment	const Segment2D&		
→	inArc	const Arc2D&		
←	outIntersectionPoint1	Conditional<Point2D>&		
←	outIntersectionPoint2	Conditional<Point2D>&		

Examples



SegmentArcIntersection *performed on segment and arc.*

SegmentArrayIntersections

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes a common points or segment of any segments from the set.

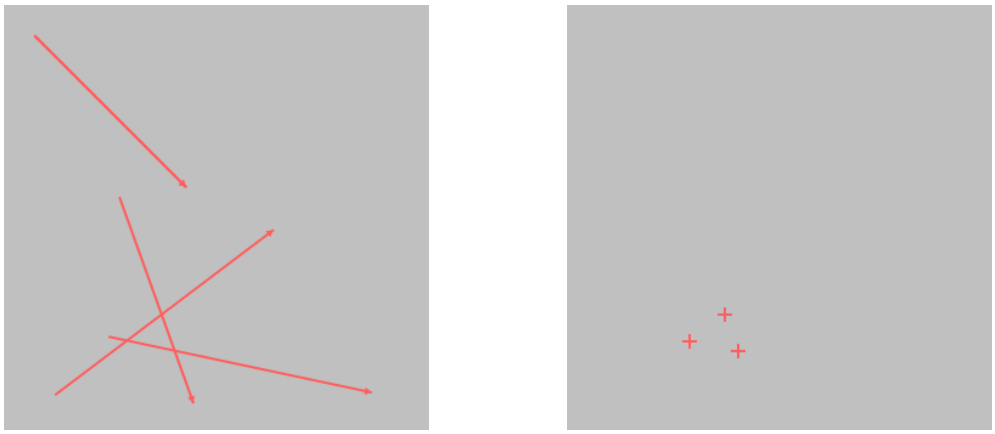
Syntax

```
void avl::SegmentArrayIntersections
(
 const atl::Array<avl::Segment2D>& inSegments,
 atl::Array<avl::Point2D>& outIntersectionPoints,
 atl::Array<int>& outFirstIntersectionIndices,
 atl::Array<int>& outSecondIntersectionIndices,
 atl::Array<avl::Segment2D>& outOverlapSegments,
 atl::Array<int>& outFirstOverlapIndices,
 atl::Array<int>& outSecondOverlapIndices
)
```

Parameters

	Name	Type	Default	Description
→	inSegments	const Array<Segment2D>&		Input segments
←	outIntersectionPoints	Array<Point2D>&		Intersection points
←	outFirstIntersectionIndices	Array<int>&		First indices of the input segments which corresponds to the intersection points
←	outSecondIntersectionIndices	Array<int>&		Second indices of the input segments which corresponds to the intersection points
←	outOverlapSegments	Array<Segment2D>&		Overlap segments
←	outFirstOverlapIndices	Array<int>&		First indices of the input segments which corresponds to the overlap segments
←	outSecondOverlapIndices	Array<int>&		Second indices of the input segments which corresponds to the overlap segments

Examples



SegmentArrayIntersections *performed on segments.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

SegmentCircleIntersection

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the common points of a circle and a segment.

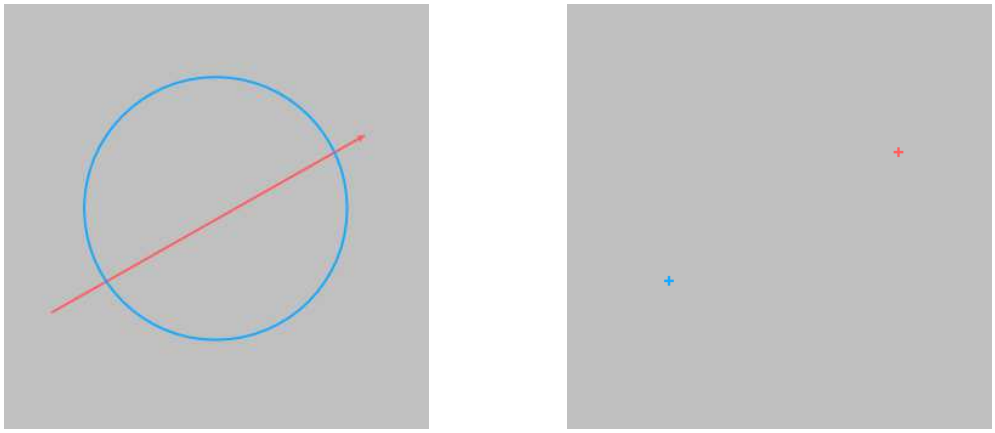
Syntax

```
void avl::SegmentCircleIntersection
(
 const avl::Segment2D& inSegment,
 const avl::Circle2D& inCircle,
 atl::Conditional<avl::Point2D>& outIntersectionPoint1,
 atl::Conditional<avl::Point2D>& outIntersectionPoint2
)
```

Parameters

	Name	Type	Default	Description
→	inSegment	const Segment2D&		
→	inCircle	const Circle2D&		
←	outIntersectionPoint1	Conditional<Point2D>&		
←	outIntersectionPoint2	Conditional<Point2D>&		

Examples



SegmentCircleIntersection *performed on segment and circle.*

SegmentSegmentIntersection

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes a common point of two segments.

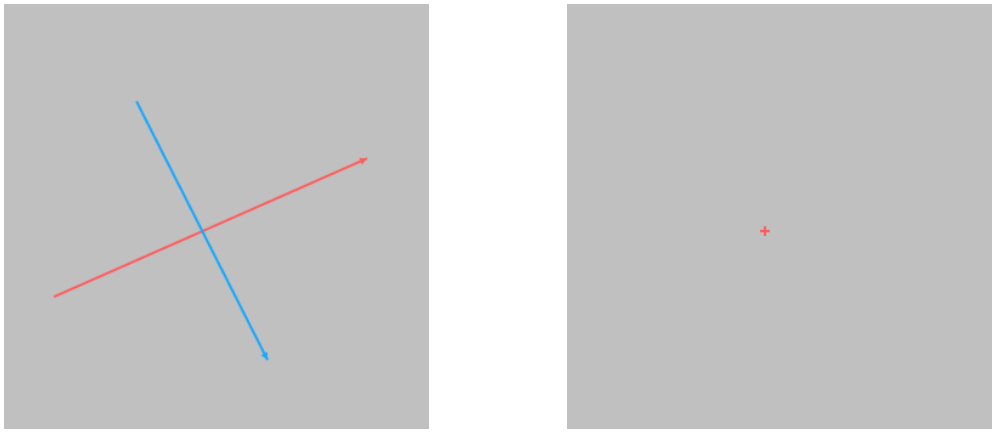
Syntax

```
void avl::SegmentSegmentIntersection
(
 const avl::Segment2D& inSegment1,
 const avl::Segment2D& inSegment2,
 atl::Conditional<avl::Point2D>& outIntersectionPoint
)
```

Parameters

	Name	Type	Default	Description
→	inSegment1	const Segment2D&		
→	inSegment2	const Segment2D&		
←	outIntersectionPoint	Conditional<Point2D>&		

Examples



SegmentSegmentIntersection *performed on two segments.*

See Also

- LineSegmentIntersection – Computes a common point of a line and a segment.

TestRectangleIntersectsWith

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Tests if two rectangles intersect.

Syntax

```
void avl::TestRectangleIntersectsWith
(
 const avl::Rectangle2D& inRectangle,
 const avl::Rectangle2D& inReferenceRectangle,
 bool& outRectanglesIntersect
)
```

Parameters

	Name	Type	Default	Description
→	inRectangle	const Rectangle2D&		
→	inReferenceRectangle	const Rectangle2D&		
←	outRectanglesIntersect	bool&		

59. Geometry 2D Constructions

Table of content:

CircleTangents

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes circle tangent lines passing through a point.

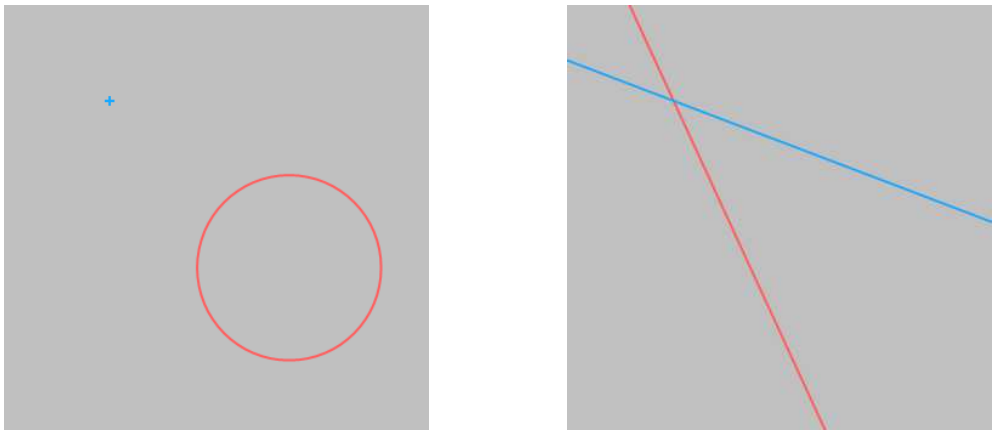
Syntax

```
void avl::CircleTangents
(
 const avl::Circle2D& inCircle,
 const avl::Point2D& inPoint,
 atl::Conditional<avl::Line2D>& outTangentLine1,
 atl::Conditional<avl::Line2D>& outTangentLine2
)
```

Parameters

	Name	Type	Default	Description
→	inCircle	const Circle2D&		
→	inPoint	const Point2D&		
←	outTangentLine1	Conditional<Line2D>&		
←	outTangentLine2	Conditional<Line2D>&		

Examples



CircleTangents *performed on circle and point.*

CircleThroughPoints

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes a circle passing through three noncollinear points.

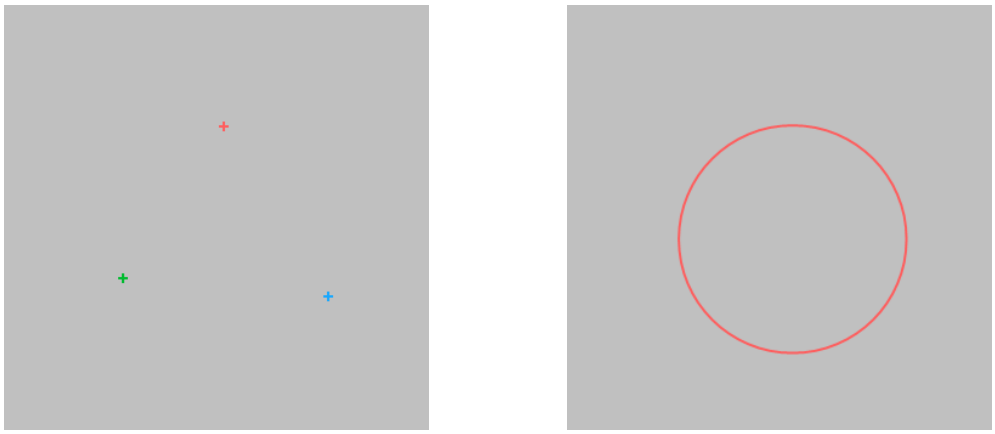
Syntax

```
void avl::CircleThroughPoints
(
 const avl::Point2D& inPoint1,
 const avl::Point2D& inPoint2,
 const avl::Point2D& inPoint3,
 atl::Conditional<avl::Circle2D>& outCircle
)
```

Parameters

	Name	Type	Default	Description
→	inPoint1	const Point2D&		
→	inPoint2	const Point2D&		
→	inPoint3	const Point2D&		
←	outCircle	Conditional<Circle2D>&		Circle passing through the specified points; or Nil if the points are collinear

Examples



CircleThroughPoints *performed on three points.*

CreateCoordinateSystemFromPoint

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a coordinate system with the specified origin.

Applications: Most often used to define an object alignment from results of 1D Edge Detection or Blob Analysis.

Syntax

```
void avl::CreateCoordinateSystemFromPoint
(
 const avl::Point2D& inPoint,
 float inAngle,
 float inScale,
 float inScaleDivisor,
 avl::CoordinateSystem2D& outCoordinateSystem
)
```

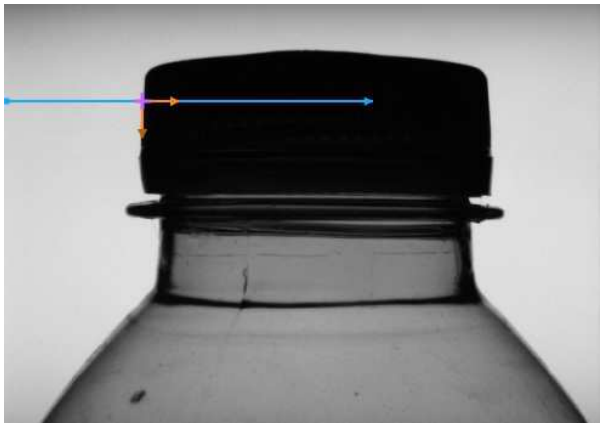
Parameters

	Name	Type	Range	Default	Description
→	inPoint	const Point2D&			Origin of the created coordinate system
→	inAngle	float			
→	inScale	float	0.001 - ∞	1.0f	
→	inScaleDivisor	float	0.001 - ∞	1.0f	
←	outCoordinateSystem	CoordinateSystem2D&			

Hints

- Pass **inPoint** to a computed point, where you want to anchor a new coordinate system.
- Optionally set **inAngle** to define the rotation.
- Optionally set **inScale** and **inScaleDivisor** to obtain a custom scale.

Examples



A local coordinate system created from a point. Here, the point is located with a 1D Edge Detection filter.

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

CreateCoordinateSystemFromRectangle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a coordinate system from a rectangle.

Applications: Most often used to define an object alignment from a filter like RegionBoundingRectangle.

Syntax

```
void avl::CreateCoordinateSystemFromRectangle
(
 const avl::Rectangle2D& inRectangle,
 const avl::Anchor2D::Type inPointAnchor,
 float inRelativeAngle,
 float inScale,
 float inScaleDivisor,
 avl::CoordinateSystem2D& outCoordinateSystem
)
```

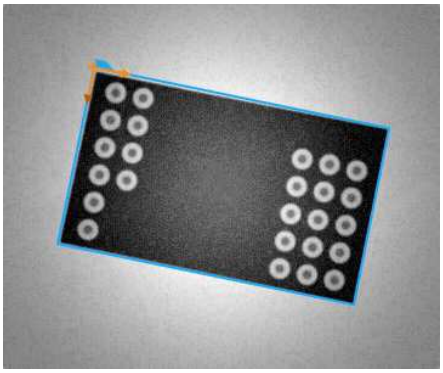
Parameters

	Name	Type	Range	Default	Description
→	inRectangle	const Rectangle2D&			
→	inPointAnchor	const Anchor2D::Type		TopLeft	
→	inRelativeAngle	float			
→	inScale	float	0.001 - ∞	1.0f	
→	inScaleDivisor	float	0.001 - ∞	1.0f	
←	outCoordinateSystem	CoordinateSystem2D&			

Hints

- Pass **inRectangle** to a computed rectangle representing a new coordinate system.
- Optionally set **inScale** and **inScaleDivisor** to obtain a custom scale.
- Optionally set **inRelativeAngle** to modify the rotation.

Examples



A local coordinate system created from a rectangle.

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

CreateCoordinateSystemFromSegment

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a coordinate system with the origin on a given segment.

Syntax

```
void avl::CreateCoordinateSystemFromSegment
(
 const avl::Segment2D& inSegment,
 float inPointAnchor,
 float inRelativeAngle,
 float inScale,
 float inScaleDivisor,
 avl::CoordinateSystem2D& outCoordinateSystem
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSegment	const Segment2D&			
→	inPointAnchor	float	-1.0 - 1.0	0.0f	
→	inRelativeAngle	float			
→	inScale	float	0.001 - ∞	1.0f	
→	inScaleDivisor	float	0.001 - ∞	1.0f	
←	outCoordinateSystem	CoordinateSystem2D&			

Hints

- Pass **inSegment** to a computed segment representing the origin and the rotation of a new coordinate system.
- Optionally set **inScale** and **inScaleDivisor** to obtain a custom scale.
- Optionally set **inPointAnchor** and/or **inRelativeAngle** to modify the origin and the rotation.

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

CreateCoordinateSystemFromTwoPoints

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a coordinate system with two points on its axes.

Applications: Most often used to define an object alignment from results of two 1D Edge Detection scans, each of which locates one side of a rectangular object.

Syntax

```
void avl::CreateCoordinateSystemFromTwoPoints
(
 const avl::Point2D& inPoint1,
 const avl::Point2D& inPoint2,
 float inAngle,
 float inScale,
 float inScaleDivisor,
 avl::CoordinateSystem2D& outCoordinateSystem
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoint1	const Point2D&			
→	inPoint2	const Point2D&			
→	inAngle	float			
→	inScale	float	0.001 - ∞	1.0f	
→	inScaleDivisor	float	0.001 - ∞	1.0f	
←	outCoordinateSystem	CoordinateSystem2D&			

Hints

- Pass**inPoint1** and **inPoint2** to computed points lying on (respectively) the X and the Y axes of a new coordinate system.
- Optionally set **inAngle** to define the rotation.
- Optionally set **inScale** and **inScaleDivisor** to obtain a custom scale.

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

CreateCoordinateSystemFromVector

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a coordinate system from a vector.

Applications: Both vectors and coordinate systems can represent object translations. This operation does a conversion.

Syntax

```
void avl::CreateCoordinateSystemFromVector
(
 const avl::Vector2D& inVector,
 float inAngle,
 float inScale,
 float inScaleDivisor,
 avl::CoordinateSystem2D& outCoordinateSystem
)
```

Parameters

	Name	Type	Range	Default	Description
→	inVector	const Vector2D&			
→	inAngle	float			
→	inScale	float	0.001 - ∞	1.0f	
→	inScaleDivisor	float	0.001 - ∞	1.0f	
←	outCoordinateSystem	CoordinateSystem2D&			

Hints

- Pass **inVector** to a computed vector representing the origin of a new coordinate system.
- Optionally set **inAngle** to define the rotation.
- Optionally set **inScale** and **inScaleDivisor** to obtain a custom scale.

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

EllipseThroughFourPoints

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes an ellipse passing through four noncollinear points.

Syntax

```
void avl::EllipseThroughFourPoints
(
 const avl::Point2D& inPoint1,
 const avl::Point2D& inPoint2,
 const avl::Point2D& inPoint3,
 const avl::Point2D& inPoint4,
 atl::Conditional<avl::Ellipse2D>& outEllipse
)
```

Parameters

	Name	Type	Default	Description
→	inPoint1	const Point2D&		
→	inPoint2	const Point2D&		
→	inPoint3	const Point2D&		
→	inPoint4	const Point2D&		
←	outEllipse	Conditional<Ellipse2D>&		Ellipse passing through the specified points; or Nil if the points don't define a proper ellipse.

EllipseThroughThreePoints

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes an ellipse passing through three noncollinear points.

Syntax

```
void avl::EllipseThroughThreePoints
(
 const avl::Point2D& inPoint1,
 const avl::Point2D& inPoint2,
 const avl::Point2D& inPoint3,
 atl::Conditional<avl::Ellipse2D>& outEllipse
)
```

Parameters

	Name	Type	Default	Description
→	inPoint1	const Point2D&		
→	inPoint2	const Point2D&		
→	inPoint3	const Point2D&		
←	outEllipse	Conditional<Ellipse2D>&		Ellipse passing through the specified points; or Nil if the points are collinear

LineThroughPoint

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes a line passing through a point with given angle.

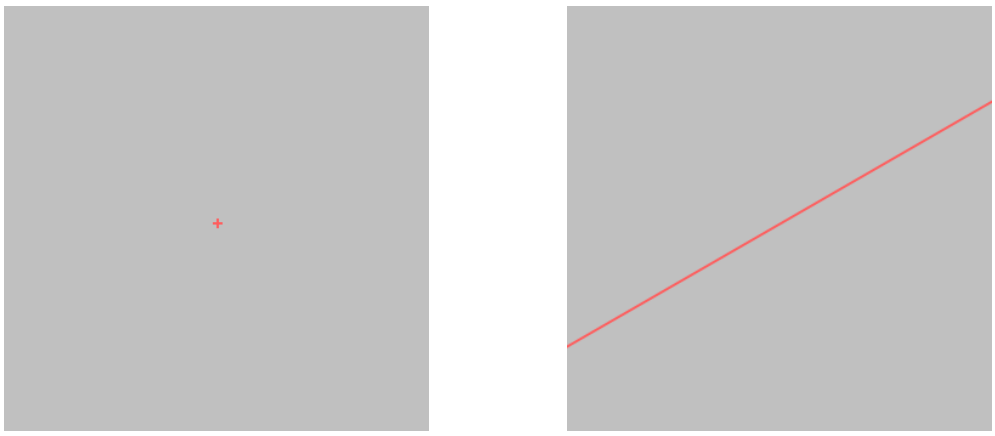
Syntax

```
void avl::LineThroughPoint
(
 const avl::Point2D& inPoint,
 float inAngle,
 avl::Line2D& outLine
)
```

Parameters

	Name	Type	Default	Description
→	inPoint	const Point2D&		
→	inAngle	float		
←	outLine	Line2D&		

Examples



LineThroughPoint *performed on point,* ***inAngle*** *= 150.*

LineThroughPoints

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes a line passing through two distinct points.

Syntax

```
void avl::LineThroughPoints
(
 const avl::Point2D& inPoint1,
 const avl::Point2D& inPoint2,
 avl::Line2D& outLine
)
```

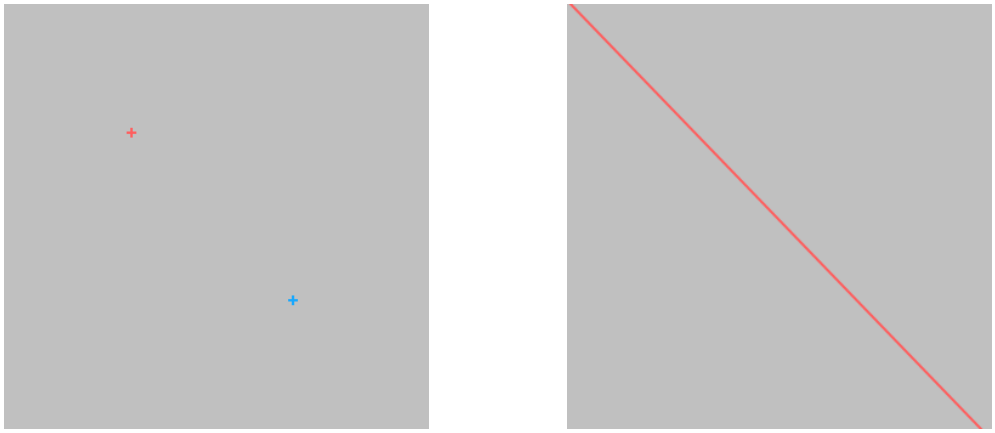
Parameters

	Name	Type	Default	Description
→	inPoint1	const Point2D&		
→	inPoint2	const Point2D&		
←	outLine	Line2D&		

Description

Note that because of inaccuracies of floating-point arithmetic, some geometric operations (including this one) may lead to unpredictable results for degenerated cases. In this filter such a case occurs when the input points are almost equal.

Examples



LineThroughPoints *performed on two points.*

ProjectPointOnCircle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Projects a point onto a circle.

Syntax

```
void avl::ProjectPointOnCircle
(
 const avl::Point2D& inPoint,
 const avl::Circle2D& inCircle,
 avl::Point2D& outProjectionPoint
)
```

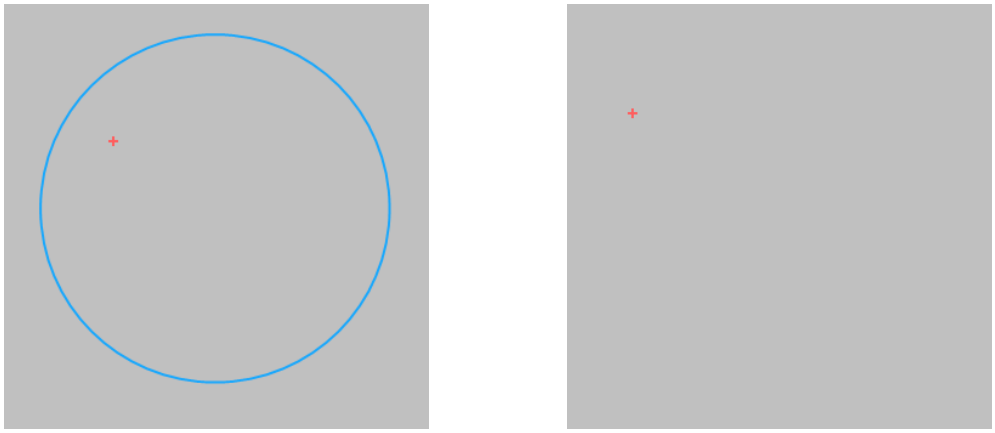
Parameters

	Name	Type	Default	Description
→	inPoint	const Point2D&		
→	inCircle	const Circle2D&		
←	outProjectionPoint	Point2D&		

Description

Note that because of inaccuracies of floating-point arithmetic, some geometric operations (including this one) may lead to unpredictable results for degenerated cases. In this filter such a case occurs when the input point and the center of the input circle are almost equal.

Examples



ProjectPointOnCircle *performed on point and circle.*

ProjectPointOnLine

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Projects a point onto a line.

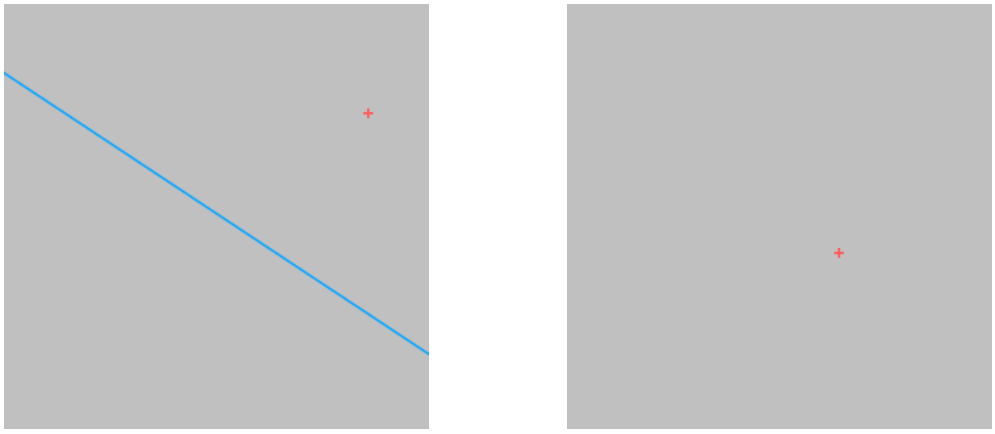
Syntax

```
void avl::ProjectPointOnLine
(
 const avl::Point2D& inPoint,
 const avl::Line2D& inLine,
 avl::Point2D& outProjectionPoint
)
```

Parameters

	Name	Type	Default	Description
→	inPoint	const Point2D&		
→	inLine	const Line2D&		
←	outProjectionPoint	Point2D&		

Examples



ProjectPointOnLine *performed on point and line.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite line on input in ProjectPointOnLine.

ProjectPointsOnCircle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Projects points onto a circle.

Syntax

```
void avl::ProjectPointsOnCircle
(
 const atl::Array<avl::Point2D>& inPoints,
 const avl::Circle2D& inCircle,
 atl::Array<avl::Point2D>& outProjectionPoints,
 atl::Conditional<avl::Arc2D>& outProjectionArc
)
```

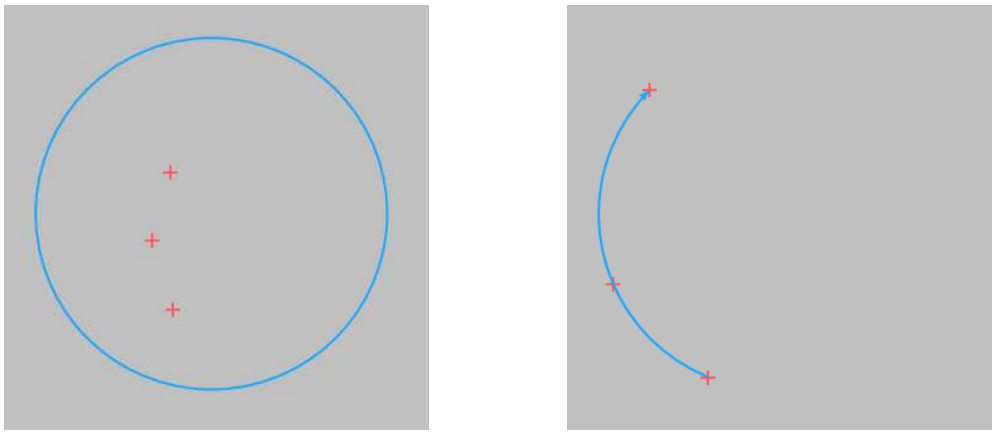
Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		Points to be projected
→	inCircle	const Circle2D&		Circle the points will be projected on
←	outProjectionPoints	Array<Point2D>&		Projected points
←	outProjectionArc	Conditional<Arc2D>&		Arc containing the projected points

Description

Note that because of inaccuracies of floating-point arithmetic, some geometric operations (including this one) may lead to unpredictable results for degenerated cases. In this filter such a case occurs when an input point and the center of the input circle are almost equal.

Examples



ProjectPointsOnCircle *performed on points and circle.*

ProjectPointsOnLine

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Projects points onto a line.

Syntax

```
void avl::ProjectPointsOnLine
(
 const atl::Array<avl::Point2D>& inPoints,
 const avl::Line2D& inLine,
 atl::Array<avl::Point2D>& outProjectionPoints,
 atl::Optional<atl::Conditional<avl::Segment2D>&> outProjectionSegment = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		
→	inLine	const Line2D&		
←	outProjectionPoints	Array<Point2D>&		
←	outProjectionSegment	Optional<Conditional<Segment2D>&>	NIL	

Optional Outputs

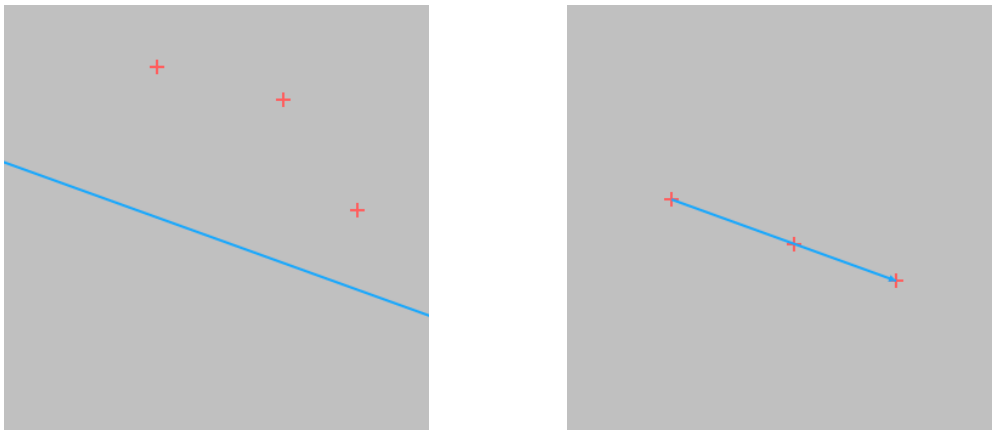
The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outProjectionSegment**.

Read more about Optional Outputs.

Description

The orientation of the resulting **outProjectionSegment** is always between 0 and 180 degrees.

Examples



ProjectPointsOnLine *performed on points and line.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite line on input in ProjectPointsOnLine.

SplitRectangle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Splits the rectangle into two along the direction.

Syntax

```
void avl::SplitRectangle
(
 const avl::Rectangle2D& inRectangle,
 const avl::SplitDirection::Type inSplitDirection,
 avl::Rectangle2D& outRectangle1,
 avl::Rectangle2D& outRectangle2
)
```

Parameters

	Name	Type	Default	Description
→	inRectangle	const Rectangle2D&		
→	inSplitDirection	const SplitDirection::Type		
←	outRectangle1	Rectangle2D&		
←	outRectangle2	Rectangle2D&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Unknown split direction in SplitRectangle.

VectorBetweenPoints

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes a vector representing distance from one point to another.

Syntax

```
void avl::VectorBetweenPoints
(
 const avl::Point2D& inFrom,
 const avl::Point2D& inTo,
 avl::Vector2D& outVector
)
```

Parameters

	Name	Type	Default	Description
→	inFrom	const Point2D&		
→	inTo	const Point2D&		
←	outVector	Vector2D&		

Examples

VectorBetweenPoints *performed on two points.*

60. Geometry 3D Constructions

Table of content:

CircleThroughPoints3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes a circle in 3D passing through three noncollinear 3D points.

Syntax

```
void avl::CircleThroughPoints3D
(
 const avl::Point3D& inPoint1,
 const avl::Point3D& inPoint2,
 const avl::Point3D& inPoint3,
 atl::Conditional<avl::Circle3D>& outCircle3D
)
```

Parameters

	Name	Type	Default	Description
→	inPoint1	const Point3D&		
→	inPoint2	const Point3D&		
→	inPoint3	const Point3D&		
←	outCircle3D	Conditional<Circle3D>&		Circle in 3D passing through the specified 3D points; or Nil if the points are collinear

LineThroughPoint3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes a line in 3D passing through a point with given direction.

Syntax

```
void avl::LineThroughPoint3D
(
 const avl::Point3D& inPoint3D,
 const avl::Vector3D& inDirectionVector,
 avl::Line3D& outLine3D
)
```

Parameters

	Name	Type	Default	Description
→	inPoint3D	const Point3D&		A point on the created line
→	inDirectionVector	const Vector3D&		Desired direction of the line
←	outLine3D	Line3D&		The resulting line in 3D

LineThroughPoints3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes a line in 3D passing through two points.

Syntax

```
void avl::LineThroughPoints3D
(
 const avl::Point3D& inPoint1,
 const avl::Point3D& inPoint2,
 avl::Line3D& outLine3D
)
```

Parameters

	Name	Type	Default	Description
→	inPoint1	const Point3D&		A point on the created line
→	inPoint2	const Point3D&		Another point on the created line
←	outLine3D	Line3D&		The resulting line in 3D

PlaneThroughLineAndPoint3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes a plane passing through a line and a point in 3D.

Syntax

```
void avl::PlaneThroughLineAndPoint3D
(
 const avl::Line3D& inLine3D,
 const avl::Point3D& inPoint3D,
 avl::Plane3D& outPlane
)
```

Parameters

	Name	Type	Default	Description
→	inLine3D	const Line3D&		
→	inPoint3D	const Point3D&		
←	outPlane	Plane3D&		

PlaneThroughPoints3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes a plane passing through three distinct points in 3D.

Syntax

```
void avl::PlaneThroughPoints3D
(
 const avl::Point3D& inPoint1,
 const avl::Point3D& inPoint2,
 const avl::Point3D& inPoint3,
 avl::Plane3D& outPlane
)
```

Parameters

	Name	Type	Default	Description
→	inPoint1	const Point3D&		
→	inPoint2	const Point3D&		
→	inPoint3	const Point3D&		
←	outPlane	Plane3D&		

ProjectPointsOnLine3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Projects points onto a line.

Syntax

```
void avl::ProjectPointsOnLine3D
(
 const atl::Array<avl::Point3D>& inPoints,
 const avl::Line3D& inLine,
 atl::Array<avl::Point3D>& outProjectionPoints,
 atl::Optional<atl::Conditional<avl::Segment3D>&> outProjectionSegment = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point3D>&		
→	inLine	const Line3D&		
←	outProjectionPoints	Array<Point3D>&		
←	outProjectionSegment	Optional<Conditional<Segment3D>&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outProjectionSegment**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite line in inLine in ProjectPointsOnLine3D.

VectorBetweenPoints3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes a vector representing distance from one point in 3D to another.

Syntax

```
void avl::VectorBetweenPoints3D
(
 const avl::Point3D& inFrom,
 const avl::Point3D& inTo,
 avl::Vector3D& outVector3D
)
```

Parameters

	Name	Type	Default	Description
→	inFrom	const Point3D&		
→	inTo	const Point3D&		
←	outVector3D	Vector3D&		

61. Geometry 2D Distance Metrics

Table of content:

CircleToCircleDistance

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Measures the distance between two circles.

Syntax

```
void avl::CircleToCircleDistance
(
 const avl::Circle2D& inCircle1,
 const avl::Circle2D& inCircle2,
 float inResolution,
 float& outDistance,
 atl::Optional<avl::Segment2D&> outConnectingSegment = atl::NIL
)
```

Parameters

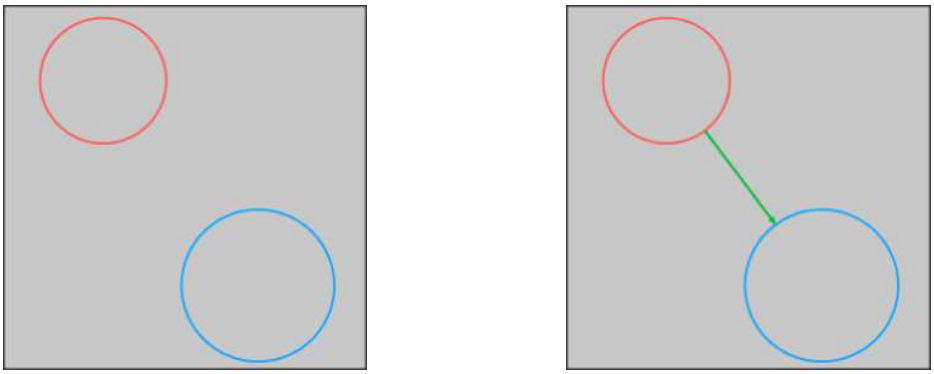
	Name	Type	Range	Default	Description
→	inCircle1	const Circle2D&			
→	inCircle2	const Circle2D&			
→	inResolution	float	0.0 - ∞	1.0f	Number of real-world units per one pixel
←	outDistance	float&			
←	outConnectingSegment	Optional<Segment2D&>		NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outConnectingSegment**.

Read more about Optional Outputs.

Examples



CircleToCircleDistance *performed on two sample circles. The green segment is the value of* ***outConnectingSegment*** *output.*

PointSequenceDistances

Header: AVL.h
Namespace: avl
Module: FoundationLite

Measures the distances between consecutive points of a point sequence.

Syntax

```
void avl::PointSequenceDistances
(
 const atl::Array<avl::Point2D>& inPoints,
 const bool inCyclic,
 float inResolution,
 atl::Array<float>& outDistances,
 atl::Optional<float&> outDistanceSum = atl::NIL,
 atl::Optional<atl::Array<avl::Segment2D>&> outConnectingSegments = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoints	const Array<Point2D>&			
→	inCyclic	const bool			
→	inResolution	float	0.0 - ∞	1.0f	Number of real-world units per one pixel
←	outDistances	Array<float>&			
←	outDistanceSum	Optional<float&>		NIL	
←	outConnectingSegments	Optional<Array<Segment2D>&>		NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outDistanceSum**, **outConnectingSegments**.

Read more about Optional Outputs.

PointToArcDistance

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Measures the distance between a point and an arc.

Syntax

```
void avl::PointToArcDistance
(
 const avl::Point2D& inPoint,
 const avl::Arc2D& inArc,
 float inResolution,
 float& outDistance,
 atl::Optional<avl::Segment2D&> outConnectingSegment = atl::NIL
)
```

Parameters

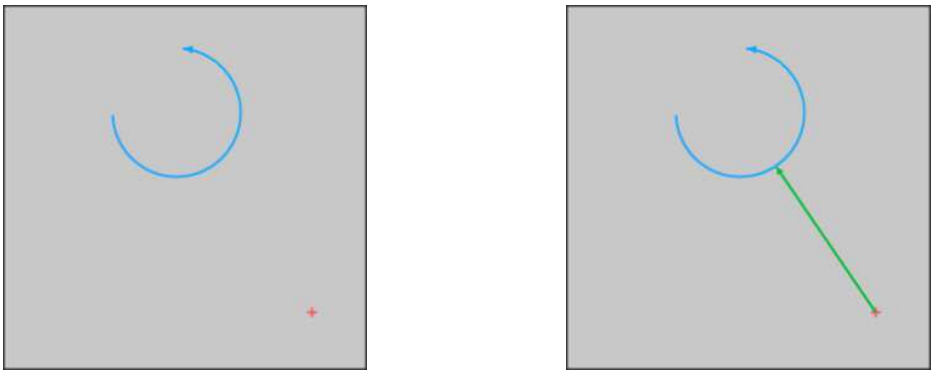
	Name	Type	Range	Default	Description
→	inPoint	const Point2D&			
→	inArc	const Arc2D&			
→	inResolution	float	0.0 - ∞	1.0f	Number of real-world units per one pixel
←	outDistance	float&			
←	outConnectingSegment	Optional<Segment2D&>		NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outConnectingSegment**.

Read more about Optional Outputs.

Examples



PointToArcDistance *performed on a sample point and a sample arc. The green segment is the value of* ***outConnectingSegment*** *output.*

PointToCircleDistance

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Measures the distance between a point and a circle.

Syntax

```
void avl::PointToCircleDistance
(
 const avl::Point2D& inPoint,
 const avl::Circle2D& inCircle,
 float inResolution,
 float& outDistance,
 atl::Optional<avl::Segment2D&> outConnectingSegment = atl::NIL
)
```

Parameters

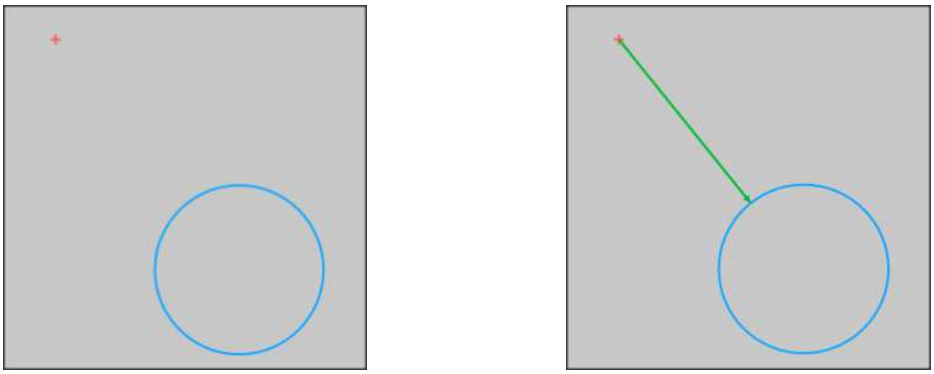
	Name	Type	Range	Default	Description
→	inPoint	const Point2D&			
→	inCircle	const Circle2D&			
→	inResolution	float	0.0 - ∞	1.0f	Number of real-world units per one pixel
←	outDistance	float&			
←	outConnectingSegment	Optional<Segment2D&>		NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outConnectingSegment**.

Read more about Optional Outputs.

Examples



PointToCircleDistance *performed on a sample point and a sample circle. The green segment is the value of* ***outConnectingSegment*** *output.*

PointToLineDistance

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Measures the distance between a point and a line.

Syntax

```
void avl::PointToLineDistance
(
 const avl::Point2D& inPoint,
 const avl::Line2D& inLine,
 float inResolution,
 float& outDistance,
 atl::Optional<avl::Segment2D&> outConnectingSegment = atl::NIL,
 atl::Optional<float&> outSignedDistance = atl::NIL
)
```

Parameters

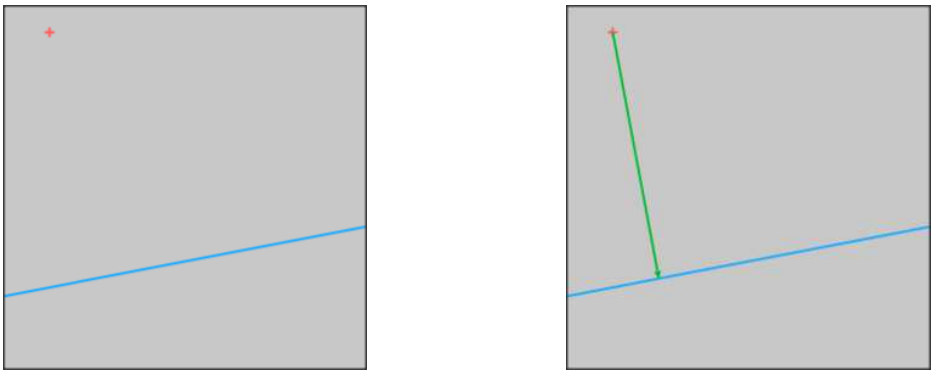
	Name	Type	Range	Default	Description
→	inPoint	const Point2D&			
→	inLine	const Line2D&			
→	inResolution	float	0.0 - ∞	1.0f	Number of real-world units per one pixel
←	outDistance	float&			
←	outConnectingSegment	Optional<Segment2D&>		NIL	
←	outSignedDistance	Optional<float&>		NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outConnectingSegment**, **outSignedDistance**.

Read more about Optional Outputs.

Examples



PointToLineDistance *performed on a sample point and a sample line. The green segment is the value of* ***outConnectingSegment*** *output.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite line on input in PointToLineDistance.

PointToLineDistance_Oriented

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Measures the signed distance between a point and a line with direction.

Syntax

```
void avl::PointToLineDistance_Oriented
(
 const avl::Point2D& inPoint,
 const avl::Segment2D& inLineSection,
 float inResolution,
 float& outDistance,
 atl::Optional<avl::Segment2D&> outConnectingSegment = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoint	const Point2D&			
→	inLineSection	const Segment2D&			
→	inResolution	float	0.0 - ∞	1.0f	Number of real-world units per one pixel
←	outDistance	float&			
←	outConnectingSegment	Optional<Segment2D&>		NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outConnectingSegment**.

Read more about Optional Outputs.

PointToPointDistance

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Measures the distance between two points.

Syntax

```
void avl::PointToPointDistance
(
 const avl::Point2D& inPoint1,
 const avl::Point2D& inPoint2,
 float inResolution,
 float& outDistance,
 atl::Optional<avl::Segment2D&> outConnectingSegment = atl::NIL
)
```

Parameters

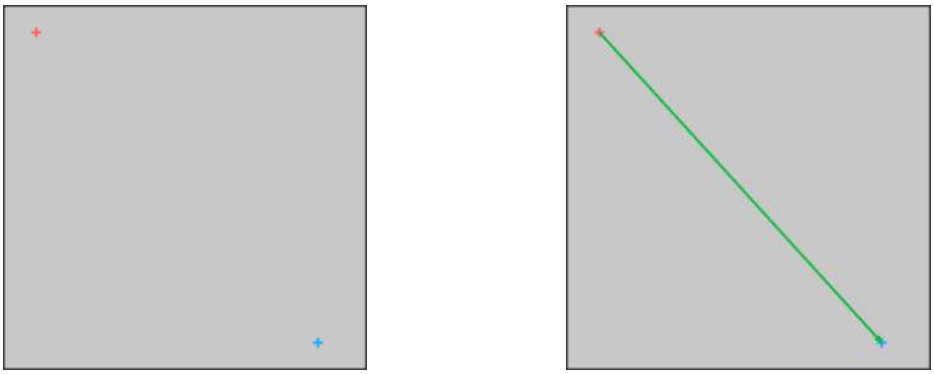
	Name	Type	Range	Default	Description
→	inPoint1	const Point2D&			
→	inPoint2	const Point2D&			
→	inResolution	float	0.0 - ∞	1.0f	Number of real-world units per one pixel
←	outDistance	float&			
←	outConnectingSegment	Optional<Segment2D&>		NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outConnectingSegment**.

Read more about Optional Outputs.

Examples



PointToPointDistance *performed on two sample points. The green segment is the value of* ***outConnectingSegment*** *output.*

PointToRectangleDistance

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Measures the distance between point and rectangle.

Syntax

```
void avl::PointToRectangleDistance
(
 const avl::Point2D& inPoint,
 const avl::Rectangle2D& inRectangle,
 const float inResolution,
 const bool inIsFilled,
 float& outDistance,
 atl::Conditional<avl::Segment2D>& outConnectingSegment
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoint	const Point2D&			
→	inRectangle	const Rectangle2D&			
→	inResolution	const float	0.0 - ∞	1.0f	Number of real-world units per one pixel
→	inIsFilled	const bool		True	Whether the rectangle is filled inside or not
←	outDistance	float&			
←	outConnectingSegment	Conditional<Segment2D>&			

PointToSegmentDistance

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Measures the distance between a point and a segment.

Syntax

```
void avl::PointToSegmentDistance
(
 const avl::Point2D& inPoint,
 const avl::Segment2D& inSegment,
 float inResolution,
 float& outDistance,
 atl::Optional<avl::Segment2D&> outConnectingSegment = atl::NIL
)
```

Parameters

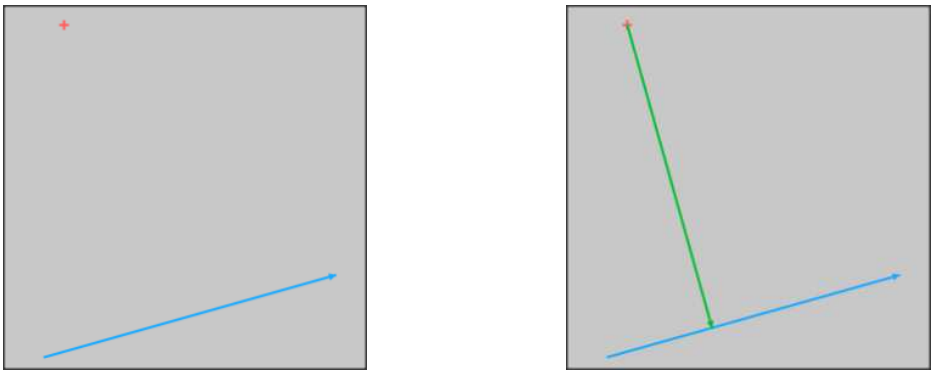
	Name	Type	Range	Default	Description
→	inPoint	const Point2D&			
→	inSegment	const Segment2D&			
→	inResolution	float	0.0 - ∞	1.0f	Number of real-world units per one pixel
←	outDistance	float&			
←	outConnectingSegment	Optional<Segment2D&>		NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outConnectingSegment**.

Read more about Optional Outputs.

Examples



PointToSegmentDistance *performed on a sample point and a sample segment. The green segment is the value of* ***outConnectingSegment*** *output.*

SegmentToSegmentDistance

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Measures the distance between two segments.

Syntax

```
void avl::SegmentToSegmentDistance
(
 const avl::Segment2D& inSegment1,
 const avl::Segment2D& inSegment2,
 float inResolution,
 float& outDistance,
 atl::Optional<avl::Segment2D&> outConnectingSegment = atl::NIL
)
```

Parameters

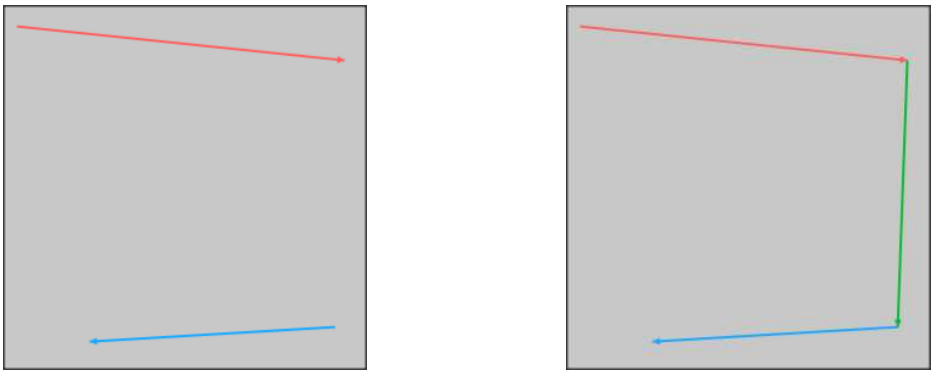
	Name	Type	Range	Default	Description
→	inSegment1	const Segment2D&			
→	inSegment2	const Segment2D&			
→	inResolution	float	0.0 - ∞	1.0f	Number of real-world units per one pixel
←	outDistance	float&			
←	outConnectingSegment	Optional<Segment2D&>		NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outConnectingSegment**.

Read more about Optional Outputs.

Examples



SegmentToSegmentDistance *performed on two sample segments. The green segment is the value of* ***outConnectingSegment*** *output.*

62. Geometry 3D Distance Metrics

Table of content:

CircleToPlaneDistance3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Measures the distance between a circle in 3D and a plane.

Syntax

```
void avl::CircleToPlaneDistance3D
(
 const avl::Circle3D& inCircle3D,
 const avl::Plane3D& inPlane,
 float inResolution,
 float& outDistance,
 atl::Optional<avl::Segment3D&> outConnectingSegment3D = atl::NIL,
 atl::Optional<float&> outSignedDistance = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inCircle3D	const Circle3D&			
→	inPlane	const Plane3D&			
→	inResolution	float	0.0 - ∞	1.0f	Number of real-world units per one pixel
←	outDistance	float&			
←	outConnectingSegment3D	Optional<Segment3D&>		NIL	
←	outSignedDistance	Optional<float&>		NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outConnectingSegment3D**, **outSignedDistance**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite circle on input in CircleToPlaneDistance3D.
DomainError	Indefinite plane on input in CircleToPlaneDistance3D.

LineToLineDistance3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Measures the distance between two lines in 3D.

Syntax

```
void avl::LineToLineDistance3D
(
 const avl::Line3D& inLine1,
 const avl::Line3D& inLine2,
 float inResolution,
 float& outDistance,
 atl::Optional<avl::Segment3D&> outConnectingSegment3D = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inLine1	const Line3D&			
→	inLine2	const Line3D&			
→	inResolution	float	0.0 - ∞	1.0f	Number of real-world units per one pixel
←	outDistance	float&			
←	outConnectingSegment3D	Optional<Segment3D&>		NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outConnectingSegment3D**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite line on input in LineToLineDistance3D.

PointSequenceDistances3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Measures the distances between consecutive points of a point sequence in 3D.

Syntax

```
void avl::PointSequenceDistances3D
(
 const atl::Array<avl::Point3D>& inPoints,
 const bool inCyclic,
 atl::Array<float>& outDistances,
 atl::Optional<float&> outDistanceSum = atl::NIL,
 atl::Optional<atl::Array<avl::Segment3D>&> outConnectingSegments = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point3D>&		
→	inCyclic	const bool		
←	outDistances	Array<float>&		
←	outDistanceSum	Optional<float&>	NIL	
←	outConnectingSegments	Optional<Array<Segment3D>&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outDistanceSum**, **outConnectingSegments**.

Read more about Optional Outputs.

PointToCircleDistance3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Measures the distance between a 3D point and a circle in 3D.

Syntax

```
void avl::PointToCircleDistance3D
(
 const avl::Point3D& inPoint3D,
 const avl::Circle3D& inCircle3D,
 float inResolution,
 float& outDistance,
 atl::Optional<avl::Segment3D&> outConnectingSegment3D = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoint3D	const Point3D&			
→	inCircle3D	const Circle3D&			
→	inResolution	float	0.0 - ∞	1.0f	Number of real-world units per one pixel
←	outDistance	float&			
←	outConnectingSegment3D	Optional<Segment3D&>		NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outConnectingSegment3D**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite circle on input in PointToCircleDistance3D.

PointToLineDistance3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Measures the distance between a 3D point and a line in 3D.

Syntax

```
void avl::PointToLineDistance3D
(
 const avl::Point3D& inPoint3D,
 const avl::Line3D& inLine3D,
 float inResolution,
 float& outDistance,
 atl::Optional<avl::Segment3D&> outConnectingSegment3D = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoint3D	const Point3D&			
→	inLine3D	const Line3D&			
→	inResolution	float	0.0 - ∞	1.0f	Number of real-world units per one pixel
←	outDistance	float&			
←	outConnectingSegment3D	Optional<Segment3D&>		NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outConnectingSegment3D**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite line on input in PointToLineDistance3D.

PointToPlaneDistance3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Measures the distance between a 3D point and a plane.

Syntax

```
void avl::PointToPlaneDistance3D
(
 const avl::Point3D& inPoint3D,
 const avl::Plane3D& inPlane,
 float inResolution,
 float& outDistance,
 atl::Optional<avl::Segment3D&> outConnectingSegment3D = atl::NIL,
 atl::Optional<float&> outSignedDistance = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoint3D	const Point3D&			
→	inPlane	const Plane3D&			
→	inResolution	float	0.0 - ∞	1.0f	Number of real-world units per one pixel
←	outDistance	float&			
←	outConnectingSegment3D	Optional<Segment3D&>		NIL	
←	outSignedDistance	Optional<float&>		NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outConnectingSegment3D**, **outSignedDistance**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite plane on input in PointToPlaneDistance3D.

PointToPointDistance3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Measures the distance between two 3D points.

Syntax

```
void avl::PointToPointDistance3D
(
 const avl::Point3D& inPoint1,
 const avl::Point3D& inPoint2,
 float inResolution,
 float& outDistance,
 atl::Optional<avl::Segment3D&> outConnectingSegment = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoint1	const Point3D&			
→	inPoint2	const Point3D&			
→	inResolution	float	0.0 - ∞	1.0f	Number of real-world units per one pixel
←	outDistance	float&			
←	outConnectingSegment	Optional<Segment3D&>		NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outConnectingSegment**.

Read more about Optional Outputs.

PointToSegmentDistance3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Measures the distance between a 3D point and a segment in 3D.

Syntax

```
void avl::PointToSegmentDistance3D
(
 const avl::Point3D& inPoint3D,
 const avl::Segment3D& inSegment3D,
 float inResolution,
 float& outDistance,
 atl::Optional<avl::Segment3D&> outConnectingSegment3D = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoint3D	const Point3D&			
→	inSegment3D	const Segment3D&			
→	inResolution	float	0.0 - ∞	1.0f	Number of real-world units per one pixel
←	outDistance	float&			
←	outConnectingSegment3D	Optional<Segment3D&>		NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outConnectingSegment3D**.

Read more about Optional Outputs.

SegmentToPlaneDistance3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Measures the distance between a segment in 3D and a plane.

Syntax

```
void avl::SegmentToPlaneDistance3D
(
 const avl::Segment3D& inSegment3D,
 const avl::Plane3D& inPlane,
 float inResolution,
 float& outDistance,
 atl::Optional<avl::Segment3D&> outConnectingSegment3D = atl::NIL,
 atl::Optional<float&> outSignedDistance = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSegment3D	const Segment3D&			
→	inPlane	const Plane3D&			
→	inResolution	float	0.0 - ∞	1.0f	Number of real-world units per one pixel
←	outDistance	float&			
←	outConnectingSegment3D	Optional<Segment3D&>		NIL	
←	outSignedDistance	Optional<float&>		NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outConnectingSegment3D**, **outSignedDistance**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite plane on input in SegmentToPlaneDistance3D.

SegmentToSegmentDistance3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Measures the distance between two segments in 3D.

Syntax

```
void avl::SegmentToSegmentDistance3D
(
 const avl::Segment3D& inSegment1,
 const avl::Segment3D& inSegment2,
 float inResolution,
 float& outDistance,
 atl::Optional<avl::Segment3D&> outConnectingSegment3D = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSegment1	const Segment3D&			
→	inSegment2	const Segment3D&			
→	inResolution	float	0.0 - ∞	1.0f	Number of real-world units per one pixel
←	outDistance	float&			
←	outConnectingSegment3D	Optional<Segment3D&>		NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outConnectingSegment3D**.

Read more about Optional Outputs.

63. Path Classification

Table of content:

ClassifyPaths

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Splits the paths of the input array - in accordance to the relation between computed feature values and the specified range.

Applications: Use this filter when you have an array of paths and you want to select some of them for further processing.

Syntax

```
void avl::ClassifyPaths
(
 const atl::Array<avl::Path>& inPaths,
 avl::PathFilter::Type inPathFilter,
 avl::PathFeature::Type inFeature,
 atl::Optional<float> inMinimum,
 atl::Optional<float> inMaximum,
 atl::Optional<atl::Array<avl::Path>&> outAccepted,
 atl::Optional<atl::Array<avl::Path>&> outRejected = atl::NIL,
 atl::Optional<atl::Array<avl::Path>&> outBelow = atl::NIL,
 atl::Optional<atl::Array<avl::Path>&> outAbove = atl::NIL,
 atl::Optional<atl::Array<float>&> outValues = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPaths	const Array<Path>&		Input paths
→	inPathFilter	PathFilter::Type		Determines which paths will take part in computation
→	inFeature	PathFeature::Type		Path feature value to be computed
→	inMinimum	Optional<float>	NIL	Lowest value of the range
→	inMaximum	Optional<float>	NIL	Highest value of the range
←	outAccepted	Optional<Array<Path>&>		Paths with feature values matching the range
←	outRejected	Optional<Array<Path>&>	NIL	Paths with feature values outside the range
←	outBelow	Optional<Array<Path>&>	NIL	Paths with feature values lower than inMinimum
←	outAbove	Optional<Array<Path>&>	NIL	Paths with feature values higher than inMaximum
←	outValues	Optional<Array<float>&>	NIL	Computed feature values

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAccepted**, **outRejected**, **outBelow**, **outAbove**, **outValues**.

Read more about Optional Outputs.

Description

The filter accepts an array of paths and splits it into output arrays, depending on how each of the computed feature values fits the (**inMinimum**, **inMaximum**) range.

- Paths corresponding to feature values lower than **inMinimum** are passed onto **outBelow** and **outRejected**.
- Paths corresponding to feature values that fit closed range (**inMinimum**, **inMaximum**) are passed onto **outAccepted**.
- Paths corresponding to feature values higher than **inMaximum** are passed onto **outAbove** and **outRejected**.

In the special case of **inMinimum** being greater than **inMaximum**, first matching condition is applied, which means that objects corresponding to values higher than **inMaximum** and lower than **inMinimum** are passed onto **outBelow**.

Paths can be filtered before classification. One can process all paths, only open or only closed ones, depending on value of **inPathFilter** input.

To learn about possible features to classify paths, one should see PathFeature documentation. To know details about particular feature, corresponding filter article should be read.

Hints

- Pass an array of paths to the **inPaths** input.
- Using the **inFeature** input select a feature that well separates the interesting objects.
- Using the **inPathFilter** input you can also separate only closed or only open paths.
- Set the range of **inMinimum** and **inMaximum** to define the accepted objects. Refer to the **outValues** output to see the feature values for all input objects.

See Also

- ClassifyByRange – Separates the elements of the input array into three output arrays, depending on whether the related values fall below, into or above the specified range.
- GetMaximumPath – Returns the path from the input array that corresponds to the largest computed feature value.
- GetMinimumPath – Returns the path from the input array that corresponds to the smallest computed feature value.
- SortPaths – Changes the order of paths from the input array according to an ascending/descending sequence of their computed feature values.
- SelectInnerPaths – Selects paths which are visible from a point.
- SelectOuterPaths – Selects paths which do not obstruct visibility of other paths from a point.

GetMaximumPath

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Returns the path from the input array that corresponds to the largest computed feature value.

Applications: Use this filter when you have an array of paths and you want to select one of them that best matches some criterion.

Syntax

```
void avl::GetMaximumPath
(
 const atl::Array<avl::Path>& inPaths,
 avl::PathFeature::Type inFeature,
 avl::Path& outPath,
 atl::Optional<float&> outValue = atl::NIL,
 atl::Optional<int&> outIndex = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPaths	const Array<Path>&		Input paths
→	inFeature	PathFeature::Type		Path feature value to be computed
←	outPath	Path&		Output path
←	outValue	Optional<float&>	NIL	Computed feature value of the output path
←	outIndex	Optional<int&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outValue**, **outIndex**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty path array on input in GetMaximumPath.

GetMaximumPath_OrNil

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Returns the path from the input array that corresponds to the largest computed feature value; returns NIL if the array or any path inside it is empty.

Applications: Use this filter when you have an array of paths and you want to select one of them that best matches some criterion.

Syntax

```
void avl::GetMaximumPath_OrNil
(
 const atl::Array<avl::Path>& inPaths,
 avl::PathFeature::Type inFeature,
 atl::Conditional<avl::Path>& outPath,
 atl::Conditional<float>& outValue,
 atl::Conditional<int>& outIndex
)
```

Parameters

	Name	Type	Default	Description
→	inPaths	const Array<Path>&		Input paths
→	inFeature	PathFeature::Type		Path feature value to be computed
←	outPath	Conditional<Path>&		Output path
←	outValue	Conditional<float>&		Computed feature value of the output path
←	outIndex	Conditional<int>&		

GetMinimumPath

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Returns the path from the input array that corresponds to the smallest computed feature value.

Applications: Use this filter when you have an array of paths and you want to select one of them that best matches some criterion.

Syntax

```
void avl::GetMinimumPath
(
 const atl::Array<avl::Path>& inPaths,
 avl::PathFeature::Type inFeature,
 avl::Path& outPath,
 atl::Optional<float&> outValue = atl::NIL,
 atl::Optional<int&> outIndex = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPaths	const Array<Path>&		Input paths
→	inFeature	PathFeature::Type		Path feature value to be computed
←	outPath	Path&		Output path
←	outValue	Optional<float&>	NIL	Computed feature value of the output path
←	outIndex	Optional<int&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outValue**, **outIndex**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty path array on input in GetMinimumPath.

GetMinimumPath_OrNil

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Returns the path from the input array that corresponds to the smallest computed feature value; returns NIL if the array or any path inside it is empty.

Applications: Use this filter when you have an array of paths and you want to select one of them that best matches some criterion.

Syntax

```
void avl::GetMinimumPath_OrNil
(
 const atl::Array<avl::Path>& inPaths,
 avl::PathFeature::Type inFeature,
 atl::Conditional<avl::Path>& outPath,
 atl::Conditional<float>& outValue,
 atl::Conditional<int>& outIndex
)
```

Parameters

	Name	Type	Default	Description
→	inPaths	const Array<Path>&		Input paths
→	inFeature	PathFeature::Type		Path feature value to be computed
←	outPath	Conditional<Path>&		Output path
←	outValue	Conditional<float>&		Computed feature value of the output path
←	outIndex	Conditional<int>&		

SelectClosedPaths

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Selects paths which are closed.

Syntax

```
void avl::SelectClosedPaths
(
 const atl::Array<avl::Path>& inPaths,
 atl::Array<avl::Path>& outClosedPaths
)
```

Parameters

	Name	Type	Default	Description
→	inPaths	const Array<Path>&		Input paths
←	outClosedPaths	Array<Path>&		Paths classified as closed

SelectInnerPaths

Header: AVL.h
Namespace: avl
Module: FoundationPro

Selects paths which are visible from a point.

Syntax

```
void avl::SelectInnerPaths
(
 const atl::Array<avl::Path>& inPaths,
 const avl::Point2D& inCenterPoint,
 float inTolerance,
 atl::Array<avl::Path>& outInnerPaths
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPaths	const Array<Path>&			Input paths from which the inner ones will be selected
→	inCenterPoint	const Point2D&			Reference point used to determine path visibility
→	inTolerance	float	0.0 - 1.0	0.0f	Fraction of the path characteristic points that are allowed to be invisible from the inCenterPoint not rendering the path not-inner
←	outInnerPaths	Array<Path>&			Paths classified as inner

Description

The operation select those paths from the **inPaths** array that are *visible* from **inCenterPoint**.

A point is *visible* iff the segment connecting this point to **inCenterPoint** does not intersect any other path in **inPaths**.

Examples

SelectInnerPaths *run on the sample data with* ***inTolerance*** *= 0.0.*

See Also

- SelectOuterPaths – Selects paths which do not obstruct visibility of other paths from a point.

SelectOpenPaths

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Selects paths which are open.

Syntax

```
void avl::SelectOpenPaths
(
 const atl::Array<avl::Path>& inPaths,
 atl::Array<avl::Path>& outOpenPaths
)
```

Parameters

	Name	Type	Default	Description
→	inPaths	const Array<Path>&		Input paths
←	outOpenPaths	Array<Path>&		Paths classified as open

SelectOuterPaths

Header: AVL.h
Namespace: avl
Module: FoundationPro

Selects paths which do not obstruct visibility of other paths from a point.

Syntax

```
void avl::SelectOuterPaths
(
 const atl::Array<avl::Path>& inPaths,
 const avl::Point2D& inCenterPoint,
 float inTolerance,
 atl::Array<avl::Path>& outOuterPaths
)
```

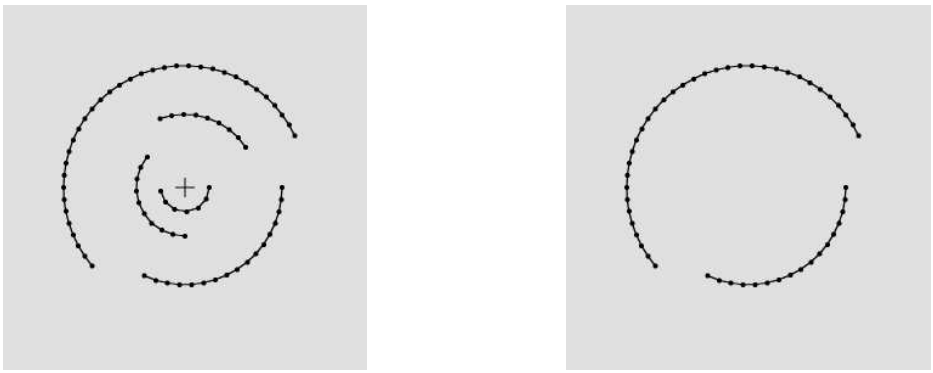
Parameters

	Name	Type	Range	Default	Description
→	inPaths	const Array<Path>&			Input paths from which the outer ones will be selected
→	inCenterPoint	const Point2D&			Reference point used to determine path visibility
→	inTolerance	float	0.0 - 1.0	0.0f	Fraction of the path sections that are allowed to obstruct visibility of other paths not rendering the path not-outer
←	outOuterPaths	Array<Path>&			Paths classified as outer

Description

The operation select those paths from the **inPaths** array that do not obstruct the visibility of any other path. *Visibility* of a characteristic point is obstructed by a path iff the segment connecting this point to **inCenterPoint** intersect the path.

Examples



SelectOuterPaths *run on the sample data with* ***inTolerance*** *= 0.0.*

See Also

- SelectInnerPaths – Selects paths which are visible from a point.

SortPaths

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Changes the order of paths from the input array according to an ascending/descending sequence of their computed feature values.

Syntax

```
void avl::SortPaths
(
 const atl::Array<avl::Path>& inPaths,
 avl::PathFilter::Type inPathFilter,
 avl::PathFeature::Type inFeature,
 avl::SortingOrder::Type inSortingOrder,
 atl::Array<avl::Path>& outSortedPaths,
 atl::Optional<atl::Array<float>&> outSortedValues = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPaths	const Array<Path>&		Input paths
→	inPathFilter	PathFilter::Type		Determines which paths will take part in computation
→	inFeature	PathFeature::Type		Path feature value to be computed
→	inSortingOrder	SortingOrder::Type		Sorting order
←	outSortedPaths	Array<Path>&		Paths sorted according to the computed feature values
←	outSortedValues	Optional<Array<float>&>	NIL	Computed feature values

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outSortedValues**.

Read more about Optional Outputs.

64. Region Relations

Table of content:

ClassifyRegions

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Splits an array of regions according to the selected feature and range.

Applications: Use this filter when you have an array of regions and you want to select some of them for further processing.

Syntax

```
void avl::ClassifyRegions
(
 const atl::Array<avl::Region>& inRegions,
 avl::RegionFeature::Type inFeature,
 atl::Optional<float> inMinimum,
 atl::Optional<float> inMaximum,
 atl::Optional<atl::Array<avl::Region>&> outAccepted,
 atl::Optional<atl::Array<avl::Region>&> outRejected = atl::NIL,
 atl::Optional<atl::Array<avl::Region>&> outBelow = atl::NIL,
 atl::Optional<atl::Array<avl::Region>&> outAbove = atl::NIL,
 atl::Optional<atl::Array<float>&> outValues = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inRegions	const Array<Region>&		Input regions
→	inFeature	RegionFeature::Type		Region feature value to be computed
→	inMinimum	Optional<float>	NIL	Lowest value of the range
→	inMaximum	Optional<float>	NIL	Highest value of the range
←	outAccepted	Optional<Array<Region>&>		Regions with feature values matching the range
←	outRejected	Optional<Array<Region>&>	NIL	Regions with feature values outside the range
←	outBelow	Optional<Array<Region>&>	NIL	Regions with feature values lower than inMinimum
←	outAbove	Optional<Array<Region>&>	NIL	Regions with feature values higher than inMaximum
←	outValues	Optional<Array<float>&>	NIL	Computed feature values

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAccepted**, **outRejected**, **outBelow**, **outAbove**, **outValues**.

Read more about Optional Outputs.

Description

The filter accepts an array of regions and splits it into output arrays, depending on how each of the computed feature values fits the (**inMinimum**, **inMaximum**) range.

- Regions corresponding to feature values lower than **inMinimum** are passed onto **outBelow** and **outRejected**.
- Regions corresponding to feature values that fit closed range (**inMinimum**, **inMaximum**) are passed onto **outAccepted**.
- Regions corresponding to feature values higher than **inMaximum** are passed onto **outAbove** and **outRejected**.

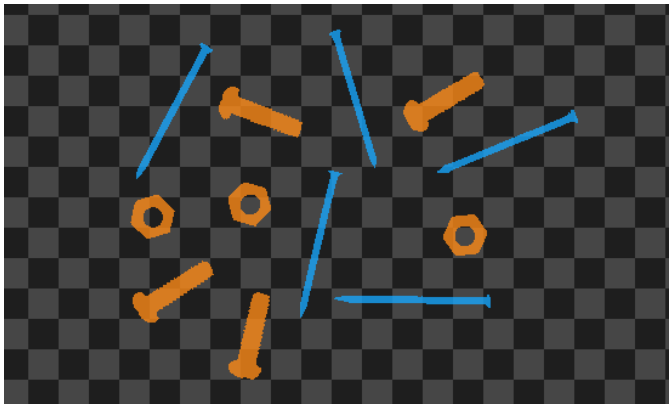
In the special case of **inMinimum** being greater than **inMaximum**, first matching condition is applied, which means that objects corresponding to values higher than **inMaximum** and lower than **inMinimum** are passed onto **outBelow**.

To learn about possible features to classify regions, one should see RegionFeature documentation. To know details about particular feature, corresponding filter article should be read.

Hints

- Pass an array of regions to the **inRegions** input.
- Using the **inFeature** input select a feature that well separates the interesting objects.
- Set the range of **inMinimum** and **inMaximum** to define the accepted objects. Refer to the **outValues** output to see the feature values for all input objects.

Examples



*ClassifyRegions performed with **Elongation** selected as the region feature and **inMaximum** = 10. The blue regions are from **outAccepted** output, the orange ones from **outRejected**.*

See Also

- ClassifyByRange – Separates the elements of the input array into three output arrays, depending on whether the related values fall below, into or above the specified range.
- GetMaximumRegion – Returns the region from the input array that corresponds to the largest computed feature value.
- GetMinimumRegion – Returns the region from the input array that corresponds to the smallest computed feature value.
- SortRegions – Changes the order of regions from the input array according to an ascending/descending sequence of their computed feature values.

GetMaximumRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Returns the region from the input array that corresponds to the largest computed feature value.

Applications: Use this filter when you have an array of regions and you want to select one of them that best matches some criterion.

Syntax

```
void avl::GetMaximumRegion
(
 const atl::Array<avl::Region>& inRegions,
 avl::RegionFeature::Type inFeature,
 avl::Region& outRegion,
 atl::Optional<float&> outValue = atl::NIL,
 atl::Optional<int&> outIndex = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inRegions	const Array<Region>&		Input regions
→	inFeature	RegionFeature::Type		Region feature value to be computed
←	outRegion	Region&		Output region
←	outValue	Optional<float&>	NIL	Computed feature value of the output region
←	outIndex	Optional<int&>	NIL	

Optional Outputs

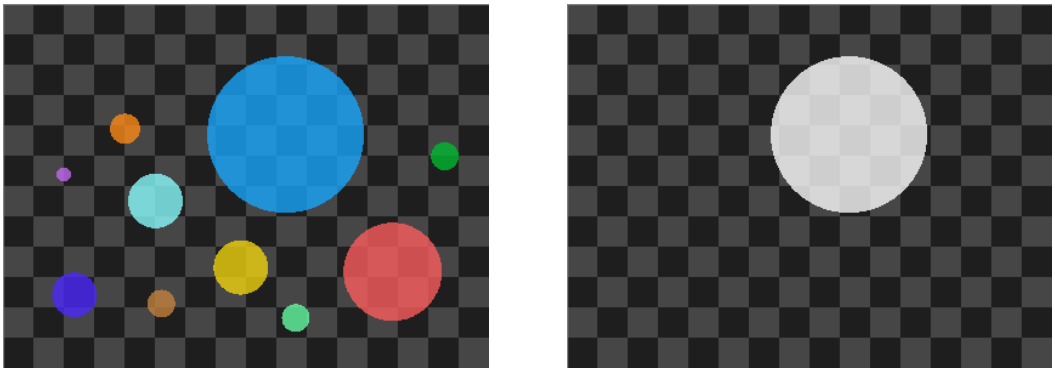
The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outValue**, **outIndex**.

Read more about Optional Outputs.

Description

The filter computes the selected feature value for every input region and returns the one for which value is the largest.

Examples



*GetMaximumRegion performed with **Area** as the selected region feature.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty region array on input in GetMaximumRegion.

See Also

- GetMinimumRegion – Returns the region from the input array that corresponds to the smallest computed feature value.
- SortRegions – Changes the order of regions from the input array according to an ascending/descending sequence of their computed feature values.
- ClassifyRegions – Splits an array of regions according to the selected feature and range.

GetMaximumRegion_OrNil

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Returns the region from the input array that corresponds to the largest computed feature value; returns NIL if the array or any region inside it is empty.

Applications: Use this filter when you have an array of regions and you want to select one of them that best matches some criterion.

Syntax

```
void avl::GetMaximumRegion_OrNil
(
 const atl::Array<avl::Region>& inRegions,
 avl::RegionFeature::Type inFeature,
 atl::Conditional<avl::Region>& outRegion,
 atl::Conditional<float>& outValue,
 atl::Conditional<int>& outIndex
)
```

Parameters

	Name	Type	Default	Description
→	inRegions	const Array<Region>&		Input regions
→	inFeature	RegionFeature::Type		Region feature value to be computed
←	outRegion	Conditional<Region>&		Output region
←	outValue	Conditional<float>&		Computed feature value of the output region
←	outIndex	Conditional<int>&		

GetMinimumRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Returns the region from the input array that corresponds to the smallest computed feature value.

Applications: Use this filter when you have an array of regions and you want to select one of them that best matches some criterion.

Syntax

```
void avl::GetMinimumRegion
(
 const atl::Array<avl::Region>& inRegions,
 avl::RegionFeature::Type inFeature,
 avl::Region& outRegion,
 atl::Optional<float&> outValue = atl::NIL,
 atl::Optional<int&> outIndex = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inRegions	const Array<Region>&		Input regions
→	inFeature	RegionFeature::Type		Region feature value to be computed
←	outRegion	Region&		Output region
←	outValue	Optional<float&>	NIL	Computed feature value of the output region
←	outIndex	Optional<int&>	NIL	

Optional Outputs

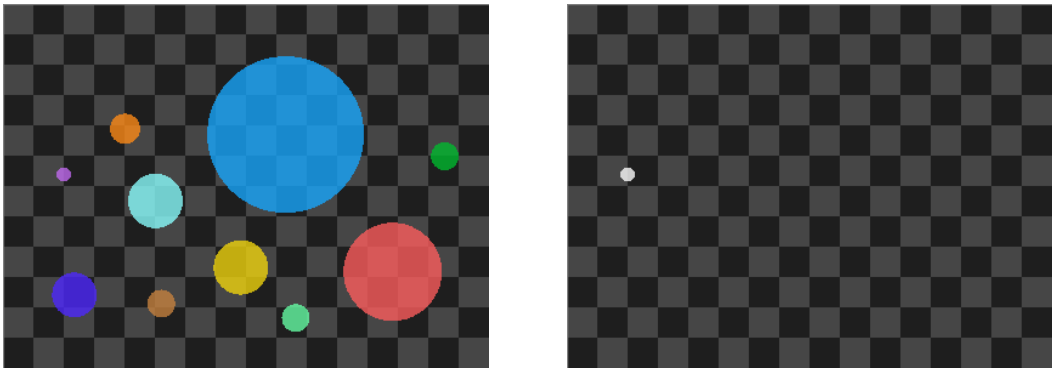
The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outValue**, **outIndex**.

Read more about Optional Outputs.

Description

The filter computes the selected feature value for every input region and returns the one for which value is the smallest.

Examples



*GetMinimumRegion performed with **Area** as the selected region feature.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty region array on input in GetMinimumRegion.

See Also

- GetMaximumRegion – Returns the region from the input array that corresponds to the largest computed feature value.
- SortRegions – Changes the order of regions from the input array according to an ascending/descending sequence of their computed feature values.
- ClassifyRegions – Splits an array of regions according to the selected feature and range.

GetMinimumRegion_OrNil

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Returns the region from the input array that corresponds to the smallest computed feature value; returns NIL if the array or any region inside it is empty.

Applications: Use this filter when you have an array of regions and you want to select one of them that best matches some criterion.

Syntax

```
void avl::GetMinimumRegion_OrNil
(
 const atl::Array<avl::Region>& inRegions,
 avl::RegionFeature::Type inFeature,
 atl::Conditional<avl::Region>& outRegion,
 atl::Conditional<float>& outValue,
 atl::Conditional<int>& outIndex
)
```

Parameters

	Name	Type	Default	Description
→	inRegions	const Array<Region>&		Input regions
→	inFeature	RegionFeature::Type		Region feature value to be computed
←	outRegion	Conditional<Region>&		Output region
←	outValue	Conditional<float>&		Computed feature value of the output region
←	outIndex	Conditional<int>&		

GroupPathsByRegions

Header: AVL.h
Namespace: avl
Module: FoundationBasic

For each region returns which paths lie inside of it or intersect with it.

Syntax

```
void avl::GroupPathsByRegions
(
 const atl::Array<avl::Path>& inPaths,
 const atl::Array<avl::Region>& inRegions,
 avl::MatchingCriterion::Type inMatchingCriterion,
 atl::Array<atl::Array<int> >& outGroupedIndices,
 atl::Optional<atl::Array<atl::Array<avl::Path> >&> outGroupedPaths = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPaths	const Array<Path>&		Array of paths that will be classified into multiple groups
→	inRegions	const Array<Region>&		Array of regions that define the groups
→	inMatchingCriterion	MatchingCriterion::Type		Specifies whether a path must fully belong to a group region or if it is enough that it intersects
←	outGroupedIndices	Array<Array<int> >&		Indices of input paths classified into multiple groups
←	outGroupedPaths	Optional<Array<Array<Path> >&>	NIL	Input paths classified into multiple groups

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outGroupedPaths**.

Read more about Optional Outputs.

See Also

- GroupPointsByRegions – For each region returns which points lie inside of it.
- AvsFilter_GetMultipleArrayElements – Extracts multiple elements from an array.

GroupPointsByRegions

Header: AVL.h
Namespace: avl
Module: FoundationBasic

For each region returns which points lie inside of it.

Syntax

```
void avl::GroupPointsByRegions
(
 const atl::Array<avl::Point2D>& inPoints,
 const atl::Array<avl::Region>& inRegions,
 atl::Array<atl::Array<int> >& outGroupedIndices,
 atl::Optional<atl::Array<atl::Array<avl::Point2D> >&> outGroupedPoints = atl::NIL
)
```

Parameters

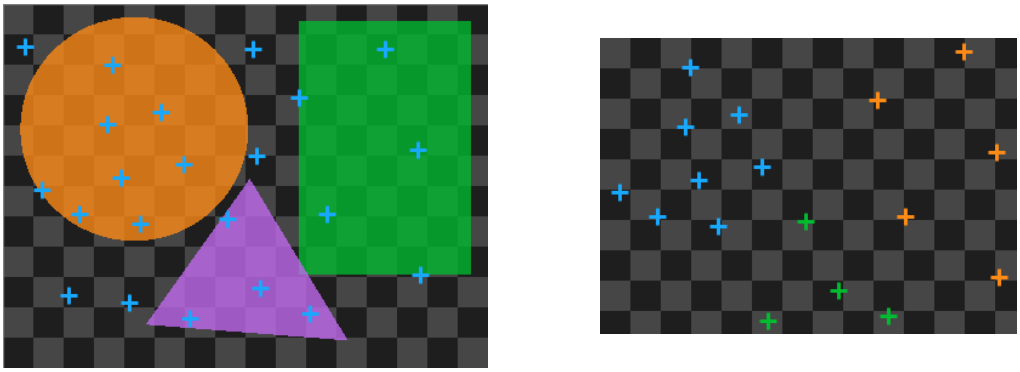
	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		
→	inRegions	const Array<Region>&		
←	outGroupedIndices	Array<Array<int> >&		
←	outGroupedPoints	Optional<Array<Array<Point2D> >&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outGroupedPoints**.

Read more about Optional Outputs.

Examples



GroupPointsByRegions performed with three input regions.

See Also

- AvsFilter_GetMultipleArrayElements – Extracts multiple elements from an array.

GroupRegionsByRegions

Header: AVL.h
Namespace: avl
Module: FoundationBasic

For each region returns which regions lie inside of it or intersect with it.

Syntax

```
void avl::GroupRegionsByRegions
(
 const atl::Array<avl::Region>& inRegions,
 const atl::Array<avl::Region>& inGroupRegions,
 avl::MatchingCriterion::Type inMatchingCriterion,
 atl::Array<atl::Array<int> >& outGroupedIndices,
 atl::Optional<atl::Array<atl::Array<avl::Region> >&> outGroupedRegions = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inRegions	const Array<Region>&		Array of regions that will be classified into multiple groups
→	inGroupRegions	const Array<Region>&		Array of regions that define the groups
→	inMatchingCriterion	MatchingCriterion::Type		Specifies whether a region must fully belong to a group region or if it is enough that it intersects
←	outGroupedIndices	Array<Array<int> >&		Indices of input regions classified into multiple groups
←	outGroupedRegions	Optional<Array<Array<Region> >&>	NIL	Input regions classified into multiple groups

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outGroupedRegions**.

Read more about Optional Outputs.

InscribeRegionInRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Shifts source region so that it is contained in target region.

Syntax

```
void avl::InscribeRegionInRegion
(
 const avl::Region& inTargetRegion,
 const avl::Region& inSourceRegion,
 avl::ShiftType::Type inPossibleShift,
 avl::FitType::Type inFitType,
 atl::Conditional<int>& outShiftX,
 atl::Conditional<int>& outShiftY,
 atl::Conditional<avl::Region>& outShiftedRegion
)
```

Parameters

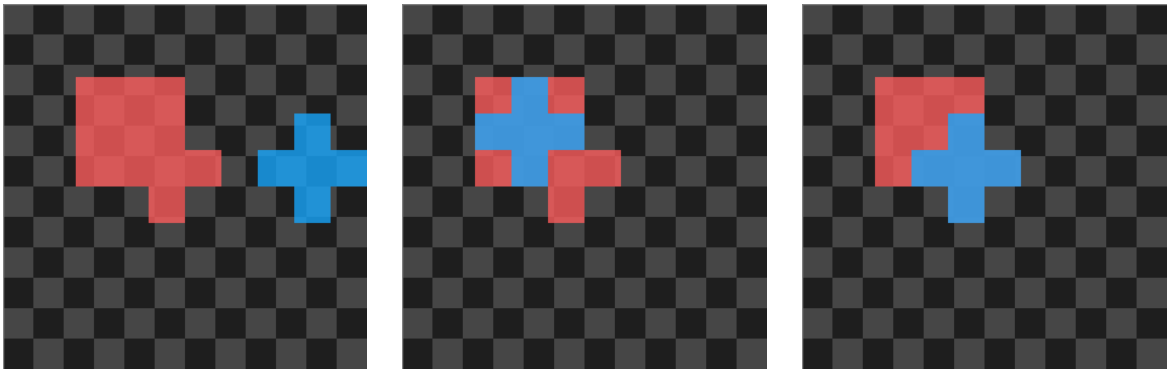
	Name	Type	Default	Description
→	inTargetRegion	const Region&		Region to contain source region
→	inSourceRegion	const Region&		Region to be shifted
→	inPossibleShift	ShiftType::Type		Possible directions of shift
→	inFitType	FitType::Type		Determines if the common border length should be maximized
←	outShiftX	Conditional<int>&		X coordinate of the shift
←	outShiftY	Conditional<int>&		Y coordinate of the shift
←	outShiftedRegion	Conditional<Region>&		Shifted source region

Description

The filter shifts **inSourceRegion** so that it is entirely contained in **inTargetRegion**. When **inFitType** is **FirstFit**, the function returns first shift. With **inFitType** set to **BestFit**, the function chooses from all possible shifts one that maximizes length of common border of target region and shifted source region. Parameter **inPossibleShift** defines directions in which source region should be shifted. Possible values are *Any, Vertical, Horizontal*.

All return values are conditional. In case appropriate shift exists, **outShiftX** and **outShiftY** define its coordinates. **outShiftedRegion** is **inSourceRegion** translated by vector (outShiftX,outShiftY) and with dimensions of **inTargetRegion**.

Examples



InscribeRegionInRegion *performed on pair of sample regions, shown on the left. Middle image shows inscribed region with* ***inFitType*** *set to FirstFit. Right one corresponds to* ***inFitType*** *equal BestFit.*

Remarks

Setting parameter **inFitType** to *BestFit* results in much higher computational cost.

Errors

List of possible exceptions:

Error type	Description
DomainError	Source region cannot be empty in InscribeRegionInRegion.

See Also

- ErodeRegion_AnyKernel – Performs a morphological erosion on a region using an arbitrary kernel.
- TranslateRegion – Translates a region by a given number of pixels along each axis.

SelectRegions

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Selects regions according to the selected feature and range.

Applications: Use this filter when you have an array of regions and you want to select some of them for further processing. It is slightly faster than ClassifyRegions.

Syntax

```
void avl::SelectRegions
(
 const atl::Array<avl::Region>& inRegions,
 avl::RegionFeature::Type inFeature,
 atl::Optional<float> inMinimum,
 atl::Optional<float> inMaximum,
 atl::Array<avl::Region>& outAccepted,
 atl::Optional<atl::Array<float>&> outValues = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inRegions	const Array<Region>&		Input regions
→	inFeature	RegionFeature::Type		Region feature value to be computed
→	inMinimum	Optional<float>	NIL	Lowest value of the range
→	inMaximum	Optional<float>	NIL	Highest value of the range
←	outAccepted	Array<Region>&		Regions with feature values matching the range
←	outValues	Optional<Array<float>&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outValues**.

Read more about Optional Outputs.

SortRegions

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Changes the order of regions from the input array according to an ascending/descending sequence of their computed feature values.

Syntax

```
void avl::SortRegions
(
 const atl::Array<avl::Region>& inRegions,
 avl::RegionFeature::Type inFeature,
 avl::SortingOrder::Type inSortingOrder,
 atl::Array<avl::Region>& outSortedRegions,
 atl::Optional<atl::Array<float>&> outSortedValues = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inRegions	const Array<Region>&		Input regions
→	inFeature	RegionFeature::Type		Region feature value to be computed
→	inSortingOrder	SortingOrder::Type		Sorting order
←	outSortedRegions	Array<Region>&		Regions sorted according to the computed feature values
←	outSortedValues	Optional<Array<float>&>	NIL	Computed feature values

Optional Outputs

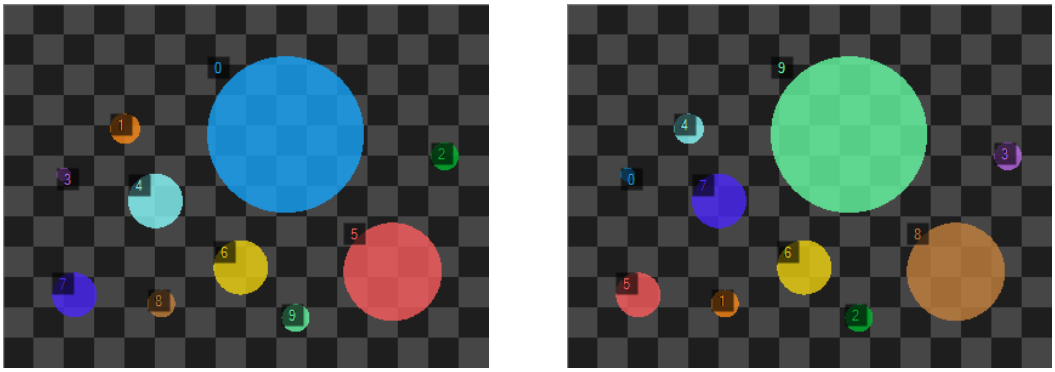
The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outSortedValues**.

Read more about Optional Outputs.

Description

The filter sorts the array of input regions according to the selected feature values computed for each of them.

Examples



*SortRegions performed with **inFeature** = Area. Input regions are on the left. On the right is the result.*

See Also

- SortArray – Changes the order of the input array elements according to an ascending/descending sequence of the value array.
- GetMaximumRegion – Returns the region from the input array that corresponds to the largest computed feature value.
- GetMinimumRegion – Returns the region from the input array that corresponds to the smallest computed feature value.
- ClassifyRegions – Splits an array of regions according to the selected feature and range.

TestPointArrayInRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Tests which points lie inside a region.

Syntax

```
void avl::TestPointArrayInRegion
(
 const atl::Array<avl::Point2D>& inPoints,
 const avl::Region& inRegion,
 atl::Array<bool>& outIsContainedArray,
 atl::Optional<atl::Array<avl::Point2D>&> outPoints = atl::NIL,
 atl::Optional<bool&> outAreAllContained = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		Points which will be tested
→	inRegion	const Region&		Input region
←	outIsContainedArray	Array<bool>&		
←	outPoints	Optional<Array<Point2D>&>	NIL	Points that are contained
←	outAreAllContained	Optional<bool&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outPoints**, **outAreAllContained**.

Read more about Optional Outputs.

TestPointInRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Tests whether a point lies inside a region.

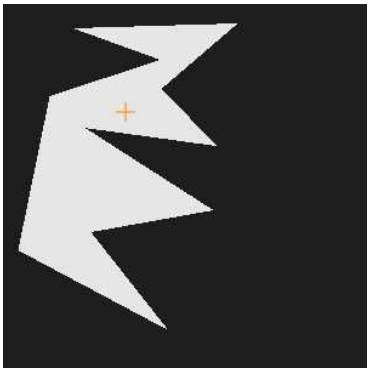
Syntax

```
void avl::TestPointInRegion
(
 const avl::Point2D& inPoint,
 const avl::Region& inRegion,
 bool& outIsContained
)
```

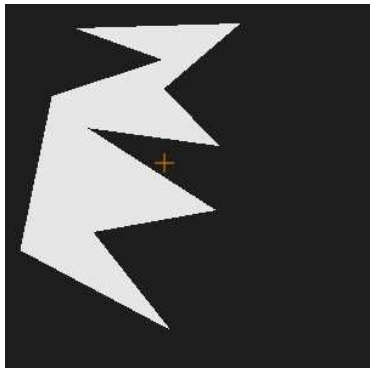
Parameters

	Name	Type	Default	Description
→	inPoint	const Point2D&		
→	inRegion	const Region&		Input region
←	outIsContained	bool&		

Examples



TestPointInRegion *run on the sample data produces the* ***outIsContained*** *= true*



TestPointInRegion *run on the sample data produces the* ***outIsContained*** *= false*

TestRegionEqualTo

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Tests whether given regions are equal.

Syntax

```
void avl::TestRegionEqualTo
(
 const avl::Region& inRegion,
 const avl::Region& inReferenceRegion,
 bool& outIsEqual
)
```

Parameters

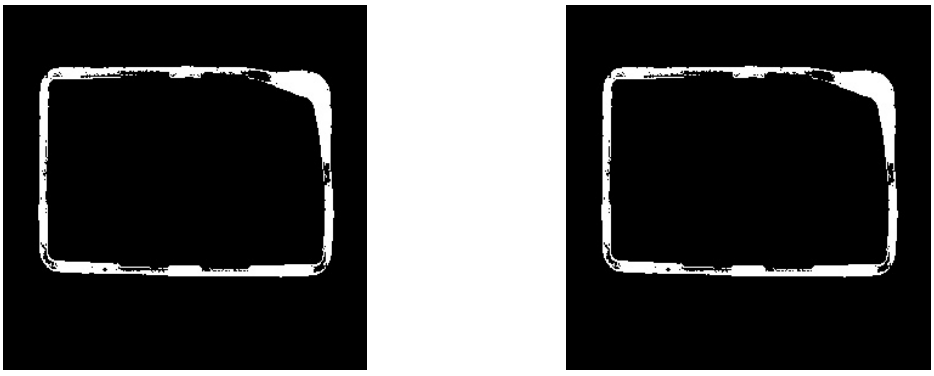
	Name	Type	Default	Description
➡	inRegion	const Region&		Input region
➡	inReferenceRegion	const Region&		
⬅	outIsEqual	bool&		

Description

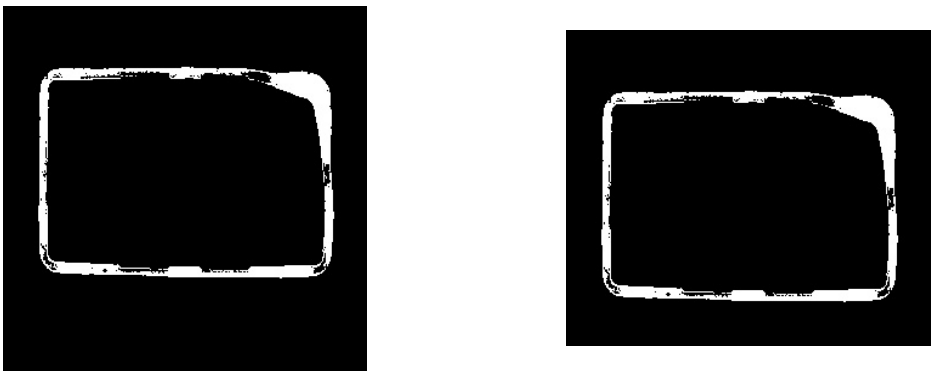
The operation tests whether the **inRegion** is the same region as contained inside **inReferenceRegion**. It means that the resulting **outIsEqual** is set to 'true' if and only if both of the conditions are met:

- **inRegion** and **inReferenceRegion** have the same dimensions
- **inRegion** and **inReferenceRegion** contain the same set of pixels

Examples



TestRegionEqualTo *run with* ***inRegion*** *set to the right region and* ***inReferenceRegion*** *set to the left, computes the* ***outIsEqual*** *= true.*



TestRegionEqualTo *run with* ***inRegion*** *set to the right region and* ***inReferenceRegion*** *set to the left, computes the* ***outIsEqual*** *= false, as dimensions of the regions differ.*

See Also

- TestRegionInRegion – Tests whether a region is contained in another one.

TestRegionInRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Tests whether a region is contained in another one.

Syntax

```
void avl::TestRegionInRegion
(
 const avl::Region& inSubregion,
 const avl::Region& inRegion,
 bool& outIsContained
)
```

Parameters

	Name	Type	Default	Description
→	inSubregion	const Region&		
→	inRegion	const Region&		Input region
←	outIsContained	bool&		

Description

The operation tests whether the **inSubregion** is contained inside the **inRegion**. The resulting **outIsContained** is set to 'true' if and only if each pixel of **inSubregion** is also included in **inRegion**.

Examples

TestRegionInRegion *run with* ***inSubregion*** *set to the left region and* ***inRegion*** *set to the right, computes the* ***outIsContained*** *= true.*

TestRegionInRegion *run with* ***inSubregion*** *set to the left region and* ***inRegion*** *set to the right, computes the* ***outIsContained*** *= false.*

See Also

- TestRegionEqualTo – Tests whether given regions are equal.

TestRegionIntersectsWith

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Tests whether two regions have non-empty intersection.

Syntax

```
void avl::TestRegionIntersectsWith
(
 const avl::Region& inRegion,
 const avl::Region& inReferenceRegion,
 bool& outRegionsIntersect
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inReferenceRegion	const Region&		
←	outRegionsIntersect	bool&		

TestRegionUnequalTo

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Tests whether given regions are not equal.

Syntax

```
void avl::TestRegionUnequalTo
(
 const avl::Region& inRegion,
 const avl::Region& inReferenceRegion,
 bool& outIsUnequal
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inReferenceRegion	const Region&		
←	outIsUnequal	bool&		

65. Clustering

Table of content:

ClusterData_KMeans

Header: AVL.h
Namespace: avl
Module: FoundationPro

Clusters data using KMeans algorithm.

Syntax

```
void avl::ClusterData_KMeans
(
 const atl::Array<atl::Array<float> >& inData,
 const int inClusters,
 const int inMaxIterations,
 const int inSeed,
 const float inTerminationFactor,
 const avl::KMeansClusteringMethod::Type inClusteringMethod,
 avl::Matrix& outCentroids,
 atl::Array<int>& outPointToClusterAssignment,
 float& outDistanceSum
)
```

Parameters

	Name	Type	Range	Default	Description
→	inData	const Array<Array<float> >&			Data set, array of examples
→	inClusters	const int	2 - +∞	2	Number of clusters to extract
→	inMaxIterations	const int	10 - 1000	200	Maximal number of procedure iterations
→	inSeed	const int	0 - ∞	5489	Seed to init random engine
→	inTerminationFactor	const float	1.0 - 2.0	1.5f	Additional factor of procedure stop
→	inClusteringMethod	const KMeansClusteringMethod::Type		KMeansPlusPlus	KMeans variant to use
←	outCentroids	Matrix&			Resulting centroid points in feature space
←	outPointToClusterAssignment	Array<int>&			Array of input point assignments to generated clusters
←	outDistanceSum	float&			Sum of squared distances from points to its respective cluster centroids

Errors

List of possible exceptions:

Error type	Description
DomainError	Cannot make more clusters than there is data in input dataset in ClusterData_KMeans.
DomainError	Empty dataset on input in ClusterData_KMeans.
DomainError	Inconsistent number of data coordinates in input dataset in ClusterData_KMeans.

ClusterPoints2D

Header: AVL.h
Namespace: avl
Module: FoundationPro

Clusters 2D points using K Means Clustering method.

Syntax

```
void avl::ClusterPoints2D
(
 const atl::Array<avl::Point2D>& inPoints,
 const int inClusters,
 const int inMaxIterations,
 atl::Optional<int> inSeed,
 const int inRunCount,
 atl::Array<atl::Conditional<atl::Array<avl::Point2D>>>& outClusters,
 atl::Array<atl::Conditional<avl::Point2D>>& outCentroids,
 float& outDistanceSum
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoints	const Array<Point2D>&			Array of points to cluster
→	inClusters	const int	2 - +∞	2	Number of clusters to extract
→	inMaxIterations	const int	10 - 1000	200	Maximal number of KMeans iterations
→	inSeed	Optional<int>	0 - +∞	5489	Seed used to initialize random number generators
→	inRunCount	const int	1 - +∞	1	Defines how many times the algorithm will be executed
←	outClusters	Array<Conditional<Array<Point2D>>>&			Resulting Point2D clusters
←	outCentroids	Array<Conditional<Point2D>>&			Center of found clusters
←	outDistanceSum	float&			Sum of distance squares from points in array to its respective cluster center

ClusterPoints2D_SingleLink

Header: AVL.h
Namespace: avl
Module: FoundationPro

Clusters data using hierarchical single-link algorithm.

Syntax

```
void avl::ClusterPoints2D_SingleLink
(
 const atl::Array<avl::Point2D>& inPoints,
 const atl::Optional<int> inClusters,
 const atl::Optional<float> inMaxDistance,
 atl::Array<atl::Array<avl::Point2D>>& outClusters
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoints	const Array<Point2D>&			Array of points to cluster
→	inClusters	const Optional<int>	2 - +∞	NIL	Number of clusters to extract
→	inMaxDistance	const Optional<float>	0.0 - ∞	NIL	Maximum distance between two closest points in a cluster
←	outClusters	Array<Array<Point2D>>&			Resulting Point2D clusters

Remarks

If input parameter **inClusters** is not set, number of clusters is determined in the following way:

- if **inMaxDistance** is set then every two points of mutual distance smaller or equal to **inMaxDistance** belong to one cluster;
- if **inMaxDistance** is not set then an array of sorted edges' lengths of minimum spanning tree of **inPoints** is determined and a pair of two subsequent elements which makes a maximum difference is found. Maximum distance is defined as the smaller element of such a pair. In other words number of clusters is the number of vertical lines in the dendrogram cut by a horizontal line that can transverse the maximum distance vertically without intersecting a cluster.

Errors

List of possible exceptions:

Error type	Description
DomainError	Desired number of clusters is greater than inPoints size in ClusterPoints2D_SingleLink.
DomainError	Filter input inPoints is empty in ClusterPoints2D_SingleLink.
DomainError	Too big inPoints array in ClusterPoints2D_SingleLink.

See Also

- ClusterPoints2D – Clusters 2D points using K Means Clustering method.

ClusterPoints3D

Header: AVL.h
Namespace: avl
Module: FoundationPro

Clusters 3D points using K Means Clustering method.

Syntax

```
void avl::ClusterPoints3D
(
 const atl::Array<avl::Point3D>& inPoints,
 const int inClusters,
 const int inMaxIterations,
 atl::Optional<int> inSeed,
 const int inRunCount,
 atl::Array<atl::Conditional<atl::Array<avl::Point3D>>>& outClusters,
 atl::Array<atl::Conditional<avl::Point3D>>& outCentroids,
 float& outDistanceSum
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoints	const Array<Point3D>&			Array of points to cluster
→	inClusters	const int	2 - +∞	2	Number of clusters to extract
→	inMaxIterations	const int	10 - 1000	200	Maximal number of KMeans iterations
→	inSeed	Optional<int>	0 - +∞	5489	Seed used to initialize random number generators
→	inRunCount	const int	1 - +∞	1	Defines how many times the algorithm will be executed
←	outClusters	Array<Conditional<Array<Point3D>>>&			Resulting Point3D clusters
←	outCentroids	Array<Conditional<Point3D>>&			Center of found clusters
←	outDistanceSum	float&			Sum of distance squares from points in array to its respective cluster center

FindConnectedComponents

Header: AVL.h
Namespace: avl
Module: FoundationPro

Finds connected components in a graph given as set of bidirectional connections.

Syntax

```
void avl::FindConnectedComponents
(
 const atl::Array<atl::Array<int> >& inConnections,
 atl::Array<atl::Array<int> >& outComponents,
 atl::Array<int>& outElementLabels,
 bool& outFound = atl::Dummy<bool>()
)
```

Parameters

	Name	Type	Default	Description
→	inConnections	const Array<Array<int> >&		List of connections for each element
←	outComponents	Array<Array<int> >&		List of input element indices for each connected component
←	outElementLabels	Array<int>&		Index of a component for each input element
←	outFound	bool&	Dummy<bool>()	

66. Image Metrics

Table of content:

ColorDistance

Header: AVL.h
Namespace: avl
Module: FoundationPro

Compares two pixels using chromatic and non-chromatic information. Assumes RGB color space.

Applications: Color comparison insensitive to changes of illumination.

Syntax

```
void avl::ColorDistance
(
 const avl::Pixel& inPixelA,
 const avl::Pixel& inPixelB,
 float inChromaAmount,
 float& outDistance
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPixelA	const Pixel&			
→	inPixelB	const Pixel&			
→	inChromaAmount	float	0.0 - 1.0	0.7f	Proportion of chromatic information in distance computation
←	outDistance	float&			

Description

The operation computes the distance between two pixels.

The distance between pixels is computed using two measures:

- Value distance - the difference between average channel values
- Chromatic distance - euclidean distance between pixels normalized to the same value level

The resulting distance is computed as a weighted average of these two values, parameter **inChromaAmount** (0.0 - 1.0) being the weight of the chromatic distance, and 1 - **inChromaAmount** being the weight of the value distance.

Errors

List of possible exceptions:

Error type	Description
RuntimeError	Each pixel component must be nonnegative in ColorDistance.

See Also

- ColorDistanceImage – Compares each pixel with the specified color using chromatic and non-chromatic information.

ColorDistanceImage

Header: AVL.h
Namespace: avl
Module: FoundationPro

Compares each pixel with the specified color using chromatic and non-chromatic information.

Applications: Color analysis insensitive to changes of illumination.

Syntax

```
void avl::ColorDistanceImage
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 const avl::Pixel& inRgbColor,
 float inChromaAmount,
 avl::Image& outValueImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Unsigned color image used in comparison
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inRgbColor	const Pixel&			Color to compare the image to
→	inChromaAmount	float	0.0 - 1.0	0.7f	Proportion of chromatic information in distance computation
←	outValueImage	Image&			Unsigned image of distances

Requirements

For input **inImage** only pixel formats are supported: 1□uint8, 2□uint8, 3□uint8, 4□uint8.

Read more about pixel formats in Image documentation.

Description

The operation computes the distance between each pixel of **inImage** and a specified color, presenting result as a monochromatic image. Input image should be in RGB color space.

The distance between pixels is computed using two measures:

- Value distance - the difference between average channel values
- Chromatic distance - euclidean distance between pixels normalized to the same value level

The resulting distance is computed as a weighted average of these two values, parameter **inChromaAmount** (0.0 - 1.0) being the weight of the chromatic distance, and 1 - **inChromaAmount** being the weight of the value distance.

Examples

ColorDistanceImage *run with* ***inRgbColor*** = , ***inChromaAmount*** = *0.8*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Invalid image Depth in ColorDistanceImage.
DomainError	Not supported inImage pixel format in ColorDistanceImage. Supported formats: 1xUInt8, 2xUInt8, 3xUInt8, 4xUInt8.
RuntimeError	Each pixel component of inRgbColor must be nonnegative in ColorDistanceImage.

See Also

- ColorDistance – Compares two pixels using chromatic and non-chromatic information. Assumes RGB color space.
- ThresholdToRegion_Color – Creates a region containing image pixels with values close to the given color.
- ThresholdImage_Color – Transforms each pixel value to maximum or minimum depending on the distance from a given color.

ColorDistance_CIE76

Header: AVL.h
Namespace: avl
Module: FoundationPro

Compares two pixels using CIE76 delta E. Assumes RGB color space.

Syntax

```
void avl::ColorDistance_CIE76
(
 const avl::Pixel& inPixelA,
 const avl::Pixel& inPixelB,
 float& outDistance
)
```

Parameters

	Name	Type	Default	Description
→	inPixelA	const Pixel&		
→	inPixelB	const Pixel&		
←	outDistance	float&		

Errors

List of possible exceptions:

Error type	Description
RuntimeError	Each pixel component must be nonnegative in ColorDistance_CIE76.

ColorDistance_CIE94

Header: AVL.h
Namespace: avl
Module: FoundationPro

Compares two pixels using CIE94 delta E. Assumes RGB color space.

Syntax

```
void avl::ColorDistance_CIE94
(
 const avl::Pixel& inPixelA,
 const avl::Pixel& inPixelB,
 const float inKL,
 const float inKC,
 const float inKH,
 const float inK1,
 const float inK2,
 float& outDistance
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPixelA	const Pixel&			
→	inPixelB	const Pixel&			
→	inKL	const float	0.0 - ∞	1.0f	The luminance weighting factor (best to use 1.0 for graphic arts and 2.0 for textiles)
→	inKC	const float	0.0 - ∞	1.0f	The chroma weighting factor (usually 1.0)
→	inKH	const float	0.0 - ∞	1.0f	The hue weighting factor (usually 1.0)
→	inK1	const float	0.0 - 1.0	0.045f	The secondary chroma weighting factor (best to use 0.045 for graphic arts and 0.048 for textiles)
→	inK2	const float	0.0 - 1.0	0.015f	The secondary luminance weighting factor (best to use 0.015 for graphic arts and 0.014 for textiles)
←	outDistance	float&			

Errors

List of possible exceptions:

Error type	Description
RuntimeError	Each pixel component must be nonnegative in ColorDistance_CIE94.

ColorDistance_CIEDE2000

Header: AVL.h
Namespace: avl
Module: FoundationPro

Compares two pixels using CIEDE2000 delta E. Assumes RGB color space.

Syntax

```
void avl::ColorDistance_CIEDE2000
(
 const avl::Pixel& inPixelA,
 const avl::Pixel& inPixelB,
 const float inKL,
 const float inKC,
 const float inKH,
 float& outDistance
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPixelA	const Pixel&			
→	inPixelB	const Pixel&			
→	inKL	const float	0.0 - ∞	1.0f	The luminance weighting factor
→	inKC	const float	0.0 - ∞	1.0f	The chroma weighting factor
→	inKH	const float	0.0 - ∞	1.0f	The hue weighting factor
←	outDistance	float&			

Errors

List of possible exceptions:

Error type	Description
RuntimeError	Each pixel component must be nonnegative in ColorDistance_CIEDE2000.

CompareImages

Header: AVL.h
Namespace: avl
Module: FoundationPro

Compares two images tile by tile using one of several available methods.

Syntax

```
void avl::CompareImages
(
 const avl::Image& inImage1,
 const avl::Image& inImage2,
 avl::CompareMeasure::Type inMeasure,
 int inTileSize,
 int inDensity,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage1	const Image&			First input image
→	inImage2	const Image&			Second input image
→	inMeasure	CompareMeasure::Type		DSSIM	
→	inTileSize	int	1 - ∞	25	
→	inDensity	int	1 - ∞		
←	outImage	Image&			Output image

Description

The operation compares two images tile by tile using one of the following methods:

- Value - Sum of absolute differences divided by tile area
- NCC - Normalized cross-correlation
- DSSIM - Structural dissimilarity

Errors

List of possible exceptions:

Error type	Description
DomainError	Formats of input images differ in CompareImages.
DomainError	Images sizes are not equal in CompareImages.
DomainError	Not supported measure type in CompareImages.

See Also

- ColorDistanceImage – Compares each pixel with the specified color using chromatic and non-chromatic information.

CompareImages_CVG

Header: AVL.h
Namespace: avl
Module: FoundationPro

Compares two images tile by tile using one of several available methods.

Syntax

```
void avl::CompareImages_CVG
(
 const avl::Image& inImage1,
 const avl::Image& inImage2,
 float inChromaAmount,
 float inValueAmount,
 float inGradientAmount,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage1	const Image&			First input image
→	inImage2	const Image&			Second input image
→	inChromaAmount	float	0.0 - ∞	0.6f	Proportion of chromatic information in output image
→	inValueAmount	float	0.0 - ∞	0.3f	Proportion of value information in output image
→	inGradientAmount	float	0.0 - ∞	0.1f	Proportion of gradient information in output image
←	outImage	Image&			Output image

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Formats of input images differ in CompareImages_CVG.
DomainError	Images sizes are not equal in CompareImages_CVG.

ImageCorrelation

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes the correlation of pattern and image.

Syntax

```
void avl::ImageCorrelation
(
 const avl::Image& inImage,
 const avl::Image& inPatternImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::CorrelationMeasure::Type inCorrelationMeasure,
 float& outCorrelation
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inPatternImage	const Image&		Pattern to be compared with input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inCorrelationMeasure	CorrelationMeasure::Type		Measure of correlation
←	outCorrelation	float&		Value of correlation of pattern and image

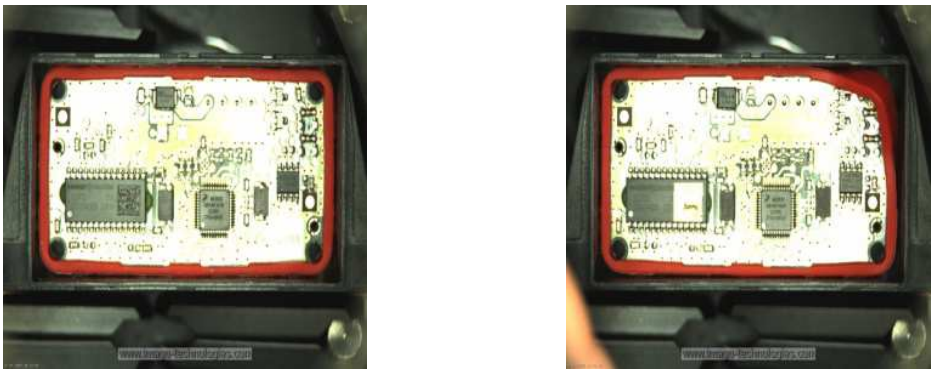
Description

The operation computes the correlation between two images using the selected measure.

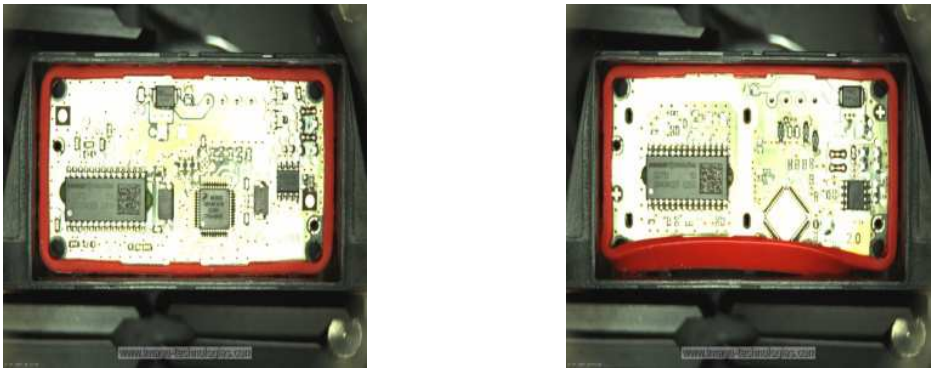
- If the **inCorrelationMeasure** is set to **CrossCorrelation** then the resulting **outCorrelation** is the sum of products of corresponding pixel values of the images.
- If the **inCorrelationMeasure** is set to **NormalizedCrossCorrelation** then the resulting **outCorrelation** is the normalized sum of products of corresponding pixel values of the images. This value always lies in closed interval $[-1,1]$, with 1 indicating the images ideal correlation.

The operation requires that the images being processed have equal format and dimensions, otherwise an error with appropriate description occurs. To obtain an image of desired dimensions one can use ResizeImage or CropImage filter. To alter the format of an image one can use filters contained in Image Conversions category.

Examples



Normalized cross correlation between the sample images equals 0.919.



Normalized cross correlation between the sample images equals 0.798.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported correlation measure in ImageCorrelation.

See Also

- ImageCorrelationImage – Computes an image of correlation between a pattern and the input image at each possible location.
- LocateMultipleObjects_NCC – Finds all occurrences of a predefined template on an image by analysing the normalized correlation between pixel values.

ImageCorrelationImage

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes an image of correlation between a pattern and the input image at each possible location.

Applications: This filter is used internally by the Gray-based Template Matching.

Syntax

```
void avl::ImageCorrelationImage
(
 const avl::Image& inImage,
 const avl::Image& inPatternImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<const avl::Region&> inPatternRoi,
 avl::CorrelationMeasure::Type inCorrelationMeasure,
 avl::Image& outImage
)
```

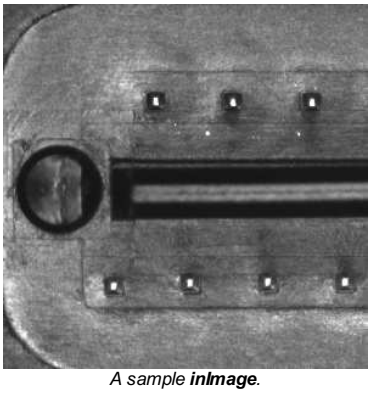
Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inPatternImage	const Image&		Pattern to be compared with input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inPatternRoi	Optional<const Region&>	NIL	Range of pixels in the pattern to be processed
→	inCorrelationMeasure	CorrelationMeasure::Type		Measure of correlation
←	outImage	Image&		Output image

Description

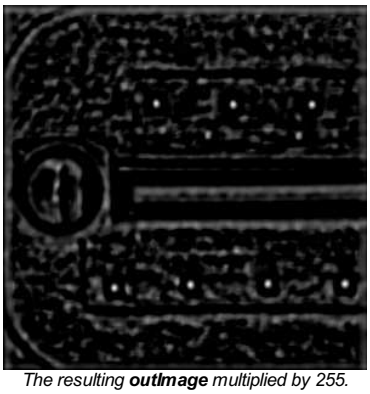
The operation computes the correlation between **inImage** and **inPatternImage**. The **inPatternImage** is aligned at each location of the **inImage** and the similarity between the **inPatternImage** and the corresponding part of the **inImage** is estimated using the **inCorrelationMeasure** as in ImageCorrelation filter. The result is stored in the pixel of the **outImage** that corresponds to the location at which the center of **inPatternImage** was aligned to the **inImage**.

Examples

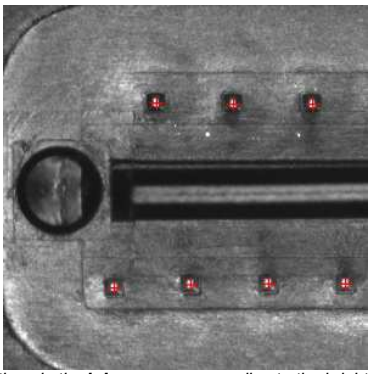


*A sample **inImage**.*

*A sample **inPatternImage**.*



*The resulting **outImage** multiplied by 255.*



*The locations in the **inImage** corresponding to the brightest points of the **outImage**.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

This operation supports processing on OpenCL compatible device (when inRoi=NIL, inPatternRoi=NIL and pixel type is uint8).

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported correlation measure in ImageCorrelationImage.

See Also

- ImageCorrelation – Computes the correlation of pattern and image.
- LocateMultipleObjects_NCC – Finds all occurrences of a predefined template on an image by analysing the normalized correlation between pixel values.

ImageDifference

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Measures dissimilarity between two images.

Syntax

```
void avl::ImageDifference
(
 const avl::Image& inImage,
 const avl::Image& inPatternImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::DistanceMeasure::Type inDistanceMeasure,
 float& outDifference
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inPatternImage	const Image&		Pattern to be compared with input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inDistanceMeasure	DistanceMeasure::Type		Measure of distance
←	outDifference	float&		Value of difference between pattern and image

Description

The operation computes the approximate difference between two images using the selected distance measure.

- If the **inDistanceMeasure** is set to **MeanError** then the resulting **outDifference** is the average difference between corresponding pixel values of the images.
- If the **inDistanceMeasure** is set to **MeanSquaredError** then the resulting **outDifference** is the average squared difference between corresponding pixel values of the images.

The operation requires that the images being processed have equal format and dimensions, otherwise an error with appropriate description occurs. To obtain an image of desired dimensions one can use ResizeImage or CropImage filter. To alter the format of an image one can use filters contained in Image Conversions category.

Examples


Mean error between the sample images equals 16.034.


Mean squared error between the sample images equals 2754.164.

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8 (for inDistanceMeasure = MeanError).

This operation is optimized for SSE3 technology for pixels of types: UINT16 (for inDistanceMeasure = MeanError), UINT8 (for inDistanceMeasure = MeanSquaredError), REAL.

This operation is optimized for SSE4 technology for pixels of types: INT16, UINT16 (for inDistanceMeasure = MeanSquaredError).

This operation is optimized for AVX2 technology for pixels of types: UINT8, INT16, UINT16, REAL.

This operation is optimized for NEON technology for pixels of types: UINT8, UINT16 (for inDistanceMeasure = MeanError), INT16 (for inDistanceMeasure = MeanError), REAL.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported distance measure in ImageDifference.

See Also

- ImageDifferenceImage – Computes an image of differences between a moving pattern and the input image.
- LocateMultipleObjects_SAD – Finds multiple occurrences of a predefined template on an image by analysing the Square Average Difference between pixel values.
- DifferenceImage – Computes the non-negative distances between corresponding pixel values.

ImageDifferenceImage

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes an image of differences between a moving pattern and the input image.

Syntax

```
void avl::ImageDifferenceImage
(
 const avl::Image& inImage,
 const avl::Image& inPatternImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<const avl::Region&> inPatternRoi,
 avl::DistanceMeasure::Type inDistanceMeasure,
 avl::Image& outImage
)
```

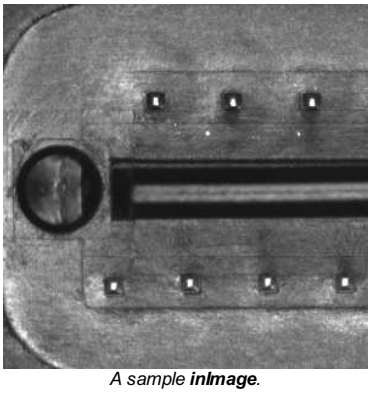
Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inPatternImage	const Image&		Pattern to be compared with input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inPatternRoi	Optional<const Region&>	NIL	Range of pixels in the pattern to be processed
→	inDistanceMeasure	DistanceMeasure::Type		Measure of distance
←	outImage	Image&		Output image

Description

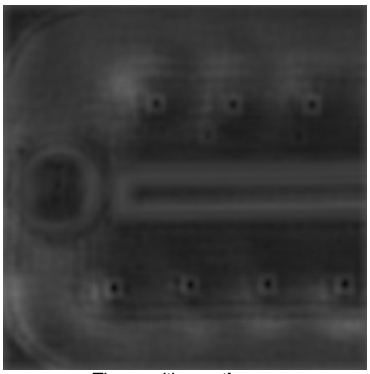
The operation computes the difference between **inImage** and **inPatternImage**. The **inPatternImage** is aligned at each location of the **inImage** and the similarity between the **inPatternImage** and the corresponding part of the **inImage** is estimated using the **inDifferenceMeasure** as in ImageDifference filter. The result is stored in the pixel of the **outImage** that corresponds to the location at which the center of **inPatternImage** was aligned to the **inImage**.

Examples

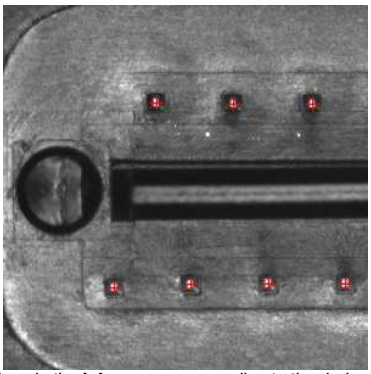


*A sample **inImage**.*

*A sample **inPatternImage**.*



*The resulting **outImage**.*



*The locations in the **inImage** corresponding to the darkest points of the **outImage**.*

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8 (for inDistanceMeasure = MeanError).

This operation is optimized for SSE3 technology for pixels of types: UINT16 (for inDistanceMeasure = MeanError), UINT8 (for inDistanceMeasure = MeanSquaredError), REAL.

This operation is optimized for SSE4 technology for pixels of types: INT16, UINT16 (for inDistanceMeasure = MeanSquaredError).

This operation is optimized for AVX2 technology for pixels of types: UINT8, INT16, UINT16, REAL.

This operation is optimized for NEON technology for pixels of types: UINT8, UINT16 (for inDistanceMeasure = MeanError), INT16 (for inDistanceMeasure = MeanError), REAL.

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported distance measure in ImageDifferenceImage.

See Also

- ImageDifference – Measures dissimilarity between two images.
- LocateMultipleObjects_SAD – Finds multiple occurrences of a predefined template on an image by analysing the Square Average Difference between pixel values.

67. Path Global Transforms

Table of content:

ConvertToEquidistantPath

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Creates a new path whose characteristic points lie on the input path, but are equally spaced.

Syntax

```
void avl::ConvertToEquidistantPath
(
 const avl::Path& inPath,
 float inStep,
 avl::EquidistanceType::Type inEquidistanceType,
 avl::Path& outPath
)
```

Parameters

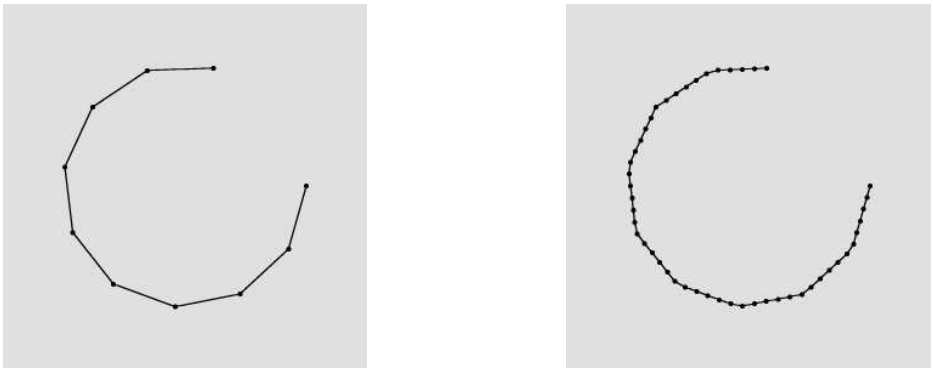
	Name	Type	Range	Default	Description
→	inPath	const Path&			Input path
→	inStep	float	0.0 - ∞	1.0f	Requested distance between consecutive points
→	inEquidistanceType	EquidistanceType::Type			Defines how the distance is measured
←	outPath	Path&			Output path

Description

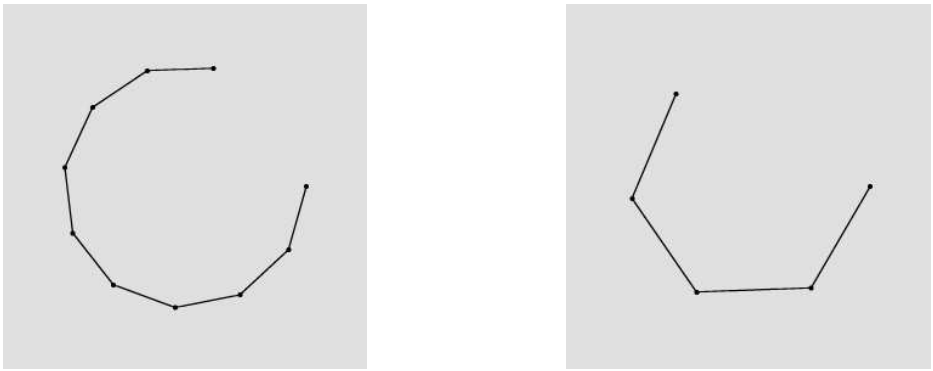
The operation follows a path from its beginning to the end, reselecting its characteristic points every **inStep** pixels. Note that this operation can significantly change the shape of a path, especially when the **inStep** value is relatively big. The output path's last point is mostly not the same as the input path's last point, because it would lead to last output path's segment being too short.

To reduce the number of points in a path preserving its shape, one can use ReducePath filter.

Examples



ConvertToEquidistantPath *run on the sample path with* ***inStep*** *= 10 and* ***inEquidistanceType*** *= OutputPathEquidistance.*



ConvertToEquidistantPath *run on the sample path with* ***inStep*** *= 100 and* ***inEquidistanceType*** *= OutputPathEquidistance.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Input step has to be positive in ConvertToEquidistantPath.

See Also

- ReducePath – Reduces the number of points in a path preserving its shape with the specified precision.

CreateBicircularCurve

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Creates a bicircular curve passing through the points of the leading path.

Applications: This is a fast algorithm that approximates Bezier's splines for dense paths.

Syntax

```
void avl::CreateBicircularCurve
(
 const avl::Path& inLeadingPath,
 int inInterpolationPointCount,
 avl::Path& outBicircularCurve
)
```

Parameters

	Name	Type	Default	Description
→	inLeadingPath	const Path&		The path that will have its corners rounded
→	inInterpolationPointCount	int	8	Number of points used for interpolation between each pair of points on the input path
←	outBicircularCurve	Path&		A smooth output path

ExtendPath

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Resizes the first or the last segment of a path.

Syntax

```
void avl::ExtendPath
(
 avl::Path& ioPath,
 float inExtraLength1,
 float inExtraLength2
)
```

Parameters

	Name	Type	Default	Description
↔	ioPath	Path&		
→	inExtraLength1	float		
→	inExtraLength2	float		

Errors

List of possible exceptions:

Error type	Description
DomainError	Cannot extend a closed path in ExtendPath.

FindLongestSubpath

Header: AVL.h
Namespace: avl
Module: FoundationPro

Creates a new path from the longest sequence of segments of the input path that turn gently.

Applications: Can be used for removing noise.

Syntax

```
void avl::FindLongestSubpath
(
 const avl::Path& inPath,
 float inMaxTurnAngle,
 avl::Path& outPath
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPath	const Path&			Input path, possibly noised
→	inMaxTurnAngle	float	0.0 - 180.0	30.0f	Maximum angle between two consecutive segments in the output path
←	outPath	Path&			Output path

ReducePath

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Reduces the number of points in a path preserving its shape with the specified precision.

Syntax

```
void avl::ReducePath
(
 const avl::Path& inPath,
 float inMaxDistance,
 avl::Path& outPath,
 atl::Array<avl::Path>& diagIntermediatePaths
)
```

Parameters

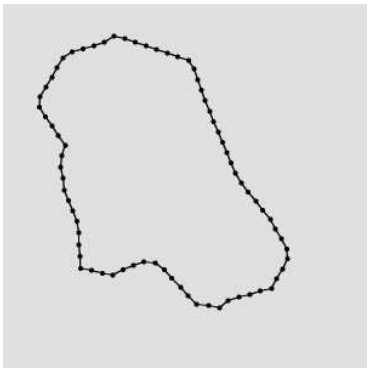
	Name	Type	Range	Default	Description
→	inPath	const Path&			Input path
→	inMaxDistance	float	0.0 - ∞	0.5f	Maximum distance between (possibly removed in the process) characteristic point of the input path and the output path.
←	outPath	Path&			Reduced path
🔍	diagIntermediatePaths	Array<Path>&			Intermediate results on all levels of recursion of the Ramer algorithm

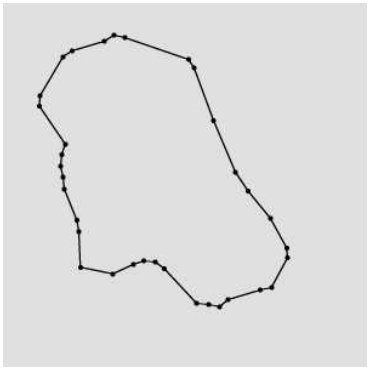
Description

The operation removes some of the characteristic points of the input path preserving its shape with the selected precision. The algorithm guarantees that each of the removed characteristic points lies within the **inMaxDistance** distance from the resulting path.

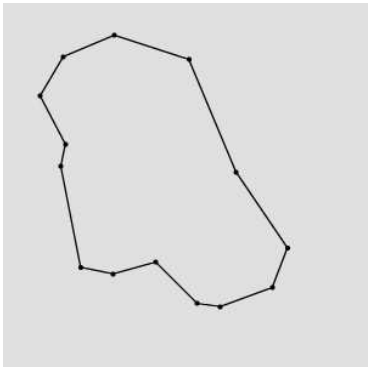
The operation works with open and closed paths as well.

Examples





ReducePath *run on the sample path array with* ***inMaxDistance*** *= 0.5.*



ReducePath *run on the sample path array with* ***inMaxDistance*** *= 5.*

See Also

- ConvertToEquidistantPath – Creates a new path whose characteristic points lie on the input path, but are equally spaced.

RemovePathSelfIntersections

Header: AVL.h
Namespace: avl
Module: FoundationPro

Removes all self-intersections from a given path.

Syntax

```
void avl::RemovePathSelfIntersections
(
 const avl::Path& inPath,
 avl::Path& outPath
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
←	outPath	Path&		Path with no self-intersections

SegmentPath

Header: AVL.h
Namespace: avl
Module: FoundationPro

Splits a path into parts that can be approximated as segments or arcs.

Applications: Usually used to recognize a particular structure of an object contour or to guide a CNC machine.

Syntax

```
void avl::SegmentPath
(
 const avl::Path& inPath,
 float inSmoothingStdDev,
 float inMaxDeviation,
 avl::PathSegmentationMode::Type inSegmentationMode,
 atl::Optional<float> inMaxArcRadius,
 atl::Array<avl::Path>& outStraight,
 atl::Array<avl::Path>& outArciform,
 atl::Array<avl::Segment2D>& outSegments,
 atl::Array<avl::Arc2D>& outArcs
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPath	const Path&			Path to be segmented
→	inSmoothingStdDev	float	0.0 - ∞		Standard deviation used for initial gaussian smoothing of the segmented path
→	inMaxDeviation	float	0.0 - ∞	0.5f	Maximal distance between any point of a classified segment to the abstract shape
→	inSegmentationMode	PathSegmentationMode::Type			Whether to use arcs for segmentation
→	inMaxArcRadius	Optional<float>	0.0 - ∞	10.0f	Maximal radius of an arc fitted to segment
←	outStraight	Array<Path>&			Parts classified as straight segments
←	outArciform	Array<Path>&			Parts classified as arciform segments
←	outSegments	Array<Segment2D>&			Segments corresponding to sections of path from outStraight
←	outArcs	Array<Arc2D>&			Arcs corresponding to sections of path from outArciform

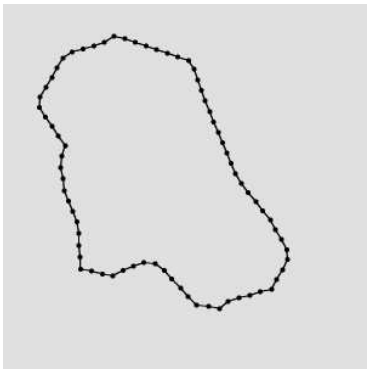
Description

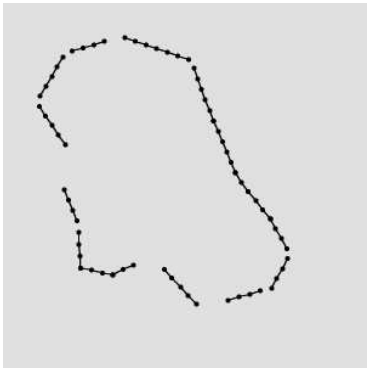
The operation segments the **inPath** into parts of preferably simple shape. Each of the resulting parts is classified as one of the following:

- **Straight section** - in which case it is returned in **outStraight** output array
- **Arciform section** - in which case it is returned in **outArciform** output array

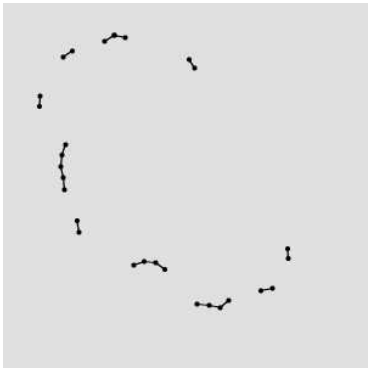
The operation guarantees that the maximal distance from any of the resulting path section to the corresponding abstract shape (line segment or circular arc) is less than **inMaxDeviation**.

Examples





*The resulting **outStraight** array (some of the paths in the picture have common ends but in fact each straight part is a separate path).*



*The resulting **outArciform** array.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty path on input in SegmentPath.

See Also

- JoinAdjacentPaths – Joins those paths of an array which endpoints lie near enough.

StretchPath

Header: AVL.h
Namespace: avl
Module: FoundationPro

Moves, rotates and scales the path so that its end-points end up at the given two points.

Syntax

```
void avl::StretchPath
(
 const avl::Path& inPath,
 const avl::Point2D& inNewEndpoint1,
 const avl::Point2D& inNewEndpoint2,
 avl::Path& outPath
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path, must be open
→	inNewEndpoint1	const Point2D&		New starting point of the path
→	inNewEndpoint2	const Point2D&		New ending point of the path
←	outPath	Path&		Output path

Errors

List of possible exceptions:

Error type	Description
DomainError	Path cannot be closed in StretchPath.
DomainError	Path too short in Stretch Path.

68. Profile Local Transforms

Table of content:

ConvolveProfile

Header: AVL.h
Namespace: avl
Module: FoundationPro

Applies a user-defined convolution to a profile.

Syntax

```
void avl::ConvolveProfile
(
 const avl::Profile& inProfile,
 atl::Optional<const avl::Range&> inRange,
 const atl::Array<float>& inMask,
 bool inNormalizeMaskValues,
 atl::Optional<int> inMaskOrigin,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Default	Description
→	inProfile	const Profile&		Profile to be processed
→	inRange	Optional<const Range&>	NIL	
→	inMask	const Array<float>&		Convolution kernel mask that will be applied to the profile
→	inNormalizeMaskValues	bool	False	If set to true, the kernel mask will be normalized so that its values sum up to one
→	inMaskOrigin	Optional<int>	NIL	Index of the kernel mask element that will be aligned against the profile values
←	outProfile	Profile&		Output profile

Description

The operation computes each value as a convolution of **inProfile** and **inMask** values. Values which mask exceeds profile dimension are set to zero.

Errors

List of possible exceptions:

Error type	Description
DomainError	Convolution mask is empty in ConvolveProfile.
DomainError	Mask origin exceeds dimensions of convolution mask in ConvolveProfile.
DomainError	Normalization of kernel mask which values add up to zero was requested in ConvolveProfile.
DomainError	Range exceeds the input profile in ConvolveProfile.

DifferentiateProfile

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes the derivative of a profile.

Syntax

```
void avl::DifferentiateProfile
(
 const avl::Profile& inProfile,
 const bool inCyclic,
 avl::DifferentiationMethod::Type inDifferentiationMethod,
 avl::Profile& outDerivative
)
```

Parameters

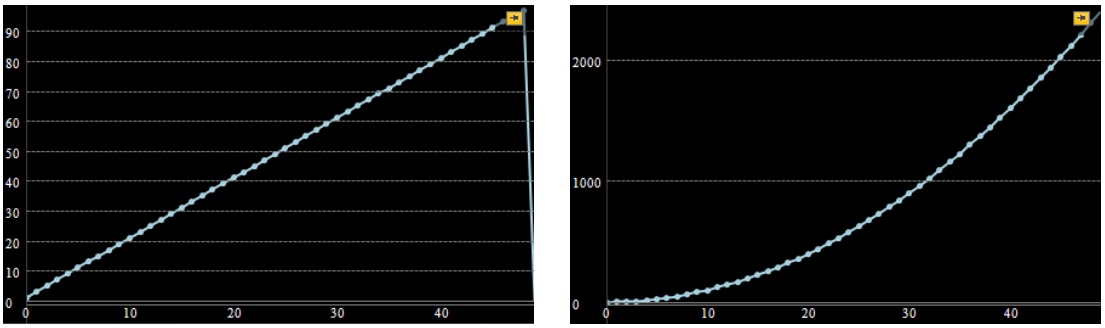
	Name	Type	Default	Description
→	inProfile	const Profile&		Input profile
→	inCyclic	const bool		Defines whether to compute differences between first and last elements
→	inDifferentiationMethod	DifferentiationMethod::Type	Central	
←	outDerivative	Profile&		

Description

Computes finite difference of profile using forward, backward or central difference method. If **inCyclic** is False then depending of chosen difference method values for first or last element of **inProfile** is undefined and thus outermost values are doubled when necessary.

For instance, in forward difference method, last element of derivative profile is always zero. In backward difference method, first element is always zero.

Examples



*On the left **DifferentiateProfile** with **inCyclic** = False, **inDifferentiationMethod** = Forward performed on profile being shown on the right.*

DifferentiateProfile_Step

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes the derivative of a profile with a given difference step.

Syntax

```
void avl::DifferentiateProfile_Step
(
 const avl::Profile& inProfile,
 const bool inCyclic,
 int inStep,
 avl::Profile& outDerivative
)
```

Parameters

	Name	Type	Range	Default	Description
→	inProfile	const Profile&			Input profile
→	inCyclic	const bool			Defines whether to compute differences between first and last elements
→	inStep	int	1 - ∞	2	
←	outDerivative	Profile&			

DilateProfile

Header: AVL.h
Namespace: avl
Module: FoundationPro

Changes a profile by choosing maximum point within a kernel.

Syntax

```
void avl::DilateProfile
(
 const avl::Profile& inProfile,
 atl::Optional<const avl::Range&> inRange,
 const int inKernelRadius,
 const bool inCyclic,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Range	Default	Description
→	inProfile	const Profile&			Profile to be processed
→	inRange	Optional<const Range&>		NIL	
→	inKernelRadius	const int	0 - ∞	3	Defines the width of the kernel as '2 * inKernelRadius + 1'
→	inCyclic	const bool			Defines whether the first element should be considered adjacent to the last element
←	outProfile	Profile&			Output profile

Errors

List of possible exceptions:

Error type	Description
DomainError	Range exceeds the input profile in DilateProfile.

ErodeProfile

Header: AVL.h
Namespace: avl
Module: FoundationPro

Changes a profile by choosing minimum point within a kernel.

Syntax

```
void avl::ErodeProfile
(
 const avl::Profile& inProfile,
 atl::Optional<const avl::Range&> inRange,
 const int inKernelRadius,
 const bool inCyclic,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Range	Default	Description
→	inProfile	const Profile&			Profile to be processed
→	inRange	Optional<const Range&>		NIL	
→	inKernelRadius	const int	0 - ∞	3	Defines the width of the kernel as '2 * inKernelRadius + 1'
→	inCyclic	const bool			Defines whether the first element should be considered adjacent to the last element
←	outProfile	Profile&			Output profile

Errors

List of possible exceptions:

Error type	Description
DomainError	Range exceeds the input profile in ErodeProfile.

SmoothProfile_Gauss

Header: AVL.h
Namespace: avl
Module: FoundationPro

Smooths a profile by averaging points within a kernel using gaussian-weighted average.

Applications: Noise removal.

Syntax

```
void avl::SmoothProfile_Gauss
(
 const avl::Profile& inProfile,
 atl::Optional<const avl::Range&> inRange,
 const float inStdDev,
 const float inKernelRelativeSize,
 const bool inCyclic,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Range	Default	Description
→	inProfile	const Profile&			Input profile
→	inRange	Optional<const Range&>		NIL	
→	inStdDev	const float	0.0 - ∞	0.6f	Standard deviation of the gaussian kernel
→	inKernelRelativeSize	const float	0.0 - ∞	3.0f	A multiple of the standard deviation determining the size of the kernel
→	inCyclic	const bool			Defines whether the first element should be considered adjacent to the last element
←	outProfile	Profile&			Output profile

Errors

List of possible exceptions:

Error type	Description
DomainError	Range exceeds the input profile in SmoothProfile_Gauss.

SmoothProfile_Gauss_Mask

Header: AVL.h
Namespace: avl
Module: FoundationPro

Smooths a profile by averaging points with one of ten pre-computed Gauss kernels.

Applications: Noise removal. Faster, but less accurate.

Syntax

```
void avl::SmoothProfile_Gauss_Mask
(
 const avl::Profile& inProfile,
 float inStdDev,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Range	Default	Description
→	inProfile	const Profile&			Input profile
→	inStdDev	float	0.0 - 3.0	0.6f	
←	outProfile	Profile&			Output profile

Hardware Acceleration

This operation is optimized for SSE2 technology.

This operation is optimized for AVX2 technology.

This operation is optimized for NEON technology.

SmoothProfile_Mean

Header: AVL.h
Namespace: avl
Module: FoundationPro

Smooths a profile by averaging points within a kernel.

Syntax

```
void avl::SmoothProfile_Mean
(
 const avl::Profile& inProfile,
 atl::Optional<const avl::Range&> inRange,
 const int inKernelRadius,
 const bool inCyclic,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Range	Default	Description
→	inProfile	const Profile&			Input profile
→	inRange	Optional<const Range&>		NIL	
→	inKernelRadius	const int	0 - ∞	3	Defines the width of the kernel as '2 * inKernelRadius + 1'
→	inCyclic	const bool			Defines whether the first element should be considered adjacent to the last element
←	outProfile	Profile&			Output profile

Errors

List of possible exceptions:

Error type	Description
DomainError	Range exceeds the input profile in SmoothProfile_Mean.

69. Image Spatial Transforms

Table of content:

CreateAffineTransformMatrix

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Creates affine transform matrix matrix. Return product of matrices: Translation * Rotation * Scale * Shearing.

Syntax

```
void avl::CreateAffineTransformMatrix
(
 const float inScaleX,
 const float inScaleY,
 const avl::Vector2D& inTranslation,
 const float inRotation,
 const float inShearingX,
 const float inShearingY,
 avl::Matrix& outTransformMatrix
)
```

Parameters

	Name	Type	Range	Default	Description
→	inScaleX	const float	-∞ - ∞	1.0f	Image scale in X axis
→	inScaleY	const float	-∞ - ∞	1.0f	Image scale in Y axis
→	inTranslation	const Vector2D&			Image translation
→	inRotation	const float		0.0f	Image rotation in degree
→	inShearingX	const float	-∞ - ∞		Shearing in X axis coefficient
→	inShearingY	const float	-∞ - ∞		Shearing in Y axis coefficient
←	outTransformMatrix	Matrix&			Return product of matrices: Translation * Rotation * Scale * Shearing

CreateImagePyramid

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Creates an array of images, each downsampled from the previous one.

Applications: Frequently used to create scale-invariant image analysis algorithms.

Syntax

```
void avl::CreateImagePyramid
(
 const avl::Image& inImage,
 int inMinPyramidLevel,
 int inMaxPyramidLevel,
 avl::DownsampleFunction::Type inFunction,
 atl::Array<avl::Image>& outImagePyramid
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inMinPyramidLevel	int	0 - 12		
→	inMaxPyramidLevel	int	0 - 12		
→	inFunction	DownsampleFunction::Type		Mean	
←	outImagePyramid	Array<Image>&			

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of type: 1xUINT8.

This operation is optimized for SSSE3 technology for pixels of type: 3xUINT8.

CreateImagePyramid_Gauss

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Creates an array of images, each downsampled from the previous, gauss-smoothed one.

Applications: Frequently used to create scale-invariant image analysis algorithms. Reduces impact of pixel aliasing.

Syntax

```
void avl::CreateImagePyramid_Gauss
(
 const avl::Image& inImage,
 avl::GaussKernel::Type inGaussKernelSize,
 int inMinPyramidLevel,
 int inMaxPyramidLevel,
 avl::DownsampleFunction::Type inFunction,
 atl::Array<avl::Image>& outImagePyramid
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inGaussKernelSize	GaussKernel::Type			
→	inMinPyramidLevel	int	0 - 12		
→	inMaxPyramidLevel	int	0 - 12		
→	inFunction	DownsampleFunction::Type		Mean	
←	outImagePyramid	Array<Image>&			

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of type: 1xUINT8.

This operation is optimized for SSSE3 technology for pixels of type: 3xUINT8.

CropImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates an image from a box-shaped fragment of the input image (with margins if requested).

Applications: Reduction of the amount of image data to be stored in memory.

Syntax

```
void avl::CropImage
(
 const avl::Image& inImage,
 const avl::Box& inSelection,
 const avl::Pixel& inBorderColor,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inSelection	const Box&		Box defining a subimage to be cropped
→	inBorderColor	const Pixel&		Color used for locations outside the selection
←	outImage	Image&		Output image

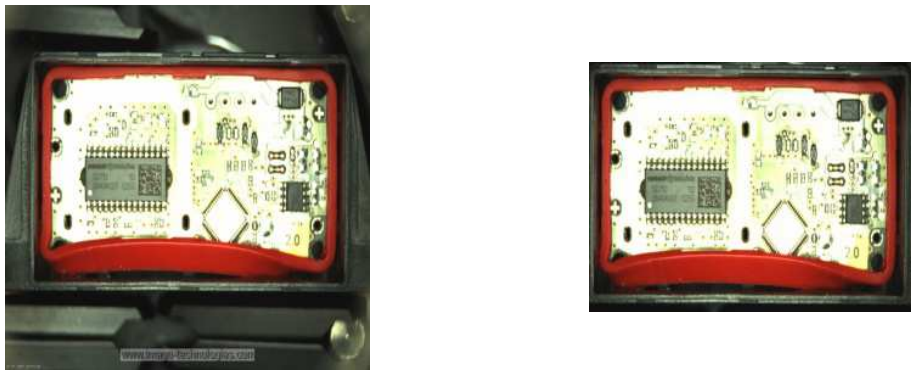
Description

The operation extracts part of the **inImage** that corresponds to the **inSelection** box.

Hints

- It is usually recommended to use Local Coordinate Systems rather than image cropping.

Examples



***CropImage** performed on the sample image.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- UncropImage – Inverse of CropImage.

CropImageToQuadrangle

Header: AVL.h
Namespace: avl
Module: FoundationPro

Creates an image from a quadrangle on another image.

Applications: Can be used for cropping an image from a slightly distorted rectangle. Do not use it for perspective distortion though.

Syntax

```
void avl::CropImageToQuadrangle
(
 const avl::Image& inImage,
 const avl::Path& inQuadrangle,
 atl::Optional<const avl::CoordinateSystem2D&> inQuadrangleAlignment,
 atl::Optional<const avl::Size&> inOutputSize,
 avl::InterpolationMethod::Type inInterpolationMethod,
 int inMargin,
 const avl::Pixel& inBorderColor,
 avl::Image& outImage,
 atl::Optional<avl::Path&> outAlignedQuadrangle = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inQuadrangle	const Path&			Rectangle defining a rotated subimage
→	inQuadrangleAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the rectangle to the position of the inspected object
→	inOutputSize	Optional<const Size&>		NIL	Dimensions of the output image
→	inInterpolationMethod	InterpolationMethod::Type		Bilinear	
→	inMargin	int	0 - 65535		Width of an additional margin for the output image
→	inBorderColor	const Pixel&			Color used for locations outside the quadrangle
←	outImage	Image&			Output image
←	outAlignedQuadrangle	Optional<Path&>		NIL	Input quadrangle after transformation (in the image coordinates)

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedQuadrangle**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Quadrangle should be specified by a closed path with exactly 4 points in CropImageToQuadrangle.

CropImageToRectangle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates an image from a rectangular fragment of another image (with black margins if requested).

Applications: Usually used for creating images of individual objects, e.g. after Template Matching.

Syntax

```
void avl::CropImageToRectangle
(
 const avl::Image& inImage,
 const avl::Rectangle2D& inRectangle,
 atl::Optional<const avl::CoordinateSystem2D&> inRectangleAlignment,
 avl::CropScaleMode::Type inScaleMode,
 avl::InterpolationMethod::Type inInterpolationMethod,
 float inMargin,
 const avl::Pixel& inBorderColor,
 avl::Image& outImage,
 atl::Optional<avl::Rectangle2D&> outAlignedRectangle = atl::NIL,
 atl::Optional<avl::CoordinateSystem2D&> outOutputAlignment = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRectangle	const Rectangle2D&			Rectangle defining a rotated subimage
→	inRectangleAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the rectangle to the position of the inspected object
→	inScaleMode	CropScaleMode::Type		InputScale	InputScale keeps the input scale unchanged, AlignedScale rescales according to the input alignment scale
→	inInterpolationMethod	InterpolationMethod::Type		Bilinear	
→	inMargin	float	0.0 - ∞		Width of an additional margin for the output image
→	inBorderColor	const Pixel&			Color used for locations outside the rectangle
←	outImage	Image&			Output image
←	outAlignedRectangle	Optional<Rectangle2D&>		NIL	Input rectangle after transformation (in the image coordinates)
←	outOutputAlignment	Optional<CoordinateSystem2D&>		NIL	Alignment of the output image

Optional Outputs

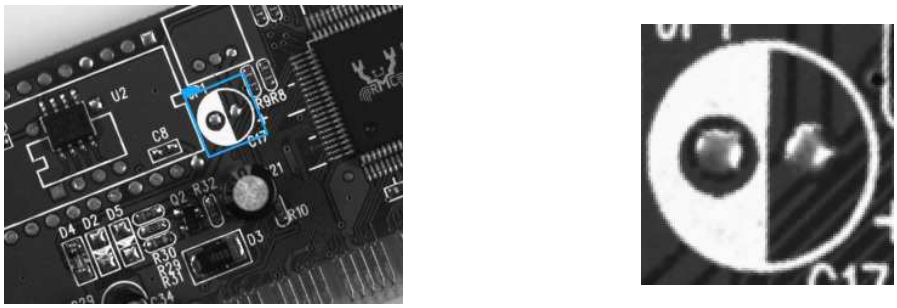
The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedRectangle**, **outOutputAlignment**.

Read more about Optional Outputs.

Hints

- It is usually recommended to use Local Coordinate Systems rather than image cropping.
- To obtain output images with constant dimensions regardless of **inRectangleAlignment** set **inScaleMode** to **AlignedScale**. The output image size will be equal to **inRectangle** size optionally modified by **inMargin**.

Examples



CropImageToRectangle *used to extract an image of an object.*

DownsampleImage

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Shrinks an image by the factor of two along each axis.

Applications: Usually used to speed-up image analysis at the cost of reduced precision. It is internally used to implement the pyramid strategy in template matching filters.

Syntax

```
void avl::DownsampleImage
(
 const avl::Image& inImage,
 const int inScaleStep,
 avl::DownsampleFunction::Type inFunction,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inScaleStep	const int	0 - 12	1	Defines how many times the image size is divided by 2
→	inFunction	DownsampleFunction::Type		Mean	
←	outImage	Image&			Output image

Description

The operation shrinks the **inImage** reducing its dimensions by a factor of two **inScaleStep** times.

Remarks

The operation produces a new image by averaging four neighboring pixels from the source image repeatedly. New width and height of image are always rounded up, thus pixels from source image at last row or column are averaged in pairs or copied when corresponding dimension is odd.

This operation can be used as faster replacement for ShrinkImageNTimes.

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT8, UINT16, SINT16, SINT32, REAL.

This operation is optimized for AVX2 technology for pixels of types: UINT8, SINT8, UINT16, SINT16, SINT32, REAL.

This operation is optimized for NEON technology for pixels of types: UINT8, SINT8, UINT16, SINT16, SINT32, REAL.

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- ShrinkImageNTimes – Shrinks an image by a natural factor along each axis.
- ResizeImage_Relative – Resizes an image by a factor along each axis.

DownsampleImage_Midlevels

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Shrinks an image by the factor of 1.5 or SQRT(2) along each axis.

Applications: Usually used to speed-up image analysis at the cost of reduced precision. It is internally used to implement the pyramid strategy.

Syntax

```
void avl::DownsampleImage_Midlevels
(
 const avl::Image& inImage,
 const int inScaleStep,
 avl::MidlevelScale::Type inMidlevelScale,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inScaleStep	const int	0 - 12	1	Defines how many times the image size is divided by 1.5 or SQRT(2)
→	inMidlevelScale	MidlevelScale::Type			Determines which scale will be used
←	outImage	Image&			Output image

Hardware Acceleration

This operation is optimized for SSE2 technology for pixels of type: 1xUINT8.

This operation is optimized for SSSE3 technology for pixels of type: 3xUINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

ImageAlongArc

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Creates an image from pixels traversed along an arc.

Applications: E.g. "Unwrapping" of object contours, so that they appear as 1D structures on the output image.

Syntax

```
void avl::ImageAlongArc
(
 const avl::Image& inImage,
 const avl::Arc2D& inAxisArc,
 atl::Optional<const avl::CoordinateSystem2D&> inAxisArcAlignment,
 int inScanWidth,
 avl::Axis::Type inAxisType,
 avl::InterpolationMethod::Type inInterpolationMethod,
 atl::Optional<const avl::Pixel&> inBorderColor,
 avl::Image& outImage,
 atl::Optional<avl::Arc2D&> outAlignedAxisArc = atl::NIL,
 atl::Array<avl::Path>& diagSamplingPoints = atl::Dummy<atl::Array<avl::Path> >()
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inAxisArc	const Arc2D&			Input arc
→	inAxisArcAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the axis arc to the position of the inspected object
→	inScanWidth	int	1 - ∞	5	The width of the stripe of pixels along the given arc
→	inAxisType	Axis::Type		Y	Type of axis the transformed axis arc will be parallel to
→	inInterpolationMethod	InterpolationMethod::Type		Bilinear	The interpolation method used to compute pixel brightness in locations of not-integer coordinates
→	inBorderColor	Optional<const Pixel&>		NIL	Color of pixel outside image. If inBorderColor = NIL then algorithm repeats color of boarder.
←	outImage	Image&			Output image
←	outAlignedAxisArc	Optional<Arc2D&>		NIL	Input arc after transformation (in the image coordinates)
🔍	diagSamplingPoints	Array<Path>&			Array of paths each one containing the sampling points corresponding to one row of the resulting image

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedAxisArc**.

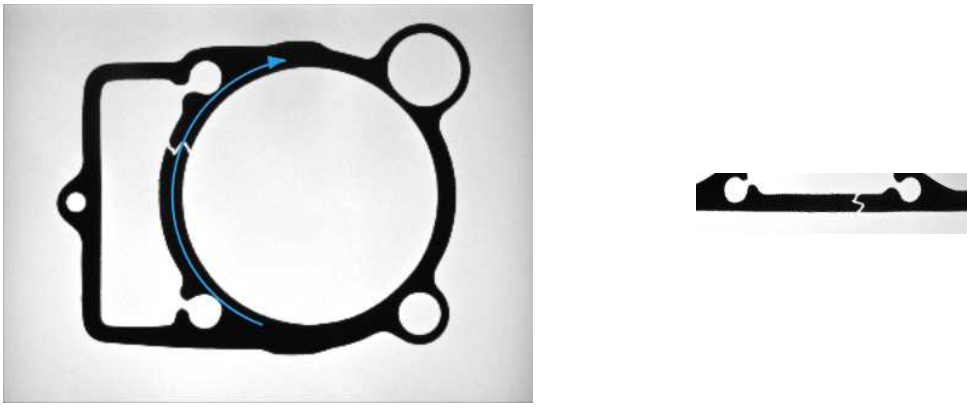
Read more about Optional Outputs.

Description

The operation transforms the stripe of pixels of width **inScanWidth** along the **inAxisArc** in the way that transforms the arc into straight segment. Two modes of pixel interpolation are available, the bilinear filtering being more precise and computationally expensive.

The optional parameter **inAxisArcAlignment** defines the transform to be performed on the **inAxisArc** so that the resulting path is defined in a new context, e.g. returned by one of Template Matching filters.

Examples



ImageAlongArc *performed on the sample image with* ***inScanWidth*** *= 50 and* ***inAxisType*** *= X.*

See Also

- ImageProfileAlongPath – Creates a series of segments across the input path, measures the average pixel intensity on each of the segments, and creates the final profile from those values.
- ImageAlongPath – Creates an image from pixels traversed along a path.

ImageAlongPath

Header: AVL.h
Namespace: avl
Module: FoundationPro

Creates an image from pixels traversed along a path.

Applications: E.g. "Unwrapping" of object contours, so that they appear as 1D structures on the output image.

Syntax

```
void avl::ImageAlongPath
(
 const avl::Image& inImage,
 const avl::Path& inAxisPath,
 atl::Optional<const avl::CoordinateSystem2D&> inAxisPathAlignment,
 int inScanWidth,
 avl::Axis::Type inAxisType,
 avl::InterpolationMethod::Type inInterpolationMethod,
 atl::Optional<const avl::Pixel&> inBorderColor,
 avl::Image& outImage,
 atl::Optional<avl::Path&> outAlignedAxisPath = atl::NIL,
 atl::Array<avl::Path>& diagSamplingPoints = atl::Dummy<atl::Array<avl::Path>>()
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inAxisPath	const Path&			Input path
→	inAxisPathAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the axis path to the position of the inspected object
→	inScanWidth	int	1 - ∞	5	The width of the stripe of pixels along the given path
→	inAxisType	Axis::Type		Y	Type of axis the transformed axis path will be parallel to
→	inInterpolationMethod	InterpolationMethod::Type		Bilinear	The interpolation method used to compute pixel brightness in locations of not-integer coordinates
→	inBorderColor	Optional<const Pixel&>		Pixel (X: 0.0f Y: 0.0f Z: 0.0f W: 0.0f)	Color of pixel outside image. If inBorderColor = NIL then algorithm repeats color of boarder.
←	outImage	Image&			Output image
←	outAlignedAxisPath	Optional<Path&>		NIL	Input path after transformation (in the image coordinates)
🔍	diagSamplingPoints	Array<Path>&			Array of paths each one containing the sampling points corresponding to one row of the resulting image

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedAxisPath**.

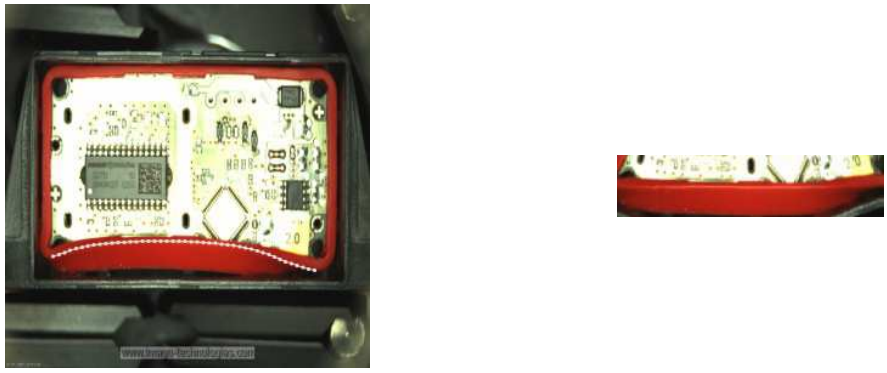
Read more about Optional Outputs.

Description

The operation transforms the stripe of pixels of width **inScanWidth** along the **inAxisPath** in the way that transforms the path into straight segment. Two modes of pixel interpolation are available, the bilinear filtering being more precise and computationally expensive.

The optional parameter **inAxisPathAlignment** defines the transform to be performed on the **inAxisPath** so that the resulting path is defined in a new context, e.g. returned by one of Template Matching filters.

Examples



ImageAlongPath *performed on the sample image with* ***inScanWidth*** *= 50 and* ***inAxisType*** *= Y. The result was transposed using TransposeImage for clarity.*

See Also

- ImageProfileAlongPath – Creates a series of segments across the input path, measures the average pixel intensity on each of the segments, and creates the final profile from those values.
- ImageAlongArc – Creates an image from pixels traversed along an arc.

ImageInversePolarTransform

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Transforms an image from polar or log-polar space to euclidean space.

Syntax

```
void avl::ImageInversePolarTransform
(
 const avl::Image& inImage,
 const avl::Point2D& inCenter,
 avl::PolarSpaceType::Type inInputSpaceType,
 avl::PolarInterpolationMethod::Type inInterpolation,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inCenter	const Point2D&		Center of the coordinate system in output image
→	inInputSpaceType	PolarSpaceType::Type		Method of transformation
→	inInterpolation	PolarInterpolationMethod::Type		Method of underlying interpolation
←	outImage	Image&		Output image

Requirements

For input **inImage** only pixel formats are supported: 1□uint8, 3□uint8, 1□int16, 3□int16, 1□uint16, 3□uint16, 1□real, 3□real.

Read more about pixel formats in Image documentation.

Description

Reverses ImagePolarTransform using the following transformation:

$$src(x,y) = dst(\rho,\phi)$$

where

$$\rho = m\sqrt{x^2+y^2}, \phi = \arctan(\frac{y}{x}),$$

for linear-polar space, and

$$\rho = k\log\sqrt{x^2+y^2}, \phi = \arctan(\frac{y}{x}),$$

for log-polar space, which emulates the human "foveal vision".

Parameters m and k are chosen so that entire source image could fit into output image.

Transformation does not change image dimensions.

Examples

*Original image, ImagePolarTransform and **ImageInversePolarTransform** performed on the sample image with **inCenter** = (150,150) for log-polar space type.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported inImage pixel format in ImageInversePolarTransform. Supported formats: 1xUInt8, 3xUInt8, 1xInt16, 3xInt16, 1xUInt16, 3xUInt16, 1xReal, 3xReal.

See Also

- ImagePolarTransform – Transforms an image to polar or log-polar space.

ImagePolarTransform

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Transforms an image to polar or log-polar space.

Syntax

```
void avl::ImagePolarTransform
(
 const avl::Image& inImage,
 const avl::Point2D& inCenter,
 avl::PolarSpaceType::Type inOutputSpaceType,
 avl::PolarInterpolationMethod::Type inInterpolation,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inCenter	const Point2D&		Center of the coordinate system in input image
→	inOutputSpaceType	PolarSpaceType::Type		Method of transformation
→	inInterpolation	PolarInterpolationMethod::Type		Method of underlying interpolation
←	outImage	Image&		Output image

Description

It uses the following transformation:

$$dst(\rho, \phi) = src(x, y),$$

where

$$\rho = m\sqrt{x^2 + y^2}, \phi = \arctan(\frac{y}{x}),$$

for linear-polar space, and

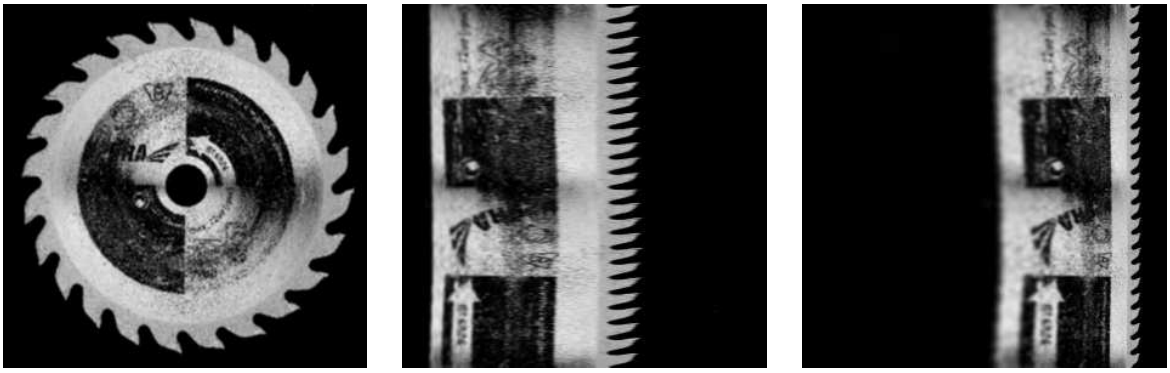
$$\rho = k \log \sqrt{x^2 + y^2}, \phi = \arctan(\frac{y}{x}),$$

for log-polar space, which emulates the human "foveal vision".

Parameters m and k are chosen so that entire source image could fit into output image.

Transformation does not change image dimensions.

Examples



ImagePolarTransform *performed on the sample image with* ***inCenter*** *= (150,150), and* ***inOutputSpaceType*** *Polar and LogPolar respectively.*

See Also

- ImageInversePolarTransform – Transforms an image from polar or log-polar space to euclidean space.

JoinImages

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a single image by glueing together the two input images in horizontal or vertical direction.

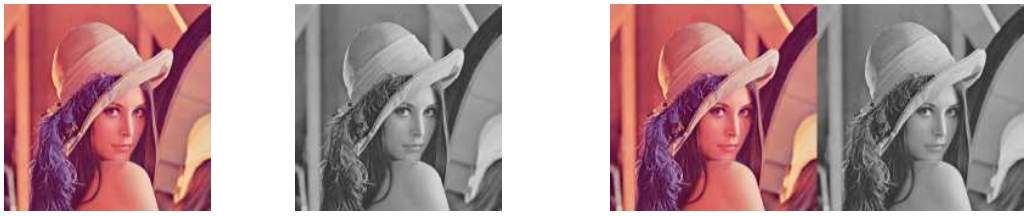
Syntax

```
void avl::JoinImages
(
 const avl::Image& inImage1,
 const avl::Image& inImage2,
 atl::Optional<const avl::Image&> inImage3,
 atl::Optional<const avl::Image&> inImage4,
 avl::JoinDirection::Type inDirection,
 const int inMargin,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage1	const Image&			First input image
→	inImage2	const Image&			Second input image
→	inImage3	Optional<const Image&>		NIL	Third input image
→	inImage4	Optional<const Image&>		NIL	Fourth input image
→	inDirection	JoinDirection::Type			Direction in which images are joined
→	inMargin	const int	0 - ∞	0	Thickness (in pixels) of black frame around joined images
←	outImage	Image&			Output image

Examples



JoinImages *used to join two images side by side into a single image.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Input images do not have compatible dimensions in JoinImages.
DomainError	Input images do not have the same pixel format in JoinImages.
DomainError	Unknown value of JoinDirection in JoinImages.

JoinImages_OfArray

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a single image by glueing together many input images in horizontal or vertical direction.

Syntax

```
void avl::JoinImages_OfArray
(
 const atl::Array<avl::Image>& inImages,
 avl::JoinDirection::Type inDirection,
 const int inMargin,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImages	const Array<Image>&			Array of input images
→	inDirection	JoinDirection::Type			Direction in which images are joined
→	inMargin	const int	0 - ∞	0	Thickness (in pixels) of black frame around joined images
←	outImage	Image&			Output image

Examples

JoinImages *used to join multiple image stripes into a single image.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Input images do not have compatible dimensions in JoinImages_OfArray.
DomainError	Input images do not have the same pixel format in JoinImages_OfArray.
DomainError	Unknown value of JoinDirection in JoinImages_OfArray.

JoinImages_OfSeries

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a single image by glueing together a series of input images in horizontal or vertical direction.

Applications: Joining images from a line-scan camera, which produces too few lines at a time.

Syntax

```
void avl::JoinImages_OfSeries
(
 JoinImages_OfSeriesState& ioState,
 const avl::Image& inImage,
 avl::JoinDirection::Type inDirection,
 const int inSeriesSize,
 const int inMargin,
 atl::Conditional<avl::Image>& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	JoinImages_OfSeriesState&			Object used to maintain state of the function.
	inImage	const Image&			Input image
	inDirection	JoinDirection::Type			Direction in which images are joined
	inSeriesSize	const int	1 - ∞		Number of images which are joined
	inMargin	const int	0 - ∞	0	Thickness (in pixels) of black frame around joined images
	outImage	Conditional<Image>&			Output image

Errors

List of possible exceptions:

Error type	Description
DomainError	Input images do not have compatible dimensions in JoinImages_OfSeries.
DomainError	Input images do not have the same pixel format in JoinImages_OfSeries.

MirrorImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Reverses the order of the input image columns or rows depending on inMirrorDirection value.

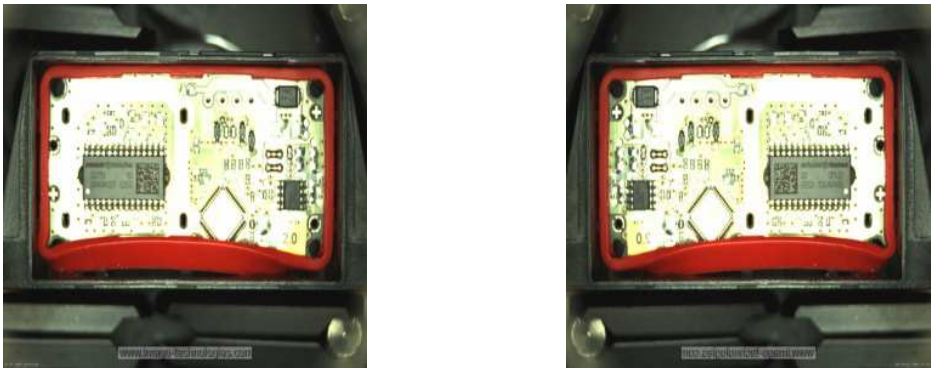
Syntax

```
void avl::MirrorImage
(
 const avl::Image& inImage,
 avl::MirrorDirection::Type inMirrorDirection,
 avl::Image& outImage
)
```

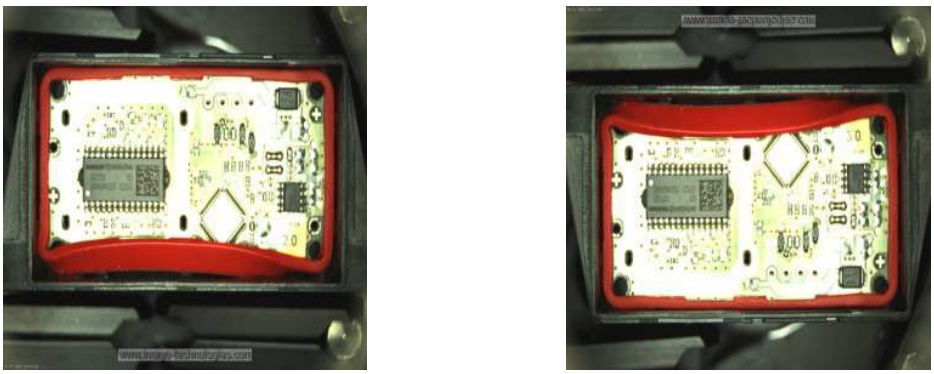
Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inMirrorDirection	MirrorDirection::Type	Vertical	Reverse the order of image columns (horizontal direction) or rows (vertical direction).
←	outImage	Image&		Output image

Examples



***MirrorImage** with **inMirrorDirection** = Horizontal.*



***MirrorImage** with **inMirrorDirection** = Vertical.*

Hardware Acceleration

This operation is optimized for SSSE3 technology for pixels of types: 1xUINT8, 1xINT8, 1xUINT16, 1xINT16, 1xINT32, 1xREAL, 2xUINT8, 2xINT8, 2xUINT16, 2xINT16, 3xUINT8, 3xINT8, 4xUINT8, 4xINT8.

This operation is optimized for NEON technology for pixels of type: 3xUINT8.

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- TransposeImage – Flips and rotates an image so that columns are exchanged with rows.

ResizeImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Enlarges or shrinks an image to new dimensions.

Syntax

```
void avl::ResizeImage
(
 const avl::Image& inImage,
 atl::Optional<const avl::Box&> inRoi,
 atl::Optional<int> inNewWidth,
 atl::Optional<int> inNewHeight,
 avl::ResizeMethod::Type inResizeMethod,
 avl::Image& outImage
)
```

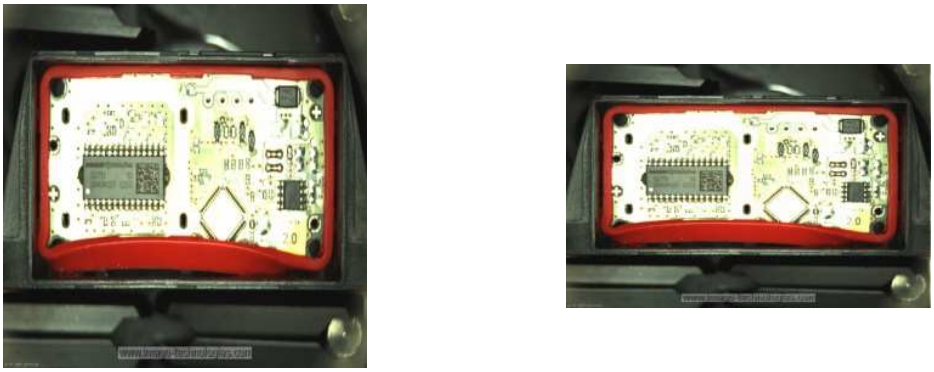
Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Box&>		NIL	Box defining a subimage to be cropped and resized.
→	inNewWidth	Optional<int>	1 - 65535	NIL	
→	inNewHeight	Optional<int>	1 - 65535	NIL	
→	inResizeMethod	ResizeMethod::Type			
←	outImage	Image&			Output image

Description

The operation stretches or shrinks the **inImage** so that the dimensions of the **outImage** equal **inNewWidth**, **inNewHeight**. Three modes of pixel interpolation are available, with Area mode giving best results, but being most computationally expensive.

Examples



ResizeImage *performed on the sample image with* ***inNewWidth*** *= 300,* ***inNewHeight*** *= 200.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

This operation is optimized for SSE2 technology for pixels of types: UINT8 (when inResizeMethod = Area), UINT16 (when inResizeMethod = Area), 3xUINT8 (when inResizeMethod = Area), 3xUINT16 (when inResizeMethod = Area).

This operation is optimized for SSE41 technology for pixels of types: UINT8(when inResizeMethod = Bilinear), 3xUINT8(when inResizeMethod = Bilinear).

This operation is optimized for AVX2 technology for pixels of types: UINT8 (when inResizeMethod = Area or inResizeMethod = Bilinear), UINT16 (when inResizeMethod = Area), 3xUINT8 (when inResizeMethod = Area or inResizeMethod = Bilinear), 3xUINT16 (when inResizeMethod = Area).

This operation is optimized for NEON technology for pixels of types: UINT8 (when inResizeMethod = Area), 3xUINT8 (when inResizeMethod = Area), UINT16 (when inResizeMethod = Area), 3xUINT16 (when inResizeMethod = Area).

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty image on input in ResizeImage.
DomainError	Input and output images are not distinct in ResizeImage.
DomainError	ROI is out of inImage range in ResizeImage.

See Also

- ResizeImage_Relative – Resizes an image by a factor along each axis.

ResizeImage_FixedAspectRatio

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Changes one dimension of the image to a desired length while the other dimension is set to a length such that the aspect ration of the image is maintained.

Syntax

```
void avl::ResizeImage_FixedAspectRatio
(
 const avl::Image& inImage,
 atl::Optional<int> inNewLength,
 avl::Dimension::Type inDimension,
 avl::ResizeMethod::Type inResizeMethod,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inNewLength	Optional<int>	1 - 65535	NIL	The length in pixels that the chosen dimension will be set to
→	inDimension	Dimension::Type			The dimension that will be set to the given length
→	inResizeMethod	ResizeMethod::Type			
←	outImage	Image&			Output image

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

This operation is optimized for SSE2 technology for pixels of types: UINT8 (when inResizeMethod = Area), UINT16 (when inResizeMethod = Area), 3xUINT8 (when inResizeMethod = Area), 3xUINT16 (when inResizeMethod = Area).

This operation is optimized for AVX2 technology for pixels of types: UINT8 (when inResizeMethod = Area), UINT16 (when inResizeMethod = Area), 3xUINT8 (when inResizeMethod = Area), 3xUINT16 (when inResizeMethod = Area).

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty image on input in ResizeImage_FixedAspectRatio.
DomainError	Unsupported dimension given in inDimension, only Width and Height are supported in ResizeImage_FixedAspectRatio.

ResizeImage_Relative

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Resizes an image by a factor along each axis.

Syntax

```
void avl::ResizeImage_Relative
(
 const avl::Image& inImage,
 float inHorizontalScale,
 float inVerticalScale,
 avl::ResizeMethod::Type inResizeMethod,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inHorizontalScale	float	0.0 - 65535.0	1.0f	
→	inVerticalScale	float	0.0 - 65535.0	1.0f	
→	inResizeMethod	ResizeMethod::Type			
←	outImage	Image&			Output image

Description

The operation stretches or shrinks the **inImage** so that its dimensions are scaled, accordingly, by the factor of **inHorizontalScale**, **inVerticalScale**. Two modes of pixel interpolation are available, the bilinear filtering being more precise and computationally expensive.

Examples

ResizeImage_Relative *performed on the sample image with* ***inHorizontalScale*** *= 0.5,* ***inVerticalScale*** *= 0.25.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

This operation is optimized for SSE2 technology for pixels of types: UINT8 (when inResizeMethod = Area), UINT16 (when inResizeMethod = Area), 3xUINT8 (when inResizeMethod = Area), 3xUINT16 (when inResizeMethod = Area).

This operation is optimized for AVX2 technology for pixels of types: UINT8 (when inResizeMethod = Area), UINT16 (when inResizeMethod = Area), 3xUINT8 (when inResizeMethod = Area), 3xUINT16 (when inResizeMethod = Area).

See Also

- ResizeImage – Enlarges or shrinks an image to new dimensions.

RotateImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Rotates an image clockwise.

Syntax

```
void avl::RotateImage
(
 const avl::Image& inImage,
 float inAngle,
 avl::RotationSizeMode::Type inSizeMode,
 avl::InterpolationMethod::Type inInterpolationMethod,
 const bool inInverse,
 avl::Image& outImage,
 atl::Optional<avl::CoordinateSystem2D&> outOutputAlignment = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inAngle	float	45.0f	Rotation angle (clockwise)
→	inSizeMode	RotationSizeMode::Type		
→	inInterpolationMethod	InterpolationMethod::Type	Bilinear	
→	inInverse	const bool	False	'True' changes rotation to counter-clockwise
←	outImage	Image&		Output image
←	outOutputAlignment	Optional<CoordinateSystem2D&>	NIL	Alignment of the output image

Optional Outputs

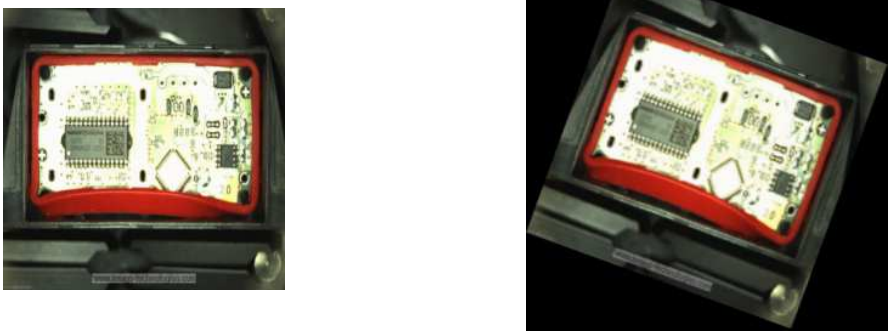
The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outOutputAlignment**.

Read more about Optional Outputs.

Description

The operation rotates the **inImage** around its center. Dimensions of the resulting image depends on **inSizeMode** parameter. In 'Fit' mode size is extended to fit the rotated image. In 'Preserve' mode size of source image is left unchanged and part of rotated image may be lost. Two modes of pixel interpolation are available, the bilinear filtering being more precise and computationally expensive.

Examples



***RotateImage** performed on the sample image in Fit mode with **inAngle** = 20.0.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Image dimensions too big in RotateImage.

ShearImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes a leant image (shifts the rows).

Applications: Image preprocessing when there are slanted objects.

Syntax

```
void avl::ShearImage
(
 const avl::Image& inImage,
 float inShear,
 avl::Axis::Type inAxis,
 avl::InterpolationMethod::Type inInterpolationMethod,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inShear	float	$-\infty$ $-\infty$	0.0f	Relative shift between consecutive rows or columns of the image
→	inAxis	Axis::Type			Switches between horizontal or vertical shearing
→	inInterpolationMethod	InterpolationMethod::Type		Bilinear	
←	outImage	Image&			Output image

Description

The filter **ShearImage** applies basic affine transform to each image's pixel.

Shear affine transform is defined as:

When **X axis** is selected

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & inShear \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

When **Y axis** is selected

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ inShear & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Examples

ShearImage *performed on the sample image with* ***inAxis*** *= X,* ***inShear*** *= 0.50.*

Errors

List of possible exceptions:

Error type	Description
DomainError	inImage and outImage are not distinct in ShearImage.

See Also

- ShearRegion – Computes a leant region.

ShrinkImageNTimes

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Shrinks an image by a natural factor along each axis.

Syntax

```
void avl::ShrinkImageNTimes
(
 const avl::Image& inImage,
 atl::Optional<const avl::Box&> inRoi,
 int inNX,
 const atl::Optional<int>& inNY,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Box&>		NIL	Range of pixels to be processed
→	inNX	int	1 - ∞	2	
→	inNY	const Optional<int>&	1 - ∞	NIL	
←	outImage	Image&			Output image

Description

The operation shrinks the **inImage** reducing its dimensions by a natural factor.

Examples

ShrinkImageNTimes *performed on the sample image with* ***inNX*** *= 2,* ***inNY*** *= 2.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

This operation is optimized for SSE2 technology for pixels of types: UINT8, SINT8, UINT16, SINT16, SINT32, REAL.

This operation is optimized for AVX2 technology for pixels of types: UINT8, SINT8, UINT16, SINT16, SINT32, REAL.

This operation is optimized for NEON technology for pixels of types: UINT8, SINT8, UINT16, SINT16, SINT32, REAL.

Errors

List of possible exceptions:

Error type	Description
DomainError	ROI is out of inImage range in ShrinkImageNTimes.

See Also

- DownsampleImage – Shrinks an image by the factor of two along each axis.
- ResizeImage – Enlarges or shrinks an image to new dimensions.
- ResizeImage_Relative – Resizes an image by a factor along each axis.
- ShrinkRegionNTimes – Shrinks a region by a natural factor along each axis.
- ShrinkProfileNTimes – Reduces the length of a profile N-times by averaging each N consecutive elements.

TransformImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Transforms an image by the provided transformation matrix.

Syntax

```
void avl::TransformImage
(
 const avl::Image& inImage,
 const avl::Matrix& inTransformMatrix,
 const bool inInverse,
 const avl::InterpolationMethod::Type& inInterpolation,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inTransformMatrix	const Matrix&		Transform matrix
→	inInverse	const bool		
→	inInterpolation	const InterpolationMethod::Type&		Image quality
←	outImage	Image&		Output image

Errors

List of possible exceptions:

Error type	Description
DomainError	Input transformation matrix must have dimensions 3x3 in TransformImage.
DomainError	Unable to transform image with provided parameters TransformImage.

TranslateImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Translates an image by a vector, without changing its dimensions.

Applications: E.g. camera shaking reduction.

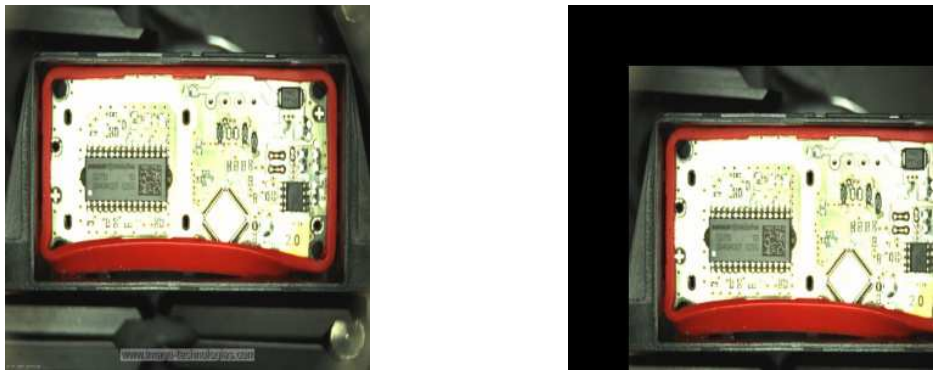
Syntax

```
void avl::TranslateImage
(
 const avl::Image& inImage,
 const avl::Pixel& inBorder,
 int inDeltaX,
 int inDeltaY,
 bool inInverse,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inBorder	const Pixel&		Color used to fill pixels outside of the translated image
→	inDeltaX	int		Horizontal shift
→	inDeltaY	int		Vertical shift
→	inInverse	bool		Switches to the inverse operation
←	outImage	Image&		Output image

Examples



TranslateImage *performed on the sample image.*

TranslatePixels

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Moves the pixels of the input image by the vectors specified with inVectorImage.

Syntax

```
void avl::TranslatePixels
(
 const avl::Image& inImage,
 const avl::Image& inVectorImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::InterpolationMethod::Type inImageInterpolation,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inVectorImage	const Image&		Two-channel image specifying translation vector for each pixel
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inImageInterpolation	InterpolationMethod::Type		
←	outImage	Image&		Output image

Requirements

For input **inVectorImage** only pixel formats are supported: 2□real.

Read more about pixel formats in Image documentation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Image size are not equal in TranslatePixels.
DomainError	Incorrect vector image format in TranslatePixels.
DomainError	Region exceeds an input image in TranslatePixels.
DomainError	Unknown interpolation method in TranslatePixels.
DomainError	Not supported inVectorImage pixel format in TranslatePixels. Supported formats: 2xReal.

TransposeImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Flips and rotates an image so that columns are exchanged with rows.

Applications: Useful when operations on image columns should be replaced with operations on image rows, which are much faster due to the cache memory.

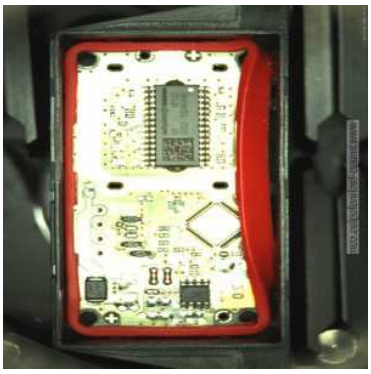
Syntax

```
void avl::TransposeImage
(
 const avl::Image& inImage,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
←	outImage	Image&		Output image

Examples



TransposeImage *performed on the sample image.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- MirrorImage – Reverses the order of the input image columns or rows depending on inMirrorDirection value.

TrimImageToRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Trims an image to the area of the specified region.

Syntax

```
void avl::TrimImageToRegion
(
 const avl::Image& inImage,
 const avl::Region& inRegion,
 const avl::Pixel& inBorderColor,
 avl::Image& outImage
)
```

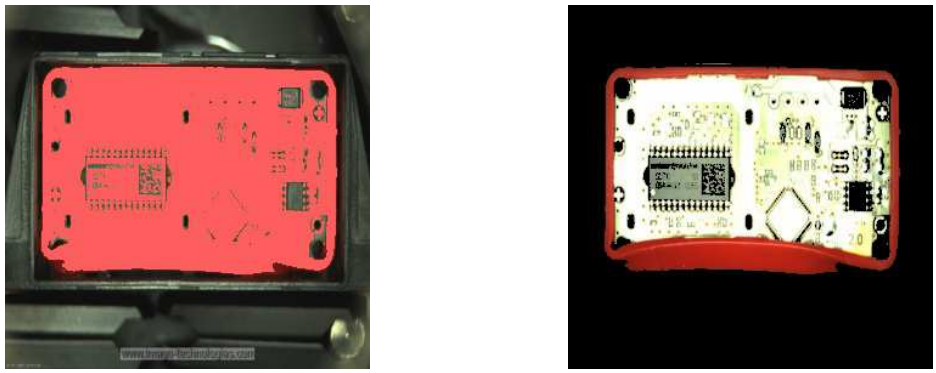
Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRegion	const Region&		Input region
→	inBorderColor	const Pixel&		Color used for locations outside the region
←	outImage	Image&		Output image

Description

The operation extracts part of the **inImage** that corresponds to the region. Other pixels are set to **inBorderColor** color.

Examples



TrimImageToRegion *performed on the sample image and region.*

UncropImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Inverse of CropImage.

Syntax

```
void avl::UncropImage
(
 const avl::Image& inImage,
 const avl::Box& inSelection,
 int inWidth,
 int inHeight,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inSelection	const Box&			The same value as in CropImage
→	inWidth	int	0 - ∞		Width of the original image
→	inHeight	int	0 - ∞		Height of the original image
←	outImage	Image&			Output image

Description

Inverse of CropImage. Can create borders around original image e.g. passing x:50 y:50 will create new image with filled 50 pixels on top and 50 on left of the original image. inWidth and inHeight specifies the new image dimensions, Width and Height from inSelection property are not used.

Examples

inSelection property	Preview
origin in x:0 y:0	
origin in x:50 y:50	

See Also

- CropImage – Creates an image from a box-shaped fragment of the input image (with margins if requested).

70. Shape Fitting

Table of content:

CreateArcFittingMap

Header: AVL.h
Namespace: avl
Module: MetrologyPro

Precomputes a data object that is required for fast arc fitting on images.

Applications: Used together with arc fitting, but can be moved before the loop.

Syntax

```
void avl::CreateArcFittingMap
(
 const avl::ImageFormat& inImageFormat,
 const avl::ArcFittingField& inFittingField,
 atl::Optional<const avl::CoordinateSystem2D&> inFittingFieldAlignment,
 const int inScanCount,
 const int inScanWidth,
 const avl::SamplingParams& inSamplingParams,
 avl::ArcFittingMap& outFittingMap,
 atl::Array<avl::Segment2D>& diagScanSegments,
 atl::Array<avl::Rectangle2D>& diagSamplingAreas
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImageFormat	const ImageFormat&			Dimensions, depth and pixel type of the images on which fitting will be performed
→	inFittingField	const ArcFittingField&			Defines a ring section in which scan segments will be created
→	inFittingFieldAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the fitting field to the position of the inspected object
→	inScanCount	const int	3 - ∞	10	The number of points that will be searched to estimate the position of the arc
→	inScanWidth	const int	1 - ∞	5	The width of each scan field (in pixels)
→	inSamplingParams	const SamplingParams&		.interpolation Bilinear	Parameters controlling the sampling process
←	outFittingMap	ArcFittingMap&			Optimized data required for arc fitting
🔍	diagScanSegments	Array<Segment2D>&			Segments along which the scans will be run
🔍	diagSamplingAreas	Array<Rectangle2D>&			Scan fields created for point detection

Description

The operation creates a series of scan maps that can be later used by other Shape Fitting filters. Each scan map corresponds to a single scan segment of **inScanWidth** length.

The optional parameter **inFittingFieldAlignment** defines the transform to be performed on the **inFittingField** so that the result is defined in a new context, e.g. returned by one of Template Matching filters.

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the Shape Fitting toolset. To read more about this technique, one can refer to the Shape Fitting chapter of our Machine Vision Guide

See Also

- FitArcToEdges – Performs a series of 1D edge detections and finds an arc that best matches the detected points.
- FitArcToRidges – Performs a series of 1D ridge detections and finds an arc that best matches the detected points.
- FitArcToStripe – Performs a series of 1D stripe detections and finds an arc that best matches the detected points.
- FitArcToEdges_Direct – Performs a series of 1D edge detections and finds an arc that best matches the detected points.
- FitArcToRidges_Direct – Performs a series of 1D ridge detections and finds an arc that best matches the detected points.
- FitArcToStripe_Direct – Performs a series of 1D stripe detections and finds an arc that best matches the detected points.

CreateCircleFittingMap

Header: AVL.h
Namespace: avl
Module: MetrologyPro

Precomputes a data object that is required for fast circle fitting on images.

Applications: Used together with circle fitting, but can be moved before the loop.

Syntax

```
void avl::CreateCircleFittingMap
(
 const avl::ImageFormat& inImageFormat,
 const avl::CircleFittingField& inFittingField,
 atl::Optional<const avl::CoordinateSystem2D&> inFittingFieldAlignment,
 const int inScanCount,
 const int inScanWidth,
 const avl::SamplingParams& inSamplingParams,
 avl::CircleFittingMap& outFittingMap,
 atl::Array<avl::Segment2D>& diagScanSegments,
 atl::Array<avl::Rectangle2D>& diagSamplingAreas
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImageFormat	const ImageFormat&			Dimensions, depth and pixel type of the images on which fitting will be performed
→	inFittingField	const CircleFittingField&			Defines a ring in which scan segments will be created
→	inFittingFieldAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the fitting field to the position of the inspected object
→	inScanCount	const int	3 - ∞	10	The number of points that will be searched to estimate the position of the circle
→	inScanWidth	const int	1 - ∞	5	The width of each scan field (in pixels)
→	inSamplingParams	const SamplingParams&		.interpolation Bilinear	Parameters controlling the sampling process
←	outFittingMap	CircleFittingMap&			Optimized data required for circle fitting
🔍	diagScanSegments	Array<Segment2D>&			Segments along which the scans will be run
🔍	diagSamplingAreas	Array<Rectangle2D>&			Scan fields created for point detection

Description

The operation creates a series of scan maps that can be later used by other Shape Fitting filters. Each scan map corresponds to a single scan segment of **inScanWidth** length.

The optional parameter **inFittingFieldAlignment** defines the transform to be performed on the **inFittingField** so that the result is defined in a new context, e.g. returned by one of Template Matching filters.

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the Shape Fitting toolset. To read more about this technique, one can refer to the Shape Fitting chapter of our Machine Vision Guide

See Also

- FitCircleToEdges – Performs a series of 1D edge detections and finds a circle that best matches the detected points.
- FitCircleToRidges – Performs a series of 1D ridge detections and finds a circle that best matches the detected points.
- FitCircleToStripe – Performs a series 1D stripe detections and finds a circle that best matches the detected points.
- FitCircleToEdges_Direct – Performs a series of 1D edge detections and finds a circle that best matches the detected points.
- FitCircleToRidges_Direct – Performs a series of 1D ridge detections and finds a circle that best matches the detected points.
- FitCircleToStripe_Direct – Performs a series 1D stripe detections and finds a circle that best matches the detected points.

CreatePathFittingMap

Header: AVL.h
Namespace: avl
Module: MetrologyPro

Precomputes a data object that is required for fast path fitting on images.

Applications: Used together with path fitting, but can be moved before the loop.

Syntax

```
void avl::CreatePathFittingMap
(
 const avl::ImageFormat& inImageFormat,
 const avl::PathFittingField& inFittingField,
 atl::Optional<const avl::CoordinateSystem2D&> inFittingFieldAlignment,
 atl::Optional<float> inScanStep,
 const int inScanWidth,
 const avl::SamplingParams& inSamplingParams,
 avl::PathFittingMap& outFittingMap,
 atl::Array<avl::Segment2D>& diagScanSegments,
 atl::Array<avl::Rectangle2D>& diagSamplingAreas
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImageFormat	const ImageFormat&			Dimensions, depth and pixel type of the images on which fitting will be performed
→	inFittingField	const PathFittingField&			Defines a stripe in which scan segments will be created
→	inFittingFieldAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the fitting field to the position of the inspected object
→	inScanStep	Optional<float>	0.0 - ∞	NIL	Optional implicit conversion of the input path to an equidistant one
→	inScanWidth	const int	1 - ∞	5	The width of each scan field (in pixels)
→	inSamplingParams	const SamplingParams&		.interpolation Bilinear	Parameters controlling the sampling process
←	outFittingMap	PathFittingMap&			Optimized data required for path fitting
🔍	diagScanSegments	Array<Segment2D>&			Segments along which the scans will be run
🔍	diagSamplingAreas	Array<Rectangle2D>&			Scan fields created for point detection

Description

The operation creates a series of scan maps that can be later used by other Shape Fitting filters. Each scan map corresponds to a single scan segment of **inScanWidth** length.

The optional parameter **inFittingFieldAlignment** defines the transform to be performed on the **inFittingField** so that the result is defined in a new context, e.g. returned by one of Template Matching filters.

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the Shape Fitting toolset. To read more about this technique, one can refer to the Shape Fitting chapter of our Machine Vision Guide

See Also

- FitPathToEdges – Performs a series of 1D edge detections and creates a path from the detected points.
- FitPathToRidges – Performs a series of 1D ridge detections and creates a path from the detected points.
- FitPathToStripe – Performs a series of 1D stripe detections and creates a path from the detected points.
- FitPathToEdges_Direct – Performs a series of 1D edge detections and creates a path from the detected points.
- FitPathToRidges_Direct – Performs a series of 1D ridge detections and creates a path from the detected points.
- FitPathToStripe_Direct – Performs a series of 1D stripe detections and creates a path from the detected points.

CreateSegmentFittingMap

Header: AVL.h
Namespace: avl
Module: MetrologyPro

Precomputes a data object that is required for fast segment fitting on images.

Applications: Used together with segment fitting, but can be moved before the loop.

Syntax

```
void avl::CreateSegmentFittingMap
(
 const avl::ImageFormat& inImageFormat,
 const avl::SegmentFittingField& inFittingField,
 atl::Optional<const avl::CoordinateSystem2D&> inFittingFieldAlignment,
 const int inScanCount,
 const int inScanWidth,
 const avl::SamplingParams& inSamplingParams,
 SegmentFittingMap& outFittingMap,
 atl::Array<avl::Segment2D>& diagScanSegments,
 atl::Array<avl::Rectangle2D>& diagSamplingAreas
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImageFormat	const ImageFormat&			Dimensions, depth and pixel type of the images on which fitting will be performed
→	inFittingField	const SegmentFittingField&			Defines a rectangle in which scan segments will be created
→	inFittingFieldAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the fitting field to the position of the inspected object
→	inScanCount	const int	3 - ∞	10	The number of points that will be searched to estimate the position of the segment
→	inScanWidth	const int	1 - ∞	5	The width of each scan field (in pixels)
→	inSamplingParams	const SamplingParams&		.interpolation Bilinear	Parameters controlling the sampling process
←	outFittingMap	SegmentFittingMap&			Optimized data required for segment fitting
🔍	diagScanSegments	Array<Segment2D>&			Segments along which the scans will be run
🔍	diagSamplingAreas	Array<Rectangle2D>&			Scan fields created for point detection

Description

The operation creates a series of scan maps that can be later used by other Shape Fitting filters. Each scan map corresponds to a single scan segment of **inScanWidth** length.

The optional parameter **inFittingFieldAlignment** defines the transform to be performed on the **inFittingField** so that the result is defined in a new context, e.g. returned by one of Template Matching filters.

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the Shape Fitting toolset. To read more about this technique, one can refer to the Shape Fitting chapter of our Machine Vision Guide

See Also

- FitSegmentToEdges – Performs a series of 1D edge detections and finds a segment that best matches the detected points.
- FitSegmentToRidges – Performs a series of 1D ridge detections and finds a segment that best matches the detected points.
- FitSegmentToStripe – Performs a series of 1D stripe detections and finds a segment that best matches the detected points.
- FitSegmentToEdges_Direct – Performs a series of 1D edge detections and finds a segment that best matches the detected points.
- FitSegmentToRidges_Direct – Performs a series of 1D ridge detections and finds a segment that best matches the detected points.
- FitSegmentToStripe_Direct – Performs a series of 1D stripe detections and finds a segment that best matches the detected points.

DetectArc_LSD

Header: AVL.h
Namespace: avl
Module: FoundationPro

Finds arcs in an image using Line Segment Detection method.

Syntax

```
void avl::DetectArc_LSD
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::GaussKernel::Type inSmoothing,
 float inEdgeThreshold,
 int inMinLength,
 atl::Array<avl::Arc2D>& outArcs
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inImage	const Image&			Image to fit the arcs to
➡	inRoi	Optional<const Region&>		NIL	Input region of interest
➡	inSmoothing	GaussKernel::Type		_5x5	Predefined Gauss kernel
➡	inEdgeThreshold	float	0.1 - ∞	4.0f	Minimum accepted edge magnitude
➡	inMinLength	int	0 - ∞	10	Minimum length of output arcs
⬅	outArcs	Array<Arc2D>&			Found arcs

DetectCircle_LSD

Header: AVL.h
Namespace: avl
Module: FoundationPro

Finds circles in the input image using Line Segment Detection method.

Syntax

```
void avl::DetectCircle_LSD
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::GaussKernel::Type inSmoothing,
 float inEdgeThreshold,
 float inToleranceCircle,
 int inMinRadius,
 atl::Array<avl::Circle2D>& outCircles
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inImage	const Image&			Image to fit the circles to
➡	inRoi	Optional<const Region&>		NIL	Input region of interest
➡	inSmoothing	GaussKernel::Type		_5x5	Predefined Gauss kernel
➡	inEdgeThreshold	float	0.1 - ∞	4.0f	Minimum accepted edge magnitude
➡	inToleranceCircle	float	0.001 - ∞	0.6f	Maximum accepted defects in output circles
➡	inMinRadius	int	0 - ∞	10	Minimum radius of output circles
⬅	outCircles	Array<Circle2D>&			Found circles

DetectPolygons_LSD

Header: AVL.h
Namespace: avl
Module: FoundationPro

Finds a specified polygons in an image using Line Segment Detection method.

Syntax

```
void avl::DetectPolygons_LSD
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 const avl::Path& inPath,
 avl::GaussKernel::Type inSmoothing,
 float inEdgeThreshold,
 float inTolerance,
 atl::Array<avl::Path>& outPaths
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image to fit the polygons to
→	inRoi	Optional<const Region&>		NIL	Input region of interest
→	inPath	const Path&			Input Polygon
→	inSmoothing	GaussKernel::Type		_5x5	Predefined Gauss kernel
→	inEdgeThreshold	float	0.1 - ∞	4.0f	Minimum accepted edge magnitude
→	inTolerance	float	0.001 - ∞	1.f	Maximum accepted defects in output polygons
←	outPaths	Array<Path>&			Found polygons

DetectSegments_LSD

Header: AVL.h
Namespace: avl
Module: FoundationPro

Finds segments in an image using Line Segment Detection method.

Syntax

```
void avl::DetectSegments_LSD
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::GaussKernel::Type inSmoothing,
 float inEdgeThreshold,
 int inMinLength,
 atl::Array<avl::Segment2D>& outSegments
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image to fit the segments to
→	inRoi	Optional<const Region&>		NIL	Input region of interest
→	inSmoothing	GaussKernel::Type		_5x5	Predefined Gauss kernel
→	inEdgeThreshold	float	0.1 - ∞	4.0f	Minimum accepted edge magnitude
→	inMinLength	int	0 - ∞	10	Minimum length of output segments
←	outSegments	Array<Segment2D>&			Found segments

FitArcToEdges

Header: AVL.h
Namespace: avl
Module: MetrologyPro

Performs a series of 1D edge detections and finds an arc that best matches the detected points.

Applications: Precise detection of an arciform edge, whose rough location is known beforehand.

Syntax

```
void avl::FitArcToEdges
(
 const avl::Image& inImage,
 const avl::ArcFittingMap& inFittingMap,
 const EdgeScanParams& inEdgeScanParams,
 avl::Selection::Type inEdgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 float inMaxIncompleteness,
 avl::CircleFittingMethod::Type inFittingMethod,
 atl::Optional<avl::MEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Arc2D>& outArc,
 atl::Optional<atl::Array<atl::Conditional<avl::Edge1D>>&> outEdges = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<atl::Array<avl::Point2D>&> outInliers = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outBrightnessProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image to fit arc to
→	inFittingMap	const ArcFittingMap&			Input fitting map
→	inEdgeScanParams	const EdgeScanParams&			Parameters controlling the edge extraction process
→	inEdgeSelection	Selection::Type		Selection::Best	Selection mode of edges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of edge points not found
→	inFittingMethod	CircleFittingMethod::Type		AlgebraicTaubin	Method used to fit an arc
→	inOutlierSuppression	Optional<MEstimator::Type>		NIL	Selects a method for ignoring incorrectly detected points
←	outArc	Conditional<Arc2D>&			Fitted arc or nothing if the fitting fails
←	outEdges	Optional<Array<Conditional<Edge1D>>&>		NIL	Found edges
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual arc points and the corresponding reference arc points
←	outInliers	Optional<Array<Point2D>&>		NIL	Points matching the fitting arc
←	outBrightnessProfiles	Optional<Array<Profile>&>		NIL	Extracted image profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the edge (derivative) operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outEdges**, **outDeviationProfile**, **outInliers**, **outBrightnessProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

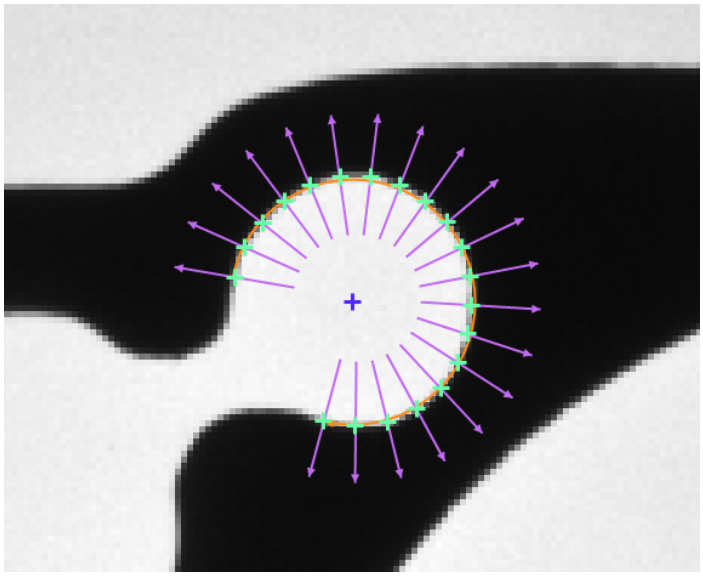
Description

The operation tries to fit a given arc to edges present in the **inImage** image. Internally, it performs a series of scans with the ScanSingleEdge filter using **inFittingMap** previously generated from the object being fitted. The found points are then used to determine the actual position of the arc in the image. Only **inMaxIncompleteness** fraction of these scans may fail. If the fitting according to the given parameters is not possible, **outArc** is set to Nil.

Hints

- Connect an input image to the **inImage** input.
- Define **inEdgeScanParams.EdgeTransition** to detect a particular edge type, and only that type.
- If no or too few edge points are found, try decreasing **inEdgeScanParams.MinMagnitude**.
- If some of the scans may fail, set the **inMaxIncompleteness** input accordingly.
- If some of the scans may produce false results, try different values of the **inOutlierSuppression** input.
- Use the **outEdges** outputs to visualize the scanning results.

Examples



Fitting an arc to the edges of a semi-hole
*(**inEdgeScanParams.Transition** = BrightToDark).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the Shape Fitting toolset. To read more about this technique, one can refer to the Shape Fitting chapter of our Machine Vision Guide

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- CreateArcFittingMap – Precomputes a data object that is required for fast arc fitting on images.
- FitArcToRidges – Performs a series of 1D ridge detections and finds an arc that best matches the detected points.
- FitArcToStripe – Performs a series of 1D stripe detections and finds an arc that best matches the detected points.
- FitArcToEdges_Direct – Performs a series of 1D edge detections and finds an arc that best matches the detected points.

FitArcToEdges_Direct

Header: AVL.h
Namespace: avl
Module: MetrologyPro

Performs a series of 1D edge detections and finds an arc that best matches the detected points.

Applications: Precise detection of an arciform edge, whose rough location is known beforehand.

Syntax

```
void avl::FitArcToEdges_Direct
(
 const avl::Image& inImage,
 const avl::ArcFittingField& inFittingField,
 atl::Optional<const avl::CoordinateSystem2D&> inFittingFieldAlignment,
 int inScanCount,
 int inScanWidth,
 const avl::SamplingParams& inSamplingParams,
 const avl::EdgeScanParams& inEdgeScanParams,
 avl::Selection::Type inEdgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 float inMaxIncompleteness,
 avl::CircleFittingMethod::Type inFittingMethod,
 atl::Optional<avl::MEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Arc2D>& outArc,
 atl::Optional<atl::Array<atl::Conditional<avl::Edge1D>>&> outEdges = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<avl::ArcFittingField&> outAlignedFittingField = atl::NIL,
 atl::Optional<atl::Array<avl::Point2D>&> outInliers = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outBrightnessProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL,
 atl::Array<avl::Segment2D>& diagScanSegments,
 atl::Array<avl::Rectangle2D>& diagSamplingAreas
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image to fit arc to
→	inFittingField	const ArcFittingField&			Arc fitting field
→	inFittingFieldAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the fitting field to the position of the inspected object
→	inScanCount	int	3 - ∞	10	The number of points that will be searched to estimate the position of the arc
→	inScanWidth	int	1 - ∞	5	The width of each scan field (in pixels)
→	inSamplingParams	const SamplingParams&			Parameters controlling the sampling process
→	inEdgeScanParams	const EdgeScanParams&		EdgeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 1.0f MinMagnitude: 5.0f EdgeTransition: BrightToDark)	Parameters controlling the edge extraction process
→	inEdgeSelection	Selection::Type			Selection mode of edges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of edge points not found
→	inFittingMethod	CircleFittingMethod::Type		AlgebraicTaubin	Method used to fit an arc
→	inOutlierSuppression	Optional<MEstimator::Type>		NIL	Selects a method for ignoring incorrectly detected points
←	outArc	Conditional<Arc2D>&			Fitted arc or nothing if the fitting fails
←	outEdges	Optional<Array<Conditional<Edge1D>>&>		NIL	Found edges
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual arc points and the corresponding reference arc points
←	outAlignedFittingField	Optional<ArcFittingField&>		NIL	Fitting field used; in the image coordinate system
←	outInliers	Optional<Array<Point2D>&>		NIL	Points matching the fitting arc
←	outBrightnessProfiles	Optional<Array<Profile>&>		NIL	Extracted image profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the edge (derivative) operator response
🔍	diagScanSegments	Array<Segment2D>&			Segments along which the scans were run
🔍	diagSamplingAreas	Array<Rectangle2D>&			Areas from which the input image is sampled

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outEdges**, **outDeviationProfile**, **outAlignedFittingField**, **outInliers**, **outBrightnessProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

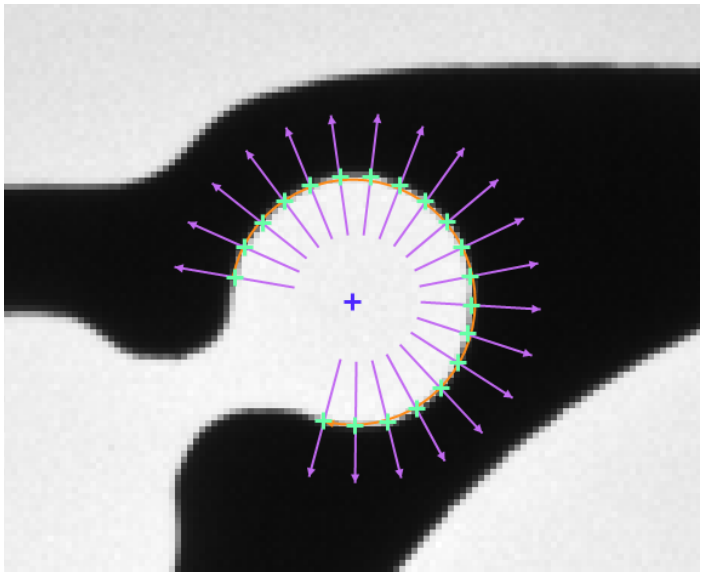
Description

The operation tries to fit a given arc to edges present in the **inImage** image. Internally, it performs a series of scans with the ScanSingleEdge_Direct filter along **inScanCount** specific scan segments which length is always equal to the **inFittingField** width and cannot be less than 4. The found points are then used to determine the actual position of the arc in the image. Only **inMaxIncompleteness** fraction of these scans may fail. If the fitting according to the given parameters is not possible, **outArc** is set to Nil.

Hints

- Connect an input image to the **inImage** input.
- Define **inEdgeScanParams.EdgeTransition** to detect a particular edge type, and only that type.
- If no or too few edge points are found, try decreasing **inEdgeScanParams.MinMagnitude**.
- If some of the scans may fail, set the **inMaxIncompleteness** input accordingly.
- If some of the scans may produce false results, try different values of the **inOutlierSuppression** input.
- Use the **outEdges** outputs to visualize the scanning results.

Examples



Fitting an arc to the edges of a semi-hole
*(**inEdgeScanParams.Transition** = BrightToDark).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the Shape Fitting toolset. To read more about this technique, one can refer to the Shape Fitting chapter of our Machine Vision Guide

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- CreateArcFittingMap – Precomputes a data object that is required for fast arc fitting on images.
- FitArcToRidges_Direct – Performs a series of 1D ridge detections and finds an arc that best matches the detected points.
- FitArcToStripe_Direct – Performs a series of 1D stripe detections and finds an arc that best matches the detected points.
- FitArcToEdges – Performs a series of 1D edge detections and finds an arc that best matches the detected points.

FitArcToRidges

Header: AVL.h
Namespace: avl
Module: MetrologyPro

Performs a series of 1D ridge detections and finds an arc that best matches the detected points.

Applications: Precise detection of a thin arciform line, whose rough location is known beforehand.

Syntax

```
void avl::FitArcToRidges
(
 const avl::Image& inImage,
 const avl::ArcFittingMap& inFittingMap,
 const RidgeScanParams& inRidgeScanParams,
 avl::Selection::Type inRidgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 float inMaxIncompleteness,
 avl::CircleFittingMethod::Type inFittingMethod,
 atl::Optional<avl::MEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Arc2D>& outArc,
 atl::Optional<atl::Array<atl::Conditional<avl::Ridge1D>>&> outRidges = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<atl::Array<avl::Point2D>&> outInliers = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outBrightnessProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image to fit arc to
→	inFittingMap	const ArcFittingMap&			Input fitting map
→	inRidgeScanParams	const RidgeScanParams&			Parameters controlling the ridge extraction process
→	inRidgeSelection	Selection::Type		Selection::Best	Selection mode of ridges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker ridges can be detected in the vicinity of stronger ridges
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of ridge points not found
→	inFittingMethod	CircleFittingMethod::Type		AlgebraicTaubin	Method used to fit an arc
→	inOutlierSuppression	Optional<MEstimator::Type>		NIL	Selects a method for ignoring incorrectly detected points
←	outArc	Conditional<Arc2D>&			Fitted arc or nothing if the fitting fails
←	outRidges	Optional<Array<Conditional<Ridge1D>>&>		NIL	Found ridges
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual arc points and the corresponding reference arc points
←	outInliers	Optional<Array<Point2D>&>		NIL	Points matching the fitting Arc
←	outBrightnessProfiles	Optional<Array<Profile>&>		NIL	Extracted image profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the ridge operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outRidges**, **outDeviationProfile**, **outInliers**, **outBrightnessProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

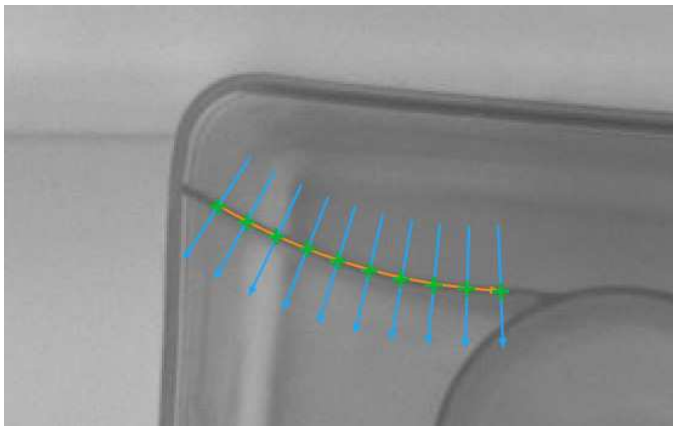
Description

The operation tries to fit a given arc to ridges present in the **inImage** image. Internally, it performs a series of scans with the ScanSingleRidge filter using **inFittingMap** previously generated from the object being fitted. The found points are then used to determine the actual position of the arc in the image. Only **inMaxIncompleteness** fraction of these scans may fail. If the fitting according to the given parameters is not possible, **outArc** is set to Nil.

Hints

- Connect an input image to the **inImage** input.
- Define **inRidgeScanParams.RidgePolarity** to detect a particular ridge type, and only that type.
- Adjust **inRidgeScanParams.RidgeWidth** to the expected thickness of the ridge (in pixels).
- If no or too few ridge points are found, try decreasing **inRidgeScanParams.MinMagnitude**.
- If some of the scans may fail, set the **inMaxIncompleteness** input accordingly.
- If some of the scans may produce false results, try different values of the **inOutlierSuppression** input.
- Use the **outRidges** outputs to visualize the scanning results.

Examples



Fitting an arc to a curved wall of a plastic capsule
*(**inRidgeScanParams.RidgeWidth** = 3, **inRidgeScanParams.Polarity** = Dark).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the Shape Fitting toolset. To read more about this technique, one can refer to the Shape Fitting chapter of our Machine Vision Guide

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- CreateArcFittingMap – Precomputes a data object that is required for fast arc fitting on images.
- FitArcToEdges – Performs a series of 1D edge detections and finds an arc that best matches the detected points.
- FitArcToStripe – Performs a series of 1D stripe detections and finds an arc that best matches the detected points.
- FitArcToRidges_Direct – Performs a series of 1D ridge detections and finds an arc that best matches the detected points.

FitArcToRidges_Direct

Header: AVL.h
Namespace: avl
Module: MetrologyPro

Performs a series of 1D ridge detections and finds an arc that best matches the detected points.

Applications: Precise detection of a thin arciform line, whose rough location is known beforehand.

Syntax

```
void avl::FitArcToRidges_Direct
(
 const avl::Image& inImage,
 const avl::ArcFittingField& inFittingField,
 atl::Optional<const avl::CoordinateSystem2D&> inFittingFieldAlignment,
 int inScanCount,
 int inScanWidth,
 const avl::SamplingParams& inSamplingParams,
 const avl::RidgeScanParams& inRidgeScanParams,
 avl::Selection::Type inRidgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 float inMaxIncompleteness,
 avl::CircleFittingMethod::Type inFittingMethod,
 atl::Optional<avl::MEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Arc2D>& outArc,
 atl::Optional<atl::Array<atl::Conditional<avl::Ridge1D>>&> outRidges = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<avl::ArcFittingField&> outAlignedFittingField = atl::NIL,
 atl::Optional<atl::Array<avl::Point2D>&> outInliers = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outBrightnessProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL,
 atl::Array<avl::Segment2D>& diagScanSegments,
 atl::Array<avl::Rectangle2D>& diagSamplingAreas
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image to fit arc to
→	inFittingField	const ArcFittingField&			Arc fitting field
→	inFittingFieldAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the fitting field to the position of the inspected object
→	inScanCount	int	3 - ∞	10	The number of points that will be searched to estimate the position of the arc
→	inScanWidth	int	1 - ∞	5	The width of each scan field (in pixels)
→	inSamplingParams	const SamplingParams&			Parameters controlling the sampling process
→	inRidgeScanParams	const RidgeScanParams&		RidgeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 1.0f RidgeWidth: 5 RidgeMargin: 2 MinMagnitude: 5.0f RidgePolarity: Dark)	Parameters controlling the ridge extraction process
→	inRidgeSelection	Selection::Type			Selection mode of ridges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker ridges can be detected in the vicinity of stronger ridges
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of ridge points not found
→	inFittingMethod	CircleFittingMethod::Type		AlgebraicTaubin	Method used to fit an arc
→	inOutlierSuppression	Optional<MEstimator::Type>		NIL	Selects a method for ignoring incorrectly detected points
←	outArc	Conditional<Arc2D>&			Fitted arc or nothing if the fitting fails
←	outRidges	Optional<Array<Conditional<Ridge1D>>&>		NIL	Found ridges
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual arc points and the corresponding reference arc points
←	outAlignedFittingField	Optional<ArcFittingField&>		NIL	Fitting field used; in the image coordinate system
←	outInliers	Optional<Array<Point2D>&>		NIL	Points matching the fitting Arc
←	outBrightnessProfiles	Optional<Array<Profile>&>		NIL	Extracted image profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the ridge operator response
🔍	diagScanSegments	Array<Segment2D>&			Segments along which the scans were run
🔍	diagSamplingAreas	Array<Rectangle2D>&			Areas from which the input image is sampled

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outRidges**, **outDeviationProfile**, **outAlignedFittingField**, **outInliers**, **outBrightnessProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

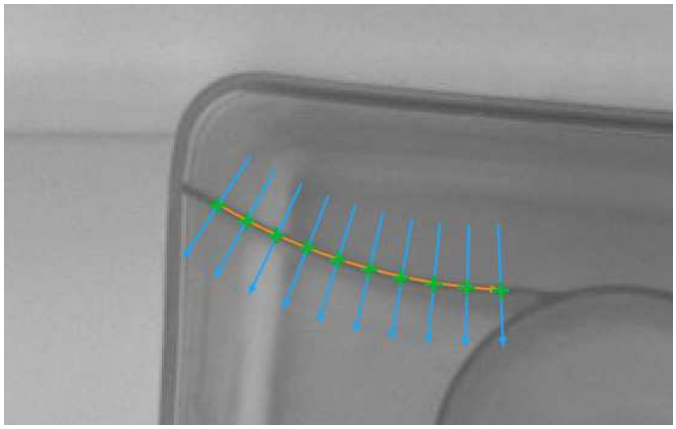
Description

The operation tries to fit a given arc to ridges present in the **inImage** image. Internally, it performs a series of scans with the ScanSingleRidge_Direct filter along **inScanCount** specific scan segments which length is always equal to the **inFittingField** width and cannot be less than 4. The found points are then used to determine the actual position of the arc in the image. Only **inMaxIncompleteness** fraction of these scans may fail. If the fitting according to the given parameters is not possible, **outArc** is set to Nil.

Hints

- Connect an input image to the **inImage** input.
- Define **inRidgeScanParams.RidgePolarity** to detect a particular ridge type, and only that type.
- Adjust **inRidgeScanParams.RidgeWidth** to the expected thickness of the ridge (in pixels).
- If no or too few ridge points are found, try decreasing **inRidgeScanParams.MinMagnitude**.
- If some of the scans may fail, set the **inMaxIncompleteness** input accordingly.
- If some of the scans may produce false results, try different values of the **inOutlierSuppression** input.
- Use the **outRidges** outputs to visualize the scanning results.

Examples



Fitting an arc to a curved wall of a plastic capsule
*(**inRidgeScanParams.RidgeWidth** = 3, **inRidgeScanParams.Polarity** = Dark).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the Shape Fitting toolset. To read more about this technique, one can refer to the Shape Fitting chapter of our Machine Vision Guide

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- CreateArcFittingMap – Precomputes a data object that is required for fast arc fitting on images.
- FitArcToEdges_Direct – Performs a series of 1D edge detections and finds an arc that best matches the detected points.
- FitArcToStripe_Direct – Performs a series of 1D stripe detections and finds an arc that best matches the detected points.
- FitArcToRidges – Performs a series of 1D ridge detections and finds an arc that best matches the detected points.

FitArcToStripe

Header: AVL.h
Namespace: avl
Module: MetrologyPro

Performs a series of 1D stripe detections and finds an arc that best matches the detected points.

Applications: Precise detection of a ring section, whose rough location is known beforehand.

Syntax

```
void avl::FitArcToStripe
(
 const avl::Image& inImage,
 const avl::ArcFittingMap& inFittingMap,
 const StripeScanParams& inStripeScanParams,
 avl::Selection::Type inStripeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 float inMaxIncompleteness,
 avl::CircleFittingMethod::Type inFittingMethod,
 atl::Optional<avl::MEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Arc2D>& outArc,
 atl::Conditional<avl::Arc2D>& outInnerArc,
 atl::Conditional<avl::Arc2D>& outOuterArc,
 atl::Optional<atl::Array<atl::Conditional<avl::Stripe1D>>&> outStripes = atl::NIL,
 atl::Optional<atl::Array<avl::Point2D>&> outStripePoints = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outBrightnessProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image to fit arc to
→	inFittingMap	const ArcFittingMap&			Input fitting map
→	inStripeScanParams	const StripeScanParams&			Parameters controlling the stripe extraction process
→	inStripeSelection	Selection::Type		Selection::Best	Selection mode of stripe
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of stripe points not found
→	inFittingMethod	CircleFittingMethod::Type		AlgebraicTaubin	Method used to fit an arc
→	inOutlierSuppression	Optional<MEstimator::Type>		NIL	Selects a method for ignoring incorrectly detected points
←	outArc	Conditional<Arc2D>&			Fitted arc in the middle of found stripe
←	outInnerArc	Conditional<Arc2D>&			Fitted inner arc
←	outOuterArc	Conditional<Arc2D>&			Fitted outer arc
←	outStripes	Optional<Array<Conditional<Stripe1D>>&>		NIL	Found stripes
←	outStripePoints	Optional<Array<Point2D>&>		NIL	Extracted points of middle arc of an image stripe
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual arc points and the corresponding reference arc points
←	outBrightnessProfiles	Optional<Array<Profile>&>		NIL	Extracted image profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the edge (derivative) operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outStripes**, **outStripePoints**, **outDeviationProfile**, **outBrightnessProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

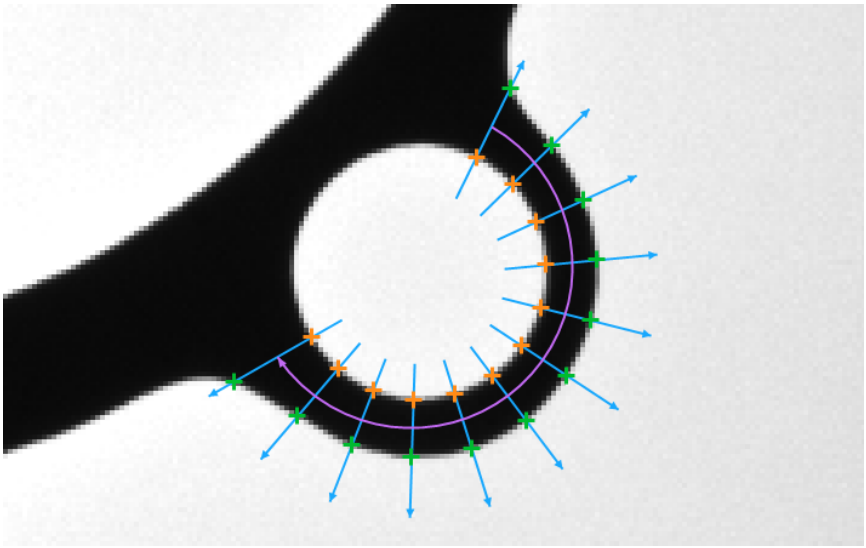
Description

The operation tries to fit a given arc to stripe present in the **inImage** image. Internally, it performs a series of scans with the ScanSingleStripe filter using **inFittingMap** previously generated from the object being fitted. The found points are then used to determine the actual position of the arc in the image. Only **inMaxIncompleteness** fraction of these scans may fail. If the fitting according to the given parameters is not possible, **outArc** is set to Nil.

Hints

- Connect an input image to the **inImage** input.
- Define **inStripeScanParams.StripePolarity** to detect a particular stripe type, and only that type.
- Adjust **inStripeScanParams.MinStripeWidth** and **inStripeScanParams.MaxStripeWidth** to the expected thickness of the stripe (in pixels).
- If no or too few stripe points are found, try decreasing **inStripeScanParams.MinMagnitude**.
- If some of the scans may fail, set the **inMaxIncompleteness** input accordingly.
- If some of the scans may produce false results, try different values of the **inOutlierSuppression** input.
- Use the **outStripePoints** outputs to visualize the scanning results.

Examples



Fitting an arc to the dark stripe formed on one side of a hole
*(**inStripeScanParams.Polarity** = Dark).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the Shape Fitting toolset. To read more about this technique, one can refer to the Shape Fitting chapter of our Machine Vision Guide

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- CreateArcFittingMap – Precomputes a data object that is required for fast arc fitting on images.
- FitArcToEdges – Performs a series of 1D edge detections and finds an arc that best matches the detected points.
- FitArcToRidges – Performs a series of 1D ridge detections and finds an arc that best matches the detected points.
- FitArcToStripe_Direct – Performs a series of 1D stripe detections and finds an arc that best matches the detected points.

FitArcToStripe_Direct

Header: AVL.h
Namespace: avl
Module: MetrologyPro

Performs a series of 1D stripe detections and finds an arc that best matches the detected points.

Applications: Precise detection of a ring section, whose rough location is known beforehand.

Syntax

```
void avl::FitArcToStripe_Direct
(
 const avl::Image& inImage,
 const avl::ArcFittingField& inFittingField,
 atl::Optional<const avl::CoordinateSystem2D&> inFittingFieldAlignment,
 int inScanCount,
 int inScanWidth,
 const avl::SamplingParams& inSamplingParams,
 const avl::StripeScanParams& inStripeScanParams,
 avl::Selection::Type inStripeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 float inMaxIncompleteness,
 avl::CircleFittingMethod::Type inFittingMethod,
 atl::Optional<avl::MEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Arc2D>& outArc,
 atl::Conditional<avl::Arc2D>& outInnerArc,
 atl::Conditional<avl::Arc2D>& outOuterArc,
 atl::Optional<atl::Array<atl::Conditional<avl::Stripe1D>>&> outStripes = atl::NIL,
 atl::Optional<atl::Array<avl::Point2D>&> outStripePoints = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<avl::ArcFittingField&> outAlignedFittingField = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outBrightnessProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL,
 atl::Array<avl::Segment2D>& diagScanSegments,
 atl::Array<avl::Rectangle2D>& diagSamplingAreas
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image to fit arc to
→	inFittingField	const ArcFittingField&			Arc fitting field
→	inFittingFieldAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the fitting field to the position of the inspected object
→	inScanCount	int	3 - ∞	10	The number of points that will be searched to estimate the position of the arc
→	inScanWidth	int	1 - ∞	5	The width of each scan field (in pixels)
→	inSamplingParams	const SamplingParams&			Parameters controlling the sampling process
→	inStripeScanParams	const StripeScanParams&		StripeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f MinMagnitude: 5.0f MaxInnerEdgeMagnitude: Nil StripePolarity: Dark MinStripeWidth: 0.0f MaxStripeWidth: Nil)	Parameters controlling the stripe extraction process
→	inStripeSelection	Selection::Type			Selection mode of stripe
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of stripe points not found
→	inFittingMethod	CircleFittingMethod::Type		AlgebraicTaubin	Method used to fit an arc
→	inOutlierSuppression	Optional<MEstimator::Type>		NIL	Selects a method for ignoring incorrectly detected points
←	outArc	Conditional<Arc2D>&			Fitted arc in the middle of found stripe
←	outInnerArc	Conditional<Arc2D>&			Fitted inner arc
←	outOuterArc	Conditional<Arc2D>&			Fitted outer arc
←	outStripes	Optional<Array<Conditional<Stripe1D>>&>		NIL	Found stripes
←	outStripePoints	Optional<Array<Point2D>&>		NIL	Extracted points of middle arc of an image stripe
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual arc points and the corresponding reference arc points
←	outAlignedFittingField	Optional<ArcFittingField&>		NIL	Fitting field used; in the image coordinate system
←	outBrightnessProfiles	Optional<Array<Profile>&>		NIL	Extracted image profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the edge (derivative) operator response
🔍	diagScanSegments	Array<Segment2D>&			Segments along which the scans were run
🔍	diagSamplingAreas	Array<Rectangle2D>&			Areas from which the input image is sampled

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outStripes**, **outStripePoints**, **outDeviationProfile**, **outAlignedFittingField**, **outBrightnessProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

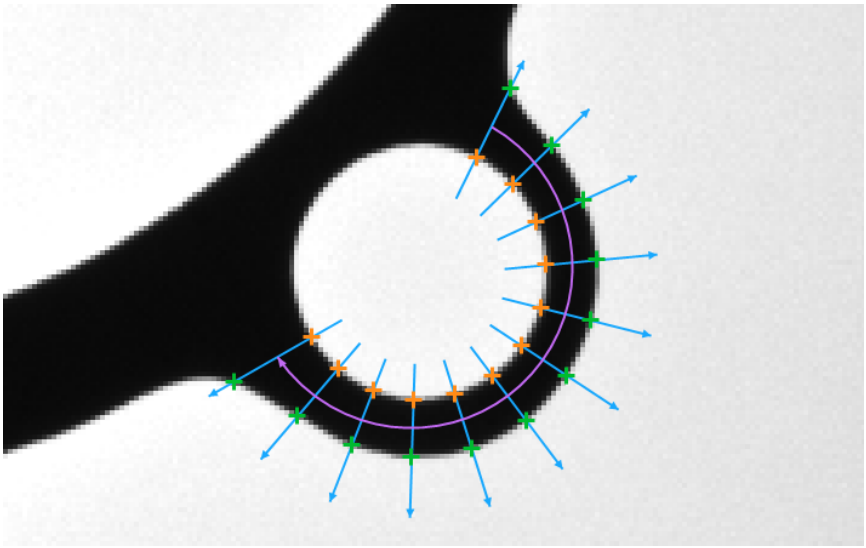
Description

The operation tries to fit a given arc to stripe present in the **inImage** image. Internally, it performs a series of scans with the ScanSingleStripe_Direct filter along **inScanCount** specific scan segments which length is always equal to the **inFittingField** width and cannot be less than 4. The found points are then used to determine the actual position of the arc in the image. Only **inMaxIncompleteness** fraction of these scans may fail. If the fitting according to the given parameters is not possible, **outArc** is set to Nil.

Hints

- Connect an input image to the **inImage** input.
- Define **inStripeScanParams.StripePolarity** to detect a particular stripe type, and only that type.
- Adjust **inStripeScanParams.MinStripeWidth** and **inStripeScanParams.MaxStripeWidth** to the expected thickness of the stripe (in pixels).
- If no or too few stripe points are found, try decreasing **inStripeScanParams.MinMagnitude**.
- If some of the scans may fail, set the **inMaxIncompleteness** input accordingly.
- If some of the scans may produce false results, try different values of the **inOutlierSuppression** input.
- Use the **outStripePoints** outputs to visualize the scanning results.

Examples



Fitting an arc to the dark stripe formed on one side of a hole
*(**inStripeScanParams.Polarity** = Dark).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the Shape Fitting toolset. To read more about this technique, one can refer to the Shape Fitting chapter of our Machine Vision Guide

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- CreateArcFittingMap – Precomputes a data object that is required for fast arc fitting on images.
- FitArcToEdges_Direct – Performs a series of 1D edge detections and finds an arc that best matches the detected points.
- FitArcToRidges_Direct – Performs a series of 1D ridge detections and finds an arc that best matches the detected points.
- FitArcToStripe – Performs a series of 1D stripe detections and finds an arc that best matches the detected points.

FitCircleToEdges

Header: AVL.h
Namespace: avl
Module: MetrologyPro

Performs a series of 1D edge detections and finds a circle that best matches the detected points.

Applications: Precise detection of a circular object or hole, whose rough location is known beforehand.

Syntax

```
void avl::FitCircleToEdges
(
 const avl::Image& inImage,
 const CircleFittingMap& inFittingMap,
 const EdgeScanParams& inEdgeScanParams,
 avl::Selection::Type inEdgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 float inMaxIncompleteness,
 avl::CircleFittingMethod::Type inFittingMethod,
 atl::Optional<avl::MEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Circle2D>& outCircle,
 atl::Optional<atl::Array<atl::Conditional<avl::Edge1D>>&> outEdges = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<atl::Array<avl::Point2D>&> outInliers = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outBrightnessProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image to fit the circle to
→	inFittingMap	const CircleFittingMap&			Input fitting map
→	inEdgeScanParams	const EdgeScanParams&			Parameters controlling the edge extraction process
→	inEdgeSelection	Selection::Type		SelectionBest	Selection mode of edges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of edge points not found
→	inFittingMethod	CircleFittingMethod::Type		AlgebraicTaubin	Method used to fit a circle
→	inOutlierSuppression	Optional<MEstimator::Type>		NIL	Selects a method for ignoring incorrectly detected points
←	outCircle	Conditional<Circle2D>&			Fitted circle or nothing if the fitting fails
←	outEdges	Optional<Array<Conditional<Edge1D>>&>		NIL	Found edges
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual circle points and the corresponding reference circle points
←	outInliers	Optional<Array<Point2D>&>		NIL	Points matching the fitting Circle
←	outBrightnessProfiles	Optional<Array<Profile>&>		NIL	Extracted image profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the edge (derivative) operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outEdges**, **outDeviationProfile**, **outInliers**, **outBrightnessProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

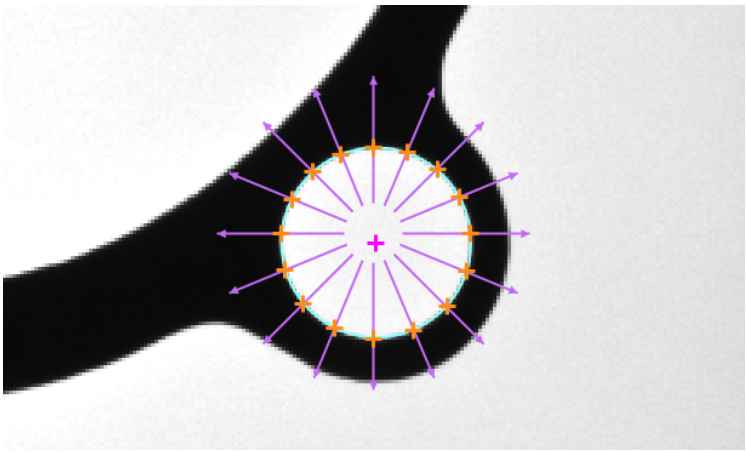
Description

The operation tries to fit a given circle to edges present in the **inImage** image. Internally, it performs a series of scans with the ScanSingleEdge filter using **inFittingMap** previously generated from the object being fitted. The found points are then used to determine the actual position of the circle in the image. Only **inMaxIncompleteness** fraction of these scans may fail. If the fitting according to the given parameters is not possible, **outCircle** is set to Nil.

Hints

- Connect an input image to the **inImage** input.
- Define **inEdgeScanParams.EdgeTransition** to detect a particular edge type, and only that type.
- If no or too few edge points are found, try decreasing **inEdgeScanParams.MinMagnitude**.
- If some of the scans may fail, set the **inMaxIncompleteness** input accordingly.
- If some of the scans may produce false results, try different values of the **inOutlierSuppression** input.
- Use the **outEdges** outputs to visualize the scanning results.

Examples



Fitting a circles to the edges of a hole
*(**inEdgeScanParams.Transition** = BrightToDark).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the Shape Fitting toolset. To read more about this technique, one can refer to the Shape Fitting chapter of our Machine Vision Guide

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- CreateCircleFittingMap – Precomputes a data object that is required for fast circle fitting on images.
- FitCircleToRidges – Performs a series of 1D ridge detections and finds a circle that best matches the detected points.
- FitCircleToStripe – Performs a series 1D stripe detections and finds a circle that best matches the detected points.
- FitCircleToEdges_Direct – Performs a series of 1D edge detections and finds a circle that best matches the detected points.

FitCircleToEdges_Direct

Header: AVL.h
Namespace: avl
Module: MetrologyPro

Performs a series of 1D edge detections and finds a circle that best matches the detected points.

Applications: Precise detection of a circular object or hole, whose rough location is known beforehand.

Syntax

```
void avl::FitCircleToEdges_Direct
(
 const avl::Image& inImage,
 const avl::CircleFittingField& inFittingField,
 atl::Optional<const avl::CoordinateSystem2D&> inFittingFieldAlignment,
 int inScanCount,
 int inScanWidth,
 const avl::SamplingParams& inSamplingParams,
 const avl::EdgeScanParams& inEdgeScanParams,
 avl::Selection::Type inEdgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 float inMaxIncompleteness,
 avl::CircleFittingMethod::Type inFittingMethod,
 atl::Optional<avl::MEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Circle2D>& outCircle,
 atl::Optional<atl::Array<atl::Conditional<avl::Edge1D>>&> outEdges = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<avl::CircleFittingField&> outAlignedFittingField = atl::NIL,
 atl::Optional<atl::Array<avl::Point2D>&> outInliers = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outBrightnessProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL,
 atl::Array<avl::Segment2D>& diagScanSegments,
 atl::Array<avl::Rectangle2D>& diagSamplingAreas
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image to fit the circle to
→	inFittingField	const CircleFittingField&			Circle fitting field
→	inFittingFieldAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the fitting field to the position of the inspected object
→	inScanCount	int	3 - ∞	10	The number of points that will be searched to estimate the position of the circle
→	inScanWidth	int	1 - ∞	5	The width of each scan field (in pixels)
→	inSamplingParams	const SamplingParams&			Parameters controlling the sampling process
→	inEdgeScanParams	const EdgeScanParams&		EdgeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 1.0f MinMagnitude: 5.0f EdgeTransition: BrightToDark)	Parameters controlling the edge extraction process
→	inEdgeSelection	Selection::Type			Selection mode of edges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of edge points not found
→	inFittingMethod	CircleFittingMethod::Type		AlgebraicTaubin	Method used to fit a circle
→	inOutlierSuppression	Optional<MEstimator::Type>		NIL	Selects a method for ignoring incorrectly detected points
←	outCircle	Conditional<Circle2D>&			Fitted circle or nothing if the fitting fails
←	outEdges	Optional<Array<Conditional<Edge1D>>&>		NIL	Found edges
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual circle points and the corresponding reference circle points
←	outAlignedFittingField	Optional<CircleFittingField&>		NIL	Fitting field used; in the image coordinate system
←	outInliers	Optional<Array<Point2D>&>		NIL	Points matching the fitting Circle
←	outBrightnessProfiles	Optional<Array<Profile>&>		NIL	Extracted image profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the edge (derivative) operator response
🔍	diagScanSegments	Array<Segment2D>&			Segments along which the scans were run
🔍	diagSamplingAreas	Array<Rectangle2D>&			Areas from which the input image is sampled

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outEdges**, **outDeviationProfile**, **outAlignedFittingField**, **outInliers**, **outBrightnessProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

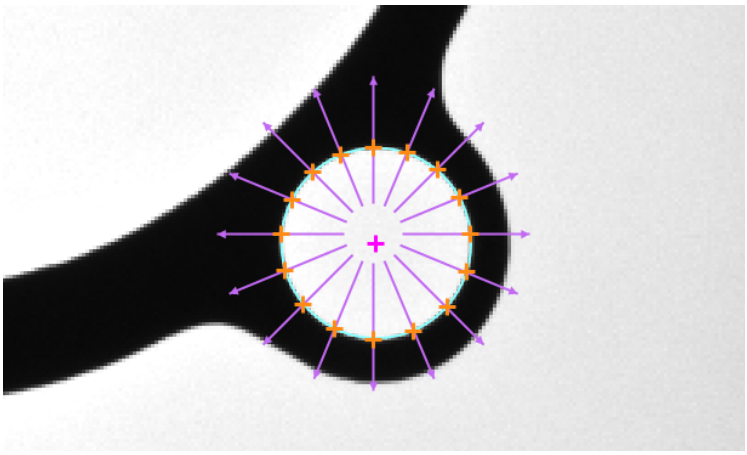
Description

The operation tries to fit a given circle to edges present in the **inImage** image. Internally, it performs a series of scans with the ScanSingleEdge_Direct filter along **inScanCount** specific scan segments which length is always equal to the **inFittingField** width and cannot be less than 4. The found points are then used to determine the actual position of the circle in the image. Only **inMaxIncompleteness** fraction of these scans may fail. If the fitting according to the given parameters is not possible, **outCircle** is set to Nil.

Hints

- Connect an input image to the **inImage** input.
- Define **inEdgeScanParams.EdgeTransition** to detect a particular edge type, and only that type.
- If no or too few edge points are found, try decreasing **inEdgeScanParams.MinMagnitude**.
- If some of the scans may fail, set the **inMaxIncompleteness** input accordingly.
- If some of the scans may produce false results, try different values of the **inOutlierSuppression** input.
- Use the **outEdges** outputs to visualize the scanning results.

Examples



Fitting a circles to the edges of a hole
*(**inEdgeScanParams.Transition** = BrightToDark).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the Shape Fitting toolset. To read more about this technique, one can refer to the Shape Fitting chapter of our Machine Vision Guide

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- CreateCircleFittingMap – Precomputes a data object that is required for fast circle fitting on images.
- FitCircleToRidges_Direct – Performs a series of 1D ridge detections and finds a circle that best matches the detected points.
- FitCircleToStripe_Direct – Performs a series 1D stripe detections and finds a circle that best matches the detected points.
- FitCircleToEdges – Performs a series of 1D edge detections and finds a circle that best matches the detected points.

FitCircleToRidges

Header: AVL.h
Namespace: avl
Module: MetrologyPro

Performs a series of 1D ridge detections and finds a circle that best matches the detected points.

Applications: Precise detection of a thin circular line, whose rough location is known beforehand.

Syntax

```
void avl::FitCircleToRidges
(
 const avl::Image& inImage,
 const avl::CircleFittingMap& inFittingMap,
 const RidgeScanParams& inRidgeScanParams,
 avl::Selection::Type inRidgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 float inMaxIncompleteness,
 avl::CircleFittingMethod::Type inFittingMethod,
 atl::Optional<avl::MEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Circle2D>& outCircle,
 atl::Optional<atl::Array<atl::Conditional<avl::Ridge1D>>&> outRidges = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<atl::Array<avl::Point2D>&> outInliers = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outBrightnessProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image to fit the circle to
→	inFittingMap	const CircleFittingMap&			Input fitting map
→	inRidgeScanParams	const RidgeScanParams&			Parameters controlling the ridge extraction process
→	inRidgeSelection	Selection::Type		Selection::Best	Selection mode of ridges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker ridges can be detected in the vicinity of stronger ridges
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of ridge points not found
→	inFittingMethod	CircleFittingMethod::Type		AlgebraicTaubin	Method used to fit a circle
→	inOutlierSuppression	Optional<MEstimator::Type>		NIL	Selects a method for ignoring incorrectly detected points
←	outCircle	Conditional<Circle2D>&			Fitted circle or nothing if the fitting fails
←	outRidges	Optional<Array<Conditional<Ridge1D>>&>		NIL	Found ridges
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual circle points and the corresponding reference circle points
←	outInliers	Optional<Array<Point2D>&>		NIL	Points matching the fitting Circle
←	outBrightnessProfiles	Optional<Array<Profile>&>		NIL	Extracted image profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the ridge operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outRidges**, **outDeviationProfile**, **outInliers**, **outBrightnessProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

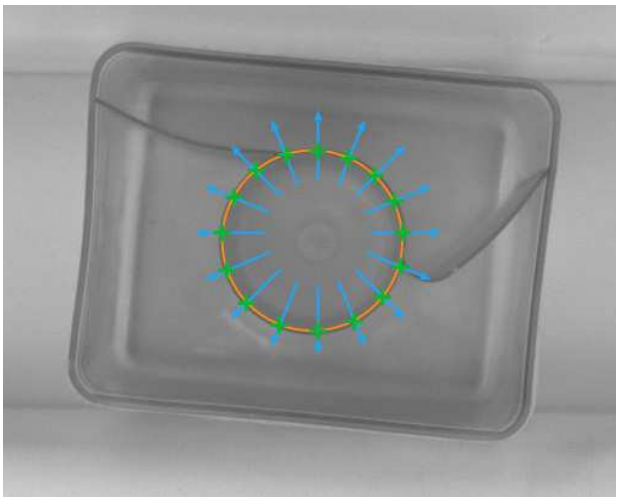
Description

The operation tries to fit a given circle to ridges present in the **inImage** image. Internally, it performs a series of scans with the ScanSingleRidge filter using **inFittingMap** previously generated from the object being fitted. The found points are then used to determine the actual position of the circle in the image. Only **inMaxIncompleteness** fraction of these scans may fail. If the fitting according to the given parameters is not possible, **outCircle** is set to Nil.

Hints

- Connect an input image to the **inImage** input.
- Define **inRidgeScanParams.RidgePolarity** to detect a particular ridge type, and only that type.
- Adjust **inRidgeScanParams.RidgeWidth** to the expected thickness of the ridge (in pixels).
- If no or too few ridge points are found, try decreasing **inRidgeScanParams.MinMagnitude**.
- If some of the scans may fail, set the **inMaxIncompleteness** input accordingly.
- If some of the scans may produce false results, try different values of the **inOutlierSuppression** input.
- Use the **outRidges** outputs to visualize the scanning results.

Examples



Fitting a circle to a wall of a plastic capsule
*(**inRidgeScanParams.Polarity** = Dark, **inRidgeScanParams.RidgeWidth** = 3).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the Shape Fitting toolset. To read more about this technique, one can refer to the Shape Fitting chapter of our Machine Vision Guide

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- CreateCircleFittingMap – Precomputes a data object that is required for fast circle fitting on images.
- FitCircleToEdges – Performs a series of 1D edge detections and finds a circle that best matches the detected points.
- FitCircleToStripe – Performs a series 1D stripe detections and finds a circle that best matches the detected points.
- FitCircleToRidges_Direct – Performs a series of 1D ridge detections and finds a circle that best matches the detected points.

FitCircleToRidges_Direct

Header: AVL.h
Namespace: avl
Module: MetrologyPro

Performs a series of 1D ridge detections and finds a circle that best matches the detected points.

Applications: Precise detection of a thin circular line, whose rough location is known beforehand.

Syntax

```
void avl::FitCircleToRidges_Direct
(
 const avl::Image& inImage,
 const avl::CircleFittingField& inFittingField,
 atl::Optional<const avl::CoordinateSystem2D&> inFittingFieldAlignment,
 int inScanCount,
 int inScanWidth,
 const avl::SamplingParams& inSamplingParams,
 const avl::RidgeScanParams& inRidgeScanParams,
 avl::Selection::Type inRidgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 float inMaxIncompleteness,
 avl::CircleFittingMethod::Type inFittingMethod,
 atl::Optional<avl::MEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Circle2D>& outCircle,
 atl::Optional<atl::Array<atl::Conditional<avl::Ridge1D>>&> outRidges = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<avl::CircleFittingField&> outAlignedFittingField = atl::NIL,
 atl::Optional<atl::Array<avl::Point2D>&> outInliers = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outBrightnessProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL,
 atl::Array<avl::Segment2D>& diagScanSegments,
 atl::Array<avl::Rectangle2D>& diagSamplingAreas
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image to fit the circle to
→	inFittingField	const CircleFittingField&			Circle fitting field
→	inFittingFieldAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the fitting field to the position of the inspected object
→	inScanCount	int	3 - ∞	10	The number of points that will be searched to estimate the position of the circle
→	inScanWidth	int	1 - ∞	5	The width of each scan field (in pixels)
→	inSamplingParams	const SamplingParams&			Parameters controlling the sampling process
→	inRidgeScanParams	const RidgeScanParams&		RidgeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 1.0f RidgeWidth: 5 RidgeMargin: 2 MinMagnitude: 5.0f RidgePolarity: Dark)	Parameters controlling the ridge extraction process
→	inRidgeSelection	Selection::Type			Selection mode of ridges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker ridges can be detected in the vicinity of stronger ridges
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of ridge points not found
→	inFittingMethod	CircleFittingMethod::Type		AlgebraicTaubin	Method used to fit a circle
→	inOutlierSuppression	Optional<MEstimator::Type>		NIL	Selects a method for ignoring incorrectly detected points
←	outCircle	Conditional<Circle2D>&			Fitted circle or nothing if the fitting fails
←	outRidges	Optional<Array<Conditional<Ridge1D>>&>		NIL	Found ridges
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual circle points and the corresponding reference circle points
←	outAlignedFittingField	Optional<CircleFittingField&>		NIL	Fitting field used; in the image coordinate system
←	outInliers	Optional<Array<Point2D>&>		NIL	Points matching the fitting Circle
←	outBrightnessProfiles	Optional<Array<Profile>&>		NIL	Extracted image profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the ridge operator response
🔍	diagScanSegments	Array<Segment2D>&			Segments along which the scans were run
🔍	diagSamplingAreas	Array<Rectangle2D>&			Areas from which the input image is sampled

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outRidges**, **outDeviationProfile**, **outAlignedFittingField**, **outInliers**, **outBrightnessProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

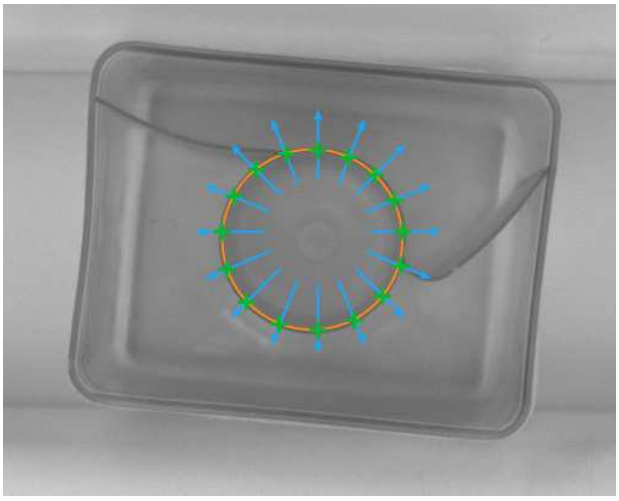
Description

The operation tries to fit a given circle to ridges present in the **inImage** image. Internally, it performs a series of scans with the ScanSingleRidge_Direct filter along **inScanCount** specific scan segments which length is always equal to the **inFittingField** width and cannot be less than 4. The found points are then used to determine the actual position of the circle in the image. Only **inMaxIncompleteness** fraction of these scans may fail. If the fitting according to the given parameters is not possible, **outCircle** is set to Nil.

Hints

- Connect an input image to the **inImage** input.
- Define **inRidgeScanParams.RidgePolarity** to detect a particular ridge type, and only that type.
- Adjust **inRidgeScanParams.RidgeWidth** to the expected thickness of the ridge (in pixels).
- If no or too few ridge points are found, try decreasing **inRidgeScanParams.MinMagnitude**.
- If some of the scans may fail, set the **inMaxIncompleteness** input accordingly.
- If some of the scans may produce false results, try different values of the **inOutlierSuppression** input.
- Use the **outRidges** outputs to visualize the scanning results.

Examples



Fitting a circle to a wall of a plastic capsule
*(**inRidgeScanParams.Polarity** = Dark, **inRidgeScanParams.RidgeWidth** = 3).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the Shape Fitting toolset. To read more about this technique, one can refer to the Shape Fitting chapter of our Machine Vision Guide

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- CreateCircleFittingMap – Precomputes a data object that is required for fast circle fitting on images.
- FitCircleToEdges_Direct – Performs a series of 1D edge detections and finds a circle that best matches the detected points.
- FitCircleToStripe_Direct – Performs a series 1D stripe detections and finds a circle that best matches the detected points.
- FitCircleToRidges – Performs a series of 1D ridge detections and finds a circle that best matches the detected points.

FitCircleToStripe

Header: AVL.h
Namespace: avl
Module: MetrologyPro

Performs a series 1D stripe detections and finds a circle that best matches the detected points.

Applications: Precise detection of ring-shaped objects, whose rough location is known beforehand.

Syntax

```
void avl::FitCircleToStripe
(
 const avl::Image& inImage,
 const avl::CircleFittingMap& inFittingMap,
 const StripeScanParams& inStripeScanParams,
 avl::Selection::Type inStripeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 float inMaxIncompleteness,
 avl::CircleFittingMethod::Type inFittingMethod,
 atl::Optional<MEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Circle2D>& outCircle,
 atl::Conditional<avl::Circle2D>& outInnerCircle,
 atl::Conditional<avl::Circle2D>& outOuterCircle,
 atl::Optional<atl::Array<atl::Conditional<avl::Stripe1D>>&> outStripes = atl::NIL,
 atl::Optional<atl::Array<avl::Point2D>&> outStripePoints = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outBrightnessProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image to fit the circle to
→	inFittingMap	const CircleFittingMap&			Input fitting map
→	inStripeScanParams	const StripeScanParams&			Parameters controlling the stripe extraction process
→	inStripeSelection	Selection::Type		Selection::Best	Selection mode of stripe
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of stripe points not found
→	inFittingMethod	CircleFittingMethod::Type		AlgebraicTaubin	Method used to fit a circle
→	inOutlierSuppression	Optional<MEstimator::Type>		NIL	Selects a method for ignoring incorrectly detected points
←	outCircle	Conditional<Circle2D>&			Fitted circle in the middle of found stripe
←	outInnerCircle	Conditional<Circle2D>&			Fitted inner circle
←	outOuterCircle	Conditional<Circle2D>&			Fitted outer circle
←	outStripes	Optional<Array<Conditional<Stripe1D>>&>		NIL	Found stripes
←	outStripePoints	Optional<Array<Point2D>&>		NIL	Extracted points of middle circle of an image stripe
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual circle points and the corresponding reference circle points
←	outBrightnessProfiles	Optional<Array<Profile>&>		NIL	Extracted image profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the edge (derivative) operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outStripes**, **outStripePoints**, **outDeviationProfile**, **outBrightnessProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

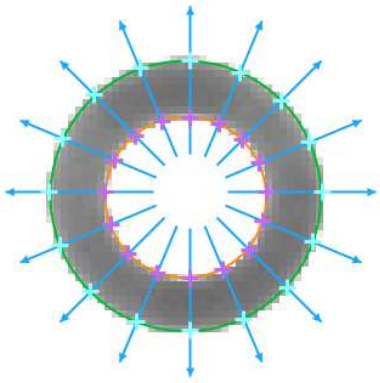
Description

The operation tries to fit a given circle to stripe present in the **inImage** image. Internally, it performs a series of scans with the ScanSingleStripe filter using **inFittingMap** previously generated from the object being fitted. The found points are then used to determine the actual position of the circle in the image. Only **inMaxIncompleteness** fraction of these scans may fail. If the fitting according to the given parameters is not possible, **outCircle** is set to Nil.

Hints

- Connect an input image to the **inImage** input.
- Define **inStripeScanParams.StripePolarity** to detect a particular stripe type, and only that type.
- Adjust **inStripeScanParams.MinStripeWidth** and **inStripeScanParams.MaxStripeWidth** to the expected thickness of the stripe (in pixels).
- If no or too few stripe points are found, try decreasing **inStripeScanParams.MinMagnitude**.
- If some of the scans may fail, set the **inMaxIncompleteness** input accordingly.
- If some of the scans may produce false results, try different values of the **inOutlierSuppression** input.
- Use the **outStripePoints** outputs to visualize the scanning results.

Examples



*Fitting a circle to the dark circular stripe of a washer (**inStripeScanParams.Polarity** = Dark).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the Shape Fitting toolset. To read more about this technique, one can refer to the Shape Fitting chapter of our Machine Vision Guide

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- CreateCircleFittingMap – Precomputes a data object that is required for fast circle fitting on images.
- FitCircleToEdges – Performs a series of 1D edge detections and finds a circle that best matches the detected points.
- FitCircleToRidges – Performs a series of 1D ridge detections and finds a circle that best matches the detected points.
- FitCircleToStripe_Direct – Performs a series 1D stripe detections and finds a circle that best matches the detected points.

FitCircleToStripe_Direct

Header: AVL.h
Namespace: avl
Module: MetrologyPro

Performs a series 1D stripe detections and finds a circle that best matches the detected points.

Applications: Precise detection of ring-shaped objects, whose rough location is known beforehand.

Syntax

```
void avl::FitCircleToStripe_Direct
(
 const avl::Image& inImage,
 const avl::CircleFittingField& inFittingField,
 atl::Optional<const avl::CoordinateSystem2D&> inFittingFieldAlignment,
 int inScanCount,
 int inScanWidth,
 const avl::SamplingParams& inSamplingParams,
 const avl::StripeScanParams& inStripeScanParams,
 avl::Selection::Type inStripeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 float inMaxIncompleteness,
 avl::CircleFittingMethod::Type inFittingMethod,
 atl::Optional<avl::MEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Circle2D>& outCircle,
 atl::Conditional<avl::Circle2D>& outInnerCircle,
 atl::Conditional<avl::Circle2D>& outOuterCircle,
 atl::Optional<atl::Array<atl::Conditional<avl::Stripe1D>>&> outStripes = atl::NIL,
 atl::Optional<atl::Array<avl::Point2D>&> outStripePoints = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<avl::CircleFittingField&> outAlignedFittingField = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outBrightnessProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL,
 atl::Array<avl::Segment2D>& diagScanSegments,
 atl::Array<avl::Rectangle2D>& diagSamplingAreas
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image to fit the circle to
→	inFittingField	const CircleFittingField&			Circle fitting field
→	inFittingFieldAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the fitting field to the position of the inspected object
→	inScanCount	int	3 - ∞	10	The number of points that will be searched to estimate the position of the circle
→	inScanWidth	int	1 - ∞	5	The width of each scan field (in pixels)
→	inSamplingParams	const SamplingParams&			Parameters controlling the sampling process
→	inStripeScanParams	const StripeScanParams&		StripeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f MinMagnitude: 5.0f MaxInnerEdgeMagnitude: Nil StripePolarity: Dark MinStripeWidth: 0.0f MaxStripeWidth: Nil)	Parameters controlling the stripe extraction process
→	inStripeSelection	Selection::Type			Selection mode of stripe
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of stripe points not found
→	inFittingMethod	CircleFittingMethod::Type		AlgebraicTaubin	Method used to fit a circle
→	inOutlierSuppression	Optional<MEstimator::Type>		NIL	Selects a method for ignoring incorrectly detected points
←	outCircle	Conditional<Circle2D>&			Fitted circle in the middle of found stripe
←	outInnerCircle	Conditional<Circle2D>&			Fitted inner circle
←	outOuterCircle	Conditional<Circle2D>&			Fitted outer circle
←	outStripes	Optional<Array<Conditional<Stripe1D>>&>		NIL	Found stripes
←	outStripePoints	Optional<Array<Point2D>&>		NIL	Extracted points of middle circle of an image stripe
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual circle points and the corresponding reference circle points
←	outAlignedFittingField	Optional<CircleFittingField&>		NIL	Fitting field used; in the image coordinate system
←	outBrightnessProfiles	Optional<Array<Profile>&>		NIL	Extracted image profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the edge (derivative) operator response
🔍	diagScanSegments	Array<Segment2D>&			Segments along which the scans were run
🔍	diagSamplingAreas	Array<Rectangle2D>&			Areas from which the input image is sampled

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outStripes**, **outStripePoints**, **outDeviationProfile**, **outAlignedFittingField**, **outBrightnessProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

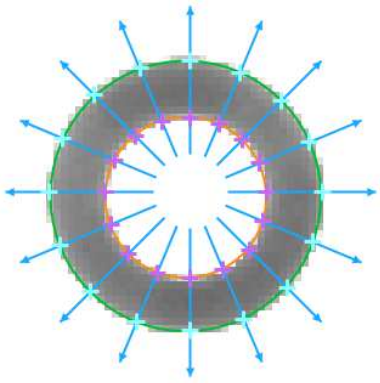
Description

The operation tries to fit a given circle to stripe present in the **inImage** image. Internally, it performs a series of scans with the ScanSingleStripe_Direct filter along **inScanCount** specific scan segments which length is always equal to the **inFittingField** width and cannot be less than 4. The found points are then used to determine the actual position of the circle in the image. Only **inMaxIncompleteness** fraction of these scans may fail. If the fitting according to the given parameters is not possible, **outCircle** is set to Nil.

Hints

- Connect an input image to the **inImage** input.
- Define **inStripeScanParams.StripePolarity** to detect a particular stripe type, and only that type.
- Adjust **inStripeScanParams.MinStripeWidth** and **inStripeScanParams.MaxStripeWidth** to the expected thickness of the stripe (in pixels).
- If no or too few stripe points are found, try decreasing **inStripeScanParams.MinMagnitude**.
- If some of the scans may fail, set the **inMaxIncompleteness** input accordingly.
- If some of the scans may produce false results, try different values of the **inOutlierSuppression** input.
- Use the **outStripePoints** outputs to visualize the scanning results.

Examples



Fitting a circle to the dark circular stripe of a washer
*(**inStripeScanParams.Polarity** = Dark).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the Shape Fitting toolset. To read more about this technique, one can refer to the Shape Fitting chapter of our Machine Vision Guide

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- CreateCircleFittingMap – Precomputes a data object that is required for fast circle fitting on images.
- FitCircleToEdges_Direct – Performs a series of 1D edge detections and finds a circle that best matches the detected points.
- FitCircleToRidges_Direct – Performs a series of 1D ridge detections and finds a circle that best matches the detected points.
- FitCircleToStripe – Performs a series 1D stripe detections and finds a circle that best matches the detected points.

FitContour_Adaptive_Experimental

Header: AVL.h
Namespace: avl

Fits the given contour to the edges of the input image.

Syntax

```
void avl::FitContour_Adaptive_Experimental
(
 const avl::Image& inImage,
 const avl::Path& inExpectedContour,
 int inMaxDeviation,
 float inMaxSlant,
 int inStep,
 avl::EdgeTransition::Type inEdgeTransition,
 int inMinEdgeMagnitude,
 int inMaxEdgeMagnitude,
 float inGammaFactor,
 avl::InterpolationMethod::Type inInterpolationMethod,
 float inSmoothingStdDev,
 avl::Path& outContour,
 avl::Image& diagEdgeImage,
 avl::Path& diagEdgeContour
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inExpectedContour	const Path&			
→	inMaxDeviation	int	1 - ∞	15	Maximal distance between the fitted and the expected contours
→	inMaxSlant	float	1.0 - 60.0	45.0f	Maximal angle between corresponding tangents of the fitted and the expected contours
→	inStep	int	1 - ∞	6	Distance between consecutive fitted points measured along the expected contour
→	inEdgeTransition	EdgeTransition::Type			
→	inMinEdgeMagnitude	int	0 - 255	0	
→	inMaxEdgeMagnitude	int	0 - 255	255	
→	inGammaFactor	float	0.01 - 10.0	1.0f	
→	inInterpolationMethod	InterpolationMethod::Type			
→	inSmoothingStdDev	float	0.0 - ∞	0.0f	
←	outContour	Path&			
🔍	diagEdgeImage	Image&			
🔍	diagEdgeContour	Path&			

FitPathToEdges

Header: AVL.h
Namespace: avl
Module: MetrologyPro

Performs a series of 1D edge detections and creates a path from the detected points.

Applications: Tracing of an object contour, whose rough location and shape is known beforehand.

Syntax

```
void avl::FitPathToEdges
(
 const avl::Image& inImage,
 const PathFittingMap& inFittingMap,
 const EdgeScanParams& inEdgeScanParams,
 avl::Selection::Type inEdgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxInterpolationLength,
 atl::Optional<float> inMaxDeviationDelta,
 float inMaxIncompleteness,
 atl::Conditional<avl::Path>& outPath,
 atl::Optional<atl::Array<atl::Conditional<avl::Edge1D>>&> outEdges = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outBrightnessProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image to fit the path to
→	inFittingMap	const PathFittingMap&			Input fitting map
→	inEdgeScanParams	const EdgeScanParams&			Parameters controlling the edge extraction process
→	inEdgeSelection	Selection::Type		Selection::Best	Selection mode of edges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxInterpolationLength	Optional<int>	0 - ∞	NIL	Maximal number of consecutive points not found
→	inMaxDeviationDelta	Optional<float>	0.0 - ∞	NIL	Maximal difference between deviations of consecutive path points
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of edge points not found
←	outPath	Conditional<Path>&			Fitted path or nothing if the fitting failed
←	outEdges	Optional<Array<Conditional<Edge1D>>&>		NIL	Found edges
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual path points and the corresponding reference path points
←	outBrightnessProfiles	Optional<Array<Profile>&>		NIL	Extracted image profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the edge (derivative) operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outEdges**, **outDeviationProfile**, **outBrightnessProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

Description

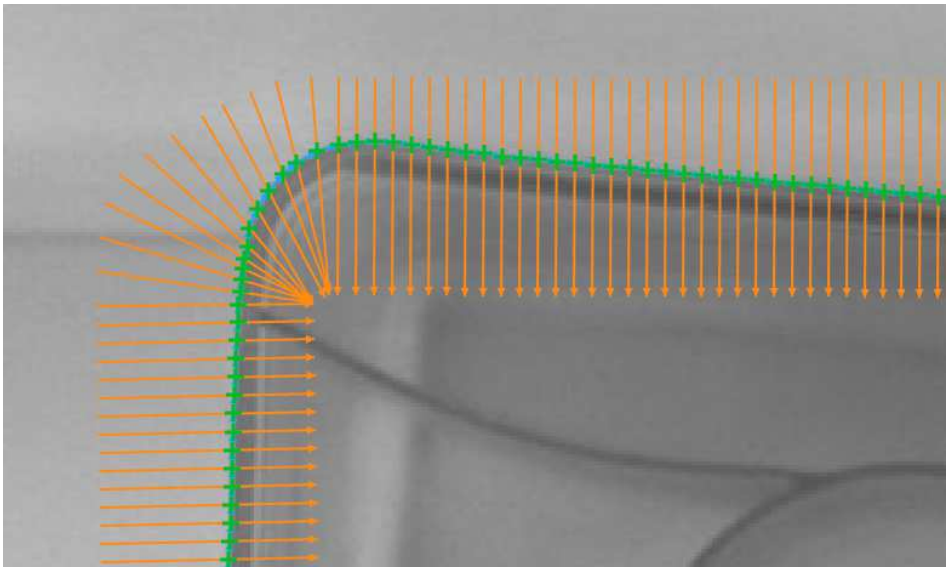
The operation tries to fit a given path to edges present in the **inImage** image. Internally, it performs a series of scans with the ScanSingleEdge filter using **inFittingMap** previously generated from the object being fitted. The found points are then used to determine the actual position of the path in the image. Only **inMaxIncompleteness** fraction of these scans may fail. If the fitting according to the given parameters is not possible, **outPath** is set to Nil.

There are also another parameters that control the path fitting process. The **inMaxDeviationDelta** parameter defines the maximal allowed difference between deviations of consecutive points from the input path points. If some of the scans fail or if some of found points are classified to be wrong according to another control parameters, output path points corresponding to them are interpolated depending on points in their nearest vicinity. No more than **inMaxInterpolationLength** consecutive points can be interpolated. The exception to this behavior are points which were not found on both ends of the input path. Those are not part of the result at all.

Hints

- Connect an input image to the **inImage** input.
- Define **inEdgeScanParams.EdgeTransition** to detect a particular edge type, and only that type.
- If no or too few edge points are found, try decreasing **inEdgeScanParams.MinMagnitude**.
- If some of the scans may fail, set the **inMaxIncompleteness** input accordingly.
- Use the **outEdges** outputs to visualize the scanning results.

Examples



Fitting a path to the edges of a plastic capsule
*(**inEdgeSelection** = First, **inEdgeScanParams.EdgeTransition** = BrightToDark).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the Shape Fitting toolset. To read more about this technique, one can refer to the Shape Fitting chapter of our Machine Vision Guide

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- CreatePathFittingMap – Precomputes a data object that is required for fast path fitting on images.
- FitPathToRidges – Performs a series of 1D ridge detections and creates a path from the detected points.
- FitPathToStripe – Performs a series of 1D stripe detections and creates a path from the detected points.
- FitPathToEdges_Direct – Performs a series of 1D edge detections and creates a path from the detected points.

FitPathToEdges_Direct

Header: AVL.h
Namespace: avl
Module: MetrologyPro

Performs a series of 1D edge detections and creates a path from the detected points.

Applications: Tracing of an object contour, whose rough location and shape is known beforehand.

Syntax

```
void avl::FitPathToEdges_Direct
(
 const avl::Image& inImage,
 const avl::PathFittingField& inFittingField,
 atl::Optional<const avl::CoordinateSystem2D&> inFittingFieldAlignment,
 atl::Optional<float> inScanStep,
 int inScanWidth,
 const avl::SamplingParams& inSamplingParams,
 const avl::EdgeScanParams& inEdgeScanParams,
 avl::Selection::Type inEdgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxInterpolationLength,
 atl::Optional<float> inMaxDeviationDelta,
 float inMaxIncompleteness,
 atl::Conditional<avl::Path>& outPath,
 atl::Optional<atl::Array<atl::Conditional<avl::Edge1D>>&> outEdges = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<avl::PathFittingField&> outAlignedFittingField = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outBrightnessProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL,
 atl::Array<avl::Segment2D>& diagScanSegments,
 atl::Array<avl::Rectangle2D>& diagSamplingAreas
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image to fit the path to
→	inFittingField	const PathFittingField&			Path fitting field
→	inFittingFieldAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the fitting field to the position of the inspected object
→	inScanStep	Optional<float>	0.0 - ∞	5.0f	Optional implicit conversion of the input path to an equidistant one
→	inScanWidth	int	1 - ∞	5	The width of each scan field (in pixels)
→	inSamplingParams	const SamplingParams&			Parameters controlling the sampling process
→	inEdgeScanParams	const EdgeScanParams&		EdgeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f MinMagnitude: 5.0f EdgeTransition: BrightToDark)	Parameters controlling the edge extraction process
→	inEdgeSelection	Selection::Type			Selection mode of edges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxInterpolationLength	Optional<int>	0 - ∞	1	Maximal number of consecutive points not found
→	inMaxDeviationDelta	Optional<float>	0.0 - ∞	NIL	Maximal difference between deviations of consecutive path points
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of edge points not found
←	outPath	Conditional<Path>&			Fitted path or nothing if the fitting failed
←	outEdges	Optional<Array<Conditional<Edge1D>>&>		NIL	Found edges
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual path points and the corresponding reference path points
←	outAlignedFittingField	Optional<PathFittingField&>		NIL	Fitting field used; in the image coordinate system
←	outBrightnessProfiles	Optional<Array<Profile>&>		NIL	Extracted image profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the edge (derivative) operator response
🔍	diagScanSegments	Array<Segment2D>&			Segments along which the scans were run
🔍	diagSamplingAreas	Array<Rectangle2D>&			Areas from which the input image is sampled

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outEdges**, **outDeviationProfile**, **outAlignedFittingField**, **outBrightnessProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

Description

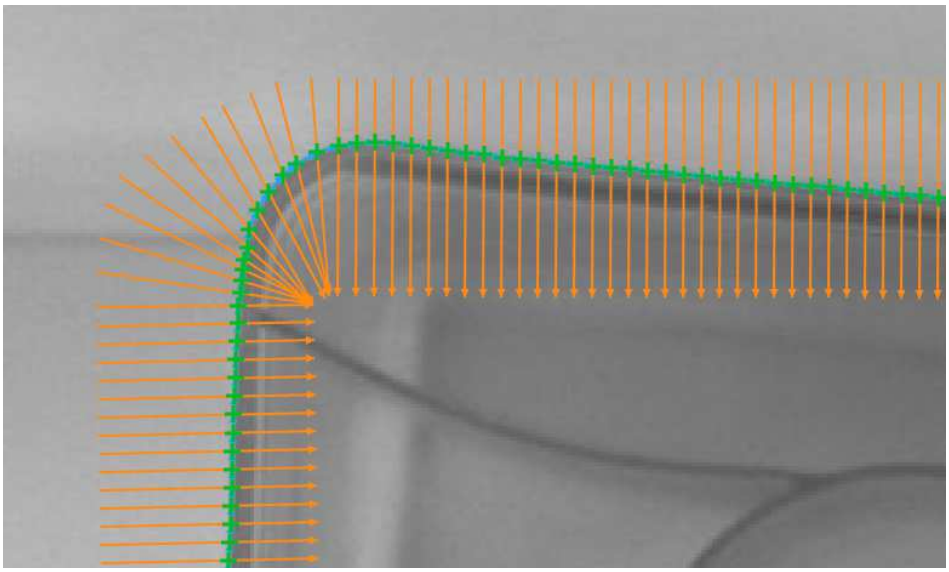
The operation tries to fit a given path to edges present in the **inImage** image. Internally, it performs a series of scans with the ScanSingleEdge_Direct filter along a number of specific scan segments which length is always equal to the **inFittingField** width and cannot be less than 4. The found points are then used to determine the actual position of the path in the image. Only **inMaxIncompleteness** fraction of these scans may fail. If the fitting according to the given parameters is not possible, **outPath** is set to Nil.

There are also another parameters that control the path fitting process. The **inMaxDeviationDelta** parameter defines the maximal allowed difference between deviations of consecutive points from the input path points. If some of the scans fail or if some of found points are classified to be wrong according to another control parameters, output path points corresponding to them are interpolated depending on points in their nearest vicinity. No more than **inMaxInterpolationLength** consecutive points can be interpolated. The exception to this behavior are points which were not found on both ends of the input path. Those are not part of the result at all.

Hints

- Connect an input image to the **inImage** input.
- Define **inEdgeScanParams.EdgeTransition** to detect a particular edge type, and only that type.
- If no or too few edge points are found, try decreasing **inEdgeScanParams.MinMagnitude**.
- If some of the scans may fail, set the **inMaxIncompleteness** input accordingly.
- Use the **outEdges** outputs to visualize the scanning results.

Examples



Fitting a path to the edges of a plastic capsule
*(**inEdgeSelection** = First, **inEdgeScanParams.EdgeTransition** = BrightToDark).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the Shape Fitting toolset. To read more about this technique, one can refer to the Shape Fitting chapter of our Machine Vision Guide

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- CreatePathFittingMap – Precomputes a data object that is required for fast path fitting on images.
- FitPathToRidges_Direct – Performs a series of 1D ridge detections and creates a path from the detected points.
- FitPathToStripe_Direct – Performs a series of 1D stripe detections and creates a path from the detected points.
- FitPathToEdges – Performs a series of 1D edge detections and creates a path from the detected points.

FitPathToRidges

Header: AVL.h
Namespace: avl
Module: MetrologyPro

Performs a series of 1D ridge detections and creates a path from the detected points.

Applications: Tracing of a thin line, whose rough location and shape is known beforehand.

Syntax

```
void avl::FitPathToRidges
(
 const avl::Image& inImage,
 const avl::PathFittingMap& inFittingMap,
 const RidgeScanParams& inRidgeScanParams,
 avl::Selection::Type inRidgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxInterpolationLength,
 atl::Optional<float> inMaxDeviationDelta,
 float inMaxIncompleteness,
 atl::Conditional<avl::Path>& outPath,
 atl::Optional<atl::Array<atl::Conditional<avl::Ridge1D>>&> outRidges = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outBrightnessProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image to fit the path to
→	inFittingMap	const PathFittingMap&			Input fitting map
→	inRidgeScanParams	const RidgeScanParams&			Parameters controlling the ridge extraction process
→	inRidgeSelection	Selection::Type		Selection::Best	Selection mode of ridges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker ridges can be detected in the vicinity of stronger ridges
→	inMaxInterpolationLength	Optional<int>	0 - ∞	NIL	Maximal number of consecutive points not found
→	inMaxDeviationDelta	Optional<float>	0.0 - ∞	NIL	Maximal difference between deviations of consecutive path points
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of ridge points not found
←	outPath	Conditional<Path>&			Fitted path or nothing if the fitting failed
←	outRidges	Optional<Array<Conditional<Ridge1D>>&>		NIL	Found ridges
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual path points and the corresponding reference path points
←	outBrightnessProfiles	Optional<Array<Profile>&>		NIL	Extracted image profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the ridge operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outRidges**, **outDeviationProfile**, **outBrightnessProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

Description

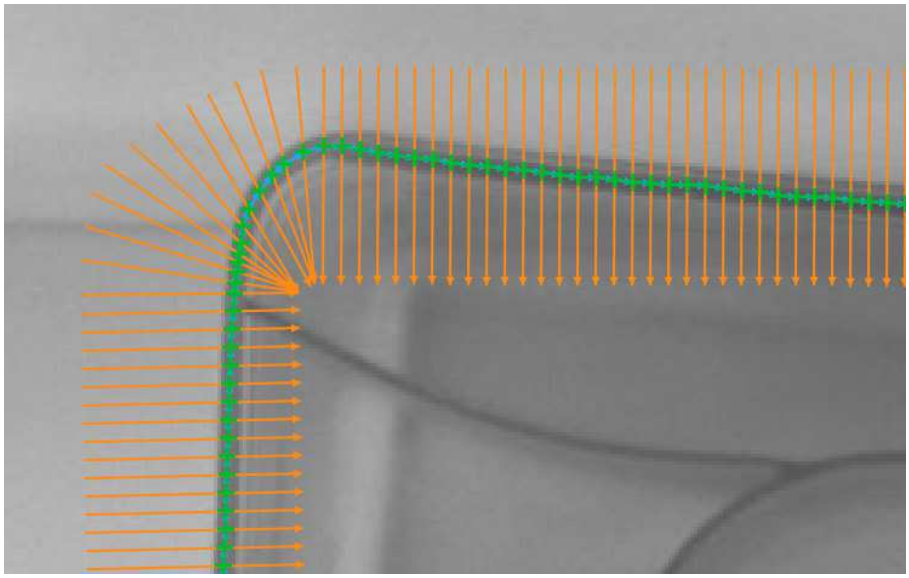
The operation tries to fit a given path to ridges present in the **inImage** image. Internally, it performs a series of scans with the ScanSingleRidge filter using **inFittingMap** previously generated from the object being fitted. The found points are then used to determine the actual position of the path in the image. Only **inMaxIncompleteness** fraction of these scans may fail. If the fitting according to the given parameters is not possible, **outPath** is set to Nil.

There are also another parameters that control the path fitting process. The **inMaxDeviationDelta** parameter defines the maximal allowed difference between deviations of consecutive points from the input path points. If some of the scans fail or if some of found points are classified to be wrong according to another control parameters, output path points corresponding to them are interpolated depending on points in their nearest vicinity. No more than **inMaxInterpolationLength** consecutive points can be interpolated. The exception to this behavior are points which were not found on both ends of the input path. Those are not part of the result at all.

Hints

- Connect an input image to the **inImage** input.
- Define **inRidgeScanParams.RidgePolarity** to detect a particular ridge type, and only that type.
- Adjust **inRidgeScanParams.RidgeWidth** to the expected thickness of the ridge (in pixels).
- If no or too few ridge points are found, try decreasing **inRidgeScanParams.MinMagnitude**.
- If some of the scans may fail, set the **inMaxIncompleteness** input accordingly.
- Use the **outRidges** outputs to visualize the scanning results.

Examples



*Fitting a path to the walls of a plastic capsule (**inRidgeScanParams.RidgePolarity** = Dark, **inRidgeScanParams.RidgeWidth** = 6, **inRidgeScanParams.RidgeOperator** = ArithmeticMean).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the Shape Fitting toolset. To read more about this technique, one can refer to the Shape Fitting chapter of our Machine Vision Guide

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- CreatePathFittingMap – Precomputes a data object that is required for fast path fitting on images.
- FitPathToEdges – Performs a series of 1D edge detections and creates a path from the detected points.
- FitPathToStripe – Performs a series of 1D stripe detections and creates a path from the detected points.
- FitPathToRidges_Direct – Performs a series of 1D ridge detections and creates a path from the detected points.

FitPathToRidges_Direct

Header: AVL.h
Namespace: avl
Module: MetrologyPro

Performs a series of 1D ridge detections and creates a path from the detected points.

Applications: Tracing of a thin line, whose rough location and shape is known beforehand.

Syntax

```
void avl::FitPathToRidges_Direct
(
 const avl::Image& inImage,
 const avl::PathFittingField& inFittingField,
 atl::Optional<const avl::CoordinateSystem2D&> inFittingFieldAlignment,
 atl::Optional<float> inScanStep,
 int inScanWidth,
 const avl::SamplingParams& inSamplingParams,
 const avl::RidgeScanParams& inRidgeScanParams,
 avl::Selection::Type inRidgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxInterpolationLength,
 atl::Optional<float> inMaxDeviationDelta,
 float inMaxIncompleteness,
 atl::Conditional<avl::Path>& outPath,
 atl::Optional<atl::Array<atl::Conditional<avl::Ridge1D>>&> outRidges = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<avl::PathFittingField&> outAlignedFittingField = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outBrightnessProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL,
 atl::Array<avl::Segment2D>& diagScanSegments,
 atl::Array<avl::Rectangle2D>& diagSamplingAreas
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image to fit the path to
→	inFittingField	const PathFittingField&			Path fitting field
→	inFittingFieldAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the fitting field to the position of the inspected object
→	inScanStep	Optional<float>	0.0 - ∞	5.0f	Optional implicit conversion of the input path to an equidistant one
→	inScanWidth	int	1 - ∞	5	The width of each scan field (in pixels)
→	inSamplingParams	const SamplingParams&			Parameters controlling the sampling process
→	inRidgeScanParams	const RidgeScanParams&		RidgeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f RidgeWidth: 5 RidgeMargin: 2 MinMagnitude: 5.0f RidgePolarity: Dark)	Parameters controlling the ridge extraction process
→	inRidgeSelection	Selection::Type			Selection mode of ridges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker ridges can be detected in the vicinity of stronger ridges
→	inMaxInterpolationLength	Optional<int>	0 - ∞	1	Maximal number of consecutive points not found
→	inMaxDeviationDelta	Optional<float>	0.0 - ∞	NIL	Maximal difference between deviations of consecutive path points
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of ridge points not found
←	outPath	Conditional<Path>&			Fitted path or nothing if the fitting failed
←	outRidges	Optional<Array<Conditional<Ridge1D>>&>		NIL	Found ridges
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual path points and the corresponding reference path points
←	outAlignedFittingField	Optional<PathFittingField&>		NIL	Fitting field used; in the image coordinate system
←	outBrightnessProfiles	Optional<Array<Profile>&>		NIL	Extracted image profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the ridge operator response
🔍	diagScanSegments	Array<Segment2D>&			Segments along which the scans were run
🔍	diagSamplingAreas	Array<Rectangle2D>&			Areas from which the input image is sampled

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outRidges**, **outDeviationProfile**, **outAlignedFittingField**, **outBrightnessProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

Description

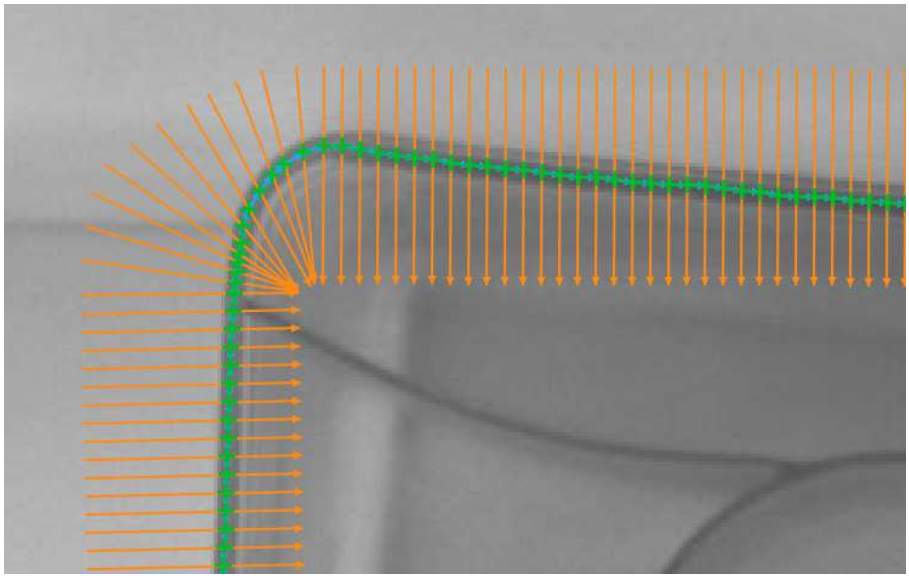
The operation tries to fit a given path to ridges present in the **inImage** image. Internally, it performs a series of scans with the ScanSingleRidge_Direct filter along a number of specific scan segments which length is always equal to the **inFittingField** width and cannot be less than 4. The found points are then used to determine the actual position of the path in the image. Only **inMaxIncompleteness** fraction of these scans may fail. If the fitting according to the given parameters is not possible, **outPath** is set to Nil.

There are also another parameters that control the path fitting process. The **inMaxDeviationDelta** parameter defines the maximal allowed difference between deviations of consecutive points from the input path points. If some of the scans fail or if some of found points are classified to be wrong according to another control parameters, output path points corresponding to them are interpolated depending on points in their nearest vicinity. No more than **inMaxInterpolationLength** consecutive points can be interpolated. The exception to this behavior are points which were not found on both ends of the input path. Those are not part of the result at all.

Hints

- Connect an input image to the **inImage** input.
- Define **inRidgeScanParams.RidgePolarity** to detect a particular ridge type, and only that type.
- Adjust **inRidgeScanParams.RidgeWidth** to the expected thickness of the ridge (in pixels).
- If no or too few ridge points are found, try decreasing **inRidgeScanParams.MinMagnitude**.
- If some of the scans may fail, set the **inMaxIncompleteness** input accordingly.
- Use the **outRidges** outputs to visualize the scanning results.

Examples



*Fitting a path to the walls of a plastic capsule (**inRidgeScanParams.RidgePolarity** = Dark, **inRidgeScanParams.RidgeWidth** = 6, **inRidgeScanParams.RidgeOperator** = ArithmeticMean).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the Shape Fitting toolset. To read more about this technique, one can refer to the Shape Fitting chapter of our Machine Vision Guide

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- CreatePathFittingMap – Precomputes a data object that is required for fast path fitting on images.
- FitPathToEdges_Direct – Performs a series of 1D edge detections and creates a path from the detected points.
- FitPathToStripe_Direct – Performs a series of 1D stripe detections and creates a path from the detected points.
- FitPathToRidges – Performs a series of 1D ridge detections and creates a path from the detected points.

FitPathToStripe

Header: AVL.h
Namespace: avl
Module: MetrologyPro

Performs a series of 1D stripe detections and creates a path from the detected points.

Applications: Tracing of a stripe, whose rough location and shape is known beforehand.

Syntax

```
void avl::FitPathToStripe
(
 const avl::Image& inImage,
 const avl::PathFittingMap& inFittingMap,
 const StripeScanParams& inStripeScanParams,
 avl::Selection::Type inStripeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxInterpolationLength,
 atl::Optional<float> inMaxDeviationDelta,
 float inMaxIncompleteness,
 atl::Conditional<avl::Path>& outPath,
 atl::Conditional<avl::Path>& outLeftPath,
 atl::Conditional<avl::Path>& outRightPath,
 atl::Array<atl::Conditional<avl::Stripe1D>>& outStripes,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outBrightnessProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image to fit the path to
→	inFittingMap	const PathFittingMap&			Input fitting map
→	inStripeScanParams	const StripeScanParams&			Parameters controlling the stripe extraction process
→	inStripeSelection	Selection::Type		Selection::Best	Selection mode of stripe
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxInterpolationLength	Optional<int>	0 - ∞	NIL	Maximal number of consecutive points not found
→	inMaxDeviationDelta	Optional<float>	0.0 - ∞	NIL	Maximal difference between deviations of consecutive path points
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of stripe points not found
←	outPath	Conditional<Path>&			Fitted path in the middle of found stripe
←	outLeftPath	Conditional<Path>&			Fitted left path
←	outRightPath	Conditional<Path>&			Fitted right path
←	outStripes	Array<Conditional<Stripe1D>>&			Found stripes
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual path points and the corresponding reference path points
←	outBrightnessProfiles	Optional<Array<Profile>&>		NIL	Extracted image profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the edge (derivative) operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outDeviationProfile**, **outBrightnessProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

Description

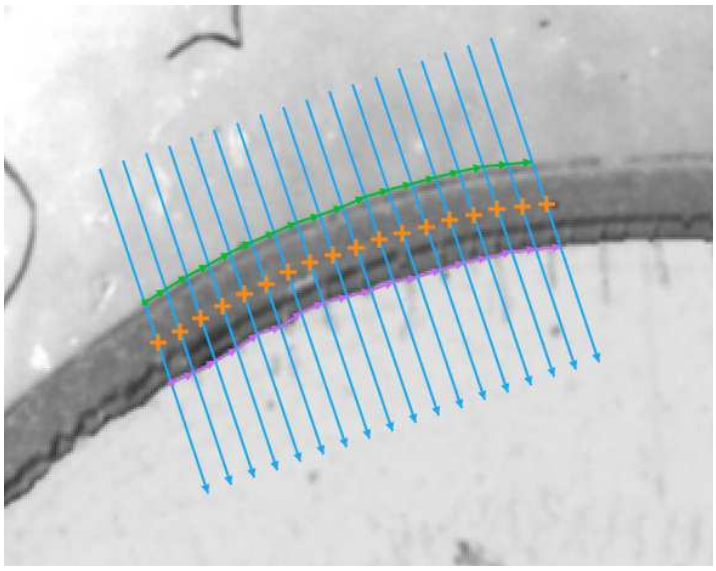
The operation tries to fit a given path to stripe present in the **inImage** image. Internally, it performs a series of scans with the ScanSingleStripe filter using **inFittingMap** previously generated from the object being fitted. The found points are then used to determine the actual position of the path in the image. Only **inMaxIncompleteness** fraction of these scans may fail. If the fitting according to the given parameters is not possible, **outPath** is set to Nil.

There are also another parameters that control the path fitting process. The **inMaxDeviationDelta** parameter defines the maximal allowed difference between deviations of consecutive points from the input path points. If some of the scans fail or if some of found points are classified to be wrong according to another control parameters, output path points corresponding to them are interpolated depending on points in their nearest vicinity. No more than **inMaxInterpolationLength** consecutive points can be interpolated. The exception to this behavior are points which were not found on both ends of the input path. Those are not part of the result at all.

Hints

- Connect an input image to the **inImage** input.
- Define **inStripeScanParams.StripePolarity** to detect a particular stripe type, and only that type.
- Adjust **inStripeScanParams.MinStripeWidth** and **inStripeScanParams.MaxStripeWidth** to the expected thickness of the stripe (in pixels).
- If no or too few stripe points are found, try decreasing **inStripeScanParams.MinMagnitude**.
- If some of the scans may fail, set the **inMaxIncompleteness** input accordingly.
- Use the **outStripes** outputs to visualize the scanning results.

Examples



Fitting a path to a dark stripe
*(**inStripeScanParams.SmoothingStdDev** = 2.0, **inStripeScanParams.MinStripeWidth** = 15.0, **inStripeScanParams.StripePolarity** = Dark).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the Shape Fitting toolset. To read more about this technique, one can refer to the Shape Fitting chapter of our Machine Vision Guide

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- CreatePathFittingMap – Precomputes a data object that is required for fast path fitting on images.
- FitPathToEdges – Performs a series of 1D edge detections and creates a path from the detected points.
- FitPathToRidges – Performs a series of 1D ridge detections and creates a path from the detected points.
- FitPathToStripe_Direct – Performs a series of 1D stripe detections and creates a path from the detected points.

FitPathToStripe_Direct

Header: AVL.h
Namespace: avl
Module: MetrologyPro

Performs a series of 1D stripe detections and creates a path from the detected points.

Applications: Tracing of a stripe, whose rough location and shape is known beforehand.

Syntax

```
void avl::FitPathToStripe_Direct
(
 const avl::Image& inImage,
 const avl::PathFittingField& inFittingField,
 atl::Optional<const avl::CoordinateSystem2D&> inFittingFieldAlignment,
 atl::Optional<float> inScanStep,
 int inScanWidth,
 const avl::SamplingParams& inSamplingParams,
 const avl::StripeScanParams& inStripeScanParams,
 avl::Selection::Type inStripeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxInterpolationLength,
 atl::Optional<float> inMaxDeviationDelta,
 float inMaxIncompleteness,
 atl::Conditional<avl::Path>& outPath,
 atl::Conditional<avl::Path>& outLeftPath,
 atl::Conditional<avl::Path>& outRightPath,
 atl::Array<atl::Conditional<avl::Stripe1D>>& outStripes,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<avl::PathFittingField&> outAlignedFittingField = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outBrightnessProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL,
 atl::Array<avl::Segment2D>& diagScanSegments,
 atl::Array<avl::Rectangle2D>& diagSamplingAreas
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image to fit the path to
→	inFittingField	const PathFittingField&			Path fitting field
→	inFittingFieldAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the fitting field to the position of the inspected object
→	inScanStep	Optional<float>	0.0 - ∞	5.0f	Optional implicit conversion of the input path to an equidistant one
→	inScanWidth	int	1 - ∞	5	The width of each scan field (in pixels)
→	inSamplingParams	const SamplingParams&			Parameters controlling the sampling process
→	inStripeScanParams	const StripeScanParams&		StripeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f MinMagnitude: 5.0f MaxInnerEdgeMagnitude: Nil StripePolarity: Dark MinStripeWidth: 0.0f MaxStripeWidth: Nil)	Parameters controlling the stripe extraction process
→	inStripeSelection	Selection::Type			Selection mode of stripe
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxInterpolationLength	Optional<int>	0 - ∞	1	Maximal number of consecutive points not found
→	inMaxDeviationDelta	Optional<float>	0.0 - ∞	NIL	Maximal difference between deviations of consecutive path points
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of stripe points not found
←	outPath	Conditional<Path>&			Fitted path in the middle of found stripe
←	outLeftPath	Conditional<Path>&			Fitted left path
←	outRightPath	Conditional<Path>&			Fitted right path
←	outStripes	Array<Conditional<Stripe1D>>&			Found stripes
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual path points and the corresponding reference path points
←	outAlignedFittingField	Optional<PathFittingField&>		NIL	Fitting field used; in the image coordinate system
←	outBrightnessProfiles	Optional<Array<Profile>&>		NIL	Extracted image profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the edge (derivative) operator response
🔍	diagScanSegments	Array<Segment2D>&			Segments along which the scans were run
🔍	diagSamplingAreas	Array<Rectangle2D>&			Areas from which the input image is sampled

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outDeviationProfile**, **outAlignedFittingField**, **outBrightnessProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

Description

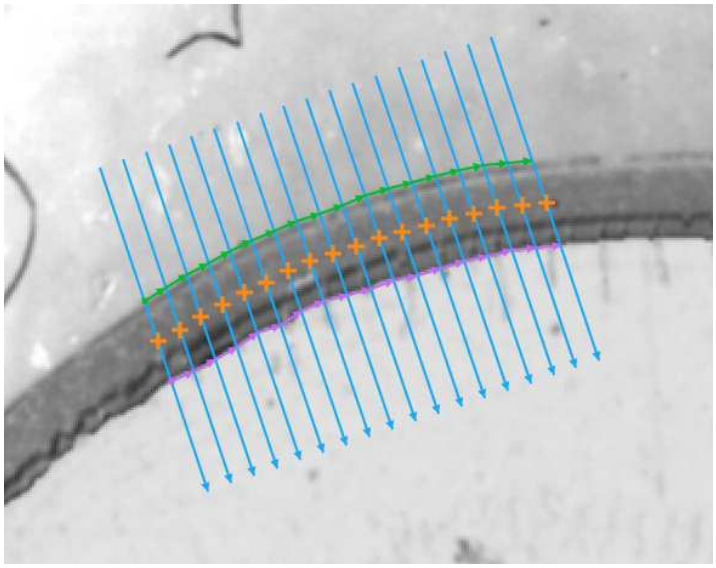
The operation tries to fit a given path to stripe present in the **inImage** image. Internally, it performs a series of scans with the ScanSingleStripe_Direct filter along a number of specific scan segments which length is always equal to the **inFittingField** width and cannot be less than 4. The found points are then used to determine the actual position of the path in the image. Only **inMaxIncompleteness** fraction of these scans may fail. If the fitting according to the given parameters is not possible, **outPath** is set to Nil.

There are also another parameters that control the path fitting process. The **inMaxDeviationDelta** parameter defines the maximal allowed difference between deviations of consecutive points from the input path points. If some of the scans fail or if some of found points are classified to be wrong according to another control parameters, output path points corresponding to them are interpolated depending on points in their nearest vicinity. No more than **inMaxInterpolationLength** consecutive points can be interpolated. The exception to this behavior are points which were not found on both ends of the input path. Those are not part of the result at all.

Hints

- Connect an input image to the **inImage** input.
- Define **inStripeScanParams.StripePolarity** to detect a particular stripe type, and only that type.
- Adjust **inStripeScanParams.MinStripeWidth** and **inStripeScanParams.MaxStripeWidth** to the expected thickness of the stripe (in pixels).
- If no or too few stripe points are found, try decreasing **inStripeScanParams.MinMagnitude**.
- If some of the scans may fail, set the **inMaxIncompleteness** input accordingly.
- Use the **outStripes** outputs to visualize the scanning results.

Examples



Fitting a path to a dark stripe
*(**inStripeScanParams.SmoothingStdDev** = 2.0, **inStripeScanParams.MinStripeWidth** = 15.0, **inStripeScanParams.StripePolarity** = Dark).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the Shape Fitting toolset. To read more about this technique, one can refer to the Shape Fitting chapter of our Machine Vision Guide

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- CreatePathFittingMap – Precomputes a data object that is required for fast path fitting on images.
- FitPathToEdges_Direct – Performs a series of 1D edge detections and creates a path from the detected points.
- FitPathToRidges_Direct – Performs a series of 1D ridge detections and creates a path from the detected points.
- FitPathToStripe – Performs a series of 1D stripe detections and creates a path from the detected points.

FitSegmentToEdges

Header: AVL.h
Namespace: avl
Module: MetrologyPro

Performs a series of 1D edge detections and finds a segment that best matches the detected points.

Applications: Precise detection of a straight edge, whose rough location is known beforehand.

Syntax

```
void avl::FitSegmentToEdges
(
 const avl::Image& inImage,
 const avl::SegmentFittingMap& inFittingMap,
 const EdgeScanParams& inEdgeScanParams,
 avl::Selection::Type inEdgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 float inMaxIncompleteness,
 atl::Optional<avl::LineMEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Segment2D>& outSegment,
 atl::Optional<atl::Array<atl::Conditional<avl::Edge1D>>&> outEdges = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<atl::Array<avl::Point2D>&> outInliers = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outBrightnessProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image to fit segment to
→	inFittingMap	const SegmentFittingMap&			Input fitting map
→	inEdgeScanParams	const EdgeScanParams&			Parameters controlling the edge extraction process
→	inEdgeSelection	Selection::Type		Selection::Best	Selection mode of edges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of edge points not found
→	inOutlierSuppression	Optional<LineMEstimator::Type>		NIL	Selects a method for ignoring incorrectly detected points
←	outSegment	Conditional<Segment2D>&			Fitted segment or nothing if the fitting fails
←	outEdges	Optional<Array<Conditional<Edge1D>>&>		NIL	Found edges
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual segment points and the corresponding reference segment points
←	outInliers	Optional<Array<Point2D>&>		NIL	Points matching the fitting segment
←	outBrightnessProfiles	Optional<Array<Profile>&>		NIL	Extracted image profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the edge (derivative) operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outEdges**, **outDeviationProfile**, **outInliers**, **outBrightnessProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

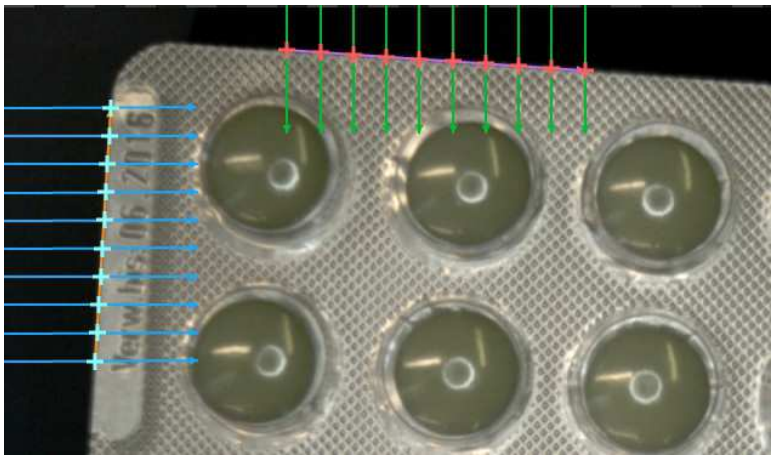
Description

The operation tries to fit a given segment to edges present in the **inImage** image. Internally, it performs a series of scans with the ScanSingleEdge filter using **inFittingMap** previously generated from the object being fitted. The found points are then used to determine the actual position of the segment in the image. Only **inMaxIncompleteness** fraction of these scans may fail. If the fitting according to the given parameters is not possible, **outSegment** is set to Nil.

Hints

- Connect an input image to the **inImage** input.
- Define **inEdgeScanParams.EdgeTransition** to detect a particular edge type, and only that type.
- If no or too few edge points are found, try decreasing **inEdgeScanParams.MinMagnitude**.
- If some of the scans may fail, set the **inMaxIncompleteness** input accordingly.
- If some of the scans may produce false results, try different values of the **inOutlierSuppression** input.
- Use the **outEdges** outputs to visualize the scanning results.

Examples



Fitting two segments to the edges of a blister
*(**inEdgeScanParams.EdgeTransition** = DarkToBright, **inEdgeScanParams.SmoothingStdDev** = 1.0).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the Shape Fitting toolset. To read more about this technique, one can refer to the Shape Fitting chapter of our Machine Vision Guide

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- CreateSegmentFittingMap – Precomputes a data object that is required for fast segment fitting on images.
- FitSegmentToRidges – Performs a series of 1D ridge detections and finds a segment that best matches the detected points.
- FitSegmentToStripe – Performs a series of 1D stripe detections and finds a segment that best matches the detected points.
- FitSegmentToEdges_Direct – Performs a series of 1D edge detections and finds a segment that best matches the detected points.

FitSegmentToEdges_Direct

Header: AVL.h
Namespace: avl
Module: MetrologyPro

Performs a series of 1D edge detections and finds a segment that best matches the detected points.

Applications: Precise detection of a straight edge, whose rough location is known beforehand.

Syntax

```
void avl::FitSegmentToEdges_Direct
(
 const avl::Image& inImage,
 const avl::SegmentFittingField& inFittingField,
 atl::Optional<const avl::CoordinateSystem2D&> inFittingFieldAlignment,
 int inScanCount,
 int inScanWidth,
 const avl::SamplingParams& inSamplingParams,
 const avl::EdgeScanParams& inEdgeScanParams,
 avl::Selection::Type inEdgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 float inMaxIncompleteness,
 atl::Optional<avl::LineMEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Segment2D>& outSegment,
 atl::Optional<atl::Array<atl::Conditional<avl::Edge1D>>&> outEdges = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<avl::SegmentFittingField&> outAlignedFittingField = atl::NIL,
 atl::Optional<atl::Array<avl::Point2D>&> outInliers = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outBrightnessProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL,
 atl::Array<avl::Segment2D>& diagScanSegments,
 atl::Array<avl::Rectangle2D>& diagSamplingAreas
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image to fit segment to
→	inFittingField	const SegmentFittingField&			Segment fitting field
→	inFittingFieldAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the fitting field to the position of the inspected object
→	inScanCount	int	3 - ∞	10	The number of points that will be searched to estimate the position of the segment
→	inScanWidth	int	1 - ∞	5	The width of each scan field (in pixels)
→	inSamplingParams	const SamplingParams&			Parameters controlling the sampling process
→	inEdgeScanParams	const EdgeScanParams&		EdgeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 1.0f MinMagnitude: 5.0f EdgeTransition: BrightToDark)	Parameters controlling the edge extraction process
→	inEdgeSelection	Selection::Type			Selection mode of edges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of edge points not found
→	inOutlierSuppression	Optional<LineMEstimator::Type>		NIL	Selects a method for ignoring incorrectly detected points
←	outSegment	Conditional<Segment2D>&			Fitted segment or nothing if the fitting fails
←	outEdges	Optional<Array<Conditional<Edge1D>>&>		NIL	Found edges
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual segment points and the corresponding reference segment points
←	outAlignedFittingField	Optional<SegmentFittingField&>		NIL	Fitting field used; in the image coordinate system
←	outInliers	Optional<Array<Point2D>&>		NIL	Points matching the fitting segment
←	outBrightnessProfiles	Optional<Array<Profile>&>		NIL	Extracted image profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the edge (derivative) operator response
🔍	diagScanSegments	Array<Segment2D>&			Segments along which the scans were run
🔍	diagSamplingAreas	Array<Rectangle2D>&			Areas from which the input image is sampled

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outEdges**, **outDeviationProfile**, **outAlignedFittingField**, **outInliers**, **outBrightnessProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

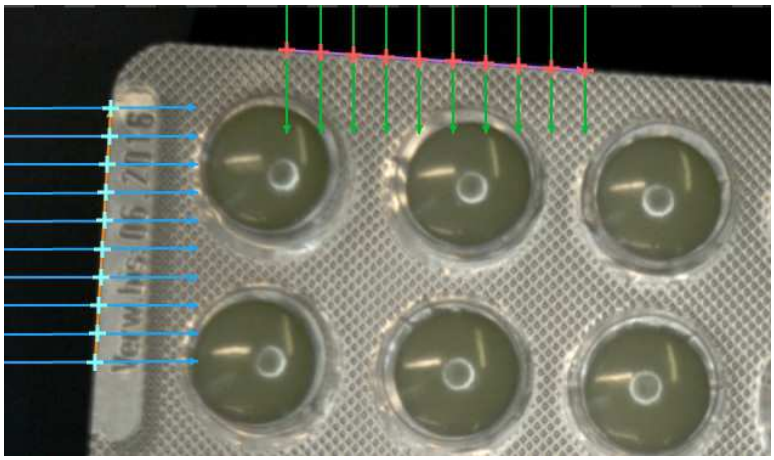
Description

The operation tries to fit a given segment to edges present in the **inImage** image. Internally, it performs a series of scans with the ScanSingleEdge_Direct filter along **inScanCount** specific scan segments which length is always equal to the **inFittingField** width and cannot be less than 4. The found points are then used to determine the actual position of the segment in the image. Only **inMaxIncompleteness** fraction of these scans may fail. If the fitting according to the given parameters is not possible, **outSegment** is set to Nil.

Hints

- Connect an input image to the **inImage** input.
- Define **inEdgeScanParams.EdgeTransition** to detect a particular edge type, and only that type.
- If no or too few edge points are found, try decreasing **inEdgeScanParams.MinMagnitude**.
- If some of the scans may fail, set the **inMaxIncompleteness** input accordingly.
- If some of the scans may produce false results, try different values of the **inOutlierSuppression** input.
- Use the **outEdges** outputs to visualize the scanning results.

Examples



Fitting two segments to the edges of a blister
*(**inEdgeScanParams.EdgeTransition** = DarkToBright, **inEdgeScanParams.SmoothingStdDev** = 1.0).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the Shape Fitting toolset. To read more about this technique, one can refer to the Shape Fitting chapter of our Machine Vision Guide

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- CreateSegmentFittingMap – Precomputes a data object that is required for fast segment fitting on images.
- FitSegmentToRidges_Direct – Performs a series of 1D ridge detections and finds a segment that best matches the detected points.
- FitSegmentToStripe_Direct – Performs a series of 1D stripe detections and finds a segment that best matches the detected points.
- FitSegmentToEdges – Performs a series of 1D edge detections and finds a segment that best matches the detected points.

FitSegmentToRidges

Header: AVL.h
Namespace: avl
Module: MetrologyPro

Performs a series of 1D ridge detections and finds a segment that best matches the detected points.

Applications: Precise detection of a thin straight line, whose rough location is known beforehand.

Syntax

```
void avl::FitSegmentToRidges
(
 const avl::Image& inImage,
 const avl::SegmentFittingMap& inFittingMap,
 const RidgeScanParams& inRidgeScanParams,
 avl::Selection::Type inRidgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 float inMaxIncompleteness,
 atl::Optional<LineMEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Segment2D>& outSegment,
 atl::Optional<atl::Array<atl::Conditional<avl::Ridge1D>>&> outRidges = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<atl::Array<avl::Point2D>&> outInliers = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outBrightnessProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image to fit segment to
→	inFittingMap	const SegmentFittingMap&			Input fitting map
→	inRidgeScanParams	const RidgeScanParams&			Parameters controlling the ridge extraction process
→	inRidgeSelection	Selection::Type		Selection::Best	Selection mode of ridges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker ridges can be detected in the vicinity of stronger ridges
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of ridge points not found
→	inOutlierSuppression	Optional<LineMEstimator::Type>		NIL	Selects a method for ignoring incorrectly detected points
←	outSegment	Conditional<Segment2D>&			Fitted segment or nothing if the fitting fails
←	outRidges	Optional<Array<Conditional<Ridge1D>>&>		NIL	Found ridges
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual segment points and the corresponding reference segment points
←	outInliers	Optional<Array<Point2D>&>		NIL	Points matching the fitting segment
←	outBrightnessProfiles	Optional<Array<Profile>&>		NIL	Extracted image profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the ridge operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outRidges**, **outDeviationProfile**, **outInliers**, **outBrightnessProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

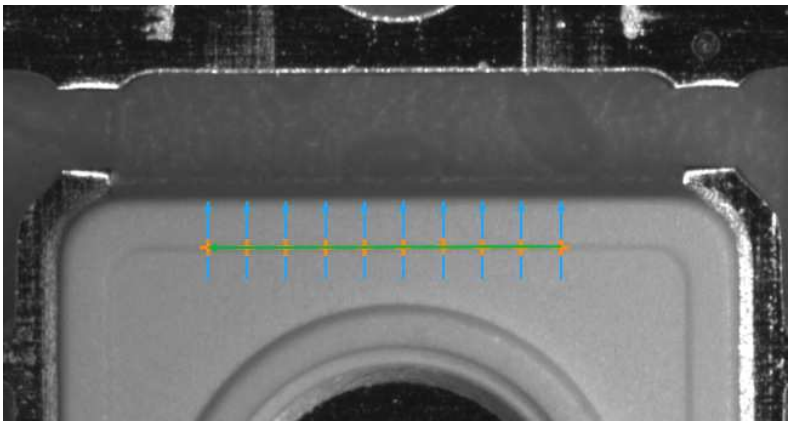
Description

The operation tries to fit a given segment to ridges present in the **inImage** image. Internally, it performs a series of scans with the ScanSingleRidge filter using **inFittingMap** previously generated from the object being fitted. The found points are then used to determine the actual position of the segment in the image. Only **inMaxIncompleteness** fraction of these scans may fail. If the fitting according to the given parameters is not possible, **outSegment** is set to Nil.

Hints

- Connect an input image to the **inImage** input.
- Define **inRidgeScanParams.RidgePolarity** to detect a particular ridge type, and only that type.
- Adjust **inRidgeScanParams.RidgeWidth** to the expected thickness of the ridge (in pixels).
- If no or too few ridge points are found, try decreasing **inRidgeScanParams.MinMagnitude**.
- If some of the scans may fail, set the **inMaxIncompleteness** input accordingly.
- If some of the scans may produce false results, try different values of the **inOutlierSuppression** input.
- Use the **outRidges** outputs to visualize the scanning results.

Examples



Fitting a segment to a line on an object
*(**inRidgeScanParams.RidgePolarity** = Dark).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the Shape Fitting toolset. To read more about this technique, one can refer to the Shape Fitting chapter of our Machine Vision Guide

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- CreateSegmentFittingMap – Precomputes a data object that is required for fast segment fitting on images.
- FitSegmentToEdges – Performs a series of 1D edge detections and finds a segment that best matches the detected points.
- FitSegmentToStripe – Performs a series of 1D stripe detections and finds a segment that best matches the detected points.
- FitSegmentToRidges_Direct – Performs a series of 1D ridge detections and finds a segment that best matches the detected points.

FitSegmentToRidges_Direct

Header: AVL.h
Namespace: avl
Module: MetrologyPro

Performs a series of 1D ridge detections and finds a segment that best matches the detected points.

Applications: Precise detection of a thin straight line, whose rough location is known beforehand.

Syntax

```
void avl::FitSegmentToRidges_Direct
(
 const avl::Image& inImage,
 const avl::SegmentFittingField& inFittingField,
 atl::Optional<const avl::CoordinateSystem2D&> inFittingFieldAlignment,
 int inScanCount,
 int inScanWidth,
 const avl::SamplingParams& inSamplingParams,
 const avl::RidgeScanParams& inRidgeScanParams,
 avl::Selection::Type inRidgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 float inMaxIncompleteness,
 atl::Optional<avl::LineMEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Segment2D>& outSegment,
 atl::Optional<atl::Array<atl::Conditional<avl::Ridge1D>>&> outRidges = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<avl::SegmentFittingField&> outAlignedFittingField = atl::NIL,
 atl::Optional<atl::Array<avl::Point2D>&> outInliers = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outBrightnessProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL,
 atl::Array<avl::Segment2D>& diagScanSegments,
 atl::Array<avl::Rectangle2D>& diagSamplingAreas
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image to fit segment to
→	inFittingField	const SegmentFittingField&			Segment fitting field
→	inFittingFieldAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the fitting field to the position of the inspected object
→	inScanCount	int	3 - ∞	10	The number of points that will be searched to estimate the position of the segment
→	inScanWidth	int	1 - ∞	5	The width of each scan field (in pixels)
→	inSamplingParams	const SamplingParams&			Parameters controlling the sampling process
→	inRidgeScanParams	const RidgeScanParams&		RidgeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 1.0f RidgeWidth: 5 RidgeMargin: 2 MinMagnitude: 5.0f RidgePolarity: Dark)	Parameters controlling the ridge extraction process
→	inRidgeSelection	Selection::Type			Selection mode of ridges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker ridges can be detected in the vicinity of stronger ridges
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of ridge points not found
→	inOutlierSuppression	Optional<LineMEstimator::Type>		NIL	Selects a method for ignoring incorrectly detected points
←	outSegment	Conditional<Segment2D>&			Fitted segment or nothing if the fitting fails
←	outRidges	Optional<Array<Conditional<Ridge1D>>&>		NIL	Found ridges
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual segment points and the corresponding reference segment points
←	outAlignedFittingField	Optional<SegmentFittingField&>		NIL	Fitting field used; in the image coordinate system
←	outInliers	Optional<Array<Point2D>&>		NIL	Points matching the fitting segment
←	outBrightnessProfiles	Optional<Array<Profile>&>		NIL	Extracted image profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the ridge operator response
🔍	diagScanSegments	Array<Segment2D>&			Segments along which the scans were run
🔍	diagSamplingAreas	Array<Rectangle2D>&			Areas from which the input image is sampled

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outRidges**, **outDeviationProfile**, **outAlignedFittingField**, **outInliers**, **outBrightnessProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

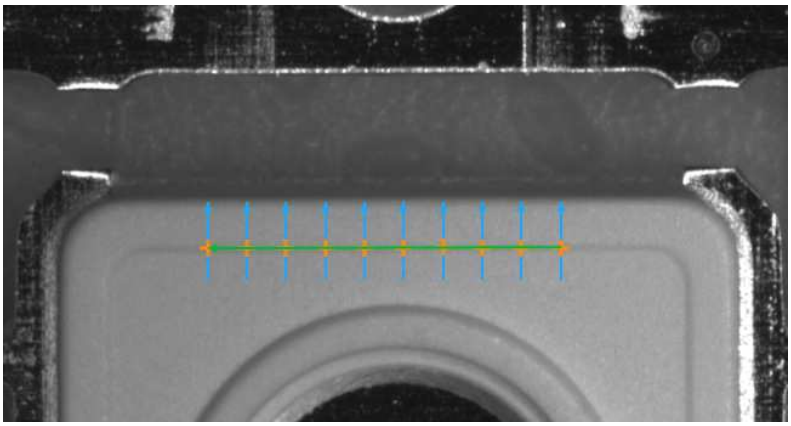
Description

The operation tries to fit a given segment to ridges present in the **inImage** image. Internally, it performs a series of scans with the ScanSingleRidge_Direct filter along **inScanCount** specific scan segments which length is always equal to the **inFittingField** width and cannot be less than 4. The found points are then used to determine the actual position of the segment in the image. Only **inMaxIncompleteness** fraction of these scans may fail. If the fitting according to the given parameters is not possible, **outSegment** is set to Nil.

Hints

- Connect an input image to the **inImage** input.
- Define **inRidgeScanParams.RidgePolarity** to detect a particular ridge type, and only that type.
- Adjust **inRidgeScanParams.RidgeWidth** to the expected thickness of the ridge (in pixels).
- If no or too few ridge points are found, try decreasing **inRidgeScanParams.MinMagnitude**.
- If some of the scans may fail, set the **inMaxIncompleteness** input accordingly.
- If some of the scans may produce false results, try different values of the **inOutlierSuppression** input.
- Use the **outRidges** outputs to visualize the scanning results.

Examples



Fitting a segment to a line on an object
*(**inRidgeScanParams.RidgePolarity** = Dark).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the Shape Fitting toolset. To read more about this technique, one can refer to the Shape Fitting chapter of our Machine Vision Guide

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- CreateSegmentFittingMap – Precomputes a data object that is required for fast segment fitting on images.
- FitSegmentToEdges_Direct – Performs a series of 1D edge detections and finds a segment that best matches the detected points.
- FitSegmentToStripe_Direct – Performs a series of 1D stripe detections and finds a segment that best matches the detected points.
- FitSegmentToRidges – Performs a series of 1D ridge detections and finds a segment that best matches the detected points.

FitSegmentToStripe

Header: AVL.h
Namespace: avl
Module: MetrologyPro

Performs a series of 1D stripe detections and finds a segment that best matches the detected points.

Applications: Precise detection of a straight stripe, whose rough location is known beforehand.

Syntax

```
void avl::FitSegmentToStripe
(
 const avl::Image& inImage,
 const avl::SegmentFittingMap& inFittingMap,
 const StripeScanParams& inStripeScanParams,
 avl::Selection::Type inStripeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 float inMaxIncompleteness,
 atl::Optional<avl::LineMEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Segment2D>& outSegment,
 atl::Conditional<avl::Segment2D>& outLeftSegment,
 atl::Conditional<avl::Segment2D>& outRightSegment,
 atl::Optional<atl::Array<atl::Conditional<avl::Stripe1D>>&> outStripes = atl::NIL,
 atl::Optional<atl::Array<avl::Point2D>&> outStripePoints = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outBrightnessProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image to fit segment to
→	inFittingMap	const SegmentFittingMap&			Input fitting map
→	inStripeScanParams	const StripeScanParams&			Parameters controlling the stripe extraction process
→	inStripeSelection	Selection::Type		Selection::Best	Selection mode of stripe
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of stripe points not found
→	inOutlierSuppression	Optional<LineMEstimator::Type>		NIL	Selects a method for ignoring incorrectly detected points
←	outSegment	Conditional<Segment2D>&			Fitted segment in the middle of found stripe
←	outLeftSegment	Conditional<Segment2D>&			Fitted left segment
←	outRightSegment	Conditional<Segment2D>&			Fitted right segment
←	outStripes	Optional<Array<Conditional<Stripe1D>>&>		NIL	Found stripes
←	outStripePoints	Optional<Array<Point2D>&>		NIL	Extracted points of middle segment of an image stripe
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual segment points and the corresponding reference segment points
←	outBrightnessProfiles	Optional<Array<Profile>&>		NIL	Extracted image profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the edge (derivative) operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outStripes**, **outStripePoints**, **outDeviationProfile**, **outBrightnessProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

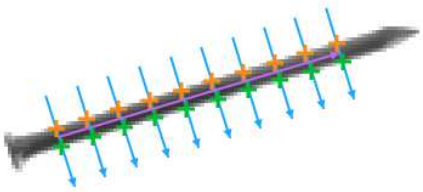
Description

The operation tries to fit a given segment to stripe present in the **inImage** image. Internally, it performs a series of scans with the ScanSingleStripe filter using **inFittingMap** previously generated from the object being fitted. The found points are then used to determine the actual position of the segment in the image. Only **inMaxIncompleteness** fraction of these scans may fail. If the fitting according to the given parameters is not possible, **outSegment** is set to Nil.

Hints

- Connect an input image to the **inImage** input.
- Define **inStripeScanParams.StripePolarity** to detect a particular stripe type, and only that type.
- Adjust **inStripeScanParams.MinStripeWidth** and **inStripeScanParams.MaxStripeWidth** to the expected thickness of the stripe (in pixels).
- If no or too few stripe points are found, try decreasing **inStripeScanParams.MinMagnitude**.
- If some of the scans may fail, set the **inMaxIncompleteness** input accordingly.
- If some of the scans may produce false results, try different values of the **inOutlierSuppression** input.
- Use the **outStripePoints** outputs to visualize the scanning results.

Examples



Fitting a segment to the dark stripe of a nail
*(**inStripeScanParams.Polarity** = Dark).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the Shape Fitting toolset. To read more about this technique, one can refer to the Shape Fitting chapter of our Machine Vision Guide

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- CreateSegmentFittingMap – Precomputes a data object that is required for fast segment fitting on images.
- FitSegmentToEdges – Performs a series of 1D edge detections and finds a segment that best matches the detected points.
- FitSegmentToRidges – Performs a series of 1D ridge detections and finds a segment that best matches the detected points.
- FitSegmentToStripe_Direct – Performs a series of 1D stripe detections and finds a segment that best matches the detected points.

FitSegmentToStripe_Direct

Header: AVL.h
Namespace: avl
Module: MetrologyPro

Performs a series of 1D stripe detections and finds a segment that best matches the detected points.

Applications: Precise detection of a straight stripe, whose rough location is known beforehand.

Syntax

```
void avl::FitSegmentToStripe_Direct
(
 const avl::Image& inImage,
 const avl::SegmentFittingField& inFittingField,
 atl::Optional<const avl::CoordinateSystem2D&> inFittingFieldAlignment,
 int inScanCount,
 int inScanWidth,
 const avl::SamplingParams& inSamplingParams,
 const avl::StripeScanParams& inStripeScanParams,
 avl::Selection::Type inStripeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 float inMaxIncompleteness,
 atl::Optional<avl::LineMEstimator::Type> inOutlierSuppression,
 atl::Conditional<avl::Segment2D>& outSegment,
 atl::Conditional<avl::Segment2D>& outLeftSegment,
 atl::Conditional<avl::Segment2D>& outRightSegment,
 atl::Optional<atl::Array<atl::Conditional<avl::Stripe1D>>&> outStripes = atl::NIL,
 atl::Optional<atl::Array<avl::Point2D>&> outStripePoints = atl::NIL,
 atl::Optional<atl::Conditional<avl::Profile>&> outDeviationProfile = atl::NIL,
 atl::Optional<avl::SegmentFittingField&> outAlignedFittingField = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outBrightnessProfiles = atl::NIL,
 atl::Optional<atl::Array<avl::Profile>&> outResponseProfiles = atl::NIL,
 atl::Array<avl::Segment2D>& diagScanSegments,
 atl::Array<avl::Rectangle2D>& diagSamplingAreas
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image to fit segment to
→	inFittingField	const SegmentFittingField&			Segment fitting field
→	inFittingFieldAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the fitting field to the position of the inspected object
→	inScanCount	int	3 - ∞	10	The number of points that will be searched to estimate the position of the segment
→	inScanWidth	int	1 - ∞	5	The width of each scan field (in pixels)
→	inSamplingParams	const SamplingParams&			Parameters controlling the sampling process
→	inStripeScanParams	const StripeScanParams&		StripeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f MinMagnitude: 5.0f MaxInnerEdgeMagnitude: Nil StripePolarity: Dark MinStripeWidth: 0.0f MaxStripeWidth: Nil)	Parameters controlling the stripe extraction process
→	inStripeSelection	Selection::Type			Selection mode of stripe
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxIncompleteness	float	0.0 - 0.999	0.1f	Maximal fraction of stripe points not found
→	inOutlierSuppression	Optional<LineMEstimator::Type>		NIL	Selects a method for ignoring incorrectly detected points
←	outSegment	Conditional<Segment2D>&			Fitted segment in the middle of found stripe
←	outLeftSegment	Conditional<Segment2D>&			Fitted left segment
←	outRightSegment	Conditional<Segment2D>&			Fitted right segment
←	outStripes	Optional<Array<Conditional<Stripe1D>>&>		NIL	Found stripes
←	outStripePoints	Optional<Array<Point2D>&>		NIL	Extracted points of middle segment of an image stripe
←	outDeviationProfile	Optional<Conditional<Profile>&>		NIL	Profile of distances between the actual segment points and the corresponding reference segment points
←	outAlignedFittingField	Optional<SegmentFittingField&>		NIL	Fitting field used; in the image coordinate system
←	outBrightnessProfiles	Optional<Array<Profile>&>		NIL	Extracted image profiles
←	outResponseProfiles	Optional<Array<Profile>&>		NIL	Profiles of the edge (derivative) operator response
🔍	diagScanSegments	Array<Segment2D>&			Segments along which the scans were run
🔍	diagSamplingAreas	Array<Rectangle2D>&			Areas from which the input image is sampled

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outStripes**, **outStripePoints**, **outDeviationProfile**, **outAlignedFittingField**, **outBrightnessProfiles**, **outResponseProfiles**.

Read more about Optional Outputs.

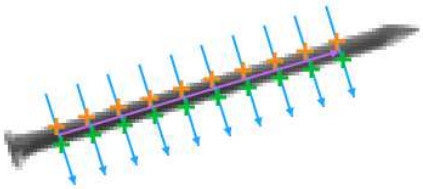
Description

The operation tries to fit a given segment to stripe present in the **inImage** image. Internally, it performs a series of scans with the ScanSingleStripe_Direct filter along **inScanCount** specific scan segments which length is always equal to the **inFittingField** width and cannot be less than 4. The found points are then used to determine the actual position of the segment in the image. Only **inMaxIncompleteness** fraction of these scans may fail. If the fitting according to the given parameters is not possible, **outSegment** is set to Nil.

Hints

- Connect an input image to the **inImage** input.
- Define **inStripeScanParams.StripePolarity** to detect a particular stripe type, and only that type.
- Adjust **inStripeScanParams.MinStripeWidth** and **inStripeScanParams.MaxStripeWidth** to the expected thickness of the stripe (in pixels).
- If no or too few stripe points are found, try decreasing **inStripeScanParams.MinMagnitude**.
- If some of the scans may fail, set the **inMaxIncompleteness** input accordingly.
- If some of the scans may produce false results, try different values of the **inOutlierSuppression** input.
- Use the **outStripePoints** outputs to visualize the scanning results.

Examples



Fitting a segment to the dark stripe of a nail
*(**inStripeScanParams.Polarity** = Dark).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the Shape Fitting toolset. To read more about this technique, one can refer to the Shape Fitting chapter of our Machine Vision Guide

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- CreateSegmentFittingMap – Precomputes a data object that is required for fast segment fitting on images.
- FitSegmentToEdges_Direct – Performs a series of 1D edge detections and finds a segment that best matches the detected points.
- FitSegmentToRidges_Direct – Performs a series of 1D ridge detections and finds a segment that best matches the detected points.
- FitSegmentToStripe – Performs a series of 1D stripe detections and finds a segment that best matches the detected points.

ImageShortestPath_Experimental

Header: AVL.h
Namespace: avl

Syntax

```
void avl::ImageShortestPath_Experimental
(
 const avl::Image& inImage,
 int inMarginX,
 int inMarginY,
 int inStep,
 float inMaxAngleDeviation,
 avl::InterpolationMethod::Type inMethod,
 avl::Path& outPath
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inMarginX	int			
→	inMarginY	int			
→	inStep	int	1 - ∞	5	
→	inMaxAngleDeviation	float	0.0 - 90.0	45.0f	
→	inMethod	InterpolationMethod::Type			
←	outPath	Path&			Output path

71. Region Basics

Table of content:

CreateBoxBorderRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a frame-shaped region with given parameters.

Syntax

```
void avl::CreateBoxBorderRegion
(
 const avl::Box& inBox,
 int inBorderWidth,
 avl::BorderPosition::Type inBorderPosition,
 int inFrameWidth,
 int inFrameHeight,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inBox	const Box&			
→	inBorderWidth	int	0 - ∞	1	
→	inBorderPosition	BorderPosition::Type		Centered	
→	inFrameWidth	int	0 - 65535		Width of the created region's frame
→	inFrameHeight	int	0 - 65535		Height of the created region's frame
←	outRegion	Region&			Output region

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect BorderPosition in CreateBoxBorderRegion.

CreateBoxRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a rectangular region corresponding to a given box.

Syntax

```
void avl::CreateBoxRegion
(
 const avl::Box& inBox,
 int inFrameWidth,
 int inFrameHeight,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inBox	const Box&			A box defining pixels that will be converted to white elements
→	inFrameWidth	int	0 - 65535		Width of the created region's frame (not to be confused with the width of the box!)
→	inFrameHeight	int	0 - 65535		Height of the created region's frame (not to be confused with the height of the box!)
←	outRegion	Region&			Output region

Description

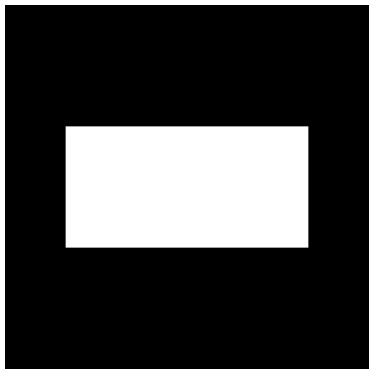
The operation creates a region containing pixels lying inside the **inBox**.

The **inFrameWidth** and **inFrameHeight** parameters most often should be set equal to the dimensions of the image this region will be used with. If the input box exceeds these dimensions, the output region will be cropped.

Hints

- Remember to set **inFrameWidth** and **inFrameHeight** inputs to specify the region frame.

Examples



CreateBoxRegion *run with* ***inBox*** *= Box(50,100,200,100).*

See Also

- CreateEllipseRegion – Creates an elliptic region of given bounding rectangle.
- CreatePolygonRegion – Creates a polygonal region corresponding to a given closed path.

CreateCircleRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a circular region corresponding to a given circle.

Syntax

```
void avl::CreateCircleRegion
(
 const avl::Circle2D& inCircle,
 atl::Optional<const avl::CoordinateSystem2D&> inCircleAlignment,
 int inFrameWidth,
 int inFrameHeight,
 avl::Region& outRegion,
 atl::Optional<avl::Circle2D&> outAlignedCircle = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inCircle	const Circle2D&			
→	inCircleAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the circle to the position of the inspected object
→	inFrameWidth	int	0 - 65535		Width of the created region's frame (not to be confused with the size of the circle!)
→	inFrameHeight	int	0 - 65535		Height of the created region's frame (not to be confused with the size of the circle!)
←	outRegion	Region&			Output region
←	outAlignedCircle	Optional<Circle2D&>		NIL	The input circle transformed to the absolute coordinate system

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedCircle**.

Read more about Optional Outputs.

Description

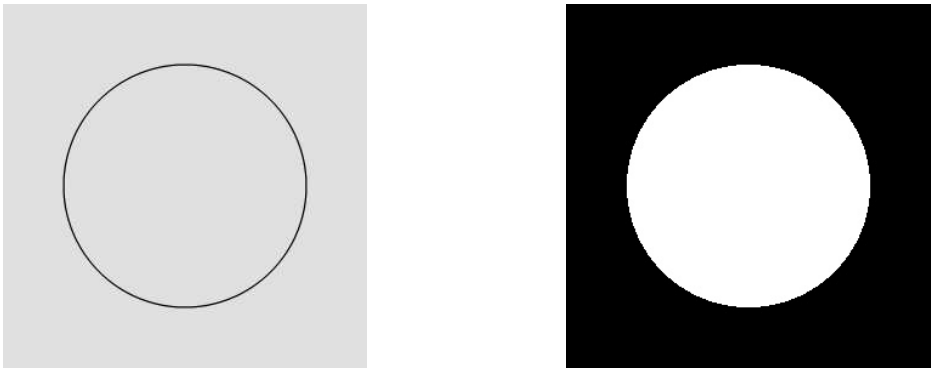
The operation creates a region containing pixels lying inside the given circle.

The **inFrameWidth** and **inFrameHeight** parameters most often should be set equal to the dimensions of the image this region will be used with. If the input circle exceeds these dimensions, the output region will be cropped.

Hints

- Remember to set **inFrameWidth** and **inFrameHeight** inputs to specify the region frame.

Examples



CreateCircleRegion *run with* ***inCircle*** *= Circle2D(150,150,100).*

See Also

- CreateEllipseRegion – Creates an elliptic region of given bounding rectangle.

CreateCrossRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a cross-shaped region with given parameters.

Syntax

```
void avl::CreateCrossRegion
(
 const avl::Box& inCrossBoundingBox,
 int inFrameWidth,
 int inFrameHeight,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inCrossBoundingBox	const Box&			
→	inFrameWidth	int	0 - 65535		Width of the created region's frame
→	inFrameHeight	int	0 - 65535		Height of the created region's frame
←	outRegion	Region&			Output region

Description

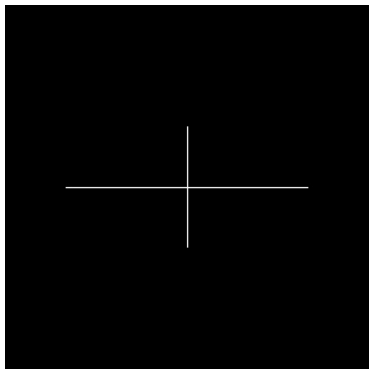
The operation creates a region in a shape of a one-pixel-wide cross. Cross is described by its width, height and coordinates of upper-left corner of its bounding box.

The **inFrameWidth** and **inFrameHeight** parameters most often should be set equal to the dimensions of the image this region will be used with. If the input cross exceeds these dimensions, the output region will be cropped.

Hints

- Remember to set **inFrameWidth** and **inFrameHeight** inputs to specify the region frame.

Examples



***CreateCrossRegion** run with **inCrossBoundingBox** parameters: **X** = 50, **Y** = 100, **Width** = 200, **inCrossHeight** = 100.*

See Also

- CreateBoxRegion – Creates a rectangular region corresponding to a given box.
- CreatePolygonRegion – Creates a polygonal region corresponding to a given closed path.

CreateEllipseRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates an elliptic region of given bounding rectangle.

Syntax

```
void avl::CreateEllipseRegion
(
 const avl::Rectangle2D& inEllipse,
 atl::Optional<const avl::CoordinateSystem2D&> inEllipseAlignment,
 int inFrameWidth,
 int inFrameHeight,
 avl::Region& outRegion,
 atl::Optional<avl::Rectangle2D&> outAlignedEllipse = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inEllipse	const Rectangle2D&			
→	inEllipseAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the ellipse to the position of the inspected object
→	inFrameWidth	int	0 - 65535		Width of the created region's frame (not to be confused with the width of the ellipse!)
→	inFrameHeight	int	0 - 65535		Height of the created region's frame (not to be confused with the height of the ellipse!)
←	outRegion	Region&			Output region
←	outAlignedEllipse	Optional<Rectangle2D&>		NIL	The input ellipse transformed to the absolute coordinate system

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedEllipse**.

Read more about Optional Outputs.

Description

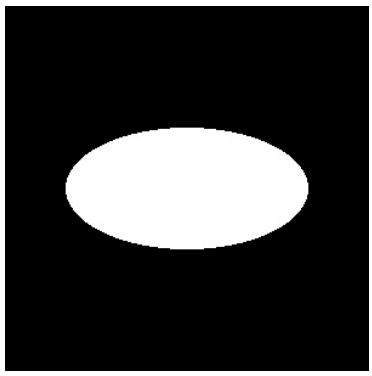
The operation creates a region containing pixels lying inside an ellipse which is described by its bounding rectangle.

The **inFrameWidth** and **inFrameHeight** parameters most often should be set equal to the dimensions of the image this region will be used with. If the input ellipse exceeds these dimensions, the output region will be cropped.

Hints

- Remember to set **inFrameWidth** and **inFrameHeight** inputs to specify the region frame.

Examples



CreateEllipseRegion *run with a sample rectangle.*

See Also

- CreateCircleRegion – Creates a circular region corresponding to a given circle.

CreateGridRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a grid-shaped region with given parameters.

Syntax

```
void avl::CreateGridRegion
(
 const avl::Box& inGridBoundingBox,
 int inHorizontalStep,
 int inVerticalStep,
 int inFrameWidth,
 int inFrameHeight,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inGridBoundingBox	const Box&			
→	inHorizontalStep	int	1 - ∞		Horizontal distance between vertical grid lines.
→	inVerticalStep	int	1 - ∞		Vertical distance between horizontal grid lines.
→	inFrameWidth	int	0 - 65535		Width of the created region's frame
→	inFrameHeight	int	0 - 65535		Height of the created region's frame
←	outRegion	Region&			Output region

Description

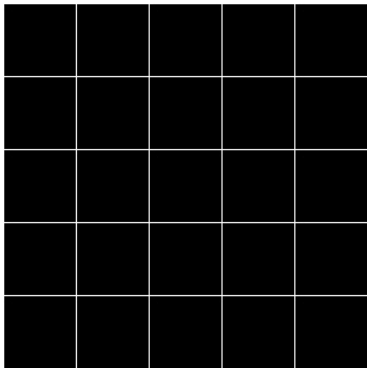
The operation creates a region in a shape of a one-pixel-wide grid. Grid is described by its width, height, distances between its lines and coordinates of its upper-left corner.

The **inFrameWidth** and **inFrameHeight** parameters most often should be set equal to the dimensions of the image this region will be used with.

Hints

- Remember to set **inFrameWidth** and **inFrameHeight** inputs to specify the region frame.

Examples



CreateGridRegion *run with* ***inGridBoundingBox*** *parameters:* ***X*** *= 0,* ***Y*** *= 0,* ***inGridWidth*** *= 300,* ***inGridHeight*** *= 300 and with* ***inHorizontalStep*** *= 60,* ***inVerticalStep*** *= 60*

See Also

- CreateBoxRegion – Creates a rectangular region corresponding to a given box.
- CreateCrossRegion – Creates a cross-shaped region with given parameters.

CreateLineRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a line region.

Syntax

```
void avl::CreateLineRegion
(
 const avl::Line2D& inLine,
 atl::Optional<const avl::CoordinateSystem2D&> inLineAlignment,
 float inWidth,
 int inFrameWidth,
 int inFrameHeight,
 avl::Region& outRegion,
 atl::Optional<avl::Line2D&> outAlignedLine = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inLine	const Line2D&			
→	inLineAlignment	Optional<const CoordinateSystem2D&>		NIL	
→	inWidth	float	1.0 - ∞		Width of output region line
→	inFrameWidth	int	0 - 65535		
→	inFrameHeight	int	0 - 65535		
←	outRegion	Region&			Output region
←	outAlignedLine	Optional<Line2D&>		NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedLine**.

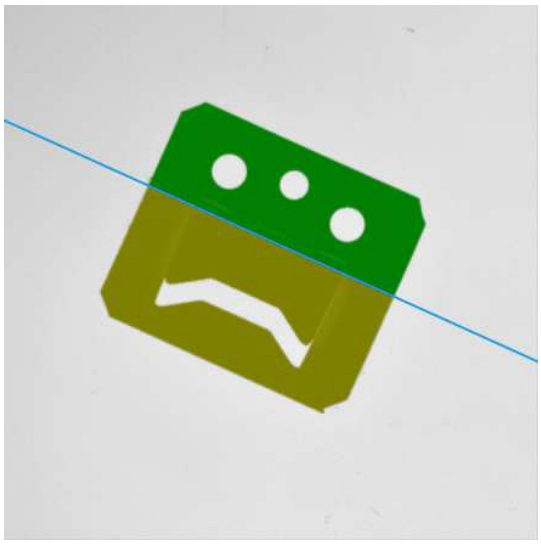
Read more about Optional Outputs.

Description

Filter creates a region defined by an input Line2D with width of **inWidth**.

Examples

Filter CreateLineRegion is used to split region int two parts:



CreatePathBorderRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a region along a given path. The region may be internal or external to the path.

Syntax

```
void avl::CreatePathBorderRegion
(
 const avl::Path& inPath,
 atl::Optional<const avl::CoordinateSystem2D&> inPathAlignment,
 float inBorderWidth,
 avl::BorderPosition::Type inBorderPosition,
 int inFrameWidth,
 int inFrameHeight,
 avl::Region& outRegion,
 atl::Optional<avl::Path&> outAlignedPath = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPath	const Path&			Input path
→	inPathAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the path to the position of the inspected object
→	inBorderWidth	float	1.0 - ∞	1.0f	
→	inBorderPosition	BorderPosition::Type		Centered	
→	inFrameWidth	int	0 - 65535		Width of the created region's frame
→	inFrameHeight	int	0 - 65535		Height of the created region's frame
←	outRegion	Region&			Output region
←	outAlignedPath	Optional<Path&>		NIL	The input path transformed to the absolute coordinate system

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedPath**.

Read more about Optional Outputs.

CreatePathRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a region along the given path.

Syntax

```
void avl::CreatePathRegion
(
 const avl::Path& inPath,
 atl::Optional<const avl::CoordinateSystem2D&> inPathAlignment,
 float inWidth,
 int inFrameWidth,
 int inFrameHeight,
 avl::Region& outRegion,
 atl::Optional<avl::Path&> outAlignedPath = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPath	const Path&			Input path
→	inPathAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the path to the position of the inspected object
→	inWidth	float	1.0 - ∞	1.0f	Width of line used to draw path
→	inFrameWidth	int	0 - 65535		Width of the created region's frame
→	inFrameHeight	int	0 - 65535		Height of the created region's frame
←	outRegion	Region&			Output region
←	outAlignedPath	Optional<Path&>		NIL	The input path transformed to the absolute coordinate system

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedPath**.

Read more about Optional Outputs.

CreatePolygonRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a polygonal region corresponding to a given closed path.

Syntax

```
void avl::CreatePolygonRegion
(
 const avl::Path& inPolygon,
 atl::Optional<const avl::CoordinateSystem2D&> inPolygonAlignment,
 int inFrameWidth,
 int inFrameHeight,
 avl::Region& outRegion,
 atl::Optional<avl::Path&> outAlignedPolygon = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPolygon	const Path&			
→	inPolygonAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the polygon to the position of the inspected object
→	inFrameWidth	int	0 - 65535		Width of the created region's frame
→	inFrameHeight	int	0 - 65535		Height of the created region's frame
←	outRegion	Region&			Output region
←	outAlignedPolygon	Optional<Path&>		NIL	The input polygon transformed to the absolute coordinate system

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedPolygon**.

Read more about Optional Outputs.

Description

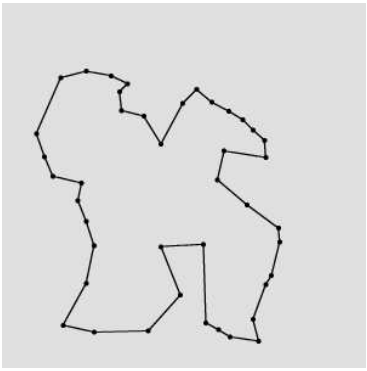
The operation creates a region containing pixels lying inside the shape described by **inPolygon**.

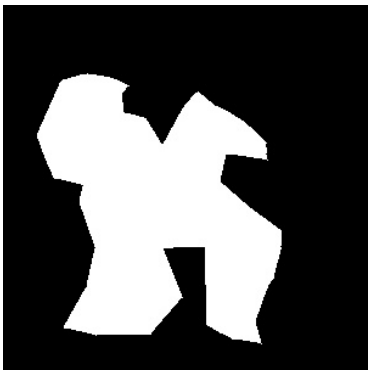
The **inFrameWidth** and **inFrameHeight** parameters most often should be set equal to the dimensions of the image this region will be used with. If the input polygon exceeds these dimensions, the output region will be cropped.

Hints

- Remember to set **inFrameWidth** and **inFrameHeight** inputs to specify the region frame.

Examples





Remarks

- **inPolygon** has to be a closed path, otherwise an error with appropriate description occurs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Open path on input in CreatePolygonRegion.

See Also

- CreateBoxRegion – Creates a rectangular region corresponding to a given box.
- CreateCrossRegion – Creates a cross-shaped region with given parameters.

CreateRectangleBorderRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a frame-shaped region with given parameters.

Syntax

```
void avl::CreateRectangleBorderRegion
(
 const avl::Rectangle2D& inRectangle,
 atl::Optional<const avl::CoordinateSystem2D&> inRectangleAlignment,
 float inBorderWidth,
 avl::BorderPosition::Type inBorderPosition,
 int inFrameWidth,
 int inFrameHeight,
 avl::Region& outRegion,
 atl::Optional<avl::Rectangle2D&> outAlignedRectangle = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inRectangle	const Rectangle2D&			
→	inRectangleAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the rectangle to the position of the inspected object
→	inBorderWidth	float	0.0 - ∞	1.0f	
→	inBorderPosition	BorderPosition::Type		Centered	
→	inFrameWidth	int	0 - 65535		Width of the created region's frame
→	inFrameHeight	int	0 - 65535		Height of the created region's frame
←	outRegion	Region&			Output region
←	outAlignedRectangle	Optional<Rectangle2D&>		NIL	The input rectangle transformed to the absolute coordinate system

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedRectangle**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect BorderPosition in CreateRectangleBorderRegion.

CreateRectangleRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a region corresponding to a given rectangle.

Syntax

```
void avl::CreateRectangleRegion
(
 const avl::Rectangle2D& inRectangle,
 atl::Optional<const avl::CoordinateSystem2D&> inRectangleAlignment,
 int inFrameWidth,
 int inFrameHeight,
 avl::Region& outRegion,
 atl::Optional<avl::Rectangle2D&> outAlignedRectangle = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inRectangle	const Rectangle2D&			
→	inRectangleAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the rectangle to the position of the inspected object
→	inFrameWidth	int	0 - 65535		Width of the created region's frame (not to be confused with the width of the rectangle!)
→	inFrameHeight	int	0 - 65535		Height of the created region's frame (not to be confused with the height of the rectangle!)
←	outRegion	Region&			Output region
←	outAlignedRectangle	Optional<Rectangle2D&>		NIL	The input rectangle transformed to the absolute coordinate system

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedRectangle**.

Read more about Optional Outputs.

Description

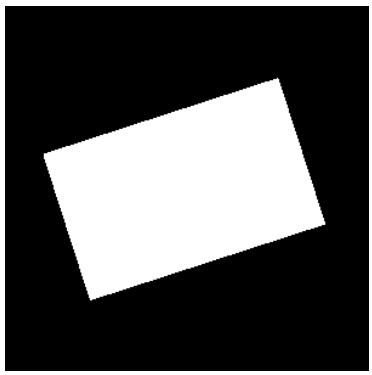
The operation creates a region containing pixels lying inside the specified rectangle.

The **inFrameWidth** and **inFrameHeight** parameters most often should be set equal to the dimensions of the image this region will be used with. If the input rectangle exceeds these dimensions, the output region will be cropped.

Hints

- Remember to set **inFrameWidth** and **inFrameHeight** inputs to specify the region frame.

Examples



See Also

- CreateCircleRegion – Creates a circular region corresponding to a given circle.

CreateRingRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a ring-shaped region with given parameters.

Syntax

```
void avl::CreateRingRegion
(
 const avl::Circle2D& inCircle,
 atl::Optional<const avl::CoordinateSystem2D&> inCircleAlignment,
 float inRingWidth,
 avl::BorderPosition::Type inBorderPosition,
 int inFrameWidth,
 int inFrameHeight,
 avl::Region& outRegion,
 atl::Optional<avl::Circle2D&> outAlignedCircle = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inCircle	const Circle2D&			Input circle
→	inCircleAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the circle to the position of the inspected object
→	inRingWidth	float	0.0 - ∞	1.0f	
→	inBorderPosition	BorderPosition::Type		Centered	
→	inFrameWidth	int	0 - 65535		Width of the created region's frame
→	inFrameHeight	int	0 - 65535		Height of the created region's frame
←	outRegion	Region&			Output region
←	outAlignedCircle	Optional<Circle2D&>		NIL	The input circle transformed to the absolute coordinate system

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedCircle**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect BorderPosition in CreateRingRegion.

CreateSegmentRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates a segment region.

Syntax

```
void avl::CreateSegmentRegion
(
 const avl::Segment2D& inSegment,
 atl::Optional<const avl::CoordinateSystem2D&> inSegmentAlignment,
 float inWidth,
 int inFrameWidth,
 int inFrameHeight,
 bool inRound,
 avl::Region& outRegion,
 atl::Optional<avl::Segment2D&> outAlignedSegment = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSegment	const Segment2D&			
→	inSegmentAlignment	Optional<const CoordinateSystem2D&>		NIL	
→	inWidth	float	1.0 - ∞		Segment width
→	inFrameWidth	int	0 - 65535		
→	inFrameHeight	int	0 - 65535		
→	inRound	bool			Make ends of segment round
←	outRegion	Region&			Output region
←	outAlignedSegment	Optional<Segment2D&>		NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedSegment**.

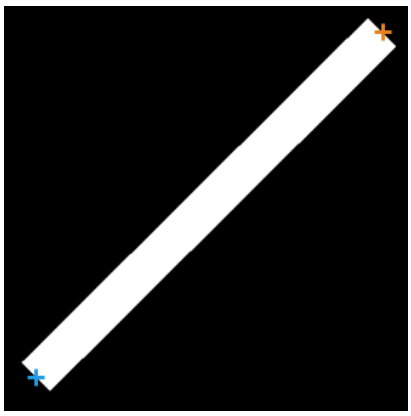
Read more about Optional Outputs.

Description

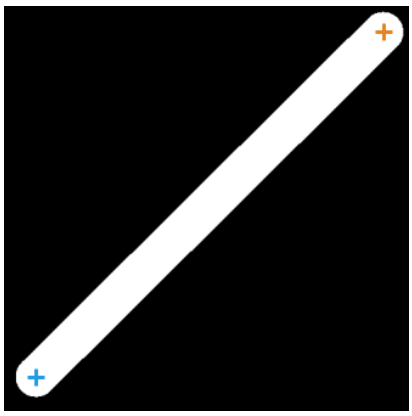
Filter creates region based on an input Segment2D. To specify line width use **inWidth**.

Filter creates segment region by creating rectangle located between **inSegment** points. To make ends round use **inRound**.

Examples



Example region with **inRound**=False



Example region with **inRound**=True

EmptyRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Creates an empty region with a given frame.

Syntax

```
void avl::EmptyRegion
(
 const int inFrameWidth,
 const int inFrameHeight,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inFrameWidth	const int	0 - 65535		Width of the created region's frame
→	inFrameHeight	const int	0 - 65535		Height of the created region's frame
←	outRegion	Region&			Output region

GetRegionFrame

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns the width and height of the entire region's frame (not to be confused with RegionBoundingBox!).

Syntax

```
void avl::GetRegionFrame
(
 const avl::Region& inRegion,
 int& outFrameWidth,
 int& outFrameHeight
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outFrameWidth	int&		
←	outFrameHeight	int&		

LocationsToRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Forms a region containing pixels of given locations.

Syntax

```
void avl::LocationsToRegion
(
 const atl::Array<avl::Location>& inLocations,
 int inFrameWidth,
 int inFrameHeight,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inLocations	const Array<Location>&			
→	inFrameWidth	int	0 - 65535		
→	inFrameHeight	int	0 - 65535		
←	outRegion	Region&			Output region

Description

The operation forms a region equivalent to the given array of pixel locations. If any of the parameters **inFrameWidth**, **inFrameHeight** is not provided, it is set to the smallest value that allows to contain all locations inside of the region.

The **inFrameWidth** and **inFrameHeight** parameters most often should be set equal to the dimensions of the image this region will be used with.

See Also

- RegionToLocations – Converts a region to an array of its pixel locations.

RegionCharacteristicPoint

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns a characteristic point of the input's region bounding box.

Syntax

```
void avl::RegionCharacteristicPoint
(
 const avl::Region& inRegion,
 const avl::Anchor2D::Type inPointAnchor,
 avl::Point2D& outPoint
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inPointAnchor	const Anchor2D::Type	TopLeft	
←	outPoint	Point2D&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty region on input in RegionCharacteristicPoint.

RegionToLocations

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a region to an array of its pixel locations.

Applications: It may be considered explicit region decompression.

Syntax

```
void avl::RegionToLocations
(
 const avl::Region& inRegion,
 atl::Array<avl::Location>& outLocations
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outLocations	Array<Location>&		

Description

The operation forms an array of pixel locations equivalent to the given region. Note that the resulting array may occupy much more space than the region because it stores each pixels separately, as opposed to the run-length encoding used in regions.

See Also

- LocationsToRegion – Forms a region containing pixels of given locations.

SetRegionFrame

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Changes the width and the height of a region's frame (but does not rescale the content).

Syntax

```
void avl::SetRegionFrame
(
 avl::Region& ioRegion,
 int inWidth,
 int inHeight
)
```

Parameters

	Name	Type	Range	Default	Description
	ioRegion	Region&			
	inWidth	int	0 - 65535		New frame width
	inHeight	int	0 - 65535		New frame height

Description

The operation sets the region frame's width and height to new values, while it does **not** change the location of any of the region pixels. Note that all of the region pixels have to be contained within region frame's dimensions. If this is not the case with selected dimensions, an error with appropriate description occurs.

Examples

SetRegionFrame *used to change dimensions from 150x150 to 300x300.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Resulting region isn't contained in new dimensions in SetRegionFrame.

See Also

- CropRegion – Creates a region from a rectangular fragment of another one.

SkipEmptyRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

If the input region contains at least one pixel, then it is copied to the output; otherwise Nil is returned.

Applications: Secures against domain errors caused by empty regions, e.g. just before the RegionMassCenter filter is to be invoked.

Syntax

```
void avl::SkipEmptyRegion
(
 const avl::Region& inRegion,
 atl::Conditional<avl::Region>& outNotEmptyRegion,
 bool& outIsNotEmpty
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outNotEmptyRegion	Conditional<Region>&		
←	outIsNotEmpty	bool&		

TestRegionEmpty

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Tests whether the size of a region equals zero.

Syntax

```
void avl::TestRegionEmpty
(
 const avl::Region& inRegion,
 bool& outIsEmpty
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outIsEmpty	bool&		

TestRegionNotEmpty

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Tests whether the size of a region doesn't equal zero.

Syntax

```
void avl::TestRegionNotEmpty
(
 const avl::Region& inRegion,
 bool& outIsNotEmpty
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outIsNotEmpty	bool&		

TestRegionOnBorder

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Tests whether a region is adjacent to its border.

Syntax

```
void avl::TestRegionOnBorder
(
 const avl::Region& inRegion,
 bool& outIsOnBorder
)
```

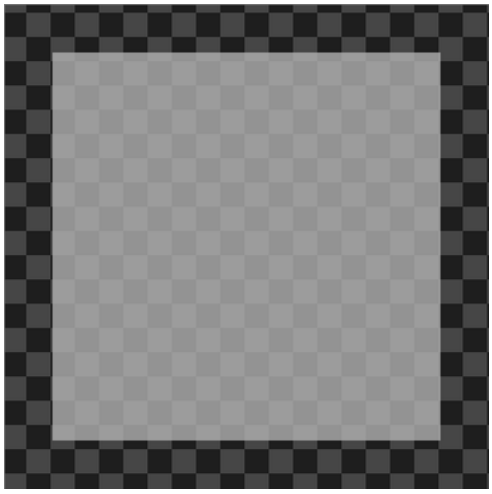
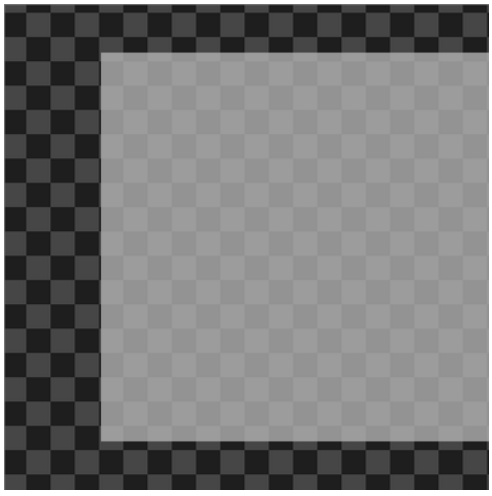
Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outIsOnBorder	bool&		

Description

Tests whether a region is adjacent to its border.

Examples

Filter output	Region preview image
False	
True	

VerifyRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Checks if the point-runs of the input region are correctly arranged (sorted, not overlapping etc.).

Applications: You only need to use this tool if you create regions from point-runs manually.

Syntax

```
void avl::VerifyRegion
(
 const avl::Region& inRegion
)
```

Parameters

	Name	Type	Default	Description
➡	inRegion	const Region&		Input region

Errors

List of possible exceptions:

Error type	Description
DomainError	Input region's dimension is negative in VerifyRegion.
DomainError	Input region's PointRun end has invalid X coordinate in VerifyRegion.
DomainError	Input region's PointRun has invalid Y coordinate in VerifyRegion.
DomainError	Input region's PointRun is empty in VerifyRegion.
DomainError	Input region's PointRuns are not sorted in VerifyRegion.
DomainError	Input region's size is negative in VerifyRegion.

72. Data Classification Common

Table of content:

CreateDataPartition

Header: AVL.h
Namespace: avl
Module: FoundationPro

Divides the input set to test and train subsets, trying to maintain balance in class distribution.

Syntax

```
void avl::CreateDataPartition
(
 const atl::Array<atl::Array<float>>& inFeatureSet,
 const atl::Array<int>& inClassAssignment,
 const float inTestToTrainingRatio,
 const int inRandomSeed,
 atl::Array<atl::Array<float>>& outTrainSet,
 atl::Array<atl::Array<float>>& outTrainResponse,
 atl::Array<atl::Array<float>>& outTestSet,
 atl::Array<atl::Array<float>>& outTestResponse
)
```

Parameters

	Name	Type	Range	Default	Description
→	inFeatureSet	const Array<Array<float>>&			
→	inClassAssignment	const Array<int>&			
→	inTestToTrainingRatio	const float	0.0 - 1.0	0.75f	
→	inRandomSeed	const int		0	
←	outTrainSet	Array<Array<float>>&			
←	outTrainResponse	Array<Array<float>>&			
←	outTestSet	Array<Array<float>>&			
←	outTestResponse	Array<Array<float>>&			

Errors

List of possible exceptions:

Error type	Description
DomainError	Inconsistent size of inFeatureSet and inClassAssignment arrays

MeasureClassificationQuality_Binary

Header: AVL.h
Namespace: avl
Module: FoundationPro

Calculates classification performance metrics for binary problems.

Syntax

```
void avl::MeasureClassificationQuality_Binary
(
 const atl::Array<bool>& inPredictedClasses,
 const atl::Array<bool>& inExpectedClasses,
 float& outAccuracy,
 float& outPrecision,
 float& outRecall,
 float& outF1Score,
 avl::Matrix& outConfusionMatrix
)
```

Parameters

	Name	Type	Default	Description
→	inPredictedClasses	const Array<bool>&		
→	inExpectedClasses	const Array<bool>&		
←	outAccuracy	float&		
←	outPrecision	float&		
←	outRecall	float&		
←	outF1Score	float&		
←	outConfusionMatrix	Matrix&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array on input in MeasureClassificationQuality_Binary.
DomainError	Inconsistent array sizes on input in MeasureClassificationQuality_Binary.

MeasureClassificationQuality_Multiclass

Header: AVL.h
Namespace: avl
Module: FoundationPro

Calculates classification performance metrics for multiclass problems.

Syntax

```
void avl::MeasureClassificationQuality_Multiclass
(
 const atl::Array<int>& inPredictedClasses,
 const atl::Array<int>& inExpectedClasses,
 float& outAccuracy,
 atl::Array<float>& outPrecisions,
 atl::Array<float>& outRecalls,
 atl::Array<float>& outF1Scores,
 avl::Matrix& outConfusionMatrix
)
```

Parameters

	Name	Type	Default	Description
→	inPredictedClasses	const Array<int>&		
→	inExpectedClasses	const Array<int>&		
←	outAccuracy	float&		
←	outPrecisions	Array<float>&		
←	outRecalls	Array<float>&		
←	outF1Scores	Array<float>&		
←	outConfusionMatrix	Matrix&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Class designator must start at 0.
DomainError	Empty array on input in MeasureClassificationQuality_Multiclass.
DomainError	Inconsistent array sizes on input in MeasureClassificationQuality_Multiclass.

73. Image Tiling

Table of content:

CreateImageTiles

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Generates an array of regions and an array of boxes covering the area of an image.

Syntax

```
void avl::CreateImageTiles
(
 const avl::Size& inImageSize,
 const int inTileWidth,
 atl::Optional<int> inTileHeight,
 atl::Optional<int> inHorizontalStep,
 atl::Optional<int> inVerticalStep,
 const avl::OverflowControl::Type inOverflowControl,
 atl::Array<avl::Region>& outTileRegions,
 atl::Array<avl::Box>& outTileBoxes,
 int& outRowCount,
 int& outColumnCount
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImageSize	const Size&			Format of image for which tiles will be produced.
→	inTileWidth	const int	1 - ∞	32	Demanded tile width.
→	inTileHeight	Optional<int>	1 - ∞	NIL	Demanded tile height; equals inTileWidth when set to Auto.
→	inHorizontalStep	Optional<int>	1 - ∞	NIL	Defines horizontal space between consecutive tiles; defaults to tile width. Can be used to produce overlapping tiles.
→	inVerticalStep	Optional<int>	1 - ∞	NIL	Defines vertical space between consecutive tiles; defaults to tile height. Can be used to produce overlapping tiles.
→	inOverflowControl	const OverflowControl::Type		KeepLast	Define what to do when overflowing tiles are present.
←	outTileRegions	Array<Region>&			Array containing produced tiles.
←	outTileBoxes	Array<Box>&			Array containing produced tiles.
←	outRowCount	int&			Number of generated tiles rows.
←	outColumnCount	int&			Number of generated tiles per row.

CreateImageTiles_AsBoxes

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Generates an array of boxes covering the area of an image.

Syntax

```
void avl::CreateImageTiles_AsBoxes
(
 const avl::Size& inImageSize,
 const int inTileWidth,
 atl::Optional<int> inTileHeight,
 atl::Optional<int> inHorizontalStep,
 atl::Optional<int> inVerticalStep,
 const avl::OverflowControl::Type inOverflowControl,
 atl::Array<avl::Box>& outTileBoxes,
 int& outRowCount,
 int& outColumnCount
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImageSize	const Size&			Format of image for which tiles will be produced.
→	inTileWidth	const int	1 - ∞	32	Demanded tile width.
→	inTileHeight	Optional<int>	1 - ∞	NIL	Demanded tile height; equals inTileWidth when set to Auto.
→	inHorizontalStep	Optional<int>	1 - ∞	NIL	Defines horizontal space between consecutive tiles; defaults to tile width. Can be used to produce overlapping tiles.
→	inVerticalStep	Optional<int>	1 - ∞	NIL	Defines vertical space between consecutive tiles; defaults to tile height. Can be used to produce overlapping tiles.
→	inOverflowControl	const OverflowControl::Type		KeepLast	Define what to do when overflowing tiles are present.
←	outTileBoxes	Array<Box>&			Array containing produced tiles.
←	outRowCount	int&			Number of generated tiles rows.
←	outColumnCount	int&			Number of generated tiles per row.

Description

Creates an array of boxes.

Examples

***CreateImageTiles_asBoxes** with inTileWidth = 50 and inOverflowControl = true.*

See Also

- JoinImageTiles – Joins previously cut tiles into single image.
- CutImageIntoTiles – Generates an array of small images by cutting the input image.

CutImageIntoTiles

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Generates an array of small images by cutting the input image.

Syntax

```
void avl::CutImageIntoTiles
(
 const avl::Image& inImage,
 const int inTileWidth,
 atl::Optional<int> inTileHeight,
 atl::Optional<int> inHorizontalStep,
 atl::Optional<int> inVerticalStep,
 const avl::OverflowControl::Type inOverflowControl,
 atl::Array<avl::Image>& outImageTiles
)
```

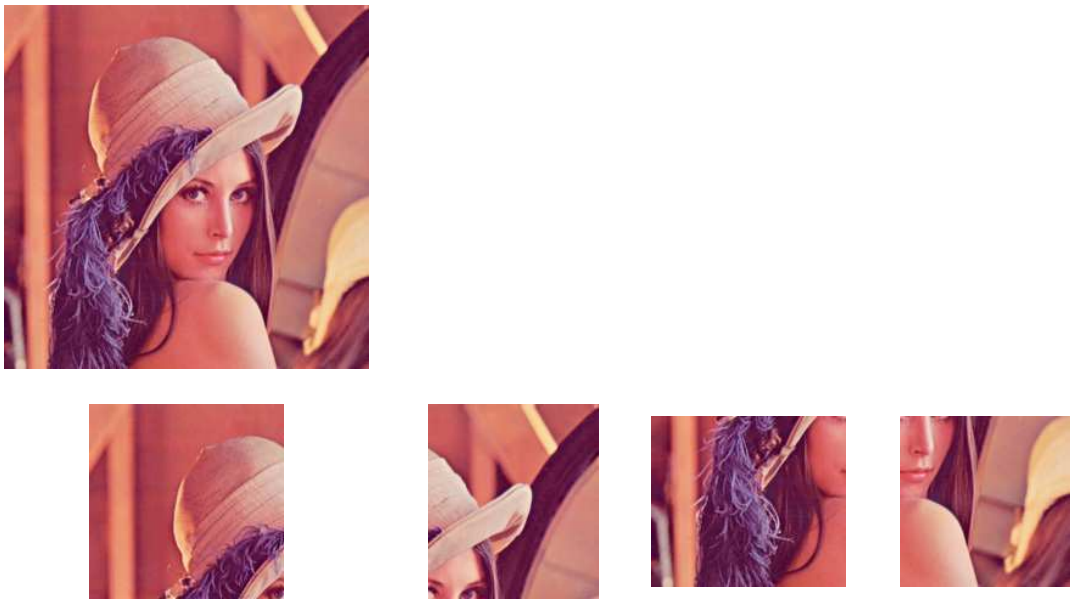
Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image to be cut into tiles
→	inTileWidth	const int	1 - ∞	1	Demanded tile width.
→	inTileHeight	Optional<int>	1 - ∞	NIL	Demanded tile height; equals inTileWidth when set to Auto.
→	inHorizontalStep	Optional<int>	1 - ∞	NIL	Defines horizontal space between consecutive tiles; defaults to tile width. Can be used to produce overlapping tiles.
→	inVerticalStep	Optional<int>	1 - ∞	NIL	Defines vertical space between consecutive tiles; defaults to tile height. Can be used to produce overlapping tiles.
→	inOverflowControl	const OverflowControl::Type			Define what to do when overflowing tiles are present.
←	outImageTiles	Array<Image>&			Resulting image tiles

Description

Generates an array of image tiles, which are cut from inImage.

Examples



CutImageIntoTiles *performed on the sample image with* ***inOverflowControl*** *= false.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- JoinImageTiles – Joins previously cut tiles into single image.

EnumerateImageTiles

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Enumerates tiles from image.

Syntax

```
bool avl::EnumerateImageTiles
(
 EnumerateImageTilesState& ioState,
 const avl::Image& inImage,
 int inTileWidth,
 atl::Optional<int> inTileHeight,
 bool inOverflowControl,
 const int inDelay,
 avl::Image& outTile,
 atl::Optional<bool&> outIsFirst = atl::NIL,
 atl::Optional<bool&> outIsLast = atl::NIL,
 avl::Box& diagTilePosition = atl::Dummy<avl::Box>()
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	EnumerateImageTilesState&			Object used to maintain state of the function.
	inImage	const Image&			Image to be enumerated
	inTileWidth	int	1 - ∞	1	Tile width
	inTileHeight	Optional<int>	1 - ∞	NIL	Tile height
	inOverflowControl	bool			Allow overflow
	inDelay	const int			Minimum time between iterations in milliseconds
	outTile	Image&			
	outIsFirst	Optional<bool&>		NIL	Flag indicating the first iteration
	outIsLast	Optional<bool&>		NIL	Flag indicating the last iteration
	diagTilePosition	Box&			Cut out tile position.

Optional Outputs

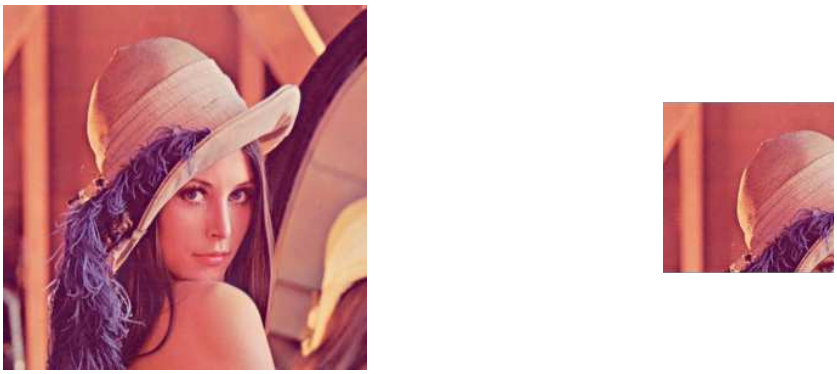
The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outIsFirst**, **outIsLast**.

Read more about Optional Outputs.

Description

Cuts out a tile from inImage and serves them one by one. Tile dimensions cannot be changed during image traversal.

Examples



EnumerateImageTiles *performed on the sample image with* ***inOverflowControl*** *= true.*

See Also

- JoinImageTiles – Joins previously cut tiles into single image.
- CutImageIntoTiles – Generates an array of small images by cutting the input image.
- JoinImageTiles – Joins previously cut tiles into single image.

JoinImageTiles

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Joins previously cut tiles into single image.

Syntax

```
void avl::JoinImageTiles
(
 const atl::Array<avl::Image>& inImages,
 const int inRowCount,
 const int inColumnCount,
 avl::Image& outJoinedImage
)
```

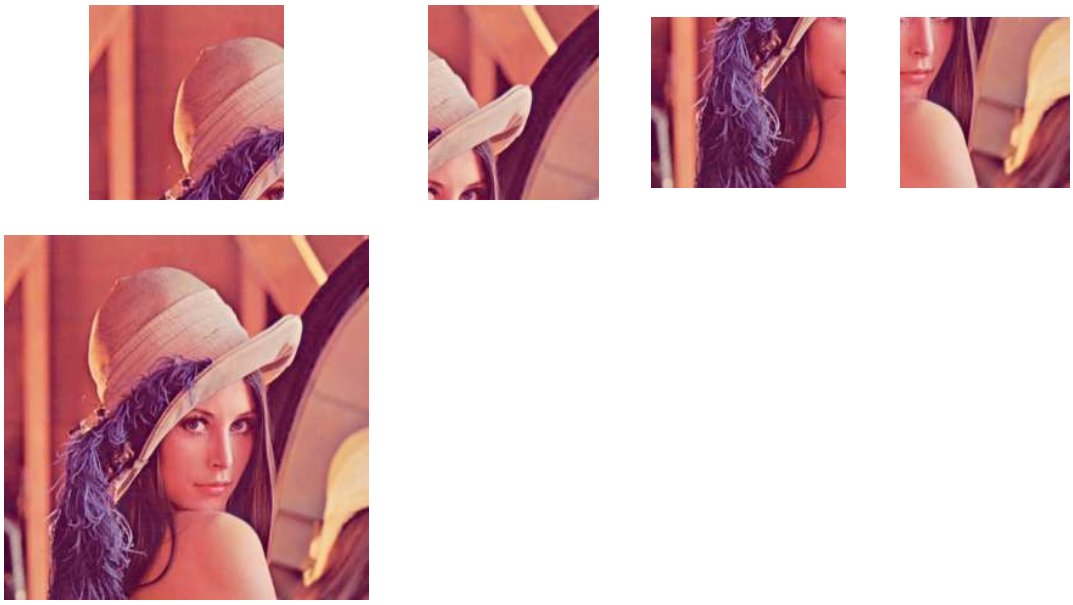
Parameters

	Name	Type	Range	Default	Description
→	inImages	const Array<Image>&			Array of image tiles.
→	inRowCount	const int	1 - ∞		Defines how many output image rows there are in inImages.
→	inColumnCount	const int	1 - ∞		Defines how many images builds one row in inImages.
←	outJoinedImage	Image&			Glued image.

Description

Joins an array of images into one image.

Examples



JoinImageTiles *glues together an array of images with inRowCount = 2 and inColumnCount = 2.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Input images do not have compatible height in JoinImageTiles.
DomainError	Input images do not have compatible width in JoinImageTiles.
DomainError	Input images do not have the same pixel format in JoinImageTiles.
DomainError	Product of inRowCount and inColumnCount must be equal to size of inImages array in JoinImageTiles.

See Also

- CutImageIntoTiles – Generates an array of small images by cutting the input image.

74. Geometry 2D Relations

Table of content:

CreatePointGraph

Header: AVL.h
Namespace: avl
Module: FoundationPro

Creates a graph of points that lie close to each other.

Syntax

```
void avl::CreatePointGraph
(
 const atl::Array<avl::Point2D>& inPoints,
 const int inMaxRank,
 float inMinDistance,
 atl::Optional<float> inMaxDistance,
 atl::Optional<float> inMaxRelativeDistance,
 atl::Array<atl::Array<int> >& outGraph,
 atl::Array<atl::Array<float> >& outDistances,
 atl::Array<avl::Segment2D>& diagSegments
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoints	const Array<Point2D>&			Input points
→	inMaxRank	const int	1 - ∞	8	Maximum connections going out of one point
→	inMinDistance	float	0.0 - ∞	0.0f	Minimum distance between adjacent point in the graph
→	inMaxDistance	Optional<float>	0.0 - ∞	10.0f	Maximum distance between adjacent point in the graph
→	inMaxRelativeDistance	Optional<float>	1.0 - ∞	NIL	Maximum distance in relation to the shortest distance for a point
←	outGraph	Array<Array<int> >&			Graph of points (adjacency list)
←	outDistances	Array<Array<float> >&			Distances between adjacent points in the graph
🔍	diagSegments	Array<Segment2D>&			Graph edges, useful for visualization

Errors

List of possible exceptions:

Error type	Description
DomainError	Minimum distance is greater than maximum distance in CreatePointGraph.

FindClosestPoints

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

For each given point, finds the closest point among the given point set.

Syntax

```
void avl::FindClosestPoints
(
 const atl::Array<avl::Point2D>& inPointSet,
 const atl::Array<avl::Point2D>& inQueries,
 atl::Array<avl::Point2D>& outPoints,
 atl::Array<int>& outIndices
)
```

Parameters

	Name	Type	Default	Description
→	inPointSet	const Array<Point2D>&		Points that will be searched.
→	inQueries	const Array<Point2D>&		
←	outPoints	Array<Point2D>&		For each point in inQueries, the closest point value from inPointSet.
←	outIndices	Array<int>&		For each point in inQueries, the closest point index of inPointSet.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

TestPointArrayInBox

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Tests which points lie inside a box.

Syntax

```
void avl::TestPointArrayInBox
(
 const atl::Array<avl::Point2D>& inPoints,
 const avl::Box& inBox,
 atl::Array<bool>& outIsContainedArray,
 atl::Optional<atl::Array<avl::Point2D>&> outPoints = atl::NIL,
 atl::Optional<bool&> outAreAllContained = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		Points which will be tested
→	inBox	const Box&		
←	outIsContainedArray	Array<bool>&		
←	outPoints	Optional<Array<Point2D>&>	NIL	Points that are contained
←	outAreAllContained	Optional<bool&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outPoints**, **outAreAllContained**.

Read more about Optional Outputs.

TestPointArrayInRectangle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Tests which points lie inside a rectangle.

Syntax

```
void avl::TestPointArrayInRectangle
(
 const atl::Array<avl::Point2D>& inPoints,
 const avl::Rectangle2D& inRectangle,
 atl::Array<bool>& outIsContainedArray,
 atl::Optional<atl::Array<avl::Point2D>&> outPoints = atl::NIL,
 atl::Optional<bool&> outAreAllContained = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		Points which will be tested
→	inRectangle	const Rectangle2D&		
←	outIsContainedArray	Array<bool>&		
←	outPoints	Optional<Array<Point2D>&>	NIL	Points that are contained
←	outAreAllContained	Optional<bool&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outPoints**, **outAreAllContained**.

Read more about Optional Outputs.

TestPointInBox

Header: AVL.h
Namespace: avl
Module: FoundationLite

Tests whether a point lies in a box.

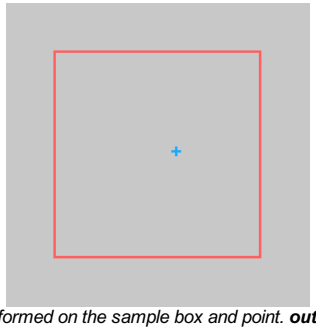
Syntax

```
void avl::TestPointInBox
(
 const avl::Point2D& inPoint,
 const avl::Box& inBox,
 bool& outIsContained
)
```

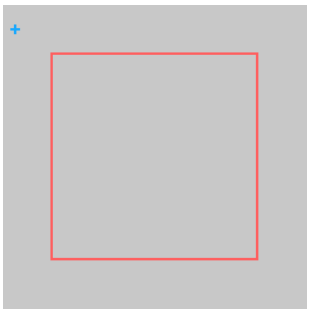
Parameters

	Name	Type	Default	Description
→	inPoint	const Point2D&		
→	inBox	const Box&		
←	outIsContained	bool&		

Examples



TestPointInBox *performed on the sample box and point.* ***outIsContained*** *= True.*



TestPointInBox *performed on the sample box and point.* ***outIsContained*** *= False.*

TestPointInCircle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Tests whether a point lies inside a circle.

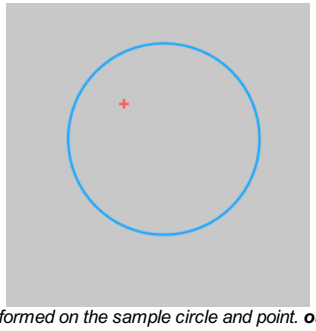
Syntax

```
void avl::TestPointInCircle
(
 const avl::Point2D& inPoint,
 const avl::Circle2D& inCircle,
 bool& outIsContained
)
```

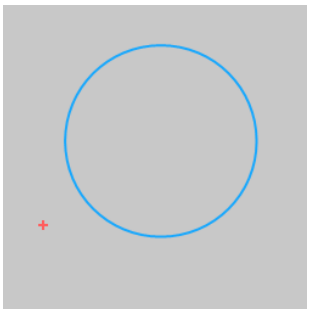
Parameters

	Name	Type	Default	Description
→	inPoint	const Point2D&		
→	inCircle	const Circle2D&		
←	outIsContained	bool&		

Examples



TestPointInCircle *performed on the sample circle and point.* ***outIsContained*** *= True.*



TestPointInCircle *performed on the sample circle and point.* ***outIsContained*** *= False.*

TestPointInRectangle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Tests whether a point lies in a rectangle.

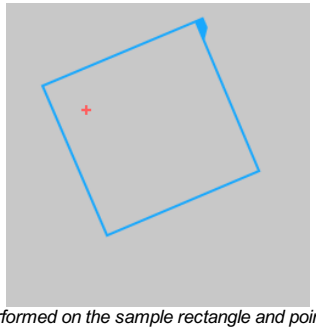
Syntax

```
void avl::TestPointInRectangle
(
 const avl::Point2D& inPoint,
 const avl::Rectangle2D& inRectangle,
 bool& outIsContained
)
```

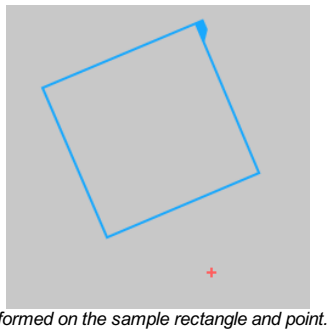
Parameters

	Name	Type	Default	Description
→	inPoint	const Point2D&		
→	inRectangle	const Rectangle2D&		
←	outIsContained	bool&		

Examples



TestPointInRectangle *performed on the sample rectangle and point.* ***outIsContained*** *= True.*



TestPointInRectangle *performed on the sample rectangle and point.* ***outIsContained*** *= False.*

75. 1D Edge Detection

Table of content:

CreateScanMap

Header: AVL.h
Namespace: avl
Module: MetrologyBasic

Precomputes a data object that is required for fast 1D edge detection.

Applications: Used together with 1D Edge Detection filters (excluding "_Direct"), but can be moved before the loop if only the scan parameters do not change.

Syntax

```
void avl::CreateScanMap
(
 const avl::ImageFormat& inImageFormat,
 const avl::Path& inScanPath,
 atl::Optional<const avl::CoordinateSystem2D&> inScanPathAlignment,
 int inScanWidth,
 const avl::SamplingParams& inSamplingParams,
 avl::ScanMap& outScanMap,
 atl::Optional<avl::Path&> outAlignedScanPath = atl::NIL,
 atl::Array<avl::Path>& diagSamplingPoints,
 float& diagSamplingStep
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImageFormat	const ImageFormat&			Dimensions, depth and pixel type of the image on which edge detection will be performed
→	inScanPath	const Path&			Path along which the scan is performed
→	inScanPathAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the scan path to the position of the inspected object
→	inScanWidth	int	1 - ∞	5	Width of the scan field in pixels
→	inSamplingParams	const SamplingParams&		.interpolation Bilinear	Parameters controlling the sampling process
←	outScanMap	ScanMap&			Optimized data object required for 1D edge detection
←	outAlignedScanPath	Optional<Path&>		NIL	Transformed input path
🔍	diagSamplingPoints	Array<Path>&			Array of paths each one containing the sampling points that contributes to a single value of the extracted profile
🔍	diagSamplingStep	float&			Used distance between consecutive sampling points on the scan path

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedScanPath**.

Read more about Optional Outputs.

Description

The operation creates a scan map from a given **inScanPath**. The scan map can be later used by other 1D Edge Detection filters.

The optional parameter **inScanPathAlignment** defines a transform to be performed on the **inScanPath** so that the actual scan path (**outAlignedScanPath**) is adjusted to the position of the object, typically detected by one of Template Matching filters.

This filter is a part of the 1D Edge Detection toolset. For a comprehensive introduction to this technique please refer to 1D Edge Detection and 1D Edge Detection - Subpixel Precision chapters of our Machine Vision Guide.

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

See Also

- ScanSingleEdge – Locates the strongest transition between dark and bright pixels along a given path.
- ScanSingleRidge – Locates the strongest dark or bright pixel peak along a given path.
- ScanSingleStripe – Locates the strongest pair of edges across a given path.

ScanExactlyNEdges

Header: AVL.h
Namespace: avl
Module: MetrologyBasic

Locates a specified number of the strongest transitions between dark and bright pixels along a given path.

Applications: Very fast object detection (or presence verification) when the expected number of edges is clearly defined.

Syntax

```
void avl::ScanExactlyNEdges
(
 const avl::Image& inImage,
 const ScanMap& inScanMap,
 const EdgeScanParams& inEdgeScanParams,
 int inEdgeCount,
 avl::Selection::Type inEdgeSelection,
 float inMinDistance,
 atl::Optional<float> inMaxDistance,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Conditional<atl::Array<avl::Edge1D> >& outEdges,
 atl::Conditional<atl::Array<avl::Gap1D> >& outGaps,
 atl::Optional<avl::Profile&> outBrightnessProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inScanMap	const ScanMap&			Data precomputed with CreateScanMap
→	inEdgeScanParams	const EdgeScanParams&		EdgeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f MinMagnitude: 5.0f EdgeTransition: BrightToDark)	Parameters controlling the edge extraction process
→	inEdgeCount	int	0 - ∞	1	Number of edges to be found
→	inEdgeSelection	Selection::Type		Selection::Best	Selection mode of the resulting edges
→	inMinDistance	float	0.0 - ∞	0.0f	Minimal distance between consecutive edges
→	inMaxDistance	Optional<float>	0.0 - ∞	NIL	Maximal distance between consecutive edges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
←	outEdges	Conditional<Array<Edge1D> >&			Found edges
←	outGaps	Conditional<Array<Gap1D> >&			Gaps between consecutive edges
←	outBrightnessProfile	Optional<Profile&>		NIL	Extracted image profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the edge (derivative) operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outBrightnessProfile**, **outResponseProfile**.

Read more about Optional Outputs.

Description

The operation scans the image using **inScanMap** previously generated from a scan path and finds a set of **inEdgeCount** image edges perpendicular to the path. If no subset (of **inEdgeCount** elements) of detected edges meets the requirements of **inEdgeScanParams.minMagnitude**, **inMinDistance**, **inEdgeScanParams.edgeTransition** then the outputs are set to NIL.

Note that in case of a scan path which is closed, the parameters controlling the distances between consecutive found objects do not control the distance between the first and the last of the found objects (counting from the beginning of the scan path).

Hints

- Set **inEdgeCount** to the number of edges that are to be found (the N number).
- Define **inEdgeScanParams.EdgeTransition** to detect a particular edge type, and only that type.
- If the expected number of edges cannot be found, try decreasing **inEdgeScanParams.MinMagnitude**. Verify this with the values on the **outResponseProfile** output.
- If consecutive edges are closer than 6 pixels apart, change **inEdgeScanParams.ProfileInterpolation** to **Quadratic3**.
- Adjust **inMinDistance** (in pixels) to filter out false edges that appear in proximity to other edges.

Examples

ScanExactlyNEdges *locates the edges using a scan map representing the scan path above (**inEdgeCount** = 61).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the 1D Edge Detection toolset. For a comprehensive introduction to this technique please refer to 1D Edge Detection and 1D Edge Detection - Subpixel Precision chapters of our Machine Vision Guide.

See Also

- CreateScanMap – Precomputes a data object that is required for fast 1D edge detection.
- ScanSingleEdge – Locates the strongest transition between dark and bright pixels along a given path.
- ScanMultipleEdges – Locates multiple transitions between dark and bright pixels along a given path.
- ScanExactlyNEdges_Direct – Locates a specified number of the strongest transitions between dark and bright pixels along a given path (without a scan map).

ScanExactlyNEdges_Direct

Header: AVL.h
Namespace: avl
Module: MetrologyBasic

Locates a specified number of the strongest transitions between dark and bright pixels along a given path (without a scan map).

Applications: Very fast object detection (or presence verification) when the expected number of edges is clearly defined.

Syntax

```
void avl::ScanExactlyNEdges_Direct
(
 const avl::Image& inImage,
 const avl::Path& inScanPath,
 atl::Optional<const avl::CoordinateSystem2D&> inScanPathAlignment,
 int inScanWidth,
 const avl::SamplingParams& inSamplingParams,
 const avl::EdgeScanParams& inEdgeScanParams,
 int inEdgeCount,
 avl::Selection::Type inEdgeSelection,
 float inMinDistance,
 atl::Optional<float> inMaxDistance,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Conditional<atl::Array<avl::Edge1D> >& outEdges,
 atl::Conditional<atl::Array<avl::Gap1D> >& outGaps,
 atl::Optional<avl::Path&> outAlignedScanPath = atl::NIL,
 atl::Optional<avl::Profile&> outBrightnessProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL,
 atl::Array<avl::Path>& diagSamplingPoints,
 float& diagSamplingStep
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inScanPath	const Path&			Path along which the scan is performed
→	inScanPathAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the scan path to the position of the inspected object
→	inScanWidth	int	1 - ∞	5	Width of the scan field in pixels
→	inSamplingParams	const SamplingParams&		SamplingParams (Interpolation: Bilinear SamplingStep: 1.0f SampleCount: Nil)	Parameters controlling the sampling process
→	inEdgeScanParams	const EdgeScanParams&		EdgeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f MinMagnitude: 5.0f EdgeTransition: BrightToDark)	Parameters controlling the edge extraction process
→	inEdgeCount	int	0 - ∞	1	Number of edges to be found
→	inEdgeSelection	Selection::Type			Selection mode of the resulting edges
→	inMinDistance	float	0.0 - ∞	0.0f	Minimal distance between consecutive edges
→	inMaxDistance	Optional<float>	0.0 - ∞	NIL	Maximal distance between consecutive edges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
←	outEdges	Conditional<Array<Edge1D> >&			Found edges
←	outGaps	Conditional<Array<Gap1D> >&			Gaps between consecutive edges
←	outAlignedScanPath	Optional<Path&>		NIL	Transformed input path
←	outBrightnessProfile	Optional<Profile&>		NIL	Extracted image profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the edge (derivative) operator response
🔍	diagSamplingPoints	Array<Path>&			Array of paths each one containing the sampling points that contributes to a single value of the extracted profile
🔍	diagSamplingStep	float&			Used distance between consecutive sampling points on the scan path

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedScanPath**, **outBrightnessProfile**, **outResponseProfile**.

Read more about Optional Outputs.

Description

The operation scans the image along **inScanPath** and finds a set of **inEdgeCount** image edges perpendicular to the path. If no subset (of **inEdgeCount** elements) of detected edges meets the requirements of **inEdgeScanParams.minMagnitude**, **inMinDistance**, **inEdgeScanParams.edgeTransition** then the outputs are set to NIL.

The optional parameter **inScanPathAlignment** defines a transform to be performed on the **inScanPath** so that the actual scan path (**outAlignedScanPath**) is adjusted to the position of the object, typically detected by one of Template Matching filters.

Note that in case of a scan path which is closed, the parameters controlling the distances between consecutive found objects do not control the distance between the first and the last of the found objects (counting from the beginning of the scan path).

Hints

- Set **inEdgeCount** to the number of edges that are to be found (the N number).
- Define **inEdgeScanParams.EdgeTransition** to detect a particular edge type, and only that type.
- If the expected number of edges cannot be found, try decreasing **inEdgeScanParams.MinMagnitude**. Verify this with the values on the **outResponseProfile** output.
- If consecutive edges are closer than 6 pixels apart, change **inEdgeScanParams.ProfileInterpolation** to **Quadratic3**.
- Adjust **inMinDistance** (in pixels) to filter out false edges that appear in proximity to other edges.

Examples

ScanExactlyNEdges_Direct *locates the edges across* ***inScanPath*** *(****inEdgeCount*** *= 61).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the 1D Edge Detection toolset. For a comprehensive introduction to this technique please refer to 1D Edge Detection and 1D Edge Detection - Subpixel Precision chapters of our Machine Vision Guide.

See Also

- CreateScanMap – Precomputes a data object that is required for fast 1D edge detection.
- ScanSingleEdge_Direct – Locates the strongest transition between dark and bright pixels along a given path (without a scan map).
- ScanMultipleEdges_Direct – Locates multiple transitions between dark and bright pixels along a given path (without a scan map).
- ScanExactlyNEdges – Locates a specified number of the strongest transitions between dark and bright pixels along a given path.

ScanExactlyNRidges

Header: AVL.h
Namespace: avl
Module: MetrologyBasic

Locates a specified number of the strongest dark or bright pixel peak along a given path.

Applications: Very fast detection (or presence verification) of thin structures like wires or scale marks.

Syntax

```
void avl::ScanExactlyNRidges
(
 const avl::Image& inImage,
 const ScanMap& inScanMap,
 const RidgeScanParams& inRidgeScanParams,
 int inRidgeCount,
 avl::Selection::Type inRidgeSelection,
 float inMinDistance,
 atl::Optional<float> inMaxDistance,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Conditional<atl::Array<avl::Ridge1D> >& outRidges,
 atl::Conditional<atl::Array<avl::Gap1D> >& outGaps,
 atl::Optional<avl::Profile&> outBrightnessProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inScanMap	const ScanMap&			Data precomputed with CreateScanMap
→	inRidgeScanParams	const RidgeScanParams&		RidgeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f RidgeWidth: 5 RidgeMargin: 2 MinMagnitude: 5.0f RidgePolarity: Dark)	Parameters controlling the ridge extraction process
→	inRidgeCount	int	0 - ∞	1	Number of ridges to be found
→	inRidgeSelection	Selection::Type		Selection::Best	Selection mode of the resulting ridges
→	inMinDistance	float	0.0 - ∞	0.0f	Minimal distance between consecutive ridges
→	inMaxDistance	Optional<float>	0.0 - ∞	NIL	Maximal distance between consecutive ridges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker ridges can be detected in the vicinity of stronger ridges
←	outRidges	Conditional<Array<Ridge1D> >&			Found ridges
←	outGaps	Conditional<Array<Gap1D> >&			Gaps between consecutive edges
←	outBrightnessProfile	Optional<Profile&>		NIL	Extracted image profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the ridge operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outBrightnessProfile**, **outResponseProfile**.

Read more about Optional Outputs.

Description

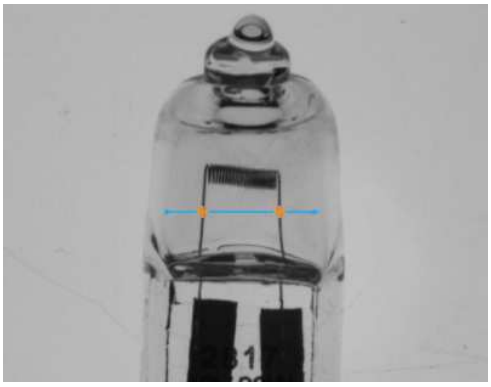
The operation scans the image using **inScanMap** previously generated from a scan path and finds a set of **inRidgeCount** image ridges perpendicular to the path. If no subset (of **inRidgeCount** elements) of detected ridges meets the requirements of **inRidgeScanParams.minMagnitude**, **inMinDistance**, **inRidgeScanParams.ridgePolarity** then the outputs are set to NIL.

Note that in case of a scan path which is closed, the parameters controlling the distances between consecutive found objects do not control the distance between the first and the last of the found objects (counting from the beginning of the scan path).

Hints

- Set **inRidgeCount** to the number of ridges that are to be found (the N number).
- Define **inEdgeScanParams.RidgePolarity** to detect a particular ridge type, and only that type.
- If the expected number of ridges cannot be found, try decreasing **inRidgeScanParams.MinMagnitude**. Verify this with the values on the **outResponseProfile** output.
- If consecutive ridges are very close to each other, change **inEdgeScanParams.ProfileInterpolation** to **Quadratic3**.
- Adjust **inMinDistance** (in pixels) to filter out false ridges that appear in proximity to other ridges.

Examples



ScanMultipleRidges *locates the ridges using a scan map representing the scan path above.*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the 1D Edge Detection toolset. For a comprehensive introduction to this technique please refer to 1D Edge Detection and 1D Edge Detection - Subpixel Precision chapters of our Machine Vision Guide.

See Also

- CreateScanMap – Precomputes a data object that is required for fast 1D edge detection.
- ScanSingleRidge – Locates the strongest dark or bright pixel peak along a given path.
- ScanExactlyNRidges – Locates a specified number of the strongest dark or bright pixel peak along a given path.
- ScanMultipleRidges_Direct – Locates multiple dark or bright pixel peaks along a given path (without a scan map).

ScanExactlyNRidges_Direct

Header: AVL.h
Namespace: avl
Module: MetrologyBasic

Locates a specified number of the strongest dark or bright pixel peak along a given path (without a scan map).

Applications: Very fast detection (or presence verification) of thin structures like wires or scale marks.

Syntax

```
void avl::ScanExactlyNRidges_Direct
(
 const avl::Image& inImage,
 const avl::Path& inScanPath,
 atl::Optional<const avl::CoordinateSystem2D&> inScanPathAlignment,
 int inScanWidth,
 const avl::SamplingParams& inSamplingParams,
 const avl::RidgeScanParams& inRidgeScanParams,
 int inRidgeCount,
 avl::Selection::Type inRidgeSelection,
 float inMinDistance,
 atl::Optional<float> inMaxDistance,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Conditional<atl::Array<avl::Ridge1D> >& outRidges,
 atl::Conditional<atl::Array<avl::Gap1D> >& outGaps,
 atl::Optional<avl::Path&> outAlignedScanPath = atl::NIL,
 atl::Optional<avl::Profile&> outBrightnessProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL,
 atl::Array<avl::Path>& diagSamplingPoints,
 float& diagSamplingStep
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inScanPath	const Path&			Path along which the scan is performed
→	inScanPathAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the scan path to the position of the inspected object
→	inScanWidth	int	1 - ∞	5	Width of the scan field in pixels
→	inSamplingParams	const SamplingParams&		SamplingParams (Interpolation: Bilinear SamplingStep: 1.0f SampleCount: Nil)	Parameters controlling the sampling process
→	inRidgeScanParams	const RidgeScanParams&		RidgeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f RidgeWidth: 5 RidgeMargin: 2 MinMagnitude: 5.0f RidgePolarity: Dark)	Parameters controlling the ridge extraction process
→	inRidgeCount	int	0 - ∞	1	Number of ridges to be found
→	inRidgeSelection	Selection::Type			Selection mode of the resulting ridges
→	inMinDistance	float	0.0 - ∞	0.0f	Minimal distance between consecutive ridges
→	inMaxDistance	Optional<float>	0.0 - ∞	NIL	Maximal distance between consecutive ridges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker ridges can be detected in the vicinity of stronger ridges
←	outRidges	Conditional<Array<Ridge1D> >&			Found ridges
←	outGaps	Conditional<Array<Gap1D> >&			Gaps between consecutive edges
←	outAlignedScanPath	Optional<Path&>		NIL	Transformed input path
←	outBrightnessProfile	Optional<Profile&>		NIL	Extracted image profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the ridge operator response
🔍	diagSamplingPoints	Array<Path>&			Array of paths each one containing the sampling points that contributes to a single value of the extracted profile
🔍	diagSamplingStep	float&			Used distance between consecutive sampling points on the scan path

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedScanPath**, **outBrightnessProfile**, **outResponseProfile**.

Read more about Optional Outputs.

Description

The operation scans the image along **inScanPath** and finds a set of **inRidgeCount** image ridges perpendicular to the path. If no subset (of **inRidgeCount** elements) of detected ridges meets the requirements of **inRidgeScanParams.minMagnitude**, **inMinDistance**, **inRidgeScanParams.ridgePolarity** then the outputs are set to NIL.

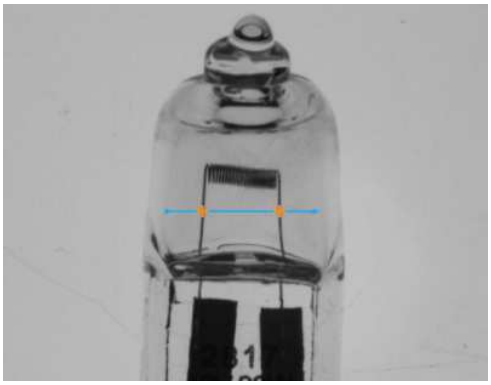
The optional parameter **inScanPathAlignment** defines a transform to be performed on the **inScanPath** so that the actual scan path (**outAlignedScanPath**) is adjusted to the position of the object, typically detected by one of Template Matching filters.

Note that in case of a scan path which is closed, the parameters controlling the distances between consecutive found objects do not control the distance between the first and the last of the found objects (counting from the beginning of the scan path).

Hints

- Set **inRidgeCount** to the number of ridges that are to be found (the N number).
- Define **inEdgeScanParams.RidgePolarity** to detect a particular ridge type, and only that type.
- If the expected number of ridges cannot be found, try decreasing **inRidgeScanParams.MinMagnitude**. Verify this with the values on the **outResponseProfile** output.
- If consecutive ridges are very close to each other, change **inEdgeScanParams.ProfileInterpolation** to **Quadratic3**.
- Adjust **inMinDistance** (in pixels) to filter out false ridges that appear in proximity to other ridges.

Examples



ScanMultipleRidges_Direct *locates the ridges across* ***inScanPath****.*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the 1D Edge Detection toolset. For a comprehensive introduction to this technique please refer to 1D Edge Detection and 1D Edge Detection - Subpixel Precision chapters of our Machine Vision Guide.

See Also

- CreateScanMap – Precomputes a data object that is required for fast 1D edge detection.
- ScanSingleRidge_Direct – Locates the strongest dark or bright pixel peak along a given path (without a scan map).
- ScanExactlyNRidges_Direct – Locates a specified number of the strongest dark or bright pixel peak along a given path (without a scan map).
- ScanMultipleRidges – Locates multiple dark or bright pixel peaks along a given path.

ScanExactlyNStripes

Header: AVL.h
Namespace: avl
Module: MetrologyBasic

Locates a specified number of multiple pairs of opposite edges across a given path.

Applications: Very fast detection (or presence verification) of multiple pairs of opposite edges.

Syntax

```
void avl::ScanExactlyNStripes
(
 const avl::Image& inImage,
 const ScanMap& inScanMap,
 const StripeScanParams& inStripeScanParams,
 int inStripeCount,
 avl::Selection::Type inStripeSelection,
 float inMinGapWidth,
 atl::Optional<float> inMaxGapWidth,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Conditional<atl::Array<avl::Stripe1D> >& outStripes,
 atl::Conditional<atl::Array<avl::Gap1D> >& outGaps,
 atl::Optional<avl::Profile&> outBrightnessProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inScanMap	const ScanMap&			Data precomputed with CreateScanMap
→	inStripeScanParams	const StripeScanParams&		StripeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f MinMagnitude: 5.0f MaxInnerEdgeMagnitude: Nil StripePolarity: Dark MinStripeWidth: 0.0f MaxStripeWidth: Nil)	Parameters controlling the stripe extraction process
→	inStripeCount	int	0 - ∞	1	Number of stripes to be found
→	inStripeSelection	Selection::Type		Selection::Best	Selection mode of the resulting stripes
→	inMinGapWidth	float	0.0 - ∞	0.0f	Minimal distance between consecutive stripes
→	inMaxGapWidth	Optional<float>	0.0 - ∞	NIL	Maximal distance between consecutive stripes
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
←	outStripes	Conditional<Array<Stripe1D> >&			Found stripes
←	outGaps	Conditional<Array<Gap1D> >&			Distances between consecutive stripes
←	outBrightnessProfile	Optional<Profile&>		NIL	Extracted image profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the edge (derivative) operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outBrightnessProfile**, **outResponseProfile**.

Read more about Optional Outputs.

Description

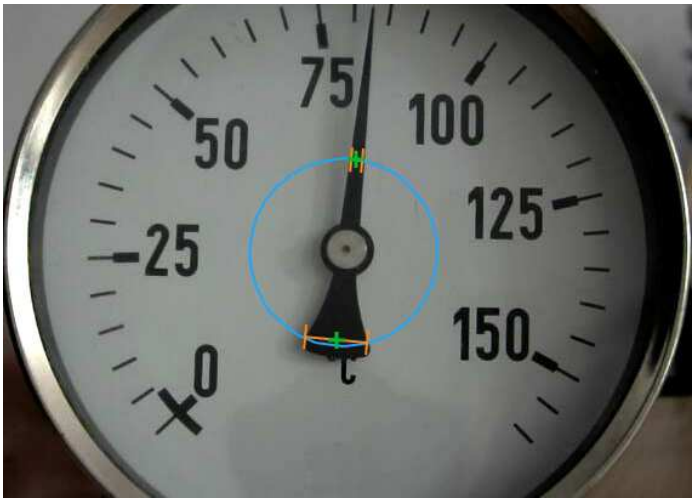
The operation scans the image using **inScanMap** previously generated from a scan path and finds a set of **inStripeCount** consecutive stripes (i.e. pairs of opposite-polarity edges running across the path). If no subset (of **inStripeCount** elements) of detected edge pairs meets the requirements of **inStripeScanParams.stripePolarity**, **inStripeScanParams.minMagnitude**, **inStripeScanParams.minStripeWidth** and **inMinGapWidth** then the outputs are set to NIL.

Note that in case of a scan path which is closed, the parameters controlling the distances between consecutive found objects do not control the distance between the first and the last of the found objects (counting from the beginning of the scan path).

Hints

- Set **inStripeCount** to the number of stripes that are to be found (the N number).
- Define **inStripeScanParams.StripePolarity** to detect a particular stripe type, and only that type.
- If the expected number of stripes cannot be found, try decreasing **inStripeScanParams.MinMagnitude**. Verify this with the values on the **outResponseProfile** output.
- If consecutive edges are closer than 6 pixels apart, change **inStripeScanParams.ProfileInterpolation** to **Quadratic3**.
- Adjust **inMinGapWidth** (in pixels) to filter out false stripes that appear in proximity to other stripes.
- Adjust **inStripeScanParams.inMinStripeWidth** and **inStripeScanParams.inMaxStripeWidth** (in pixels) to increase reliability.

Examples



ScanExactlyNStripes *locates the edge pairs using a scan map representing the scan path above (**inStripeCount** = 2, **inStripeScanParams.stripePolarity** = Dark).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the 1D Edge Detection toolset. For a comprehensive introduction to this technique please refer to 1D Edge Detection and 1D Edge Detection - Subpixel Precision chapters of our Machine Vision Guide.

See Also

- CreateScanMap – Precomputes a data object that is required for fast 1D edge detection.
- ScanSingleStripe – Locates the strongest pair of edges across a given path.
- ScanMultipleStripes – Locates multiple pairs of edges across a given path.
- ScanExactlyNStripes_Direct – Locates a specified number of multiple pairs of opposite edges across a given path (without a scan map).

ScanExactlyNStripes_Direct

Header: AVL.h
Namespace: avl
Module: MetrologyBasic

Locates a specified number of multiple pairs of opposite edges across a given path (without a scan map).

Applications: Very fast detection (or presence verification) of multiple pairs of opposite edges.

Syntax

```
void avl::ScanExactlyNStripes_Direct
(
 const avl::Image& inImage,
 const avl::Path& inScanPath,
 atl::Optional<const avl::CoordinateSystem2D&> inScanPathAlignment,
 int inScanWidth,
 const avl::SamplingParams& inSamplingParams,
 const avl::StripeScanParams& inStripeScanParams,
 int inStripeCount,
 avl::Selection::Type inStripeSelection,
 float inMinGapWidth,
 atl::Optional<float> inMaxGapWidth,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Conditional<atl::Array<avl::Stripe1D> >& outStripes,
 atl::Conditional<atl::Array<avl::Gap1D> >& outGaps,
 atl::Optional<avl::Path&> outAlignedScanPath = atl::NIL,
 atl::Optional<avl::Profile&> outBrightnessProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL,
 atl::Array<avl::Path>& diagSamplingPoints,
 float& diagSamplingStep
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inScanPath	const Path&			Path along which the scan is performed
→	inScanPathAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the scan path to the position of the inspected object
→	inScanWidth	int	1 - ∞	5	Width of the scan field in pixels
→	inSamplingParams	const SamplingParams&		SamplingParams (Interpolation: Bilinear SamplingStep: 1.0f SampleCount: Nil)	Parameters controlling the sampling process
→	inStripeScanParams	const StripeScanParams&		StripeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f MinMagnitude: 5.0f MaxInnerEdgeMagnitude: Nil StripePolarity: Dark MinStripeWidth: 0.0f MaxStripeWidth: Nil)	Parameters controlling the stripe extraction process
→	inStripeCount	int	0 - ∞	1	Number of stripes to be found
→	inStripeSelection	Selection::Type			Selection mode of the resulting stripes
→	inMinGapWidth	float	0.0 - ∞	0.0f	Minimal distance between consecutive stripes
→	inMaxGapWidth	Optional<float>	0.0 - ∞	NIL	Maximal distance between consecutive stripes
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
←	outStripes	Conditional<Array<Stripe1D> >&			Found stripes
←	outGaps	Conditional<Array<Gap1D> >&			Distances between consecutive stripes
←	outAlignedScanPath	Optional<Path&>		NIL	Transformed input path
←	outBrightnessProfile	Optional<Profile&>		NIL	Extracted image profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the edge (derivative) operator response
🔍	diagSamplingPoints	Array<Path>&			Array of paths each one containing the sampling points that contributes to a single value of the extracted profile
🔍	diagSamplingStep	float&			Used distance between consecutive sampling points on the scan path

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedScanPath**, **outBrightnessProfile**, **outResponseProfile**.

Read more about Optional Outputs.

Description

The operation scans the image along **inScanPath** and finds a set of **inStripeCount** consecutive stripes (i.e. pairs of opposite-polarity edges running across the path). If no subset (of **inStripeCount** elements) of detected edge pairs meets the requirements of **inStripeScanParams.stripePolarity**, **inStripeScanParams.minMagnitude**, **inStripeScanParams.minStripeWidth** and **inMinGapWidth** then the outputs are set to NIL.

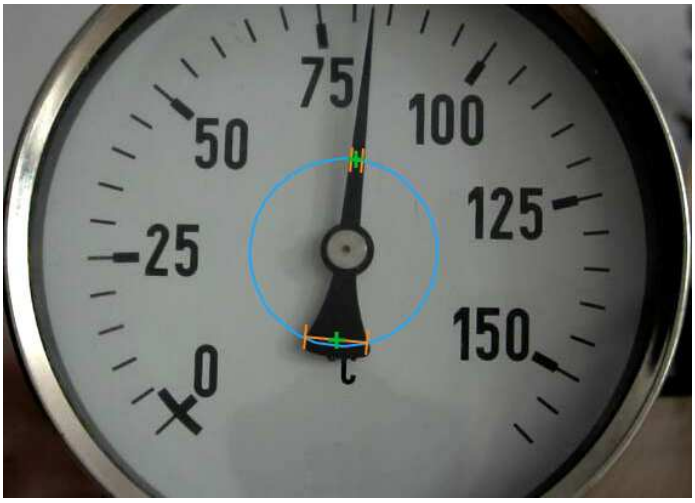
The optional parameter **inScanPathAlignment** defines a transform to be performed on the **inScanPath** so that the actual scan path (**outAlignedScanPath**) is adjusted to the position of the object, typically detected by one of Template Matching filters.

Note that in case of a scan path which is closed, the parameters controlling the distances between consecutive found objects do not control the distance between the first and the last of the found objects (counting from the beginning of the scan path).

Hints

- Set **inStripeCount** to the number of stripes that are to be found (the N number).
- Define **inStripeScanParams.StripePolarity** to detect a particular stripe type, and only that type.
- If the expected number of stripes cannot be found, try decreasing **inStripeScanParams.MinMagnitude**. Verify this with the values on the **outResponseProfile** output.
- If consecutive edges are closer than 6 pixels apart, change **inStripeScanParams.ProfileInterpolation** to **Quadratic3**.
- Adjust **inMinGapWidth** (in pixels) to filter out false stripes that appear in proximity to other stripes.
- Adjust **inStripeScanParams.inMinStripeWidth** and **inStripeScanParams.inMaxStripeWidth** (in pixels) to increase reliability.

Examples



ScanExactlyNStripes_Direct *locates the edge pairs across* ***inScanPath*** *(****inStripeCount*** *= 2,* ***inStripeScanParams.stripePolarity*** *= Dark).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the 1D Edge Detection toolset. For a comprehensive introduction to this technique please refer to 1D Edge Detection and 1D Edge Detection - Subpixel Precision chapters of our Machine Vision Guide.

See Also

- CreateScanMap – Precomputes a data object that is required for fast 1D edge detection.
- ScanSingleStripe_Direct – Locates the strongest pair of edges across a given path (without a scan map).
- ScanMultipleStripes_Direct – Locates multiple pairs of edges across a given path (without a scan map).
- ScanExactlyNStripes – Locates a specified number of multiple pairs of opposite edges across a given path.

ScanMultipleEdges

Header: AVL.h
Namespace: avl
Module: MetrologyBasic

Locates multiple transitions between dark and bright pixels along a given path.

Applications: Very fast detection of multiple edge points - usually for object counting or displacement detection.

Syntax

```
void avl::ScanMultipleEdges
(
 const avl::Image& inImage,
 const ScanMap& inScanMap,
 const EdgeScanParams& inEdgeScanParams,
 float inMinDistance,
 atl::Optional<float> inMaxDistance,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Array<avl::Edge1D>& outEdges,
 atl::Array<avl::Gap1D>& outGaps,
 atl::Optional<avl::Profile&> outBrightnessProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inScanMap	const ScanMap&			Data precomputed with CreateScanMap
→	inEdgeScanParams	const EdgeScanParams&		EdgeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f MinMagnitude: 5.0f EdgeTransition: BrightToDark)	Parameters controlling the edge extraction process
→	inMinDistance	float	0.0 - ∞	0.0f	Minimal distance between consecutive edges
→	inMaxDistance	Optional<float>	0.0 - ∞	NIL	Maximal distance between consecutive edges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
←	outEdges	Array<Edge1D>&			Found edges
←	outGaps	Array<Gap1D>&			Gaps between consecutive edges
←	outBrightnessProfile	Optional<Profile&>		NIL	Extracted image profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the edge (derivative) operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outBrightnessProfile**, **outResponseProfile**.

Read more about Optional Outputs.

Description

The operation scans the image using **inScanMap** previously generated from a scan path and detects image edges perpendicular to the path. Depending on the **inEdgeScanParams.edgeTransition** parameter, edges representing increase or decrease (or both) of image brightness along the path will be taken into account.

When the number of edges to be found is known, one can use the ScanExactlyNEdges.

Note that in case of a scan path which is closed, the parameters controlling the distances between consecutive found objects do not control the distance between the first and the last of the found objects (counting from the beginning of the scan path).

Hints

- Define **inEdgeScanParams.EdgeTransition** to detect a particular edge type, and only that type.
- If too few edges are found, try decreasing **inEdgeScanParams.MinMagnitude**. Verify this with the values on the **outResponseProfile** output.
- If consecutive edges are closer than 6 pixels apart, change **inEdgeScanParams.ProfileInterpolation** to **Quadratic3**.
- Adjust **inMinDistance** (in pixels) to filter out false edges that appear in proximity to other edges.

Examples

ScanExactlyNEdges *locates the edges using a scan map representing the scan path above.*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the 1D Edge Detection toolset. For a comprehensive introduction to this technique please refer to 1D Edge Detection and 1D Edge Detection - Subpixel Precision chapters of our Machine Vision Guide.

See Also

- CreateScanMap – Precomputes a data object that is required for fast 1D edge detection.
- ScanSingleEdge – Locates the strongest transition between dark and bright pixels along a given path.
- ScanExactlyNEdges – Locates a specified number of the strongest transitions between dark and bright pixels along a given path.
- ScanExactlyNEdges_Direct – Locates a specified number of the strongest transitions between dark and bright pixels along a given path (without a scan map).

ScanMultipleEdges_Direct

Header: AVL.h
Namespace: avl
Module: MetrologyBasic

Locates multiple transitions between dark and bright pixels along a given path (without a scan map).

Applications: Very fast detection of multiple edge points - usually for object counting or displacement detection.

Syntax

```
void avl::ScanMultipleEdges_Direct
(
 const avl::Image& inImage,
 const avl::Path& inScanPath,
 atl::Optional<const avl::CoordinateSystem2D&> inScanPathAlignment,
 int inScanWidth,
 const avl::SamplingParams& inSamplingParams,
 const avl::EdgeScanParams& inEdgeScanParams,
 float inMinDistance,
 atl::Optional<float> inMaxDistance,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Array<avl::Edge1D>& outEdges,
 atl::Array<avl::Gap1D>& outGaps,
 atl::Optional<avl::Path&> outAlignedScanPath = atl::NIL,
 atl::Optional<avl::Profile&> outBrightnessProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL,
 atl::Array<avl::Path>& diagSamplingPoints,
 float& diagSamplingStep
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inScanPath	const Path&			Path along which the scan is performed
→	inScanPathAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the scan path to the position of the inspected object
→	inScanWidth	int	1 - ∞	5	Width of the scan field in pixels
→	inSamplingParams	const SamplingParams&		SamplingParams (Interpolation: Bilinear SamplingStep: 1.0f SampleCount: Nil)	Parameters controlling the sampling process
→	inEdgeScanParams	const EdgeScanParams&		EdgeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f MinMagnitude: 5.0f EdgeTransition: BrightToDark)	Parameters controlling the edge extraction process
→	inMinDistance	float	0.0 - ∞	0.0f	Minimal distance between consecutive edges
→	inMaxDistance	Optional<float>	0.0 - ∞	NIL	Maximal distance between consecutive edges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
←	outEdges	Array<Edge1D>&			Found edges
←	outGaps	Array<Gap1D>&			Gaps between consecutive edges
←	outAlignedScanPath	Optional<Path&>		NIL	Transformed input path
←	outBrightnessProfile	Optional<Profile&>		NIL	Extracted image profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the edge (derivative) operator response
🔍	diagSamplingPoints	Array<Path>&			Array of paths each one containing the sampling points that contributes to a single value of the extracted profile
🔍	diagSamplingStep	float&			Used distance between consecutive sampling points on the scan path

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedScanPath**, **outBrightnessProfile**, **outResponseProfile**.

Read more about Optional Outputs.

Description

The operation scans the image along **inScanPath** and detects image edges perpendicular to the path. Depending on the **inEdgeScanParams.edgeTransition** parameter, edges representing increase or decrease (or both) of image brightness along the path will be taken into account.

When the number of edges to be found is known, one can use the ScanExactlyNEdges_Direct.

The optional parameter **inScanPathAlignment** defines a transform to be performed on the **inScanPath** so that the actual scan path (**outAlignedScanPath**) is adjusted to the position of the object, typically detected by one of Template Matching filters.

Note that in case of a scan path which is closed, the parameters controlling the distances between consecutive found objects do not control the distance between the first and the last of the found objects (counting from the beginning of the scan path).

Hints

- Define **inEdgeScanParams.EdgeTransition** to detect a particular edge type, and only that type.
- If too few edges are found, try decreasing **inEdgeScanParams.MinMagnitude**. Verify this with the values on the **outResponseProfile** output.
- If consecutive edges are closer than 6 pixels apart, change **inEdgeScanParams.ProfileInterpolation** to **Quadratic3**.
- Adjust **inMinDistance** (in pixels) to filter out false edges that appear in proximity to other edges.

Examples

***ScanExactlyNEdges_Direct** locates the edges across **inScanPath**.*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the 1D Edge Detection toolset. For a comprehensive introduction to this technique please refer to 1D Edge Detection and 1D Edge Detection - Subpixel Precision chapters of our Machine Vision Guide.

See Also

- CreateScanMap – Precomputes a data object that is required for fast 1D edge detection.
- ScanSingleEdge_Direct – Locates the strongest transition between dark and bright pixels along a given path (without a scan map).
- ScanExactlyNEdges_Direct – Locates a specified number of the strongest transitions between dark and bright pixels along a given path (without a scan map).
- ScanExactlyNEdges – Locates a specified number of the strongest transitions between dark and bright pixels along a given path.

ScanMultipleRidges

Header: AVL.h
Namespace: avl
Module: MetrologyBasic

Locates multiple dark or bright pixel peaks along a given path.

Applications: Very fast detection of multiple thin structures like wires or scale marks - usually for counting or distance measurements.

Syntax

```
void avl::ScanMultipleRidges
(
 const avl::Image& inImage,
 const ScanMap& inScanMap,
 const RidgeScanParams& inRidgeScanParams,
 float inMinDistance,
 atl::Optional<float> inMaxDistance,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Array<avl::Ridge1D>& outRidges,
 atl::Array<avl::Gap1D>& outGaps,
 atl::Optional<avl::Profile&> outBrightnessProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inScanMap	const ScanMap&			Data precomputed with CreateScanMap
→	inRidgeScanParams	const RidgeScanParams&		RidgeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f RidgeWidth: 5 RidgeMargin: 2 MinMagnitude: 5.0f RidgePolarity: Dark)	Parameters controlling the ridge extraction process
→	inMinDistance	float	0.0 - ∞	0.0f	Minimal distance between consecutive ridges
→	inMaxDistance	Optional<float>	0.0 - ∞	NIL	Maximal distance between consecutive ridges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker ridges can be detected in the vicinity of stronger ridges
←	outRidges	Array<Ridge1D>&			Found ridges
←	outGaps	Array<Gap1D>&			Gaps between consecutive ridges
←	outBrightnessProfile	Optional<Profile&>		NIL	Extracted image profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the ridge operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outBrightnessProfile**, **outResponseProfile**.

Read more about Optional Outputs.

Description

The operation scans the image using **inScanMap** previously generated from a scan path and detects ridges. Depending on the **inRidgeScanParams.ridgePolarity** parameter, dark, bright or both ridges will be taken into account.

When the number of ridges to be measured is known, one can use the ScanExactlyNRidges filter.

Note that in case of a scan path which is closed, the parameters controlling the distances between consecutive found objects do not control the distance between the first and the last of the found objects (counting from the beginning of the scan path).

Hints

- Define **inRidgeScanParams.RidgePolarity** to detect a particular ridge type, and only that type.
- If too few ridges are found, try decreasing **inRidgeScanParams.MinMagnitude**. Verify this with the values on the **outResponseProfile** output.
- If consecutive ridges are very close to each other, change **inRidgeScanParams.ProfileInterpolation** to **Quadratic3**.
- Adjust **inMinDistance** (in pixels) to filter out false ridges that appear in proximity to other ridges.

Examples

ScanMultipleRidges *locates the ridges using a scan map representing the scan path above.*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the 1D Edge Detection toolset. For a comprehensive introduction to this technique please refer to 1D Edge Detection and 1D Edge Detection - Subpixel Precision chapters of our Machine Vision Guide.

See Also

- CreateScanMap – Precomputes a data object that is required for fast 1D edge detection.
- ScanSingleRidge – Locates the strongest dark or bright pixel peak along a given path.
- ScanExactlyNRidges – Locates a specified number of the strongest dark or bright pixel peak along a given path.
- ScanMultipleRidges_Direct – Locates multiple dark or bright pixel peaks along a given path (without a scan map).

ScanMultipleRidges_Direct

Header: AVL.h
Namespace: avl
Module: MetrologyBasic

Locates multiple dark or bright pixel peaks along a given path (without a scan map).

Applications: Very fast detection of multiple thin structures like wires or scale marks - usually for counting or distance measurements.

Syntax

```
void avl::ScanMultipleRidges_Direct
(
 const avl::Image& inImage,
 const avl::Path& inScanPath,
 atl::Optional<const avl::CoordinateSystem2D&> inScanPathAlignment,
 int inScanWidth,
 const avl::SamplingParams& inSamplingParams,
 const avl::RidgeScanParams& inRidgeScanParams,
 float inMinDistance,
 atl::Optional<float> inMaxDistance,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Array<avl::Ridge1D>& outRidges,
 atl::Array<avl::Gap1D>& outGaps,
 atl::Optional<avl::Path&> outAlignedScanPath = atl::NIL,
 atl::Optional<avl::Profile&> outBrightnessProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL,
 atl::Array<avl::Path>& diagSamplingPoints,
 float& diagSamplingStep
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inScanPath	const Path&			Path along which the scan is performed
→	inScanPathAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the scan path to the position of the inspected object
→	inScanWidth	int	1 - ∞	5	Width of the scan field in pixels
→	inSamplingParams	const SamplingParams&		SamplingParams (Interpolation: Bilinear SamplingStep: 1.0f SampleCount: Nil)	Parameters controlling the sampling process
→	inRidgeScanParams	const RidgeScanParams&		RidgeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f RidgeWidth: 5 RidgeMargin: 2 MinMagnitude: 5.0f RidgePolarity: Dark)	Parameters controlling the ridge extraction process
→	inMinDistance	float	0.0 - ∞	0.0f	Minimal distance between consecutive ridges
→	inMaxDistance	Optional<float>	0.0 - ∞	NIL	Maximal distance between consecutive ridges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker ridges can be detected in the vicinity of stronger ridges
←	outRidges	Array<Ridge1D>&			Found ridges
←	outGaps	Array<Gap1D>&			Gaps between consecutive ridges
←	outAlignedScanPath	Optional<Path&>		NIL	Transformed input path
←	outBrightnessProfile	Optional<Profile&>		NIL	Extracted image profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the ridge operator response
🔍	diagSamplingPoints	Array<Path>&			Array of paths each one containing the sampling points that contributes to a single value of the extracted profile
🔍	diagSamplingStep	float&			Used distance between consecutive sampling points on the scan path

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedScanPath**, **outBrightnessProfile**, **outResponseProfile**.

Read more about Optional Outputs.

Description

The operation scans the image along **inScanPath** and detects ridges. Depending on the **inRidgeScanParams.ridgePolarity** parameter, dark, bright or both ridges will be taken into account.

When the number of ridges to be measured is known, one can use the ScanExactlyNRidges_Direct filter.

The optional parameter **inScanPathAlignment** defines a transform to be performed on the **inScanPath** so that the actual scan path (**outAlignedScanPath**) is adjusted to the position of the object, typically detected by one of Template Matching filters.

Note that in case of a scan path which is closed, the parameters controlling the distances between consecutive found objects do not control the distance between the first and the last of the found objects (counting from the beginning of the scan path).

Hints

- Define **inRidgeScanParams.RidgePolarity** to detect a particular ridge type, and only that type.
- If too few ridges are found, try decreasing **inRidgeScanParams.MinMagnitude**. Verify this with the values on the **outResponseProfile** output.
- If consecutive ridges are very close to each other, change **inRidgeScanParams.ProfileInterpolation** to **Quadratic3**.
- Adjust **inMinDistance** (in pixels) to filter out false ridges that appear in proximity to other ridges.

Examples

ScanMultipleRidges_Direct *locates the ridges across* ***inScanPath****.*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the 1D Edge Detection toolset. For a comprehensive introduction to this technique please refer to 1D Edge Detection and 1D Edge Detection - Subpixel Precision chapters of our Machine Vision Guide.

See Also

- CreateScanMap – Precomputes a data object that is required for fast 1D edge detection.
- ScanSingleRidge_Direct – Locates the strongest dark or bright pixel peak along a given path (without a scan map).
- ScanExactlyNRidges_Direct – Locates a specified number of the strongest dark or bright pixel peak along a given path (without a scan map).
- ScanMultipleRidges – Locates multiple dark or bright pixel peaks along a given path.

ScanMultipleStripes

Header: AVL.h
Namespace: avl
Module: MetrologyBasic

Locates multiple pairs of edges across a given path.

Applications: Very fast detection of multiple pairs of opposite edges - usually for counting or width measurements.

Syntax

```
void avl::ScanMultipleStripes
(
 const avl::Image& inImage,
 const ScanMap& inScanMap,
 const StripeScanParams& inStripeScanParams,
 float inMinGapWidth,
 atl::Optional<float> inMaxGapWidth,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Array<avl::Stripe1D>& outStripes,
 atl::Array<avl::Gap1D>& outGaps,
 atl::Optional<avl::Profile&> outBrightnessProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inScanMap	const ScanMap&			Data precomputed with CreateScanMap
→	inStripeScanParams	const StripeScanParams&		StripeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f MinMagnitude: 5.0f MaxInnerEdgeMagnitude: Nil StripePolarity: Dark MinStripeWidth: 0.0f MaxStripeWidth: Nil)	Parameters controlling the stripe extraction process
→	inMinGapWidth	float	0.0 - ∞	0.0f	Minimal distance between consecutive stripes
→	inMaxGapWidth	Optional<float>	0.0 - ∞	NIL	Maximal distance between consecutive stripes
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
←	outStripes	Array<Stripe1D>&			Found stripes
←	outGaps	Array<Gap1D>&			Distances between consecutive stripes
←	outBrightnessProfile	Optional<Profile&>		NIL	Extracted image profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the edge (derivative) operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outBrightnessProfile**, **outResponseProfile**.

Read more about Optional Outputs.

Description

The operation scans the image using **inScanMap** previously generated from a scan path and detects consecutive stripes (i.e. pairs of opposite-polarity edges running across the path). Depending on the **inStripeScanParams.stripePolarity** parameter, dark or bright stripes will be taken into account.

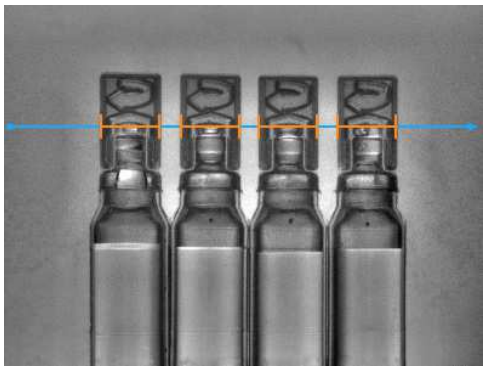
When the number of edge pairs to be measured is known, one can use the ScanExactlyNStripes filter.

Note that in case of a scan path which is closed, the parameters controlling the distances between consecutive found objects do not control the distance between the first and the last of the found objects (counting from the beginning of the scan path).

Hints

- Define **inStripeScanParams.StripePolarity** to detect a particular edge type, and only that type.
- If too few stripes are found, try decreasing **inStripeScanParams.MinMagnitude**. Verify this with the values on the **outResponseProfile** output.
- If consecutive edges are closer than 6 pixels apart, change **inStripeScanParams.ProfileInterpolation** to **Quadratic3**.
- Adjust **inMinGapWidth** (in pixels) to filter out false stripes that appear in proximity to other edges.
- Adjust **inStripeScanParams.inMinStripeWidth** and **inStripeScanParams.inMaxStripeWidth** (in pixels) to increase reliability.

Examples



ScanMultipleStripes *locates the edge pairs using a scan map representing the scan path above (**inStripeScanParams.stripePolarity** = Dark, **inStripeScanParams.minStripeWidth** = 45).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the 1D Edge Detection toolset. For a comprehensive introduction to this technique please refer to 1D Edge Detection and 1D Edge Detection - Subpixel Precision chapters of our Machine Vision Guide.

See Also

- CreateScanMap – Precomputes a data object that is required for fast 1D edge detection.
- ScanSingleStripe – Locates the strongest pair of edges across a given path.
- ScanExactlyNStripes – Locates a specified number of multiple pairs of opposite edges across a given path.
- ScanMultipleStripes_Direct – Locates multiple pairs of edges across a given path (without a scan map).

ScanMultipleStripes_Direct

Header: AVL.h
Namespace: avl
Module: MetrologyBasic

Locates multiple pairs of edges across a given path (without a scan map).

Applications: Very fast detection of multiple pairs of opposite edges - usually for counting or width measurements.

Syntax

```
void avl::ScanMultipleStripes_Direct
(
 const avl::Image& inImage,
 const avl::Path& inScanPath,
 atl::Optional<const avl::CoordinateSystem2D&> inScanPathAlignment,
 int inScanWidth,
 const avl::SamplingParams& inSamplingParams,
 const avl::StripeScanParams& inStripeScanParams,
 float inMinGapWidth,
 atl::Optional<float> inMaxGapWidth,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Array<avl::Stripe1D>& outStripes,
 atl::Array<avl::Gap1D>& outGaps,
 atl::Optional<avl::Path&> outAlignedScanPath = atl::NIL,
 atl::Optional<avl::Profile&> outBrightnessProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL,
 atl::Array<avl::Path>& diagSamplingPoints,
 float& diagSamplingStep
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inScanPath	const Path&			Path along which the scan is performed
→	inScanPathAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the scan path to the position of the inspected object
→	inScanWidth	int	1 - ∞	5	Width of the scan field in pixels
→	inSamplingParams	const SamplingParams&		SamplingParams (Interpolation: Bilinear SamplingStep: 1.0f SampleCount: Nil)	Parameters controlling the sampling process
→	inStripeScanParams	const StripeScanParams&		StripeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f MinMagnitude: 5.0f MaxInnerEdgeMagnitude: Nil StripePolarity: Dark MinStripeWidth: 0.0f MaxStripeWidth: Nil)	Parameters controlling the stripe extraction process
→	inMinGapWidth	float	0.0 - ∞	0.0f	Minimal distance between consecutive stripes
→	inMaxGapWidth	Optional<float>	0.0 - ∞	NIL	Maximal distance between consecutive stripes
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
←	outStripes	Array<Stripe1D>&			Found stripes
←	outGaps	Array<Gap1D>&			Distances between consecutive stripes
←	outAlignedScanPath	Optional<Path&>		NIL	Transformed input path
←	outBrightnessProfile	Optional<Profile&>		NIL	Extracted image profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the edge (derivative) operator response
🔍	diagSamplingPoints	Array<Path>&			Array of paths each one containing the sampling points that contributes to a single value of the extracted profile
🔍	diagSamplingStep	float&			Used distance between consecutive sampling points on the scan path

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedScanPath**, **outBrightnessProfile**, **outResponseProfile**.

Read more about Optional Outputs.

Description

The operation scans the image along **inScanPath** and detects consecutive stripes (i.e. pairs of opposite-polarity edges running across the path). Depending on the **inStripeScanParams.stripePolarity** parameter, dark or bright stripes will be taken into account.

When the number of edge pairs to be measured is known, one can use the ScanExactlyNStripes_Direct filter.

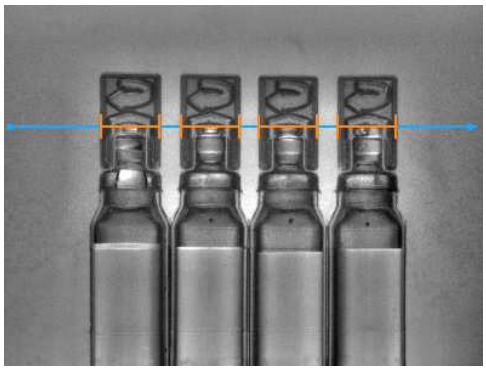
The optional parameter **inScanPathAlignment** defines a transform to be performed on the **inScanPath** so that the actual scan path (**outAlignedScanPath**) is adjusted to the position of the object, typically detected by one of Template Matching filters.

Note that in case of a scan path which is closed, the parameters controlling the distances between consecutive found objects do not control the distance between the first and the last of the found objects (counting from the beginning of the scan path).

Hints

- Define **inStripeScanParams.StripePolarity** to detect a particular edge type, and only that type.
- If too few stripes are found, try decreasing **inStripeScanParams.MinMagnitude**. Verify this with the values on the **outResponseProfile** output.
- If consecutive edges are closer than 6 pixels apart, change **inStripeScanParams.ProfileInterpolation** to **Quadratic3**.
- Adjust **inMinGapWidth** (in pixels) to filter out false stripes that appear in proximity to other edges.
- Adjust **inStripeScanParams.inMinStripeWidth** and **inStripeScanParams.inMaxStripeWidth** (in pixels) to increase reliability.

Examples



ScanMultipleStripes_Direct *locates the edge pairs across* ***inScanPath*** *(****inStripeScanParams.stripePolarity*** *= Dark,* ***inStripeScanParams.minStripeWidth*** *= 45).*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the 1D Edge Detection toolset. For a comprehensive introduction to this technique please refer to 1D Edge Detection and 1D Edge Detection - Subpixel Precision chapters of our Machine Vision Guide.

See Also

- CreateScanMap – Precomputes a data object that is required for fast 1D edge detection.
- ScanSingleStripe_Direct – Locates the strongest pair of edges across a given path (without a scan map).
- ScanExactlyNStripes_Direct – Locates a specified number of multiple pairs of opposite edges across a given path (without a scan map).
- ScanMultipleStripes – Locates multiple pairs of edges across a given path.

ScanSingleEdge

Header: AVL.h
Namespace: avl
Module: MetrologyBasic

Locates the strongest transition between dark and bright pixels along a given path.

Applications: Very fast detection of an object (e.g. horizontal displacement of a bottle) and simple measurements (e.g. liquid level in a bottle).

Syntax

```
void avl::ScanSingleEdge
(
 const avl::Image& inImage,
 const avl::ScanMap& inScanMap,
 const avl::EdgeScanParams& inEdgeScanParams,
 avl::Selection::Type inEdgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Conditional<avl::Edge1D>& outEdge,
 atl::Optional<avl::Profile&> outBrightnessProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
➡	inImage	const Image&		Input image
➡	inScanMap	const ScanMap&		Data precomputed with CreateScanMap
➡	inEdgeScanParams	const EdgeScanParams&	EdgeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f MinMagnitude: 5.0f EdgeTransition: BrightToDark)	Parameters controlling the edge extraction process
➡	inEdgeSelection	Selection::Type	Selection::Best	Selection mode of the resulting edge
➡	inLocalBlindness	Optional<const LocalBlindness&>	NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
⬅	outEdge	Conditional<Edge1D>&		Found edge
⬅	outBrightnessProfile	Optional<Profile&>	NIL	Extracted image profile
⬅	outResponseProfile	Optional<Profile&>	NIL	Profile of the edge (derivative) operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outBrightnessProfile**, **outResponseProfile**.

Read more about Optional Outputs.

Description

The operation scans the image using **inScanMap** previously generated from a scan path and locates the strongest edge perpendicular to the path. If the strongest edge is weaker than **inEdgeScanParams.minMagnitude** then the outputs are set to NIL.

Hints

- Define **inEdgeScanParams.EdgeTransition** to detect a particular edge type, and only that type.
- If no edge is found, try decreasing **inEdgeScanParams.MinMagnitude**. Verify this with the values on the **outResponseProfile** output.
- If consecutive edges are closer than 6 pixels apart, change **inEdgeScanParams.ProfileInterpolation** to **Quadratic3**.

Examples

ScanSingleEdge *locates the strongest edge using a scan map representing the scan path above.*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the 1D Edge Detection toolset. For a comprehensive introduction to this technique please refer to 1D Edge Detection and 1D Edge Detection - Subpixel Precision chapters of our Machine Vision Guide.

See Also

- CreateScanMap – Precomputes a data object that is required for fast 1D edge detection.
- ScanMultipleEdges – Locates multiple transitions between dark and bright pixels along a given path.
- ScanExactlyNEdges – Locates a specified number of the strongest transitions between dark and bright pixels along a given path.
- ScanSingleEdge_Direct – Locates the strongest transition between dark and bright pixels along a given path (without a scan map).

ScanSingleEdge_Direct

Header: AVL.h
Namespace: avl
Module: MetrologyBasic

Locates the strongest transition between dark and bright pixels along a given path (without a scan map).

Applications: Very fast detection of an object (e.g. horizontal displacement of a bottle) and simple measurements (e.g. liquid level in a bottle).

Syntax

```
void avl::ScanSingleEdge_Direct
(
 const avl::Image& inImage,
 const avl::Path& inScanPath,
 atl::Optional<const avl::CoordinateSystem2D&> inScanPathAlignment,
 int inScanWidth,
 const avl::SamplingParams& inSamplingParams,
 const avl::EdgeScanParams& inEdgeScanParams,
 avl::Selection::Type inEdgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Conditional<avl::Edge1D>& outEdge,
 atl::Optional<avl::Path&> outAlignedScanPath = atl::NIL,
 atl::Optional<avl::Profile&> outBrightnessProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL,
 atl::Array<avl::Path>& diagSamplingPoints,
 float& diagSamplingStep
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inScanPath	const Path&			Path along which the scan is performed
→	inScanPathAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the scan path to the position of the inspected object
→	inScanWidth	int	1 - ∞	5	Width of the scan field in pixels
→	inSamplingParams	const SamplingParams&		SamplingParams (Interpolation: Bilinear SamplingStep: 1.0f SampleCount: Nil)	Parameters controlling the sampling process
→	inEdgeScanParams	const EdgeScanParams&		EdgeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f MnMagnitude: 5.0f EdgeTransition: BrightToDark)	Parameters controlling the edge extraction process
→	inEdgeSelection	Selection::Type			Selection mode of the resulting edge
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
←	outEdge	Conditional<Edge1D>&			Found edge
←	outAlignedScanPath	Optional<Path&>		NIL	Transformed input path
←	outBrightnessProfile	Optional<Profile&>		NIL	Extracted image profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the edge (derivative) operator response
🔍	diagSamplingPoints	Array<Path>&			Array of paths each one containing the sampling points that contributes to a single value of the extracted profile
🔍	diagSamplingStep	float&			Used distance between consecutive sampling points on the scan path

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedScanPath**, **outBrightnessProfile**, **outResponseProfile**.

Read more about Optional Outputs.

Description

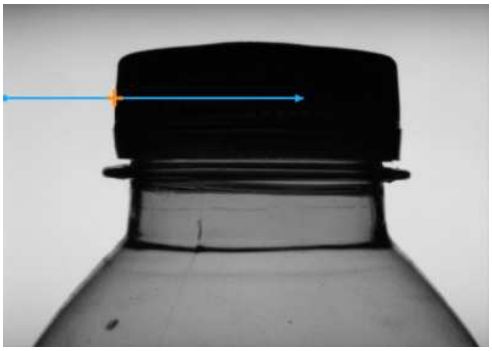
The operation scans the image along **inScanPath** and locates the strongest edge perpendicular to the path. If the strongest edge is weaker than **inEdgeScanParams.minMagnitude** then the outputs are set to NIL.

The optional parameter **inScanPathAlignment** defines a transform to be performed on the **inScanPath** so that the actual scan path (**outAlignedScanPath**) is adjusted to the position of the object, typically detected by one of Template Matching filters.

Hints

- Define **inEdgeScanParams.EdgeTransition** to detect a particular edge type, and only that type.
- If no edge is found, try decreasing **inEdgeScanParams.MinMagnitude**. Verify this with the values on the **outResponseProfile** output.
- If consecutive edges are closer than 6 pixels apart, change **inEdgeScanParams.ProfileInterpolation** to **Quadratic3**.

Examples



ScanSingleEdge_Direct *locates the strongest edge across* ***inScanPath****.*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the 1D Edge Detection toolset. For a comprehensive introduction to this technique please refer to 1D Edge Detection and 1D Edge Detection - Subpixel Precision chapters of our Machine Vision Guide.

See Also

- CreateScanMap – Precomputes a data object that is required for fast 1D edge detection.
- ScanMultipleEdges_Direct – Locates multiple transitions between dark and bright pixels along a given path (without a scan map).
- ScanExactlyNEdges_Direct – Locates a specified number of the strongest transitions between dark and bright pixels along a given path (without a scan map).
- ScanSingleEdge – Locates the strongest transition between dark and bright pixels along a given path.

ScanSingleRidge

Header: AVL.h
Namespace: avl
Module: MetrologyBasic

Locates the strongest dark or bright pixel peak along a given path.

Applications: Very fast detection of a thin structure like a wire or a scale mark.

Syntax

```
void avl::ScanSingleRidge
(
 const avl::Image& inImage,
 const ScanMap& inScanMap,
 const RidgeScanParams& inRidgeScanParams,
 avl::Selection::Type inRidgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Conditional<avl::Ridge1D>& outRidge,
 atl::Optional<avl::Profile&> outBrightnessProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inScanMap	const ScanMap&		Data precomputed with CreateScanMap
→	inRidgeScanParams	const RidgeScanParams&	RidgeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f RidgeWidth: 5 RidgeMargin: 2 MinMagnitude: 5.0f RidgePolarity: Dark)	Parameters controlling the ridge extraction process
→	inRidgeSelection	Selection::Type	Selection::Best	Selection mode of the resulting ridge
→	inLocalBlindness	Optional<const LocalBlindness&>	NIL	Defines conditions in which weaker ridges can be detected in the vicinity of stronger ridges
←	outRidge	Conditional<Ridge1D>&		Found ridge
←	outBrightnessProfile	Optional<Profile&>	NIL	Extracted image profile
←	outResponseProfile	Optional<Profile&>	NIL	Profile of the ridge operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outBrightnessProfile**, **outResponseProfile**.

Read more about Optional Outputs.

Description

The operation scans the image using **inScanMap** previously generated from a scan path and locates the strongest ridge of the given characteristics. If there is no such ridge then the outputs are set to NIL.

Hints

- Adjust **inRidgeScanParams.RidgeWidth** to the expected thickness of the ridge (in pixels).
- Define **inRidgeScanParams.RidgePolarity** to detect a particular ridge type, and only that type.
- If no ridge is found, try decreasing **inRidgeScanParams.MinMagnitude**.
- For difficult cases try different settings of the **inRidgeScanParams.RidgeOperator** parameter.

Examples

ScanSingleRidge *locates the strongest ridge using a scan map representing the scan path above.*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the 1D Edge Detection toolset. For a comprehensive introduction to this technique please refer to 1D Edge Detection and 1D Edge Detection - Subpixel Precision chapters of our Machine Vision Guide.

See Also

- CreateScanMap – Precomputes a data object that is required for fast 1D edge detection.
- ScanMultipleRidges – Locates multiple dark or bright pixel peaks along a given path.
- ScanExactlyNRidges – Locates a specified number of the strongest dark or bright pixel peak along a given path.
- ScanSingleRidge_Direct – Locates the strongest dark or bright pixel peak along a given path (without a scan map).

ScanSingleRidge_Direct

Header: AVL.h
Namespace: avl
Module: MetrologyBasic

Locates the strongest dark or bright pixel peak along a given path (without a scan map).

Applications: Very fast detection of a thin structure like a wire or a scale mark.

Syntax

```
void avl::ScanSingleRidge_Direct
(
 const avl::Image& inImage,
 const avl::Path& inScanPath,
 atl::Optional<const avl::CoordinateSystem2D&> inScanPathAlignment,
 int inScanWidth,
 const avl::SamplingParams& inSamplingParams,
 const avl::RidgeScanParams& inRidgeScanParams,
 avl::Selection::Type inRidgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Conditional<avl::Ridge1D>& outRidge,
 atl::Optional<avl::Path&> outAlignedScanPath = atl::NIL,
 atl::Optional<avl::Profile&> outBrightnessProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL,
 atl::Array<avl::Path>& diagSamplingPoints,
 float& diagSamplingStep
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inScanPath	const Path&			Path along which the scan is performed
→	inScanPathAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the scan path to the position of the inspected object
→	inScanWidth	int	1 - ∞	5	Width of the scan field in pixels
→	inSamplingParams	const SamplingParams&		SamplingParams (Interpolation: Bilinear SamplingStep: 1.0f SampleCount: Nil)	Parameters controlling the sampling process
→	inRidgeScanParams	const RidgeScanParams&		RidgeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f RidgeWidth: 5 RidgeMargin: 2 MinMagnitude: 5.0f RidgePolarity: Dark)	Parameters controlling the ridge extraction process
→	inRidgeSelection	Selection::Type			Selection mode of the resulting ridge
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker ridges can be detected in the vicinity of stronger ridges
←	outRidge	Conditional<Ridge1D>&			Found ridge
←	outAlignedScanPath	Optional<Path&>		NIL	Transformed input path
←	outBrightnessProfile	Optional<Profile&>		NIL	Extracted image profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the ridge operator response
🔍	diagSamplingPoints	Array<Path>&			Array of paths each one containing the sampling points that contributes to a single value of the extracted profile
🔍	diagSamplingStep	float&			Used distance between consecutive sampling points on the scan path

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedScanPath**, **outBrightnessProfile**, **outResponseProfile**.

Read more about Optional Outputs.

Description

The operation scans the image along **inScanPath** and locates the strongest ridge of the given characteristics. If there is no such ridge then the outputs are set to NIL.

The optional parameter **inScanPathAlignment** defines a transform to be performed on the **inScanPath** so that the actual scan path (**outAlignedScanPath**) is adjusted to the position of the object, typically detected by one of Template Matching filters.

Hints

- Adjust **inRidgeScanParams.RidgeWidth** to the expected thickness of the ridge (in pixels).
- Define **inRidgeScanParams.RidgePolarity** to detect a particular ridge type, and only that type.
- If no ridge is found, try decreasing **inRidgeScanParams.MinMagnitude**.
- For difficult cases try different settings of the **inRidgeScanParams.RidgeOperator** parameter.

Examples

ScanSingleRidge_Direct *locates the strongest ridge across* ***inScanPath****.*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the 1D Edge Detection toolset. For a comprehensive introduction to this technique please refer to 1D Edge Detection and 1D Edge Detection - Subpixel Precision chapters of our Machine Vision Guide.

See Also

- CreateScanMap – Precomputes a data object that is required for fast 1D edge detection.
- ScanMultipleRidges_Direct – Locates multiple dark or bright pixel peaks along a given path (without a scan map).
- ScanExactlyNRidges_Direct – Locates a specified number of the strongest dark or bright pixel peak along a given path (without a scan map).
- ScanSingleRidge – Locates the strongest dark or bright pixel peak along a given path.

ScanSingleStripe

Header: AVL.h
Namespace: avl
Module: MetrologyBasic

Locates the strongest pair of edges across a given path.

Applications: Very fast detection or measurement of an object defined by a pair of opposite edges.

Syntax

```
void avl::ScanSingleStripe
(
 const avl::Image& inImage,
 const ScanMap& inScanMap,
 const StripeScanParams& inStripeScanParams,
 avl::Selection::Type inStripeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Conditional<avl::Stripe1D>& outStripe,
 atl::Optional<avl::Profile&> outBrightnessProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inScanMap	const ScanMap&		Data precomputed with CreateScanMap
→	inStripeScanParams	const StripeScanParams&	StripeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f MinMagnitude: 5.0f MaxInnerEdgeMagnitude: Nil StripePolarity: Dark MinStripeWidth: 0.0f MaxStripeWidth: Nil)	Parameters controlling the stripe extraction process
→	inStripeSelection	Selection::Type	Selection::Best	Selection mode of the resulting stripe
→	inLocalBlindness	Optional<const LocalBlindness&>	NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
←	outStripe	Conditional<Stripe1D>&		Found stripe
←	outBrightnessProfile	Optional<Profile&>	NIL	Extracted image profile
←	outResponseProfile	Optional<Profile&>	NIL	Profile of the edge (derivative) operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outBrightnessProfile**, **outResponseProfile**.

Read more about Optional Outputs.

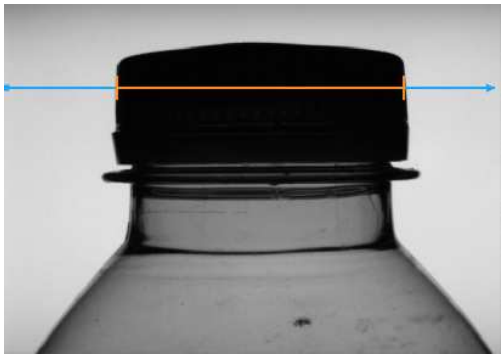
Description

The operation scans the image using **inScanMap** previously generated from a scan path and locates the strongest stripe (i.e. a pair of opposite-polarity edges running across the path) of the given characteristics. If there is no such stripe then the outputs are set to NIL.

Hints

- Define **inStripeScanParams.StripePolarity** to detect a particular edge type, and only that type.
- If no stripe is found, try decreasing **inStripeScanParams.MinMagnitude**. Verify this with the values on the **outResponseProfile** output.
- If consecutive edges are closer than 6 pixels apart, change **inStripeScanParams.ProfileInterpolation** to **Quadratic3**.

Examples



ScanSingleStripe *locates the strongest stripe using a scan map representing the scan path above.*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the 1D Edge Detection toolset. For a comprehensive introduction to this technique please refer to 1D Edge Detection and 1D Edge Detection - Subpixel Precision chapters of our Machine Vision Guide.

See Also

- CreateScanMap – Precomputes a data object that is required for fast 1D edge detection.
- ScanMultipleStripes – Locates multiple pairs of edges across a given path.
- ScanExactlyNStripes – Locates a specified number of multiple pairs of opposite edges across a given path.
- ScanSingleStripe_Direct – Locates the strongest pair of edges across a given path (without a scan map).

ScanSingleStripe_Direct

Header: AVL.h
Namespace: avl
Module: MetrologyBasic

Locates the strongest pair of edges across a given path (without a scan map).

Applications: Very fast detection or measurement of an object defined by a pair of opposite edges.

Syntax

```
void avl::ScanSingleStripe_Direct
(
 const avl::Image& inImage,
 const avl::Path& inScanPath,
 atl::Optional<const avl::CoordinateSystem2D&> inScanPathAlignment,
 int inScanWidth,
 const avl::SamplingParams& inSamplingParams,
 const avl::StripeScanParams& inStripeScanParams,
 avl::Selection::Type inStripeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Conditional<avl::StripeID>& outStripe,
 atl::Optional<avl::Path&> outAlignedScanPath = atl::NIL,
 atl::Optional<avl::Profile&> outBrightnessProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL,
 atl::Array<avl::Path>& diagSamplingPoints,
 float& diagSamplingStep
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inScanPath	const Path&			Path along which the scan is performed
→	inScanPathAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the scan path to the position of the inspected object
→	inScanWidth	int	1 - ∞	5	Width of the scan field in pixels
→	inSamplingParams	const SamplingParams&		SamplingParams (Interpolation: Bilinear SamplingStep: 1.0f SampleCount: Nil)	Parameters controlling the sampling process
→	inStripeScanParams	const StripeScanParams&		StripeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f MinMagnitude: 5.0f MaxInnerEdgeMagnitude: Nil StripePolarity: Dark MinStripeWidth: 0.0f MaxStripeWidth: Nil)	Parameters controlling the stripe extraction process
→	inStripeSelection	Selection::Type			Selection mode of the resulting stripe
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
←	outStripe	Conditional<Stripe1D>&			Found stripe
←	outAlignedScanPath	Optional<Path&>		NIL	Transformed input path
←	outBrightnessProfile	Optional<Profile&>		NIL	Extracted image profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the edge (derivative) operator response
🔍	diagSamplingPoints	Array<Path>&			Array of paths each one containing the sampling points that contributes to a single value of the extracted profile
🔍	diagSamplingStep	float&			Used distance between consecutive sampling points on the scan path

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedScanPath**, **outBrightnessProfile**, **outResponseProfile**.

Read more about Optional Outputs.

Description

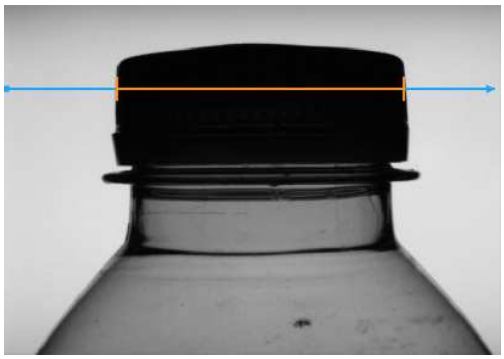
The operation scans the image along **inScanPath** and locates the strongest stripe (i.e. a pair of opposite-polarity edges running across the path) of the given characteristics. If there is no such stripe then the outputs are set to NIL.

The optional parameter **inScanPathAlignment** defines a transform to be performed on the **inScanPath** so that the actual scan path (**outAlignedScanPath**) is adjusted to the position of the object, typically detected by one of Template Matching filters.

Hints

- Define **inStripeScanParams.StripePolarity** to detect a particular edge type, and only that type.
- If no stripe is found, try decreasing **inStripeScanParams.MinMagnitude**. Verify this with the values on the **outResponseProfile** output.
- If consecutive edges are closer than 6 pixels apart, change **inStripeScanParams.ProfileInterpolation** to **Quadratic3**.

Examples



ScanSingleStripe_Direct *locates the strongest stripe across* ***inScanPath****.*

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

This filter is a part of the 1D Edge Detection toolset. For a comprehensive introduction to this technique please refer to 1D Edge Detection and 1D Edge Detection - Subpixel Precision chapters of our Machine Vision Guide.

See Also

- CreateScanMap – Precomputes a data object that is required for fast 1D edge detection.
- ScanMultipleStripes_Direct – Locates multiple pairs of edges across a given path (without a scan map).
- ScanExactlyNStripes_Direct – Locates a specified number of multiple pairs of opposite edges across a given path (without a scan map).
- ScanSingleStripe – Locates the strongest pair of edges across a given path.

76. 1D Edge Detection 3D

Table of content:

CreateSurfaceScanMap

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Precomputes a data object that is required for fast 1D edge detection in 3D.

Applications: Used together with 1D Edge Detection 3D filters, but can be moved before the loop if only the scan parameters do not change.

Syntax

```
void avl::CreateSurfaceScanMap
(
 const avl::SurfaceFormat& inSurfaceFormat,
 const avl::Path& inScanPath,
 atl::Optional<const avl::CoordinateSystem2D&> inScanPathAlignment,
 atl::Optional<float> inSamplingStep,
 int inScanWidth,
 avl::InterpolationMethod::Type inSurfaceInterpolation,
 avl::ScanMap& outScanMap,
 atl::Optional<avl::Path&> outAlignedScanPath = atl::NIL,
 atl::Array<avl::Path>& diagSamplingPoints,
 float& diagSamplingStep
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurfaceFormat	const SurfaceFormat&			Dimensions, depth image pixel type, coordinate offsets and scales of a surface on which edge detection will be performed
→	inScanPath	const Path&			Path along which the scan is performed
→	inScanPathAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the scan path to the position of the inspected object
→	inSamplingStep	Optional<float>	0.0 - ∞	NIL	Desired distance between consecutive sampling points on the scan path; if Nil, the bigger of surface X and Y scales is chosen
→	inScanWidth	int	1 - ∞	5	Width of the scan field in pixels of the surface depth image
→	inSurfaceInterpolation	InterpolationMethod::Type		Bilinear	Interpolation method used for extraction of depth image pixel values
←	outScanMap	ScanMap&			Optimized data object required for 1D edge detection in 3D
←	outAlignedScanPath	Optional<Path&>		NIL	Transformed input path
🔍	diagSamplingPoints	Array<Path>&			Array of paths each one containing the sampling points that contributed to a single value of the extracted profile; in the image coordinate system
🔍	diagSamplingStep	float&			Used distance between consecutive sampling points on the scan path

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedScanPath**.

Read more about Optional Outputs.

Description

The operation creates a scan map from a given **inScanPath**. The scan map can be later used by other 1D Edge Detection 3D filters.

The optional parameter **inScanPathAlignment** defines a transform to be performed on the **inScanPath** so that the actual scan path (**outAlignedScanPath**) is adjusted to the position of the object.

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

See Also

- ScanSingleEdge3D – Locates the strongest change of surface height along a given path.

ScanExactlyNEdges3D

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Locates a specified number of the strongest changes of surface height along a given path.

Syntax

```
void avl::ScanExactlyNEdges3D
(
 const avl::Surface& inSurface,
 const avl::ScanMap& inScanMap,
 const avl::EdgeScanParams3D& inEdgeScanParams,
 int inEdgeCount,
 avl::Selection::Type inEdgeSelection,
 float inMinDistance,
 atl::Optional<float> inMaxDistance,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 atl::Conditional<atl::Array<avl::SurfaceEdge1D> >& outEdges,
 atl::Optional<avl::Profile&> outHeightProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inScanMap	const ScanMap&			Data precomputed with CreateSurfaceScanMap
→	inEdgeScanParams	const EdgeScanParams3D&			Parameters controlling the surface edge extraction process
→	inEdgeCount	int	0 - ∞	1	Number of surface edges to be found
→	inEdgeSelection	Selection::Type		Selection::Best	Selection mode of the resulting edges
→	inMinDistance	float	0.0 - ∞	0.0f	Minimal distance between consecutive edges
→	inMaxDistance	Optional<float>	0.0 - ∞	NIL	Maximal distance between consecutive edges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
←	outEdges	Conditional<Array<SurfaceEdge1D> >&			Found surface edges
←	outHeightProfile	Optional<Profile&>		NIL	Extracted surface height profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the edge (derivative) operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outHeightProfile**, **outResponseProfile**.

Read more about Optional Outputs.

Description

The operation scans the surface using **inScanMap** previously generated from a scan path and finds a set of **inEdgeCount** surface edges perpendicular to the path. If no subset (of **inEdgeCount** elements) of detected edges meets the requirements of **inEdgeScanParams.minMagnitude**, **inMinDistance**, **inEdgeScanParams.edgeTransition** then the outputs are set to NIL.

Note that in case of a scan path which is closed, the parameters controlling the distances between consecutive found objects do not control the distance between the first and the last of the found objects (counting from the beginning of the scan path).

The operation is very similar to ScanExactlyNEdges from 1D Edge Detection category, but there are some substantial differences. One of these is the possibility of absence of information, because some surface points may not exist at all. To detect such edges, where solely change of existence matters, **Valid/Invalid** options of **inEdgeScanParams.EdgeTransition** can be used. Outside the surface domain (i.e. rectangle defined by input surface width, height, offsets and scales) there are no valid or invalid points, so no edge can be found in the direct vicinity of the domain border.

Because in the **Valid/Invalid** mode all edges have equal strength, if **inEdgeSelection** is set to **Best**, it will be implicitly substituted with **First** selection option.

Please note that when the input surface has unequal scales along X and Y axes and the scan path is not parallel to any of the axes, the results may be slightly less accurate because of uneven sampling along axes.

Hints

- Set **inEdgeCount** to the number of edges that are to be found (the N number).
- Define **inEdgeScanParams.EdgeTransition** to detect a particular edge type, and only that type.
- If the expected number of edges cannot be found, try decreasing **inEdgeScanParams.MinMagnitude**. Verify this with the values on the **outResponseProfile** output.
- If consecutive edges are closer than 6 pixels apart, change **inEdgeScanParams.ProfileInterpolation** to **Quadratic3**.
- Adjust **inMinDistance** (in surface coordinates) to filter out false edges that appear in proximity to other edges.

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

See Also

- CreateSurfaceScanMap – Precomputes a data object that is required for fast 1D edge detection in 3D.
- ScanSingleEdge3D – Locates the strongest change of surface height along a given path.
- ScanExactlyNEdges3D – Locates a specified number of the strongest changes of surface height along a given path.
- ScanExactlyNEdges3D_Direct – Locates a specified number of the strongest changes of surface height along a given path.

ScanExactlyNEdges3D_Direct

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Locates a specified number of the strongest changes of surface height along a given path.

Syntax

```
void avl::ScanExactlyNEdges3D_Direct
(
 const avl::Surface& inSurface,
 const avl::Path& inScanPath,
 atl::Optional<const avl::CoordinateSystem2D&> inScanPathAlignment,
 atl::Optional<float> inSamplingStep,
 int inScanWidth,
 avl::InterpolationMethod::Type inSurfaceInterpolation,
 const avl::EdgeScanParams3D& inEdgeScanParams,
 int inEdgeCount,
 avl::Selection::Type inEdgeSelection,
 float inMinDistance,
 atl::Optional<float> inMaxDistance,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 atl::Conditional<atl::Array<avl::SurfaceEdge1D> >& outEdges,
 atl::Optional<avl::Path&> outAlignedScanPath = atl::NIL,
 atl::Optional<avl::Profile&> outHeightProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL,
 atl::Array<avl::Path>& diagSamplingPoints,
 float& diagSamplingStep
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inScanPath	const Path&			Path along which the scan is performed
→	inScanPathAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the scan path to the position of the inspected object
→	inSamplingStep	Optional<float>	0.0 - ∞	NIL	Distance between consecutive sampling points on the scan path; if Nil, the bigger of surface X and Y scales is chosen
→	inScanWidth	int	1 - ∞	5	Width of the scan field in pixels
→	inSurfaceInterpolation	InterpolationMethod::Type		Bilinear	Interpolation method used for extraction of surface points
→	inEdgeScanParams	const EdgeScanParams3D&		EdgeScanParams3D (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f MinMagnitude: 5.0f EdgeTransition: LowToHigh)	Parameters controlling the surface edge extraction process
→	inEdgeCount	int	0 - ∞	1	Number of surface edges to be found
→	inEdgeSelection	Selection::Type			Selection mode of the resulting edges
→	inMinDistance	float	0.0 - ∞	0.0f	Minimal distance between consecutive edges
→	inMaxDistance	Optional<float>	0.0 - ∞	NIL	Maximal distance between consecutive edges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
←	outEdges	Conditional<Array<SurfaceEdge1D> >&			Found surface edges
←	outAlignedScanPath	Optional<Path&>		NIL	Path along which the scan is performed; in the image coordinate system
←	outHeightProfile	Optional<Profile&>		NIL	Extracted surface height profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the edge (derivative) operator response
🔍	diagSamplingPoints	Array<Path>&			Array of paths each one containing the sampling points that contributed to a single value of the extracted profile; in the image coordinate system
🔍	diagSamplingStep	float&			Used distance between consecutive sampling points on the scan path

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedScanPath**, **outHeightProfile**, **outResponseProfile**.

Read more about Optional Outputs.

Description

The operation scans the surface along **inScanPath** and finds a set of **inEdgeCount** surface edges perpendicular to the path. If no subset (of **inEdgeCount** elements) of detected edges meets the requirements of **inEdgeScanParams.minMagnitude**, **inMinDistance**, **inEdgeScanParams.edgeTransition** then the outputs are set to NIL.

The optional parameter **inScanPathAlignment** defines a transform to be performed on the **inScanPath** so that the actual scan path (**outAlignedScanPath**) is adjusted to the position of the object.

Note that in case of a scan path which is closed, the parameters controlling the distances between consecutive found objects do not control the distance between the first and the last of the found objects (counting from the beginning of the scan path).

The operation is very similar to ScanExactlyNEdges_Direct from 1D Edge Detection category, but there are some substantial differences. One of these is the possibility of absence of information, because some surface points may not exist at all. To detect such edges, where solely change of existence matters, **Valid/Invalid** options of **inEdgeScanParams.EdgeTransition** can be used. Outside the surface domain (i.e. rectangle defined by input surface width, height, offsets and scales) there are no valid or invalid points, so no edge can be found in the direct vicinity of the domain border.

Because in the **Valid/Invalid** mode all edges have equal strength, if **inEdgeSelection** is set to **Best**, it will be implicitly substituted with **First** selection option.

Please note that when the input surface has unequal scales along X and Y axes and the scan path is not parallel to any of the axes, the results may be slightly less accurate because of uneven sampling along axes.

Hints

- Set **inEdgeCount** to the number of edges that are to be found (the N number).
- Define **inEdgeScanParams.EdgeTransition** to detect a particular edge type, and only that type.
- If the expected number of edges cannot be found, try decreasing **inEdgeScanParams.MinMagnitude**. Verify this with the values on the **outResponseProfile** output.
- If consecutive edges are closer than 6 pixels apart, change **inEdgeScanParams.ProfileInterpolation** to **Quadratic3**.
- Adjust **inMinDistance** (in surface coordinates) to filter out false edges that appear in proximity to other edges.

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

See Also

- CreateSurfaceScanMap – Precomputes a data object that is required for fast 1D edge detection in 3D.
- ScanSingleEdge3D_Direct – Locates the strongest change of surface height along a given path.
- ScanExactlyNEdges3D_Direct – Locates a specified number of the strongest changes of surface height along a given path.
- ScanExactlyNEdges3D – Locates a specified number of the strongest changes of surface height along a given path.

ScanExactlyNRidges3D

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Locates a specified number of the strongest high or low peaks of surface height along a given path.

Syntax

```
void avl::ScanExactlyNRidges3D
(
 const avl::Surface& inSurface,
 const ScanMap& inScanMap,
 const avl::RidgeScanParams3D& inRidgeScanParams,
 int inRidgeCount,
 avl::Selection::Type inRidgeSelection,
 float inMinDistance,
 atl::Optional<float> inMaxDistance,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 atl::Conditional<atl::Array<avl::SurfaceRidge1D> >& outRidges,
 atl::Optional<avl::Profile&> outHeightProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inSurface	const Surface&			Input surface
➡	inScanMap	const ScanMap&			Data precomputed with CreateSurfaceScanMap
➡	inRidgeScanParams	const RidgeScanParams3D&			Parameters controlling the surface ridge extraction process
➡	inRidgeCount	int	0 - ∞	1	Number of ridges to be found
➡	inRidgeSelection	Selection::Type		Selection::Best	Selection mode of the resulting ridges
➡	inMinDistance	float	0.0 - ∞	0.0f	Minimal distance between consecutive ridges
➡	inMaxDistance	Optional<float>	0.0 - ∞	NIL	Maximal distance between consecutive ridges
➡	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker ridges can be detected in the vicinity of stronger ridges
➡	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
⬅	outRidges	Conditional<Array<SurfaceRidge1D> >&			Found surface ridges
⬅	outHeightProfile	Optional<Profile&>		NIL	Extracted surface height profile
⬅	outResponseProfile	Optional<Profile&>		NIL	Profile of the ridge operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outHeightProfile**, **outResponseProfile**.

Read more about Optional Outputs.

ScanExactlyNRidges3D_Direct

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Locates a specified number of the strongest high or low peaks of surface height along a given path.

Syntax

```
void avl::ScanExactlyNRidges3D_Direct
(
 const avl::Surface& inSurface,
 const avl::Path& inScanPath,
 atl::Optional<const avl::CoordinateSystem2D&> inScanPathAlignment,
 atl::Optional<float> inSamplingStep,
 int inScanWidth,
 avl::InterpolationMethod::Type inSurfaceInterpolation,
 const avl::RidgeScanParams3D& inRidgeScanParams,
 int inRidgeCount,
 avl::Selection::Type inRidgeSelection,
 float inMinDistance,
 atl::Optional<float> inMaxDistance,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 atl::Conditional<atl::Array<avl::SurfaceRidge1D> >& outRidges,
 atl::Optional<avl::Path&> outAlignedScanPath = atl::NIL,
 atl::Optional<avl::Profile&> outHeightProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL,
 atl::Array<avl::Path>& diagSamplingPoints,
 float& diagSamplingStep
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inScanPath	const Path&			Path along which the scan is performed
→	inScanPathAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the scan path to the position of the inspected object
→	inSamplingStep	Optional<float>	0.0 - ∞	NIL	Distance between consecutive sampling points on the scan path; if Nil, the bigger of surface X and Y scales is chosen
→	inScanWidth	int	1 - ∞	5	Width of the scan field in pixels
→	inSurfaceInterpolation	InterpolationMethod::Type		Bilinear	Interpolation method used for extraction of surface points
→	inRidgeScanParams	const RidgeScanParams3D&		RidgeScanParams3D (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f RidgeWidth: 5.0f RidgeMargin: 2.0f RidgeOperator: Minimum MinMagnitude: 5.0f RidgePolarity: Any)	Parameters controlling the ridge extraction process
→	inRidgeCount	int	0 - ∞	1	Number of ridges to be found
→	inRidgeSelection	Selection::Type			Selection mode of the resulting ridges
→	inMinDistance	float	0.0 - ∞	0.0f	Minimal distance between consecutive ridges
→	inMaxDistance	Optional<float>	0.0 - ∞	NIL	Maximal distance between consecutive ridges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker ridges can be detected in the vicinity of stronger ridges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
←	outRidges	Conditional<Array<SurfaceRidge1D> >&			Found surface ridges
←	outAlignedScanPath	Optional<Path&>		NIL	Path along which the scan is performed; in the image coordinate system
←	outHeightProfile	Optional<Profile&>		NIL	Extracted surface height profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the ridge operator response
🔍	diagSamplingPoints	Array<Path>&			Array of paths each one containing the sampling points that contributed to a single value of the extracted profile; in the image coordinate system
🔍	diagSamplingStep	float&			Used distance between consecutive sampling points on the scan path

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedScanPath**, **outHeightProfile**, **outResponseProfile**.

Read more about Optional Outputs.

ScanExactlyNStripes3D

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Locates a specified number of multiple pairs of changes of surface height along a given path.

Syntax

```
void avl::ScanExactlyNStripes3D
(
 const avl::Surface& inSurface,
 const ScanMap& inScanMap,
 const StripeScanParams3D& inStripeScanParams,
 int inStripeCount,
 avl::Selection::Type inStripeSelection,
 float inMinGapWidth,
 atl::Optional<float> inMaxGapWidth,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 atl::Conditional<atl::Array<avl::SurfaceStripe1D> >& outStripes,
 atl::Optional<avl::Profile&> outHeightProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inScanMap	const ScanMap&			Data precomputed with CreateSurfaceScanMap
→	inStripeScanParams	const StripeScanParams3D&			Parameters controlling the surface stripe extraction process
→	inStripeCount	int	0 - ∞	1	Number of surface stripes to be found
→	inStripeSelection	Selection::Type		Selection::Best	Selection mode of the resulting stripes
→	inMinGapWidth	float	0.0 - ∞		Minimal distance between consecutive surface stripes
→	inMaxGapWidth	Optional<float>	0.0 - ∞	NIL	Maximal distance between consecutive surface stripes
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
←	outStripes	Conditional<Array<SurfaceStripe1D> >&			Found surface stripes
←	outHeightProfile	Optional<Profile&>		NIL	Extracted surface height profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the edge (derivative) operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outHeightProfile**, **outResponseProfile**.

Read more about Optional Outputs.

ScanExactlyNStripes3D_Direct

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Locates a specified number of multiple pairs of changes of surface height along a given path.

Syntax

```
void avl::ScanExactlyNStripes3D_Direct
(
 const avl::Surface& inSurface,
 const avl::Path& inScanPath,
 atl::Optional<const avl::CoordinateSystem2D&> inScanPathAlignment,
 atl::Optional<float> inSamplingStep,
 int inScanWidth,
 avl::InterpolationMethod::Type inSurfaceInterpolation,
 const avl::StripeScanParams3D& inStripeScanParams,
 int inStripeCount,
 avl::Selection::Type inStripeSelection,
 float inMinGapWidth,
 atl::Optional<float> inMaxGapWidth,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 atl::Conditional<atl::Array<avl::SurfaceStripe1D> >& outStripes,
 atl::Optional<avl::Path&> outAlignedScanPath = atl::NIL,
 atl::Optional<avl::Profile&> outHeightProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL,
 atl::Array<avl::Path>& diagSamplingPoints,
 float& diagSamplingStep
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inScanPath	const Path&			Path along which the scan is performed
→	inScanPathAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the scan path to the position of the inspected object
→	inSamplingStep	Optional<float>	0.0 - ∞	NIL	Distance between consecutive sampling points on the scan path; if Nil, the bigger of surface X and Y scales is chosen
→	inScanWidth	int	1 - ∞	5	Width of the scan field in pixels
→	inSurfaceInterpolation	InterpolationMethod::Type		Bilinear	Interpolation method used for extraction of surface points
→	inStripeScanParams	const StripeScanParams3D&		StripeScanParams3D (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f MinMagnitude: 5.0f MaxInnerEdgeMagnitude: Nil StripePolarity: Low MinStripeWidth: 0.0f MaxStripeWidth: Nil)	Parameters controlling the surface stripe extraction process
→	inStripeCount	int	0 - ∞	1	Number of surface stripes to be found
→	inStripeSelection	Selection::Type			Selection mode of the resulting stripes
→	inMinGapWidth	float	0.0 - ∞	0.0f	Minimal distance between consecutive surface stripes
→	inMaxGapWidth	Optional<float>	0.0 - ∞	NIL	Maximal distance between consecutive surface stripes
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
←	outStripes	Conditional<Array<SurfaceStripe1D> >&			Found surface stripes
←	outAlignedScanPath	Optional<Path&>		NIL	Path along which the scan is performed; in the image coordinate system
←	outHeightProfile	Optional<Profile&>		NIL	Extracted surface height profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the edge (derivative) operator response
🔍	diagSamplingPoints	Array<Path>&			Array of paths each one containing the sampling points that contributed to a single value of the extracted profile; in the image coordinate system
🔍	diagSamplingStep	float&			Used distance between consecutive sampling points on the scan path

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedScanPath**, **outHeightProfile**, **outResponseProfile**.

Read more about Optional Outputs.

ScanMultipleEdges3D

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Locates multiple changes of surface height along a given path.

Syntax

```
void avl::ScanMultipleEdges3D
(
 const avl::Surface& inSurface,
 const avl::ScanMap& inScanMap,
 const avl::EdgeScanParams3D& inEdgeScanParams,
 float inMinDistance,
 atl::Optional<float> inMaxDistance,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 atl::Array<avl::SurfaceEdge1D>& outEdges,
 atl::Optional<avl::Profile&> outHeightProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inScanMap	const ScanMap&			Data precomputed with CreateSurfaceScanMap
→	inEdgeScanParams	const EdgeScanParams3D&			Parameters controlling the surface edge extraction process
→	inMinDistance	float	0.0 - ∞		Minimal distance between consecutive edges
→	inMaxDistance	Optional<float>	0.0 - ∞	NIL	Maximal distance between consecutive edges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	NIL	Maximal number of consecutive not existing profile points
←	outEdges	Array<SurfaceEdge1D>&			Found surface edges
←	outHeightProfile	Optional<Profile&>		NIL	Extracted surface height profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the edge (derivative) operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outHeightProfile**, **outResponseProfile**.

Read more about Optional Outputs.

Description

The operation scans the surface using **inScanMap** previously generated from a scan path and detects surface edges perpendicular to the path. Depending on the **inEdgeScanParams.edgeTransition** parameter, edges representing increase or decrease (or both) of surface height along the path will be taken into account.

When the number of edges to be found is known, one can use the ScanExactlyNEdges3D.

Note that in case of a scan path which is closed, the parameters controlling the distances between consecutive found objects do not control the distance between the first and the last of the found objects (counting from the beginning of the scan path).

The operation is very similar to ScanMultipleEdges from 1D Edge Detection category, but there are some substantial differences. One of these is the possibility of absence of information, because some surface points may not exist at all. To detect such edges, where solely change of existence matters, **Valid/Invalid** options of **inEdgeScanParams.EdgeTransition** can be used. Outside the surface domain (i.e. rectangle defined by input surface width, height, offsets and scales) there are no valid or invalid points, so no edge can be found in the direct vicinity of the domain border.

Please note that when the input surface has unequal scales along X and Y axes and the scan path is not parallel to any of the axes, the results may be slightly less accurate because of uneven sampling along axes.

Hints

- Define **inEdgeScanParams.EdgeTransition** to detect a particular edge type, and only that type.
- If too few edges are found, try decreasing **inEdgeScanParams.MinMagnitude**. Verify this with the values on the **outResponseProfile** output.
- If consecutive edges are closer than 6 pixels apart, change **inEdgeScanParams.ProfileInterpolation** to **Quadratic3**.
- Adjust **inMinDistance** (in surface coordinates) to filter out false edges that appear in proximity to other edges.

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

See Also

- CreateSurfaceScanMap – Precomputes a data object that is required for fast 1D edge detection in 3D.
- ScanSingleEdge3D – Locates the strongest change of surface height along a given path.
- ScanExactlyNEdges3D – Locates a specified number of the strongest changes of surface height along a given path.
- ScanMultipleEdges3D_Direct – Locates multiple changes of surface height along a given path.

 ScanMultipleEdges3D_Direct

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Locates multiple changes of surface height along a given path.

Syntax

```
void avl::ScanMultipleEdges3D_Direct
(
 const avl::Surface& inSurface,
 const avl::Path& inScanPath,
 atl::Optional<const avl::CoordinateSystem2D&> inScanPathAlignment,
 atl::Optional<float> inSamplingStep,
 int inScanWidth,
 avl::InterpolationMethod::Type inSurfaceInterpolation,
 const avl::EdgeScanParams3D& inEdgeScanParams,
 float inMinDistance,
 atl::Optional<float> inMaxDistance,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 atl::Array<avl::SurfaceEdge1D>& outEdges,
 atl::Optional<avl::Path&> outAlignedScanPath = atl::NIL,
 atl::Optional<avl::Profile&> outHeightProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL,
 atl::Array<avl::Path>& diagSamplingPoints,
 float& diagSamplingStep
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inScanPath	const Path&			Path along which the scan is performed
→	inScanPathAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the scan path to the position of the inspected object
→	inSamplingStep	Optional<float>	0.0 - ∞	NIL	Distance between consecutive sampling points on the scan path; if Nil, the bigger of surface X and Y scales is chosen
→	inScanWidth	int	1 - ∞	5	Width of the scan field in pixels
→	inSurfaceInterpolation	InterpolationMethod::Type		Bilinear	Interpolation method used for extraction of surface points
→	inEdgeScanParams	const EdgeScanParams3D&		EdgeScanParams3D (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f MinMagnitude: 5.0f EdgeTransition: LowToHigh)	Parameters controlling the surface edge extraction process
→	inMinDistance	float	0.0 - ∞	0.0f	Minimal distance between consecutive edges
→	inMaxDistance	Optional<float>	0.0 - ∞	NIL	Maximal distance between consecutive edges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
←	outEdges	Array<SurfaceEdge1D>&			Found surface edges
←	outAlignedScanPath	Optional<Path&>		NIL	Path along which the scan is performed; in the image coordinate system
←	outHeightProfile	Optional<Profile&>		NIL	Extracted surface height profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the edge (derivative) operator response
🔍	diagSamplingPoints	Array<Path>&			Array of paths each one containing the sampling points that contributed to a single value of the extracted profile; in the image coordinate system
🔍	diagSamplingStep	float&			Used distance between consecutive sampling points on the scan path

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedScanPath**, **outHeightProfile**, **outResponseProfile**.

Read more about Optional Outputs.

Description

The operation scans the surface along **inScanPath** and detects surface edges perpendicular to the path. Depending on the **inEdgeScanParams.edgeTransition** parameter, edges representing increase or decrease (or both) of surface height along the path will be taken into account.

When the number of edges to be found is known, one can use the ScanExactlyNEdges3D_Direct.

The optional parameter **inScanPathAlignment** defines a transform to be performed on the **inScanPath** so that the actual scan path (**outAlignedScanPath**) is adjusted to the position of the object.

Note that in case of a scan path which is closed, the parameters controlling the distances between consecutive found objects do not control the distance between the first and the last of the found objects (counting from the beginning of the scan path).

The operation is very similar to ScanMultipleEdges_Direct from 1D Edge Detection category, but there are some substantial differences. One of these is the possibility of absence of information, because some surface points may not exist at all. To detect such edges, where solely change of existence matters, **Valid/Invalid** options of **inEdgeScanParams.EdgeTransition** can be used. Outside the surface domain (i.e. rectangle defined by input surface width, height, offsets and scales) there are no valid or invalid points, so no edge can be found in the direct vicinity of the domain border.

Please note that when the input surface has unequal scales along X and Y axes and the scan path is not parallel to any of the axes, the results may be slightly less accurate because of uneven sampling along axes.

Hints

- Define **inEdgeScanParams.EdgeTransition** to detect a particular edge type, and only that type.
- If too few edges are found, try decreasing **inEdgeScanParams.MinMagnitude**. Verify this with the values on the **outResponseProfile** output.
- If consecutive edges are closer than 6 pixels apart, change **inEdgeScanParams.ProfileInterpolation** to **Quadratic3**.
- Adjust **inMinDistance** (in surface coordinates) to filter out false edges that appear in proximity to other edges.

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

See Also

- CreateSurfaceScanMap – Precomputes a data object that is required for fast 1D edge detection in 3D.
- ScanSingleEdge3D_Direct – Locates the strongest change of surface height along a given path.
- ScanExactlyNEdges3D_Direct – Locates a specified number of the strongest changes of surface height along a given path.
- ScanMultipleEdges3D – Locates multiple changes of surface height along a given path.

ScanMultipleRidges3D

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Locates multiple high or low peaks of surface height along a given path.

Syntax

```
void avl::ScanMultipleRidges3D
(
 const avl::Surface& inSurface,
 const ScanMap& inScanMap,
 const avl::RidgeScanParams3D& inRidgeScanParams,
 float inMinDistance,
 atl::Optional<float> inMaxDistance,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 atl::Array<avl::SurfaceRidge1D>& outRidges,
 atl::Optional<avl::Profile&> outHeightProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inScanMap	const ScanMap&			Data precomputed with CreateSurfaceScanMap
→	inRidgeScanParams	const RidgeScanParams3D&			Parameters controlling the surface ridge extraction process
→	inMinDistance	float	0.0 - ∞	0.0f	Minimal distance between consecutive ridges
→	inMaxDistance	Optional<float>	0.0 - ∞	NIL	Maximal distance between consecutive ridges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker ridges can be detected in the vicinity of stronger ridges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
←	outRidges	Array<SurfaceRidge1D>&			Found surface ridges
←	outHeightProfile	Optional<Profile&>		NIL	Extracted surface height profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the ridge operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outHeightProfile**, **outResponseProfile**.

Read more about Optional Outputs.

ScanMultipleRidges3D_Direct

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Locates multiple high or low peaks of surface height along a given path.

Syntax

```
void avl::ScanMultipleRidges3D_Direct
(
 const avl::Surface& inSurface,
 const avl::Path& inScanPath,
 atl::Optional<const avl::CoordinateSystem2D&> inScanPathAlignment,
 atl::Optional<float> inSamplingStep,
 int inScanWidth,
 avl::InterpolationMethod::Type inSurfaceInterpolation,
 const avl::RidgeScanParams3D& inRidgeScanParams,
 float inMinDistance,
 atl::Optional<float> inMaxDistance,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 atl::Array<avl::SurfaceRidge1D>& outRidges,
 atl::Optional<avl::Path&> outAlignedScanPath = atl::NIL,
 atl::Optional<avl::Profile&> outHeightProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL,
 atl::Array<avl::Path>& diagSamplingPoints,
 float& diagSamplingStep
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inScanPath	const Path&			Path along which the scan is performed
→	inScanPathAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the scan path to the position of the inspected object
→	inSamplingStep	Optional<float>	0.0 - ∞	NIL	Distance between consecutive sampling points on the scan path; if Nil, the bigger of surface X and Y scales is chosen
→	inScanWidth	int	1 - ∞	5	Width of the scan field in pixels
→	inSurfaceInterpolation	InterpolationMethod::Type		Bilinear	Interpolation method used for extraction of surface points
→	inRidgeScanParams	const RidgeScanParams3D&		RidgeScanParams3D (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f RidgeWidth: 5.0f RidgeMargin: 2.0f RidgeOperator: Minimum MinMagnitude: 5.0f RidgePolarity: Any)	Parameters controlling the ridge extraction process
→	inMinDistance	float	0.0 - ∞	0.0f	Minimal distance between consecutive ridges
→	inMaxDistance	Optional<float>	0.0 - ∞	NIL	Maximal distance between consecutive ridges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker ridges can be detected in the vicinity of stronger ridges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
←	outRidges	Array<SurfaceRidge1D>&			Found surface ridges
←	outAlignedScanPath	Optional<Path&>		NIL	Path along which the scan is performed; in the image coordinate system
←	outHeightProfile	Optional<Profile&>		NIL	Extracted surface height profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the ridge operator response
🔍	diagSamplingPoints	Array<Path>&			Array of paths each one containing the sampling points that contributed to a single value of the extracted profile; in the image coordinate system
🔍	diagSamplingStep	float&			Used distance between consecutive sampling points on the scan path

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedScanPath**, **outHeightProfile**, **outResponseProfile**.

Read more about Optional Outputs.

ScanMultipleStripes3D

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Locates multiple pairs of changes of surface height along a given path.

Syntax

```
void avl::ScanMultipleStripes3D
(
 const avl::Surface& inSurface,
 const avl::ScanMap& inScanMap,
 const avl::StripeScanParams3D& inStripeScanParams,
 float inMinGapWidth,
 atl::Optional<float> inMaxGapWidth,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 atl::Array<avl::SurfaceStripe1D>& outStripes,
 atl::Optional<avl::Profile&> outHeightProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inScanMap	const ScanMap&			Data precomputed with CreateSurfaceScanMap
→	inStripeScanParams	const StripeScanParams3D&			Parameters controlling the surface stripe extraction process
→	inMinGapWidth	float	0.0 - ∞	0.0f	Minimal distance between consecutive surface stripes
→	inMaxGapWidth	Optional<float>	0.0 - ∞	NIL	Maximal distance between consecutive surface stripes
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
←	outStripes	Array<SurfaceStripe1D>&			Found surface stripes
←	outHeightProfile	Optional<Profile&>		NIL	Extracted surface height profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the edge (derivative) operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outHeightProfile**, **outResponseProfile**.

Read more about Optional Outputs.

ScanMultipleStripes3D_Direct

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Locates multiple pairs of changes of surface height along a given path.

Syntax

```
void avl::ScanMultipleStripes3D_Direct
(
 const avl::Surface& inSurface,
 const avl::Path& inScanPath,
 atl::Optional<const avl::CoordinateSystem2D&> inScanPathAlignment,
 atl::Optional<float> inSamplingStep,
 int inScanWidth,
 avl::InterpolationMethod::Type inSurfaceInterpolation,
 const avl::StripeScanParams3D& inStripeScanParams,
 float inMinGapWidth,
 atl::Optional<float> inMaxGapWidth,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 atl::Array<avl::SurfaceStripe1D>& outStripes,
 atl::Optional<avl::Path&> outAlignedScanPath = atl::NIL,
 atl::Optional<avl::Profile&> outHeightProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL,
 atl::Array<avl::Path>& diagSamplingPoints,
 float& diagSamplingStep
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inScanPath	const Path&			Path along which the scan is performed
→	inScanPathAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the scan path to the position of the inspected object
→	inSamplingStep	Optional<float>	0.0 - ∞	NIL	Distance between consecutive sampling points on the scan path; if Nil, the bigger of surface X and Y scales is chosen
→	inScanWidth	int	1 - ∞	5	Width of the scan field in pixels
→	inSurfaceInterpolation	InterpolationMethod::Type		Bilinear	Interpolation method used for extraction of surface points
→	inStripeScanParams	const StripeScanParams3D&		StripeScanParams3D (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f MinMagnitude: 5.0f MaxInnerEdgeMagnitude: Nil StripePolarity: Low MinStripeWidth: 0.0f MaxStripeWidth: Nil)	Parameters controlling the surface stripe extraction process
→	inMinGapWidth	float	0.0 - ∞	0.0f	Minimal distance between consecutive surface stripes
→	inMaxGapWidth	Optional<float>	0.0 - ∞	NIL	Maximal distance between consecutive surface stripes
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
←	outStripes	Array<SurfaceStripe1D>&			Found surface stripes
←	outAlignedScanPath	Optional<Path&>		NIL	Path along which the scan is performed; in the image coordinate system
←	outHeightProfile	Optional<Profile&>		NIL	Extracted surface height profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the edge (derivative) operator response
🔍	diagSamplingPoints	Array<Path>&			Array of paths each one containing the sampling points that contributed to a single value of the extracted profile; in the image coordinate system
🔍	diagSamplingStep	float&			Used distance between consecutive sampling points on the scan path

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedScanPath**, **outHeightProfile**, **outResponseProfile**.

Read more about Optional Outputs.

ScanSingleEdge3D

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Locates the strongest change of surface height along a given path.

Syntax

```
void avl::ScanSingleEdge3D
(
 const avl::Surface& inSurface,
 const avl::ScanMap& inScanMap,
 const avl::EdgeScanParams3D& inEdgeScanParams,
 avl::Selection::Type inEdgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 atl::Conditional<avl::SurfaceEdge1D>& outEdge,
 atl::Optional<avl::Profile&> outHeightProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inScanMap	const ScanMap&			Data precomputed with CreateSurfaceScanMap
→	inEdgeScanParams	const EdgeScanParams3D&			Parameters controlling the surface edge extraction process
→	inEdgeSelection	Selection::Type		Selection::Best	Selection mode of the resulting edge
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
←	outEdge	Conditional<SurfaceEdge1D>&			Found surface edge
←	outHeightProfile	Optional<Profile&>		NIL	Extracted surface height profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the edge (derivative) operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outHeightProfile**, **outResponseProfile**.

Read more about Optional Outputs.

Description

The operation scans the surface using **inScanMap** previously generated from a scan path and locates the strongest edge perpendicular to the path. If the strongest edge is weaker than **inEdgeScanParams.minMagnitude** then the outputs are set to NIL.

The operation is very similar to ScanSingleEdge from 1D Edge Detection category, but there are some substantial differences. One of these is the possibility of absence of information, because some surface points may not exist at all. To detect such edges, where solely change of existence matters, **Valid/Invalid** options of **inEdgeScanParams.EdgeTransition** can be used. Outside the surface domain (i.e. rectangle defined by input surface width, height, offsets and scales) there are no valid or invalid points, so no edge can be found in the direct vicinity of the domain border.

Because in the **Valid/Invalid** mode all edges have equal strength, if **inEdgeSelection** is set to **Best**, it will be implicitly substituted with **First** selection option.

Please note that when the input surface has unequal scales along X and Y axes and the scan path is not parallel to any of the axes, the results may be slightly less accurate because of uneven sampling along axes.

Hints

- Define **inEdgeScanParams.EdgeTransition** to detect a particular edge type, and only that type.
- If no edge is found, try decreasing **inEdgeScanParams.MinMagnitude**. Verify this with the values on the **outResponseProfile** output.
- If consecutive edges are closer than 6 pixels apart, change **inEdgeScanParams.ProfileInterpolation** to **Quadratic3**.

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

See Also

- CreateSurfaceScanMap – Precomputes a data object that is required for fast 1D edge detection in 3D.
- ScanMultipleEdges3D – Locates multiple changes of surface height along a given path.
- ScanExactlyNEdges3D – Locates a specified number of the strongest changes of surface height along a given path.
- ScanSingleEdge3D_Direct – Locates the strongest change of surface height along a given path.

ScanSingleEdge3D_Direct

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Locates the strongest change of surface height along a given path.

Syntax

```
void avl::ScanSingleEdge3D_Direct
(
 const avl::Surface& inSurface,
 const avl::Path& inScanPath,
 atl::Optional<const avl::CoordinateSystem2D&> inScanPathAlignment,
 atl::Optional<float> inSamplingStep,
 int inScanWidth,
 avl::InterpolationMethod::Type inSurfaceInterpolation,
 const avl::EdgeScanParams3D& inEdgeScanParams,
 avl::Selection::Type inEdgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 atl::Conditional<avl::SurfaceEdge1D>& outEdge,
 atl::Optional<avl::Path&> outAlignedScanPath = atl::NIL,
 atl::Optional<avl::Profile&> outHeightProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL,
 atl::Array<avl::Path>& diagSamplingPoints,
 float& diagSamplingStep
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inScanPath	const Path&			Path along which the scan is performed
→	inScanPathAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the scan path to the position of the inspected object
→	inSamplingStep	Optional<float>	0.0 - ∞	NIL	Distance between consecutive sampling points on the scan path; if Nil, the bigger of surface X and Y scales is chosen
→	inScanWidth	int	1 - ∞	5	Width of the scan field in pixels
→	inSurfaceInterpolation	InterpolationMethod::Type		Bilinear	Interpolation method used for extraction of surface points
→	inEdgeScanParams	const EdgeScanParams3D&		EdgeScanParams3D (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f MinMagnitude: 5.0f EdgeTransition: LowToHigh)	Parameters controlling the surface edge extraction process
→	inEdgeSelection	Selection::Type			If many edge points are possible, defines which one is selected
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
←	outEdge	Conditional<SurfaceEdge1D>&			Found surface edge
←	outAlignedScanPath	Optional<Path&>		NIL	Path along which the scan is performed; in the image coordinate system
←	outHeightProfile	Optional<Profile&>		NIL	Extracted surface height profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the edge (derivative) operator response
🔍	diagSamplingPoints	Array<Path>&			Array of paths each one containing the sampling points that contributed to a single value of the extracted profile; in the image coordinate system
🔍	diagSamplingStep	float&			Used distance between consecutive sampling points on the scan path

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedScanPath**, **outHeightProfile**, **outResponseProfile**.

Read more about Optional Outputs.

Description

The operation scans the surface along **inScanPath** and locates the strongest edge perpendicular to the path. If the strongest edge is weaker than **inEdgeScanParams.minMagnitude** then the outputs are set to NIL.

The optional parameter **inScanPathAlignment** defines a transform to be performed on the **inScanPath** so that the actual scan path (**outAlignedScanPath**) is adjusted to the position of the object.

The operation is very similar to ScanSingleEdge_Direct from 1D Edge Detection category, but there are some substantial differences. One of these is the possibility of absence of information, because some surface points may not exist at all. To detect such edges, where solely change of existence matters, **Valid/Invalid** options of **inEdgeScanParams.EdgeTransition** can be used. Outside the surface domain (i.e. rectangle defined by input surface width, height, offsets and scales) there are no valid or invalid points, so no edge can be found in the direct vicinity of the domain border.

Because in the **Valid/Invalid** mode all edges have equal strength, if **inEdgeSelection** is set to **Best**, it will be implicitly substituted with **First** selection option.

Please note that when the input surface has unequal scales along X and Y axes and the scan path is not parallel to any of the axes, the results may be slightly less accurate because of uneven sampling along axes.

Hints

- Define **inEdgeScanParams.EdgeTransition** to detect a particular edge type, and only that type.
- If no edge is found, try decreasing **inEdgeScanParams.MinMagnitude**. Verify this with the values on the **outResponseProfile** output.
- If consecutive edges are closer than 6 pixels apart, change **inEdgeScanParams.ProfileInterpolation** to **Quadratic3**.

Remarks

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

See Also

- CreateSurfaceScanMap – Precomputes a data object that is required for fast 1D edge detection in 3D.
- ScanMultipleEdges3D_Direct – Locates multiple changes of surface height along a given path.
- ScanExactlyNEdges3D_Direct – Locates a specified number of the strongest changes of surface height along a given path.
- ScanSingleEdge3D – Locates the strongest change of surface height along a given path.

ScanSingleRidge3D

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Locates the strongest high or low peak of surface height along a given path.

Syntax

```
void avl::ScanSingleRidge3D
(
 const avl::Surface& inSurface,
 const avl::ScanMap& inScanMap,
 const avl::RidgeScanParams3D& inRidgeScanParams,
 avl::Selection::Type inRidgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 atl::Conditional<avl::SurfaceRidge1D>& outRidge,
 atl::Optional<avl::Profile&> outHeightProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inScanMap	const ScanMap&			Data precomputed with CreateSurfaceScanMap
→	inRidgeScanParams	const RidgeScanParams3D&			Parameters controlling the surface ridge extraction process
→	inRidgeSelection	Selection::Type		Selection::Best	Selection mode of the resulting ridge
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker ridges can be detected in the vicinity of stronger ridges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
←	outRidge	Conditional<SurfaceRidge1D>&			Found surface ridge
←	outHeightProfile	Optional<Profile&>		NIL	Extracted surface height profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the ridge operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outHeightProfile**, **outResponseProfile**.

Read more about Optional Outputs.

ScanSingleRidge3D_Direct

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Locates the strongest high or low peak of surface height along a given path.

Syntax

```
void avl::ScanSingleRidge3D_Direct
(
 const avl::Surface& inSurface,
 const avl::Path& inScanPath,
 atl::Optional<const avl::CoordinateSystem2D&> inScanPathAlignment,
 atl::Optional<float> inSamplingStep,
 int inScanWidth,
 avl::InterpolationMethod::Type inSurfaceInterpolation,
 const avl::RidgeScanParams3D& inRidgeScanParams,
 avl::Selection::Type inRidgeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 atl::Conditional<avl::SurfaceRidge1D>& outRidge,
 atl::Optional<avl::Path&> outAlignedScanPath = atl::NIL,
 atl::Optional<avl::Profile&> outHeightProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL,
 atl::Array<avl::Path>& diagSamplingPoints,
 float& diagSamplingStep
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inScanPath	const Path&			Path along which the scan is performed
→	inScanPathAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the scan path to the position of the inspected object
→	inSamplingStep	Optional<float>	0.0 - ∞	NIL	Distance between consecutive sampling points on the scan path; if Nil, the bigger of surface X and Y scales is chosen
→	inScanWidth	int	1 - ∞	5	Width of the scan field in pixels
→	inSurfaceInterpolation	InterpolationMethod::Type		Bilinear	Interpolation method used for extraction of surface points
→	inRidgeScanParams	const RidgeScanParams3D&		RidgeScanParams3D (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f RidgeWidth: 5.0f RidgeMargin: 2.0f RidgeOperator: Minimum MinMagnitude: 5.0f RidgePolarity: Any)	Parameters controlling the surface ridge extraction process
→	inRidgeSelection	Selection::Type			Selection mode of the resulting ridge
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker ridges can be detected in the vicinity of stronger ridges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
←	outRidge	Conditional<SurfaceRidge1D>&			Found surface ridge
←	outAlignedScanPath	Optional<Path&>		NIL	Path along which the scan is performed; in the image coordinate system
←	outHeightProfile	Optional<Profile&>		NIL	Extracted surface height profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the ridge operator response
🔍	diagSamplingPoints	Array<Path>&			Array of paths each one containing the sampling points that contributed to a single value of the extracted profile; in the image coordinate system
🔍	diagSamplingStep	float&			Used distance between consecutive sampling points on the scan path

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedScanPath**, **outHeightProfile**, **outResponseProfile**.

Read more about Optional Outputs.

ScanSingleStripe3D

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Locates the strongest pair of changes of surface height along a given path.

Syntax

```
void avl::ScanSingleStripe3D
(
 const avl::Surface& inSurface,
 const avl::ScanMap& inScanMap,
 const avl::StripeScanParams3D& inStripeScanParams,
 avl::Selection::Type inStripeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 atl::Conditional<avl::SurfaceStripe1D>& outStripe,
 atl::Optional<avl::Profile&> outHeightProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inScanMap	const ScanMap&			Data precomputed with CreateSurfaceScanMap
→	inStripeScanParams	const StripeScanParams3D&			Parameters controlling the surface stripe extraction process
→	inStripeSelection	Selection::Type		Selection::Best	Selection mode of the resulting stripe
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
←	outStripe	Conditional<SurfaceStripe1D>&			Found surface stripe
←	outHeightProfile	Optional<Profile&>		NIL	Extracted surface height profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the edge (derivative) operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outHeightProfile**, **outResponseProfile**.

Read more about Optional Outputs.

ScanSingleStripe3D_Direct

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Locates the strongest pair of changes of surface height along a given path.

Syntax

```
void avl::ScanSingleStripe3D_Direct
(
 const avl::Surface& inSurface,
 const avl::Path& inScanPath,
 atl::Optional<const avl::CoordinateSystem2D&> inScanPathAlignment,
 atl::Optional<float> inSamplingStep,
 int inScanWidth,
 avl::InterpolationMethod::Type inSurfaceInterpolation,
 const avl::StripeScanParams3D& inStripeScanParams,
 avl::Selection::Type inStripeSelection,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Optional<int> inMaxProfileGapWidth,
 atl::Conditional<avl::SurfaceStripe1D>& outStripe,
 atl::Optional<avl::Path&> outAlignedScanPath = atl::NIL,
 atl::Optional<avl::Profile&> outHeightProfile = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL,
 atl::Array<avl::Path>& diagSamplingPoints,
 float& diagSamplingStep
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inScanPath	const Path&			Path along which the scan is performed
→	inScanPathAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the scan path to the position of the inspected object
→	inSamplingStep	Optional<float>	0.0 - ∞	NIL	Distance between consecutive sampling points on the scan path; if Nil, the bigger of surface X and Y scales is chosen
→	inScanWidth	int	1 - ∞	5	Width of the scan field in pixels
→	inSurfaceInterpolation	InterpolationMethod::Type		Bilinear	Interpolation method used for extraction of surface points
→	inStripeScanParams	const StripeScanParams3D&		StripeScanParams3D (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f MinMagnitude: 5.0f MaxInnerEdgeMagnitude: Nil StripePolarity: Low MinStripeWidth: 0.0f MaxStripeWidth: Nil)	Parameters controlling the surface stripe extraction process
→	inStripeSelection	Selection::Type			Selection mode of the resulting surface stripe
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
→	inMaxProfileGapWidth	Optional<int>	0 - ∞	1	Maximal number of consecutive not existing profile points
←	outStripe	Conditional<SurfaceStripe1D>&			Found surface stripe
←	outAlignedScanPath	Optional<Path&>		NIL	Path along which the scan is performed; in the image coordinate system
←	outHeightProfile	Optional<Profile&>		NIL	Extracted surface height profile
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the edge (derivative) operator response
🔍	diagSamplingPoints	Array<Path>&			Array of paths each one containing the sampling points that contributed to a single value of the extracted profile; in the image coordinate system
🔍	diagSamplingStep	float&			Used distance between consecutive sampling points on the scan path

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedScanPath**, **outHeightProfile**, **outResponseProfile**.

Read more about Optional Outputs.

77. Histogram Spatial Transforms

Table of content:

CropHistogram

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Confines a histogram to its continuous segment.

Syntax

```
void avl::CropHistogram
(
 const avl::Histogram& inHistogram,
 const int inStart,
 const int inBinCount,
 avl::Histogram& outHistogram
)
```

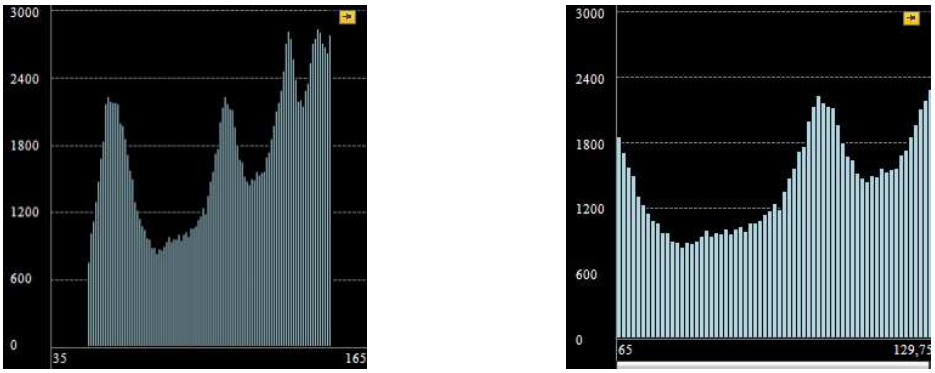
Parameters

	Name	Type	Range	Default	Description
→	inHistogram	const Histogram&			Input histogram
→	inStart	const int	0 - ∞		Index of the first element of the input histogram that will be included in the output histogram
→	inBinCount	const int	0 - ∞		Number of bins in the output histogram
←	outHistogram	Histogram&			Output histogram

Description

Operation crops histogram to bins which are enclosed in **inBinCount** bins beginning from **inStart** bin.

Examples



CropHistogram *performed on the sample histogram with* ***inStart*** *= 15,* ***inBinCount*** *= 65.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Selected index range exceeds the input histogram in CropHistogram.

See Also

- DivideHistogram – Divides each bin value by a number.
- MultiplyHistogram – Multiplies each bin value by a number.
- SubtractFromHistogram – Decreases each bin value by a number.

78. Profile Spatial Transforms

Table of content:

CropProfile

Header: AVL.h
Namespace: avl
Module: FoundationPro

Confines a profile to its continuous subsequence.

Syntax

```
void avl::CropProfile
(
 const avl::Profile& inProfile,
 const int inStart,
 atl::Optional<const int&> inLength,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Range	Default	Description
→	inProfile	const Profile&			Input profile
→	inStart	const int	0 - ∞		Index of the first element of the input profile that will be included in the output profile
→	inLength	Optional<const int&>	0 - ∞	NIL	Length of the output profile
←	outProfile	Profile&			Output profile

Description

The operation crops profile to a new size. It removes values from **inProfile**, except those, which are included between **inStart** and **inStart** + **inLength** indexes.

Remarks

Value of **inLength** set to Auto means that the operation is performed on all elements starting from **inStart** till the end of **inProfile**.

Errors

List of possible exceptions:

Error type	Description
DomainError	Selected index range exceeds the input profile in CropProfile.

See Also

- BlendImages – Computes weighted sum pixel by pixel.
- LerpImages – Interpolates two images linearly pixel by pixel.

ResizeProfile

Header: AVL.h
Namespace: avl
Module: FoundationPro

Changes the size of a profile, preserving its shape.

Syntax

```
void avl::ResizeProfile
(
 const avl::Profile& inProfile,
 const int inNewSize,
 avl::ResizeProfileInterpolation::Type inInterpolation,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Range	Default	Description
→	inProfile	const Profile&			Input profile
→	inNewSize	const int	1 - ∞	1	
→	inInterpolation	ResizeProfileInterpolation::Type			
←	outProfile	Profile&			Output profile

RotateProfile

Header: AVL.h
Namespace: avl
Module: FoundationPro

Moves the last element ahead of the first one (inShift times).

Syntax

```
void avl::RotateProfile
(
 const avl::Profile& inProfile,
 const int inShift,
 avl::Profile& outProfile
)
```

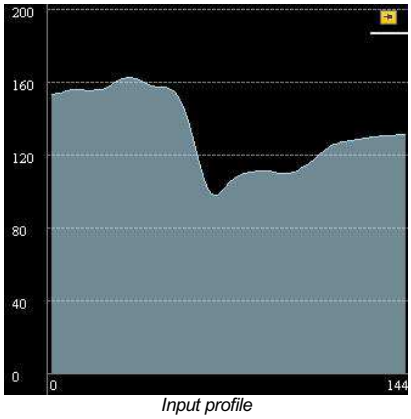
Parameters

	Name	Type	Default	Description
→	inProfile	const Profile&		Input profile
→	inShift	const int	0	
←	outProfile	Profile&		Output profile

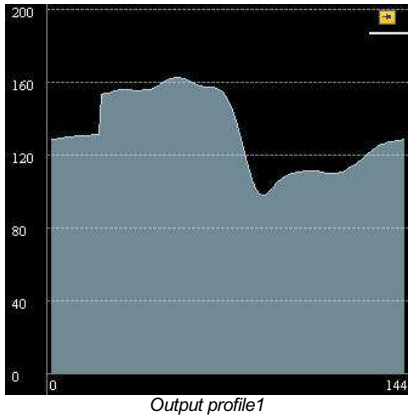
Description

Operation computes the profile of input profile elements shifted cyclically.

Examples

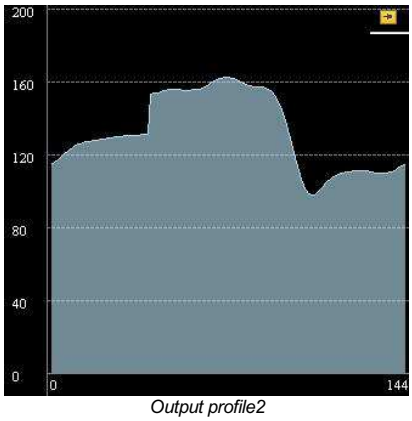


Input profile

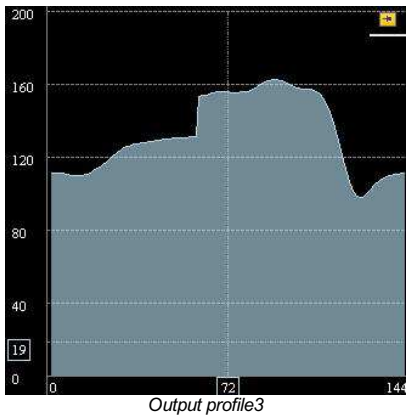


Output profile1

RotateProfile** performed on input profile with **inShift** = 20 produced **outProfile** = **Output profile1



Output profile2



Output profile3

RotateProfile** performed on input profile with **inShift** = 40 produced **outProfile** = **Output profile2** and operation performed on the same input profile with **inShift** = 60 produced **outProfile** = **Output profile3

ShrinkProfileNTimes

Header: AVL.h
Namespace: avl
Module: FoundationPro

Reduces the length of a profile N-times by averaging each N consecutive elements.

Syntax

```
void avl::ShrinkProfileNTimes
(
 const avl::Profile& inProfile,
 const int inN,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Range	Default	Description
→	inProfile	const Profile&			Input profile
→	inN	const int	1 - ∞	2	The scaling coefficient
←	outProfile	Profile&			Output profile

UncropProfile

Header: AVL.h
Namespace: avl
Module: FoundationPro

Extends the profile by adding zeros at the beginning and at the end.

Applications: Undoes CropProfile.

Syntax

```
void avl::UncropProfile
(
 const avl::Profile& inProfile,
 int inStart,
 int inLength,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Range	Default	Description
→	inProfile	const Profile&			Input profile
→	inStart	int	0 - ∞		Same as in CropProfile
→	inLength	int	0 - ∞		Length of the output profile
←	outProfile	Profile&			Output profile

79. Image Pixel Statistics

Table of content:

DaPImageMedian

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes the average direction within the region of interest.

Syntax

```
void avl::DaPImageMedian
(
 const avl::Image& inDaPImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::AngleRange::Type inAngleRange,
 float& outMedianAngle
)
```

Parameters

	Name	Type	Default	Description
→	inDaPImage	const Image&		A result of GradientDirAndPresenceImage filter
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inAngleRange	AngleRange::Type	_0_360	
←	outMedianAngle	float&		

Requirements

For input **inDaPImage** only pixel formats are supported: 1□uint8.

Read more about pixel formats in Image documentation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported inDaPImage pixel format in DaPImageMedian. Supported formats: 1xUInt8.

ImageAverage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the average of the image pixel values.

Syntax

```
void avl::ImageAverage
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::Pixel& outAverageColor,
 atl::Optional<float&> outAverageValue = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
←	outAverageColor	Pixel&		Average of each channel
←	outAverageValue	Optional<float&>	NIL	Average of the entire image

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAverageValue**.

Read more about Optional Outputs.

Description

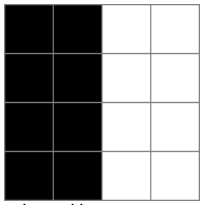
The function calculates average of pixel values in the **inImage**.

Average values of pixels calculated for each channel are returned in **outAverageColor**, while calculated for the region of interest - in **outAverageValue**. If region of interest isn't specified, average is calculated for the entire image.

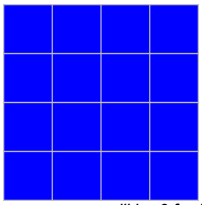
Hints

- Connect **inImage** with the output of your image acquisition filter.

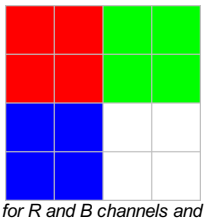
Examples



In this single-channel image, average will be 127.5 for the channel and for the entire image (if the region of interested isn't specified).



In this three-channel RGB image, average will be 0 for R and G channels and 255 for the B channel. If we don't specify the region of interest, average of the image will be 85.0 (because arithmetic mean of 0, 0, and 255 is 85.0).



In this three-channel RGB image, average will be 63.75 for R and B channels and 127.5 for the G channel. If we don't specify the region of interest, average of the image will be 85.0 (because arithmetic mean of 63.75, 63.75 and 127.5 is 85.0).

Remarks

Minimal image size requirement

The input image shouldn't be empty.

Minimal region of interest size requirement

The input region of interest shouldn't be empty.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty image on input in ImageAverage.
DomainError	Empty region of interest on input in ImageAverage.
DomainError	Region exceeds an input image in ImageAverage.

See Also

- ImageAverageHSx – Computes the average of the HSV, HSL or HSI image pixel values.
- ImageMaximum – Finds the location and the value of the brightest pixel.
- ImageMinimum – Finds the location and the value of the darkest pixel.
- ImageSum – Computes the sum of the image pixel values.

ImageAverageHSx

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the average of the HSV, HSL or HSI image pixel values.

Syntax

```
void avl::ImageAverageHSx
(
 const avl::Image& inRgbImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::HSxColorModel::Type inColorModel,
 atl::Optional<int> inMinSaturation,
 int& outHAverage,
 int& outSAverage,
 int& outBAverage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inRgbImage	const Image&			
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inColorModel	HSxColorModel::Type			Selected color model
→	inMinSaturation	Optional<int>	0 - 255	0	
←	outHAverage	int&			Output hue average
←	outSAverage	int&			Output saturation average
←	outBAverage	int&			Output brightness average

Requirements

For input **inRgbImage** only pixel formats are supported: 3□uint8.

Read more about pixel formats in Image documentation.

Description

The filter finds average values of the HSV, HSL or HSI color space image from the RGB image **inRgbImage** within specific region **inRoi**.

The calculations take into account only these pixels which have saturation a greater than **inMinSaturation**.

Filter returns average values of a hue, saturation and brightness.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in ImageAverageHSx.
DomainError	Not supported inRgbImage pixel format in ImageAverageHSx. Supported formats: 3xUInt8.

ImageMaximum

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Finds the location and the value of the brightest pixel.

Syntax

```
void avl::ImageMaximum
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<avl::Location&> outMaximumLocation,
 atl::Optional<float&> outMaximumValue = atl::NIL
)
```

Parameters

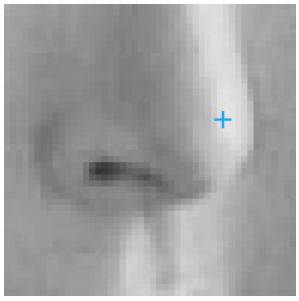
	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
←	outMaximumLocation	Optional<Location&>		
←	outMaximumValue	Optional<float&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outMaximumLocation**, **outMaximumValue**.

Read more about Optional Outputs.

Examples



ImageMaximum *used to detect the brightest point.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty region on input in ImageMaximum.
DomainError	Region exceeds an input image in ImageMaximum.

ImageMinimum

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Finds the location and the value of the darkest pixel.

Syntax

```
void avl::ImageMinimum
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<avl::Location&> outMinimumLocation,
 atl::Optional<float&> outMinimumValue = atl::NIL
)
```

Parameters

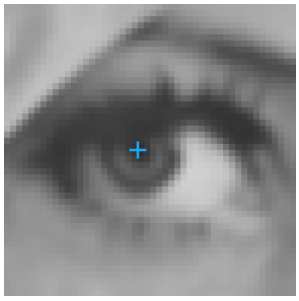
	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
←	outMinimumLocation	Optional<Location&>		
←	outMinimumValue	Optional<float&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outMinimumLocation**, **outMinimumValue**.

Read more about Optional Outputs.

Examples



ImageMinimum *used to detect the darkest point.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty region on input in ImageMinimum.
DomainError	Region exceeds an input image in ImageMinimum.

ImageStatistics

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes various statistics of the image pixel values.

Syntax

```
void avl::ImageStatistics
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Optional<avl::Location&> outMinimumLocation = atl::NIL,
 atl::Optional<float&> outMinimumValue = atl::NIL,
 atl::Optional<avl::Location&> outMaximumLocation = atl::NIL,
 atl::Optional<float&> outMaximumValue = atl::NIL,
 atl::Optional<avl::Pixel&> outAverageColor = atl::NIL,
 atl::Optional<float&> outAverageValue = atl::NIL,
 atl::Optional<avl::Pixel&> outSumColor = atl::NIL,
 atl::Optional<float&> outSumValue = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
←	outMinimumLocation	Optional<Location&>	NIL	
←	outMinimumValue	Optional<float&>	NIL	
←	outMaximumLocation	Optional<Location&>	NIL	
←	outMaximumValue	Optional<float&>	NIL	
←	outAverageColor	Optional<Pixel&>	NIL	
←	outAverageValue	Optional<float&>	NIL	
←	outSumColor	Optional<Pixel&>	NIL	
←	outSumValue	Optional<float&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outMinimumLocation**, **outMinimumValue**, **outMaximumLocation**, **outMaximumValue**, **outAverageColor**, **outAverageValue**, **outSumColor**, **outSumValue**.

Read more about Optional Outputs.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in ImageStatistics.

ImageSum

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the sum of the image pixel values.

Syntax

```
void avl::ImageSum
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::Pixel& outSumColor,
 atl::Optional<float&> outSumValue = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
←	outSumColor	Pixel&		
←	outSumValue	Optional<float&>	NIL	

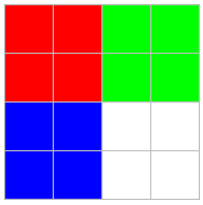
Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outSumValue**.

Read more about Optional Outputs.

Examples

Assume, that the subject of the analysis is 4x4 image made of pixels in different colors - red, green, blue and white:



Red (255, 0, 0)

Green (0, 255, 0)

Blue (0, 0, 255)

White (255, 255, 255)

On the output **outSumColor** the sum of pixel values for each channel of an image will be computed.
For above example it will be R, G and B components, but it is also possible to use another color spaces.
Sum of color for R channel will be: $4 * 255 + 4 * 255 = 2040$ - sum of four red pixels and four red-components from white pixels.
Similarly, sum of colors for G channel and B channel will also be: $4 * 255 + 4 * 255 = 2040$
Finally, the result for above example will be 1x4 dimensional vector [2040, 2040, 2040, 0]. Notice, that zero is present in result vector, because RGB color space has only three components.

On the output **outSumValue** the average value of pixel of the image will be calculated according to the following formula:

Sum of elements of **outSumColor** / number of components in color space

Therefore, he result for above example will be $\frac{2040+2040+2040}{3} = \frac{6120}{3} = 2040$

Remarks

When only a part of an image should be processed use **inRoi** input to specify region of interest.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in ImageSum.

80. Barcodes

Table of content:

DecodeBarcode

Header: AVL.h
Namespace: avl
Module: Barcodes

Translates an array of bar widths to sequence of digits or text in accordance to the selected barcode standard.

Applications: Decoding of barcodes whose bars have been measured in a non-standard way.

Syntax

```
void avl::DecodeBarcode
(
 const atl::Array<float>& inBarWidths,
 atl::Optional<avl::BarcodeFormat::Type> inBarcodeFormat,
 atl::Conditional<atl::String>& outDecodedText,
 atl::Conditional<avl::BarcodeFormat::Type>& outBarcodeFormat
)
```

Parameters

	Name	Type	Default	Description
→	inBarWidths	const Array<float>&		Widths of the barcode bars (starting with width of the black bar)
→	inBarcodeFormat	Optional<BarcodeFormat::Type>	EAN13	Format of the barcode
←	outDecodedText	Conditional<String>&		Decoded barcode text or nothing if decoding failed
←	outBarcodeFormat	Conditional<BarcodeFormat::Type>&		Decoded barcode format or nothing if decoding failed

Description

This filter is especially useful when a barcode is not provided in a standard black and white printed form.

Use this filter to decode a barcode printed or engraved on circular or distorted surfaces. Also the filter can be used to decode a barcode when one of colors is transparent due to printing on glass or other transparent surface.

It can be useful in decoding damaged codes in which barcode position detection is very complex.

Use the ScanMultipleStripes filter to get the bar widths.

Hints

- Pass an array of computed widths of consecutive black and white bars to the **inBarWidths** input.
- Select **inBarcodeFormat** according to the type of codes you want to read. If you choose the wrong format, the codes will not be recognized. The Auto value causes decoding of UPC-A codes as EAN-13 codes.

Remarks

Minimal bar width requirement

To provide precise detection of the barcode width of the thinnest bar should be at least **1.5** pixels.

Depending on the barcode format guard or start/end code patterns must be readable.

Using a relative coordinate systems

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

See Also

- ReadSingleBarcode – Detects and recognizes a single barcode on the input image.
- ReadMultipleBarcodes – Detects and recognizes multiple barcodes on the input image.
- RecognizeBarcode – Extracts information from a barcode located on the input image at a given position.

ReadMultipleBarcodes

Header: AVL.h
Namespace: avl
Module: Barcodes

Detects and recognizes multiple barcodes on the input image.

Applications: To be used as an easy all-in-one solution for typical barcode reading applications.

Syntax

```
void avl::ReadMultipleBarcodes
(
 const avl::Image& inImage,
 atl::Optional<const avl::ShapeRegion&> inRoi,
 atl::Optional<const avl::CoordinateSystem2D&> inRoiAlignment,
 const avl::BarcodeParams& inBarcodeParams,
 const avl::BarcodeDetectionParams& inDetectionParams,
 int inMaxBarcodeCount,
 atl::Array<avl::Barcode>& outBarcodes,
 atl::Optional<atl::Array<avl::Rectangle2D>&> outBarcodeCandidates = atl::NIL,
 atl::Optional<avl::ShapeRegion&> outAlignedRoi = atl::NIL,
 avl::Image& diagGradientImage,
 atl::Array<atl::Array<avl::Segment2D>>& diagScheduledScanSegments
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const ShapeRegion&>		NIL	Region of interest
→	inRoiAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the region of interest to the position of the inspected object
→	inBarcodeParams	const BarcodeParams&			Specification of barcodes that can be detected
→	inDetectionParams	const BarcodeDetectionParams&			Parameters of the barcode detection algorithm
→	inMaxBarcodeCount	int	1 - ∞	2	Maximum number of barcodes in one image
←	outBarcodes	Array<Barcode>&			List of barcodes that have been correctly detected and decoded
←	outBarcodeCandidates	Optional<Array<Rectangle2D>&>		NIL	Places with high gradient values that are further investigated
←	outAlignedRoi	Optional<ShapeRegion&>		NIL	Input ROI after transformation (in the image coordinates)
🔍	diagGradientImage	Image&			Image of gradient directions
🔍	diagScheduledScanSegments	Array<Array<Segment2D>>&			Scheduled scan segments

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outBarcodeCandidates**, **outAlignedRoi**.

Read more about Optional Outputs.

Description

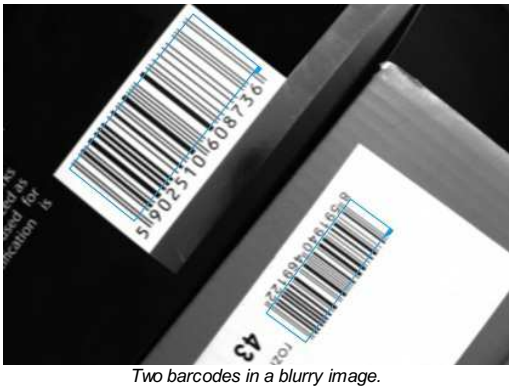
This filter detects barcodes present in the input image and reads their content.

Hints

- Connect **inImage** with the output of your image acquisition filter.
- Select **inBarcodeParams.Format** according to the type of codes you want to read. If you choose the wrong format, the codes will not be recognized. Setting its value to Auto can increase the computation time considerably. Furthermore, the Auto value causes detection of UPC-A codes as EAN-13 codes.
- If the image resolution is high, set **inBarcodeParams.MaxModuleSize** accordingly or resize/downsample the input image.

Examples

Two barcodes in a good quality image.



Two barcodes in a blurry image.

Remarks

Minimal bar width requirement

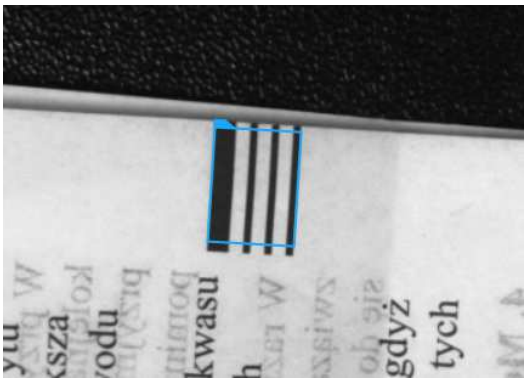
To provide precise detection of the barcode width of the thinnest bar should be at least **1.5** pixels.

Depending on the barcode format guard or start/end code patterns must be readable.

Pharmacode usage

The pharmacode barcode type can be read correctly in both directions. To get results from both directions use a **Pharmacode** and **PharmacodeInversed** barcode types.

Before decoding a Pharmacode the code orientation angle is normalized to a range from -45° to 135° what makes the code decoding more stable



*Results of reading using a different Pharmacode directions: **Pharmacode** = 23 and **PharmacodeInversed** = 16.*

Using a relative coordinate systems

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- ReadSingleBarcode – Detects and recognizes a single barcode on the input image.
- DecodeBarcode – Translates an array of bar widths to sequence of digits or text in accordance to the selected barcode standard.
- RecognizeBarcode – Extracts information from a barcode located on the input image at a given position.

ReadSingleBarcode

Header: AVL.h
Namespace: avl
Module: Barcodes

Detects and recognizes a single barcode on the input image.

Applications: To be used as an easy all-in-one solution for typical barcode reading applications.

Syntax

```
void avl::ReadSingleBarcode
(
 const avl::Image& inImage,
 atl::Optional<const avl::ShapeRegion&> inRoi,
 atl::Optional<const avl::CoordinateSystem2D&> inRoiAlignment,
 const avl::BarcodeParams& inBarcodeParams,
 const avl::BarcodeDetectionParams& inDetectionParams,
 atl::Conditional<avl::Barcode>& outBarcode,
 atl::Optional<atl::Array<avl::Rectangle2D>&> outBarcodeCandidates = atl::NIL,
 atl::Optional<avl::ShapeRegion&> outAlignedRoi = atl::NIL,
 avl::Image& diagGradientImage,
 atl::Array<atl::Array<avl::Segment2D>>& diagScheduledScanSegments
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const ShapeRegion&>	NIL	Region of interest
→	inRoiAlignment	Optional<const CoordinateSystem2D&>	NIL	Adjusts the region of interest to the position of the inspected object
→	inBarcodeParams	const BarcodeParams&		Specification of barcode that can be detected
→	inDetectionParams	const BarcodeDetectionParams&		Parameters of the barcode detection algorithm
←	outBarcode	Conditional<Barcode>&		A barcode that has been correctly detected and decoded
←	outBarcodeCandidates	Optional<Array<Rectangle2D>&>	NIL	Places with high gradient values that are further investigated
←	outAlignedRoi	Optional<ShapeRegion&>	NIL	Input ROI after transformation (in the image coordinates)
🔍	diagGradientImage	Image&		Image of gradient directions
🔍	diagScheduledScanSegments	Array<Array<Segment2D>>&		Scheduled scan segments

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outBarcodeCandidates**, **outAlignedRoi**.

Read more about Optional Outputs.

Description

This filter detects a barcode present in the input image and reads its content.

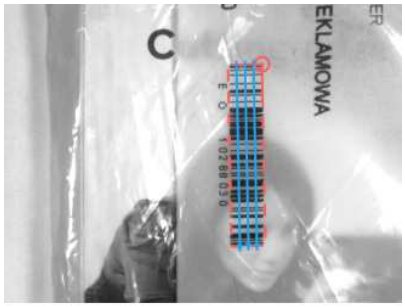
Hints

- Connect **inImage** with the output of your image acquisition filter.
- Select **inBarcodeParams.Format** according to the type of codes you want to read. If you choose the wrong format, the codes will not be recognized. Setting its value to Auto can increase the computation time considerably. Furthermore, the Auto value causes detection of UPC-A codes as EAN-13 codes.
- If the image resolution is high, set **inBarcodeParams.MaxModuleSize** accordingly or resize/downsample the input image.

Examples

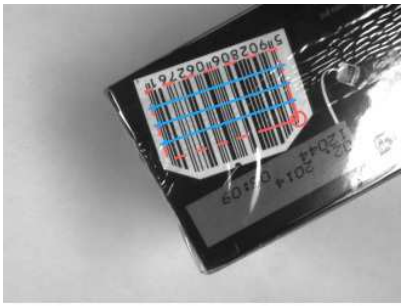
5901086000562

Rotated barcode.



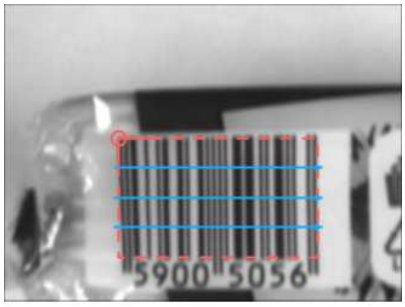
EO10288030

Low quality barcode printed on plastic foil.



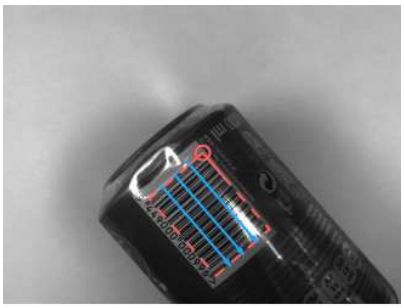
5902806062761

Barcode on package wrapped in plastic foil.



59005056

Barcode in a blurry image.

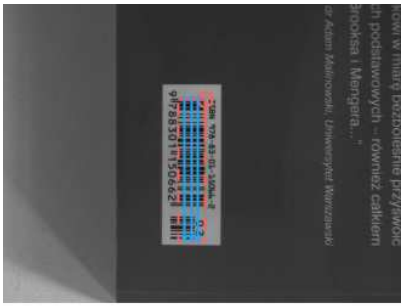


5449000000996

Barcode on standard 330ml can.

4009900382250

Barcode on reflective and wrapped surface.



9788301150662 02

EAN-13 with add-on 2 used to indicate a book edition.

9780321334879 54999

EAN-13 with add-on 5 to give a suggestion for the price.

Remarks

Minimal bar width requirement

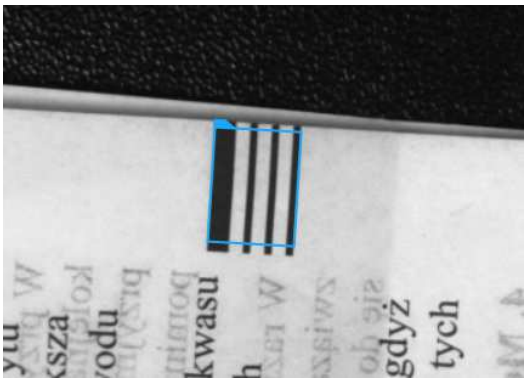
To provide precise detection of the barcode width of the thinnest bar should be at least **1.5** pixels.

Depending on the barcode format guard or start/end code patterns must be readable.

Pharmacode usage

The pharmacode barcode type can be read correctly in both directions. To get results from both directions use a **Pharmacode** and **PharmacodeInversed** barcode types.

Before decoding a Pharmacode the code orientation angle is normalized to a range from -45° to 135° what makes the code decoding more stable



Results of reading using a different Pharmacode directions: ***Pharmacode*** *= 23 and* ***PharmacodeInversed*** *= 16.*

Using a relative coordinate systems

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- ReadMultipleBarcodes – Detects and recognizes multiple barcodes on the input image.
- DecodeBarcode – Translates an array of bar widths to sequence of digits or text in accordance to the selected barcode standard.
- RecognizeBarcode – Extracts information from a barcode located on the input image at a given position.

RecognizeBarcode

Header: AVL.h
Namespace: avl
Module: Barcodes

Extracts information from a barcode located on the input image at a given position.

Applications: Most often used after a barcode detection filter.

Syntax

```
void avl::RecognizeBarcode
(
 const avl::Image& inImage,
 const avl::Rectangle2D& inBarcodePosition,
 atl::Optional<const avl::CoordinateSystem2D&> inBarcodePositionAlignment,
 const avl::BarcodeParams& inBarcodeParams,
 const int inScanCount,
 const int inScanWidth,
 const float inMinEdgeStrength,
 atl::Conditional<avl::Barcode>& outBarcode,
 atl::Optional<avl::Rectangle2D&> outAlignedBarcodePosition = atl::NIL,
 atl::Array<avl::Segment2D>& diagScheduledScanSegments
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inBarcodePosition	const Rectangle2D&			Position of the input image in which the barcode is located
→	inBarcodePositionAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the barcode rectangle to the position of the inspected object
→	inBarcodeParams	const BarcodeParams&			Specification of barcode that can be read
→	inScanCount	const int	1 - ∞	5	Number of parallel scans run until first successful read
→	inScanWidth	const int	1 - ∞	5	Width of the single scan
→	inMinEdgeStrength	const float	0.0 - ∞	5.0f	Minimal strength of an extracted edge
←	outBarcode	Conditional<Barcode>&			Decoded barcode or nothing if all of the scans failed
←	outAlignedBarcodePosition	Optional<Rectangle2D&>		NIL	
	diagScheduledScanSegments	Array<Segment2D>&			Scheduled scan segments

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedBarcodePosition**.

Read more about Optional Outputs.

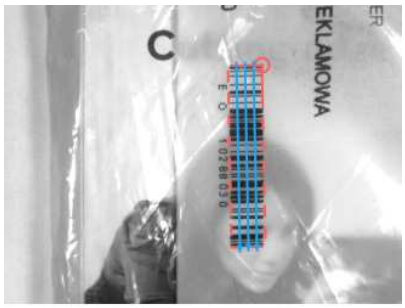
Hints

- Connect **inImage** with the output of your image acquisition filter.
- Select **inBarcodeParams.Format** according to the type of codes you want to read. If you choose the wrong format, the codes will not be recognized. Setting its value to Auto can increase the computation time considerably. Furthermore, the Auto value causes detection of UPC-A codes as EAN-13 codes.

Examples

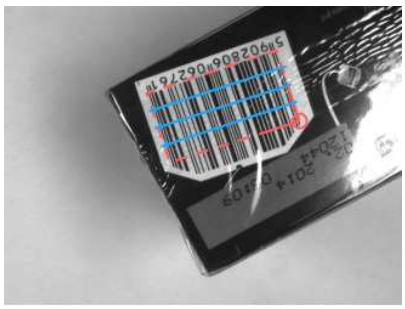
5901086000562

Rotated barcode.



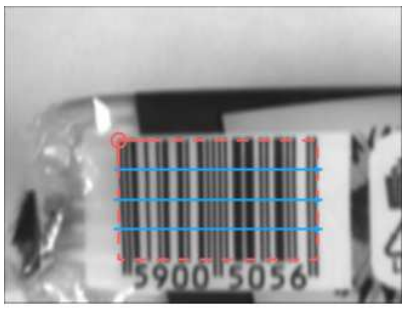
EO10288030

Low quality barcode printed on plastic foil.



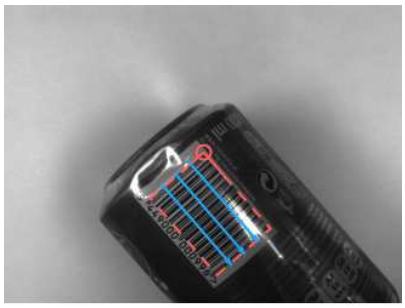
5902806062761

Barcode on package wrapped in plastic foil.



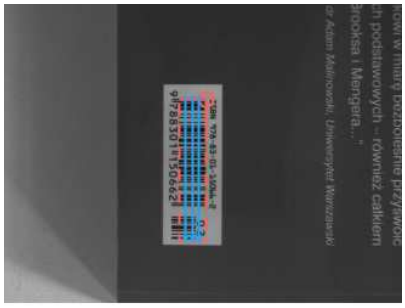
59005056

Barcode on blurry image.



Barcode on standard 330ml can.

Barcode on reflective and wrapped surface.



EAN-13 with add-on 2 used to indicate a book edition.

EAN-13 with add-on 5 to give a suggestion for the price.

Remarks

Minimal bar width requirement

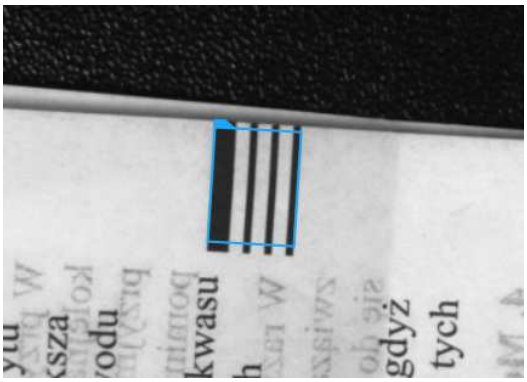
To provide precise detection of the barcode width of the thinnest bar should be at least **1.5** pixels.

Depending on the barcode format guard or start/end code patterns must be readable.

Pharmacode usage

The pharmacode barcode type can be read correctly in both directions. To get results from both directions use a **Pharmacode** and **PharmacodeInversed** barcode types.

Before decoding a Pharmacode the code orientation angle is normalized to a range from -45° to 135° what makes the code decoding more stable



*Results of reading using a different Pharmacode directions: **Pharmacode** = 23 and **PharmacodeInversed** = 16.*

Using a relative coordinate systems

Read more about Local Coordinate Systems in Machine Vision Guide: Local Coordinate Systems.

See Also

- ReadSingleBarcode – Detects and recognizes a single barcode on the input image.
- ReadMultipleBarcodes – Detects and recognizes multiple barcodes on the input image.
- DecodeBarcode – Translates an array of bar widths to sequence of digits or text in accordance to the selected barcode standard.

81. Datacodes

Table of content:

DecodeDataMatrix

Header: AVL.h
Namespace: avl
Module: Datacodes

Translates a matrix of 0 or 1 values to a text in accordance to how DataMatrix codes are encoded.

Applications: Decoding of data matrix codes whose black and white segments have been determined in a non-standard way.

Syntax

```
void avl::DecodeDataMatrix
(
 const avl::Matrix& inMatrixValues,
 bool inAllowMirrored,
 atl::Conditional<atl::String>& outText,
 atl::Conditional<avl::DataMatrixType::Type>& outType
)
```

Parameters

	Name	Type	Default	Description
→	inMatrixValues	const Matrix&		Matrix of binary values
→	inAllowMirrored	bool		Allows reading mirrored/transposed codes
←	outText	Conditional<String>&		Decoded text or nothing if decoding failed
←	outType	Conditional<DataMatrixType::Type>&		DataMatrix code type

Description

Supported code types: ECC 200 and ECC 000-140.

See Also

- ReadMultipleDataMatrixCodes – Detects and recognizes several Data Matrix codes in one image.
- ReadSingleDataMatrixCode – Detects and recognizes one Data Matrix code.
- ReadSingleBarcode – Detects and recognizes a single barcode on the input image.
- ReadMultipleBarcodes – Detects and recognizes multiple barcodes on the input image.
- RecognizeBarcode – Extracts information from a barcode located on the input image at a given position.
- ReadSingleQRCode – Detects and recognizes a single QR code on the input image.

DecodeQRCode

Header: AVL.h
Namespace: avl
Module: Datacodes

Translates a matrix of 0 or 1 values to a text in accordance to how QR codes are encoded.

Applications: Decoding of QR codes whose black and white segments have been determined in a non-standard way.

Syntax

```
void avl::DecodeQRCode
(
 const avl::Matrix& inBinaryMatrix,
 atl::Conditional<atl::String>& outDecodedText
)
```

Parameters

	Name	Type	Default	Description
→	inBinaryMatrix	const Matrix&		Square matrix of zero (black) and nonzero (white) values
←	outDecodedText	Conditional<String>&		Decoded text, if matrix represents valid code

ReadMultipleCodes_IK

Header: AVL.h
Namespace: avl
Module: Datacodes

Detects and recognizes Codes in one image.

Syntax

```
void avl::ReadMultipleCodes_IK
(
 const avl::Image& inImage,
 atl::Optional<const avl::Rectangle2D&> inRoi,
 avl::CodeInverse1D::Type inInverse,
 atl::Optional<int> inTimeout,
 const avl::CodeDetectionParameters& inDetectionParameters,
 const avl::ManyCodeSettings& inManyCodeSettings,
 bool inEnableCode39,
 bool inEnableCode128,
 bool inEnableInterleaved2of5,
 atl::Optional<const avl::DatamatrixParameters&> inEnableDatamatrix,
 bool inEnablePDF417,
 bool inEnableQRCode,
 bool inEnableUPCEAN,
 bool inEnableCode93,
 bool inEnableDotCode,
 bool inEnableMaxiCode,
 bool inEnableAztec,
 bool inEnableMSI,
 atl::Optional<const avl::CodabarParameters&> inEnableCodabar,
 atl::Array<atl::String>& outResult,
 atl::Array<avl::Path>& outOutline,
 atl::Array<avl::CodeFormat::Type>& outCodeFormat
)
```

Parameters

	Name	Type	Default	Description
➡	inImage	const Image&		Input image
➡	inRoi	Optional<const Rectangle2D&>	NIL	Region of interest
➡	inInverse	CodeInverse1D::Type		
➡	inTimeout	Optional<int>	NIL	Processing timeout
➡	inDetectionParameters	const CodeDetectionParameters&		
➡	inManyCodeSettings	const ManyCodeSettings&		
➡	inEnableCode39	bool	True	
➡	inEnableCode128	bool		
➡	inEnableInterleaved2of5	bool		
➡	inEnableDatamatrix	Optional<const DatamatrixParameters&>	DatamatrixParameters ( )	
➡	inEnablePDF417	bool		
➡	inEnableQRCode	bool		
➡	inEnableUPCEAN	bool		
➡	inEnableCode93	bool		
➡	inEnableDotCode	bool		
➡	inEnableMaxiCode	bool		
➡	inEnableAztec	bool		
➡	inEnableMSI	bool		
➡	inEnableCodabar	Optional<const CodabarParameters&>	NIL	
⬅	outResult	Array<String>&		
⬅	outOutline	Array<Path>&		
⬅	outCodeFormat	Array<CodeFormat::Type>&		

Errors

List of possible exceptions:

Error type	Description
DomainError	ReadMultipleCodes_IK is supported only in x64 environment.

ReadMultipleDataMatrixCodes

Header: AVL.h
Namespace: avl
Module: Datacodes

Detects and recognizes several Data Matrix codes in one image.

Syntax

```
void avl::ReadMultipleDataMatrixCodes
(
 const avl::Image& inImage,
 atl::Optional<const avl::Rectangle2D&> inRoi,
 atl::Optional<const avl::CoordinateSystem2D&> inRoiAlignment,
 const avl::DataMatrixCodeParams& inCodeParams,
 const avl::DataMatrixDetectionParams& inDetectionParams,
 int inMaxCodeCount,
 bool inAllowMultipleScales,
 atl::Array<avl::DataCode>& outDataMatrixCodes,
 atl::Array<avl::Path>& outCandidates,
 atl::Optional<avl::Rectangle2D&> outAlignedRoi = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Rectangle2D&>		NIL	Region of interest
→	inRoiAlignment	Optional<const CoordinateSystem2D&>		NIL	Coordinate system for the region of interest
→	inCodeParams	const DataMatrixCodeParams&			Specification of codes that can be detected
→	inDetectionParams	const DataMatrixDetectionParams&			Parameters of the code detection algorithm
→	inMaxCodeCount	int	0 - ∞	2	Maximum number of codes in one image
→	inAllowMultipleScales	bool		False	Specifies whether to continue search for codes at a higher resolution even when some codes have already been found at a lower resolution
←	outDataMatrixCodes	Array<DataCode>&			List of codes that have been correctly detected and decoded
←	outCandidates	Array<Path>&			Diagnostic information about detection results
←	outAlignedRoi	Optional<Rectangle2D&>		NIL	Input ROI after transformation (in the image coordinates)

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedRoi**.

Read more about Optional Outputs.

Description

This filter detects and recognizes data matrix codes on an image.

Supported code types: ECC 200 and ECC 000-140.

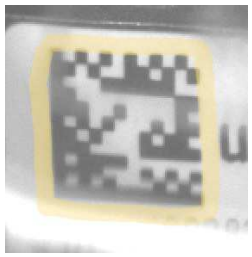
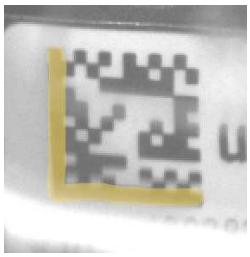
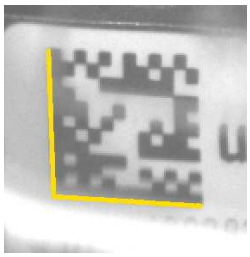
Groups of parameters

For optimal efficiency the tool may require careful setting of parameters. Most of the parameters are divided into two groups:

1. **inCodeParams** – parameters concerning the code; these can be set precisely according to what codes we expect to read
2. **inDetectionParams** – parameters concerning the algorithm; these may require some experimentation as different approaches may work best in different scenarios

Detection methods

The most important setting is the selection of the code detection algorithm, **inDetectionParams.DetectionMethod**. The tool offers three options:

	Quiet Zone	Finder Pattern	Finder Edges
Where it looks?			
How it works?	Looks for the blank space surrounding the code. In theory it should be empty for a width of at least one module.	Looks for the blob of the L-shaped pattern. In theory it should be continuous and not connected with anything outside of the code.	Looks for the edges of the L-shaped pattern. It uses a robust Line Segment Detector internally.
When to use it?	This method is good even for partially broken Finder Pattern, but it will fail if there is any contamination in the Quiet Zone.	This method may deal with some contamination in the Quiet Zone, but it will fail if the Finder Pattern itself is not good.	This method works well even if Quiet Zone is contaminated and Finder Pattern is not perfect, but may be confused by additional lines appearing near the code.

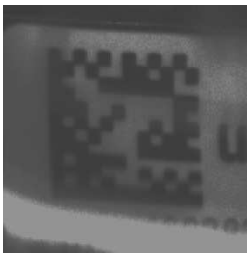
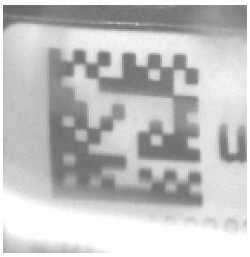
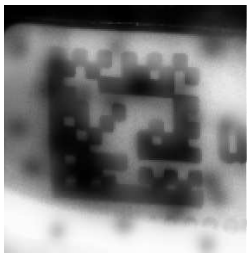
Pyramid

The tool tries to detect codes at different scale levels. It investigates images at different resolutions, starting with the lowest resolution. If it finds a sufficient number of codes at one level, it may skip looking further, thus achieving shorter execution time. If there are no codes in the input image, it will always scan all the levels, resulting in the highest execution time.

NOTE: If you experience instable execution times, for example switching between 5 ms and 20 ms, it may mean that for some images the tool finds a code at a higher level, while for some others it has to process a higher resolution too.

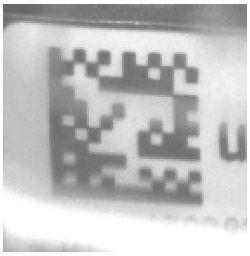
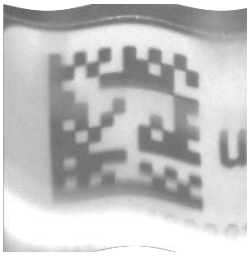
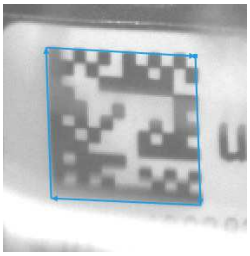
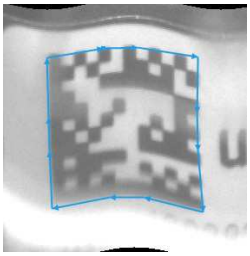
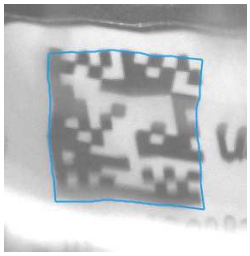
Finder Tradeoff ("FinderEdges" only)

When using the "FinderEdges" method, you should be aware that there is a tradeoff between the ability of the tool to detect rugged (uneven, noised or dot printed) edges and the ability to detect low contrast edges (or ones neighboring with a very narrow Quite Zone). The default setting is "Balanced". However, if you expect the edges of your code to be relatively straight and clean, but less strong, you may want to change it to "Sensitive". On the other hand, if the edges of the code are highly jagged or the surrounding is noisy, it may be better to choose "Robust". You may also set it to "Auto" and it will test several possibilities at the cost of much higher execution time.

Use "Sensitive"	Use "Balanced"	Use "Robust"
		

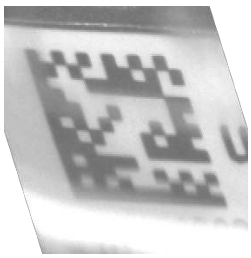
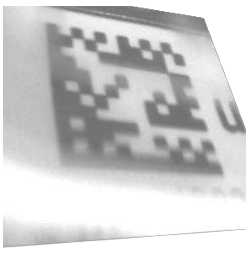
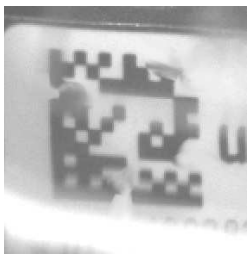
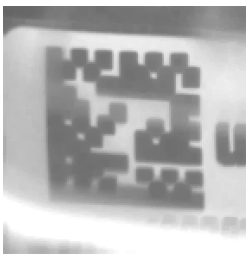
Distortions

The tool can deal with various types of code deformations. There are three possible settings, as demonstrated below. Please be aware that when we make the algorithm more robust to distortions, at the same time we make it less robust to broken code patterns. For this reason, we should stay with lower settings whenever possible.

Use "Low"	Use "Medium"	Use "High"
		
		
Each side is represented with a single line segment.	Each side is represented with a 3-segment line.	Each side is exactly traced with a polyline.

Damaged codes and other difficulties

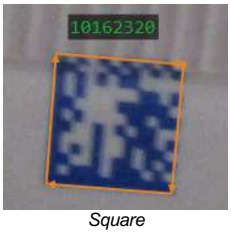
By using appropriate **inCodeParams** settings you can also find codes that are partially damaged. See the below table for some of the most common cases.

Use MaxSlant=30	Use AllowPerspective	Use AllowBrokenFinder and AllowBrokenTiming	Use AllowOversizedModules
			

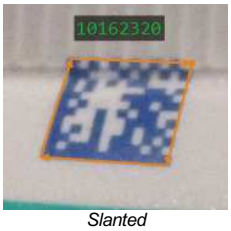
Hints

- Specify the range of possible codes in an image by setting the **inCodeParams** input. The more narrow is the specification, the faster the filter works.
  - Set **inCodeParams.Polarity** to specify whether dark-on-bright or bright-on-dark codes are to be read.
  - Specify the range of possible code sizes with **inCodeParams.MinRowCount**, **inCodeParams.MaxRowCount**, **inCodeParams.MinColumnCount** and **inCodeParams.MinColumnCount**.
  - Specify the range of possible module sizes (in pixels) with **inCodeParams.MinModuleSize** and **inCodeParams.MaxModuleSize**.
  - Set **inCodeParams.ExpectedGapSize** according to the following rules:
    - Zero – no gaps at all (modules are fully filled); use it also when there is other print very near to the code.
    - Small – gaps up to 25% of the module size; this is the default setting.
    - Medium – gaps up to 50% of the module size; it may require a bigger Quiet Zone.
    - Large – gaps up to 75%; comes with no guarantee.
  - Set **inCodeParams.MaxRectangleRatio** to 1 if you expect to work with square codes only. The value specifies maximal ratio between the length of the longer side to the length of the shorter side.
  - Pay attention to codes that are bent or unevenly printed.
    - For bent codes (e.g. printed on a bottle) use **inCodeParams.AllowDistortion** = Medium.
    - For unevenly printed codes or ones that are printed on wrinkled material use **inCodeParams.AllowDistortion** = High.
  - For potentially damaged codes, consider using **inCodeParams.AllowBrokenFinder** and **inCodeParams.AllowBrokenTiming**. These settings make the tool more robust against missing pieces of Finder Pattern and Timing Pattern respectively.
- If some codes are not detected, try modifying **inDetectionParams**:
  - In the first place make sure you set **inCodeParams** correctly.
  - Then try different detection methods. "QuietZone" and "FinderPattern" are faster because they work with blobs. "FinderEdges" however is usually the most reliable.
  - You may also try using more reliable but slower strategies for building the image pyramid (**inDetectionParams.PyramidStrategy**) and for fitting of the code's outline (**inDetectionParams.OutlineStrategy**). "Precise" should be ok for most cases, but "Strict" double-checks some options while "Extended" completely ignores performance for the sake of making sure we do not miss any code.
  - Increase **inDetectionParams.ContrastThreshold** if the noise is high. Decrease it, if the noise is low and the difference between the bright and the dark modules is low.

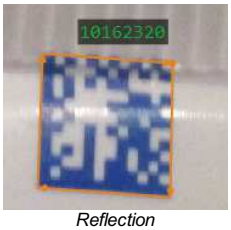
Examples



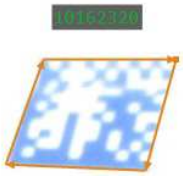
Square



Slanted



Reflection



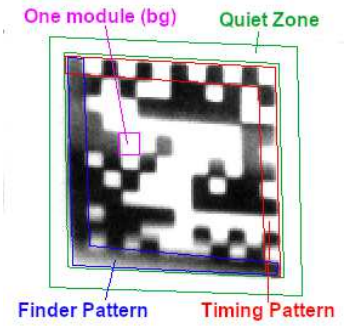
Overexposure

Remarks

For maximum reliability of reading Data Matrix codes we recommend making sure that the image fulfills the following standard requirements:

- Module size should be at least 4 pixels, while 5 pixels is recommended for the highest reliability. It is possible to read codes with modules as low as 1.25 pixels, but this comes with no guarantee and is highly dependent on the image and printing quality.
- Gaps between dot-printed marks should be not bigger than 50% of the module size, but the less the better.
- There should be an empty space around the code ("quiet zone"). It should have the width at least equal to the module size. Contamination, additional print or shadows in this area is the most frequent source of reduced reliability.
- Finder Pattern slant (angular deviation from 90 degrees) should not be greater than 30 degrees.
- Data Matrix codes are not designed for reading when there is strong perspective projection.

For explanation of the used terminology, please refer to the below picture:



See Also

- ReadSingleDataMatrixCode – Detects and recognizes one Data Matrix code.

ReadMultiplePDF417Codes

Header: AVL.h
Namespace: avl
Module: Datacodes

Detects and recognizes several PDF417 codes.

Syntax

```
void avl::ReadMultiplePDF417Codes
(
 const avl::Image& inImage,
 atl::Optional<const avl::Rectangle2D&> inRoi,
 atl::Optional<const avl::CoordinateSystem2D&> inRoiAlignment,
 const avl::PDF417CodeParams& inCodeParams,
 const avl::PDF417DetectionParams& inDetectionParams,
 bool inAllowMultipleScales,
 atl::Array<avl::PDF417Code>& outPDF417Codes,
 atl::Optional<avl::Rectangle2D&> outAlignedRoi = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Rectangle2D&>	NIL	Region of interest
→	inRoiAlignment	Optional<const CoordinateSystem2D&>	NIL	Coordinate system for the region of interest
→	inCodeParams	const PDF417CodeParams&		Specification of codes that can be detected
→	inDetectionParams	const PDF417DetectionParams&		Specification of the way the codes are being detected
→	inAllowMultipleScales	bool	False	Specifies whether codes of different module sizes are expected
←	outPDF417Codes	Array<PDF417Code>&		Found PDF417 codes
←	outAlignedRoi	Optional<Rectangle2D&>	NIL	Input ROI after transformation (in the image coordinates)

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedRoi**.

Read more about Optional Outputs.

Hints

If codes are not detected, try to use ResizeImage on them before processing with ReadMultiplePDF417Codes.

Remarks

In *inCodeParams* the parameter *MinModuleSize* should be more or equal to 3. For values less than 3 it is not guaranteed, that ReadMultiplePDF417Codes will process codes properly.

ReadMultipleQRCodes

Header: AVL.h
Namespace: avl
Module: Datacodes

Detects and recognizes all QR codes on the input image.

Syntax

```
void avl::ReadMultipleQRCodes
(
 const avl::Image& inImage,
 atl::Optional<const avl::Rectangle2D&> inRoi,
 atl::Optional<const avl::CoordinateSystem2D&> inRoiAlignment,
 int inMaxCodeCount,
 float inMinModuleSize,
 float inMaxModuleSize,
 atl::Optional<float> inContrastThreshold,
 atl::Optional<int> inPatternQuality,
 bool inAllowRotation,
 avl::Polarity::Type inPolarity,
 atl::Optional<float> inMinLineMagnitude,
 atl::Array<avl::QRCode>& outQRCodes,
 atl::Optional<avl::Rectangle2D&> outAlignedRoi = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Rectangle2D&>		NIL	Range of pixels to be processed
→	inRoiAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the region of interest to the position of the inspected object
→	inMaxCodeCount	int	1 - 255	2	Maximum number of codes in one image
→	inMinModuleSize	float	1.5 - 100.0	4.0f	Lower estimated size of a code unit in pixels
→	inMaxModuleSize	float	1.5 - 100.0	40.0f	Upper estimated size of a code unit in pixels
→	inContrastThreshold	Optional<float>	1.0 - 255.0	NIL	Guaranteed gray level difference between dark and bright modules
→	inPatternQuality	Optional<int>	1 - 3	NIL	Quality of the code from 1 (extremely deformed) to 3 (perfect)
→	inAllowRotation	bool		True	Allows codes rotated in relation to the axes of the input image
→	inPolarity	Polarity::Type		Any	Specifies whether code is darker or brighter than the background
→	inMinLineMagnitude	Optional<float>	1.0 - 255.0	NIL	Strength of an edge within the code; default depends on parameters
←	outQRCodes	Array<QRCode>&			
←	outAlignedRoi	Optional<Rectangle2D&>		NIL	Input ROI after transformation (in the image coordinates)

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedRoi**.

Read more about Optional Outputs.

Description

The filter locates and decodes QR code on the image (**inImage**) within given region (**inRoi**). The encoded text length must be greater than 2.

inMinModuleSize is the expected lower bound of one module (smallest unit of the code). If there are multiple codes, size of the smallest one should be provided.

inMinModuleSize is the expected upper bound of one module (smallest unit of the code). If there are multiple codes, size of the smallest one should be provided.

Parameter **inMaxCodeCount** determines how many codes are to be found within the image. Thus the algorithm continues searching until it reaches the desired number of codes or has nowhere to search. Too high value can significantly extend the filter execution time.

Parameter **inContrastThreshold** describes contrast of the code - namely, it denotes the brightness difference between light and dark modules of the code. If not given, the algorithm uses normalization of the image to enhance the code and computes this value automatically.

inPatternQuality is an integer from the range <1,3> describing how much the code is blurred or the positional patterns are deformed. For instance, value 3 corresponds to a perfect quality code, while value 1 to an extremely blurred one. For standard cases it is recommended to use quality of 2 or to leave it default.

inAllowRotation Allows codes rotated in relation to the axes of the input image.

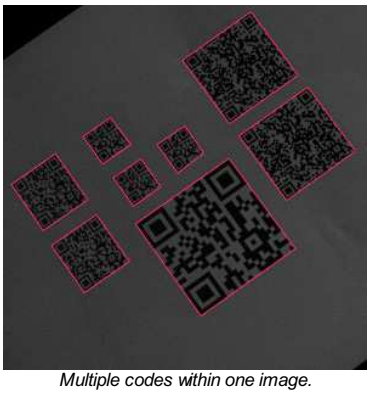
inPolarity Setup polarity of QR code can improve performance of ReadingQRCodes

inMinLineMagnitude describes strength of an edge within the code. This value determined automatically is usually correct, so this parameter can be viewed as a hint for the algorithm in nonstandard, tough cases.

Examples

Simple case.

Dark and lower quality code.



Multiple codes within one image.

Code viewed from an angle.

Remarks

To be correctly detected, the code should have safety area around it (of same brightness as the code background color) at least as wide as its unit. Moreover, unit size should be at least 2 pixels.

See Also

- ReadSingleQRCode – Detects and recognizes a single QR code on the input image.

ReadSingleCode_IK

Header: AVL.h
Namespace: avl
Module: Datacodes

Detects and recognizes Codes in one image.

Syntax

```
void avl::ReadSingleCode_IK
(
 const avl::Image& inImage,
 atl::Optional<const avl::Rectangle2D&> inRoi,
 avl::CodeInverse1D::Type inInverse,
 atl::Optional<int> inTimeout,
 const avl::CodeDetectionParameters& inDetectionParameters,
 bool inEnableCode39,
 bool inEnableCode128,
 bool inEnableInterleaved2of5,
 atl::Optional<const avl::DatamatrixParameters&> inEnableDatamatrix,
 bool inEnablePDF417,
 bool inEnableQRCode,
 bool inEnableUPCEAN,
 bool inEnableCode93,
 bool inEnableDotCode,
 bool inEnableMaxiCode,
 bool inEnableAztec,
 bool inEnableMSI,
 atl::Optional<const avl::CodabarParameters&> inEnableCodabar,
 atl::Conditional<atl::String>& outResult,
 atl::Conditional<avl::Path>& outOutline,
 atl::Conditional<avl::CodeFormat::Type>& outCodeFormat
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Rectangle2D&>	NIL	Region of interest
→	inInverse	CodeInverse1D::Type		
→	inTimeout	Optional<int>	NIL	Processing timeout
→	inDetectionParameters	const CodeDetectionParameters&		
→	inEnableCode39	bool	True	
→	inEnableCode128	bool	True	
→	inEnableInterleaved2of5	bool	True	
→	inEnableDatamatrix	Optional<const DatamatrixParameters&>	DatamatrixParameters ( )	
→	inEnablePDF417	bool	True	
→	inEnableQRCode	bool	True	
→	inEnableUPCEAN	bool	True	
→	inEnableCode93	bool		
→	inEnableDotCode	bool		
→	inEnableMaxiCode	bool		
→	inEnableAztec	bool		
→	inEnableMSI	bool		
→	inEnableCodabar	Optional<const CodabarParameters&>	NIL	
←	outResult	Conditional<String>&		
←	outOutline	Conditional<Path>&		
←	outCodeFormat	Conditional<CodeFormat::Type>&		

Errors

List of possible exceptions:

Error type	Description
DomainError	ReadSingleCode_IK is supported only in x64 environment.

ReadSingleDataMatrixCode

Header: AVL.h
Namespace: avl
Module: Datacodes

Detects and recognizes one Data Matrix code.

Syntax

```
void avl::ReadSingleDataMatrixCode
(
 const avl::Image& inImage,
 atl::Optional<const avl::Rectangle2D&> inRoi,
 atl::Optional<const avl::CoordinateSystem2D&> inRoiAlignment,
 const avl::DataMatrixCodeParams& inCodeParams,
 const avl::DataMatrixDetectionParams& inDetectionParams,
 atl::Conditional<avl::DataCode>& outDataMatrixCode,
 atl::Array<avl::Path>& outCandidates,
 atl::Optional<avl::Rectangle2D&> outAlignedRoi = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Rectangle2D&>	NIL	Region of interest
→	inRoiAlignment	Optional<const CoordinateSystem2D&>	NIL	Coordinate system for the region of interest
→	inCodeParams	const DataMatrixCodeParams&		Specification of codes that can be detected
→	inDetectionParams	const DataMatrixDetectionParams&		Parameters of the code detection algorithm
←	outDataMatrixCode	Conditional<DataCode>&		A code that has been correctly detected and decoded
←	outCandidates	Array<Path>&		Diagnostic information about detection results
←	outAlignedRoi	Optional<Rectangle2D&>	NIL	Input ROI after transformation (in the image coordinates)

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedRoi**.

Read more about Optional Outputs.

Description

This filter detects and recognizes one data matrix code in an image. If there are more codes the tool may find any of them.

Supported code types: ECC 200 and ECC 000-140.

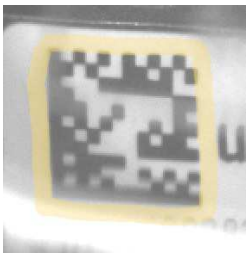
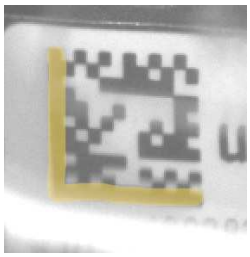
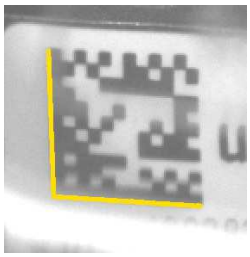
Groups of parameters

For optimal efficiency the tool may require careful setting of parameters. Most of the parameters are divided into two groups:

1. **inCodeParams** – parameters concerning the code; these can be set precisely according to what codes we expect to read
2. **inDetectionParams** – parameters concerning the algorithm; these may require some experimentation as different approaches may work best in different scenarios

Detection methods

The most important setting is the selection of the code detection algorithm, **inDetectionParams.DetectionMethod**. The tool offers three options:

	Quiet Zone	Finder Pattern	Finder Edges
Where it looks?			
How it works?	Looks for the blank space surrounding the code. In theory it should be empty for a width of at least one module.	Looks for the blob of the L-shaped pattern. In theory it should be continuous and not connected with anything outside of the code.	Looks for the edges of the L-shaped pattern. It uses a robust Line Segment Detector internally.
When to use it?	This method is good even for partially broken Finder Pattern, but it will fail if there is any contamination in the Quiet Zone.	This method may deal with some contamination in the Quiet Zone, but it will fail if the Finder Pattern itself is not good.	This method works well even if Quiet Zone is contaminated and Finder Pattern is not perfect, but may be confused by additional lines appearing near the code.

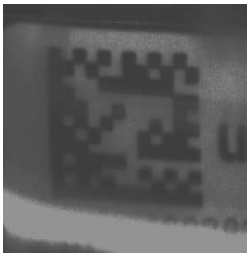
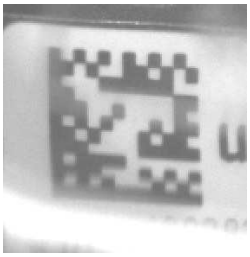
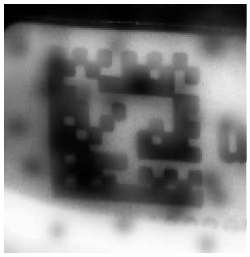
Pyramid

The tool tries to detect codes at different scale levels. It investigates images at different resolutions, starting with the lowest resolution. If it finds a sufficient number of codes at one level, it may skip looking further, thus achieving shorter execution time. If there are no codes in the input image, it will always scan all the levels, resulting in the highest execution time.

NOTE: If you experience instable execution times, for example switching between 5 ms and 20 ms, it may mean that for some images the tool finds a code at a higher level, while for some others it has to process a higher resolution too.

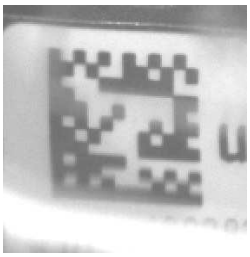
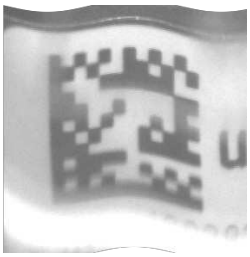
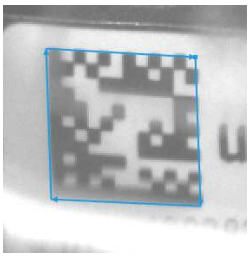
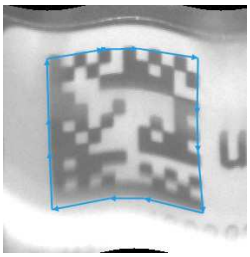
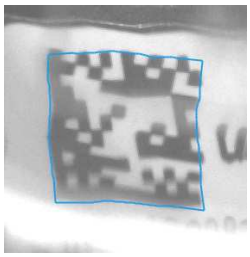
Finder Tradeoff ("FinderEdges" only)

When using the "FinderEdges" method, you should be aware that there is a tradeoff between the ability of the tool to detect rugged (uneven, noised or dot printed) edges and the ability to detect low contrast edges (or ones neighboring with a very narrow Quite Zone). The default setting is "Balanced". However, if you expect the edges of your code to be relatively straight and clean, but less strong, you may want to change it to "Sensitive". On the other hand, if the edges of the code are highly jagged or the surrounding is noisy, it may be better to choose "Robust". You may also set it to "Auto" and it will test several possibilities at the cost of much higher execution time.

Use "Sensitive"	Use "Balanced"	Use "Robust"
		

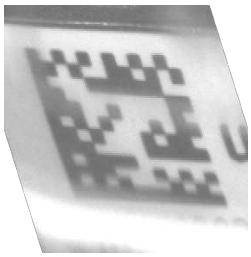
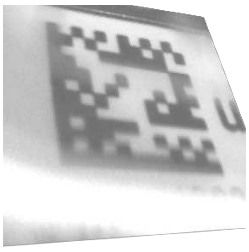
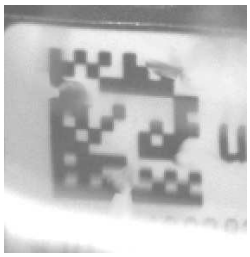
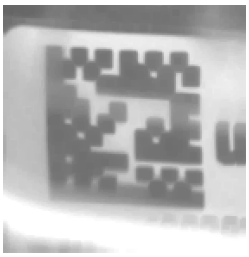
Distortions

The tool can deal with various types of code deformations. There are three possible settings, as demonstrated below. Please be aware that when we make the algorithm more robust to distortions, at the same time we make it less robust to broken code patterns. For this reason, we should stay with lower settings whenever possible.

Use "Low"	Use "Medium"	Use "High"
		
		
Each side is represented with a single line segment.	Each side is represented with a 3-segment line.	Each side is exactly traced with a polyline.

Damaged codes and other difficulties

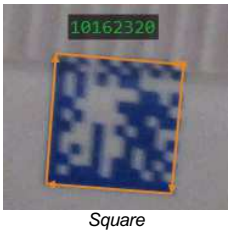
By using appropriate **inCodeParams** settings you can also find codes that are partially damaged. See the below table for some of the most common cases.

Use MaxSlant=30	Use AllowPerspective	Use AllowBrokenFinder and AllowBrokenTiming	Use AllowOversizedModules
			

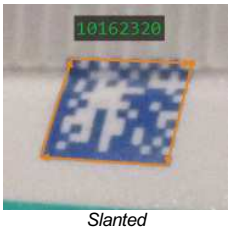
Hints

- Specify the range of possible codes in an image by setting the **inCodeParams** input. The more narrow is the specification, the faster the filter works.
  - Set **inCodeParams.Polarity** to specify whether dark-on-bright or bright-on-dark codes are to be read.
  - Specify the range of possible code sizes with **inCodeParams.MinRowCount**, **inCodeParams.MaxRowCount**, **inCodeParams.MinColumnCount** and **inCodeParams.MinColumnCount**.
  - Specify the range of possible module sizes (in pixels) with **inCodeParams.MinModuleSize** and **inCodeParams.MaxModuleSize**.
  - Set **inCodeParams.ExpectedGapSize** according to the following rules:
    - Zero – no gaps at all (modules are fully filled); use it also when there is other print very near to the code.
    - Small – gaps up to 25% of the module size; this is the default setting.
    - Medium – gaps up to 50% of the module size; it may require a bigger Quiet Zone.
    - Large – gaps up to 75%; comes with no guarantee.
  - Set **inCodeParams.MaxRectangleRatio** to 1 if you expect to work with square codes only. The value specifies maximal ratio between the length of the longer side to the length of the shorter side.
  - Pay attention to codes that are bent or unevenly printed.
    - For bent codes (e.g. printed on a bottle) use **inCodeParams.AllowDistortion** = Medium.
    - For unevenly printed codes or ones that are printed on wrinkled material use **inCodeParams.AllowDistortion** = High.
  - For potentially damaged codes, consider using **inCodeParams.AllowBrokenFinder** and **inCodeParams.AllowBrokenTiming**. These settings make the tool more robust against missing pieces of Finder Pattern and Timing Pattern respectively.
- If some codes are not detected, try modifying **inDetectionParams**:
  - In the first place make sure you set **inCodeParams** correctly.
  - Then try different detection methods. "QuietZone" and "FinderPattern" are faster because they work with blobs. "FinderEdges" however is usually the most reliable.
  - You may also try using more reliable but slower strategies for building the image pyramid (**inDetectionParams.PyramidStrategy**) and for fitting of the code's outline (**inDetectionParams.OutlineStrategy**). "Precise" should be ok for most cases, but "Strict" double-checks some options while "Extended" completely ignores performance for the sake of making sure we do not miss any code.
  - Increase **inDetectionParams.ContrastThreshold** if the noise is high. Decrease it, if the noise is low and the difference between the bright and the dark modules is low.

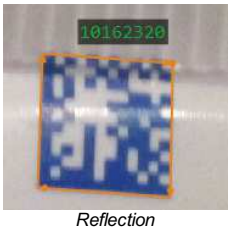
Examples



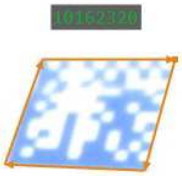
Square



Slanted



Reflection



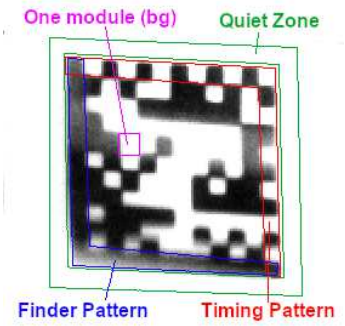
Overexposure

Remarks

For maximum reliability of reading Data Matrix codes we recommend making sure that the image fulfills the following standard requirements:

- Module size should be at least 4 pixels, while 5 pixels is recommended for the highest reliability. It is possible to read codes with modules as low as 1.25 pixels, but this comes with no guarantee and is highly dependent on the image and printing quality.
- Gaps between dot-printed marks should be not bigger than 50% of the module size, but the less the better.
- There should be an empty space around the code ("quiet zone"). It should have the width at least equal to the module size. Contamination, additional print or shadows in this area is the most frequent source of reduced reliability.
- Finder Pattern slant (angular deviation from 90 degrees) should not be greater than 30 degrees.
- Data Matrix codes are not designed for reading when there is strong perspective projection.

For explanation of the used terminology, please refer to the below picture:



See Also

- ReadMultipleDataMatrixCodes – Detects and recognizes several Data Matrix codes in one image.

ReadSinglePDF417Code

Header: AVL.h
Namespace: avl
Module: Datacodes

Detects and recognizes one PDF417 code.

Syntax

```
void avl::ReadSinglePDF417Code
(
 const avl::Image& inImage,
 atl::Optional<const avl::Rectangle2D&> inRoi,
 atl::Optional<const avl::CoordinateSystem2D&> inRoiAlignment,
 const avl::PDF417CodeParams& inCodeParams,
 const avl::PDF417DetectionParams& inDetectionParams,
 atl::Conditional<avl::PDF417Code>& outPDF417Code,
 atl::Optional<avl::Rectangle2D&> outAlignedRoi = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
➡	inImage	const Image&		Input image
➡	inRoi	Optional<const Rectangle2D&>	NIL	Region of interest
➡	inRoiAlignment	Optional<const CoordinateSystem2D&>	NIL	Coordinate system for the region of interest
➡	inCodeParams	const PDF417CodeParams&		Specification of codes that can be detected
➡	inDetectionParams	const PDF417DetectionParams&		Specification of the way the code is being detected
⬅	outPDF417Code	Conditional<PDF417Code>&		Found PDF417 code
⬅	outAlignedRoi	Optional<Rectangle2D&>	NIL	Input ROI after transformation (in the image coordinates)

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedRoi**.

Read more about Optional Outputs.

Hints

If code is not detected, try to use ResizeImage on it before processing with ReadSinglePDF417Code.

Remarks

In *inCodeParams* the parameter *MinModuleSize* should be more or equal to 3. For values less than 3 it is not guaranteed, that ReadSinglePDF417Code will process code properly.

ReadSingleQRCode

Header: AVL.h
Namespace: avl
Module: Datacodes

Detects and recognizes a single QR code on the input image.

Syntax

```
void avl::ReadSingleQRCode
(
 const avl::Image& inImage,
 atl::Optional<const avl::Rectangle2D&> inRoi,
 atl::Optional<const avl::CoordinateSystem2D&> inRoiAlignment,
 float inMinModuleSize,
 float inMaxModuleSize,
 atl::Optional<float> inContrastThreshold,
 atl::Optional<int> inPatternQuality,
 bool inAllowRotation,
 avl::Polarity::Type inPolarity,
 atl::Optional<float> inMinLineMagnitude,
 atl::Conditional<avl::QRCode>& outQRCode,
 atl::Optional<avl::Rectangle2D&> outAlignedRoi = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Rectangle2D&>		NIL	Range of pixels to be processed
→	inRoiAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the region of interest to the position of the inspected object
→	inMinModuleSize	float	1.5 - 100.0	4.0f	Lower estimated size of a code unit in pixels
→	inMaxModuleSize	float	1.5 - 100.0	40.0f	Upper estimated size of a code unit in pixels
→	inContrastThreshold	Optional<float>	1.0 - 255.0	NIL	Guaranteed gray level difference between dark and bright modules
→	inPatternQuality	Optional<int>	1 - 3	NIL	Quality of the code from 1 (extremely deformed) to 3 (perfect)
→	inAllowRotation	bool		True	Allows codes rotated in relation to the axes of the input image
→	inPolarity	Polarity::Type		Any	Specifies whether code is darker or brighter than the background
→	inMinLineMagnitude	Optional<float>	1.0 - 255.0	NIL	Strength of an edge within the code; default depends on parameters
←	outQRCode	Conditional<QRCode>&			
←	outAlignedRoi	Optional<Rectangle2D&>		NIL	Input ROI after transformation (in the image coordinates)

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedRoi**.

Read more about Optional Outputs.

Description

The filter locates and decodes QR code on the image (**inImage**) within given region (**inRoi**). The encoded text length must be greater than 2.

inMinModuleSize is the expected lower bound of one module (smallest unit of the code).

inMinModuleSize is the expected upper bound of one module (smallest unit of the code).

Parameter **inContrastThreshold** describes contrast of the code - namely, it denotes the brightness difference between light and dark modules of the code. If not given, the algorithm uses normalization of the image to enhance the code and computes this value automatically.

inPatternQuality is an integer from the range <1,3> describing how much the code is blurred or the positional patterns are deformed. For instance, value 3 corresponds to a perfect quality code, while value 1 to an extremely blurred one. For standard cases it is recommended to use quality of 2 or to leave it default.

inAllowRotation Allows codes rotated in relation to the axes of the input image.

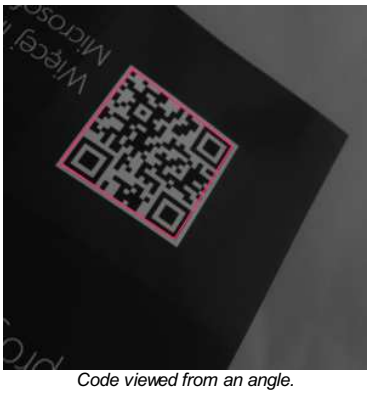
inPolarity Setup polarity of QR code can improve performance of ReadingQRCodes

inMinLineMagnitude describes strength of an edge within the code. This value determined automatically is usually correct, so this parameter can be viewed as a hint for the algorithm in nonstandard, tough cases.

Examples

Simple case.

Dark and lower quality code.



Code viewed from an angle.

Remarks

To be correctly detected, the code should have safety area around it (of same brightness as the code background color) at least as wide as its unit. Moreover, unit size should be at least 2 pixels.

See Also

- ReadMultipleQRCodes – Detects and recognizes all QR codes on the input image.

82. 2D Edge Detection

Table of content:

DetectEdges

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Extracts a binary image of pixel-precise continuous edges.

Applications: Consistent detection of pixels that belong to contours of variable or unpredictable shape, e.g. screw thread outline or a custom piece of textile.

Syntax

```
void avl::DetectEdges
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::EdgeFilter::Type inEdgeFilter,
 float inStdDevX,
 atl::Optional<float> inStdDevY,
 float inEdgeThreshold,
 float inEdgeHysteresis,
 float inMaxJoiningDistance,
 const int inMinBlobArea,
 avl::Image& outEdgesImage,
 avl::Image& diagGradientMagnitudeImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image from which edges will be extracted
→	inRoi	Optional<const Region&>		NIL	Region of the image from which edges will be extracted
→	inEdgeFilter	EdgeFilter::Type			Type of edge filter used for computing gradients
→	inStdDevX	float	0.0 - ∞	2.0f	Amount of horizontal smoothing used by the edge filter
→	inStdDevY	Optional<float>	0.0 - ∞	NIL	Amount of vertical smoothing used by the edge filter (Auto = inStdDevX)
→	inEdgeThreshold	float	0.0 - ∞	15.0f	Sufficient edge strength; edges of that strength will always be detected
→	inEdgeHysteresis	float	0.0 - ∞	5.0f	Value by which the edge threshold is decreased for edge points neighboring with sufficiently strong edges
→	inMaxJoiningDistance	float	0.0 - ∞	0.0f	Maximal distance between edges that can be joined
→	inMinBlobArea	const int	0 - ∞	1	Minimal area of an edge blob
←	outEdgesImage	Image&			Image of found edges
🔍	diagGradientMagnitudeImage	Image&			Visualization of the gradient magnitude

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inImage** and **outEdgesImage**

Read more about In-place Computation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

DetectEdges_AsPaths

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Extracts subpixel-precise paths that represent continuous edges.

Applications: Consistent detection of contours of variable or unpredictable shape, e.g. screw thread outline or a custom piece of textile.

Syntax

```
void avl::DetectEdges_AsPaths
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::EdgeFilter::Type inEdgeFilter,
 float inStdDevX,
 atl::Optional<float> inStdDevY,
 float inEdgeThreshold,
 float inEdgeHysteresis,
 atl::Optional<float> inMaxJoiningDistance,
 float inMaxJoiningAngle,
 float inJoiningDistanceBalance,
 atl::Optional<float> inJoiningEndingLength,
 float inMinEdgeLength,
 atl::Array<avl::Path>& outEdges,
 avl::Image& diagGradientMagnitudeImage,
 avl::Region& diagEdgeRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image from which edges will be extracted
→	inRoi	Optional<const Region&>		NIL	Region of the image from which edges will be extracted
→	inEdgeFilter	EdgeFilter::Type			Type of edge filter used for computing gradients
→	inStdDevX	float	0.0 - ∞	2.0f	Amount of horizontal smoothing used by the edge filter
→	inStdDevY	Optional<float>	0.0 - ∞	NIL	Amount of vertical smoothing used by the edge filter (Auto = inStdDevX)
→	inEdgeThreshold	float	0.0 - ∞	15.0f	Sufficient edge strength; edges of that strength will always be detected
→	inEdgeHysteresis	float	0.0 - ∞	5.0f	Value by which the edge threshold is decreased for edge points neighboring with sufficiently strong edges
→	inMaxJoiningDistance	Optional<float>	0.0 - ∞	NIL	Maximal distance between edges that can be joined
→	inMaxJoiningAngle	float	0.0 - 180.0	30.0f	Maximal allowed angle between edges being joined
→	inJoiningDistanceBalance	float	0.0 - 1.0	0.0f	Determines how important distance between edges is according to their angle difference
→	inJoiningEndingLength	Optional<float>	1.0 - ∞	NIL	Determines the length of the edge end used for edge angle computing
→	inMinEdgeLength	float	0.0 - ∞	0.0f	Minimal length of an edge
←	outEdges	Array<Path>&			Paths representing found edges
🔍	diagGradientMagnitudeImage	Image&			Visualization of the gradient magnitude
🔍	diagEdgeRegion	Region&			Region representing found edges

Description

The operation extracts edges from the **inRoi** region in the **inImage** image and stores the result in the array of paths **outEdges**. The extraction process is the same as in DetectEdges_AsRegion, the only difference being the data type of the result. This filter returns an array of subpixel-precise paths rather than region computed by DetectEdges_AsRegion.

The extraction process starts from gradient computing. Depending on which edge filter is chosen, gradients are computed using recursive (Deriche or Lanser) or standard non-recursive filter proposed by Canny. The **inStdDevX** and **inStdDevY** parameters control size of the filter mask. The greater their values are, the larger this size is. It should be noted that the execution time of the non-recursive filter depends greatly on the size of the filter mask, while the recursive filters execution time is independent of it.

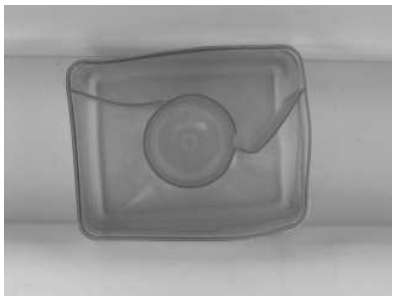
On the so computed gradient image threshold with hysteresis (as in ThresholdImage_Hysteresis) is performed with **inEdgeThreshold** and **inEdgeHysteresis** parameters. After this step only gradients which are strong enough are present. The resulting edge region can be much too thick, thus it has to be thinned. To achieve this, the non-maximum suppression is used. Every pixel with at least one of its neighbors having larger gradient is no longer considered to be an edge pixel (only neighbors in the direction of pixel's gradient matter).

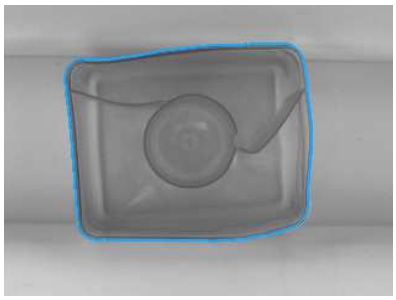
The positions of found edges are then determined with subpixel precision. The so extracted edges can then undergo further post-processing, if necessary. An edge can be joined with another one in its vicinity, but only if the distance between them is not greater than **inMaxJoiningDistance** and neither of them forms an angle greater than **inMaxJoiningAngle** with the segment connecting their endings. The **inJoiningEndingLength** parameter determines what part of the edge ending is used to determine above mentioned turn angle. If it is set to Nil, only the last edge segment is decisive. Sometimes two or more edges can be joined with an edge. Because only one of them can be, each of the candidates is evaluated based on the distance from the fixed edge's ending and the turn angle associated with their connection. The greater **inJoiningDistanceBalance** is, the less influence on the evaluation score the latter value has. After the joining phase, edges that are shorter than **inMinEdgeLength** are removed from the final results.

Hints

- Connect an input image to the **inImage** input.
- Start with **inEdgeHysteresis** = 0 and set **inEdgeThreshold** so that each important edge is at least partially detected. Then keep increasing **inEdgeHysteresis** until the edges are detected completely.
- If the edges are rugged or there are too many false edges, then try increasing **inStdDevX**.
- Do not change **inEdgeFilter**.
- Define **inMaxJoiningDistance** and **inMaxJoiningAngle** to fix small discontinuities in the resulting edges. This may be very useful when some gaps appear due to noise, dust or other irregularities that preclude correct object detection.
- Sometimes some of the edges being joined have bent endings and **inMaxJoiningAngle** would have to be large. Increase **inJoiningEndingLength** to 3-5 to improve that.
- Use **inMinEdgeLength** to remove short edges that appear due to noise.

Examples





DetectEdges_AsPaths *performed on the sample image with* ***inEdgeFilter*** *= Canny,* ***inStdDevX*** *= 2.5,* ***inStdDevY*** *= 2.5,* ***inEdgeThreshold*** *= 10,* ***inEdgeHysteresis*** *= 5,* ***inMinEdgeLength*** *= 1000.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- DetectEdges_AsRegion – Extracts a pixel-precise region of continuous edges.
- DetectEdges_AsRegion_Mask – Extracts a pixel-precise region of continuous edges. Faster, yet less accurate version.
- DetectEdges_AsPaths_Mask – Extracts subpixel-precise paths that represent continuous edges. Faster, yet less accurate version.

DetectEdges_AsPaths_Mask

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Extracts subpixel-precise paths that represent continuous edges. Faster, yet less accurate version.

Applications: Consistent detection of contours of variable or unpredictable shape, e.g. screw thread outline or a custom piece of textile.

Syntax

```
void avl::DetectEdges_AsPaths_Mask
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::EdgeMaskFilter::Type inEdgeMaskFilter,
 float inEdgeThreshold,
 float inEdgeHysteresis,
 atl::Optional<float> inMaxJoiningDistance,
 float inMaxJoiningAngle,
 float inJoiningDistanceBalance,
 atl::Optional<float> inJoiningEndingLength,
 float inMinEdgeLength,
 atl::Array<avl::Path>& outEdges,
 avl::Image& diagGradientMagnitudeImage,
 avl::Region& diagEdgeRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Image from which edges will be extracted
→	inRoi	Optional<const Region&>		NIL	Region of the image from which edges will be extracted
→	inEdgeMaskFilter	EdgeMaskFilter::Type			Type of edge filter used for computing gradients
→	inEdgeThreshold	float	0.0 - ∞	35.0f	Sufficient edge strength; edges of that strength will always be detected
→	inEdgeHysteresis	float	0.0 - ∞	15.0f	Value by which the edge threshold is decreased for edge points neighboring with sufficiently strong edges
→	inMaxJoiningDistance	Optional<float>	0.0 - ∞	NIL	Maximal distance between edges that can be joined
→	inMaxJoiningAngle	float	0.0 - 180.0	30.0f	Maximal allowed angle between edges being joined
→	inJoiningDistanceBalance	float	0.0 - 1.0	0.0f	Determines how important distance between edges is according to their angle difference
→	inJoiningEndingLength	Optional<float>	1.0 - ∞	NIL	Determines the length of the edge end used for edge angle computing
→	inMinEdgeLength	float	0.0 - ∞	0.0f	Minimal length of an edge
←	outEdges	Array<Path>&			Paths representing found edges
🔍	diagGradientMagnitudeImage	Image&			Visualization of the gradient magnitude
🔍	diagEdgeRegion	Region&			Region representing found edges

Description

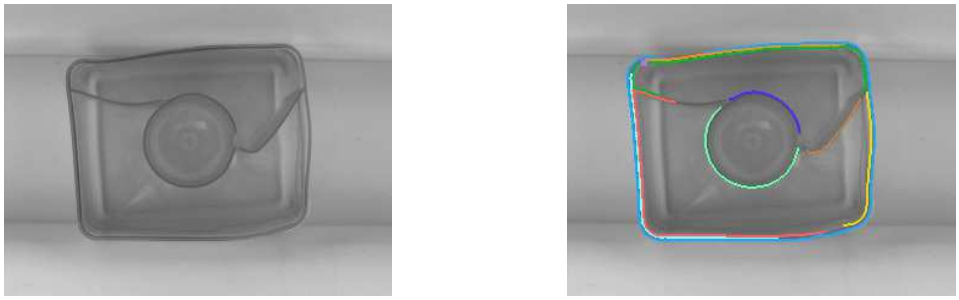
The operation extracts edges from the **inRoi** region in the **inImage** image and stores the result in the array of paths **outEdges**. The extraction process is the same as in DetectEdges_AsRegion_Mask, the only difference being the data type of the result. This filter returns an array of subpixel-precise paths rather than region computed by DetectEdges_AsRegion_Mask.

The extraction process starts from gradient computing, what is done using chosen non-recursive filter with fixed size mask.

On the so computed gradient image threshold with hysteresis (as in ThresholdImage_Hysteresis) is performed with **inEdgeThreshold** and **inEdgeHysteresis** parameters. After this step only gradients which are strong enough are present. The resulting edge region can be much too thick, thus it has to be thinned. To achieve this, the non-maximum suppression is used. Every pixel with at least one of its neighbors having larger gradient is no longer considered to be an edge pixel (only neighbors in the direction of pixel's gradient matter).

The positions of found edges are then determined with subpixel precision. The so extracted edges can then undergo further post-processing, if necessary. An edge can be joined with another one in its vicinity, but only if the distance between them is not greater than **inMaxJoiningDistance** and neither of them forms an angle greater than **inMaxJoiningAngle** with the segment connecting their endings. The **inJoiningEndingLength** parameter determines what part of the edge ending is used to determine above mentioned turn angle. If it is set to Nil, only the last edge segment is decisive. Sometimes two or more edges can be joined with an edge. Because only one of them can be, each of the candidates is evaluated based on the distance from the fixed edge's ending and the turn angle associated with their connection. The greater **inJoiningDistanceBalance** is, the less influence on the evaluation score the latter value has. After the joining phase, edges that are shorter than **inMinEdgeLength** are removed from the final results.

Examples



DetectEdges_AsPaths_Mask *performed on the sample image with* ***inEdgeMaskFilter*** *= Sobel,* ***inEdgeThreshold*** *= 20,* ***inEdgeHysteresis*** *= 5.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Unsupported edge mask filter in DetectEdges_AsPaths_Mask.

See Also

- DetectEdges_AsRegion – Extracts a pixel-precise region of continuous edges.
- DetectEdges_AsRegion_Mask – Extracts a pixel-precise region of continuous edges. Faster, yet less accurate version.
- DetectEdges_AsPaths – Extracts subpixel-precise paths that represent continuous edges.

DetectEdges_AsRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Extracts a pixel-precise region of continuous edges.

Applications: Consistent detection of pixels that belong to contours of variable or unpredictable shape, e.g. screw thread outline or a custom piece of textile.

Syntax

```
void avl::DetectEdges_AsRegion
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::EdgeFilter::Type inEdgeFilter,
 float inStdDevX,
 atl::Optional<float> inStdDevY,
 float inEdgeThreshold,
 float inEdgeHysteresis,
 float inMaxJoiningDistance,
 const int inMinBlobArea,
 avl::Region& outEdgeRegion,
 avl::Image& diagGradientMagnitudeImage
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inImage	const Image&			Image from which edges will be extracted
➡	inRoi	Optional<const Region&>		NIL	Region of the image from which edges will be extracted
➡	inEdgeFilter	EdgeFilter::Type			Type of edge filter used for computing gradients
➡	inStdDevX	float	0.0 - ∞	2.0f	Amount of horizontal smoothing used by the edge filter
➡	inStdDevY	Optional<float>	0.0 - ∞	NIL	Amount of vertical smoothing used by the edge filter
➡	inEdgeThreshold	float	0.0 - ∞	15.0f	Sufficient edge strength; edges of that strength will always be detected
➡	inEdgeHysteresis	float	0.0 - ∞	5.0f	Value by which the edge threshold is decreased for edge points neighboring with sufficiently strong edges
➡	inMaxJoiningDistance	float	0.0 - ∞	0.0f	Maximal distance between edges that can be joined
➡	inMinBlobArea	const int	0 - ∞	1	Minimal area of an edge blob
⬅	outEdgeRegion	Region&			Region of the found edges
🔍	diagGradientMagnitudeImage	Image&			Visualization of the gradient magnitude

Description

The operation extracts edges from the **inRoi** region in the **inImage** image and stores the result in the **outEdgeRegion** region. The extraction process is the same as in DetectEdges_AsPaths, the only difference being the data type of the result. This filter returns a region rather than array of subpixel-precise paths computed by DetectEdges_AsPaths.

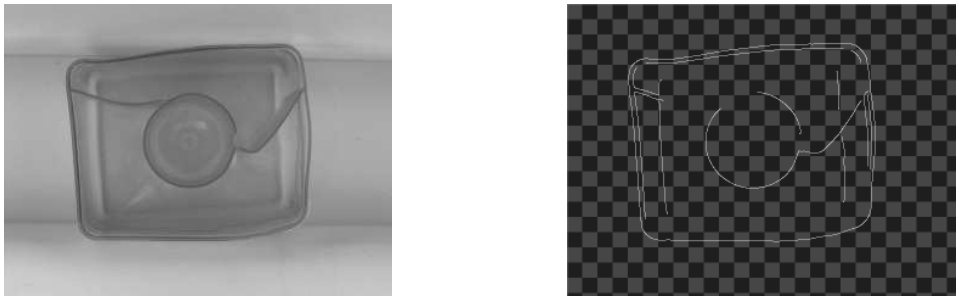
The extraction process starts from gradient computing. Depending on which edge filter is chosen, gradients are computed using recursive (Deriche or Lanser) or standard non-recursive filter proposed by Canny. The **inStdDevX** and **inStdDevY** parameters control size of the filter mask. The greater their values are, the larger this size is. It should be noted that the execution time of the non-recursive filter depends greatly on the size of the filter mask, while the recursive filters execution time is independent of it.

On the so computed gradient image threshold with hysteresis (as in ThresholdImage_Hysteresis) is performed with **inEdgeThreshold** and **inEdgeHysteresis** parameters. After this step only gradients which are strong enough are present. The resulting edge region can be much too thick, thus it has to be thinned. To achieve this, the non-maximum suppression is used. Every pixel with at least one of its neighbors having larger gradient is no longer considered to be an edge pixel (only neighbors in the direction of pixel's gradient matter).

Hints

- Connect an input image to the **inImage** input.
- Start with **inEdgeHysteresis** = 0 and set **inEdgeThreshold** so that each important edge is at least partially detected. Then keep increasing **inEdgeHysteresis** until the edges are detected completely.
- If the edges are rugged or there are too many false edges, then try increasing **inStdDevX**.
- Do not change **inEdgeFilter**.

Examples



DetectEdges_AsRegion *performed on the sample image with* ***inEdgeFilter*** *= Canny,* ***inStdDevX*** *= 2.0,* ***inStdDevY*** *= 2.0,* ***inEdgeThreshold*** *= 10,* ***inEdgeHysteresis*** *= 5.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- DetectEdges_AsRegion_Mask – Extracts a pixel-precise region of continuous edges. Faster, yet less accurate version.
- DetectEdges_AsPaths – Extracts subpixel-precise paths that represent continuous edges.
- DetectEdges_AsPaths_Mask – Extracts subpixel-precise paths that represent continuous edges. Faster, yet less accurate version.

DetectEdges_AsRegion_Mask

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Extracts a pixel-precise region of continuous edges. Faster, yet less accurate version.

Applications: Consistent detection of pixels that belong to contours of variable or unpredictable shape, e.g. screw thread outline or a custom piece of textile.

Syntax

```
void avl::DetectEdges_AsRegion_Mask
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::EdgeMaskFilter::Type inEdgeMaskFilter,
 float inEdgeThreshold,
 float inEdgeHysteresis,
 float inMaxJoiningDistance,
 const int inMinBlobArea,
 avl::Region& outEdgeRegion,
 avl::Image& diagGradientMagnitudeImage
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inImage	const Image&			Image from which edges will be extracted
➡	inRoi	Optional<const Region&>		NIL	Region of the image from which edges will be extracted
➡	inEdgeMaskFilter	EdgeMaskFilter::Type			Type of edge filter used for computing gradients
➡	inEdgeThreshold	float	0.0 - ∞	35.0f	Sufficient edge strength; edges of that strength will always be detected
➡	inEdgeHysteresis	float	0.0 - ∞	15.0f	Value by which the edge threshold is decreased for edge points neighboring with sufficiently strong edges
➡	inMaxJoiningDistance	float	0.0 - ∞	0.0f	Maximal distance between edges that can be joined
➡	inMinBlobArea	const int	0 - ∞	1	Minimal area of an edge blob
⬅	outEdgeRegion	Region&			Region of the found edges
🔍	diagGradientMagnitudeImage	Image&			Visualization of the gradient magnitude

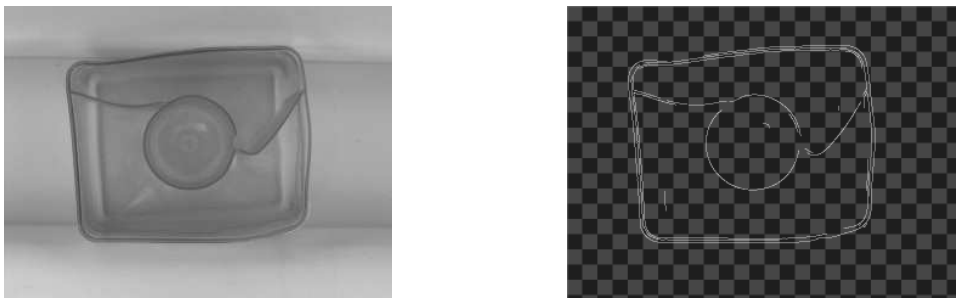
Description

The operation extracts edges from the **inRoi** region in the **inImage** image and stores the result in the **outEdgeRegion** region. The extraction process is the same as in DetectEdges_AsPaths_Mask, the only difference being the data type of the result. This filter returns a region rather than array of subpixel-precise paths computed by DetectEdges_AsPaths_Mask.

The extraction process starts from gradient computing, what is done using chosen non-recursive filter with fixed size mask.

On the so computed gradient image threshold with hysteresis (as in ThresholdImage_Hysteresis) is performed with **inEdgeThreshold** and **inEdgeHysteresis** parameters. After this step only gradients which are strong enough are present. The resulting edge region can be much too thick, thus it has to be thinned. To achieve this, the non-maximum suppression is used. Every pixel with at least one of its neighbors having larger gradient is no longer considered to be an edge pixel (only neighbors in the direction of pixel's gradient matter).

Examples



DetectEdges_AsRegion_Mask *performed on the sample image with* ***inEdgeMaskFilter*** *= Sobel,* ***inEdgeThreshold*** *= 20,* ***inEdgeHysteresis*** *= 5.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Unsupported edge mask filter in DetectEdges_AsRegion_Mask.

See Also

- DetectEdges_AsRegion – Extracts a pixel-precise region of continuous edges.
- DetectEdges_AsPaths – Extracts subpixel-precise paths that represent continuous edges.
- DetectEdges_AsPaths_Mask – Extracts subpixel-precise paths that represent continuous edges. Faster, yet less accurate version.

DetectRidges_AsPaths

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Extracts subpixel-precise paths that represent bright or dark thin lines.

Applications: Consistent detection of thin structures like scratches, cracks or lines.

Syntax

```
void avl::DetectRidges_AsPaths
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 float inStdDevX,
 atl::Optional<float> inStdDevY,
 float inRidgeThreshold,
 float inRidgeHysteresis,
 avl::Polarity::Type inPolarity,
 atl::Array<avl::Path>& outRidges
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Region of interest
→	inStdDevX	float	0.0 - ∞	3.0f	Smoothing standard deviation (horizontal)
→	inStdDevY	Optional<float>	0.0 - ∞	NIL	Smoothing standard deviation (vertical, or Auto = horizontal)
→	inRidgeThreshold	float	0.0 - ∞	5.0f	Sufficient ridge strength; ridges of that strength will always be detected
→	inRidgeHysteresis	float	0.0 - ∞	4.0f	Value by which the ridge threshold is decreased for ridge points neighboring with sufficiently strong edges
→	inPolarity	Polarity::Type			
←	outRidges	Array<Path>&			

Hints

- Connect an input image to the **inImage** input.
- Set **inPolarity** in accordance to which type of thin lines you want to detect.
- Start with **inRidgeHysteresis** = 0 and set **inRidgeThreshold** so that each important edge is at least partially detected. Then keep increasing **inRidgeHysteresis** until the edges are detected completely.
- If the edges are rugged or there are too many false edges, then try increasing **inStdDevX**.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

DetectRidges_AsRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Extracts a pixel-precise region of bright or dark thin lines.

Applications: Consistent detection of thin structures like scratches, cracks or lines.

Syntax

```
void avl::DetectRidges_AsRegion
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 float inStdDevX,
 atl::Optional<float> inStdDevY,
 float inRidgeThreshold,
 float inRidgeHysteresis,
 avl::Polarity::Type inPolarity,
 const int inMinBlobArea,
 avl::Region& outRidgeRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Region of interest
→	inStdDevX	float	0.0 - ∞	3.0f	Smoothing standard deviation (horizontal)
→	inStdDevY	Optional<float>	0.0 - ∞	NIL	Smoothing standard deviation (vertical, or Auto = horizontal)
→	inRidgeThreshold	float	0.0 - ∞	5.0f	Sufficient ridge strength; ridges of that strength will always be detected
→	inRidgeHysteresis	float	0.0 - ∞	4.0f	Value by which the ridge threshold is decreased for ridge points neighboring with sufficiently strong edges
→	inPolarity	Polarity::Type			
→	inMinBlobArea	const int	0 - ∞	1	Minimal area of a ridge blob
←	outRidgeRegion	Region&			

Hints

- Connect an input image to the **inImage** input.
- Set **inPolarity** in accordance to which type of thin lines you want to detect.
- Start with **inRidgeHysteresis** = 0 and set **inRidgeThreshold** so that each important edge is at least partially detected. Then keep increasing **inRidgeHysteresis** until the edges are detected completely.
- If the edges are rugged or there are too many false edges, then try increasing **inStdDevX**.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

83. Hough Transform

Table of content:

DetectLines

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Finds lines in an image using Hough Transform.

Syntax

```
void avl::DetectLines
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 float inAngleResolution,
 float inMinAngleDelta,
 float inMinDistance,
 float inMinScore,
 float inEdgeThreshold,
 atl::Array<avl::Line2D>& outLines,
 atl::Array<float>& outScores,
 avl::Image& diagGradientMagnitudeImage,
 avl::Image& diagScoreImage
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inImage	const Image&			Input image
➡	inRoi	Optional<const Region&>		NIL	Input region of interest
➡	inAngleResolution	float	0.1 - 180.0	1.0f	Resolution of lines' orientation
➡	inMinAngleDelta	float	0.0 - ∞	20.0f	Minimum angle between two lines
➡	inMinDistance	float	0.0 - ∞	20.0f	Minimum distance between two lines
➡	inMinScore	float	0.0 - ∞	20.0f	Minimum matching score
➡	inEdgeThreshold	float		10.0f	Minimum accepted edge magnitude
⬅	outLines	Array<Line2D>&			Output lines
⬅	outScores	Array<float>&			Output scores
🔍	diagGradientMagnitudeImage	Image&			Visualized gradients magnitude of an input image
🔍	diagScoreImage	Image&			Calculated score for each pixel of an input image

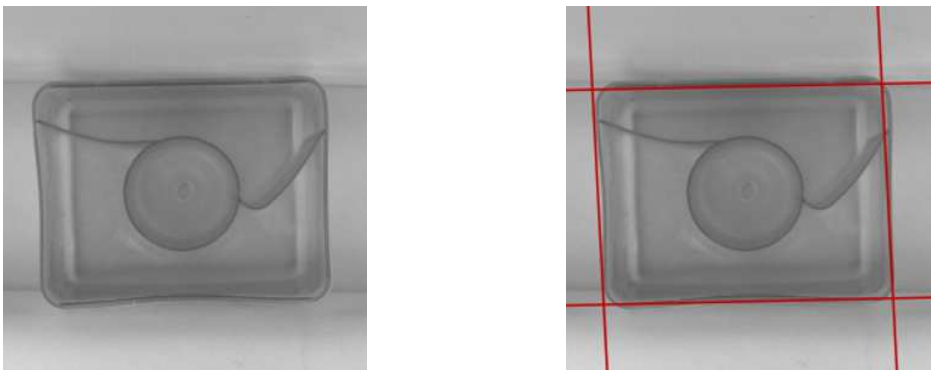
Description

The operation detects straight edges in the **inImage** using the Hough Transform approach and treats them as infinite lines. The output array is ordered from best matching to worst matching results.

The parameter **inAngleResolution** specifies the precision of detected lines' orientations. Value of *n* means the filter will be able to reliably distinguish lines in *n*-degree increments.

Parameters **inMinAngleDelta** and **inMinDistance** are used for neighbouring results suppression. **inMinAngleDelta** specifies the minimum angle between two lines while **inMinDistance** specifies the minimum distance between two parallel lines.

Examples



DetectLines *performed on the sample image with* ***inMinScore*** *= 0.3.*

Remarks

Low values of **inAngleResolution** (under 0.5) may cause high memory consumption and decrease in performance.

See Also

- DetectMultipleCircles – Finds circles of a given radius in the input image using Hough Transform.
- DetectPaths – Finds a specified shape in an image using Hough Transform.
- DetectSegments – Finds segments in an image using Hough Transform.

DetectMultipleCircles

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Finds circles of a given radius in the input image using Hough Transform.

Applications: Detection of circular or close-to-circular objects like holes, pins, pills, particles.

Syntax

```
void avl::DetectMultipleCircles
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 float inRadius,
 float inMaxOverlap,
 float inMinScore,
 float inEdgeThreshold,
 atl::Array<avl::HoughCircle>& outCircles,
 avl::Image& diagGradientMagnitudeImage,
 avl::Image& diagScoreImage
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inImage	const Image&			Input image
➡	inRoi	Optional<const Region&>		NIL	Input region of interest
➡	inRadius	float	0.0 - ∞	10.0f	Circles' radius
➡	inMaxOverlap	float	0.0 - 1.0	0.1f	Maximum accepted overlapping coefficient
➡	inMinScore	float	0.0 - ∞	20.0f	Minimum matching score
➡	inEdgeThreshold	float		10.0f	Minimum accepted edge magnitude
⬅	outCircles	Array<HoughCircle>&			Found circles
🔍	diagGradientMagnitudeImage	Image&			Visualized gradients magnitude of an input image
🔍	diagScoreImage	Image&			Calculated score for each pixel of an input image

Description

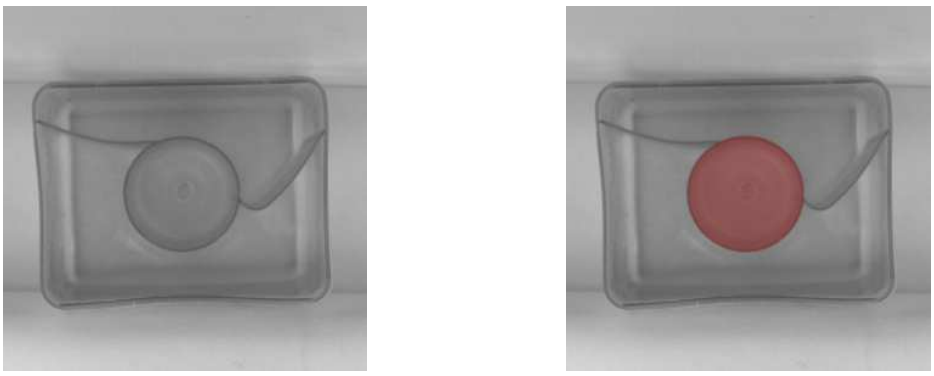
The operation detects circular objects of given radius (in pixels) in the **inImage** using the Hough Transform approach. The output array is ordered from best matching to worst matching results.

The parameter **inMaxOverlap** defines how much the detected circles can overlap. The value of 0 means no overlapping is allowed, and also that each circle must be fully contained in the search ROI, whereas the value of 1 allows full overlapping.

Hints

- Pass an input image to the **inImage** input.
- Define the expected circle radius on the **inRadius** input.
- Set **inEdgeThreshold** to define the minimum strength of edges that will be taken into account. Verify this value with the **diagGradientMagnitudeImage** output.
- Experimentally set the **inMinScore** value, whose meaning is more or less "the number of pixels voting for a particular object location".

Examples



DetectMultipleCircles *performed on the sample image.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- DetectSingleCircle – Finds the strongest circle of a given radius in the input image.
- DetectLines – Finds lines in an image using Hough Transform.
- DetectPaths – Finds a specified shape in an image using Hough Transform.
- DetectSegments – Finds segments in an image using Hough Transform.

DetectPaths

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Finds a specified shape in an image using Hough Transform.

Applications: This is an old algorithm for template matching. Quite slow.

Syntax

```
void avl::DetectPaths
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 const avl::Path& inPath,
 float inMinScore,
 float inEdgeThreshold,
 atl::Array<avl::Path>& outPaths,
 atl::Array<float>& outScores,
 avl::Image& diagGradientMagnitudeImage,
 avl::Image& diagScoreImage
)
```

Parameters

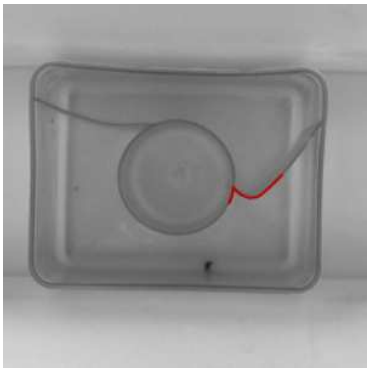
	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Input region of interest
→	inPath	const Path&			Input path
→	inMinScore	float	0.0 - ∞	20.0f	Minimum matching score
→	inEdgeThreshold	float		10.0f	Minimum accepted edge magnitude
←	outPaths	Array<Path>&			Output paths
←	outScores	Array<float>&			Output scores
🔍	diagGradientMagnitudeImage	Image&			Visualized gradients magnitude of an input image
🔍	diagScoreImage	Image&			Calculated score for each pixel of an input image

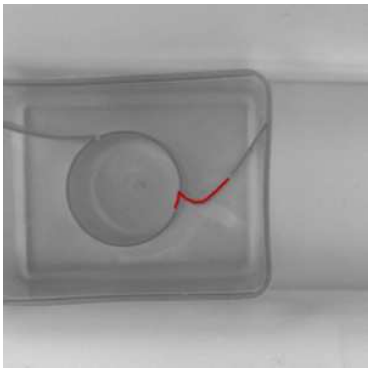
Description

The operation detects paths in the **inImage** using the Generalized Hough Transform approach. The output array is ordered from best matching to worst matching results.

Examples

A sample path (scaled for convenience).





***DetectPaths** performed on the sample images with **inMinScore** = 0.7.*

Remarks

DetectPaths is not scale- or rotation-invariant (slightly scaled or rotated paths are, however, detected properly).

Long **inPath**s cause long computation time.

Errors

List of possible exceptions:

Error type	Description
DomainError	Degenerate path in DetectPaths.

See Also

- DetectLines – Finds lines in an image using Hough Transform.
- DetectMultipleCircles – Finds circles of a given radius in the input image using Hough Transform.
- DetectSegments – Finds segments in an image using Hough Transform.

DetectSegments

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Finds segments in an image using Hough Transform.

Syntax

```
void avl::DetectSegments
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 float inAngleResolution,
 float inMinAngleDelta,
 float inMinDistance,
 float inMinLength,
 float inMinScore,
 float inEdgeThreshold,
 atl::Array<avl::Segment2D>& outSegments,
 avl::Image& diagGradientMagnitudeImage,
 avl::Image& diagScoreImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Input region of interest
→	inAngleResolution	float	0.1 - 180.0	1.0f	Resolution of segments' orientation
→	inMinAngleDelta	float	0.0 - ∞	20.0f	Minimum angle between two segments
→	inMinDistance	float	0.0 - ∞	20.0f	Minimum distance between two segments
→	inMinLength	float	0.0 - ∞	20.0f	Minimum segment length
→	inMinScore	float	0.0 - ∞	20.0f	Minimum matching score
→	inEdgeThreshold	float		10.0f	Minimum accepted edge magnitude
←	outSegments	Array<Segment2D>&			Output segments
🔍	diagGradientMagnitudeImage	Image&			Visualized gradients magnitude of an input image
🔍	diagScoreImage	Image&			Calculated score for each pixel of an input image

Description

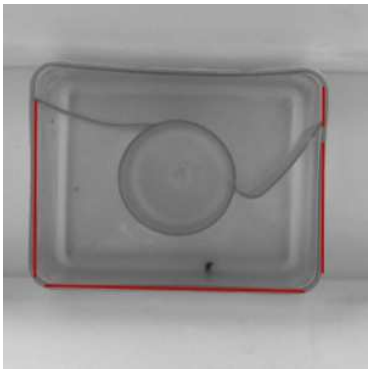
The operation detects straight edges in the **inImage** using the Hough Transform approach and treats them as segments by tracing gradients along lines. The output array is ordered from best matching to worst matching results.

The parameter **inAngleResolution** specifies the precision of detected lines' orientations. Value of *n* means the filter will be able to reliably distinguish lines in *n*-degree increments.

Parameters **inMinAngleDelta** and **inMinDistance** are used for neighbouring results suppression. **inMinAngleDelta** specifies the minimum angle between two lines while **inMinDistance** specifies the minimum distance between two parallel lines. Parameter **inMinLength** limits the length of segments (in pixels), suppressing too short segments.

The orientations of the resulting segments are always between 0 and 180 degrees.

Examples



***DetectSegments** performed on the sample image with **inMinScore** = 0.5, **inMinLength** = 25.*

Remarks

Low values of **inAngleResolution** (under 0.5) may cause high memory consumption and decrease in performance.

See Also

- DetectLines – Finds lines in an image using Hough Transform.
- DetectMultipleCircles – Finds circles of a given radius in the input image using Hough Transform.
- DetectPaths – Finds a specified shape in an image using Hough Transform.

DetectSingleCircle

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Finds the strongest circle of a given radius in the input image.

Applications: Detection of a circular or close-to-circular object like a hole, pin, pill or particle.

Syntax

```
void avl::DetectSingleCircle
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 float inRadius,
 float inMinScore,
 float inEdgeThreshold,
 atl::Conditional<avl::HoughCircle>& outCircle,
 avl::Image& diagGradientMagnitudeImage,
 avl::Image& diagScoreImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Input region of interest
→	inRadius	float	0.0 - ∞	10.0f	Circle's radius
→	inMinScore	float	0.0 - ∞	20.0f	Minimum matching score
→	inEdgeThreshold	float		10.0f	Minimum accepted edge magnitude
←	outCircle	Conditional<HoughCircle>&			Found circle
🔍	diagGradientMagnitudeImage	Image&			Visualized gradients magnitude of an input image
🔍	diagScoreImage	Image&			Calculated score for each pixel of an input image

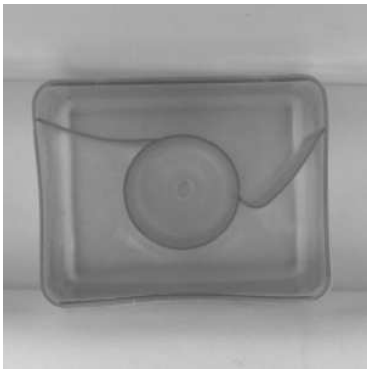
Description

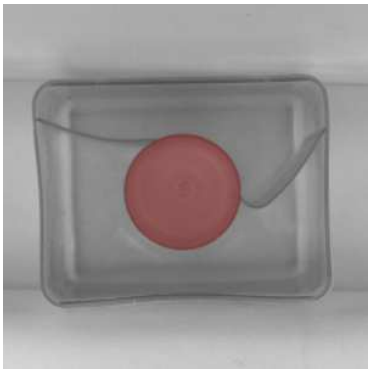
The operation detects circular object of given radius (in pixels) in the **inImage** using the Hough Transform approach.

Hints

- Pass an input image to the **inImage** input.
- Define the expected circle radius on the **inRadius** input.
- Set **inEdgeThreshold** to define the minimum strength of edges that will be taken into account. Verify this value with the **diagGradientMagnitudeImage** output.
- Experimentally set the **inMinScore** value, whose meaning is more or less "the number of pixels voting for a particular object location".

Examples





DetectSingleCircle *performed on the sample image.*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- DetectMultipleCircles – Finds circles of a given radius in the input image using Hough Transform.
- DetectLines – Finds lines in an image using Hough Transform.
- DetectPaths – Finds a specified shape in an image using Hough Transform.
- DetectSegments – Finds segments in an image using Hough Transform.

84. Shape Region Morphology

Table of content:

DilateShapeRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Performs a morphological dilation on a shape region.

Applications: Making the shape region thicker or filling-in small holes within it.

Syntax

```
void avl::DilateShapeRegion
(
 const avl::ShapeRegion& inShapeRegion,
 const float inRadius,
 avl::ShapeRegion& outShapeRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inShapeRegion	const ShapeRegion&			
→	inRadius	const float	0.0 - ∞	1.0f	
←	outShapeRegion	ShapeRegion&			

Errors

List of possible exceptions:

Error type	Description
DomainError	Unable to create Polygon from ShapeRegion in DilateShapeRegion.

ErodeShapeRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Performs a morphological erosion on a shape region.

Applications: Making the shape region thinner or removing small parts.

Syntax

```
void avl::ErodeShapeRegion
(
 const avl::ShapeRegion& inShapeRegion,
 const float inRadius,
 avl::ShapeRegion& outShapeRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inShapeRegion	const ShapeRegion&			
→	inRadius	const float	0.0 - ∞	1.0f	
←	outShapeRegion	ShapeRegion&			

Errors

List of possible exceptions:

Error type	Description
DomainError	Unable to create Polygon from ShapeRegion in ErodeShapeRegion.

85. Image Features

Table of content:

DistanceTransform

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes an image in which the pixel values denote the estimated distances to the nearest bright pixel in the input image.

Applications: This is an initial step of some advanced traditional image analysis algorithms.

Syntax

```
void avl::DistanceTransform
(
 const avl::Image& inImage,
 int inThreshold,
 float inStraightDistance,
 float inDiagonalDistance,
 float inMaxDistance,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inThreshold	int	0 - +∞	128	Minimal brightness above which pixels are called bright
→	inStraightDistance	float	0.0 - ∞	1.0f	Distance between two neighboring in a row or a column pixels
→	inDiagonalDistance	float	0.0 - ∞	1.414f	Distance between two pixels connected by vertices
→	inMaxDistance	float	0.0 - ∞	255.0f	Maximum value of the calculated distance
←	outImage	Image&			Output distance image

GradientImageRidges

Header: AVL.h
Namespace: avl
Module: FoundationPro

Finds ridge pixels in a gradient image.

Syntax

```
void avl::GradientImageRidges
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::GradientMaskOperator::Type inOperator,
 avl::MagnitudeMeasure::Type inMeasure,
 avl::NonMaximaMethod::Type inMethod,
 int inMinValue,
 avl::Region& outRegion,
 avl::Image& outMagnitude = atl::Dummy<avl::Image>()
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inOperator	GradientMaskOperator::Type	Sobel	Gradient operator
→	inMeasure	MagnitudeMeasure::Type	Hypot	Gradient magnitude measure
→	inMethod	NonMaximaMethod::Type	Double	Ridge computation method
→	inMinValue	int	4	Minimum gradient magnitude of a ridge
←	outRegion	Region&		Output region
←	outMagnitude	Image&	Dummy<Image>()	

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in GradientImageRidges.

ImageCenter

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns the geometrical center of an image.

Syntax

```
void avl::ImageCenter
(
 const avl::Image& inImage,
 avl::Point2D& outCenter
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
←	outCenter	Point2D&		

ImageHistogram

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the histogram of the image pixel values.

Syntax

```
void avl::ImageHistogram
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 int inChannelIndex,
 float inDomainBegin,
 const float inBinSize,
 int inBinCount,
 avl::Histogram& outHistogram
)
```

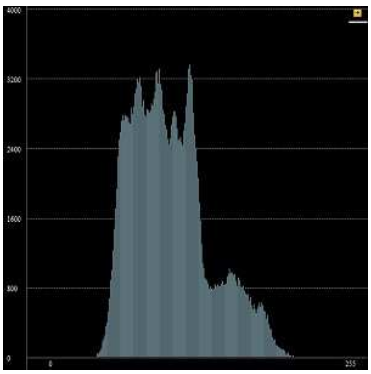
Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inChannelIndex	int	0 - 3		Selects a channel of the input image
→	inDomainBegin	float		0.0f	The lowest value that will be considered in the output histogram
→	inBinSize	const float	0.0 - ∞	1.0f	Width of a single histogram bin
→	inBinCount	int		256	The upper-bound for values that will be considered in the output histogram
←	outHistogram	Histogram&			Output histogram

Description

Histogram in Aurora Vision Studio is a graphical representation of data contained in image. That is, the resulting histogram contains number of pixel values from specified channel with selected **inBinSize** (interval).

Examples



ImageHistogram *performed on the Lena image with* ***inChannelIndex*** *= 0,* ***inBinSize*** *= 1.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in ImageHistogram.
DomainError	Selected bin size equals zero in ImageHistogram.
DomainError	Selected channel index is out of range in ImageHistogram.

See Also

- MakeHistogram – Creates a histogram out of an array of bin values.

ImageLocalMaxima

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Finds image locations characterized by locally maximal pixel values.

Applications: Detection of characteristic points, usually after some image transformations.

Syntax

```
void avl::ImageLocalMaxima
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 bool inConsiderPlateaus,
 atl::Optional<float> inMinValue,
 atl::Optional<float> inMaxValue,
 float inMinDistance,
 atl::Optional<const avl::ImageLocalExtremaVerification&> inMaximaVerification,
 atl::Optional<atl::Array<avl::Extremum2D>&> outLocalMaxima,
 atl::Optional<atl::Array<avl::Region>&> outMaximaRegions = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inConsiderPlateaus	bool			Consider multi-pixel maxima (plateaus) or not
→	inMinValue	Optional<float>		NIL	Minimal value of maximum to be considered
→	inMaxValue	Optional<float>		NIL	Maximal value of maximum to be considered
→	inMinDistance	float	0.0 - ∞		Minimal distance between two found maxima
→	inMaximaVerification	Optional<const ImageLocalExtremaVerification&>		NIL	Maxima verification structure
←	outLocalMaxima	Optional<Array<Extremum2D>&>			Found local maxima
←	outMaximaRegions	Optional<Array<Region>&>		NIL	Regions of local maxima (plateaus and singletons)

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outLocalMaxima**, **outMaximaRegions**.

Read more about Optional Outputs.

Description

The operation finds local maxima on the image **inImage** within specific region **inRoi**.

It returns three arrays of the same length:

- Subpixel precise positions of the maxima
- Values of the maxima
- Regions of equally bright pixels around the maxima

If the given image consists of more than one channel, they are averaged to obtain monochromatic image, so that the values of the extrema are of type float.

There are two types of local extrema:

- Peak/hole: single pixel (strictly) brighter/darker than its neighbourhood (eight pixels). The extremum's position is determined with subpixel precision using gradient method with Sobel gradient. Returned region contains only one pixel.
- Plateau/lowland: a connected set of equally bright pixels (strictly) brighter/darker from their neighbourhood. Returned position for this extremum is the center of mass of the plateau.

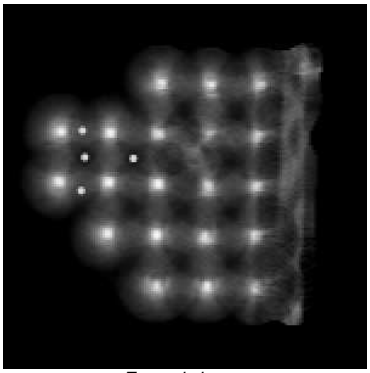
If parameter **inConsiderPlateaus** is set to true, the filter finds both types. Otherwise, it finds only strict maxima.

Parameter **inMinValue** determines the minimal value of the maximum to be considered.

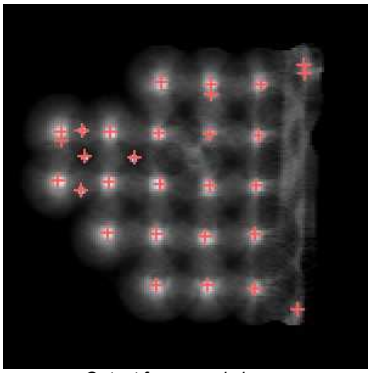
Hints

- If some points are not detected, try decreasing **inMinValue** or setting **inConsiderPlateaus** to **True**.
- Consider adding some Gaussian smoothing before this function.

Examples



Example image



Output for example image

Remarks

If a plateau/lowland is concave, it's center of mass may lie outside it. In such case, the value of the extremum (which is equal to the common value of its pixels) may be different from the value of pixel indicated by position of the extremum.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in ImageLocalMaxima.

See Also

- ImageLocalMinima – Finds image locations characterized by locally minimal pixel values.

ImageLocalMinima

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Finds image locations characterized by locally minimal pixel values.

Applications: Detection of characteristic points, usually after some image transformations.

Syntax

```
void avl::ImageLocalMinima
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 bool inConsiderLowlands,
 atl::Optional<float> inMinValue,
 atl::Optional<float> inMaxValue,
 float inMinDistance,
 atl::Optional<const avl::ImageLocalExtremaVerification&> inMinimaVerification,
 atl::Optional<atl::Array<avl::Extremum2D>&> outLocalMinima,
 atl::Optional<atl::Array<avl::Region>&> outMinimaRegions = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inConsiderLowlands	bool			Consider multi-pixel minima (lowlands) or not
→	inMinValue	Optional<float>		NIL	Minimal value of minimum to be considered
→	inMaxValue	Optional<float>		NIL	Maximal value of minimum to be considered
→	inMinDistance	float	0.0 - ∞		Minimal distance between two found minima
→	inMinimaVerification	Optional<const ImageLocalExtremaVerification&>		NIL	Minima verification structure
←	outLocalMinima	Optional<Array<Extremum2D>&>			Found local minima
←	outMinimaRegions	Optional<Array<Region>&>		NIL	Regions of local minima (plateaus and singletons)

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outLocalMinima**, **outMinimaRegions**.

Read more about Optional Outputs.

Description

The operation finds local minima on the image **inImage** within specific region **inRoi**.

It returns three arrays of the same length:

- Subpixel precise positions of the minima
- Values of the minima
- Regions of equally bright pixels around the minima

If the given image consists of more than one channel, they are averaged to obtain monochromatic image, so that the values of the extrema are of type float.

There are two types of local extrema:

- Peak/hole: single pixel (strictly) brighter/darker than its neighbourhood (eight pixels). The extremum's position is determined with subpixel precision using gradient method with Sobel gradient. Returned region contains only one pixel.
- Plateau/lowland: a connected set of equally bright pixels (strictly) brighter/darker from their neighbourhood. Returned position for this extremum is the center of mass of the plateau.

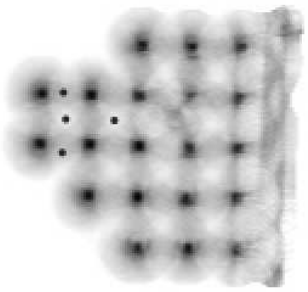
If parameter **inConsiderLowlands** is set to true, the filter finds both types. Otherwise, it finds only strict minima.

Parameter **inMaxValue** determines the maximal value of the minimum to be considered.

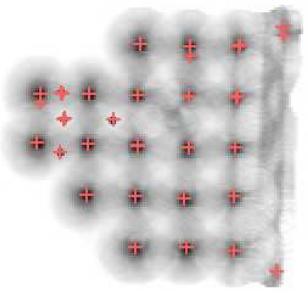
Hints

- If some points are not detected, try decreasing **inMaxValue** or setting **inConsiderLowlands** to **True**.
- Consider adding some Gaussian smoothing before this function.

Examples



Example image



Output for example image

Remarks

If a plateau/lowland is concave, it's center of mass may lie outside it. In such case, the value of the extremum (which is equal to the common value of its pixels) may be different from the value of pixel indicated by position of the extremum.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in ImageLocalMinima.

See Also

- ImageLocalMaxima – Finds image locations characterized by locally maximal pixel values.

ImageMassCenter

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes a point with coordinates equal to image mass center in brightness scale.

Syntax

```
void avl::ImageMassCenter
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::Point2D& outMassCenter
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
←	outMassCenter	Point2D&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Black image on input in ImageMassCenter.
DomainError	Empty region on input in ImageMassCenter.
DomainError	Region exceeds an input image in ImageMassCenter.

ImageMoment

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the selected moment of an image in regular and normalized (divided by sum of pixel values) variant.

Syntax

```
void avl::ImageMoment
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::ImageMomentType::Type inMomentType,
 bool inCentral,
 float& outMoment,
 atl::Optional<float&> outNormMoment = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inMomentType	ImageMomentType::Type		
→	inCentral	bool		
←	outMoment	float&		
←	outNormMoment	Optional<float&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outNormMoment**.

Read more about Optional Outputs.

Description

The operation computes the mathematical features of an image called moments. Those are sums computed as follows:

$$Moment_{0,0}(Image, Roi) = \sum_{p \in Roi} Image(p_x, p_y) \cdot 1$$
$$Moment_{0,1}(Image, Roi) = \sum_{p \in Roi} Image(p_x, p_y) \cdot p_y$$
$$Moment_{1,0}(Image, Roi) = \sum_{p \in Roi} Image(p_x, p_y) \cdot p_x$$
$$Moment_{0,2}(Image, Roi) = \sum_{p \in Roi} Image(p_x, p_y) \cdot p_y^2$$
$$Moment_{1,1}(Image, Roi) = \sum_{p \in Roi} Image(p_x, p_y) \cdot p_x p_y$$
$$Moment_{2,0}(Image, Roi) = \sum_{p \in Roi} Image(p_x, p_y) \cdot p_x^2$$

The summing is conducted over all pixels in **inRoi** region, while p_x and p_y denote, accordingly, x and y coordinate of a pixel.

When **inCentral** parameter is set, the image is shifted before computations, so that its mass center is at location (0,0).

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Black image on input in ImageMoment.
DomainError	Empty region on input in ImageMoment.
DomainError	Region exceeds an input image in ImageMoment.

See Also

- RegionMoment – Computes selected second-order moment of a region in regular and normalized (divided by region area) variant.
- PolygonMoment – Computes the selected second-order moment of a polygon in regular and normalized (divided by polygon area) variant.

ImageOrientation

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the orientation of an image using image moments. The result range is from 0.0 to 180.0.

Syntax

```
void avl::ImageOrientation
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 float& outOrientationAngle
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
←	outOrientationAngle	float&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Black image on input in ImageOrientation.
DomainError	Empty region on input in ImageOrientation.
DomainError	Region exceeds an input image in ImageOrientation.

ImagePixels

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns an array of pixels from the input image.

Syntax

```
void avl::ImagePixels
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Array<avl::Pixel>& outPixels
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
←	outPixels	Array<Pixel>&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in ImagePixels.

ImagePixelValues

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns an array of pixel values from the input image.

Syntax

```
void avl::ImagePixelValues
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 atl::Array<float>& outPixelValues
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
←	outPixelValues	Array<float>&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in ImagePixelValues.

ImageProfileAlongPath

Header: AVL.h
Namespace: avl
Module: FoundationPro

Creates a series of segments across the input path, measures the average pixel intensity on each of the segments, and creates the final profile from those values.

Applications: This is the first step of all 1D Edge Detection operations. Here available for direct use by the user.

Syntax

```
void avl::ImageProfileAlongPath
(
 ScanMapState& ioState,
 const avl::Image& inImage,
 const avl::Path& inScanPath,
 atl::Optional<const avl::CoordinateSystem2D&> inScanPathAlignment,
 int inScanWidth,
 const avl::SamplingParams& inSamplingParams,
 float inSmoothingStdDev,
 avl::AccumulationMode::Type inAccumulationMode,
 atl::Optional<avl::Pixel> inBorderColor,
 avl::Profile& outProfile,
 avl::Path& outPath,
 atl::Optional<avl::Path&> outAlignedScanPath = atl::NIL,
 atl::Array<avl::Path>& diagSamplingPoints,
 float& diagSamplingStep
)
```

Parameters

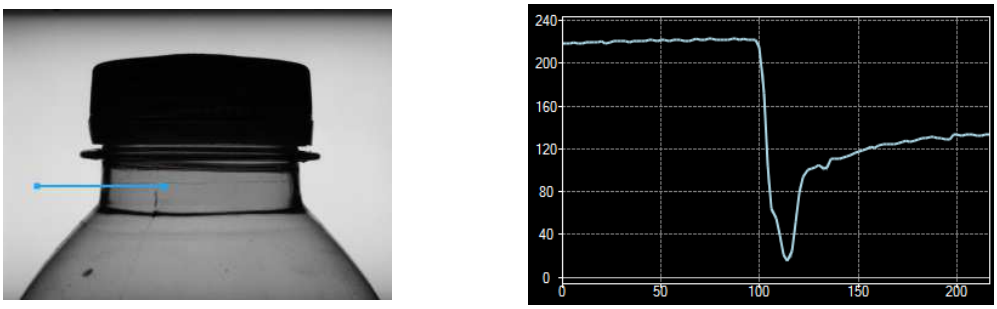
	Name	Type	Range	Default	Description
	ioState	ScanMapState&			Object used to maintain state of the function.
	inImage	const Image&			Input image
	inScanPath	const Path&			Path along which the profile is extracted
	inScanPathAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the scan path to the position of the inspected object
	inScanWidth	int	1 - ∞	5	Width of the scan field in pixels
	inSamplingParams	const SamplingParams&			Parameters controlling the sampling process
	inSmoothingStdDev	float	0.0 - ∞	0.6f	Standard deviation of the gaussian smoothing applied to the extracted profile
	inAccumulationMode	AccumulationMode::Type			Determines how the pixel values are combined
	inBorderColor	Optional<Pixel>		Pixel (X: 0.0f Y: 0.0f Z: 0.0f W: 0.0f)	Color of pixel outside image
	outProfile	Profile&			The resulting profile of the pixel brightness
	outPath	Path&			The path consisting of the points from which the resulting profile is extracted
	outAlignedScanPath	Optional<Path&>		NIL	Input scan path after transformation (in the image coordinates)
	diagSamplingPoints	Array<Path>&			Array of paths each one containing the sampling points that contributed to a single value of the extracted profile
	diagSamplingStep	float&			Used distance between consecutive sampling points on the scan path

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedScanPath**.

Read more about Optional Outputs.

Examples



ImageProfileAlongPath *applied on an image of a bottle (**inScanWidth** = 5)*

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the average (or other statistic) for each image row or column and then merges the obtained results into a profile.

Syntax

```
void avl::ImageProjection
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::ProjectionDirection::Type inProjectionDirection,
 avl::ProjectionMode::Type inProjectionMode,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Monochromatic image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inProjectionDirection	ProjectionDirection::Type		Combine pixel values for image rows (horizontal) or columns (vertical)
→	inProjectionMode	ProjectionMode::Type	Sum	Determines how the pixel values are combined
←	outProfile	Profile&		Output profile

Requirements

For input **inImage** only pixel formats are supported: 1□uint8, 1□int8, 1□uint16, 1□int16, 1□int32, 1□real.

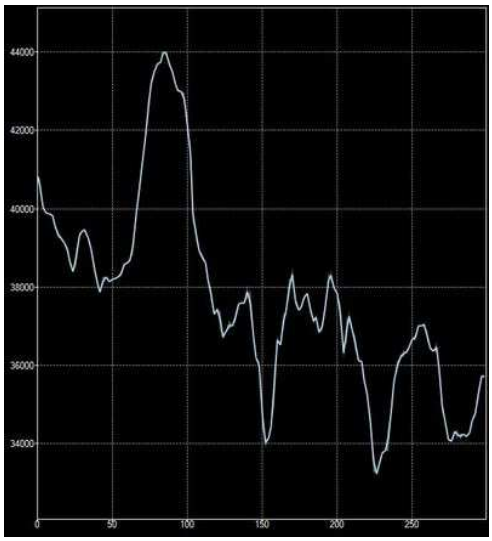
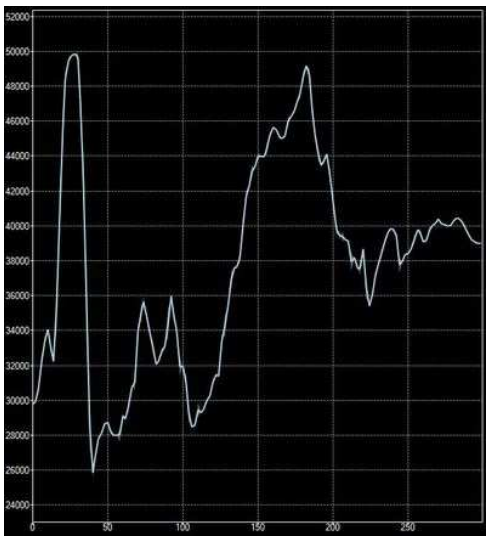
Read more about pixel formats in Image documentation.

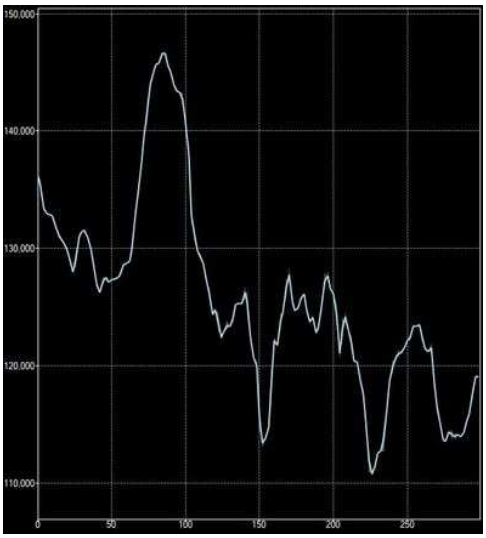
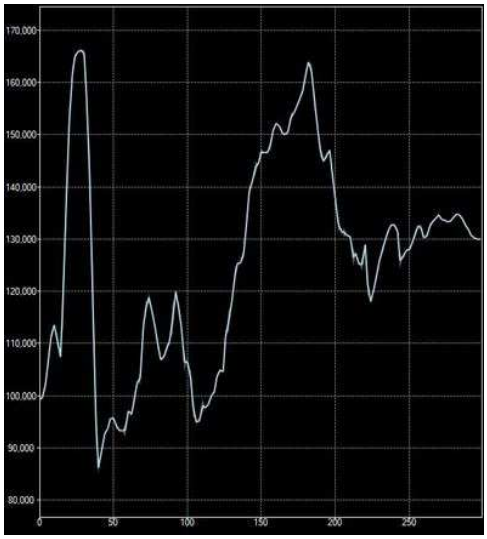
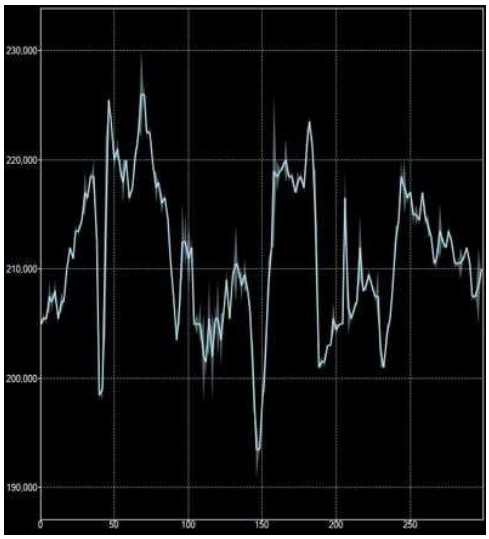
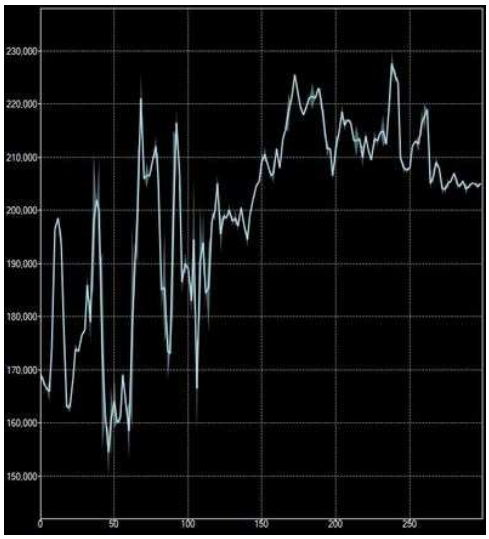
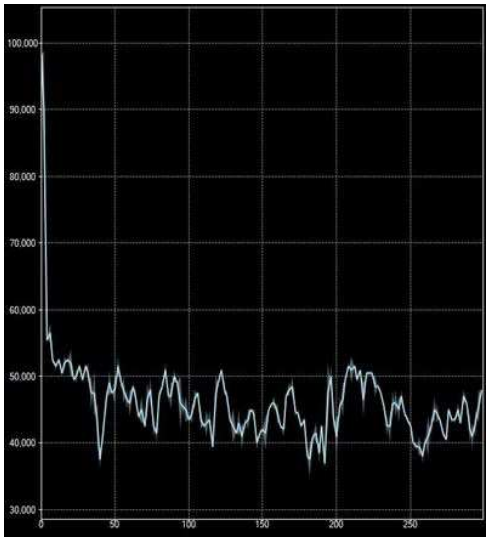
Description

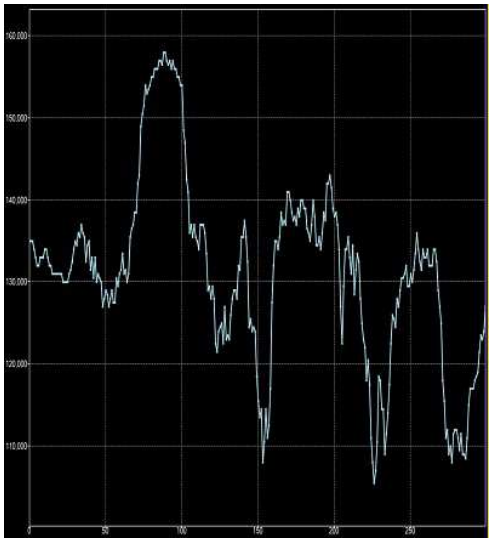
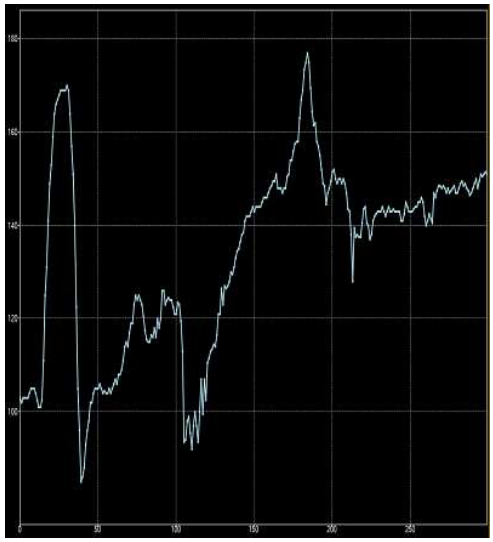
Computes the average (or other statistic) for each image row or column and then merges the obtained results into a profile.

Examples

ImageProjection *performed on a lena sample.*

Mode	Horizontal Direction	Vertical Direction
Sum		

Average		
Maximum		
Minimum		

Median		

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in ImageProjection.
DomainError	Not supported inImage pixel format in ImageProjection. Supported formats: 1xUInt8, 1xInt8, 1xUInt16, 1xInt16, 1xInt32, 1xReal.

ImageRidges

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Finds ridge pixels in an image.

Syntax

```
void avl::ImageRidges
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::Polarity::Type inPolarity,
 atl::Optional<float> inMinValue,
 atl::Optional<float> inMaxValue,
 const float inNoiseLevel,
 avl::Region& outRidges
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inPolarity	Polarity::Type			Specifies the type of ridges to be detected (Bright, Dark or Any)
→	inMinValue	Optional<float>		NIL	Minimal value of pixel to be considered
→	inMaxValue	Optional<float>		NIL	Maximal value of pixel to be considered
→	inNoiseLevel	const float	0.0 - ∞		Defines how much stronger a pixel has to be than its neighbors to be deemed a ridge pixel
←	outRidges	Region&			Region of ridges

Description

The operation finds region of pixels being local ridges on the image **inImage** within specific region **inRoi**. For a pixel to be deemed a ridge pixel it is necessary to be stronger than both its neighbor pixels in at least one from four main directions (horizontal, vertical and two diagonals) by at least **inNoiseLevel**. Furthermore, the pixel value must be between **inMinValue** and **inMaxValue**.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in ImageRidges.

See Also

- ImageLocalMinima – Finds image locations characterized by locally minimal pixel values.
- ImageLocalMaxima – Finds image locations characterized by locally maximal pixel values.

ImageSharpness

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Returns a value dependent on sharpness of the image. The value is bigger for sharper images.

Applications: Designed for manual or automatic camera focus adjustment.

Syntax

```
void avl::ImageSharpness
(
 const avl::Image& inImage,
 atl::Optional<const avl::Box&> inRoi,
 float& outSharpness
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Box&>	NIL	Range of pixels to be processed
←	outSharpness	float&		

Examples

Two images having different sharpness level – 38.326 for the left one, 6.059 for the right one.

Errors

List of possible exceptions:

Error type	Description
DomainError	inRoi exceeds image dimensions in ImageSharpness.

ImageStandardDeviation

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Calculates standard deviation of image pixel values.

Syntax

```
void avl::ImageStandardDeviation
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 float& outStandardDeviation
)
```

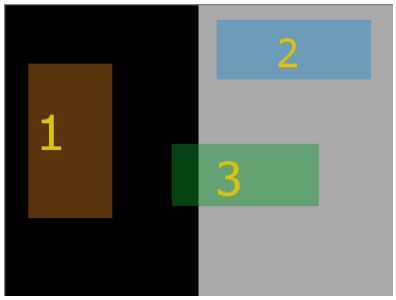
Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image.
→	inRoi	Optional<const Region&>	NIL	Region of Interest.
←	outStandardDeviation	float&		Calculated standard deviation.

Description

Computes how input image pixel values are spread out.

Examples



ImageStandardDeviation *yielded 0.0 for regions 1 and 2, but 93.3 for 3.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in ImageStandardDeviation.

86. Image Drawing

Table of content:

DrawArc

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Draws an arc on an image.

Syntax

```
void avl::DrawArc
(
 avl::Image& ioImage,
 const avl::Arc2D& inArc,
 atl::Optional<const avl::CoordinateSystem2D&> inArcAlignment,
 const avl::Pixel& inColor,
 const avl::DrawingStyle& inDrawingStyle
)
```

Parameters

	Name	Type	Default	Description
	ioImage	Image&		
	inArc	const Arc2D&		
	inArcAlignment	Optional<const CoordinateSystem2D&>	NIL	
	inColor	const Pixel&		
	inDrawingStyle	const DrawingStyle&		

DrawBox

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Draws a box on an image.

Syntax

```
void avl::DrawBox
(
 avl::Image& ioImage,
 const avl::Box& inBox,
 const avl::Pixel& inColor,
 const float inOpacity
)
```

Parameters

	Name	Type	Range	Default	Description
	ioImage	Image&			
	inBox	const Box&			
	inColor	const Pixel&			
	inOpacity	const float	0.0 - 1.0		

DrawCircle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Draws a circle on an image.

Syntax

```
void avl::DrawCircle
(
 avl::Image& ioImage,
 const avl::Circle2D& inCircle,
 atl::Optional<const avl::CoordinateSystem2D&> inCircleAlignment,
 const avl::Pixel& inColor,
 const avl::DrawingStyle& inDrawingStyle
)
```

Parameters

	Name	Type	Default	Description
	ioImage	Image&		
	inCircle	const Circle2D&		
	inCircleAlignment	Optional<const CoordinateSystem2D&>	NIL	
	inColor	const Pixel&		
	inDrawingStyle	const DrawingStyle&		

Description

The operation draws a single of circle on the **ioImage**. Circle may exceed the image dimensions - those will be drawn partially or not at all, but the filter execution will succeed.

See Also

- DrawPoints_MultiColor – Draws points on an image with multiple colors.
- DrawPoints_SingleColor – Draws points on an image with a single color.

DrawCoordinateSystem

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Draws a pair of arrows representing a coordinate system on an image.

Syntax

```
void avl::DrawCoordinateSystem
(
 avl::Image& ioImage,
 const avl::CoordinateSystem2D& inCoordinateSystem,
 atl::Optional<const avl::CoordinateSystem2D&> inCoordinateSystemAlignment,
 const avl::Pixel& inColor,
 const avl::DrawingStyle& inDrawingStyle,
 const float inArrowSize,
 const float inPixelScale
)
```

Parameters

	Name	Type	Range	Default	Description
	ioImage	Image&			
	inCoordinateSystem	const CoordinateSystem2D&			
	inCoordinateSystemAlignment	Optional<const CoordinateSystem2D&>		NIL	
	inColor	const Pixel&			
	inDrawingStyle	const DrawingStyle&			
	inArrowSize	const float	0.0 - ∞	5.0f	
	inPixelScale	const float	0.0 - ∞	20.0f	How many pixels long will be axis of a coordinate system with unit scale

DrawCrosshair

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Draws a crosshair on an image.

Syntax

```
void avl::DrawCrosshair
(
 avl::Image& ioImage,
 const avl::Location& inLocation,
 atl::Optional<const avl::CoordinateSystem2D&> inLocationAlignment,
 const avl::Pixel& inColor,
 const avl::CrosshairShape::Type inCrosshairShape,
 const avl::DrawingStyle& inDrawingStyle
)
```

Parameters

	Name	Type	Default	Description
	ioImage	Image&		
	inLocation	const Location&		
	inLocationAlignment	Optional<const CoordinateSystem2D&>	NIL	
	inColor	const Pixel&		
	inCrosshairShape	const CrosshairShape::Type		
	inDrawingStyle	const DrawingStyle&		

DrawDimensionLine

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Draws a dimension line on an image.

Syntax

```
void avl::DrawDimensionLine
(
 avl::Image& ioImage,
 const avl::Segment2D& inSegment,
 atl::Optional<const avl::CoordinateSystem2D&> inSegmentAlignment,
 const avl::Pixel& inColor,
 const avl::DrawingStyle& inDrawingStyle,
 float inHeadSize,
 float inScale,
 avl::MetricUnit::Type inMetricUnit,
 const int inFractionalDigitCount,
 float inTextSize = 12
)
```

Parameters

	Name	Type	Range	Default	Description
	ioImage	Image&			
	inSegment	const Segment2D&			
	inSegmentAlignment	Optional<const CoordinateSystem2D&>		NIL	
	inColor	const Pixel&			
	inDrawingStyle	const DrawingStyle&			
	inHeadSize	float	0.0 - ∞	5.0f	
	inScale	float	0.0 - ∞	1.0f	px / mm
	inMetricUnit	MetricUnit::Type		MetricUnit::Millimeters	
	inFractionalDigitCount	const int	0 - 100	3	How many characters the fractional part of the number should have
	inTextSize	float	10.0 - ∞	12.0f	

DrawEllipse

Header: AVL.h
Namespace: avl
Module: FoundationLite

Draws an ellipse on an image.

Syntax

```
void avl::DrawEllipse
(
 avl::Image& ioImage,
 const avl::Rectangle2D& inEllipse,
 atl::Optional<const avl::CoordinateSystem2D&> inEllipseAlignment,
 const avl::Pixel& inColor,
 const avl::DrawingStyle& inDrawingStyle
)
```

Parameters

	Name	Type	Default	Description
	ioImage	Image&		
	inEllipse	const Rectangle2D&		
	inEllipseAlignment	Optional<const CoordinateSystem2D&>	NIL	
	inColor	const Pixel&		
	inDrawingStyle	const DrawingStyle&		

DrawGridImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Draws an image as a tile on an image considered to be a grid of tiles.

Syntax

```
void avl::DrawGridImage
(
 avl::Image& ioImage,
 const avl::Image& inTileImage,
 int inColumnIndex,
 int inRowIndex
)
```

Parameters

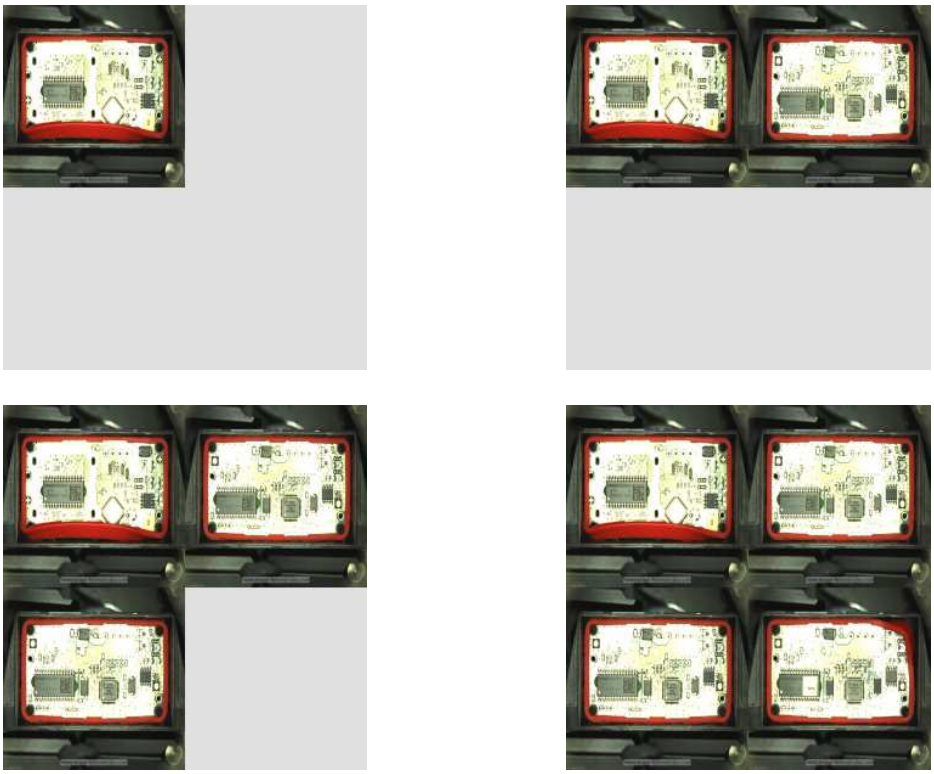
	Name	Type	Default	Description
	ioImage	Image&		
	inTileImage	const Image&		Image to be pasted to the grid
	inColumnIndex	int		Column index in the grid
	inRowIndex	int		Row index in the grid

Description

The operation supports drawing image grids composed of equally sized images. The filters draws single **inImage** on the **inImage** at the location being the selected multiple of **inTileImage** dimensions.

Parameters pair **inColumnIndex** and **inRowIndex** set to (0, 0) draws **inTileImage** at the location of (0,0) pixels of the **inImage**.

Examples



*Four consecutive instances of the **DrawGridImage** used to plot an image grid. Each of the filters receives different **inImage** and different pair of the **inColumnIndex**, **inRowIndex** parameters - accordingly: (0,0) (0,1) (1,0) (1,1).*

See Also

- DrawImage – Draws an image on another one.
- JoinImages – Creates a single image by glueing together the two input images in horizontal or vertical direction.
- ComposeImages – Takes pixels from the first image within the specified region and from the other one elsewhere.

DrawImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Draws an image on another one.

Syntax

```
void avl::DrawImage
(
 avl::Image& ioImage,
 const avl::Image& inDrawnImage,
 const avl::Location& inLocation
)
```

Parameters

	Name	Type	Default	Description
	ioImage	Image&		
	inDrawnImage	const Image&		Image to be pasted on ioImage
	inLocation	const Location&		Location at which the image will be pasted

Description

The operation draws the **inDrawnImage** on the **inImage** so that the upper-left corner of the **inDrawnImage** is aligned at the **inLocation** of the **inImage**.

Examples

*The **DrawImage** used to draw an image on another, empty image.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Incompatible image depths in DrawImage.

See Also

- DrawGridImage – Draws an image as a tile on an image considered to be a grid of tiles.

DrawLine

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Draws a line on an image.

Syntax

```
void avl::DrawLine
(
 avl::Image& ioImage,
 const avl::Line2D& inLine,
 atl::Optional<const avl::CoordinateSystem2D&> inLineAlignment,
 const avl::Pixel& inColor,
 const avl::DrawingStyle& inDrawingStyle
)
```

Parameters

	Name	Type	Default	Description
	ioImage	Image&		
	inLine	const Line2D&		
	inLineAlignment	Optional<const CoordinateSystem2D&>	NIL	
	inColor	const Pixel&		
	inDrawingStyle	const DrawingStyle&		

DrawPath

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Draws a path on an image.

Syntax

```
void avl::DrawPath
(
 avl::Image& ioImage,
 const avl::Path& inPath,
 atl::Optional<const avl::CoordinateSystem2D&> inPathAlignment,
 const avl::Pixel& inColor,
 const avl::DrawingStyle& inDrawingStyle
)
```

Parameters

	Name	Type	Default	Description
	ioImage	Image&		
	inPath	const Path&		Input path
	inPathAlignment	Optional<const CoordinateSystem2D&>	NIL	
	inColor	const Pixel&		
	inDrawingStyle	const DrawingStyle&		

DrawPoint

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Draws a point on an image.

Syntax

```
void avl::DrawPoint
(
 avl::Image& ioImage,
 const avl::Point2D& inPoint,
 atl::Optional<const avl::CoordinateSystem2D&> inPointAlignment,
 const avl::Pixel& inColor,
 const avl::DrawingStyle& inDrawingStyle
)
```

Parameters

	Name	Type	Default	Description
	ioImage	Image&		
	inPoint	const Point2D&		
	inPointAlignment	Optional<const CoordinateSystem2D&>	NIL	
	inColor	const Pixel&		
	inDrawingStyle	const DrawingStyle&		

DrawRectangle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Draws a rectangle on an image.

Syntax

```
void avl::DrawRectangle
(
 avl::Image& ioImage,
 const avl::Rectangle2D& inRectangle,
 atl::Optional<const avl::CoordinateSystem2D&> inRectangleAlignment,
 const avl::Pixel& inColor,
 const avl::DrawingStyle& inDrawingStyle
)
```

Parameters

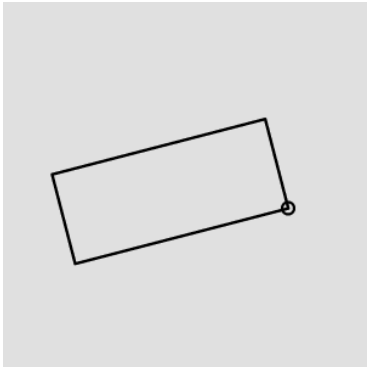
	Name	Type	Default	Description
	ioImage	Image&		
	inRectangle	const Rectangle2D&		
	inRectangleAlignment	Optional<const CoordinateSystem2D&>	NIL	
	inColor	const Pixel&		
	inDrawingStyle	const DrawingStyle&		

Description

Filter draws rectangles on an input image.

If an rectangle dimensions exceeds image size it will not be drawn or will be drawn partially.

To indicate the Rectangle orientation set **inDrawingStyle.PointShape** and **inDrawingStyle.Size**.



*Rectangle drawn with set orientation indicator **inDrawingStyle.PointShape** = **Circle** and **inDrawingStyle.Size** = **10**.*

Hints

- Define **inRectangle**. This will be the primitives to be drawn.
- Define **inColor**. Please note, that on an N-channel image only first N components of the color will be used.
- Set **inDrawingStyle** to control quality, opacity, thickness, filling, point shapes and sizes.

See Also

- DrawLine – Draws a line on an image.
- DrawPath – Draws a path on an image.
- DrawCircle – Draws a circle on an image.
- DrawPoint – Draws a point on an image.

DrawRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Draws a region on an image.

Syntax

```
void avl::DrawRegion
(
 avl::Image& ioImage,
 const avl::Region& inRegion,
 atl::Optional<const avl::CoordinateSystem2D&> inRegionAlignment,
 const avl::Pixel& inColor,
 const float inOpacity
)
```

Parameters

	Name	Type	Range	Default	Description
	ioImage	Image&			
	inRegion	const Region&			Input region
	inRegionAlignment	Optional<const CoordinateSystem2D&>		NIL	
	inColor	const Pixel&			
	inOpacity	const float	0.0 - 1.0		

DrawSegment

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Draws a segment on an image.

Syntax

```
void avl::DrawSegment
(
 avl::Image& ioImage,
 const avl::Segment2D& inSegment,
 atl::Optional<const avl::CoordinateSystem2D&> inSegmentAlignment,
 const avl::Pixel& inColor,
 const avl::DrawingStyle& inDrawingStyle,
 const avl::MarkerType::Type inMarkerType,
 const float inMarkerSize
)
```

Parameters

	Name	Type	Range	Default	Description
	ioImage	Image&			
	inSegment	const Segment2D&			
	inSegmentAlignment	Optional<const CoordinateSystem2D&>		NIL	
	inColor	const Pixel&			
	inDrawingStyle	const DrawingStyle&			
	inMarkerType	const MarkerType::Type		Arrow	
	inMarkerSize	const float	0.0 - ∞	5.0f	

DrawShapeRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Draws a shape region on an image.

Syntax

```
void avl::DrawShapeRegion
(
 avl::Image& ioImage,
 const avl::ShapeRegion& inRoi,
 atl::Optional<const avl::CoordinateSystem2D&> inRegionAlignment,
 const avl::Pixel& inColor,
 const avl::DrawingStyle& inDrawingStyle,
 const bool inForceRgb
)
```

Parameters

	Name	Type	Default	Description
	ioImage	Image&		
	inRoi	const ShapeRegion&		Range of pixels to be processed
	inRegionAlignment	Optional<const CoordinateSystem2D&>	NIL	
	inColor	const Pixel&		
	inDrawingStyle	const DrawingStyle&	DrawingStyle (DrawingMode: HighQuality Opacity: 1.0f Thickness: 1.0f Filled: False PointShape: Nil PointSize: 1.0f)	
	inForceRgb	const bool	True	Filter will convert monochromatic image to RGB if needed

DrawString

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Draws a string on an image.

Syntax

```
void avl::DrawString
(
 avl::Image& ioImage,
 const atl::String& inString,
 const avl::Location& inLocation,
 atl::Optional<const avl::CoordinateSystem2D&> inLocationAlignment,
 const avl::Anchor2D::Type inLocationAnchor,
 const avl::Pixel& inColor,
 const avl::DrawingStyle& inDrawingStyle,
 float inSize,
 float inOrientationAngle,
 atl::Optional<const avl::Pixel&> inBackgroundColor
)
```

Parameters

	Name	Type	Default	Description
	ioImage	Image&		
	inString	const String&		
	inLocation	const Location&		
	inLocationAlignment	Optional<const CoordinateSystem2D&>	NIL	
	inLocationAnchor	const Anchor2D::Type	MiddleCenter	
	inColor	const Pixel&		
	inDrawingStyle	const DrawingStyle&		
	inSize	float	12.0f	Height of a character
	inOrientationAngle	float	0.0f	
	inBackgroundColor	Optional<const Pixel&>	NIL	

Description

The operation draws a string on the **ioImage** aligning the **inLocationAnchor** location of the text at the **inLocation** of the **ioImage**. The height of the font is fixed and equals **16** pixels.

Usage of DrawString with various settings.

See Also

- DrawLine – Draws a line on an image.
- DrawPath – Draws a path on an image.
- DrawCircle – Draws a circle on an image.
- DrawRectangle – Draws a rectangle on an image.

DrawVector

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Draws a vector at an associated initial point on an image.

Syntax

```
void avl::DrawVector
(
 avl::Image& ioImage,
 const avl::Vector2D& inVector,
 const avl::Point2D& inInitialPoint,
 atl::Optional<const avl::CoordinateSystem2D&> inVectorAlignment,
 const avl::Pixel& inColor,
 const avl::DrawingStyle& inDrawingStyle,
 const float inHeadSize
)
```

Parameters

	Name	Type	Default	Description
	ioImage	Image&		
	inVector	const Vector2D&		
	inInitialPoint	const Point2D&		
	inVectorAlignment	Optional<const CoordinateSystem2D&>	NIL	
	inColor	const Pixel&		
	inDrawingStyle	const DrawingStyle&		
	inHeadSize	const float	5.0f	

VisualizeHeatmap

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Colorizes heat-map and blends it with background image.

Syntax

```
void avl::VisualizeHeatmap
(
 atl::Optional<const avl::Image&> inImage,
 const avl::Heatmap& inHeatmap,
 const avl::ColorPalette::Type& inPalette,
 atl::Optional<float> inPercentFuzziness,
 bool inForceMono,
 bool inThreshold,
 avl::Image& outImage,
 avl::Image& diagPalette
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	Optional<const Image&>		NIL	Input image
→	inHeatmap	const Heatmap&			Confidence of defect at each pixel
→	inPalette	const ColorPalette::Type&		GreenYellowRed	
→	inPercentFuzziness	Optional<float>	0.0 - ∞	0.0f	
→	inForceMono	bool			Converts input image to monochromatic
→	inThreshold	bool			Remove all values lower than value of minimal threshold.
←	outImage	Image&			Output image
🔍	diagPalette	Image&			Used palette preview

Requirements

For input **inImage** only pixel formats are supported: 1□uint8, 1□uint16, 2□uint8, 2□uint16, 3□uint8, 3□uint16, 4□uint8, 4□uint16.

For input **inHeatmap** only pixel formats are supported: 1□uint8.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported inHeatmap pixel format in VisualizeHeatmap. Supported formats: 1xUInt8.
DomainError	Not supported inImage pixel format in VisualizeHeatmap. Supported formats: 1xUInt8, 1xUInt16, 2xUInt8, 2xUInt16, 3xUInt8, 3xUInt16, 4xUInt8, 4xUInt16.

VisualizeHeatmap_Old

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Colorizes a heat-map and blends it with a background image.

Syntax

```
void avl::VisualizeHeatmap_Old
(
 avl::ColorizeImageState& ioState,
 const avl::Image& inImage,
 const avl::Image& inHeatmap,
 const avl::ColorPalette::Type& inPalette,
 int inThreshold,
 atl::Optional<int> inFuzziness,
 atl::Optional<int> inMinValue,
 atl::Optional<int> inMaxValue,
 bool inForceMono,
 avl::Image& outImage,
 avl::Image& diagPalette
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	ColorizeImageState&			Object used to maintain state of the function.
	inImage	const Image&			Input image
	inHeatmap	const Image&			Confidence of defect at each pixel
	inPalette	const ColorPalette::Type&		BlackYellowRed	
	inThreshold	int	0 - 255	64	Minimum defect confidence for choosing more of heat-map color than of input image color
	inFuzziness	Optional<int>	0 - ∞	0	Confidence distance from inThreshold within which heat-map colors and input image colors are linearly interpolated; Auto = INF
	inMinValue	Optional<int>		0	
	inMaxValue	Optional<int>		255	
	inForceMono	bool			Converts input image to monochromatic
	outImage	Image&			Output image
	diagPalette	Image&			Used palette preview

Requirements

For input **inImage** only pixel formats are supported: 1□uint8, 1□uint16, 2□uint8, 2□uint16, 3□uint8, 3□uint16, 4□uint8, 4□uint16.

For input **inHeatmap** only pixel formats are supported: 1□uint8.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Image sizes are not equal in VisualizeHeatmap_Old.
DomainError	Input heatmap must have 1xUInt8 format in VisualizeHeatmap_Old.
DomainError	Input image must have UInt8 or UInt16 pixel format in VisualizeHeatmap_Old.
DomainError	Not supported inHeatmap pixel format in VisualizeHeatmap_Old. Supported formats: 1xUInt8.
DomainError	Not supported inImage pixel format in VisualizeHeatmap_Old. Supported formats: 1xUInt8, 1xUInt16, 2xUInt8, 2xUInt16, 3xUInt8, 3xUInt16, 4xUInt8, 4xUInt16.

87. Image Segmentation

Table of content:

ExtractBlobs_Color

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Segments an image into blobs by color-based thresholding.

Syntax

```
void avl::ExtractBlobs_Color
(
 const avl::Image& inImage,
 atl::Optional<const avl::ShapeRegion&> inRoi,
 atl::Optional<const avl::CoordinateSystem2D&> inRoiAlignment,
 const avl::ColorThresholdParams& inThresholdParams,
 const avl::PreSplitProcessingParams& inProcessingParams,
 const avl::SplittingParams& inSplittingParams,
 atl::Array<avl::Region>& outBlobs,
 avl::Region& outBaseRegion
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Image from which blobs are extracted
→	inRoi	Optional<const ShapeRegion&>	NIL	Range of pixels to be processed
→	inRoiAlignment	Optional<const CoordinateSystem2D&>	NIL	Adjusts the region
→	inThresholdParams	const ColorThresholdParams&		Parameters for color-based thresholding
→	inProcessingParams	const PreSplitProcessingParams&		Parameters for postprocessing of the extracted region
→	inSplittingParams	const SplittingParams&		Parameters for splitting region into blobs
←	outBlobs	Array<Region>&		Blobs extracted from the input image
←	outBaseRegion	Region&		Region of pixels right after thresholding

Requirements

For input **inImage** only pixel formats are supported: 1□uint8, 2□uint8, 3□uint8, 4□uint8.

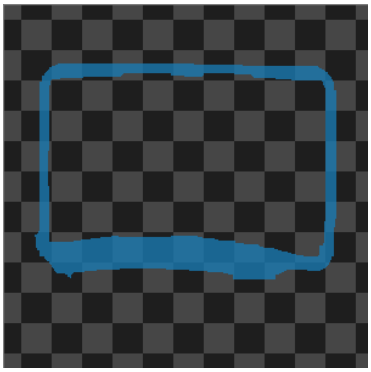
Read more about pixel formats in Image documentation.

Description

This filter can be used to quickly segment an image using color-based thresholding. It performs a series of operations on **inImage**:

- image is thresholded by ThresholdToRegion_Color using parameters from **inThresholdParams**
- resulting region is opened and then closed by OpenRegion and CloseRegion, also holes are removed by FillRegionHoles using parameters from **inProcessingParams**
- resulting region is split into blobs by SplitRegionIntoBlobs using parameters from **inSplittingParams**

Examples



ExtractBlobs_Color *performed on the sample image with* ***inThresholdParams.RgbColor*** *= (189, 36, 25),* ***inThresholdParams.MaxDifference*** *= 40.0 and* ***inProcessingParams.ClosingRadius*** *= 5.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in ExtractBlobs_Color.
DomainError	Not supported inImage pixel format in ExtractBlobs_Color. Supported formats: 1xUInt8, 2xUInt8, 3xUInt8, 4xUInt8.

See Also

- ExtractBlobs_Intensity – Segments an image into blobs by thresholding using a single value.
- ExtractBlobs_Dynamic – Segments an image into blobs by dynamic thresholding.

ExtractBlobs_Dynamic

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Segments an image into blobs by dynamic thresholding.

Syntax

```
void avl::ExtractBlobs_Dynamic
(
 const avl::Image& inImage,
 atl::Optional<const avl::ShapeRegion&> inRoi,
 atl::Optional<const avl::CoordinateSystem2D&> inRoiAlignment,
 const avl::DynamicThresholdParams& inThresholdParams,
 const avl::PreSplitProcessingParams& inProcessingParams,
 const avl::SplittingParams& inSplittingParams,
 atl::Array<avl::Region>& outBlobs,
 avl::Region& outBaseRegion
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Image from which blobs are extracted
→	inRoi	Optional<const ShapeRegion&>	NIL	Range of pixels to be processed
→	inRoiAlignment	Optional<const CoordinateSystem2D&>	NIL	Adjusts the region
→	inThresholdParams	const DynamicThresholdParams&		Parameters for dynamic thresholding
→	inProcessingParams	const PreSplitProcessingParams&		Parameters for postprocessing of the extracted region
→	inSplittingParams	const SplittingParams&		Parameters for splitting region into blobs
←	outBlobs	Array<Region>&		Blobs extracted from the input image
←	outBaseRegion	Region&		Region of pixels right after thresholding

Description

This filter can be used to quickly segment an image with uneven illumination. It performs a series of operations on **inImage**:

- image is thresholded by ThresholdToRegion_Dynamic using parameters from **inThresholdParams**
- resulting region is opened and then closed by OpenRegion and CloseRegion, also holes are removed by FillRegionHoles using parameters from **inProcessingParams**
- resulting region is split into blobs by SplitRegionIntoBlobs using parameters from **inSplittingParams**

Examples

ExtractBlobs_Dynamic *performed on the sample image with* ***inThresholdParams.Polarity*** *= Dark,* ***inThresholdParams.Threshold*** *= 5.0, and* ***inProcessingParams.ClosingRadius*** *= 3*

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in ExtractBlobs_Dynamic.

See Also

- ExtractBlobs_Intensity – Segments an image into blobs by thresholding using a single value.
- ExtractBlobs_Color – Segments an image into blobs by color-based thresholding.

ExtractBlobs_Intensity

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Segments an image into blobs by thresholding using a single value.

Syntax

```
void avl::ExtractBlobs_Intensity
(
 const avl::Image& inImage,
 atl::Optional<const avl::ShapeRegion&> inRoi,
 atl::Optional<const avl::CoordinateSystem2D&> inRoiAlignment,
 const avl::IntensityThresholdParams& inThresholdParams,
 const avl::PreSplitProcessingParams& inProcessingParams,
 const avl::SplittingParams& inSplittingParams,
 atl::Array<avl::Region>& outBlobs,
 avl::Region& outBaseRegion
)
```

Parameters

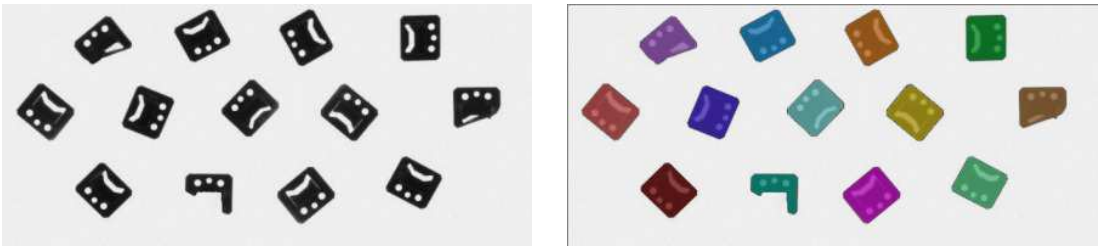
	Name	Type	Default	Description
→	inImage	const Image&		Image from which blobs are extracted
→	inRoi	Optional<const ShapeRegion&>	NIL	Range of pixels to be processed
→	inRoiAlignment	Optional<const CoordinateSystem2D&>	NIL	Adjusts the region
→	inThresholdParams	const IntensityThresholdParams&		Parameters for thresholding an image
→	inProcessingParams	const PreSplitProcessingParams&		Parameters for postprocessing of the extracted region
→	inSplittingParams	const SplittingParams&		Parameters for splitting region into blobs
←	outBlobs	Array<Region>&		Blobs extracted from the input image
←	outBaseRegion	Region&		Region of pixels right after thresholding

Description

This filter can be used to quickly segment an image. It performs a series of operations on **inImage**:

- image is thresholded by ThresholdToRegion using parameters from **inThresholdParams**
- resulting region is opened and then closed by OpenRegion and CloseRegion, also holes are removed by FillRegionHoles using parameters from **inProcessingParams**
- resulting region is split into blobs by SplitRegionIntoBlobs using parameters from **inSplittingParams**

Examples



ExtractBlobs_Intensity *performed on the sample image with* ***inThresholdParams.Polarity*** *= Dark,* ***inThresholdParams.Threshold*** *= 84.0, and* ***inProcessingParams.ClosingRadius*** *= 10.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in ExtractBlobs_Intensity.

See Also

- ExtractBlobs_Dynamic – Segments an image into blobs by dynamic thresholding.
- ExtractBlobs_Color – Segments an image into blobs by color-based thresholding.

ExtractBlobs_Range

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Segments an image into blobs by thresholding using a range of values.

Syntax

```
void avl::ExtractBlobs_Range
(
 const avl::Image& inImage,
 atl::Optional<const avl::ShapeRegion&> inRoi,
 atl::Optional<const avl::CoordinateSystem2D&> inRoiAlignment,
 const avl::RangeThresholdParams& inThresholdParams,
 const avl::PreSplitProcessingParams& inProcessingParams,
 const avl::SplittingParams& inSplittingParams,
 atl::Array<avl::Region>& outBlobs,
 avl::Region& outBaseRegion
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Image from which blobs are extracted
→	inRoi	Optional<const ShapeRegion&>	NIL	Range of pixels to be processed
→	inRoiAlignment	Optional<const CoordinateSystem2D&>	NIL	Adjusts the region
→	inThresholdParams	const RangeThresholdParams&		Parameters for thresholding an image
→	inProcessingParams	const PreSplitProcessingParams&		Parameters for postprocessing of the extracted region
→	inSplittingParams	const SplittingParams&		Parameters for splitting region into blobs
←	outBlobs	Array<Region>&		Blobs extracted from the input image
←	outBaseRegion	Region&		Region of pixels right after thresholding

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in ExtractBlobs_Range.

FindMaxStableExtremalRegions

Header: AVL.h
Namespace: avl
Module: FoundationPro

Segments an image by binarizing it with many different thresholds and by looking which blobs appear "stable".

Applications: Most frequently used for finding correspondence points between two images.

Syntax

```
void avl::FindMaxStableExtremalRegions
(
 const avl::Image& inImage,
 const int inDelta,
 const int inMinArea,
 const int inMaxArea,
 const float inMaxVariation,
 const float inMinDiversity,
 avl::RegionConnectivity::Type inConnectivity,
 atl::Array<avl::Region>& outRegions
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inDelta	const int	1 - 255	30	Area variance is calculated against ancestor with color difference of delta
→	inMinArea	const int	0 - ∞	50	Minimum area of stable region
→	inMaxArea	const int	0 - ∞	2000	Maximum area of stable region
→	inMaxVariation	const float	0.0 - ∞	0.1f	Maximum area variance with containing larger region specified by delta parameter, for region to be considered as stable
→	inMinDiversity	const float	0.0 - ∞	2.0f	Minimum area diversity that region must have in order to be stable when compared to stable regions within it
→	inConnectivity	RegionConnectivity::Type		EightDirections	
←	outRegions	Array<Region>&			

Requirements

For input **inImage** only pixel formats are supported: 1□uint8.

Read more about pixel formats in Image documentation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported inImage pixel format in FindMaxStableExtremalRegions. Supported formats: 1xUInt8.

ImageWatersheds

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes dark basins which are separated by at least inThreshold height watershed.

Syntax

```
void avl::ImageWatersheds
(
 const avl::Image& inImage,
 const int inThreshold,
 atl::Array<avl::Region>& outBasins
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inThreshold	const int	0 - + ∞	10	Input minimum separating watershed height
←	outBasins	Array<Region>&			Output dark basins found

SegmentImage_Color

Header: AVL.h
Namespace: avl
Module: FoundationPro

Segments an image basing on distance to model colors.

Applications: Detection of objects of undefined shape, but characterized by uniform color and good contrast to the background.

Syntax

```
void avl::SegmentImage_Color
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 const atl::Array<avl::Pixel>& inReferenceColors,
 const float inMaxDifference,
 atl::Optional<const atl::Array<float>&> inDifferenceMultipliers,
 float inChromaAmount,
 bool inForceDisjointRegions,
 atl::Array<avl::Region>& outRegions
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inReferenceColors	const Array<Pixel>&			Colors to compare pixels to
→	inMaxDifference	const float	0.0 - ∞	5.0f	Maximal difference between pixel and reference color to be accepted
→	inDifferenceMultipliers	Optional<const Array<float>&>		NIL	Scales for maximum differences for each color
→	inChromaAmount	float	0.0 - 1.0	0.7f	Proportion of chromatic information in distance computation
→	inForceDisjointRegions	bool			Force output regions to be disjoint
←	outRegions	Array<Region>&			Regions of pixels closest to colors

Description

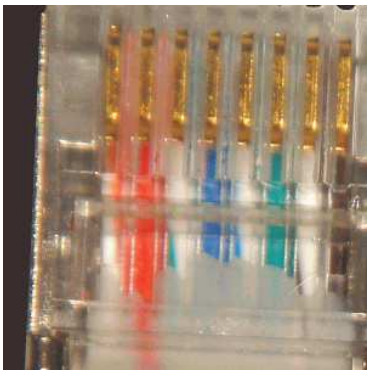
The operation finds regions of similar colors in RGB color space within **inImage**.

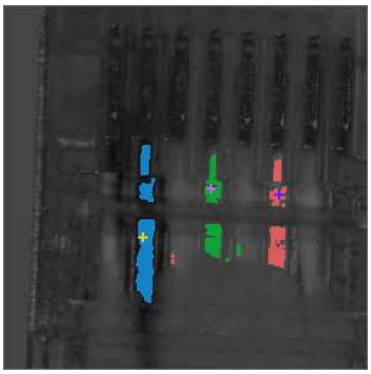
First, for each pixel of the input image its distance to each color in **inReferenceColors** is computed (see ColorDistanceImage function).

Then, for each reference color **c**, the function computes region of pixels that are closer to **c** than its corresponding threshold. In case **inDifferenceMultipliers** is **Nil**, threshold is equal to **inMaxDifference**. If **inDifferenceMultipliers** is not **Nil**, corresponding threshold is the product of **inMaxDifference** and the element of **inDifferenceMultipliers** that corresponds to **c**.

If flag **inForceDisjointRegions** is set to True, the regions are forced to be disjoint. In case a pixel has distance less than threshold for more than one color, distances are divided by thresholds and the minimum of ratios is taken. In particular, if the function works in one-threshold mode, the operation described above becomes simply minimum of distances.

Examples





SegmentImage_Color *run with colors taken from three points with* ***inChromaAmount*** *= 0.7 and* ***inMaxDifference*** *= 15*

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Inconsistent array sizes in SegmentImage_Color.
DomainError	Reference color array size too big in SegmentImage_Color.

See Also

- ColorDistance – Compares two pixels using chromatic and non-chromatic information. Assumes RGB color space.
- ColorDistanceImage – Compares each pixel with the specified color using chromatic and non-chromatic information.

SegmentImage_Edges

Header: AVL.h
Namespace: avl
Module: FoundationPro

Segments an image into blobs using image edges as their borders.

Applications: Detection of objects of undefined shape, but characterized by good contrast to the background and fairly uniform internal brightness.

Syntax

```
void avl::SegmentImage_Edges
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 const int inFrameSize,
 float inStdDev,
 float inEdgeThreshold,
 float inEdgeHysteresis,
 const float inMaxJoiningDistance,
 const int inMinArea,
 atl::Optional<int> inMaxArea,
 bool inComputeNestingLevels,
 const int inEdgeClosing,
 atl::Array<avl::Region>& outBlobs,
 atl::Optional<atl::Array<int>&> outNestingLevels = atl::NIL,
 avl::Region& diagEdgeRegion = atl::Dummy<avl::Region>()
)
```

Parameters

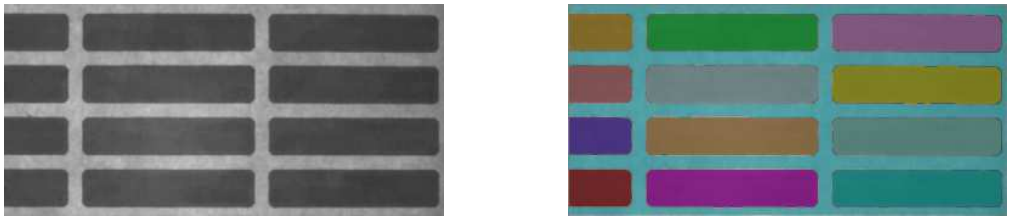
	Name	Type	Range	Default	Description
➡	inImage	const Image&			Image from which blobs are extracted
➡	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
➡	inFrameSize	const int	0 - ∞	1	How many pixels from the region border are excluded from the results
➡	inStdDev	float	0.0 - ∞	2.0f	Amount of smoothing used by the edge filter
➡	inEdgeThreshold	float	0.0 - ∞	15.0f	Sufficient edge strength; edges of that strength will always be detected
➡	inEdgeHysteresis	float	0.0 - ∞	5.0f	Value by which the edge threshold is decreased for edge points neighboring with sufficiently strong edges
➡	inMaxJoiningDistance	const float	0.0 - ∞	0.0f	Maximal distance between edges that can be joined
➡	inMinArea	const int	0 - ∞	50	Minimal area of a detected blob
➡	inMaxArea	Optional<int>	0 - ∞	NIL	Maximal area of a detected blob
➡	inComputeNestingLevels	bool			Flag indicating whether nesting levels should be computed
➡	inEdgeClosing	const int	0 - ∞	1	Radius of enclosing small holes in the detected blobs
⬅	outBlobs	Array<Region>&			Blobs extracted from the input image
⬅	outNestingLevels	Optional<Array<int>&>		NIL	Nesting level of each extracted blob
🔍	diagEdgeRegion	Region&			Region of the found edges

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outNestingLevels**.

Read more about Optional Outputs.

Examples



SegmentImage_Edges *performed on an image of labels; default parameters.*

SegmentImage_Gray

Header: AVL.h
Namespace: avl
Module: FoundationPro

Segments an image into blobs examining differences between neighbouring pixels values.

Applications: Detection of objects of undefined shape, but characterized by uniform brightness and good contrast to the background.

Syntax

```
void avl::SegmentImage_Gray
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 const float inMaxDifference,
 const avl::BlobsDifferenceMeasure::Type inDifferenceMeasure,
 const int inMinArea,
 atl::Optional<int> inMaxArea,
 atl::Array<avl::Region>& outBlobs
)
```

Parameters

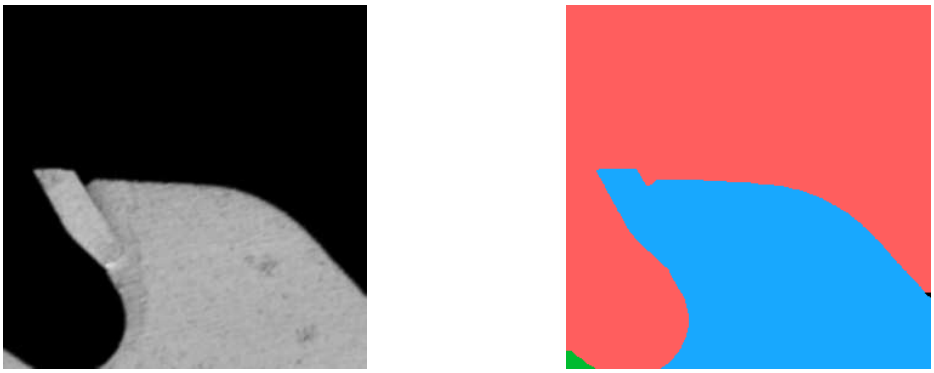
	Name	Type	Range	Default	Description
→	inImage	const Image&			Image from which blobs are extracted
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inMaxDifference	const float	0.0 - ∞	5.0f	Maximal difference between two neighbouring blobs to be merged
→	inDifferenceMeasure	const BlobsDifferenceMeasure::Type		Neighbour	Measure of blobs difference
→	inMinArea	const int	0 - ∞	50	Minimal area of a blob
→	inMaxArea	Optional<int>	0 - ∞	NIL	Maximal area of a blob
←	outBlobs	Array<Region>&			Blobs extracted from the input image

Description

The filter segments the **inImage** image into blobs of adjacent pixels which gray values do not differ too much. The classification is different depending on the **inDifferenceMeasure** method:

- if **Mean** is selected, pixel is considered to belong to an adjacent blob when its value differs by at most **inMaxDifference** from the mean value of this blob's pixels
- if **Neighbour** is selected, two adjacent pixels are considered to belong to the same blob when their values differ by at most **inMaxDifference**

Examples



SegmentImage_Gray *performed on the sample image with* ***inMaxDifference*** *= 55,* ***inDifferenceMeasure*** *= Mean and* ***inMinArea*** *= 50.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported difference measure in SegmentImage_Gray.

See Also

- SegmentImage_Edges – Segments an image into blobs using image edges as their borders.
- SegmentImage_Watersheds – Computes dark or bright watershed basins of an image.

SegmentImage_Gray_Linear

Header: AVL.h
Namespace: avl
Module: FoundationPro

Segments an image into blobs examining differences between pixels values.

Applications: Detection of objects of undefined shape, but characterized by uniform brightness and good contrast to the background.

Syntax

```
void avl::SegmentImage_Gray_Linear
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::BlobMergingMethod::Type inMergingMethod,
 avl::RegionConnectivity::Type inConnectivity,
 int inMaxDifference,
 int inHysteresis,
 int inPassCount,
 bool inDirectional,
 int inMinArea,
 atl::Optional<int> inMaxArea,
 atl::Array<avl::Region>& outBlobs
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inImage	const Image&			Input image
➡	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
➡	inMergingMethod	BlobMergingMethod::Type		Neighbor	Defines a criterion by which two pixels can be merged into one blob
➡	inConnectivity	RegionConnectivity::Type			Defines if pixels can be merged only horizontally and vertically, or also diagonally
➡	inMaxDifference	int	0 - ∞	5	Maximal tonal difference that allows two pixels to be merged into one blob
➡	inHysteresis	int			Total change of MaxDifference during all image passes
➡	inPassCount	int	1 - 5		Number of iterations in which the entire image is scanned for pixels that can be merged
➡	inDirectional	bool			When 'True' the filter can segment results of GradientDirAndPresenceImage
➡	inMinArea	int	0 - ∞	20	Minimal area of a blob that may be accepted
➡	inMaxArea	Optional<int>		NIL	Minimal area of a blob that may be accepted
⬅	outBlobs	Array<Region>&			

Requirements

For input **inImage** only pixel formats are supported: 1□uint8, 1□int8, 1□uint16, 1□int16, 1□int32, 1□real, 3□uint8, 3□int8, 3□uint16, 3□int16, 3□int32, 3□real.

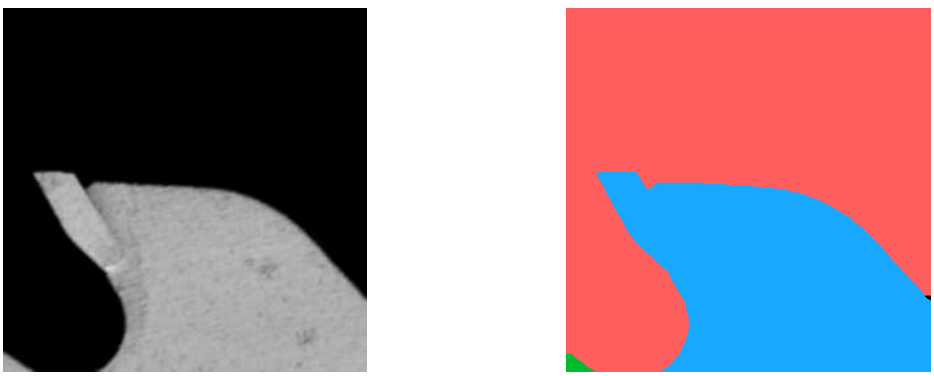
Read more about pixel formats in Image documentation.

Description

The filter segments the **inImage** image into blobs of pixels which gray values do not differ too much. The classification is different depending on the **inMergingMethod** input:

- **Neighbor** – two adjacent pixels are considered to belong to the same blob when their values differ by at most **inMaxDifference**
- **NeighborVerified** – as above, but an additional verification is performed to avoid undersegmentation
- **Mean** – pixel is considered to belong to an adjacent blob when its value differs by at most **inMaxDifference** from the mean value of this blob's pixels
- **MeanSorted** – not implemented in this filter

Examples



SegmentImage_Gray_Linear *performed on the sample image with* ***inMaxDifference*** *= 55,* ***inMergingMethod*** *= MeanLinear and* ***inMinArea*** *= 50.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Directional method only supports single-channel images.
DomainError	Directional method only supports UInt8 images.
DomainError	Two- and four-channel images are not supported in SegmentImage_Gray.
DomainError	Not supported inImage pixel format in SegmentImage_Gray_Linear. Supported formats: 1xUInt8, 1xInt8, 1xUInt16, 1xInt16, 1xInt32, 1xReal, 3xUInt8, 3xInt8, 3xUInt16, 3xInt16, 3xInt32, 3xReal.

See Also

- SegmentImage_Gray_Tiled – Segments an image into blobs examining differences between pixels values, first pass is tiled.
- SegmentImage_Edges – Segments an image into blobs using image edges as their borders.
- SegmentImage_Watersheds – Computes dark or bright watershed basins of an image.

SegmentImage_Gray_Tiled

Header: AVL.h
Namespace: avl
Module: FoundationPro

Segments an image into blobs examining differences between pixels values, first pass is tiled.

Applications: Detection of objects of undefined shape, but characterized by uniform brightness and good contrast to the background.

Syntax

```
void avl::SegmentImage_Gray_Tiled
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::BlobMergingMethod::Type inMergingMethod,
 avl::RegionConnectivity::Type inConnectivity,
 int inMaxDifference,
 int inHysteresis,
 bool inTonalDrift,
 bool inDirectional,
 int inMinArea,
 atl::Optional<int> inMaxArea,
 atl::Array<avl::Region>& outBlobs
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inMergingMethod	BlobMergingMethod::Type		Neighbor	Defines a criterion by which two pixels can be merged into one blob
→	inConnectivity	RegionConnectivity::Type			
→	inMaxDifference	int	0 - ∞	5	Maximal tonal difference that allows two pixels to be merged into one blob
→	inHysteresis	int			Change of MaxDifference between first and second iteration
→	inTonalDrift	bool		True	When 'True' blobs can be merged even if the illumination is not even
→	inDirectional	bool			When 'True' the filter can segment results of GradientDirAndPresenceImage
→	inMinArea	int	0 - ∞	20	Minimal area of a blob that may be accepted
→	inMaxArea	Optional<int>		NIL	Minimal area of a blob that may be accepted
←	outBlobs	Array<Region>&			

Requirements

For input **inImage** only pixel formats are supported: 1□uint8, 3□uint8.

Read more about pixel formats in Image documentation.

Description

The filter segments the **inImage** image in two phases. In the first step segmentation is done within tiles, while in the second step blobs are merged between consecutive tiles.

Errors

List of possible exceptions:

Error type	Description
DomainError	Directional method only supports single-channel images.
DomainError	Not supported inImage pixel format in SegmentImage_Gray_Tiled. Supported formats: 1xUInt8, 3xUInt8.

See Also

- SegmentImage_Gray_Linear – Segments an image into blobs examining differences between pixels values.
- SegmentImage_Edges – Segments an image into blobs using image edges as their borders.
- SegmentImage_Watersheds – Computes dark or bright watershed basins of an image.

SegmentImage_Watersheds

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes dark or bright watershed basins of an image.

Applications: A classic algorithm, useful for segmentation of ball-shaped objects like plant seeds, cells or fruits.

Syntax

```
void avl::SegmentImage_Watersheds
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 const atl::Array<avl::Region>& inMarkers,
 avl::Polarity::Type inBasinsPolarity,
 atl::Array<avl::Region>& outBasins
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inMarkers	const Array<Region>&		Local minima markers
→	inBasinsPolarity	Polarity::Type		Whether to look for bright or dark basins
←	outBasins	Array<Region>&		

Requirements

For input **inImage** only pixel formats are supported: uint8.

Read more about pixel formats in Image documentation.

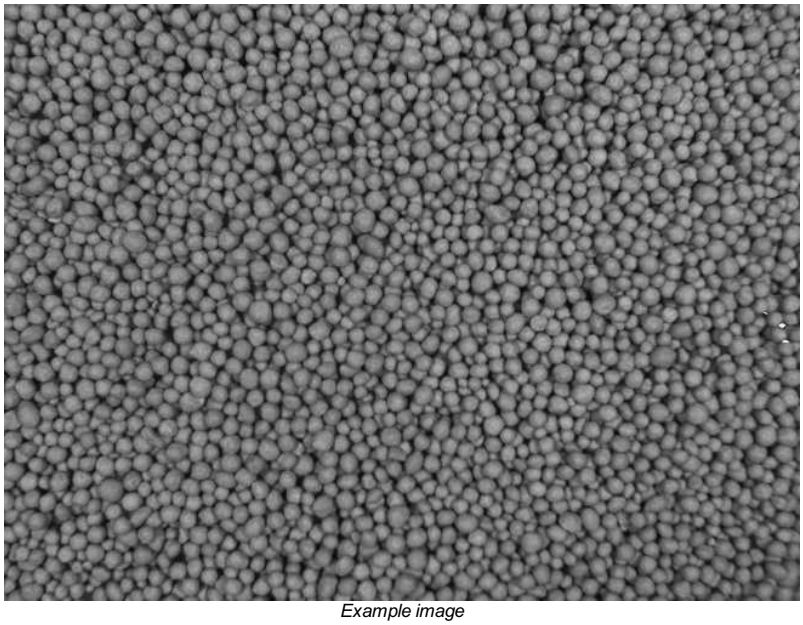
Description

This filter segments the image into basins based on a set of markers (**inMarkers**) and a flooding watersheds implementation for grayscale images.

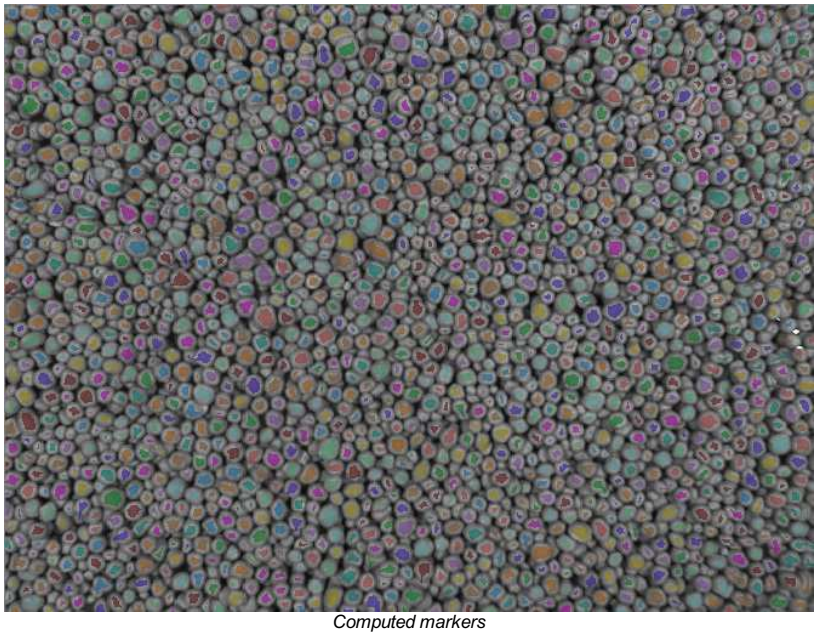
By default a darker pixel is considered lower but that can be changes using the **inBasinsPolarity** parameter.

Markers have to be computed beforehand. One way to do that is to detect the ridges of the image and threshold it into a single region, Pass the resulting region into the **DistanceTransform** filter. Then threshold the distance image and split it into blobs.

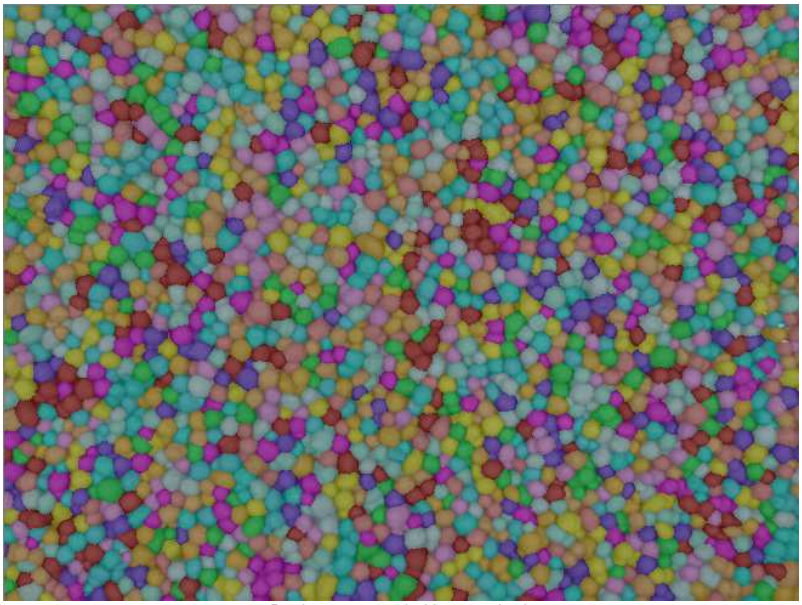
Examples



Example image



Computed markers



Basins computed with watersheds

Errors

List of possible exceptions:

Error type	Description
DomainError	No markers present in SegmentImage_Watersheds.
DomainError	Not supported inImage pixel format in SegmentImage_Watersheds. Supported formats: UInt8.

See Also

- DetectRidges_AsRegion – Extracts a pixel-precise region of bright or dark thin lines.
- ThresholdToRegion – Creates a region containing image pixels with values within the specified range.
- DistanceTransform – Computes an image in which the pixel values denote the estimated distances to the nearest bright pixel in the input image.

88. Region Global Transforms

Table of content:

FillRegionHoles

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Adds pixels to the input region so that it contains no holes.

Syntax

```
void avl::FillRegionHoles
(
 const avl::Region& inRegion,
 avl::RegionConnectivity::Type inConnectivity,
 int inMinHoleArea,
 atl::Optional<int> inMaxHoleArea,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inRegion	const Region&			Input region
➡	inConnectivity	RegionConnectivity::Type			Type of connectivity used for the region foreground
➡	inMinHoleArea	int	0 - ∞		Minimal area of a hole to be filled
➡	inMaxHoleArea	Optional<int>	0 - ∞	NIL	Maximal area of a hole to be filled
⬅	outRegion	Region&			Output region

Description

The operation extends a region to contain all pixels inside any of the region holes. Holes of a region are those connected areas of pixels **not** belonging to the region, that do not touch the boundary of the region frame.

Examples

FillRegionHoles *run on a sample region.*

Remarks

This filter is mostly used in **Blob Analysis Technique** please refer to our **Machine Vision Guide - Blob Analysis** article.

See Also

- RegionNumberOfHoles – Computes the number of holes in a region.

RegionBoundaries

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Removes interior pixels from a region.

Syntax

```
void avl::RegionBoundaries
(
 const avl::Region& inRegion,
 avl::RegionConnectivity::Type inConnectivity,
 avl::Region& outRegion
)
```

Parameters

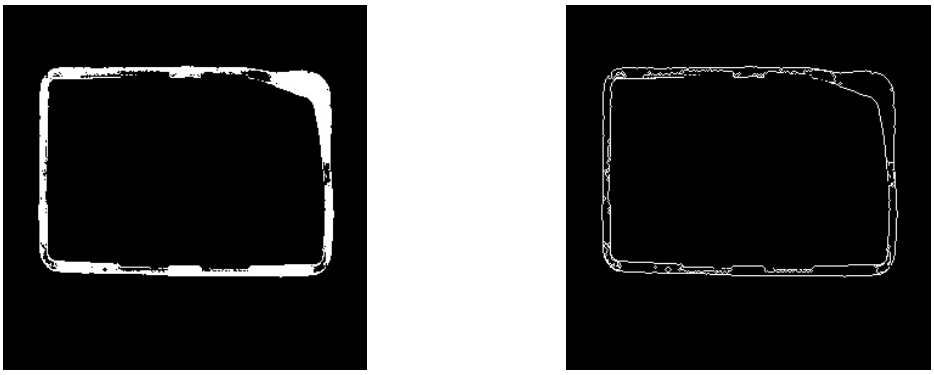
	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inConnectivity	RegionConnectivity::Type		Type of connectivity used for the region foreground
←	outRegion	Region&		Output region

Description

The operation removes all internal pixels from the region (thus leaving only the boundary pixels). The definition of *internal pixel* depends on the **inConnectivity** value:

- If **inConnectivity** is set to **Four directions**, internal pixels are those having neighbours in all four major directions (up, down, left, right) also contained in the region
- If **inConnectivity** is set to **Eight directions**, internal pixels are those having neighbours in all eight directions (four major + four diagonal) also contained in the region

Examples



RegionBoundaries *run on a sample region.*

See Also

- RegionContours – Computes an array of closed paths corresponding to the contours of the input region.
- RegionInteriors – Removes boundary pixels from a region.

RegionConvexHull

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the smallest convex region containing the input region.

Syntax

```
void avl::RegionConvexHull
(
 const avl::Region& inRegion,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outRegion	Region&		Output region

Description

The operation computes the smallest of all convex region containing the given one.

Examples

***RegionConvexHull** run on a sample region.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Input and output regions are not distinct in RegionConvexHull.

See Also

- RegionConvexity – Computes the area of a region divided by area of its convex hull.
- PathConvexHull – Computes the smallest convex shape that contains the given path.

RegionConvexHull_AsPath

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the smallest convex polygon containing the input region.

Syntax

```
void avl::RegionConvexHull_AsPath
(
 const avl::Region& inRegion,
 avl::Path& outConvexHull
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outConvexHull	Path&		

RegionInteriors

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Removes boundary pixels from a region.

Syntax

```
void avl::RegionInteriors
(
 const avl::Region& inRegion,
 avl::RegionConnectivity::Type inConnectivity,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inConnectivity	RegionConnectivity::Type		Type of connectivity used for the region foreground
←	outRegion	Region&		Output region

Description

The operation removes all boundary pixels from the region (thus leaving only the interior pixels). The definition of *boundary pixel* depends on the **inConnectivity** value:

- If **inConnectivity** is set to **Four directions**, boundary pixels are those having **at least one** of the neighbours in four major directions (up, down, left, right) **not** contained in the region
- If **inConnectivity** is set to **Eight directions**, boundary pixels are those having **at least one** of the neighbours in all eight directions (four major + four diagonal) **not** contained in the region

Examples

RegionInteriors *run on a sample region.*

Remarks

This filter is mostly used in **Blob Analysis Technique** please refer to our **Machine Vision Guide - Blob Analysis** article.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported connectivity type in RegionInteriors.

See Also

- RegionBoundaries – Removes interior pixels from a region.

RegionOuterBoundaries

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Returns the top, right, bottom and left boundaries of the input region.

Syntax

```
void avl::RegionOuterBoundaries
(
 const avl::Region& inRegion,
 atl::Array< avl::Location >& outTopBoundary,
 atl::Array< avl::Location >& outRightBoundary,
 atl::Array< avl::Location >& outBottomBoundary,
 atl::Array< avl::Location >& outLeftBoundary
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outTopBoundary	Array< Location >&		
←	outRightBoundary	Array< Location >&		
←	outBottomBoundary	Array< Location >&		
←	outLeftBoundary	Array< Location >&		

RemoveBordersFromRegion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Removes from a region the pixels that lie very close to the region's frame.

Applications: Assures that the resulting region is not "touching" the borders.

Syntax

```
void avl::RemoveBordersFromRegion
(
 const avl::Region& inRegion,
 int inMarginX,
 int inMarginY,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inRegion	const Region&			Input region
→	inMarginX	int	0 - ∞		Horizontal margin from the region border
→	inMarginY	int	0 - ∞		Vertical margin from the region border
←	outRegion	Region&			Output region

RemoveRegionBlobs

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Splits a region into blobs, removes blobs not fulfilling the specified condition, and merges the rest back into one region.

Syntax

```
void avl::RemoveRegionBlobs
(
 const avl::Region& inRegion,
 avl::RegionConnectivity::Type inConnectivity,
 avl::RegionFeature::Type inFeature,
 atl::Optional<float> inMinimum,
 atl::Optional<float> inMaximum,
 bool inRemoveBoundaryBlobs,
 avl::Region& outRegion
)
```

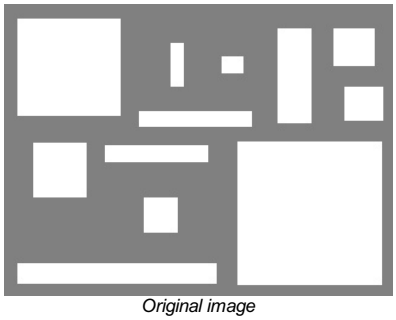
Parameters

	Name	Type	Range	Default	Description
➡	inRegion	const Region&			Input region
➡	inConnectivity	RegionConnectivity::Type			Type of connectivity used for the region foreground
➡	inFeature	RegionFeature::Type			Region feature value to be computed
➡	inMinimum	Optional<float>	0.0 - ∞	1.0f	Minimal value of the considered feature
➡	inMaximum	Optional<float>	0.0 - ∞	NIL	Maximal value of the considered feature
➡	inRemoveBoundaryBlobs	bool		False	Flag indicating whether the blobs on border of the input region should be removed or not
⬅	outRegion	Region&			Output region

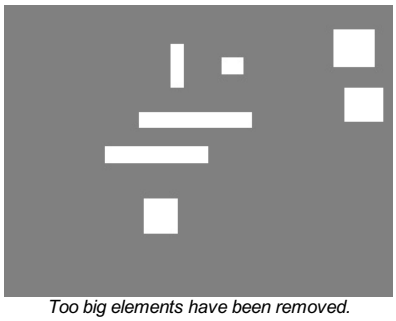
Description

This filter removes regions, which don't fulfill specific conditions given in **inMinimum** and **inMaximum** inputs. Both inputs are optional, which means that you don't have to set fixed limitations of your parameter's range. Classification can be based on different region features, for example area, convexity, mass center.

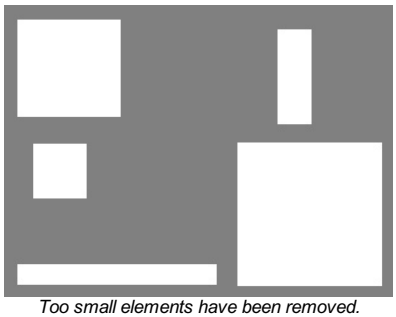
Examples



Original image



Too big elements have been removed.



Too small elements have been removed.

See Also

- ClassifyRegions – Splits an array of regions according to the selected feature and range.
- SplitRegionIntoBlobs – Splits a region into an array of regions corresponding to its connected components.
- RegionUnion_OfArray – Computes a region containing all the pixels that any of the input regions contains.

RemoveRegionBoundaryBlobs

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Removes all blobs intersecting a boundary of a given region.

Syntax

```
void avl::RemoveRegionBoundaryBlobs
(
 const avl::Region& inRegion,
 const avl::Region& inBorderRegion,
 avl::RegionConnectivity::Type inConnectivity,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inBorderRegion	const Region&		Region which boundary is used.
→	inConnectivity	RegionConnectivity::Type		Type of connectivity used for the region foreground
←	outRegion	Region&		Output region

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported connectivity type in RemoveRegionBoundaryBlobs.

SplitRegionIntoBlobs

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Splits a region into an array of regions corresponding to its connected components.

Applications: Segmentation of a region into individual objects when the objects do not touch each other.

Syntax

```
void avl::SplitRegionIntoBlobs
(
 const avl::Region& inRegion,
 avl::RegionConnectivity::Type inConnectivity,
 const int inMinBlobArea,
 atl::Optional<int> inMaxBlobArea,
 bool inRemoveBoundaryBlobs,
 atl::Array<avl::Region>& outBlobs
)
```

Parameters

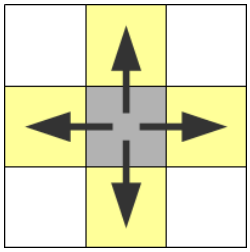
	Name	Type	Range	Default	Description
→	inRegion	const Region&			Input region
→	inConnectivity	RegionConnectivity::Type			Type of connectivity used for the region foreground
→	inMinBlobArea	const int	0 - ∞	1	Minimal area of a resulting blob
→	inMaxBlobArea	Optional<int>	0 - ∞	NIL	Maximal area of a resulting blob
→	inRemoveBoundaryBlobs	bool		False	Flag indicating whether the blobs on border of the input region should be removed or not
←	outBlobs	Array<Region>&			

Description

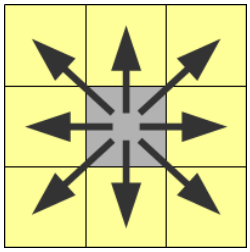
The operation computes an array of connected regions corresponding to the connected components of the input region pixels. Each region in the resulting array will have dimensions equal to those of the input region.

Resulting array will contain only regions which have area in range defined by inputs: **inMinBlobArea** and **inMaxBlobArea**.

Images below shows neighbor of the gray pixel in different **inConnectivity** selection.



inConnectivity.FourDirections



inConnectivity.EightDirections

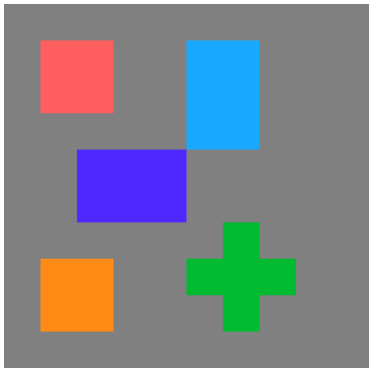
Hints

- In most cases this function works without any parametrization. Just pass a single region to the **inRegion** input.
- To reject small blobs, which are often caused by noise, increase **inMinBlobArea**.
- To reject big blobs, which sometimes appear due to some background pattern, define **inMaxBlobArea**.
- Set **inRemoveBoundaryBlobs** to ignore blobs of partially visible objects, touching the edges of the image.

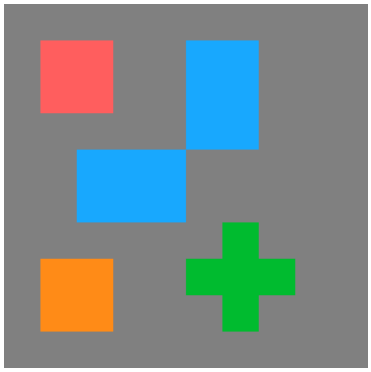
Examples

The image below show an input 10x10 pixels region.

Images below shows result of **SplitRegionIntoBlobs** filter with different **inConnectivity**.

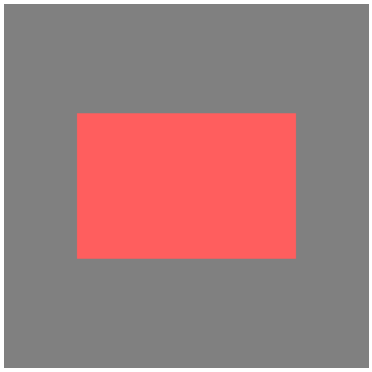


inConnectivity.FourDirections

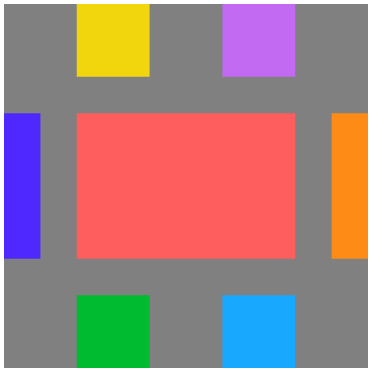


inConnectivity.EightDirections

Images below shows result of **SplitRegionIntoBlobs** filter with different **inRemoveBoundaryBlobs**.



inRemoveBoundaryBlobs = True



inRemoveBoundaryBlobs = False

Remarks

When **inRemoveBoundaryBlobs** is set to **True** all filters near the region boundary will be removed. The region boundary is described by region frame.

This filter is mostly used in **Blob Analysis Technique** please refer to our **Machine Vision Guide - Blob Analysis** article.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- SplitRegionIntoComponents – Splits a region into an array of regions. Operates by merging blobs in accordance to the inMaxDistance parameter.
- SplitRegionIntoMultipleCharacters – Splits the input region into an array of regions corresponding to individual characters.
- CropRegion – Creates a region from a rectangular fragment of another one.

SplitRegionIntoComponents

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Splits a region into an array of regions. Operates by merging blobs in accordance to the inMaxDistance parameter.

Syntax

```
void avl::SplitRegionIntoComponents
(
 const avl::Region& inRegion,
 atl::Optional<int> inMaxDistance,
 float inDistanceBalance,
 atl::Optional<float> inMaxJointDiameter,
 atl::Optional<int> inMaxJointWidth,
 atl::Optional<int> inMaxJointHeight,
 const int inMinComponentArea,
 atl::Optional<int> inMaxComponentArea,
 bool inRemoveBoundaryBlobs,
 atl::Array<avl::Region>& outComponents
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inRegion	const Region&			Input region
➡	inMaxDistance	Optional<int>	0 - ∞	5	
➡	inDistanceBalance	float	-1.0 - 1.0		Defines how much important the distance between regions in x coordinate is according to distance in y coordinate
➡	inMaxJointDiameter	Optional<float>	0.0 - ∞	NIL	
➡	inMaxJointWidth	Optional<int>	0 - ∞	NIL	
➡	inMaxJointHeight	Optional<int>	0 - ∞	NIL	
➡	inMinComponentArea	const int	0 - ∞	1	Minimal area of a resulting component
➡	inMaxComponentArea	Optional<int>	0 - ∞	NIL	Maximal area of a resulting component
➡	inRemoveBoundaryBlobs	bool		False	Flag indicating whether the blobs on border of the input region should be removed or not
⬅	outComponents	Array<Region>&			

Description

The filter splits the input region into blobs and iteratively joins some of them into bigger components. Only blobs that are distant from each other by at most **inMaxDistance** can be joined. The joining order is determined based on modified distance between two blobs, so the closest ones are joined first. This modified distance between two blobs is computed as follows:

1. The shortest segment connecting two blobs is computed.
2. The segment is scaled by

   $$0.5 \cdot (1 + inDistanceBalance)$$

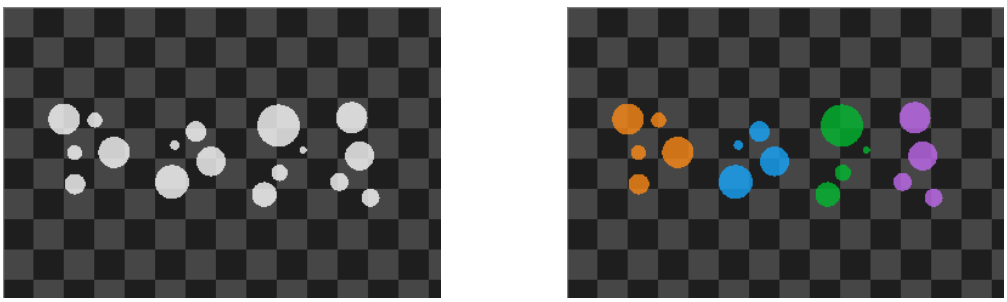
   along the X axis and by

   $$0.5 \cdot (1 - inDistanceBalance)$$

   along the Y axis.
3. Finally the length of the so scaled segment is computed.

Every region being a result of such blob joining has to satisfy conditions defined by **inMaxJointDiameter**, **inMaxJointWidth** and **inMaxJointHeight** parameters. Finally, only regions with appropriate area between **inMinComponentArea** and **inMaxComponentArea** are parts of the output component table.

Examples



SplitRegionIntoComponents *used with* ***inMaxDistance*** *= 50.*

Remarks

This filter is mostly used in **Blob Analysis Technique** please refer to our **Machine Vision Guide - Blob Analysis** article.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

See Also

- SplitRegionIntoBlobs – Splits a region into an array of regions corresponding to its connected components.
- SplitRegionIntoExactlyNComponents – Splits a region into a fixed-size array of regions.

SplitRegionIntoExactlyNComponents

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Splits a region into a fixed-size array of regions.

Syntax

```
void avl::SplitRegionIntoExactlyNComponents
(
 const avl::Region& inRegion,
 int inComponentCount,
 atl::Optional<int> inMaxDistance,
 float inDistanceBalance,
 atl::Optional<float> inMaxJointDiameter,
 atl::Optional<int> inMaxJointWidth,
 atl::Optional<int> inMaxJointHeight,
 bool inRemoveBoundaryBlobs,
 atl::Conditional<atl::Array<avl::Region> >& outComponents
)
```

Parameters

	Name	Type	Range	Default	Description
→	inRegion	const Region&			Input region
→	inComponentCount	int	1 - ∞		
→	inMaxDistance	Optional<int>	0 - ∞	5	
→	inDistanceBalance	float	-1.0 - 1.0		Defines how much important the distance between regions in x coordinate is according to distance in y coordinate
→	inMaxJointDiameter	Optional<float>	0.0 - ∞	NIL	
→	inMaxJointWidth	Optional<int>	0 - ∞	NIL	
→	inMaxJointHeight	Optional<int>	0 - ∞	NIL	
→	inRemoveBoundaryBlobs	bool		False	Flag indicating whether the blobs on border of the input region should be removed or not
←	outComponents	Conditional<Array<Region> >&			

Description

The filter splits the input region into blobs and iteratively joins some of them into bigger components until only **inComponentCount** components remain. Only blobs that are distant from each other by at most **inMaxDistance** can be joined. The joining order is determined based on modified distance between two blobs, so the closest ones are joined first. This modified distance between two blobs is computed as follows:

1. The shortest segment connecting two blobs is computed.
2. The segment is scaled by

   $$0.5 \cdot (1 + inDistanceBalance)$$

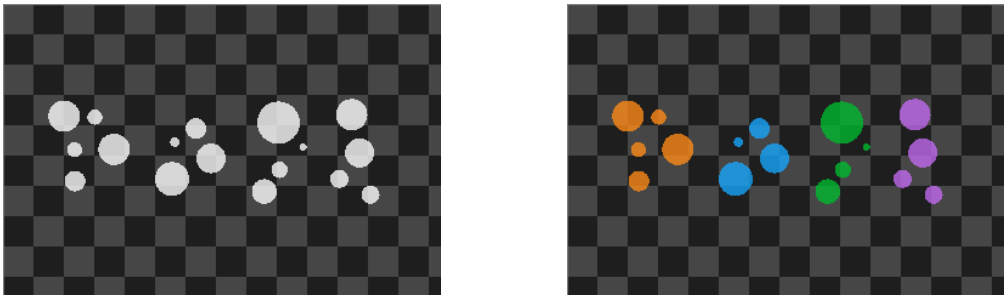
   along the X axis and by

   $$0.5 \cdot (1 - inDistanceBalance)$$

   along the Y axis.
3. Finally the length of the so scaled segment is computed.

Every region being a result of such blob joining has to satisfy conditions defined by **inMaxJointDiameter**, **inMaxJointWidth** and **inMaxJointHeight** parameters.

Examples



SplitRegionIntoExactlyNComponents *used with* ***inComponentCount*** *= 4.*

Remarks

This filter is mostly used in **Blob Analysis Technique** please refer to our **Machine Vision Guide - Blob Analysis** article.

See Also

- SplitRegionIntoBlobs – Splits a region into an array of regions corresponding to its connected components.
- SplitRegionIntoComponents – Splits a region into an array of regions. Operates by merging blobs in accordance to the inMaxDistance parameter.

89. Statistics

Table of content:

FindDataMode_FixedCount

Header: AVL.h
Namespace: avl
Module: FoundationPro

Finds the mode in a set of data values by looking for highest concentration of a fixed number of samples.

Applications: Can be used to determine a histogram maximum without actually creating the histogram.

Syntax

```
void avl::FindDataMode_FixedCount
(
 const atl::Array<float>& inValues,
 atl::Optional<float> inMinValue,
 atl::Optional<float> inMaxValue,
 int inCount,
 atl::Optional<float> inCycle,
 float& outMode,
 float& outMiddle,
 float& outSpread
)
```

Parameters

	Name	Type	Range	Default	Description
→	inValues	const Array<float>&			
→	inMinValue	Optional<float>		NIL	
→	inMaxValue	Optional<float>		NIL	
→	inCount	int	2 - ∞	3	
→	inCycle	Optional<float>		NIL	
←	outMode	float&			
←	outMiddle	float&			
←	outSpread	float&			

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array in FindDataMode_FixedCount.

FindDataMode_FixedSpread

Header: AVL.h
Namespace: avl
Module: FoundationPro

Finds the mode in a set of data values by looking for highest number of samples withing the specified spread.

Applications: Can be used to determine a histogram maximum without actually creating the histogram.

Syntax

```
void avl::FindDataMode_FixedSpread
(
 const atl::Array<float>& inValues,
 float inSpread,
 atl::Optional<float> inCycle,
 float& outMode,
 float& outMiddle,
 int& outCount,
 float& outSpread
)
```

Parameters

	Name	Type	Range	Default	Description
→	inValues	const Array<float>&			
→	inSpread	float	0.0 - ∞	2.0f	
→	inCycle	Optional<float>		NIL	
←	outMode	float&			
←	outMiddle	float&			
←	outCount	int&			
←	outSpread	float&			

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array in FindDataMode_FixedSpread.

FindDataMode_MeanShift

Header: AVL.h
Namespace: avl
Module: FoundationPro

Finds the mode in a set of data values by iteratively computing its median.

Applications: Can be used to determine a histogram maximum without actually creating the histogram.

Syntax

```
void avl::FindDataMode_MeanShift
(
 const atl::Array<float>& inValues,
 int inIterationCount,
 float inSpread,
 float& outMode
)
```

Parameters

	Name	Type	Range	Default	Description
→	inValues	const Array<float>&			
→	inIterationCount	int	0 - 1000	1	
→	inSpread	float	0.0 - ∞	2.0f	
←	outMode	float&			

FindDensityMaxima_FixedCount

Header: AVL.h
Namespace: avl
Module: FoundationPro

Finds local density maxima in set of values by looking for the highest concentration of a fixed number of samples.

Applications: Can be used to determine histogram's local maxima without actually creating the histogram.

Syntax

```
void avl::FindDensityMaxima_FixedCount
(
 const atl::Array<float>& inValues,
 int inCount,
 atl::Optional<float> inMaxSpread,
 bool inAllowOverlapping,
 atl::Optional<float> inCycle,
 avl::DataModeFunction::Type inModeFunction,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Array<float>& outModes,
 atl::Array<float>& outSpreads
)
```

Parameters

	Name	Type	Range	Default	Description
→	inValues	const Array<float>&			Data points array
→	inCount	int	2 - ∞	3	Number of points in a single window
→	inMaxSpread	Optional<float>		NIL	Maximum spread of points within a window
→	inAllowOverlapping	bool			If true all local maxima will be returned otherwise for each group of ranges that overlap a single mode will be returned (the middle of the group)
→	inCycle	Optional<float>		NIL	Length of the cycle if data is to be considered cyclically
→	inModeFunction	DataModeFunction::Type		Mean	Method of calculating the center of a data mode
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Helps to sieve out unnecessary points
←	outModes	Array<float>&			Computed data modes
←	outSpreads	Array<float>&			Spreads of windows that had a mode

Description

Find approximations of local density maxima in a data sample by looking for highest concentrations in a window with a fixed number of samples.

Overlapping windows (i.e. a series of windows with the same count and the same spread) can be returned as a single mode (the middle those windows) if **inAllowOverlapping** is set to false.

inMaxSpread and **inLocalBlindness** can be used to further filter out potentially unnecessary points in the output.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array in FindDensityMaxima_FixedCount.
DomainError	Unsupported mode function in FindDensityMaxima_FixedCount.

See Also

- FindDataMode_FixedCount – Finds the mode in a set of data values by looking for highest concentration of a fixed number of samples.
- FindDensityMaxima_FixedSpread – Finds local density maxima in a set of values by looking for the highest number of samples withing a range determined by the given spread.

FindDensityMaxima_FixedSpread

Header: AVL.h
Namespace: avl
Module: FoundationPro

Finds local density maxima in a set of values by looking for the highest number of samples withing a range determined by the given spread.

Applications: Can be used to determine histogram's local maxima without actually creating the histogram.

Syntax

```
void avl::FindDensityMaxima_FixedSpread
(
 const atl::Array<float>& inValues,
 float inSpread,
 int inMinCount,
 bool inAllowOverlapping,
 atl::Optional<float> inCycle,
 avl::DataModeFunction::Type inModeFunction,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Array<float>& outModes,
 atl::Array<int>& outCounts
)
```

Parameters

	Name	Type	Range	Default	Description
→	inValues	const Array<float>&			Data points array
→	inSpread	float	0.0 - ∞	2.0f	Maximum spread of a single window
→	inMinCount	int	1 - ∞	1	Windows with a smaller count will not be considered
→	inAllowOverlapping	bool			If true all local maxima will be returned otherwise for each group of ranges that overlap only the one with the smallest real spread will be returned
→	inCycle	Optional<float>		NIL	Length of the cycle if data is to be considered cyclically
→	inModeFunction	DataModeFunction::Type		Mean	Method of calculating the center of a data mode
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Helps to sieve out unnecessary points
←	outModes	Array<float>&			Computed data modes
←	outCounts	Array<int>&			Sizes of windows that had a mode

Description

Find approximations of local density maxima in a data sample by looking for highest concentrations in a window with a bounded spread.

Overlapping windows (i.e. a series of windows with the same count) can be sieved if **inAllowOverlapping** is set to false. The sieving removes all modes that are shadowed by (i.e. are overlapping with) another mode with a smaller spread.

inMinCount and **inLocalBlindness** can be used to further filter out potentially unnecessary points in the output.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array in FindDensityMaxima_FixedSpread.

See Also

- FindDataMode_FixedSpread – Finds the mode in a set of data values by looking for highest number of samples withing the specified spread.
- FindDensityMaxima_FixedCount – Finds local density maxima in set of values by looking for the highest concentration of a fixed number of samples.

FindMatchingRegions_IoU

Header: AVL.h
Namespace: avl
Module: FoundationPro

Finds corresponding regions in two arrays based on IoU value.

Syntax

```
void avl::FindMatchingRegions_IoU
(
 const atl::Array<avl::Region>& inMasks,
 const atl::Array<int>& inMasksClasses,
 const atl::Array<avl::Region>& inPredicted,
 const atl::Array<int>& inPredictedClasses,
 double inThreshold,
 atl::Array<atl::Conditional<int> >& outMatchedIndexes,
 atl::Array<atl::Conditional<avl::Region>>& outMatchedRegions,
 atl::Array<atl::Conditional<double>>& outScores,
 atl::Array<atl::Conditional<double>>& diagCandidateScores
)
```

Parameters

	Name	Type	Range	Default	Description
→	inMasks	const Array<Region>&			Original masks
→	inMasksClasses	const Array<int>&			
→	inPredicted	const Array<Region>&			Regions from the classifier
→	inPredictedClasses	const Array<int>&			
→	inThreshold	double	0.0 - 1.0	0.5D	
←	outMatchedIndexes	Array<Conditional<int> >&			
←	outMatchedRegions	Array<Conditional<Region>>&			
←	outScores	Array<Conditional<double>>&			Returns scores of accepted regions
🔍	diagCandidateScores	Array<Conditional<double>>&			Returns best scores for regions

Errors

List of possible exceptions:

Error type	Description
DomainError	All regions in inMasks and inPredicted should have this same format.
DomainError	inMasks and inMasksClasses has different element count
DomainError	inPredicted and inPredictedClasses has different element count

RegionsIoU

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes intersection over union value for two regions.

Syntax

```
void avl::RegionsIoU
(
 const avl::Region& inRegion1,
 const avl::Region& inRegion2,
 double& outIoT
)
```

Parameters

	Name	Type	Default	Description
→	inRegion1	const Region&		First input region
→	inRegion2	const Region&		Second input region
←	outIoT	double&		

Errors

List of possible exceptions:

Error type	Description
DomainError	inRegion1 has different frame dimensions than inRegion2 in RegionsIoU

TableOfConfusion_Basic

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes statistics from a confusion matrix for given TP, FP, TN, FN.

Syntax

```
void avl::TableOfConfusion_Basic
(
 atl::int64 inTruePositive,
 atl::int64 inFalsePositive,
 atl::int64 inTrueNegative,
 atl::int64 inFalseNegative,
 avl::ConfusionTable& outConfusionTable
)
```

Parameters

	Name	Type	Default	Description
→	inTruePositive	int64		
→	inFalsePositive	int64		
→	inTrueNegative	int64		
→	inFalseNegative	int64		
←	outConfusionTable	ConfusionTable&		

TableOfConfusion_BoolArray

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes statistics from a confusion matrix for an array of groundTruth and results.

Syntax

```
void avl::TableOfConfusion_BoolArray
(
 const atl::Array<bool>& inGroundTruth,
 const atl::Array<bool>& inResults,
 atl::int64& outTruePositive,
 atl::int64& outFalsePositive,
 atl::int64& outTrueNegative,
 atl::int64& outFalseNegative,
 avl::ConfusionTable& outConfusionTable
)
```

Parameters

	Name	Type	Default	Description
→	inGroundTruth	const Array<bool>&		
→	inResults	const Array<bool>&		
←	outTruePositive	int64&		
←	outFalsePositive	int64&		
←	outTrueNegative	int64&		
←	outFalseNegative	int64&		
←	outConfusionTable	ConfusionTable&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Array inGroundTruth is empty in TableOfConfusion_BoolArray.

TableOfConfusion_Histograms

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes confusion matrix based on two histograms and threshold value.

Syntax

```
void avl::TableOfConfusion_Histograms
(
 const avl::Histogram& inPositiveValues,
 const avl::Histogram& inNegativeValues,
 float inThreshold,
 avl::ConfusionTable& outConfusionTable
)
```

Parameters

	Name	Type	Default	Description
→	inPositiveValues	const Histogram&		Pixels values under binary classification mask
→	inNegativeValues	const Histogram&		Pixels values not covered by mask
→	inThreshold	float	128.0f	
←	outConfusionTable	ConfusionTable&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Histogram inNegativeValues is empty in TableOfConfusion_Histograms.
DomainError	Histogram inPositiveValues is empty in TableOfConfusion_Histograms.
DomainError	Histograms inPositiveValues and inNegativeValues has different domain in TableOfConfusion_Histograms.
DomainError	inThreshold is larger than domain end in TableOfConfusion_Histograms.
DomainError	inThreshold is smaller than domain start in TableOfConfusion_Histograms.

TableOfConfusion_Images

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes statistics from a confusion matrix for image of groundTruth and results.

Syntax

```
void avl::TableOfConfusion_Images
(
 const avl::Image& inImageGroundTruth,
 const avl::Image& inImageResults,
 int inThreshold,
 atl::int64& outTruePositive,
 atl::int64& outFalsePositive,
 atl::int64& outTrueNegative,
 atl::int64& outFalseNegative,
 avl::ConfusionTable& outConfusionTable
)
```

Parameters

	Name	Type	Default	Description
→	inImageGroundTruth	const Image&		Binary image of ground truth
→	inImageResults	const Image&		
→	inThreshold	int	0	
←	outTruePositive	int64&		
←	outFalsePositive	int64&		
←	outTrueNegative	int64&		
←	outFalseNegative	int64&		
←	outConfusionTable	ConfusionTable&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Height of inImageGroundTruth isn't correspond to inImageResults's height in TableOfConfusion_Images.
DomainError	Image inGroundTruthImage is empty in TableOfConfusion_Images.
DomainError	Width of inImageGroundTruth isn't correspond to inImageResults's width in TableOfConfusion_Images.

Also in AVL Lite

Average

Header: STD.h
Namespace: avl
Module: FoundationLite

Computes the average of an array of real numbers. The array must be not empty.

Syntax

```
void avl::Average
(
 const atl::Array<float>& inValues,
 float& outAverage
)
```

Parameters

	Name	Type	Default	Description
→	inValues	const Array<float>&		
←	outAverage	float&		

Hints

- If the input array is not guaranteed to be non-empty, precede this filter with SkipEmptyArray.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array on input in Average.

Median

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Computes the median of an array of real numbers. The array must be not empty. Optional weights, when supplied, must be positive.

Syntax

```
void avl::Median
(
 const atl::Array<float>& inValues,
 const atl::Optional<const atl::Array<float>&>& inWeights,
 float& outMedian
)
```

Parameters

	Name	Type	Default	Description
→	inValues	const Array<float>&		
→	inWeights	const Optional<const Array<float>&>&	NIL	
←	outMedian	float&		

Hints

- If the input array is not guaranteed to be non-empty, precede this filter with SkipEmptyArray.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array on input in Median.

Median_InPlace

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Computes the median of an array of real numbers. Modifies the input array for the purpose of speed.

Syntax

```
void avl::Median_InPlace
(
 atl::Array<float>& ioValues,
 float& outMedian
)
```

Parameters

	Name	Type	Default	Description
↔	ioValues	Array<float>&		
←	outMedian	float&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array on input in Median_InPlace.

NthValue

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Computes n-th smallest value in an array (0-indexed). The array must be not empty.

Syntax

```
void avl::NthValue
(
 const atl::Array<float>& inValues,
 int inN,
 float& outNthValue
)
```

Parameters

	Name	Type	Default	Description
→	inValues	const Array<float>&		
→	inN	int		
←	outNthValue	float&		

Hints

- If the input array is not guaranteed to be non-empty, precede this filter with SkipEmptyArray.

Errors

List of possible exceptions:

Error type	Description
DomainError	Selected N exceed the array index range in NthValue.

PearsonCorrelation

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Computes Pearson product-moment correlation coefficient. The array must be not empty.

Syntax

```
void avl::PearsonCorrelation
(
 const atl::Array<float>& inValues1,
 const atl::Array<float>& inValues2,
 float& outCorrelation
)
```

Parameters

	Name	Type	Default	Description
→	inValues1	const Array<float>&		Array of values of the first variable
→	inValues2	const Array<float>&		Array of values of the second variable
←	outCorrelation	float&		Computed correlation coefficient between two variables

Hints

- Make sure that the standard deviation of both input arrays is non-zero.

Errors

List of possible exceptions:

Error type	Description
DomainError	Cannot compute the value of PearsonCorrelation. The standard deviation of some set of input values is equal to zero.
DomainError	Empty array on input in PearsonCorrelation.
DomainError	Inconsistent size of arrays in PearsonCorrelation.

Quantile

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Computes the specified quantile of an array of real numbers. The array must be not empty.

Syntax

```
void avl::Quantile
(
 const atl::Array<float>& inValues,
 float inQuantilePoint,
 float& outQuantileValue
)
```

Parameters

	Name	Type	Range	Default	Description
→	inValues	const Array<float>&			
→	inQuantilePoint	float	0.0 - 1.0	0.5f	
←	outQuantileValue	float&			

Hints

- If the input array is not guaranteed to be non-empty, precede this filter with SkipEmptyArray.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array on input in Quantile.

StandardDeviation

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Computes the standard deviation of an array of real numbers. The array must be not empty.

Syntax

```
void avl::StandardDeviation
(
 const atl::Array<float>& inValues,
 const bool inUseSampleFormula,
 float& outStandardDeviation
)
```

Parameters

	Name	Type	Default	Description
→	inValues	const Array<float>&		
→	inUseSampleFormula	const bool		
←	outStandardDeviation	float&		

Hints

- If the input array is not guaranteed to be non-empty, precede this filter with SkipEmptyArray.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array on input in StandardDeviation.

Variance

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Computes the variance of an array of real numbers. The array must be not empty.

Syntax

```
void avl::Variance
(
 const atl::Array<float>& inValues,
 const bool inUseSampleFormula,
 float& outVariance
)
```

Parameters

	Name	Type	Default	Description
→	inValues	const Array<float>&		
→	inUseSampleFormula	const bool		
←	outVariance	float&		

Hints

- If the input array is not guaranteed to be non-empty, precede this filter with SkipEmptyArray.

Errors

List of possible exceptions:

Error type	Description
DomainError	Cannot calculate Variance of sample of size 1. Correct input data, or use Variance filter for population.

90. Surface Fitting

Table of content:

FitCircleToSurfaceHole

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Fits a circle to a hole in a surface plane.

Syntax

```
void avl::FitCircleToSurfaceHole
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 avl::MEstimator::Type inPlaneOutlierSuppression,
 float inClippingFactor,
 int inIterationCount,
 atl::Optional<const avl::Plane3D&> inInitialPlane,
 avl::CircleFittingMethod::Type inCircleFittingMethod,
 atl::Optional<avl::MEstimator::Type> inCircleOutlierSuppression,
 atl::Conditional<avl::Circle3D>& outCircle3D,
 avl::Plane3D& outPlane,
 atl::Optional<atl::Array<avl::Point3D>&> outInliers = atl::NIL,
 atl::Optional<atl::Array<float>&> outDistances = atl::NIL,
 atl::Optional<float&> outSignedDistanceSum = atl::NIL,
 atl::Optional<float&> outDistanceSum = atl::NIL,
 atl::Optional<atl::Array<float>&> outSquaredDistances = atl::NIL,
 atl::Optional<float&> outSquaredDistanceSum = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inRoi	Optional<const Region&>		NIL	Region of interest
→	inPlaneOutlierSuppression	MEstimator::Type			Selects a method for ignoring points not lying on the fitted plane
→	inClippingFactor	float	0.675 - 6.0	2.5f	Multitude of standard deviation within which points are considered inliers
→	inIterationCount	int	0 - ∞	5	Number of iterations of outlier suppressing algorithm
→	inInitialPlane	Optional<const Plane3D&>		NIL	Initial approximation of a plane (if available)
→	inCircleFittingMethod	CircleFittingMethod::Type			Select a method for fitting a circle to a set of points
→	inCircleOutlierSuppression	Optional<MEstimator::Type>		NIL	Selects a method for ignoring points not lying on the fitted circle
←	outCircle3D	Conditional<Circle3D>&			Circle fitted to a surface hole
←	outPlane	Plane3D&			Plane fitted to the surface points
←	outInliers	Optional<Array<Point3D>&>		NIL	Points matching the computed plane
←	outDistances	Optional<Array<float>&>		NIL	Distances of the input surface points to a resulting plane
←	outSignedDistanceSum	Optional<float&>		NIL	Sum of signed distances of the input surface points to a resulting plane
←	outDistanceSum	Optional<float&>		NIL	Sum of distances of the input surface points to a resulting plane
←	outSquaredDistances	Optional<Array<float>&>		NIL	Squared distances of the input surface points to a resulting plane
←	outSquaredDistanceSum	Optional<float&>		NIL	Sum of squared distances of the input surface points to a resulting plane

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outInliers**, **outDistances**, **outSignedDistanceSum**, **outDistanceSum**, **outSquaredDistances**, **outSquaredDistanceSum**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region of interest exceeds an input surface in FitCircleToSurfaceHole.

FitPlaneToSurface

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Approximates points of the input surface with a plane using the Least Squares method.

Syntax

```
void avl::FitPlaneToSurface
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 avl::Plane3D& outPlane,
 atl::Optional<atl::Array<float>&> outDistances = atl::NIL,
 atl::Optional<float&> outSignedDistanceSum = atl::NIL,
 atl::Optional<float&> outDistanceSum = atl::NIL,
 atl::Optional<atl::Array<float>&> outSquaredDistances = atl::NIL,
 atl::Optional<float&> outSquaredDistanceSum = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inSurface	const Surface&		Input surface
→	inRoi	Optional<const Region&>	NIL	Region of interest
←	outPlane	Plane3D&		Fitted plane
←	outDistances	Optional<Array<float>&>	NIL	Distances of the input surface points to a resulting plane
←	outSignedDistanceSum	Optional<float&>	NIL	Sum of signed distances of the input surface points to a resulting plane
←	outDistanceSum	Optional<float&>	NIL	Sum of distances of the input surface points to a resulting plane
←	outSquaredDistances	Optional<Array<float>&>	NIL	Squared distances of the input surface points to a resulting plane
←	outSquaredDistanceSum	Optional<float&>	NIL	Sum of squared distances of the input surface points to a resulting plane

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outDistances**, **outSignedDistanceSum**, **outDistanceSum**, **outSquaredDistances**, **outSquaredDistanceSum**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	No points to fit the plane to in FitPlaneToSurface.
DomainError	Region of interest exceeds an input surface in FitPlaneToSurface.

FitPlaneToSurface_M

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Approximates points of the input surface with a plane using selected M-estimator for outlier suppression.

Applications: Finding a locally optimal plane. Good enough when the number of outliers is small.

Syntax

```
void avl::FitPlaneToSurface_M
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 avl::MEstimator::Type inOutlierSuppression,
 float inClippingFactor,
 int inIterationCount,
 atl::Optional<const avl::Plane3D&> inInitialPlane,
 avl::Plane3D& outPlane,
 atl::Optional<atl::Array<avl::Point3D>&> outInliers = atl::NIL,
 atl::Optional<atl::Array<float>&> outDistances = atl::NIL,
 atl::Optional<float&> outSignedDistanceSum = atl::NIL,
 atl::Optional<float&> outDistanceSum = atl::NIL,
 atl::Optional<atl::Array<float>&> outSquaredDistances = atl::NIL,
 atl::Optional<float&> outSquaredDistanceSum = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inRoi	Optional<const Region&>		NIL	Region of interest
→	inOutlierSuppression	MEstimator::Type			Selects a method for ignoring points being much different from the rest
→	inClippingFactor	float	0.675 - 6.0	2.5f	Multitude of standard deviation within which points are considered inliers
→	inIterationCount	int	0 - ∞	5	Number of iterations of outlier suppressing algorithm
→	inInitialPlane	Optional<const Plane3D&>		NIL	Initial approximation (if available)
←	outPlane	Plane3D&			Fitted plane
←	outInliers	Optional<Array<Point3D>&>		NIL	Points matching the computed plane
←	outDistances	Optional<Array<float>&>		NIL	Distances of the input points to a resulting plane
←	outSignedDistanceSum	Optional<float&>		NIL	Sum of signed distances of the input points to a resulting plane
←	outDistanceSum	Optional<float&>		NIL	Sum of distances of the input points to a resulting plane
←	outSquaredDistances	Optional<Array<float>&>		NIL	Squared distances of the input points to a resulting plane
←	outSquaredDistanceSum	Optional<float&>		NIL	Sum of squared distances of the input points to a resulting plane

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outInliers**, **outDistances**, **outSignedDistanceSum**, **outDistanceSum**, **outSquaredDistances**, **outSquaredDistanceSum**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	No points to fit the plane to in FitPlaneToSurface_M
DomainError	Region of interest exceeds an input surface in FitPlaneToSurface_M

91. Format

Table of content:

Also in AVL Lite

FormatLocationToString

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a location to a string of format "(X, Y)".

Syntax

```
void avl::FormatLocationToString
(
 const avl::Location& inLocation,
 const int inDigitCount,
 const atl::String& inTrailingCharacter,
 const bool inForceSignPrinting,
 const atl::String& inSuffix,
 const int inSystemBase,
 const bool inPrintBrackets,
 atl::String& outString
)
```

Parameters

	Name	Type	Range	Default	Description
→	inLocation	const Location&			
→	inDigitCount	const int	0 - ∞		How many characters the output coordinate should have at least
→	inTrailingCharacter	const String&		\"0\"	Defines the trailing character
→	inForceSignPrinting	const bool		False	Forces printing the signs of the coordinates even if the number is positive
→	inSuffix	const String&		\"\"	Defines a suffix. Generally it is an unit of value (e.g. mm)
→	inSystemBase	const int	2 - 16	10	The base of the numeral system
→	inPrintBrackets	const bool		True	Determines whether the brackets should be printed or not
←	outString	String&			

Also in AVL Lite

FormatPoint2DToString

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a 2D point to a string of format "(X, Y)".

Syntax

```
void avl::FormatPoint2DToString
(
 const avl::Point2D& inPoint,
 const int inIntegerDigitCount,
 const int inFractionalDigitCount,
 const atl::String& inDecimalMark,
 const atl::String& inTrailingCharacter,
 const bool inForceSignPrinting,
 const atl::String& inSuffix,
 const bool inPrintBrackets,
 atl::String& outString
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoint	const Point2D&			
→	inIntegerDigitCount	const int	0 - ∞		How many characters the integer part of the coordinates should have at least
→	inFractionalDigitCount	const int	0 - 100	3	How many characters the fractional part of the coordinates should have
→	inDecimalMark	const String&		\".\"	The symbol used to separate the integer part from the fractional part of the coordinates
→	inTrailingCharacter	const String&		\"0\"	Defines the trailing character
→	inForceSignPrinting	const bool		False	Forces printing the signs of the numbers even if the number is positive
→	inSuffix	const String&		\"\"	Defines a suffix. Generally it is an unit of value (e.g. mm)
→	inPrintBrackets	const bool		True	Determines whether the brackets should be printed or not
←	outString	String&			

FormatPoint3DToString

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a 3D point to a string of format "(X, Y, Z)".

Syntax

```
void avl::FormatPoint3DToString
(
 const avl::Point3D& inPoint,
 const int inIntegerDigitCount,
 const int inFractionalDigitCount,
 const atl::String& inDecimalMark,
 const atl::String& inTrailingCharacter,
 const bool inForceSignPrinting,
 const atl::String& inSuffix,
 const bool inPrintBrackets,
 atl::String& outString
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoint	const Point3D&			
→	inIntegerDigitCount	const int	0 - ∞		How many characters the integer part of the coordinates should have at least
→	inFractionalDigitCount	const int	0 - 100	3	How many characters the fractional part of the coordinates should have
→	inDecimalMark	const String&		\".\"	The symbol used to separate the integer part from the fractional part of the coordinates
→	inTrailingCharacter	const String&		\"0\"	Defines the trailing character
→	inForceSignPrinting	const bool		False	Forces printing the signs of the numbers even if the number is positive
→	inSuffix	const String&		\"\"	Defines a suffix. Generally it is an unit of value (e.g. mm)
→	inPrintBrackets	const bool		True	Determines whether the brackets should be printed or not
←	outString	String&			

92. Fourier Analysis

Table of content:

FourierTransform

Header: AVL.h
Namespace: avl
Module: FoundationPro

Transforms an image into frequency domain using Fourier transformation.

Syntax

```
void avl::FourierTransform
(
 const avl::Image& inSpatialDomainImage,
 avl::Image& outFrequencyDomainImage
)
```

Parameters

	Name	Type	Default	Description
→	inSpatialDomainImage	const Image&		
←	outFrequencyDomainImage	Image&		

Requirements

For input **inSpatialDomainImage** only pixel formats are supported: 1□uint8, 1□int8, 1□uint16, 1□int16, 1□int32, 1□real.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect input image depth in FourierTransform.
DomainError	Not supported inSpatialDomainImage pixel format in FourierTransform. Supported formats: 1xUInt8, 1xInt8, 1xUInt16, 1xInt16, 1xInt32, 1xReal.

FrequencyDomain_FilterFrequencies

Header: AVL.h
Namespace: avl
Module: FoundationPro

Filters the frequencies in a frequency domain image suppressing the elements of too low or too high frequency.

Syntax

```
void avl::FrequencyDomain_FilterFrequencies
(
 const avl::Image& inFrequencyDomainImage,
 const atl::Optional<float>& inMinFrequency,
 const atl::Optional<float>& inMaxFrequency,
 avl::Image& outFrequencyDomainImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inFrequencyDomainImage	const Image&			Input image in frequency domain
→	inMinFrequency	const Optional<float>&	0.0 - ∞	NIL	Minimum frequency that will be kept
→	inMaxFrequency	const Optional<float>&	0.0 - ∞	NIL	Maximum frequency that will be kept
←	outFrequencyDomainImage	Image&			Filtered image in frequency domain

Requirements

For input **inFrequencyDomainImage** only pixel formats are supported: 2□real.

Read more about pixel formats in Image documentation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported inFrequencyDomainImage pixel format in FrequencyDomain_FilterFrequencies. Supported formats: 2xReal.

FrequencyDomain_ModulusImage

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes the modulus of each frequency domain image pixel.

Syntax

```
void avl::FrequencyDomain_ModulusImage
(
 const avl::Image& inFrequencyDomainImage,
 avl::Image& outModulusImage
)
```

Parameters

	Name	Type	Default	Description
→	inFrequencyDomainImage	const Image&		
←	outModulusImage	Image&		

Requirements

For input **inFrequencyDomainImage** only pixel formats are supported: 2□real.

Read more about pixel formats in Image documentation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported inFrequencyDomainImage pixel format in FrequencyDomain_ModulusImage. Supported formats: 2xReal.

FrequencyDomain_PhaseImage

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes the phase of each frequency domain image pixel.

Syntax

```
void avl::FrequencyDomain_PhaseImage
(
 const avl::Image& inFrequencyDomainImage,
 avl::Image& outPhaseImage
)
```

Parameters

	Name	Type	Default	Description
→	inFrequencyDomainImage	const Image&		
←	outPhaseImage	Image&		

Requirements

For input **inFrequencyDomainImage** only pixel formats are supported: 2□real.

Read more about pixel formats in Image documentation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported inFrequencyDomainImage pixel format in FrequencyDomain_PhaseImage. Supported formats: 2xReal.

InverseFourierTransform

Header: AVL.h
Namespace: avl
Module: FoundationPro

Transforms an image in frequency domain back to spatial domain using inverse Fourier transformation.

Syntax

```
void avl::InverseFourierTransform
(
 const avl::Image& inFrequencyDomainImage,
 avl::Image& outSpatialDomainImage
)
```

Parameters

	Name	Type	Default	Description
→	inFrequencyDomainImage	const Image&		
←	outSpatialDomainImage	Image&		

Requirements

For input **inFrequencyDomainImage** only pixel formats are supported: 2□real.

Read more about pixel formats in Image documentation.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported inFrequencyDomainImage pixel format in InverseFourierTransform. Supported formats: 2xReal.

93. FTP

Table of content:

Ftp_ReceiveFile

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Downloads a file from a remote server using FTP (File Transfer Protocol).

Syntax

```
void avl::Ftp_ReceiveFile
(
 const atl::File& inFilePath,
 const atl::String& inHostName,
 const atl::String& inFtpFilePath,
 atl::Optional<const atl::String&> inUsername,
 atl::Optional<const atl::String&> inPassword,
 bool inUsePassiveMode
)
```

Parameters

	Name	Type	Default	Description
➡	inFilePath	const File&		Location of the file on a remote server
➡	inHostName	const String&	\"ftp://\"	URL address of the remote server.
➡	inFtpFilePath	const String&		Location of the file on a remote server.
➡	inUsername	Optional<const String&>	NIL	User name needed to log in.
➡	inPassword	Optional<const String&>	NIL	Password associated with a user name.
➡	inUsePassiveMode	bool		Protocol communication mode. May be necessary when using a firewall.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty ftp file path on input in Ftp_ReceiveFile
DomainError	Empty host name on input in Ftp_ReceiveFile
DomainError	Empty path on input in Ftp_ReceiveFile

Ftp_ReceiveImage

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Downloads an image from a remote server using FTP (File Transfer Protocol).

Syntax

```
void avl::Ftp_ReceiveImage
(
 const atl::String& inHostName,
 const atl::String& inFtpFilePath,
 atl::Optional<const atl::String&> inUsername,
 atl::Optional<const atl::String&> inPassword,
 bool inUsePassiveMode,
 bool inLoadAlphaChannel,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inHostName	const String&	\"ftp://\"	URL address of the remote server.
→	inFtpFilePath	const String&		Location of the file on a remote server.
→	inUsername	Optional<const String&>	NIL	User name needed to log in
→	inPassword	Optional<const String&>	NIL	Password associated with a user name.
→	inUsePassiveMode	bool		Protocol communication mode. May be necessary when using a firewall.
→	inLoadAlphaChannel	bool		Whether to load the alpha channel (if exists) as an additional image channel
←	outImage	Image&		Image downloaded from the remote server.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty ftp file path on input in Ftp_ReceiveImage
DomainError	Empty host name on input in Ftp_ReceiveImage

Ftp_ReceiveString

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Downloads a text string from a remote server using FTP (File Transfer Protocol).

Syntax

```
void avl::Ftp_ReceiveString
(
 const atl::String& inHostName,
 const atl::String& inFtpFilePath,
 atl::Optional<const atl::String&> inUsername,
 atl::Optional<const atl::String&> inPassword,
 bool inUsePassiveMode,
 atl::String& outText
)
```

Parameters

	Name	Type	Default	Description
→	inHostName	const String&	\"ftp://\"	URL address of the remote server.
→	inFtpFilePath	const String&		Location of the file on a remote server.
→	inUsername	Optional<const String&>	NIL	User name needed to log in.
→	inPassword	Optional<const String&>	NIL	Password associated with a user name.
→	inUsePassiveMode	bool		Protocol communication mode. May be necessary when using a firewall.
←	outText	String&		Text read from a remote server.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty ftp file path on input in Ftp_ReceiveString
DomainError	Empty host name on input in Ftp_ReceiveString

Ftp_SendFile

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Sends a file to a remote server using FTP (File Transfer Protocol).

Syntax

```
void avl::Ftp_SendFile
(
 const atl::File& inFilePath,
 const atl::String& inHostName,
 const atl::String& inDestinationFileName,
 atl::Optional<const atl::String&> inUsername,
 atl::Optional<const atl::String&> inPassword,
 bool inUsePassiveMode
)
```

Parameters

	Name	Type	Default	Description
➡	inFilePath	const File&		Path to a file stored on the local drive.
➡	inHostName	const String&	\"ftp://\"	URL address of the remote server.
➡	inDestinationFileName	const String&		Path where a file should be saved on a remote server.
➡	inUsername	Optional<const String&>	NIL	User name needed to log in.
➡	inPassword	Optional<const String&>	NIL	Password associated with a user name.
➡	inUsePassiveMode	bool		Protocol communication mode. May be necessary when using a firewall.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty destination file name on input in Ftp_SendFile
DomainError	Empty file path on input in Ftp_SendFile
DomainError	Empty host name on input in Ftp_SendFile

Ftp_SendImage

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Sends an image to a remote server using FTP (File Transfer Protocol).

Syntax

```
void avl::Ftp_SendImage
(
 const avl::Image& inImage,
 atl::Optional<avl::ImageFileFormat::Type> inImageFileFormat,
 const atl::String& inHostName,
 const atl::String& inDestinationFileName,
 atl::Optional<const atl::String&> inUsername,
 atl::Optional<const atl::String&> inPassword,
 bool inUsePassiveMode
)
```

Parameters

	Name	Type	Default	Description
➡	inImage	const Image&		Image to send.
➡	inImageFileFormat	Optional<ImageFileFormat::Type>	NIL	If Nil the format will be chosen on the basis of extension
➡	inHostName	const String&	\"ftp://\"	URL address of the remote server.
➡	inDestinationFileName	const String&		Path where an image file should be saved on a remote server.
➡	inUsername	Optional<const String&>	NIL	User name needed to log in.
➡	inPassword	Optional<const String&>	NIL	Password associated with a user name.
➡	inUsePassiveMode	bool		Protocol communication mode. May be necessary when using a firewall.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty destination file name on input in Ftp_SendImage
DomainError	Empty host name on input in Ftp_SendImage

Ftp_SendString

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Sends a string to a remote serve using FTP (File Transfer Protocol).

Syntax

```
void avl::Ftp_SendString
(
 const atl::String& inText,
 const atl::String& inHostName,
 const atl::String& inDestinationFileName,
 atl::Optional<const atl::String&> inUsername,
 atl::Optional<const atl::String&> inPassword,
 bool inUsePassiveMode
)
```

Parameters

	Name	Type	Default	Description
➡	inText	const String&		Text to send
➡	inHostName	const String&	\"ftp://\"	URL address of the remote server.
➡	inDestinationFileName	const String&		Path where a text file should be saved on a remote server.
➡	inUsername	Optional<const String&>	NIL	User name needed to log in.
➡	inPassword	Optional<const String&>	NIL	Password associated with a user name.
➡	inUsePassiveMode	bool		Protocol communication mode. May be necessary when using a firewall.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty destination file name on input in Ftp_SendString
DomainError	Empty host name on input in Ftp_SendString

94. Histogram Features

Table of content:

HistogramAverage

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the average of histogram bin values.

Syntax

```
void avl::HistogramAverage
(
 const avl::Histogram& inHistogram,
 double& outAverage
)
```

Parameters

	Name	Type	Default	Description
→	inHistogram	const Histogram&		Input histogram
←	outAverage	double&		Output average

Errors

List of possible exceptions:

Error type	Description
DomainError	Input histogram is empty in HistogramAverage.

See Also

- HistogramMinimum – Computes histogram bar values minimum - its location and value.
- HistogramMaximum – Computes histogram bar values maximum - its location and value.

HistogramLocalExtrema

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the histogram local extrema.

Syntax

```
void avl::HistogramLocalExtrema
(
 const avl::Histogram& inHistogram,
 atl::Optional<const avl::Range&> inRange,
 atl::Optional<double> inMinValue,
 atl::Optional<double> inMaxValue,
 const bool inCyclic,
 const bool inConsiderPlateaus,
 avl::ExtremumType::Type inExtremumType,
 avl::HistogramInterpolationMethod::Type inInterpolationMethod,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Array<avl::HistogramExtremum>& outLocalExtrema
)
```

Parameters

	Name	Type	Default	Description
→	inHistogram	const Histogram&		Input histogram
→	inRange	Optional<const Range&>	NIL	Search range
→	inMinValue	Optional<double>	NIL	Minimum value of an extremum
→	inMaxValue	Optional<double>	NIL	Maximum value of an extremum
→	inCyclic	const bool	False	Whether to check differences between the first and last elements
→	inConsiderPlateaus	const bool		Indicates whether the result should include centers of plateau extrema
→	inExtremumType	ExtremumType::Type		Type of extremum to find
→	inInterpolationMethod	HistogramInterpolationMethod::Type	Linear	Histogram bins interpolation method
→	inLocalBlindness	Optional<const LocalBlindness&>	NIL	Defines conditions in which weaker extrema can be detected in the vicinity of stronger ones
←	outLocalExtrema	Array<HistogramExtremum>&		Output histogram extrema

HistogramMaximum

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes histogram bar values maximum - its location and value.

Syntax

```
void avl::HistogramMaximum
(
 const avl::Histogram& inHistogram,
 atl::Optional<const avl::Range&> inRange,
 avl::HistogramInterpolationMethod::Type inInterpolationMethod,
 int& outMaximumLocation,
 float& outPoint,
 double& outMaximumValue
)
```

Parameters

	Name	Type	Default	Description
→	inHistogram	const Histogram&		Input histogram
→	inRange	Optional<const Range&>	NIL	Search range
→	inInterpolationMethod	HistogramInterpolationMethod::Type		Histogram bins interpolation method
←	outMaximumLocation	int&		Output maximum location
←	outPoint	float&		Output subpixel maximum location
←	outMaximumValue	double&		Output maximum value

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty histogram on input in HistogramMaximum.
DomainError	Unsupported interpolation method in HistogramMaximum.

See Also

- HistogramMinimum – Computes histogram bar values minimum - its location and value.
- HistogramAverage – Computes the average of histogram bin values.

HistogramMinimum

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes histogram bar values minimum - its location and value.

Syntax

```
void avl::HistogramMinimum
(
 const avl::Histogram& inHistogram,
 atl::Optional<const avl::Range&> inRange,
 avl::HistogramInterpolationMethod::Type inInterpolationMethod,
 int& outMinimumLocation,
 float& outPoint,
 double& outMinimumValue
)
```

Parameters

	Name	Type	Default	Description
→	inHistogram	const Histogram&		Input histogram
→	inRange	Optional<const Range&>	NIL	Search range
→	inInterpolationMethod	HistogramInterpolationMethod::Type		Histogram bins interpolation method
←	outMinimumLocation	int&		Output minimum location
←	outPoint	float&		Output subpixel minimum location
←	outMinimumValue	double&		Output minimum value

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty histogram on input in HistogramMinimum.
DomainError	Unsupported interpolation method in HistogramMinimum.

See Also

- HistogramAverage – Computes the average of histogram bin values.
- HistogramMaximum – Computes histogram bar values maximum - its location and value.

HistogramSize

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the number of histogram bins.

Syntax

```
void avl::HistogramSize
(
 const avl::Histogram& inHistogram,
 int& outSize
)
```

Parameters

	Name	Type	Default	Description
→	inHistogram	const Histogram&		Input histogram
←	outSize	int&		Output size

HistogramSum

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the sum of histogram bin values.

Syntax

```
void avl::HistogramSum
(
 const avl::Histogram& inHistogram,
 double& outSum
)
```

Parameters

	Name	Type	Default	Description
→	inHistogram	const Histogram&		Input histogram
←	outSum	double&		Output sum

95. Histogram Data Statistics

Table of content:

HistogramDataAverage

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the average of the histogrammed numeric data.

Syntax

```
void avl::HistogramDataAverage
(
 const avl::Histogram& inHistogram,
 float& outAverage
)
```

Parameters

	Name	Type	Default	Description
→	inHistogram	const Histogram&		Input histogram
←	outAverage	float&		Average of the histogrammed data

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty histogram on input in HistogramDataAverage.
DomainError	Input histogram contains negative bins or its data is empty in HistogramDataAverage.

HistogramDataMaximum

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the approximation of the largest value of the histogrammed numeric data.

Syntax

```
void avl::HistogramDataMaximum
(
 const avl::Histogram& inHistogram,
 float& outMaximum
)
```

Parameters

	Name	Type	Default	Description
→	inHistogram	const Histogram&		Input histogram
←	outMaximum	float&		Maximum of the histogrammed data

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty histogram on input in HistogramDataMaximum.
DomainError	The histogrammed data is empty in HistogramDataMaximum.

HistogramDataMedian

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the median of the histogrammed numeric data.

Syntax

```
void avl::HistogramDataMedian
(
 const avl::Histogram& inHistogram,
 float& outMedian
)
```

Parameters

	Name	Type	Default	Description
→	inHistogram	const Histogram&		Input histogram
←	outMedian	float&		Median of the histogrammed data

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty histogram on input in HistogramDataMedian.
DomainError	Input histogram contains negative bins or its data is empty in HistogramDataMedian.

See Also

- HistogramDataAverage – Computes the average of the histogrammed numeric data.
- HistogramDataQuantile – Computes the specified quantile of the histogrammed numeric data.
- HistogramDataStandardDeviation – Computes the standard deviation of the histogrammed numeric data.
- HistogramDataVariance – Computes the variance of the histogrammed numeric data.

HistogramDataMinimum

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the approximation of the smallest value of the histogrammed numeric data.

Syntax

```
void avl::HistogramDataMinimum
(
 const avl::Histogram& inHistogram,
 float& outMinimum
)
```

Parameters

	Name	Type	Default	Description
→	inHistogram	const Histogram&		Input histogram
←	outMinimum	float&		Minimum of the histogrammed data

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty histogram on input in HistogramDataMinimum.
DomainError	The histogrammed data is empty in HistogramDataMinimum.

HistogramDataNthValue

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes nth smallest (or largest) value of the histogrammed numeric data.

Syntax

```
void avl::HistogramDataNthValue
(
 const avl::Histogram& inHistogram,
 const double inN,
 const avl::SortingOrder::Type inSortingOrder,
 float& outNthValue
)
```

Parameters

	Name	Type	Default	Description
→	inHistogram	const Histogram&		Input histogram
→	inN	const double		
→	inSortingOrder	const SortingOrder::Type		
←	outNthValue	float&		Nth smallest (or largest) value of the histogrammed data

Errors

List of possible exceptions:

Error type	Description
DomainError	Input histogram contains negative bins in HistogramDataNthValue.
DomainError	Selected N exceeds index range in HistogramDataNthValue.

HistogramDataQuantile

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the specified quantile of the histogrammed numeric data.

Syntax

```
void avl::HistogramDataQuantile
(
 const avl::Histogram& inHistogram,
 const float inQuantilePoint,
 float& outQuantileValue
)
```

Parameters

	Name	Type	Range	Default	Description
→	inHistogram	const Histogram&			Input histogram
→	inQuantilePoint	const float	0.0 - 1.0	0.5f	
←	outQuantileValue	float&			

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty histogram on input in HistogramDataQuantile.
DomainError	Input histogram contains negative bins or its data is empty in HistogramDataQuantile.

See Also

- HistogramDataAverage – Computes the average of the histogrammed numeric data.
- HistogramDataMedian – Computes the median of the histogrammed numeric data.
- HistogramDataStandardDeviation – Computes the standard deviation of the histogrammed numeric data.
- HistogramDataVariance – Computes the variance of the histogrammed numeric data.

HistogramDataSize

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the number of elements of the histogrammed numeric data.

Syntax

```
void avl::HistogramDataSize
(
 const avl::Histogram& inHistogram,
 double& outSize
)
```

Parameters

	Name	Type	Default	Description
→	inHistogram	const Histogram&		Input histogram
←	outSize	double&		Number of elements of the histogrammed data

HistogramDataStandardDeviation

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the standard deviation of the histogrammed numeric data.

Syntax

```
void avl::HistogramDataStandardDeviation
(
 const avl::Histogram& inHistogram,
 float& outStandardDeviation
)
```

Parameters

	Name	Type	Default	Description
→	inHistogram	const Histogram&		Input histogram
←	outStandardDeviation	float&		Standard deviation of the histogrammed data

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty histogram on input in HistogramDataStandardDeviation.
DomainError	Input histogram contains negative bins or its data is empty in HistogramDataStandardDeviation.

See Also

- HistogramDataAverage – Computes the average of the histogrammed numeric data.
- HistogramDataMedian – Computes the median of the histogrammed numeric data.
- HistogramDataQuantile – Computes the specified quantile of the histogrammed numeric data.
- HistogramDataVariance – Computes the variance of the histogrammed numeric data.

HistogramDataVariance

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the variance of the histogrammed numeric data.

Syntax

```
void avl::HistogramDataVariance
(
 const avl::Histogram& inHistogram,
 float& outVariance
)
```

Parameters

	Name	Type	Default	Description
→	inHistogram	const Histogram&		Input histogram
←	outVariance	float&		Variance of the histogrammed data

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty histogram on input in HistogramDataVariance.
DomainError	Input histogram contains negative bins or its data is empty in HistogramDataVariance.

See Also

- HistogramDataAverage – Computes the average of the histogrammed numeric data.
- HistogramDataMedian – Computes the median of the histogrammed numeric data.
- HistogramDataStandardDeviation – Computes the standard deviation of the histogrammed numeric data.
- HistogramDataQuantile – Computes the specified quantile of the histogrammed numeric data.

96. Histogram Metrics

Table of content:

HistogramDistance

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the numeric distance between two histograms.

Syntax

```
void avl::HistogramDistance
(
 const avl::Histogram& inHistogram1,
 const avl::Histogram& inHistogram2,
 avl::DistanceMeasure::Type inDistanceMeasure,
 double& outDistance
)
```

Parameters

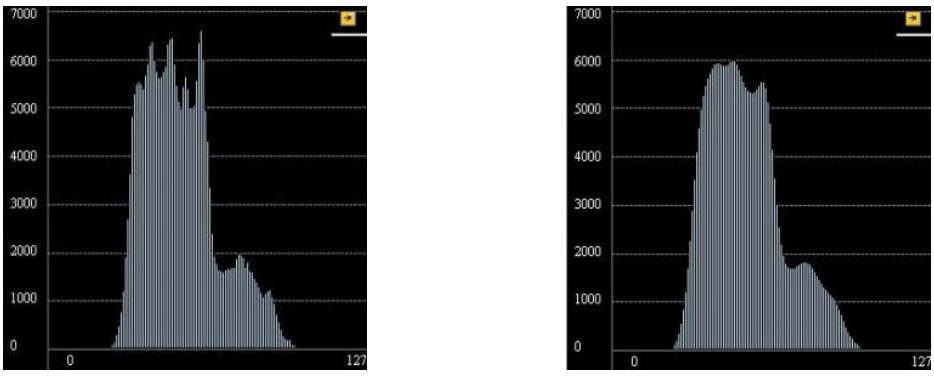
	Name	Type	Default	Description
→	inHistogram1	const Histogram&		First input histogram
→	inHistogram2	const Histogram&		Second input histogram
→	inDistanceMeasure	DistanceMeasure::Type		Measure of distance
←	outDistance	double&		Output distance value

Description

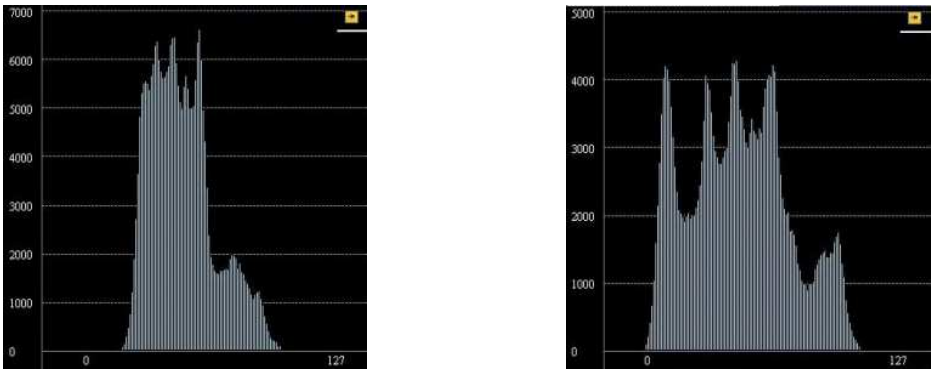
The operation computes the approximate difference between two histograms using the selected distance measure.

- If the **inDistanceMeasure** is set to **MeanError** then the resulting **outDistance** is the average difference between corresponding bins of the histograms.
- If the **inDistanceMeasure** is set to **MeanSquaredError** then the resulting **outDistance** is the average squared difference between corresponding bins of the histograms.

Examples



Mean Squared Error between the sample histograms equals 68302.590.



Mean Squared Error between the sample histograms equals 3043686.

Remarks

- **inHistogram1** and **inHistogram2** must have the same BinSizes, otherwise an error with appropriate description occurs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty histograms on input in HistogramDistance.
DomainError	Inconsistent histogram bin sizes / domains in HistogramDistance.
DomainError	Not supported DistanceMeasure in HistogramDistance.

See Also

- ImageHistogram – Computes the histogram of the image pixel values.

HistogramIntersection

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Calculates normalized histogram intersection norm.

Syntax

```
void avl::HistogramIntersection
(
 const avl::Histogram& inHistogram1,
 const avl::Histogram& inHistogram2,
 double& outHistogramIntersection
)
```

Parameters

	Name	Type	Default	Description
➡	inHistogram1	const Histogram&		First input histogram
➡	inHistogram2	const Histogram&		Second input histogram
⬅	outHistogramIntersection	double&		

Description

The operation computes the normalized histogram intersection defined as:

$$\frac{\sum_{j=1}^{n} \min\left(inHistogram1_j, inHistogram2_j\right)}{\sum_{j=1}^{n} inHistogram2_j}$$

Remarks

- Data sets for the input histograms cannot be empty, otherwise an error with appropriate description occurs,
- **inHistogram1** and **inHistogram2** must have the same BinSizes, otherwise an error with appropriate description occurs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty histogram on input in HistogramIntersection.
DomainError	Input histogram formats are not the same in HistogramIntersection.
DomainError	Negative value in a histogram in HistogramIntersection.

See Also

- ImageHistogram – Computes the histogram of the image pixel values.

97. HTTP

Table of content:

Http_DecodeURL

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts text from URL friendly text to a string.

Syntax

```
void avl::Http_DecodeURL
(
 const atl::String& inEncoded,
 atl::String& outDecoded
)
```

Parameters

	Name	Type	Default	Description
→	inEncoded	const String&		Percent encoded string
←	outDecoded	String&		Decoded string

Description

Filter decodes an percent-encoded string to the raw text.

More details can be found in Percent-encoding standard.

Examples

String **Node%20%3D%201%2B2%2B3** will be decoded to **Node = 1+2+3**.

See Also

- Http_EncodeURL – Converts string to URL friendly text.

Http_EncodeURL

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Converts string to URL friendly text.

Syntax

```
void avl::Http_EncodeURL
(
 const atl::String& inDecoded,
 atl::String& outEncoded
)
```

Parameters

	Name	Type	Default	Description
→	inDecoded	const String&		String to be encoded
←	outEncoded	String&		Percent-encoded string

Description

Filter converts a raw text to the format which can be used in a GET or POST request. Most of white spaces are converted to percent encoded characters.

More details can be found in Percent-encoding standard.

Examples

String **Node = 1+2+3** will be encoded to **Node%20%3D%201%2B2%2B3**.

See Also

- Http_DecodeURL – Converts text from URL friendly text to a string.

Http_SendBinaryData

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Sends a raw HTTP request.

Syntax

```
void avl::Http_SendBinaryData
(
 const atl::String& inUrl,
 const avl::ByteBuffer& inData,
 int inTimeout,
 atl::Conditional<avl::ByteBuffer>& outAnswer,
 int& outResponseCode
)
```

Parameters

	Name	Type	Range	Default	Description
→	inUrl	const String&			URL of request.
→	inData	const ByteBuffer&			Binary data to be send
→	inTimeout	int	0 - ∞	60	Request timeout in seconds.
←	outAnswer	Conditional<ByteBuffer>&			Answer data
←	outResponseCode	int&			Answer code. Typically 200 for OK.

Http_SendRequest_GET

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Sends a GET request to server and receives a text answer.

Syntax

```
void avl::Http_SendRequest_GET
(
 const atl::String& inUrl,
 const atl::Array<atl::String>& inFields,
 const atl::Array<atl::String>& inFieldsData,
 int inTimeout,
 atl::Conditional<atl::String>& outAnswer,
 int& outResponseCode
)
```

Parameters

	Name	Type	Range	Default	Description
→	inUrl	const String&			URL of request. Without parameters.
→	inFields	const Array<String>&			Request data to send. It will be automatically encoded.
→	inFieldsData	const Array<String>&			Request data to send. It will be automatically encoded.
→	inTimeout	int	0 - ∞	60	Request timeout in seconds.
←	outAnswer	Conditional<String>&			Answer text if provided in UTF-8 encoding.
←	outResponseCode	int&			Answer code. Typically 200 for OK.

Description

Filter sends a GET request to the server. Filter waits for the sever text answer.

Examples

Filter executed with parameters:

- **inUrl** = http://localhost/test
- **inFields** = ["param1", "param2"]
- **inFieldsData** = ["1", "2"]

Request sent to the server:

```
GET /test?param1=1&param2=2 HTTP/1.1
Host: localhost
User-Agent: Aurora Vision/1.0
Accept: */*
```

Remarks

Filter only accepts text answers encoded using UTF-8 or plain ASCII. To receive answer in arbitrary format please use Http_SendRequest_GET_ByteBuffer.

Errors

List of possible exceptions:

Error type	Description
DomainError	Different size of inFields and inFieldsData in Http_SendRequest_GET.
DomainError	Empty inUrl in Http_SendRequest_GET.
DomainError	Invalid inUrl in Http_SendRequest_GET.
DomainError	Secured HTTPS connection is not supported Http_SendRequest_GET.

See Also

- Http_SendRequest_POST – Sends a POST request to the server and receives a text answer.
- Http_SendRequest_GET_ByteBuffer – Sends a GET request to server and receives a binary answer.

Http_SendRequest_GET_ByteBuffer

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Sends a GET request to server and receives a binary answer.

Syntax

```
void avl::Http_SendRequest_GET_ByteBuffer
(
 const atl::String& inUrl,
 const atl::Array<atl::String>& inFields,
 const atl::Array<atl::String>& inFieldsData,
 int inTimeout,
 atl::Conditional<avl::ByteBuffer>& outAnswer,
 int& outResponseCode
)
```

Parameters

	Name	Type	Range	Default	Description
→	inUrl	const String&			URL of request. Without parameters.
→	inFields	const Array<String>&			Request data to send. It will be automatically encoded.
→	inFieldsData	const Array<String>&			Request data to send. It will be automatically encoded.
→	inTimeout	int	0 - ∞	60	Request timeout in seconds.
←	outAnswer	Conditional<ByteBuffer>&			Answer as binary byte buffer.
←	outResponseCode	int&			Answer code. Typically 200 for OK.

Description

Filter sends a GET request to the server then waits for the sever answer. Filter can be used to download files from the remote server.

Examples

Filter executed with parameters:

- **inUrl** = http://localhost/test
- **inFields** = ["param1", "param2"]
- **inFieldsData** = ["1", "2"]

Request sent to the server:

```
GET /test?param1=1&param2=2 HTTP/1.1
Host: localhost
User-Agent: Aurora Vision/1.0
Accept: */*
```

Errors

List of possible exceptions:

Error type	Description
DomainError	Different size of inFields and inFieldsData in Http_SendRequest_GET_ByteBuffer.
DomainError	Empty inUrl in Http_SendRequest_GET_ByteBuffer.
DomainError	Invalid inUrl in Http_SendRequest_GET_ByteBuffer.
DomainError	Secured HTTPS connection is not supported Http_SendRequest_GET_ByteBuffer.

See Also

- Http_SendRequest_POST – Sends a POST request to the server and receives a text answer.

Http_SendRequest_POST

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Sends a POST request to the server and receives a text answer.

Syntax

```
void avl::Http_SendRequest_POST
(
 const atl::String& inUrl,
 const atl::Array<atl::String>& inFields,
 const atl::Array<atl::String>& inFieldsData,
 int inTimeout,
 atl::Conditional<atl::String>& outAnswer,
 int& outResponseCode
)
```

Parameters

	Name	Type	Range	Default	Description
→	inUrl	const String&			URL of request.
→	inFields	const Array<String>&			Request data to send. It will be automatically encoded.
→	inFieldsData	const Array<String>&			Request data to send. It will be automatically encoded.
→	inTimeout	int	0 - ∞	60	Request timeout in seconds.
←	outAnswer	Conditional<String>&			Answer text if provided in UTF-8 encoding.
←	outResponseCode	int&			Answer code. Typically 200 for OK.

Description

Filter sends a POST request to the server using **application/x-www-form-urlencoded** format. Filter waits for the sever text answer.

Examples

Filter executed with parameters:

- **inUrl** = http://localhost/test
- **inFields** = ["param1", "param2"]
- **inFieldsData** = ["1", "2"]

Request sent to the server:

```
POST /test HTTP/1.1
Host: localhost
User-Agent: Aurora Vision/1.0
Accept: */*
Content-Length: 17
Content-Type: application/x-www-form-urlencoded

param1=1&param2=2
```

Remarks

Filter only accepts text answers encoded using UTF-8 or plain ASCII. To receive answer in arbitrary format please use Http_SendRequest_POST_ByteBuffer.

Errors

List of possible exceptions:

Error type	Description
DomainError	Different size of inFields and inFieldsData in Http_SendRequest_POST.
DomainError	Empty inUrl in Http_SendRequest_POST.
DomainError	Invalid inUrl in Http_SendRequest_POST.
DomainError	Secured HTTPS connection is not supported in Http_SendRequest_POST.

See Also

- Http_SendRequest_POST_JSON – Sends a POST request in JSON format to the server and receives a text answer.
- Http_SendRequest_POST_ByteBuffer – Sends a POST request to the server and receives a text answer.

Http_SendRequest_POST_ByteBuffer

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Sends a POST request to the server and receives a text answer.

Syntax

```
void avl::Http_SendRequest_POST_ByteBuffer
(
 const atl::String& inUrl,
 const atl::Array<atl::String>& inFields,
 const atl::Array<atl::String>& inFieldsData,
 int inTimeout,
 atl::Conditional<avl::ByteBuffer>& outAnswer,
 int& outResponseCode
)
```

Parameters

	Name	Type	Range	Default	Description
→	inUrl	const String&			URL of request.
→	inFields	const Array<String>&			Request data to send. It will be automatically encoded.
→	inFieldsData	const Array<String>&			Request data to send. It will be automatically encoded.
→	inTimeout	int	0 - ∞	60	Request timeout in seconds.
←	outAnswer	Conditional<ByteBuffer>&			Answer as a binary byte buffer.
←	outResponseCode	int&			Answer code. Typically 200 for OK.

Description

Filter sends a POST request to the server using **application/x-www-form-urlencoded** format. Filter waits for the sever binary answer. Filter can be used to download files from the remote server.

Examples

Filter executed with parameters:

- **inUrl** = http://localhost/test
- **inFields** = ["param1", "param2"]
- **inFieldsData** = ["1", "2"]

Request sent to the server:

```
POST /test HTTP/1.1
Host: localhost
User-Agent: Aurora Vision/1.0
Accept: */*
Content-Length: 17
Content-Type: application/x-www-form-urlencoded

param1=1&param2=2
```

Remarks

Filter only accepts text answers encoded using UTF-8 or plain ASCII. To receive answer in arbitrary format please use Http_SendRequest_POST_ByteBuffer.

Errors

List of possible exceptions:

Error type	Description
DomainError	Different size of inFields and inFieldsData in Http_SendRequest_POST_ByteBuffer.
DomainError	Empty inUrl in Http_SendRequest_POST_ByteBuffer.
DomainError	Invalid inUrl in Http_SendRequest_POST_ByteBuffer.
DomainError	Secured HTTPS connection is not supported in Http_SendRequest_POST_ByteBuffer.

See Also

- Http_SendRequest_POST_JSON – Sends a POST request in JSON format to the server and receives a text answer.

Http_SendRequest_POST_JSON

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Sends a POST request in JSON format to the server and receives a text answer.

Syntax

```
void avl::Http_SendRequest_POST_JSON
(
 const atl::String& inUrl,
 const atl::String& inJsonData,
 int inTimeout,
 atl::Conditional<atl::String>& outAnswer,
 int& outResponseCode
)
```

Parameters

	Name	Type	Range	Default	Description
→	inUrl	const String&			URL of request.
→	inJsonData	const String&			Request JSON to send.
→	inTimeout	int	0 - ∞	60	Request timeout in seconds.
←	outAnswer	Conditional<String>&			Answer text if provided in UTF-8 encoding.
←	outResponseCode	int&			Answer code. Typically 200 for OK.

Description

Filter sends a POST request to the server using **application/json** format. Filter waits for the text sever answer.

Examples

Filter executed with parameters:

- **inUrl** = http://localhost/test
- **inJsonData** = "{'value': 5}"

Request sent to the server:

```
POST /test HTTP/1.1
Host: localhost
User-Agent: Aurora Vision/1.0
Accept: application/json
Content-Type: application/json
charset: utf-8
Content-Length: 12

{'value': 5}
```

Remarks

Filter only accepts text answers encoded using UTF-8 or plain ASCII.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty inUrl in Http_SendRequest_POST_JSON.
DomainError	Invalid inUrl in Http_SendRequest_POST_JSON.
DomainError	Secured HTTPS connection is not supported Http_SendRequest_POST_JSON.

See Also

- Http_SendRequest_POST – Sends a POST request to the server and receives a text answer.

98. Nearest Neighbors

Table of content:

KNN_Classify

Header: AVL.h
Namespace: avl
Module: FoundationPro

Classify data using the KNN classifier.

Syntax

```
void avl::KNN_Classify
(
 const avl::KNNModel& inKNNModel,
 const atl::Array<float>& inFeature,
 const int inK,
 const avl::Metric::Type inDistanceType,
 int& outClass
)
```

Parameters

	Name	Type	Range	Default	Description
→	inKNNModel	const KNNModel&			Trained KNN model
→	inFeature	const Array<float>&			Vector of features
→	inK	const int	1 - +∞		Numbers of neighbors
→	inDistanceType	const Metric::Type			Geometry distance type used to calculate neighbors
←	outClass	int&			

Errors

List of possible exceptions:

Error type	Description
DomainError	The inFeature has different dimensions count than KNNModel in KNN_Classify.
DomainError	The inK is bigger than data samples count in KNN_Classify.
DomainError	Using not trained classifier in KNN_Classify.

KNN_Init

Header: AVL.h
Namespace: avl
Module: FoundationPro

Initializes the KNN classifier.

Syntax

```
void avl::KNN_Init
(
 const int inClassCount,
 const int inDimensionCount,
 avl::KNNModel& outKNNModel
)
```

Parameters

	Name	Type	Range	Default	Description
→	inClassCount	const int	2 - +∞		Number of classes
→	inDimensionCount	const int	1 - +∞		Length of feature vector
←	outKNNModel	KNNModel&			Initialized KNN model

Errors

List of possible exceptions:

Error type	Description
DomainError	At least two classes must be provided.
DomainError	Feature vector should have at least one dimension.

KNN_Train

Header: AVL.h
Namespace: avl
Module: FoundationPro

Trains KNN classifier using sample data.

Syntax

```
void avl::KNN_Train
(
 const avl::KNNModel& inKNNModel,
 const atl::Array<atl::Array<float> >& inFeatures,
 const atl::Array< int >& inClasses,
 avl::KNNModel& outKNNModel
)
```

Parameters

	Name	Type	Default	Description
→	inKNNModel	const KNNModel&		Initialized KNN model
→	inFeatures	const Array<Array<float> >&		Array of features array
→	inClasses	const Array< int >&		Array of classes corresponding to feature array elements
←	outKNNModel	KNNModel&		Trained KNN model

Errors

List of possible exceptions:

Error type	Description
DomainError	Array inClasses contains negative values in KNN_Train.
DomainError	Array inClasses contains values grater than maximal class value in KNN_Train.
DomainError	Input array inClasses is empty in KNN_Train.
DomainError	Input array inFeatures is empty in KNN_Train.
DomainError	Input inFeatures contains array of different sizes in KNN_Train.
DomainError	The inFeatures size is different than inClasses size in KNN_Train.
DomainError	Using uninitialized classifier in KNN_Train.

99. Texture Analysis

Table of content:

LawsFilter

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Filters image with one of the classic LAWS filter.

Syntax

```
void avl::LawsFilter
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 const avl::LawsFilterType::Type inVerticalFilter,
 const avl::LawsFilterType::Type inHorizontalFilter,
 const avl::LawsFilterSize::Type inFilterSize,
 const int inMacroBlockSize,
 const bool inNormalizeLocalContrast,
 avl::Image& outTextureImage,
 avl::Image& outTextureEnergyImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input mono image.
→	inRoi	Optional<const Region&>		NIL	Region of Interest.
→	inVerticalFilter	const LawsFilterType::Type		Level	Vertical part of filter.
→	inHorizontalFilter	const LawsFilterType::Type		Edge	Horizontal part of filter.
→	inFilterSize	const LawsFilterSize::Type			Filter window size.
→	inMacroBlockSize	const int	3 - ∞	15	Macroblock is used to gather generated texture energy.
→	inNormalizeLocalContrast	const bool		False	Whether to normalize image before processing.
←	outTextureImage	Image&			Filtered image.
←	outTextureEnergyImage	Image&			Texture energy (AVG).

Requirements

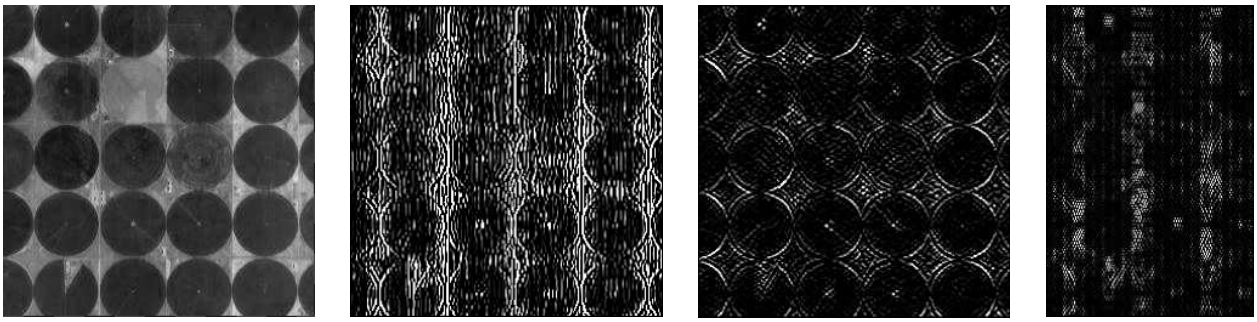
For input **inImage** only pixel formats are supported: 1□uint8.

Read more about pixel formats in Image documentation.

Description

The operation applies texture filters to input image to obtain energy measures.
Laws filter types are: Level, Edge, Wave, Spot, Ripple, Oscillation and Undulation.

Examples



From left: sample image, applied Level and Wave filters, Spot and Edge, Spot and Ripple.

Errors

List of possible exceptions:

Error type	Description
DomainError	Laws filters can be applied to single channel images only.
DomainError	Not supported inImage pixel format in LawsFilter. Supported formats: 1xUInt8.

LinearBinaryPattern

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Creates histogram and map of Linear Binary Patterns (with radius 1 and size 8) of provided image.

Syntax

```
void avl::LinearBinaryPattern
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::LinearBinaryPatternType::Type inPatternType,
 avl::Histogram& outPatternHistogram,
 avl::Image& outTextureImage
)
```

Parameters

	Name	Type	Default	Description
➡	inImage	const Image&		Mono-channel image.
➡	inRoi	Optional<const Region&>	NIL	Region of Interest.
➡	inPatternType	LinearBinaryPatternType::Type		Type of LBP to produce.
⬅	outPatternHistogram	Histogram&		Histogram of LBP codes found in inImage.
⬅	outTextureImage	Image&		LBP map of input image.

Requirements

For input **inImage** only pixel formats are supported: 1□uint8.

Read more about pixel formats in Image documentation.

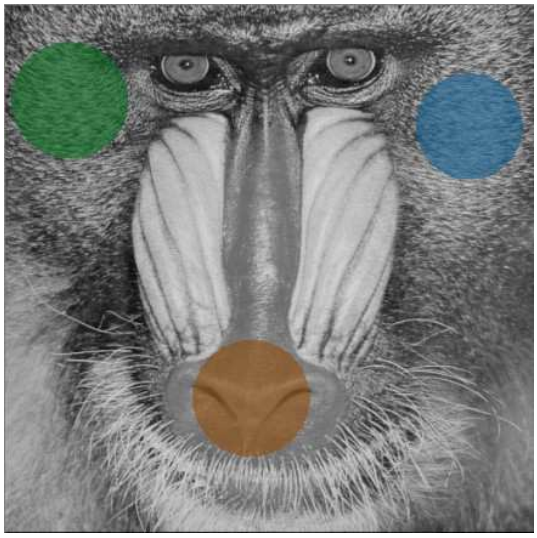
Description

The LPB histogram (treated as a 256 dimensional feature vector) is a texture visual descriptor useful for texture classification. It is obtained by calculating a pattern for each pixel, and then producing a histogram from all these numbers. The pattern is a 8-bit array (coded in uint8) with results of comparisons of the center pixel value to all its neighbours.
Following types of LPB can be calculated:

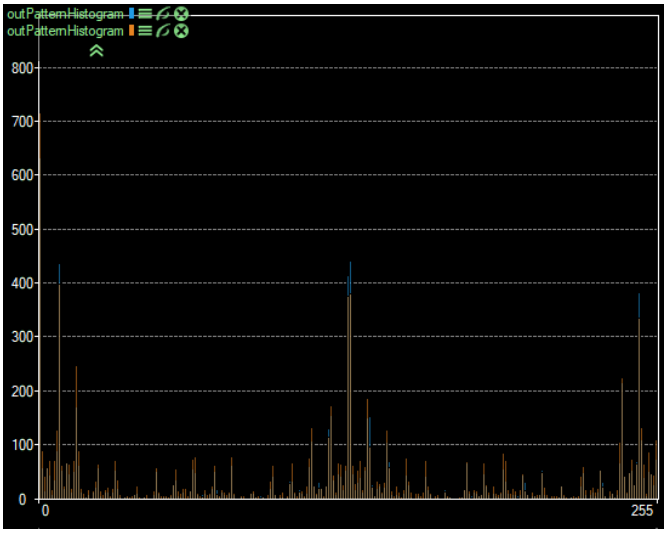
- **Plain**
- **RotationInvariant** - all patterns (and thus histogram bins) that can be obtained by binary shifts from each other are merged into one
- **Uniform** - non-uniform patterns *(these containing more than two 01 or 10 transitions - e.g. 00110010 has 4)* are merged into one bin
- **UniformRotationInvariant** - apply both RotationInvariant and Uniform operations

More information can be found at https://en.wikipedia.org/wiki/Local_binary_patterns

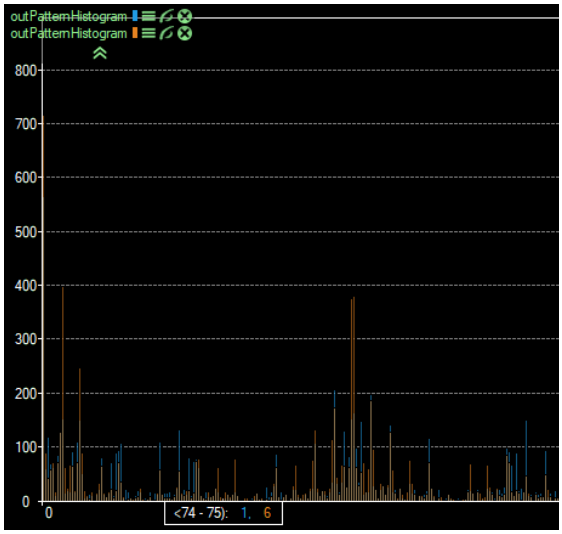
Examples



Source image with following ROIs defined:
top-left fur (reference), top-right fur, and bottom nose.



LBP histograms calculated for top-left and top-right regions (different parts of fur)
*The mean-squared distance between these feature vectors (histograms) is **264,6**.*



LBP histograms calculated for top-left region and bottom region
*The mean-squared distance between these feature vectors is **1***

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported inImage pixel format in LinearBinaryPattern. Supported formats: 1xUInt8.

100. Geometry 3D Interpolations

Table of content:

LerpPoints3D

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Linearly interpolates between two points in 3D.

Syntax

```
void avl::LerpPoints3D
(
 const avl::Point3D& inPoint0,
 const avl::Point3D& inPoint1,
 const float inLambda,
 avl::Point3D& outPoint3D
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inPoint0	const Point3D&			
➡	inPoint1	const Point3D&			
➡	inLambda	const float	-∞ - ∞	0.5f	Interpolation between the input points where 0.0 value is equal to inPoint0 and 1.0 to inPoint1
⬅	outPoint3D	Point3D&			

LerpSegments3D

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Linearly interpolates between two segments in 3D.

Syntax

```
void avl::LerpSegments3D
(
 const avl::Segment3D& inSegment0,
 const avl::Segment3D& inSegment1,
 float inLambda,
 avl::Segment3D& outSegment3D
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inSegment0	const Segment3D&			
➡	inSegment1	const Segment3D&			
➡	inLambda	float	-∞ - ∞	0.5f	Interpolation between the input segments where 0.0 value is equal to inSegment0 and 1.0 to inSegment
⬅	outSegment3D	Segment3D&			

LerpVectors3D

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Linearly interpolates between two 3D vectors.

Syntax

```
void avl::LerpVectors3D
(
 const avl::Vector3D& inVector0,
 const avl::Vector3D& inVector1,
 const float inLambda,
 avl::Vector3D& outVector3D
)
```

Parameters

	Name	Type	Range	Default	Description
→	inVector0	const Vector3D&			
→	inVector1	const Vector3D&			
→	inLambda	const float	-∞ - ∞	0.5f	Interpolation between the input 3D vectors where 0.0 value is equal to inVector0 and 1.0 to inVector1
←	outVector3D	Vector3D&			

101. Regression Analysis

Table of content:

LinearRegression

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes linear regression of given point set.

Syntax

```
void avl::LinearRegression
(
 const atl::Array<float>& inYValues,
 atl::Optional<const atl::Array<float> &> inXValues,
 avl::LinearFunction& outLinearFunction,
 atl::Array<float>& outEstimatedValues,
 atl::Array<float>& outResiduals,
 float& outRSquared
)
```

Parameters

	Name	Type	Default	Description
→	inYValues	const Array<float>&		Sequence of ordinates
→	inXValues	Optional<const Array<float> &>	NIL	Sequence of abscissae, or {0, 1, 2, ...} by default
←	outLinearFunction	LinearFunction&		Linear function approximating the given point set
←	outEstimatedValues	Array<float>&		The result of application of the computed function to the X values
←	outResiduals	Array<float>&		Difference between an input Y value and the corresponding estimated value
←	outRSquared	float&		Coefficient of determination of output function

Description

The operation fits a straight line through the set of points in such a way, that sum of squared distances (residuals) between points and fitted line is as small as possible.
Fitted line parameters are calculated as follows:

$$B = \frac{n\sum_{i=0}^{n} x_i y_i - \sum_{i=0}^{n} x_i \sum_{i=0}^{n} y_i}{n\sum_{i=0}^{n} x_i^2 - \sum_{i=0}^{n} x^2}$$

$$A = \frac{\sum_{i=0}^{n} x_i}{n} - B\frac{\sum_{i=0}^{n} y_i}{n}$$

Errors

List of possible exceptions:

Error type	Description
DomainError	Inconsistent size of arrays in LinearRegression.

LinearRegression_LTE

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes linear regression of given point set using Least Trimmed Error algorithm.

Syntax

```
void avl::LinearRegression_LTE
(
 const atl::Array<float>& inYValues,
 const atl::Optional<const atl::Array<float>&> inXValues,
 int inSeedSubsetSize,
 atl::Optional<int> inEvalSubsetSize,
 avl::LinearFunction& outLinearFunction,
 atl::Array<float>& outEstimatedValues,
 atl::Array<float>& outResiduals,
 atl::Array<float>& outYInliers,
 atl::Array<float>& outXInliers,
 float& outLTError,
 int& diagIterationCount = atl::Dummy<int>()
)
```

Parameters

	Name	Type	Range	Default	Description
→	inYValues	const Array<float>&			Sequence of ordinates
→	inXValues	const Optional<const Array<float>&>		NIL	Sequence of abscissae, or {0, 1, 2, ...} by default
→	inSeedSubsetSize	int	2 - 10	3	Number of points in one combination for getting a sample line
→	inEvalSubsetSize	Optional<int>	3 - ∞	NIL	Number of closest points used for evaluation of a sample line, or Auto if seed points are to be used
←	outLinearFunction	LinearFunction&			Linear function approximating the given point set
←	outEstimatedValues	Array<float>&			The result of application of the computed function to the X values
←	outResiduals	Array<float>&			Difference between an input Y value and the corresponding estimated value
←	outYInliers	Array<float>&			Coordinate of the inlying points of the best LTE line
←	outXInliers	Array<float>&			Coordinate of the inlying points of the best LTE line
←	outLTError	float&			The Least Trimmed Error
🔍	diagIterationCount	int&			Number of combinations considered

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array of points in LinearRegression_LTE.
DomainError	Inconsistent size of arrays in LinearRegression_LTE.

LinearRegression_M

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes linear regression of given point set using selected M-estimator for outlier suppression.

Syntax

```
void avl::LinearRegression_M
(
 const atl::Array<float>& inYValues,
 atl::Optional<const atl::Array<float>&> inXValues,
 avl::MEstimator::Type inOutlierSuppression,
 float inClippingFactor,
 int inIterationCount,
 atl::Optional<const avl::LinearFunction&> inInitialLinearFunction,
 avl::LinearFunction& outLinearFunction,
 atl::Array<float>& outEstimatedValues,
 atl::Array<float>& outResiduals,
 atl::Optional<atl::Array<float>&> outYInliers = atl::NIL,
 atl::Optional<atl::Array<float>&> outXInliers = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inYValues	const Array<float>&			Sequence of ordinates
→	inXValues	Optional<const Array<float>&>		NIL	Sequence of abscissae, or {0, 1, 2, ...} by default
→	inOutlierSuppression	MEstimator::Type			
→	inClippingFactor	float	0.675 - 6.0	2.5f	Multitude of standard deviation within which points are considered inliers
→	inIterationCount	int	0 - ∞	5	Number of iterations of outlier suppressing algorithm
→	inInitialLinearFunction	Optional<const LinearFunction&>		NIL	Initial approximation of the output linear function (if available)
←	outLinearFunction	LinearFunction&			Linear function approximating the given point set
←	outEstimatedValues	Array<float>&			The result of application of the computed function to the X values
←	outResiduals	Array<float>&			Difference between an input Y value and the corresponding estimated value
←	outYInliers	Optional<Array<float>&>		NIL	Coordinate of the inlying points of the best line
←	outXInliers	Optional<Array<float>&>		NIL	Coordinate of the inlying points of the best line

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outYInliers**, **outXInliers**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Inconsistent size of arrays in LinearRegression_M

LinearRegression_RANSAC

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes linear regression of given point set using RANSAC.

Syntax

```
void avl::LinearRegression_RANSAC
(
 const atl::Array<float>& inYValues,
 atl::Optional<const atl::Array<float>& > inXValues,
 atl::Optional<int> inMaxOutlierCount,
 float inMaxInlierDistance,
 atl::Optional<int> inIterationCount,
 avl::LinearFunction& outLinearFunction,
 atl::Array<float>& outEstimatedValues,
 atl::Array<float>& outResiduals
)
```

Parameters

	Name	Type	Default	Description
→	inYValues	const Array<float>&		Sequence of ordinates
→	inXValues	Optional<const Array<float>& >	NIL	Sequence of abscissae, or {0, 1, 2, ...} by default
→	inMaxOutlierCount	Optional<int>	NIL	Determines how many outlier points can be present to end the search
→	inMaxInlierDistance	float		Distance from a line for point to be considered an inlier
→	inIterationCount	Optional<int>	NIL	Number of iterations; Auto means that all point pairs will be used
←	outLinearFunction	LinearFunction&		Linear function approximating the given point set
←	outEstimatedValues	Array<float>&		The result of application of the computed function to the X values
←	outResiduals	Array<float>&		Difference between an input Y value and the corresponding estimated value

Errors

List of possible exceptions:

Error type	Description
DomainError	Inconsistent size of arrays in LinearRegression_RANSAC.

LinearRegression_TheilSen

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes linear regression of given point set using TheilSen algorithm.

Syntax

```
void avl::LinearRegression_TheilSen
(
 const atl::Array<float>& inYValues,
 atl::Optional<const atl::Array<float>&> inXValues,
 avl::TheilSenVariant::Type inVariant,
 avl::LinearFunction& outLinearFunction,
 atl::Array<float>& outEstimatedValues,
 atl::Array<float>& outResiduals
)
```

Parameters

	Name	Type	Default	Description
→	inYValues	const Array<float>&		Sequence of ordinates
→	inXValues	Optional<const Array<float>&>	NIL	Sequence of abscissae, or {0, 1, 2, ...} by default
→	inVariant	TheilSenVariant::Type		Switches between Theil-Sen and Siegel methods
←	outLinearFunction	LinearFunction&		Linear function approximating the given point set
←	outEstimatedValues	Array<float>&		The result of application of the computed function to the X values
←	outResiduals	Array<float>&		Difference between an input Y value and the corresponding estimated value

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty array of points in LinearRegression_TheilSen.
DomainError	Inconsistent size of arrays in LinearRegression_TheilSen.

QuadraticRegression

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes quadratic regression of given point set.

Syntax

```
void avl::QuadraticRegression
(
 const atl::Array<float>& inYValues,
 atl::Optional<const atl::Array<float> &> inXValues,
 avl::QuadraticFunction& outQuadraticFunction,
 atl::Array<float>& outEstimatedValues,
 atl::Array<float>& outResiduals,
 float& outRSquared
)
```

Parameters

	Name	Type	Default	Description
→	inYValues	const Array<float>&		Sequence of ordinates
→	inXValues	Optional<const Array<float> &>	NIL	Sequence of abscissae, or {0, 1, 2, ...} by default
←	outQuadraticFunction	QuadraticFunction&		Quadratic function approximating the given point set
←	outEstimatedValues	Array<float>&		The result of application of the computed function to the X values
←	outResiduals	Array<float>&		Difference between an input Y value and the corresponding estimated value
←	outRSquared	float&		Coefficient of determination of output function

Errors

List of possible exceptions:

Error type	Description
DomainError	Inconsistent size of arrays in QuadraticRegression.

QuadraticRegression_M

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes quadratic regression of given point set using selected M-estimator for outlier suppression.

Syntax

```
void avl::QuadraticRegression_M
(
 const atl::Array<float>& inYValues,
 atl::Optional<const atl::Array<float>&> inXValues,
 avl::MEstimator::Type inOutlierSuppression,
 float inClippingFactor,
 int inIterationCount,
 atl::Optional<const avl::QuadraticFunction&> inInitialQuadraticFunction,
 avl::QuadraticFunction& outQuadraticFunction,
 atl::Array<float>& outEstimatedValues,
 atl::Array<float>& outResiduals,
 atl::Optional<atl::Array<float>&> outYInliers = atl::NIL,
 atl::Optional<atl::Array<float>&> outXInliers = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inYValues	const Array<float>&			Sequence of ordinates
→	inXValues	Optional<const Array<float>&>		NIL	Sequence of abscissae, or {0, 1, 2, ...} by default
→	inOutlierSuppression	MEstimator::Type			
→	inClippingFactor	float	0.675 - 6.0	2.5f	Multitude of standard deviation within which points are considered inliers
→	inIterationCount	int	0 - ∞	5	Number of iterations of outlier suppressing algorithm
→	inInitialQuadraticFunction	Optional<const QuadraticFunction&>		NIL	Initial approximation of the output quadratic function (if available)
←	outQuadraticFunction	QuadraticFunction&			Quadratic function approximating the given point set
←	outEstimatedValues	Array<float>&			The result of application of the computed function to the X values
←	outResiduals	Array<float>&			Difference between an input Y value and the corresponding estimated value
←	outYInliers	Optional<Array<float>&>		NIL	Coordinate of the inlying points of the best parabola
←	outXInliers	Optional<Array<float>&>		NIL	Coordinate of the inlying points of the best parabola

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outYInliers**, **outXInliers**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Inconsistent size of arrays in QuadraticRegression_M.

QuadraticRegression_RANSAC

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes quadratic regression of given point set using RANSAC.

Syntax

```
void avl::QuadraticRegression_RANSAC
(
 const atl::Array<float>& inYValues,
 atl::Optional<const atl::Array<float>& > inXValues,
 atl::Optional<int> inMaxOutlierCount,
 float inMaxInlierDistance,
 atl::Optional<int> inIterationCount,
 avl::QuadraticFunction& outQuadraticFunction,
 atl::Array<float>& outEstimatedValues,
 atl::Array<float>& outResiduals
)
```

Parameters

	Name	Type	Range	Default	Description
→	inYValues	const Array<float>&			Sequence of ordinates
→	inXValues	Optional<const Array<float>& >		NIL	Sequence of abscissae, or {0, 1, 2, ...} by default
→	inMaxOutlierCount	Optional<int>	0 - ∞	NIL	Determines how many outlier points can be present to end the search
→	inMaxInlierDistance	float	0.0 - ∞		Distance from a parabola for point to be considered an inlier
→	inIterationCount	Optional<int>	1 - ∞	NIL	Number of iterations; Auto means that all point triples will be used
←	outQuadraticFunction	QuadraticFunction&			Quadratic function approximating the given point set
←	outEstimatedValues	Array<float>&			The result of application of the computed function to the X values
←	outResiduals	Array<float>&			Difference between an input Y value and the corresponding estimated value

Errors

List of possible exceptions:

Error type	Description
DomainError	Inconsistent size of arrays in QuadraticRegression_RANSAC.

Statistics_OfArray

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes basic statistical information out of an array of real numbers. The array must be not empty.

Syntax

```
void avl::Statistics_OfArray
(
 const atl::Array<float>& inValues,
 const int inOutlierCount,
 const bool inUseSampleFormula,
 atl::Optional<float&> outMean = atl::NIL,
 atl::Optional<float&> outMedian = atl::NIL,
 atl::Optional<float&> outStandardDeviation = atl::NIL,
 atl::Optional<float&> outMinimum = atl::NIL,
 atl::Optional<float&> outMaximum = atl::NIL,
 atl::Optional<float&> outSpread = atl::NIL,
 atl::Optional<float&> outLinearTrend = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inValues	const Array<float>&			Values used to compute statistical informations
→	inOutlierCount	const int	0 - +∞	0	Defines how many outliers should be removed from the input values
→	inUseSampleFormula	const bool		False	Defines, whether to use population, or sample formulas.
←	outMean	Optional<float&>		NIL	Mean of the input values
←	outMedian	Optional<float&>		NIL	Median of the input values
←	outStandardDeviation	Optional<float&>		NIL	Standard deviation of the input values, treated as population
←	outMinimum	Optional<float&>		NIL	Minimum of the input values
←	outMaximum	Optional<float&>		NIL	Maximum of the input values
←	outSpread	Optional<float&>		NIL	Difference between maximum and minimum of the input values
←	outLinearTrend	Optional<float&>		NIL	First parameter of the linear regression function (multiplier)

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outMean**, **outMedian**, **outStandardDeviation**, **outMinimum**, **outMaximum**, **outSpread**, **outLinearTrend**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	No data available on input in Statistics_OfArray.

Statistics_OfLoop

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes basic statistical information out of real numbers appearing in consecutive iterations.

Syntax

```
void avl::Statistics_OfLoop
(
 Statistics_OfLoopState& ioState,
 const float inValue,
 const int inBufferSize,
 const int inOutlierCount,
 const bool inUseSampleFormula,
 atl::Optional<float&> outMean = atl::NIL,
 atl::Optional<float&> outMedian = atl::NIL,
 atl::Optional<float&> outStandardDeviation = atl::NIL,
 atl::Optional<float&> outMinimum = atl::NIL,
 atl::Optional<float&> outMaximum = atl::NIL,
 atl::Optional<float&> outSpread = atl::NIL,
 atl::Optional<float&> outLinearTrend = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	Statistics_OfLoopState&			Object used to maintain state of the function.
→	inValue	const float			Value used to compute statistical informations
→	inBufferSize	const int	1 - +∞	10	Defines how many numbers are taken into account
→	inOutlierCount	const int	0 - +∞	0	Defines how many outliers should be removed from the input values
→	inUseSampleFormula	const bool			Defines, whether to use population, or sample formulas.
←	outMean	Optional<float&>		NIL	Mean of the input values
←	outMedian	Optional<float&>		NIL	Median of the input values
←	outStandardDeviation	Optional<float&>		NIL	Standard deviation of the input values
←	outMinimum	Optional<float&>		NIL	Minimum of the input values
←	outMaximum	Optional<float&>		NIL	Maximum of the input values
←	outSpread	Optional<float&>		NIL	Difference between maximum and minimum of the input values
←	outLinearTrend	Optional<float&>		NIL	First parameter of the linear regression function (multiplier)

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outMean**, **outMedian**, **outStandardDeviation**, **outMinimum**, **outMaximum**, **outSpread**, **outLinearTrend**.

Read more about Optional Outputs.

102. Camera Calibration IO

Table of content:

LoadAnyCameraModel

Header: AVL.h
Namespace: avl
Module: Calibration

Loads serialized AnyCameraModel object from AVDATA file.

Syntax

```
void avl::LoadAnyCameraModel
(
 const atl::File& inFilename,
 avl::AnyCameraModel& outAnyCameraModel
)
```

Parameters

	Name	Type	Default	Description
→	inFilename	const File&		Name of the source file
←	outAnyCameraModel	AnyCameraModel&		Deserialized output AnyCameraModel

LoadRectificationMap

Header: AVL.h
Namespace: avl
Module: Calibration

Loads serialized RectificationMap object from AVDATA file.

Syntax

```
void avl::LoadRectificationMap
(
 const atl::File& inFilename,
 avl::RectificationMap& outRectificationMap
)
```

Parameters

	Name	Type	Default	Description
→	inFilename	const File&		Name of the source file
←	outRectificationMap	RectificationMap&		Deserialized output RectificationMap

LoadRectificationTransform

Header: AVL.h
Namespace: avl
Module: Calibration

Loads serialized RectificationTransform object from AVDATA file.

Syntax

```
void avl::LoadRectificationTransform
(
 const atl::File& inFilename,
 avl::RectificationTransform& outRectificationTransform
)
```

Parameters

	Name	Type	Default	Description
→	inFilename	const File&		Name of the source file
←	outRectificationTransform	RectificationTransform&		Deserialized output LoadRectificationTransform

SaveAnyCameraModel

Header: AVL.h
Namespace: avl
Module: Calibration

Saves serialized AnyCameraModel object as AVDATA file.

Syntax

```
void avl::SaveAnyCameraModel
(
 const avl::AnyCameraModel& inAnyCameraModel,
 const atl::File& inFilename
)
```

Parameters

	Name	Type	Default	Description
➡	inAnyCameraModel	const AnyCameraModel&		AnyCameraModel to be serialized
➡	inFilename	const File&		Name of the target file

SaveRectificationMap

Header: AVL.h
Namespace: avl
Module: Calibration

Saves serialized RectificationMap object as AVDATA file.

Syntax

```
void avl::SaveRectificationMap
(
 const avl::RectificationMap& inRectificationMap,
 const atl::File& inFilename
)
```

Parameters

	Name	Type	Default	Description
➡	inRectificationMap	const RectificationMap&		RectificationMap to be serialized
➡	inFilename	const File&		Name of the target file

SaveRectificationTransform

Header: AVL.h
Namespace: avl
Module: Calibration

Saves serialized RectificationTransform object as AVDATA file.

Syntax

```
void avl::SaveRectificationTransform
(
 const avl::RectificationTransform& inRectificationTransform,
 const atl::File& inFilename
)
```

Parameters

	Name	Type	Default	Description
➡	inRectificationTransform	const RectificationTransform&		RectificationTransform to be serialized
➡	inFilename	const File&		Name of the target file

103. Interoperability

Table of content:

LoadEntities_FromDxf

Header: AVL.h
Namespace: avl
Module: FoundationPro

Loads entities from a DXF file as an array of point arrays.

Syntax

```
void avl::LoadEntities_FromDxf
(
 const atl::File& inDxfFile,
 const float inScale,
 atl::Optional<const atl::Array<atl::String> &> inLayersToDraw,
 avl::Point2D& outOrigin,
 atl::Array<atl::Array<avl::Point2D> >& outEntities,
 atl::Array<atl::String>& diagFileLayersNames
)
```

Parameters

	Name	Type	Range	Default	Description
→	inDxfFile	const File&			
→	inScale	const float	0.0 - ∞	10.0f	Scale (pixels per millimeter)
→	inLayersToDraw	Optional<const Array<String> &>		NIL	Names of layers to draw. Names of all layers are available at the 'diagFileLayersNames' output
←	outOrigin	Point2D&			Location of the origin from DXF on 'outImage'
←	outEntities	Array<Array<Point2D> >&			
🔍	diagFileLayersNames	Array<String>&			Names of all layers in DXF file

LoadImage_FromDxf

Header: AVL.h
Namespace: avl
Module: FoundationPro

Loads data from a DXF file as an image.

Syntax

```
void avl::LoadImage_FromDxf
(
 const atl::File& inDxfFile,
 const float inScale,
 const atl::Optional<avl::Size> inSize,
 const avl::Pixel& inBackgroundColor,
 atl::Optional<const atl::Array<atl::String> &> inLayersToDraw,
 avl::Point2D& outOrigin,
 avl::Image& outImage,
 atl::Array<atl::String>& diagFileLayersNames
)
```

Parameters

	Name	Type	Range	Default	Description
→	inDxfFile	const File&			
→	inScale	const float	0.0 - ∞	10.0f	Scale (pixels per millimeter)
→	inSize	const Optional<Size>		NIL	Size of the output image. Scaled entities from file will be centered on image
→	inBackgroundColor	const Pixel&			
→	inLayersToDraw	Optional<const Array<String> &>		NIL	Names of layers to draw. Names of all layers are available at the 'diagFileLayersNames' output
←	outOrigin	Point2D&			Location of the origin from DXF on 'outImage'
←	outImage	Image&			Output image
🔍	diagFileLayersNames	Array<String>&			Names of all layers in DXF file

Remarks

Partial Support

This filter offers only a partial support of DXF files. Notably, *Polyline* and *MText* objects will have to be converted to *2D Polyline*, *Line* or *Text* objects for them to be displayed. Also, SHX files are not supported.

LoadImage_FromWebsite

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Loads a JPG image from a website.

Syntax

```
void avl::LoadImage_FromWebsite
(
 const atl::String& inAddress,
 atl::Optional<const atl::String&> inLogin,
 atl::Optional<const atl::String&> inPassword,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inAddress	const String&		
→	inLogin	Optional<const String&>	NIL	Login for Basic authorization
→	inPassword	Optional<const String&>	NIL	Password for Basic authorization
←	outImage	Image&		Output image

LoadPoint3DGrid

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Loads entities from a file of one of available types as an array of points in 3D.

Syntax

```
void avl::LoadPoint3DGrid
(
 const atl::File& inFile,
 avl::Point3DFileFormat::Type inFileFormat,
 bool inRemoveDuplicates,
 avl::Point3DGrid& outPoint3DGrid
)
```

Parameters

	Name	Type	Default	Description
→	inFile	const File&		
→	inFileFormat	Point3DFileFormat::Type		
→	inRemoveDuplicates	bool	False	Filter will remove duplicate points in point clouds created from STL files.
←	outPoint3DGrid	Point3DGrid&		

Description

The operation loads an grid of points from a file in one of the standard 3D file formats. Currently the filter supports the following formats:

- STL (*.stl),
- PLY (*.ply).
- PCD (*.pcd).

PLY (**Polygon File Format**) also known as the **Stanford Triangle Format** is a format for storing graphical objects described as a collection of polygons. Both ASCII and binary (big-endian and little-endian) versions are supported. Please note that only vertex coordinates can be loaded. Other elements and properties are omitted.

STL (**STereoLithography**) is a file format commonly used in the stereolithography CAD software. Only binary STL is supported.

PCD (**Point Cloud Data**) is a file format for storing point cloud data.

Errors

List of possible exceptions:

Error type	Description
DomainError	Dimensions x, y and z should have 1 element.
DomainError	Invalid point dimensions in the PCD file - no x, y, z coordinates.
DomainError	Unknown file format in LoadPoint3DGrid.
IoError	An error occurred when mapping the PCD file.
IoError	An error occurred when unmapping the PCD file.
IoError	Error during reading compressed binary file.
IoError	Error during searching for the data section.
IoError	Inconsistent dimensions in PCD file.
IoError	Inconsistent number of parameters in PCD file.
IoError	Inconsistent point definition in PCD file.
IoError	Loading Point3DGrid failed. Too many vertex properties.
IoError	Size of decompressed data does not match with the size stored in the header of PCD file.
IoError	The PCD file is corrupted. The file is is smaller than expected.
IoError	Too few elements in a header entry.
IoError	Too large data type of vertices.
IoError	Too large height.
IoError	Too large width.
IoError	Too many elements of dimension.
IoError	Too many points in PCD file.
IoError	Unable to read PCD file.
IoError	Unknown data type in PCD file
IoError	Unrecognized vertex type.
IoError	Wrong order of header entries in the PCD file.
IoError	Wrong PLY file format. End of header not found.
IoError	Wrong PLY file format. Invalid element definition.
IoError	Wrong PLY file format. Invalid file signature.
IoError	Wrong PLY file format. Invalid format definition.
IoError	Wrong PLY file format. Invalid vertex data.
IoError	Wrong PLY file format. Invalid vertex definition.
IoError	Wrong PLY file format. Invalid vertex property definition.
IoError	Wrong PLY file format. Invalid vertex property type.
IoError	Wrong PLY file format. No vertices found.
IoError	Wrong PLY file format. Unrecognized data encoding.
IoError	Wrong type signature in PCD file.

LoadPoint3DGridWithImage

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Loads entities from a file of one of available types as an array of points in 3D and an image. Assumes an ordered rectangular grid with no holes.

Syntax

```
void avl::LoadPoint3DGridWithImage
(
 const atl::File& inFile,
 avl::Point3DFileFormat::Type inFileFormat,
 atl::Optional<int> inImageWidth,
 atl::Optional<int> inImageHeight,
 avl::Point3DGrid& outPoint3DGrid,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inFile	const File&			
→	inFileFormat	Point3DFileFormat::Type			
→	inImageWidth	Optional<int>	1 - 65535	NIL	An override for the point cloud width
→	inImageHeight	Optional<int>	1 - 65535	NIL	An override for the point cloud height
←	outPoint3DGrid	Point3DGrid&			
←	outImage	Image&			Output image

Errors

List of possible exceptions:

Error type	Description
DomainError	Unknown file format in LoadPoint3DGridWithImage.

LoadSurfaceWithImage

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Loads entities from a file of one of available types as an array of points in 3D and an image. Assumes an unordered grid with possible holes.

Syntax

```
void avl::LoadSurfaceWithImage
(
 const atl::File& inFile,
 avl::Point3DFileFormat::Type inFileFormat,
 const double inXScale,
 const avl::ValueLimits_f64& inXLimits,
 const double inYScale,
 const avl::ValueLimits_f64& inYLimits,
 const double inZOffset,
 const double inZScale,
 avl::PlainType::Type inPointType,
 avl::SurfaceMultipointHeight::Type inMultipointHeight,
 const avl::Pixel& inBackgroundColor,
 avl::Surface& outSurface,
 avl::Image& outImage,
 atl::Optional<double&> outMinX = atl::NIL,
 atl::Optional<double&> outMinY = atl::NIL,
 avl::Region& diagSurfaceValidPointsRegion
)
```

Parameters

	Name	Type	Range	Default	Description
→	inFile	const File&			
→	inFileFormat	Point3DFileFormat::Type			
→	inXScale	const double	0.000001 - ∞	1.0D	
→	inXLimits	const ValueLimits_f64&			
→	inYScale	const double	0.000001 - ∞	1.0D	
→	inYLimits	const ValueLimits_f64&			
→	inZOffset	const double			
→	inZScale	const double	0.000001 - ∞	1.0D	
→	inPointType	PlainType::Type		Real	Type of single surface point Z coordinate
→	inMultipointHeight	SurfaceMultipointHeight::Type		Mean	Determines the Z coordinate of a surface point created from more than one point
→	inBackgroundColor	const Pixel&			
←	outSurface	Surface&			
←	outImage	Image&			Output image
←	outMinX	Optional<double&>		NIL	
←	outMinY	Optional<double&>		NIL	
🔍	diagSurfaceValidPointsRegion	Region&			Region of locations where the surface points are valid

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outMinX**, **outMinY**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Unknown file format in LoadSurfaceWithImage.

SavePoint3DGrid

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Saves Point3DGrid in to file in one of the supported formats.

Syntax

```
void avl::SavePoint3DGrid
(
 const atl::File& inFile,
 const avl::Point3DGrid& inPoint3DGrid,
 avl::Point3DFileFormat::Type inFileFormat,
 avl::StreamMode::Type inStreamMode
)
```

Parameters

	Name	Type	Default	Description
➡	inFile	const File&		
➡	inPoint3DGrid	const Point3DGrid&		
➡	inFileFormat	Point3DFileFormat::Type	PLY	
➡	inStreamMode	StreamMode::Type		

Errors

List of possible exceptions:

Error type	Description
DomainError	Unknown file format in SavePoint3DGrid.
DomainError	Unsupported file format in SavePoint3DGrid.
IoError	Incorrect float conversion.
IoError	Wrong PLY file format. Invalid vertex property type.

104. Histogram IO

Table of content:

LoadHistogram

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Loads serialized Histogram object from AVDATA file.

Syntax

```
void avl::LoadHistogram
(
 const atl::File& inFilename,
 avl::Histogram& outHistogram
)
```

Parameters

	Name	Type	Default	Description
→	inFilename	const File&		Name of the source file
←	outHistogram	Histogram&		Deserialized output Histogram

SaveHistogram

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Saves serialized Histogram object as AVDATA file.

Syntax

```
void avl::SaveHistogram
(
 const avl::Histogram& inHistogram,
 const atl::File& inFilename
)
```

Parameters

	Name	Type	Default	Description
→	inHistogram	const Histogram&		Histogram to be serialized
→	inFilename	const File&		Name of the target file

105. Path IO

Table of content:

LoadPath

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Loads serialized Path object from AVDATA file.

Syntax

```
void avl::LoadPath
(
 const atl::File& inFilename,
 avl::Path& outPath
)
```

Parameters

	Name	Type	Default	Description
→	inFilename	const File&		Name of the source file
←	outPath	Path&		Deserialized output Path

SavePath

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Saves serialized Path object as AVDATA file.

Syntax

```
void avl::SavePath
(
 const avl::Path& inPath,
 const atl::File& inFilename
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Path to be serialized
→	inFilename	const File&		Name of the target file

106. Point3DGrid IO

Table of content:

LoadPoint3DGridObject

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Deserializes a profile from an AVDATA file.

Syntax

```
void avl::LoadPoint3DGridObject
(
 const atl::File& inFilename,
 avl::Point3DGrid& outPoint3DGrid
)
```

Parameters

	Name	Type	Default	Description
→	inFilename	const File&		Name of the source file
←	outPoint3DGrid	Point3DGrid&		Deserialized output Point3DGrid

SavePoint3DGridObject

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Serializes a profile to an AVDATA file.

Syntax

```
void avl::SavePoint3DGridObject
(
 const avl::Point3DGrid& inPoint3DGrid,
 const atl::File& inFilename
)
```

Parameters

	Name	Type	Default	Description
→	inPoint3DGrid	const Point3DGrid&		Point3DGrid to be serialized
→	inFilename	const File&		Name of the target file

107. Profile IO

Table of content:

LoadProfile

Header: AVL.h
Namespace: avl
Module: FoundationPro

Deserializes a profile from an AVDATA file.

Syntax

```
void avl::LoadProfile
(
 const atl::File& inFilename,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Default	Description
→	inFilename	const File&		Name of the source file
←	outProfile	Profile&		Deserialized output Profile

SaveProfile

Header: AVL.h
Namespace: avl
Module: FoundationPro

Serializes a profile to an AVDATA file.

Syntax

```
void avl::SaveProfile
(
 const avl::Profile& inProfile,
 const atl::File& inFilename
)
```

Parameters

	Name	Type	Default	Description
→	inProfile	const Profile&		Profile to be serialized
→	inFilename	const File&		Name of the target file

108. Region IO

Table of content:

LoadRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Loads a region from an AVDATA file.

Syntax

```
void avl::LoadRegion
(
 const atl::File& inFilename,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Default	Description
→	inFilename	const File&		Name of the source file
←	outRegion	Region&		Deserialized output Region

SaveRegion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Saves a serialized region object to an AVDATA file.

Syntax

```
void avl::SaveRegion
(
 const avl::Region& inRegion,
 const atl::File& inFilename
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Region to be serialized
→	inFilename	const File&		Name of the target file

109. Surface IO

Table of content:

LoadSurface

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Deserializes a surface from an AVDATA file.

Syntax

```
void avl::LoadSurface
(
 const atl::File& inFilename,
 avl::Surface& outSurface
)
```

Parameters

	Name	Type	Default	Description
→	inFilename	const File&		Name of the source file
←	outSurface	Surface&		Deserialized output Surface

SaveSurface

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Serializes a surface to an AVDATA file.

Syntax

```
void avl::SaveSurface
(
 const avl::Surface& inSurface,
 const atl::File& inFilename
)
```

Parameters

	Name	Type	Default	Description
→	inSurface	const Surface&		Surface to be serialized
→	inFilename	const File&		Name of the target file

110. Image Vector Transforms

Table of content:

MeasurePixelVectors

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes gradient magnitudes of an image.

Syntax

```
void avl::MeasurePixelVectors
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
←	outImage	Image&		Output image

Requirements

For input **inImage** only pixel formats are supported: 2□uint8, 2□uint16, 2□int8, 2□int16, 2□int32, 2□real.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in MeasurePixelVectors.
DomainError	Not supported inImage pixel format in MeasurePixelVectors. Supported formats: 2xUInt8, 2xUInt16, 2xInt8, 2xInt16, 2xInt32, 2xReal.

NormalizePixelVectors

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Adjusts gradient angles to a given range.

Syntax

```
void avl::NormalizePixelVectors
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::GradientAngleRange::Type inAngleRange,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inAngleRange	GradientAngleRange::Type		
←	outImage	Image&		Output image

Requirements

For input **inImage** only pixel formats are supported: 2□uint8, 2□uint16, 2□int8, 2□int16, 2□int32, 2□real.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in NormalizePixelVectors.
DomainError	Not supported inImage pixel format in NormalizePixelVectors. Supported formats: 2xUInt8, 2xUInt16, 2xInt8, 2xInt16, 2xInt32, 2xReal.

PixelVectorDirAndPresence

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes gradient orientations of an image.

Syntax

```
void avl::PixelVectorDirAndPresence
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
←	outImage	Image&		Output image

Requirements

For input **inImage** only pixel formats are supported: 2□uint8, 2□uint16, 2□int8, 2□int16, 2□int32, 2□real.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in PixelVectorDirAndPresence.
DomainError	Not supported inImage pixel format in PixelVectorDirAndPresence. Supported formats: 2xUInt8, 2xUInt16, 2xInt8, 2xInt16, 2xInt32, 2xReal.

ResizePixelVectors

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Rescales an image treating pixels as vectors.

Syntax

```
void avl::ResizePixelVectors
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 const float inVectorLength,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inImage	const Image&			Input image
→	inRoi	Optional<const Region&>		NIL	Region of interest
→	inVectorLength	const float	0.0 - ∞	255.0f	Desired vector length after normalization
←	outImage	Image&			Rescaled image

Description

The filter treats pixels as a vector (each channel value as a coordinate). The operation scales pixels values to desired vector length:

$$||Pixel|| = \sqrt{Pixel.X^2 + Pixel.Y^2 + Pixel.Z^2 + Pixel.W^2}$$
$$Pixel.X = \frac{Pixel.X}{||Pixel||} \cdot inVectorLength$$
$$Pixel.Y = \frac{Pixel.Y}{||Pixel||} \cdot inVectorLength$$
$$Pixel.Z = \frac{Pixel.Z}{||Pixel||} \cdot inVectorLength$$
$$Pixel.W = \frac{Pixel.W}{||Pixel||} \cdot inVectorLength$$

Remarks

If a vector length of a pixel from an input image is zero, the output value equals zero.

If computed channel value exceeds its range, it is cut to maximal allowed value. Therefore the resulting vector length can be smaller than desired.

Computed values are rounded down to the next integer for integer pixel types.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in ResizePixelVectors.

See Also

- NormalizeImage – Rescales an image linearly, so that its minimum becomes inNewMinimum and the maximum of the remaining pixels becomes inNewMaximum.

RotatePixelVectors

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Rotates image gradients.

Syntax

```
void avl::RotatePixelVectors
(
 const avl::Image& inImage,
 atl::Optional<const avl::Region&> inRoi,
 const float inAngle,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inAngle	const float		
←	outImage	Image&		Output image

Requirements

For input **inImage** only pixel formats are supported: 2□uint8, 2□uint16, 2□int8, 2□int16, 2□int32, 2□real.

Read more about pixel formats in Image documentation.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region exceeds an input image in RotatePixelVectors.
DomainError	Not supported inImage pixel format in RotatePixelVectors. Supported formats: 2xUInt8, 2xUInt16, 2xInt8, 2xInt16, 2xInt32, 2xReal.

111. Multilayer Perceptron

Table of content:

MLP_Init

Header: AVL.h
Namespace: avl
Module: FoundationPro

Creates multilayer perceptron model.

Syntax

```
void avl::MLP_Init
(
 atl::Optional<const atl::Array<int>&> inHiddenLayers,
 avl::ActivationFunction::Type inActivationFunction,
 avl::MlpPreprocessing::Type inPreprocessing,
 atl::Optional<int> inRandomSeed,
 int inInputCount,
 int inOutputCount,
 avl::MlpModel& outMlpModel
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inHiddenLayers	Optional<const Array<int>&>		NIL	Internal structure of MLP network
➡	inActivationFunction	ActivationFunction::Type			Type of activation function used to calculate neural response
➡	inPreprocessing	MpPreprocessing::Type			Method of processing input data before learning
➡	inRandomSeed	Optional<int>	0 - ∞	NIL	Number used as starting random seed
➡	inInputCount	int	1 - ∞	1	MLP network input count
➡	inOutputCount	int	1 - ∞	1	MLP network output count
⬅	outMlpModel	MpModel&			Initialized MlpModel

Description

Filter initializes and sets structure of the MlpModel.

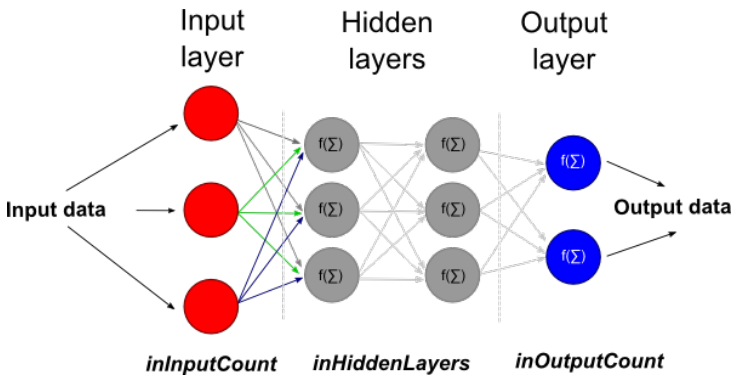


*Image: Internal structure of MlpModel. Function f denotes the **inActivationFunction**.*

Parameter **inHiddenLayers** represents number of neurons in consecutive hidden layers.

The parameter **inActivationFunction** is a function used to calculate internal neuron activation.

The weights of the multilayer perceptron are initialized by a random numbers. Their values depend on **inRandomSeed** value.

Parameters **inInputCount** and **inOutputCount** defines network inputs and outputs count.

See Also

- MLP_Train – Creates and trains multilayer perceptron classifier.
- MLP_Respond – Calculates multilayer perceptron answer.

MLP_Respond

Header: AVL.h
Namespace: avl
Module: FoundationPro

Calculates multilayer perceptron answer.

Syntax

```
void avl::MLP_Respond
(
 const avl::MlpModel& inMlpModel,
 const atl::Array<float>& inInputVector,
 atl::Array<float>& outResponseVector
)
```

Parameters

	Name	Type	Default	Description
→	inMlpModel	const MlpModel&		Trained MlpModel object
→	inInputVector	const Array<float>&		Input vector of features used to calculate classifier response
←	outResponseVector	Array<float>&		Calculated response

Description

The operation calculates response of multilayer perceptron classifier for the input data provided in **inInputVector**.

The parameter **inInputVector** size must be equal to trained MlpModel input count. Input count is set in MLP_Init.

The operation result **outResponseVector** is size of output count provided during MlpModel initializing operation.

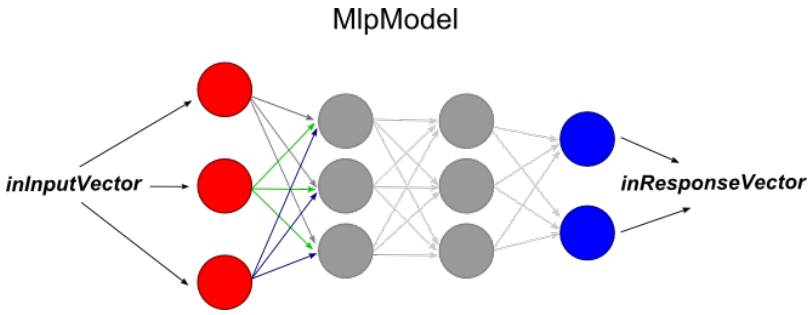


*Image: Visualization of **MLP_Respond** filter.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect or uninitialized MlpModel in MLP_Respond.

See Also

- MLP_Train – Creates and trains multilayer perceptron classifier.
- MLP_Init – Creates multilayer perceptron model.

MLP_Train

Header: AVL.h
Namespace: avl
Module: FoundationPro

Creates and trains multilayer perceptron classifier.

Syntax

```
void avl::MLP_Train
(
 const avl::MlpModel& inMlpModel,
 const atl::Array<atl::Array<float>>& inInputVectorArray,
 const atl::Array<atl::Array<float>>& inResponseVectorArray,
 atl::Optional<const atl::Array<atl::Array<float>>&> inTestInputVectorArray,
 atl::Optional<const atl::Array<atl::Array<float>>&> inTestResponseVectorArray,
 int inIterationCount,
 float inLearningRate,
 float inMomentum,
 atl::Optional<int> inRandomSeed,
 avl::MlpModel& outMlpModel,
 avl::Profile& diagErrorChartLearning,
 avl::Profile& diagErrorChartTesting
)
```

Parameters

	Name	Type	Range	Default	Description
→	inMlpModel	const MlpModel&			Initialized MLP model
→	inInputVectorArray	const Array<Array<float>>&			Array of features used to train model
→	inResponseVectorArray	const Array<Array<float>>&			Array of answers which classifier should get.
→	inTestInputVectorArray	Optional<const Array<Array<float>>&>		NIL	Array of features used to test classifier during training process
→	inTestResponseVectorArray	Optional<const Array<Array<float>>&>		NIL	Array of answers used to test classifier during training process
→	inIterationCount	int	1 - ∞	100	Learning iteration count
→	inLearningRate	float	0.01 - 1.0	1.0f	Learning factor
→	inMomentum	float	0.0 - 1.0	0.01f	Learning momentum ratio
→	inRandomSeed	Optional<int>	0 - ∞	NIL	Number used as starting random seed
←	outMlpModel	MlpModel&			Trained MlpModel
🔍	diagErrorChartLearning	Profile&			Mean error of testing results data during learning process
🔍	diagErrorChartTesting	Profile&			Mean error during learning process

Description

The filter trains multilayer perceptron classifier. The **inInputVectorArray** contains an array of data used to train classifier.

The size of input vector should be constant for each provided input array. Each input vector size must be the same as input count provided in MLP_Init filter during classifier initialization.

The **inResponseVectorArray** contains answer for each data vector provided in **inInputVectorArray**. Size of all response vectors should be the same and equal to output count set in MLP_Init.

The **inLearningRate** determines step size during following function gradient. Too big step size may cause miss of optimization function minimum. Small values may cause learning process too long.

The parameter **inMomentum** defines how learning step results should depend on previous step results.

The **inIterationCount** specifies the number of iterations of the learning process.

Errors

List of possible exceptions:

Error type	Description
DomainError	Different sizes of answer vectors in inResponseVectorArray in MLP_Train.
DomainError	Different sizes of feature vector array and answer vector array on input in MLP_Train.
DomainError	Different sizes of feature vectors in inInputVectorArray in MLP_Train.
DomainError	Empty feature array on input in MLP_Train.
DomainError	Empty inInputVectorArray in MLP_Train
DomainError	Empty inResponseVectorArray in MLP_Train
DomainError	Incorrect or uninitialized MlpModel in MLP_Train.
DomainError	Using uninitialized MlpModel in MLP_Train
DomainError	Wrong size of answer vector in inTestResponseVectorArray in MLP_Train.
DomainError	Wrong size of feature vector in inTestInputVectorArray in MLP_Train.

See Also

- MLP_Respond – Calculates multilayer perceptron answer.
- MLP_Init – Creates multilayer perceptron model.

112. Geometry 2D Normalizations

Table of content:

NormalizeRectangleOrientation

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Changes orientation of the given rectangle according to parameters.

Syntax

```
void avl::NormalizeRectangleOrientation
(
 const avl::Rectangle2D& inRectangle,
 const float inReferenceAngle,
 const avl::RectangleOrientation::Type inRectangleOrientation,
 avl::Rectangle2D& outRectangle
)
```

Parameters

	Name	Type	Default	Description
→	inRectangle	const Rectangle2D&		Input rectangle
→	inReferenceAngle	const float	0.0f	The middle angle of the valid range of the output rectangle's angle
→	inRectangleOrientation	const RectangleOrientation::Type	Horizontal	Orientation of the output rectangle
←	outRectangle	Rectangle2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inRectangle** and **outRectangle**

Read more about In-place Computation.

Description

The operation changes the input rectangle's marked corner so the output rectangle's rotation angle is the lowest possible not lower than **inReferenceAngle** - 90 satisfying the **inRectangleOrientation** parameter at the same time. Some sample parameters configurations are:

- **inReferenceAngle** = 90, **inRectangleOrientation** = Any – the marked corner will have the least y coordinate possible
- **inReferenceAngle** = 0, **inRectangleOrientation** = Any – the marked corner will have the least x coordinate possible
- **inRectangleOrientation** = Horizontal – the output rectangle's width will be not shorter than its height
- **inRectangleOrientation** = Vertical – the output rectangle's height will be not shorter than its width

Examples

NormalizeRectangleOrientation *performed on a sample rectangle with* ***inReferenceAngle*** *= 90 and* ***inRectangleOrientation*** *= Any.*

NormalizeSegmentOrientation_ByBaseOrientation

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Changes orientation of the given segment according to base orientation.

Syntax

```
void avl::NormalizeSegmentOrientation_ByBaseOrientation
(
 const avl::Segment2D& inSegment,
 const float inBaseOrientation,
 avl::Segment2D& outSegment
)
```

Parameters

	Name	Type	Default	Description
→	inSegment	const Segment2D&		
→	inBaseOrientation	const float	0.0f	An angle relative to which angle differences are calculated
←	outSegment	Segment2D&		A segment which orientation is closer to inBaseOrientation

Description

This operation changes the orientation of the **inSegment** segment according to the given angle. The resulting segment has possibly the smallest difference between its orientation and **inBaseOrientation**. In other words, the function returns the same segment as **inSegment** or rotated by 180 degrees - depending on the difference between the orientations.

Examples

Let's consider two cases with the same input segment, but different **inBaseOrientation**. The coordinates of the segment are as follows:

inSegment.Point1.X	99,000
inSegment.Point1.Y	100,000
inSegment.Point2.X	200,000
inSegment.Point2.Y	201,000

1. Let's assume the **inBaseOrientation** to be equal to **90** degrees.

   The orientation of the input segment is 45 degrees, whereas the orientation of the segment rotated by 180 degrees would be 225 degrees. The difference between 90 and 45 degrees is smaller than the one between 90 and 225, so the output segment would be the same as input.

2. Now, let's assume the **inBaseOrientation** to be **180** degrees.

   The orientation of the input segment is still 45 degrees, and the orientation of the segment rotated by 180 degrees is also still 225 degrees. But now, the difference between 90 and 45 degrees is greater than the one between 90 and 225, so the output segment would be rotated by 180 degrees. The vertexes of the output segment would be swapped as follows:

inSegment.Point2.X	200,000
inSegment.Point2.Y	201,000
inSegment.Point1.X	99,000
inSegment.Point1.Y	100,000

See Also

- SegmentOrientation – Computes the orientation of a segment.
- NormalizeSegmentOrientation_ByPointDistance – Changes orientation of the given segment according to distance to the given point.
- NormalizeSegmentOrientation_ByCoordinate – Changes orientation of the given segment according to coordinates along selected axis.

NormalizeSegmentOrientation_ByCoordinate

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Changes orientation of the given segment according to coordinates along selected axis.

Syntax

```
void avl::NormalizeSegmentOrientation_ByCoordinate
(
 const avl::Segment2D& inSegment,
 const avl::Axis::Type inAxis,
 avl::Segment2D& outSegment
)
```

Parameters

	Name	Type	Default	Description
→	inSegment	const Segment2D&		
→	inAxis	const Axis::Type	X	An axis relative to which coordinates are compared
←	outSegment	Segment2D&		A segment in which the first point will have smaller coordinate along the selected axis

Description

This operation changes the orientation of the **inSegment** segment according to the coordinates of the input segment's vertices. The first point in the **outSegment** will have smaller coordinate along selected axis.

Examples

Let's consider two cases with the same input segment, but different **inAxis**. The coordinates of the segment are as follows:

inSegment.Point1.X 201,000

inSegment.Point1.Y 100,000

inSegment.Point2.X 100,000

inSegment.Point2.Y 199,000

1. Let's assume the **inAxis** to be **X** axis.

   The coordinates along X axis are 201 and 100. Obviously, 100 is smaller than 201, so the resulting segment will have reversed vertices. The coordinates will be as follows:

   inSegment.Point2.X 100,000

   inSegment.Point2.Y 199,000

   inSegment.Point1.X 201,000

   inSegment.Point1.Y 100,000

2. Now, let's assume the **inAxis** to be **Y** axis.

   The coordinates along Y axis are 100 and 199. As in the previous case, 100 is smaller than 199, so the resulting segment will have the same coordinates as **inSegment**.

See Also

- SegmentOrientation – Computes the orientation of a segment.
- NormalizeSegmentOrientation_ByPointDistance – Changes orientation of the given segment according to distance to the given point.
- NormalizeSegmentOrientation_ByBaseOrientation – Changes orientation of the given segment according to base orientation.

NormalizeSegmentOrientation_ByPointDistance

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Changes orientation of the given segment according to distance to the given point.

Syntax

```
void avl::NormalizeSegmentOrientation_ByPointDistance
(
 const avl::Segment2D& inSegment,
 const avl::Point2D& inReferencePoint,
 avl::Segment2D& outSegment
)
```

Parameters

	Name	Type	Default	Description
→	inSegment	const Segment2D&		
→	inReferencePoint	const Point2D&		A point relative to which distances are measured
←	outSegment	Segment2D&		A segment in which the first point is closer to inReferencePoint

Description

This operation changes the orientation of the **inSegment** segment according to the distance between the input segment's vertices and given **inReferencePoint**. The resulting segment has the first vertex closer to the **inReferencePoint**.

Examples

Let's consider two cases with the same input segment, but different **inReferencePoint**. The coordinates of the segment are as follows:

inSegment.Point1.X 99,000

inSegment.Point1.Y 100,000

inSegment.Point2.X 200,000

inSegment.Point2.Y 201,000

1. Let's assume the **inReferencePoint** to have coordinates **X = 99, Y = 0**.

   The distance between the first vertex of the segment (X = 99, Y = 100) is equal to 100, whereas the distance between the second vertex of the segment (X = 200, Y = 201) is about 225. That means the first vertex is closer to the **inReferencePoint**, so the resulting segment will have the same coordinates as the input segment.
2. Now, let's assume the **inReferencePoint** to have coordinates **X = 300, Y = 201**.

   The distance between the first vertex of the segment (X = 99, Y = 100) is about 225, whereas the distance between the second vertex of the segment (X = 200, Y = 201) is equal to 100. That means the second vertex is closer to **inReferencePoint**, so the resulting segment will have reversed vertices. Their coordinates will be as follows:

   inSegment.Point2.X 200,000

   inSegment.Point2.Y 201,000

   inSegment.Point1.X 99,000

   inSegment.Point1.Y 100,000

See Also

- SegmentOrientation – Computes the orientation of a segment.
- NormalizeSegmentOrientation_ByCoordinate – Changes orientation of the given segment according to coordinates along selected axis.
- NormalizeSegmentOrientation_ByBaseOrientation – Changes orientation of the given segment according to base orientation.

RemoveInvalidPoints

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Removes invalid points from an array (i.e. points with NaN or INF coordinates).

Syntax

```
void avl::RemoveInvalidPoints
(
 const atl::Array<avl::Point2D>& inPoints,
 atl::Array<avl::Point2D>& outPoints
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		
←	outPoints	Array<Point2D>&		

ReorientRectangle

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Changes orientation of the given rectangle by multiple of 90 degrees.

Syntax

```
void avl::ReorientRectangle
(
 const avl::Rectangle2D& inRectangle,
 const int inRotationCount,
 avl::Rectangle2D& outRectangle
)
```

Parameters

	Name	Type	Default	Description
→	inRectangle	const Rectangle2D&		Input rectangle
→	inRotationCount	const int	0	Defines by what multiplicity of 90 degrees rectangle orientation should be changed
←	outRectangle	Rectangle2D&		

In-place Processing

This function supports in-place data processing - you can pass the same reference to **inRectangle** and **outRectangle**

Read more about In-place Computation.

113. Path Features

Table of content:

PathArrayPoints

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts an array of paths to an array of points.

Syntax

```
void avl::PathArrayPoints
(
 const atl::Array<avl::Path>& inPaths,
 atl::Array<avl::Point2D>& outPoints
)
```

Parameters

	Name	Type	Default	Description
→	inPaths	const Array<Path>&		
←	outPoints	Array<Point2D>&		

PathAverageTurnAngle

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes the average absolute turn angle of a path per unit of length.

Syntax

```
void avl::PathAverageTurnAngle
(
 const avl::Path& inPath,
 float& outAverageTurnAngle
)
```

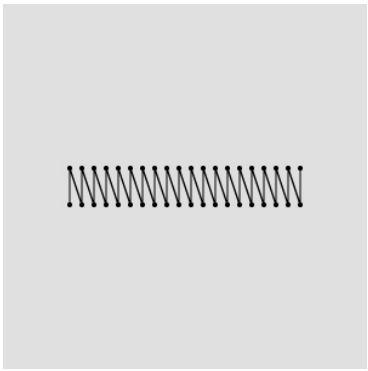
Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
←	outAverageTurnAngle	float&		

Description

The operation measures acuteness of the path turn angles. It is computed as the sum of absolute values of path turn angles at path characteristic points divided by length of the path.

Examples



PathAverageTurnAngle *of the sample path equals to 161.57.*

PathAverageTurnAngle *of the sample path equals to 81.16.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Input path contains not enough points in PathAverageTurnAngle.

See Also

- PathTurnAngleProfile – Computes the profile of turn angles at characteristic points of a path.

PathBoundingBox

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the smallest box containing a path.

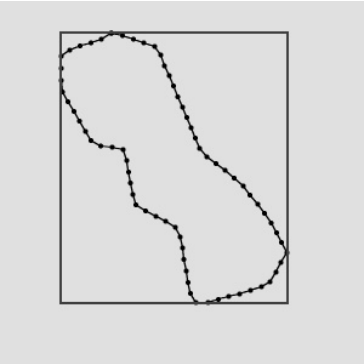
Syntax

```
void avl::PathBoundingBox
(
 const avl::Path& inPath,
 avl::Box& outBoundingBox
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
←	outBoundingBox	Box&		

Examples



*The resulting **outBoundingBox** box drawn onto the sample path.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty path on input in PathBoundingBox.

See Also

- PathBoundingCircle – Computes the smallest circle enclosing a path.
- RegionBoundingBox – Computes the smallest box containing a region.

PathBoundingBox_OrNil

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the smallest box containing a path; returns NIL if the path is empty.

Syntax

```
void avl::PathBoundingBox_OrNil
(
 const avl::Path& inPath,
 atl::Conditional<avl::Box>& outBoundingBox
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
←	outBoundingBox	Conditional<Box>&		

PathBoundingCircle

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the smallest circle enclosing a path.

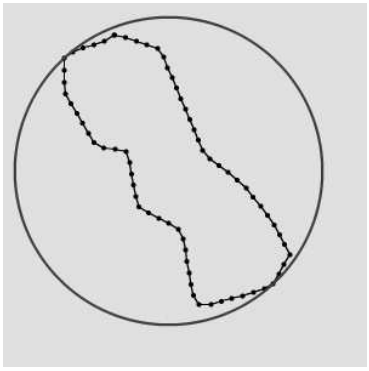
Syntax

```
void avl::PathBoundingCircle
(
 const avl::Path& inPath,
 avl::Circle2D& outBoundingCircle
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
←	outBoundingCircle	Circle2D&		

Examples



*The resulting **outBoundingCircle** circle drawn onto the sample path.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty path on input in PathBoundingCircle.

See Also

- PathBoundingBox – Computes the smallest box containing a path.
- RegionBoundingCircle – Computes the smallest circle enclosing a region.

PathBoundingCircle_OrNil

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the smallest circle enclosing a path; returns NIL if the path is empty.

Syntax

```
void avl::PathBoundingCircle_OrNil
(
 const avl::Path& inPath,
 atl::Conditional<avl::Circle2D>& outBoundingCircle
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
←	outBoundingCircle	Conditional<Circle2D>&		

PathBoundingParallelogram

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes the smallest parallelogram containing a path.

Syntax

```
void avl::PathBoundingParallelogram
(
 const avl::Path& inPath,
 avl::BoundingRectangleFeature::Type inBoundingParallelogramFeature,
 avl::Path& outBoundingParallelogram,
 atl::Optional<avl::Point2D&> outCenter = atl::NIL,
 atl::Optional<float&> outLongSide = atl::NIL,
 atl::Optional<float&> outShortSide = atl::NIL,
 atl::Optional<float&> outAngle = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
→	inBoundingParallelogramFeature	BoundingRectangleFeature::Type	MinimalArea	Determines what kind of bounding parallelogram will be computed
←	outBoundingParallelogram	Path&		Smallest bounding parallelogram of the input points
←	outCenter	Optional<Point2D&>	NIL	Center of the bounding parallelogram
←	outLongSide	Optional<float&>	NIL	Length of the bounding parallelogram long side
←	outShortSide	Optional<float&>	NIL	Length of the bounding parallelogram short side
←	outAngle	Optional<float&>	NIL	Angle of the bounding parallelogram

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outCenter**, **outLongSide**, **outShortSide**, **outAngle**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty path on input in PathBoundingParallelogram.

PathBoundingRectangle

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the smallest rectangle containing a path.

Syntax

```
void avl::PathBoundingRectangle
(
 const avl::Path& inPath,
 avl::BoundingRectangleFeature::Type inBoundingRectangleFeature,
 float inReferenceAngle,
 avl::RectangleOrientation::Type inRectangleOrientation,
 avl::Rectangle2D& outBoundingRectangle,
 atl::Optional<avl::Point2D&> outCenter = atl::NIL,
 atl::Optional<float&> outLongSide = atl::NIL,
 atl::Optional<float&> outShortSide = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
→	inBoundingRectangleFeature	BoundingRectangleFeature::Type	MinimalArea	Determines what kind of bounding rectangle will be computed
→	inReferenceAngle	float	0.0f	The middle angle of the valid range of the output rectangle's angle
→	inRectangleOrientation	RectangleOrientation::Type	Horizontal	Orientation of the output rectangle
←	outBoundingRectangle	Rectangle2D&		The smallest bounding rectangle of the input path
←	outCenter	Optional<Point2D&>	NIL	Center of the bounding rectangle
←	outLongSide	Optional<float&>	NIL	Length of the bounding rectangle long side
←	outShortSide	Optional<float&>	NIL	Length of the bounding rectangle short side

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outCenter**, **outLongSide**, **outShortSide**.

Read more about Optional Outputs.

Description

The filter computes a rectangle with the smallest possible selected feature that contains all points of the given path. The angle of the resulting rectangle is then normalized as in the NormalizeRectangleOrientation filter.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty path on input in PathBoundingRectangle.

See Also

- NormalizeRectangleOrientation – Changes orientation of the given rectangle according to parameters.

PathBoundingRectangle_FixedAngle

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the smallest rectangle with the given orientation angle containing a path.

Syntax

```
void avl::PathBoundingRectangle_FixedAngle
(
 const avl::Path& inPath,
 float inAngle,
 avl::Rectangle2D& outBoundingRectangle,
 atl::Optional<avl::Point2D&> outCenter = atl::NIL,
 atl::Optional<float&> outLongSide = atl::NIL,
 atl::Optional<float&> outShortSide = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
→	inAngle	float		Expected angle of the resulting rectangle
←	outBoundingRectangle	Rectangle2D&		Smallest bounding rectangle of the input path
←	outCenter	Optional<Point2D&>	NIL	Center of the bounding rectangle
←	outLongSide	Optional<float&>	NIL	Length of the bounding rectangle long side
←	outShortSide	Optional<float&>	NIL	Length of the bounding rectangle short side

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outCenter**, **outLongSide**, **outShortSide**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty path on input in PathBoundingRectangle_FixedAngle.

PathBoundingRectangle_FixedAngle_OrNil

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the smallest rectangle with the given orientation angle containing a path; returns NIL when the path is empty.

Syntax

```
void avl::PathBoundingRectangle_FixedAngle_OrNil
(
 const avl::Path& inPath,
 float inAngle,
 atl::Conditional<avl::Rectangle2D>& outBoundingRectangle,
 atl::Conditional<avl::Point2D>& outCenter,
 atl::Conditional<float>& outLongSide,
 atl::Conditional<float>& outShortSide
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
→	inAngle	float		Expected angle of the resulting rectangle
←	outBoundingRectangle	Conditional<Rectangle2D>&		Smallest bounding rectangle of the input path
←	outCenter	Conditional<Point2D>&		Center of the bounding rectangle
←	outLongSide	Conditional<float>&		Length of the bounding rectangle long side
←	outShortSide	Conditional<float>&		Length of the bounding rectangle short side

PathBoundingRectangle_OrNil

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the smallest rectangle containing a path; returns NIL when the path is empty.

Syntax

```
void avl::PathBoundingRectangle_OrNil
(
 const avl::Path& inPath,
 avl::BoundingRectangleFeature::Type inBoundingRectangleFeature,
 float inReferenceAngle,
 avl::RectangleOrientation::Type inRectangleOrientation,
 atl::Conditional<avl::Rectangle2D>& outBoundingRectangle,
 atl::Conditional<avl::Point2D>& outCenter,
 atl::Conditional<float>& outLongSide,
 atl::Conditional<float>& outShortSide
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
→	inBoundingRectangleFeature	BoundingRectangleFeature::Type	MinimalArea	Determines what kind of bounding rectangle will be computed
→	inReferenceAngle	float	0.0f	The middle angle of the valid range of the output rectangle's angle
→	inRectangleOrientation	RectangleOrientation::Type	Horizontal	Orientation of the output rectangle
←	outBoundingRectangle	Conditional<Rectangle2D>&		Smallest bounding rectangle of the input path
←	outCenter	Conditional<Point2D>&		Center of the bounding rectangle
←	outLongSide	Conditional<float>&		Length of the bounding rectangle long side
←	outShortSide	Conditional<float>&		Length of the bounding rectangle short side

PathCaliperDiameter

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the longest and the shortest width of the input path measured as distance between parallel lines containing the whole path.

Syntax

```
void avl::PathCaliperDiameter
(
 const avl::Path& inPath,
 atl::Optional<avl::Segment2D&> outMinDiameter = atl::NIL,
 atl::Optional<float&> outMinDiameterLength = atl::NIL,
 atl::Optional<avl::Segment2D&> outMaxDiameter = atl::NIL,
 atl::Optional<float&> outMaxDiameterLength = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
←	outMinDiameter	Optional<Segment2D&>	NIL	
←	outMinDiameterLength	Optional<float&>	NIL	
←	outMaxDiameter	Optional<Segment2D&>	NIL	
←	outMaxDiameterLength	Optional<float&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outMinDiameter**, **outMinDiameterLength**, **outMaxDiameter**, **outMaxDiameterLength**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty path on input in PathCaliperDiameter.

PathConvexHull

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the smallest convex shape that contains the given path.

Syntax

```
void avl::PathConvexHull
(
 const avl::Path& inPath,
 avl::Path& outConvexHull
)
```

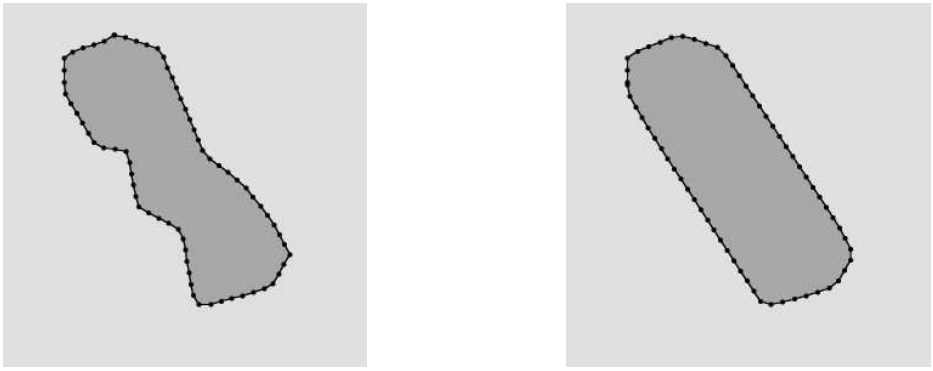
Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
←	outConvexHull	Path&		A closed path representing the computed convex hull

Description

The operation computes the smallest of all convex shapes containing the given path.

Examples



***PathConvexHull** run on a sample path*

See Also

- PolygonConvexity – Computes the area of a polygon divided by the area of its convex hull.
- RegionConvexHull – Computes the smallest convex region containing the input region.

PathDiameter

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Finds the longest segment connecting two characteristic points of a path.

Syntax

```
void avl::PathDiameter
(
 const avl::Path& inPath,
 atl::Optional<avl::Segment2D&> outDiameter,
 atl::Optional<float&> outDiameterLength
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
←	outDiameter	Optional<Segment2D&>		
←	outDiameterLength	Optional<float&>		

Optional Outputs

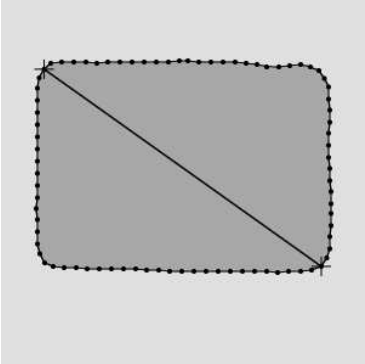
The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outDiameter**, **outDiameterLength**.

Read more about Optional Outputs.

Description

The operation finds the most distant pair of points within a path and returns the distance between them (**outDiameterLength**) and the segment representing the diameter (**outDiameter**). If there is more than one pair of maximal distance, the returned segment will correspond to one of them. The orientation of the resulting **outDiameter** is always between 0 and 180 degrees.

Examples



*The resulting **outDiameterLength** = 280.0921, **outDiameter** segment was drawn onto the sample path.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty path on input in PathDiameter.

See Also

- RegionDiameter – Computes the longest segment connecting two pixels contained in region and its length.

PathEndpoints

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns the two endpoints of a path.

Syntax

```
void avl::PathEndpoints
(
 const avl::Path& inPath,
 avl::Point2D& outFirstPoint,
 avl::Point2D& outLastPoint
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
←	outFirstPoint	Point2D&		
←	outLastPoint	Point2D&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty path on input in PathEndpoints.

PathLength

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the total length of the input path.

Syntax

```
void avl::PathLength
(
 const avl::Path& inPath,
 float& outLength
)
```

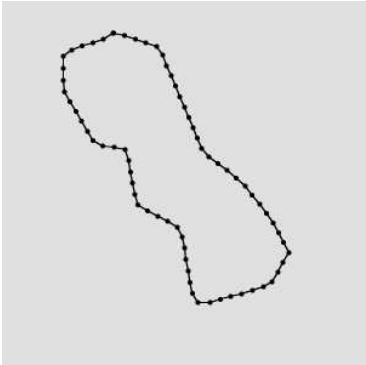
Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
←	outLength	float&		

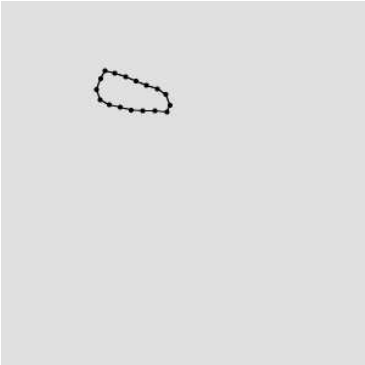
Description

The operation computes the length of the path defined as the sum of the length of the segments it consists of.

Examples



Length of the sample path equals to 633.999.



Length of the sample path equals to 152.002.

See Also

- PathSize – Returns the number of characteristic points on the input path.

PathMassCenter

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the average of the path points (all, not only characteristic ones).

Syntax

```
void avl::PathMassCenter
(
 const avl::Path& inPath,
 avl::Point2D& outMassCenter
)
```

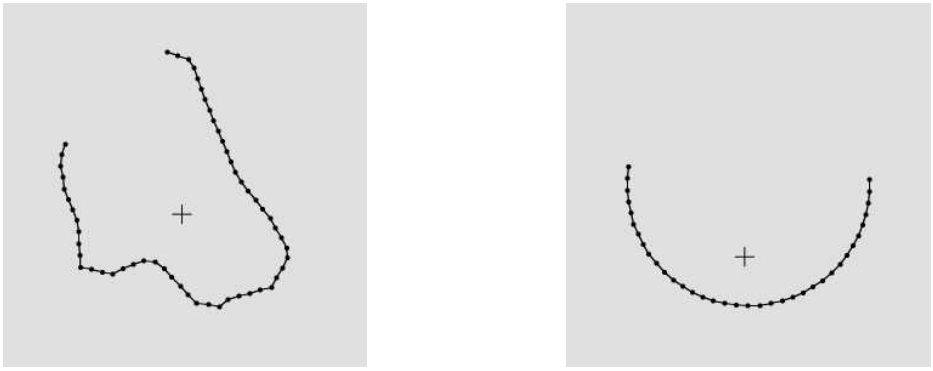
Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
←	outMassCenter	Point2D&		

Description

The operation computes the mass center of a path by averaging its points.

Examples



*The resulting **outMassCenter**s drawn onto the sample paths.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty path on input in PathMassCenter.

See Also

- RegionMassCenter – Computes a point with coordinates equal to the average coordinates of the region's pixels.
- PolygonMassCenter – Computes the mass center of polygon.

PathMassCenter_OrNil

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the average of the path points (all, not only characteristic ones); returns NIL if the path is empty.

Syntax

```
void avl::PathMassCenter_OrNil
(
 const avl::Path& inPath,
 atl::Conditional<avl::Point2D>& outMassCenter
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
←	outMassCenter	Conditional<Point2D>&		

PathSegments

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Converts a path to an array of line segments.

Syntax

```
void avl::PathSegments
(
 const avl::Path& inPath,
 atl::Array<avl::Segment2D>& outSegments
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
←	outSegments	Array<Segment2D>&		

PathSelfIntersections

Header: AVL.h
Namespace: avl
Module: FoundationPro

Checks if a path has any self-intersections and computes all of them.

Syntax

```
void avl::PathSelfIntersections
(
 const avl::Path& inPath,
 atl::Array<avl::Point2D>& outSelfIntersections,
 atl::Optional<atl::Array<int>&> outFirstSegmentIndices = atl::NIL,
 atl::Optional<atl::Array<int>&> outSecondSegmentIndices = atl::NIL,
 atl::Optional<bool&> outIsSelfIntersecting = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
←	outSelfIntersections	Array<Point2D>&		
←	outFirstSegmentIndices	Optional<Array<int>&>	NIL	First indices of the segments of the path which generate found intersection points
←	outSecondSegmentIndices	Optional<Array<int>&>	NIL	Second indices of the segments of the path which generate found intersection points
←	outIsSelfIntersecting	Optional<bool&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outFirstSegmentIndices**, **outSecondSegmentIndices**, **outIsSelfIntersecting**.

Read more about Optional Outputs.

PathSize

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Returns the number of characteristic points on the input path.

Syntax

```
void avl::PathSize
(
 const avl::Path& inPath,
 int& outSize
)
```

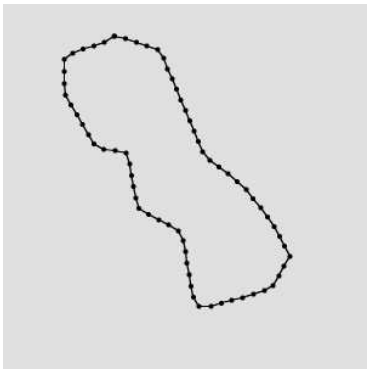
Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
←	outSize	int&		

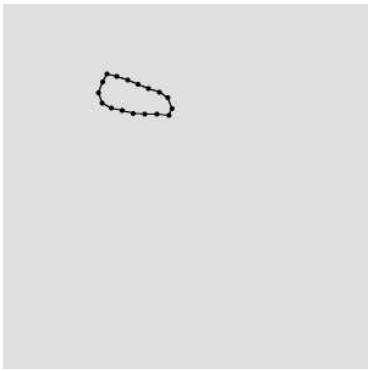
Description

The operation computes the size of the path defined as the number of its characteristic points.

Examples



Size of the sample path equals to 69.



Size of the sample path equals to 17.

See Also

- PathLength – Computes the total length of the input path.

PathTurnAngleLocalMaxima

Header: AVL.h
Namespace: avl
Module: FoundationPro

Finds the local maxima of the profile of turn angles of a path.

Applications: Detection of feature points related to object corners.

Syntax

```
void avl::PathTurnAngleLocalMaxima
(
 const avl::Path& inPath,
 avl::TurnAngleDirection::Type inAllowedTurnDirection,
 avl::TurnAnglePrecision::Type inResultPrecision,
 const float inMinTurnAngle,
 const float inMinDistance,
 const float inSmoothingStdDev,
 atl::Array<float>& outTurnAngleMaximaIndices,
 atl::Array<avl::Point2D>& outTurnAngleMaximaPoints,
 atl::Array<float>& outTurnAngleMaximaAngles,
 avl::Path& diagSmoothedPath,
 avl::Profile& diagTurnAngleProfile
)
```

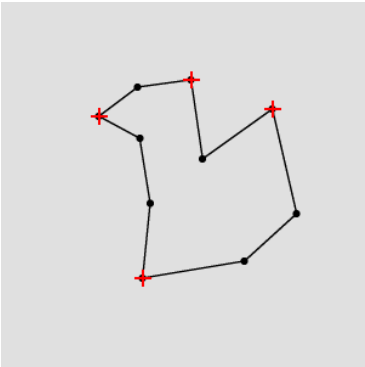
Parameters

	Name	Type	Range	Default	Description
→	inPath	const Path&			Input path
→	inAllowedTurnDirection	TurnAngleDirection::Type		All	Allows to detect only left-turns, only right-turns or both
→	inResultPrecision	TurnAnglePrecision::Type			Switches between pixel-precise or subpixel-precise detection of the found maxima
→	inMinTurnAngle	const float	0.0 - 180.0	30.0f	Minimal value of a relevant angle
→	inMinDistance	const float	0.0 - ∞	0.0f	Minimal distance on the path between two local maxima assuming each path segment has unit length
→	inSmoothingStdDev	const float	0.0 - ∞	0.6f	Standard deviation of the gaussian smoothing applied to the input path
←	outTurnAngleMaximaIndices	Array<float>&			Indices of found local maxima
←	outTurnAngleMaximaPoints	Array<Point2D>&			Found local maxima of turn angle profile of the smoothed input path
←	outTurnAngleMaximaAngles	Array<float>&			Turn angles of found local maxima
🔍	diagSmoothedPath	Path&			Input path smoothed with gaussian kernel
🔍	diagTurnAngleProfile	Profile&			Profile of turn angles at characteristic points of the smoothed input path

Description

The operation computes the profile of turn angles of a path and finds the local maxima of the profile having value at least **inMinTurnAngle**. The **inAllowedTurnDirection** parameter restricts kind of turns taken into consideration (left-turns only, right-turns only or both), while **inResultPrecision** determines the precision of the resulting maxima points. Found maxima have to be at least **inMinDistance** from each other (the distance is measured along the input path assuming each path segment has unit length).

Examples



PathTurnAngleLocalMaxima *run on the sample path with* ***inAllowedTurnDirection*** *= Right*

See Also

- PathTurnAngleMaximum – Finds the maximum of the profile of turn angles of a path.

PathTurnAngleMaximum

Header: AVL.h
Namespace: avl
Module: FoundationPro

Finds the maximum of the profile of turn angles of a path.

Syntax

```
void avl::PathTurnAngleMaximum
(
 const avl::Path& inPath,
 avl::TurnAngleDirection::Type inAllowedTurnDirection,
 avl::TurnAnglePrecision::Type inResultPrecision,
 float& outMaximumTurnAngleIndex,
 avl::Point2D& outMaximumTurnAnglePoint,
 float& outMaximumTurnAngleAngle
)
```

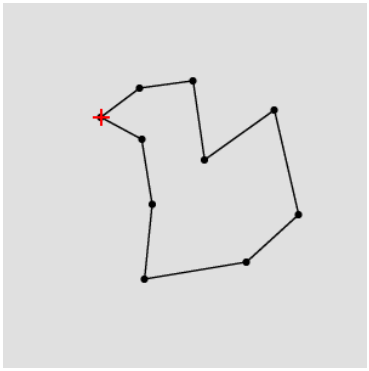
Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
→	inAllowedTurnDirection	TurnAngleDirection::Type	All	Allows to detect only left-turns, only right-turns or both
→	inResultPrecision	TurnAnglePrecision::Type		Switches between pixel-precise or subpixel-precise detection of the found maximum
←	outMaximumTurnAngleIndex	float&		Index of found maximum
←	outMaximumTurnAnglePoint	Point2D&		Found point of a path with maximum turn angle
←	outMaximumTurnAngleAngle	float&		Maximal turn angle of the input path

Description

The operation computes the profile of turn angles of a path and finds the maximum of the profile. The **inAllowedTurnDirection** parameter restricts kind of turns taken into consideration (left-turns only, right-turns only or both), while **inResultPrecision** determines the precision of the resulting maximum point.

Examples



PathTurnAngleMaximum *run on the sample path with* ***inAllowedTurnDirection*** *= Right*

Errors

List of possible exceptions:

Error type	Description
DomainError	Path with no turn angle on input in PathTurnAngleMaximum.

See Also

- PathTurnAngleLocalMaxima – Finds the local maxima of the profile of turn angles of a path.

PathTurnAngleMaximum_OrNil

Header: AVL.h
Namespace: avl
Module: FoundationPro

Finds the maximum of the profile of turn angles of a path; returns NIL if the path has no turn angles.

Syntax

```
void avl::PathTurnAngleMaximum_OrNil
(
 const avl::Path& inPath,
 avl::TurnAngleDirection::Type inAllowedTurnDirection,
 avl::TurnAnglePrecision::Type inResultPrecision,
 atl::Conditional<float>& outMaximumTurnAngleIndex,
 atl::Conditional<avl::Point2D>& outMaximumTurnAnglePoint,
 atl::Conditional<float>& outMaximumTurnAngleAngle
)
```

Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
→	inAllowedTurnDirection	TurnAngleDirection::Type	All	Allows to detect only left-turns, only right-turns or both
→	inResultPrecision	TurnAnglePrecision::Type		Switches between pixel-precise or subpixel-precise detection of the found maximum
←	outMaximumTurnAngleIndex	Conditional<float>&		Index of found maximum
←	outMaximumTurnAnglePoint	Conditional<Point2D>&		Found point of a path with maximum turn angle
←	outMaximumTurnAngleAngle	Conditional<float>&		Maximal turn angle of the input path

PathTurnAngleProfile

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes the profile of turn angles at characteristic points of a path.

Syntax

```
void avl::PathTurnAngleProfile
(
 const avl::Path& inPath,
 avl::Profile& outTurnAngleProfile
)
```

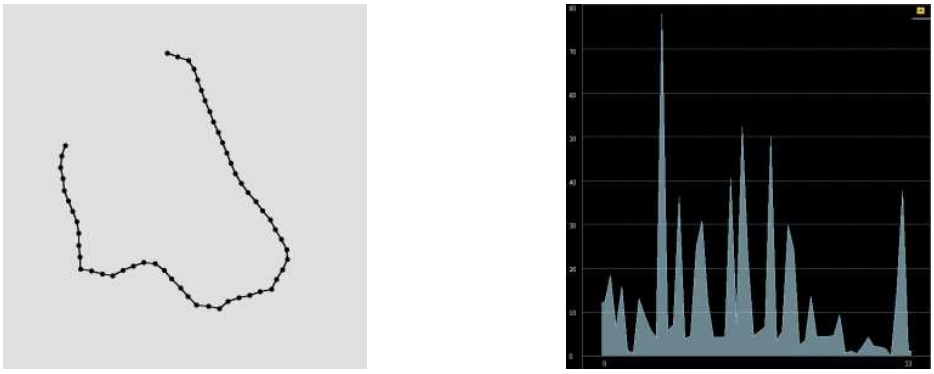
Parameters

	Name	Type	Default	Description
→	inPath	const Path&		Input path
←	outTurnAngleProfile	Profile&		

Description

The operation iterates over the characteristic points of the **inPath** and computes the absolute value of path turn angle at each one. The resulting **outTurnAngleProfile** profile consists of the angles computed at consecutive points of the path.

Examples



PathTurnAngleProfile *run on the sample path*

Errors

List of possible exceptions:

Error type	Description
DomainError	Input path contains a pair of equal consecutive points in PathTurnAngleProfile.

See Also

- PathAverageTurnAngle – Computes the average absolute turn angle of a path per unit of length.

114. Path Metrics

Table of content:

PathEditDistance

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes the edit distance between the input paths.

Syntax

```
void avl::PathEditDistance
(
 const avl::Path& inOldPath,
 const avl::Path& inNewPath,
 int& outAdditions,
 int& outRemovals,
 int& outEditions,
 atl::Optional<avl::CoordinateSystem2D&> outAlignment = atl::NIL,
 atl::Optional<atl::Array<avl::Point2D>&> outOldCommonPoints = atl::NIL,
 atl::Optional<atl::Array<avl::Point2D>&> outNewCommonPoints = atl::NIL,
 atl::Optional<atl::Array<avl::Point2D>&> outAddedPoints = atl::NIL,
 atl::Optional<atl::Array<avl::Point2D>&> outRemovedPoints = atl::NIL,
 atl::Optional<atl::Array<avl::Point2D>&> outOldEditedPoints = atl::NIL,
 atl::Optional<atl::Array<avl::Point2D>&> outNewEditedPoints = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inOldPath	const Path&		
→	inNewPath	const Path&		
←	outAdditions	int&		
←	outRemovals	int&		
←	outEditions	int&		
←	outAlignment	Optional<CoordinateSystem2D&>	NIL	
←	outOldCommonPoints	Optional<Array<Point2D>&>	NIL	
←	outNewCommonPoints	Optional<Array<Point2D>&>	NIL	
←	outAddedPoints	Optional<Array<Point2D>&>	NIL	
←	outRemovedPoints	Optional<Array<Point2D>&>	NIL	
←	outOldEditedPoints	Optional<Array<Point2D>&>	NIL	
←	outNewEditedPoints	Optional<Array<Point2D>&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignment**, **outOldCommonPoints**, **outNewCommonPoints**, **outAddedPoints**, **outRemovedPoints**, **outOldEditedPoints**, **outNewEditedPoints**.

Read more about Optional Outputs.

PathToLineDistance

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the smallest distance between a path and a line.

Syntax

```
void avl::PathToLineDistance
(
 const avl::Path& inPath,
 const avl::Line2D& inLine,
 float inResolution,
 float& outDistance,
 atl::Optional<avl::Segment2D&> outConnectingSegment = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPath	const Path&			Input path
→	inLine	const Line2D&			Input line
→	inResolution	float	0.0 - ∞	1.0f	Number of real-world units per one pixel
←	outDistance	float&			Minimal distance between input path and input line
←	outConnectingSegment	Optional<Segment2D&>		NIL	Segment connecting input path and input line having minimal length

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outConnectingSegment**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty path on input in PathToLineDistance.
DomainError	Indefinite line on input in PathToLineDistance.

PathToLineDistanceProfile

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the profile of distances between a line and the characteristic points of a path.

Syntax

```
void avl::PathToLineDistanceProfile
(
 const avl::Path& inPath,
 const avl::Line2D& inLine,
 float inResolution,
 avl::Profile& outDistanceProfile,
 atl::Optional<atl::Array<float>&> outDistances = atl::NIL,
 atl::Optional<atl::Array<avl::Segment2D>&> outConnectingSegments = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPath	const Path&			Input path
→	inLine	const Line2D&			Input line
→	inResolution	float	0.0 - ∞	1.0f	Number of real-world units per one pixel
←	outDistanceProfile	Profile&			Profile of distances between input line and consecutive points of input path
←	outDistances	Optional<Array<float>&>		NIL	Distances between input line and consecutive points of input path
←	outConnectingSegments	Optional<Array<Segment2D>&>		NIL	Segments connecting input line and consecutive points of input path having minimal length

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outDistances**, **outConnectingSegments**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite line on input in PathToLineDistanceProfile.

PathToPathDistance

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes the minimum distance between two paths.

Syntax

```
void avl::PathToPathDistance
(
 const avl::Path& inPath1,
 const avl::Path& inPath2,
 avl::PathDistanceMode::Type inPathDistanceMode,
 float inResolution,
 float& outDistance,
 atl::Optional<avl::Segment2D&> outConnectingSegment = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPath1	const Path&			First input path
→	inPath2	const Path&			Second input path
→	inPathDistanceMode	PathDistanceMode::Type			Distance measuring method
→	inResolution	float	0.0 - ∞	1.0f	
←	outDistance	float&			Minimal distance between input paths
←	outConnectingSegment	Optional<Segment2D&>		NIL	Segment connecting input paths having minimal length

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outConnectingSegment**.

Read more about Optional Outputs.

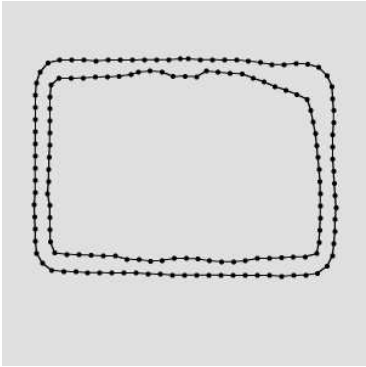
Description

The operation finds the minimal distance between a characteristic point of **inPath1** and path **inPath2**. The distance between a point and a path is computed as follows, depending on the value of **inPathDistanceMode**:

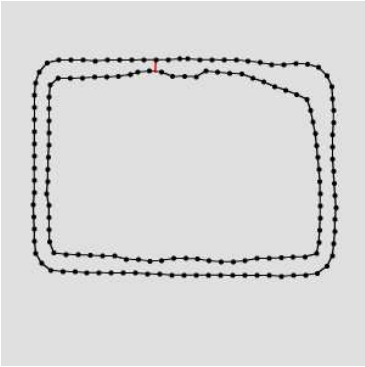
- The distance to nearest characteristic point of **inPath2**, if **inPathDistanceMode** is set to PointToPoint.
- The minimal distance to **inPath2** segments adjacent to the nearest characteristic point of **inPath2** (which is much more precise), if **inPathDistanceMode** is set to PointToSegment.

The operation computes the **outDistance** distance and, additionally, the **outConnectingSegment** line segment corresponding to the result.

Examples



*The **PathToPathDistance** run on the sample paths produces **outDistance** = 9.509.*



*The resulting **outConnectingSegment** drawn onto the input paths.*

Errors

List of possible exceptions:

Error type	Description
DomainError	One or both input paths are empty in PathToPathDistance.

See Also

- PathToPathDistanceProfile – Computes the profile of distances between two paths.

PathToPathDistanceProfile

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes the profile of distances between two paths.

Syntax

```
void avl::PathToPathDistanceProfile
(
 const avl::Path& inPath1,
 const avl::Path& inPath2,
 avl::PathDistanceMode::Type inPathDistanceMode,
 float inResolution,
 avl::Profile& outDistanceProfile,
 atl::Optional<atl::Array<float>&> outDistances = atl::NIL,
 atl::Optional<atl::Array<avl::Segment2D>&> outConnectingSegments = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inPath1	const Path&			First input path
➡	inPath2	const Path&			Second input path
➡	inPathDistanceMode	PathDistanceMode::Type			Distance measuring method
➡	inResolution	float	0.0 - ∞	1.0f	
⬅	outDistanceProfile	Profile&			Profile of distances between second path and consecutive points of first path
⬅	outDistances	Optional<Array<float>&>		NIL	Distances between second path and consecutive points of first path
⬅	outConnectingSegments	Optional<Array<Segment2D>&>		NIL	Segments connecting second path and consecutive points of first path having minimal length

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outDistances**, **outConnectingSegments**.

Read more about Optional Outputs.

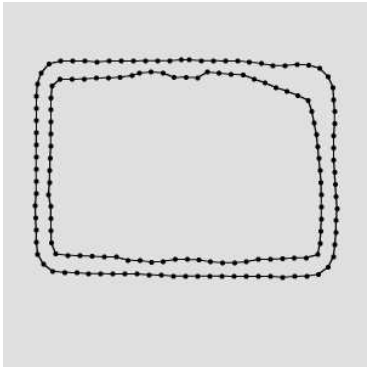
Description

The operation iterates over characteristic points of **inPath1** and at each point computes the distance from this point to **inPath2**. The distance is computed as follows, depending on the value of **inPathDistanceMode**:

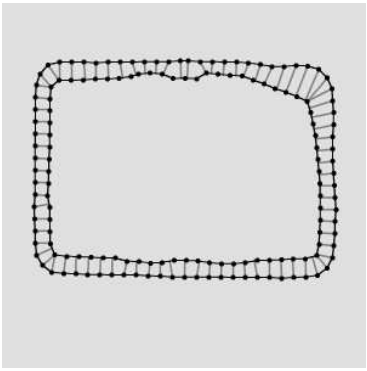
- The distance to nearest characteristic point of **inPath2**, if **inPathDistanceMode** is set to PointToPoint.
- The minimal distance to **inPath2** segments adjacent to the nearest characteristic point of **inPath2** (which is much more precise), if **inPathDistanceMode** is set to PointToSegment.

The operation computes **outDistanceProfile** profile of consecutive distances and, additionally, **outConnectingSegments** array of corresponding line segments.

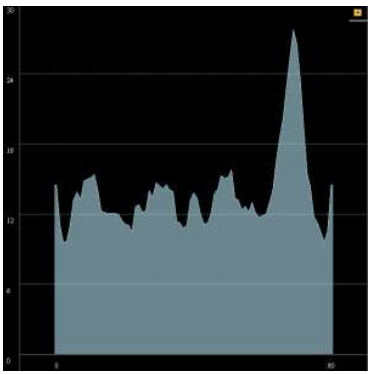
Examples



Sample paths



*The resulting **outConnectingSegments** drawn onto the input paths.*



*The resulting **outDistanceProfile**.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Second input path is empty and first input path is not empty in PathToPathDistanceProfile.

See Also

- PathToPathDistance – Computes the minimum distance between two paths.

PathToPathMaximumDistance

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes the distance between the farthest point of the input path from the other path.

Syntax

```
void avl::PathToPathMaximumDistance
(
 const avl::Path& inPath1,
 const avl::Path& inPath2,
 avl::PathDistanceMode::Type inPathDistanceMode,
 float inResolution,
 float& outDistance,
 atl::Optional<avl::Segment2D&> outConnectingSegment = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPath1	const Path&			First input path
→	inPath2	const Path&			Second input path
→	inPathDistanceMode	PathDistanceMode::Type			Distance measuring method
→	inResolution	float	0.0 - ∞	1.0f	
←	outDistance	float&			Distance between farthest point of the input path the other input path
←	outConnectingSegment	Optional<Segment2D&>		NIL	Segment connecting input paths having such distance

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outConnectingSegment**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	One or both input paths are empty in PathToPathMaximumDistance.

PathToPointDistance

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the smallest distance between a path and a point.

Syntax

```
void avl::PathToPointDistance
(
 const avl::Point2D& inPoint,
 const avl::Path& inPath,
 float inResolution,
 float& outDistance,
 atl::Optional<avl::Segment2D&> outConnectingSegment = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoint	const Point2D&			Input point
→	inPath	const Path&			Input path
→	inResolution	float	0.0 - ∞	1.0f	
←	outDistance	float&			Minimal distance between input path and input point
←	outConnectingSegment	Optional<Segment2D&>		NIL	Segment connecting input path and input point having minimal length

Optional Outputs

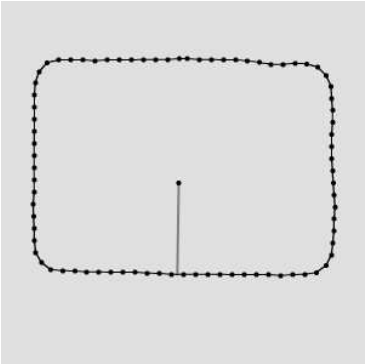
The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outConnectingSegment**.

Read more about Optional Outputs.

Description

The operation computes the minimal distance between the **inPoint** and **inPath** and, additionally, the **outConnectingSegment** segment corresponding to the result.

Examples



*The **PathToPointDistance** run on the sample data produces **outDistance** = 75.373.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty path on input in PathToPointDistance.

See Also

- PathToPointDistanceProfile – Computes the profile of distances between the specified point and the characteristic points of a path.

PathToPointDistanceProfile

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the profile of distances between the specified point and the characteristic points of a path.

Syntax

```
void avl::PathToPointDistanceProfile
(
 const avl::Point2D& inPoint,
 const avl::Path& inPath,
 float inResolution,
 avl::Profile& outDistanceProfile,
 atl::Optional<atl::Array<float>&> outDistances = atl::NIL,
 atl::Optional<atl::Array<avl::Segment2D>&> outConnectingSegments = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPoint	const Point2D&			Input point
→	inPath	const Path&			Input path
→	inResolution	float	0.0 - ∞	1.0f	
←	outDistanceProfile	Profile&			Profile of distances between input point and consecutive points of input path
←	outDistances	Optional<Array<float>&>		NIL	Distances between input point and consecutive points of input path
←	outConnectingSegments	Optional<Array<Segment2D>&>		NIL	Segments connecting input point and consecutive points of input path

Optional Outputs

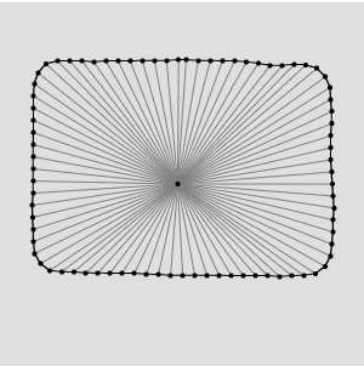
The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outDistances**, **outConnectingSegments**.

Read more about Optional Outputs.

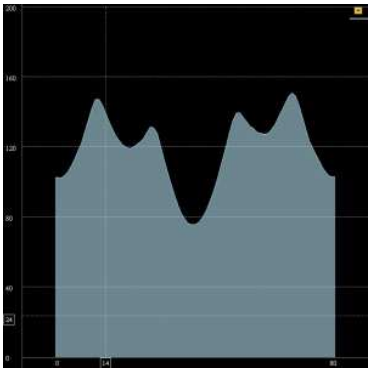
Description

The operation iterates over characteristic points of **inPath** and at each point computes the distance from this point to **inPoint**. The operation computes **outDistanceProfile** profile of consecutive distances and, additionally, the **outConnectingSegments** array of corresponding line segments.

Examples



*The resulting **outConnectingSegments** drawn onto the input data.*



*The resulting **outDistanceProfile**.*

See Also

- PathToPointDistance – Computes the smallest distance between a path and a point.

115. Point3DGrid Features

Table of content:

Point3DGridHole

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Finds a biggest hole in a given section of point cloud.

Syntax

```
void avl::Point3DGridHole
(
 const avl::Point3DGrid& inPointGrid,
 atl::Optional<const avl::Region&> inRoi,
 avl::MEstimator::Type inPlaneOutlierSuppression,
 float inClippingFactor,
 int inIterationCount,
 atl::Optional<const avl::Plane3D&> inInitialPlane,
 atl::Conditional<avl::Region>& outHoleRegion,
 atl::Conditional<avl::Point3D>& outHoleCenter,
 avl::Plane3D& outPlane,
 atl::Optional<atl::Array<avl::Point3D>&> outInliers = atl::NIL,
 atl::Optional<atl::Array<float>&> outDistances = atl::NIL,
 atl::Optional<float&> outSignedDistanceSum = atl::NIL,
 atl::Optional<float&> outDistanceSum = atl::NIL,
 atl::Optional<atl::Array<float>&> outSquaredDistances = atl::NIL,
 atl::Optional<float&> outSquaredDistanceSum = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPointGrid	const Point3DGrid&			
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inPlaneOutlierSuppression	MEstimator::Type			
→	inClippingFactor	float	0.675 - 6.0	2.5f	Multitude of standard deviation within which points are considered inliers
→	inIterationCount	int	0 - ∞	5	Number of iterations of outlier suppressing algorithm
→	inInitialPlane	Optional<const Plane3D&>		NIL	Initial approximation of a plane (if available)
←	outHoleRegion	Conditional<Region>&			Region of the found hole
←	outHoleCenter	Conditional<Point3D>&			Center of the found hole
←	outPlane	Plane3D&			
←	outInliers	Optional<Array<Point3D>&>		NIL	Points matching the computed plane
←	outDistances	Optional<Array<float>&>		NIL	Distances of the input points to a resulting plane
←	outSignedDistanceSum	Optional<float&>		NIL	Sum of signed distances of the input points to a resulting plane
←	outDistanceSum	Optional<float&>		NIL	Sum of distances of the input points to a resulting plane
←	outSquaredDistances	Optional<Array<float>&>		NIL	Squared distances of the input points to a resulting plane
←	outSquaredDistanceSum	Optional<float&>		NIL	Sum of squared distances of the input points to a resulting plane

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outInliers**, **outDistances**, **outSignedDistanceSum**, **outDistanceSum**, **outSquaredDistances**, **outSquaredDistanceSum**.

Read more about Optional Outputs.

Point3DGridValidPointsRegion

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Computes region of locations where points are valid in a points 3D grid and where they are invalid.

Syntax

```
void avl::Point3DGridValidPointsRegion
(
 const avl::Point3DGrid& inPoint3DGrid,
 atl::Optional<const avl::Region&> inRoi,
 avl::Region& outValidPointsRegion,
 avl::Region& outInvalidPointsRegion
)
```

Parameters

	Name	Type	Default	Description
→	inPoint3DGrid	const Point3DGrid&		
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
←	outValidPointsRegion	Region&		Region of locations where points are valid
←	outInvalidPointsRegion	Region&		Region of locations where points are invalid

116. Polygon Features

Table of content:

PolygonArea

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the area of polygon.

Syntax

```
void avl::PolygonArea
(
 const avl::Path& inPolygon,
 float& outArea
)
```

Parameters

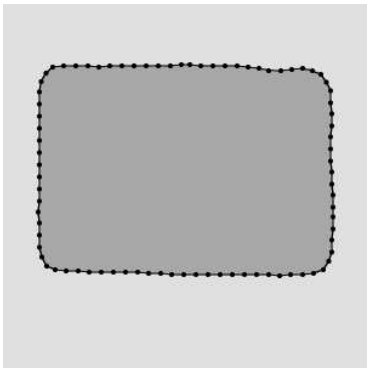
	Name	Type	Default	Description
→	inPolygon	const Path&		
←	outArea	float&		

Description

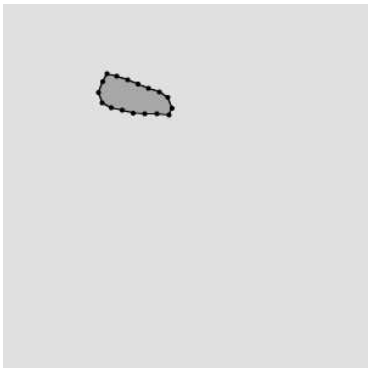
The operation computes the area of a polygon bounded by a closed path. The area is measured in square pixels. As paths in AVS are subpixel-precise, the result may be (and usually is) non-integer.

Note that if the input path is not a valid polygon (i.e. it has at least one self-intersection), the computation may lead to results that are not intuitive.

Examples



Area of the sample shape equals to 40652.37.



Area of the sample shape equals to 1287.675.

Errors

List of possible exceptions:

Error type	Description
DomainError	Open path on input in PolygonArea.

See Also

- RegionArea – Computes the number of pixels contained in a region.

PolygonCircularity

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the area of a polygon divided by the area of a circle having the same feature.

Syntax

```
void avl::PolygonCircularity
(
 const avl::Path& inPolygon,
 const avl::CircularityMeasure::Type inCircularityMeasure,
 float& outCircularity
)
```

Parameters

	Name	Type	Default	Description
➡	inPolygon	const Path&		
➡	inCircularityMeasure	const CircularityMeasure::Type	RadiusPreserving	
⬅	outCircularity	float&		

Description

Circularity is a measure of similarity of a polygon to the perfect circle. Circular polygons have circularity close to 1.0, while the more elongated the shape is (or contains more holes), the closer to 0.0 is its circularity.

Mathematically, the circularity is calculated as follows:

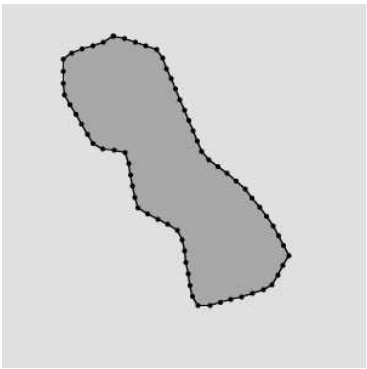
$$Circularity(s) = \frac{Area(s)}{Area(c)}$$

Where **c** denotes a circular polygon having the same feature as input polygon **s**. The feature being considered depends on the **inCircularityMeasure** chosen and it is:

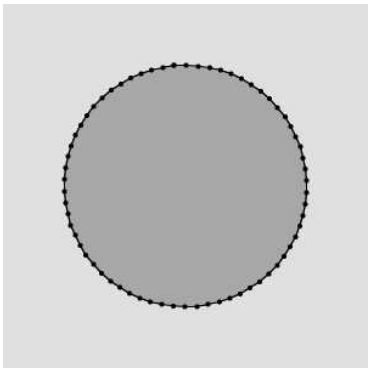
- the minimal bounding circle in case of **BoundingCirclePreserving**
- the perimeter in case of **PerimeterPreserving**
- the radius (maximal distance from mass center to any of the shape points) in case of **RadiusPreserving**

Note that if the input path is not a valid polygon (i.e. it has at least one self-intersection), the computation may lead to results that are not intuitive.

Examples



*Circularity with **RadiusPreserving** of the sample polygon equals to 0.340.*



*Circularity with **PerimeterPreserving** of the sample polygon equals to 0.998.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Degenerate polygon on input in PolygonCircularity.
DomainError	Not supported circularity measure in PolygonCircularity.
DomainError	Open path on input in PolygonCircularity.

See Also

- RegionCircularity – Computes the area of a region divided by the area of a circular region having the same feature.
- PolygonConvexity – Computes the area of a polygon divided by the area of its convex hull.
- PolygonElongation – Computes the elongation factor of a polygon (perfect circle has minimal elongation equal 1.0).

PolygonConvexity

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the area of a polygon divided by the area of its convex hull.

Syntax

```
void avl::PolygonConvexity
(
 const avl::Path& inPolygon,
 float& outConvexity
)
```

Parameters

	Name	Type	Default	Description
→	inPolygon	const Path&		
←	outConvexity	float&		

Description

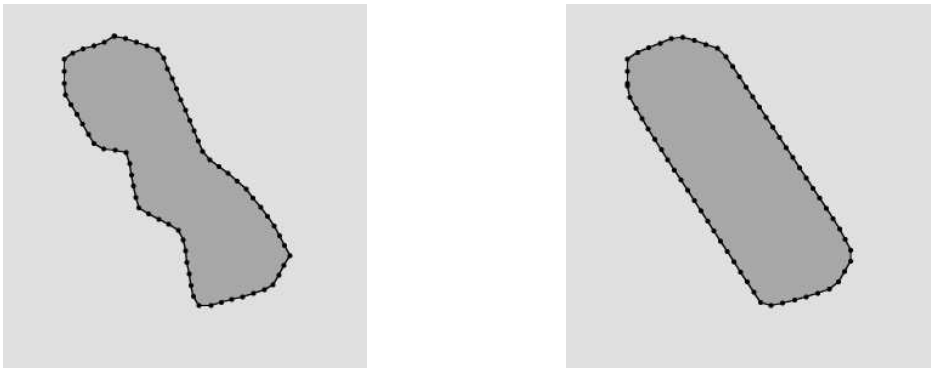
Convexity is a measure of how close a polygon is to being convex. Convex polygons have convexity equal to 1.0, while the more concave the polygon is, the closer to 0.0 is its convexity.

Mathematically, the convexity is calculated as follows:

$$Convexity(s) = \frac{Area(s)}{Area(ConvexHull(s))}$$

Note that if the input path is not a valid polygon (i.e. it has at least one self-intersection), the computation may lead to results that are not intuitive.

Examples



Area of the sample polygon (on the left) equals to 17529.730, while area of its convex hull (on the right) equals to 20176.250, so the convexity of the polygon equals to 17529.730/20176.250 = 0.867.

Errors

List of possible exceptions:

Error type	Description
DomainError	Degenerate polygon on input in PolygonConvexity.
DomainError	Open path on input in PolygonConvexity.

See Also

- RegionConvexity – Computes the area of a region divided by area of its convex hull.
- PathConvexHull – Computes the smallest convex shape that contains the given path.
- PolygonElongation – Computes the elongation factor of a polygon (perfect circle has minimal elongation equal 1.0).
- PolygonCircularity – Computes the area of a polygon divided by the area of a circle having the same feature.

PolygonEllipticAxes

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes axes of an ellipse having the same first and second order moments as the given polygon.

Syntax

```
void avl::PolygonEllipticAxes
(
 const avl::Path& inPolygon,
 avl::Segment2D& outMajorAxis,
 avl::Segment2D& outMinorAxis
)
```

Parameters

	Name	Type	Default	Description
→	inPolygon	const Path&		
←	outMajorAxis	Segment2D&		
←	outMinorAxis	Segment2D&		

Description

The orientations of the resulting axes are always between 0 and 180 degrees.

Note that if the input path is not a valid polygon (i.e. it has at least one self-intersection), the computation may lead to results that are not intuitive.

Errors

List of possible exceptions:

Error type	Description
DomainError	Degenerate polygon on input in PolygonEllipticAxes.

PolygonElongation

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the elongation factor of a polygon (perfect circle has minimal elongation equal 1.0).

Syntax

```
void avl::PolygonElongation
(
 const avl::Path& inPolygon,
 float& outElongation
)
```

Parameters

	Name	Type	Default	Description
→	inPolygon	const Path&		
←	outElongation	float&		

Description

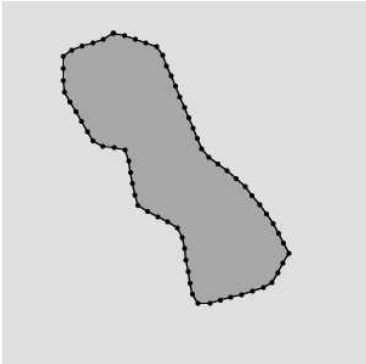
Elongation is a measure of how long the polygon is in relation to its width. The perfect circle has the minimal elongation equal to 1.0, while the upper bound doesn't exist. The operation internally approximates the polygon with an ellipse, and then computes the result as:

$$Elongation(s) = \frac{Length(L)}{Length(S)}$$

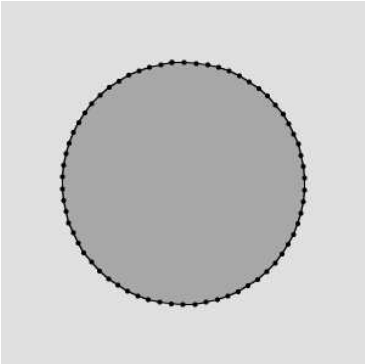
Where **L** denotes the longer axis of the approximating ellipse, and **S** denotes the shorter one.

Note that if the input path is not a valid polygon (i.e. it has at least one self-intersection), the computation may lead to results that are not intuitive.

Examples



Elongation of the sample polygon equals to 3.283.



Elongation of the sample polygon equals to 1.001.

Errors

List of possible exceptions:

Error type	Description
DomainError	Degenerate polygon on input in PolygonElongation.
DomainError	Open path on input in PolygonElongation.

See Also

- PolygonCircularity – Computes the area of a polygon divided by the area of a circle having the same feature.
- RegionElongation – Computes the elongation factor of a region (perfect circle has minimal elongation equal 1.0).

PolygonInscribedCircle

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the circle with the maximal possible area contained in the input polygon.

Syntax

```
void avl::PolygonInscribedCircle
(
 const avl::Path& inPolygon,
 avl::Circle2D& outInscribedCircle
)
```

Parameters

	Name	Type	Default	Description
→	inPolygon	const Path&		
←	outInscribedCircle	Circle2D&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty path on input in PolygonInscribedCircle.
DomainError	Open path on input in PolygonInscribedCircle.

PolygonMassCenter

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the mass center of polygon.

Syntax

```
void avl::PolygonMassCenter
(
 const avl::Path& inPolygon,
 avl::Point2D& outMassCenter
)
```

Parameters

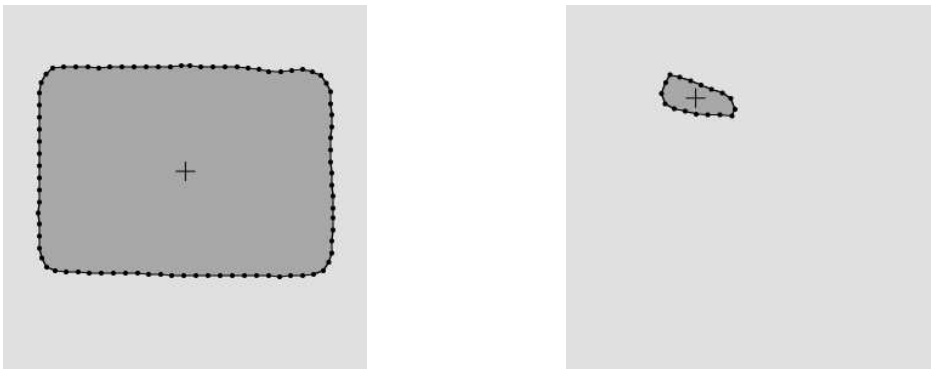
	Name	Type	Default	Description
→	inPolygon	const Path&		
←	outMassCenter	Point2D&		

Description

The operation computes the mass center of a polygon bounded by a closed path.

Note that if the input path is not a valid polygon (i.e. it has at least one self-intersection), the computation may lead to results that are not intuitive.

Examples



*The resulting **outMassCenter**s drawn onto the sample polygon.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Degenerate polygon on input in PolygonMassCenter.
DomainError	Open path on input in PolygonMassCenter.

See Also

- PathMassCenter – Computes the average of the path points (all, not only characteristic ones).
- RegionMassCenter – Computes a point with coordinates equal to the average coordinates of the region's pixels.

PolygonMoment

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the selected second-order moment of a polygon in regular and normalized (divided by polygon area) variant.

Syntax

```
void avl::PolygonMoment
(
 const avl::Path& inPolygon,
 avl::PolygonMomentType::Type inMomentType,
 bool inCentral,
 float& outMoment,
 float& outNormMoment
)
```

Parameters

	Name	Type	Default	Description
→	inPolygon	const Path&		
→	inMomentType	PolygonMomentType::Type		
→	inCentral	bool		
←	outMoment	float&		
←	outNormMoment	float&		

Description

The operation computes the mathematical features of a polygon called moments. Those are integrals computed as follows:

$$Moment_{2,0}(S) = \int_S p_x^2$$
$$Moment_{1,1}(S) = \int_S p_x p_y$$
$$Moment_{0,2}(S) = \int_S p_y^2$$

The integration is conducted over polygon surface, while p_x and p_y denote, accordingly, x and y coordinate of a point.

When **inCentral** parameter is set, the polygon is shifted before computations so that its mass center is at location (0,0).

Note that if the input path is not a valid polygon (i.e. it has at least one self-intersection), the computation may lead to results that are not intuitive.

Errors

List of possible exceptions:

Error type	Description
DomainError	Not supported moment type in PolygonMoment.
DomainError	Open path on input in PolygonMoment.

See Also

- RegionMoment – Computes selected second-order moment of a region in regular and normalized (divided by region area) variant.
- ImageMoment – Computes the selected moment of an image in regular and normalized (divided by sum of pixel values) variant.

PolygonOrientation

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the polygon orientation as angle with value in range 0.0 - 180.0.

Syntax

```
void avl::PolygonOrientation
(
 const avl::Path& inPolygon,
 float& outOrientationAngle
)
```

Parameters

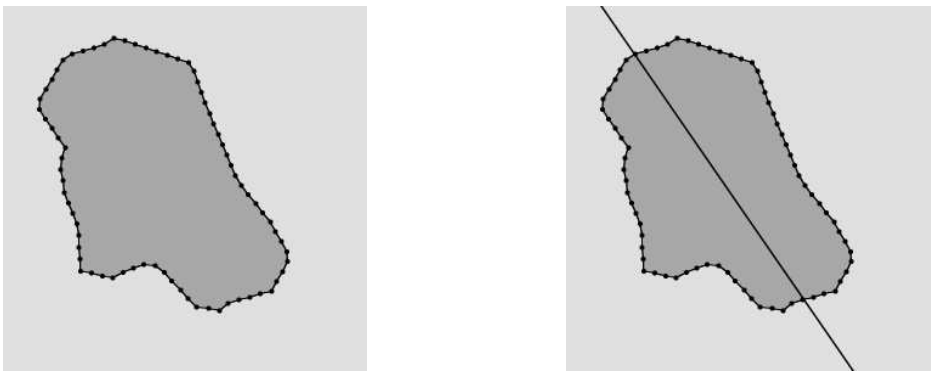
	Name	Type	Default	Description
→	inPolygon	const Path&		
←	outOrientationAngle	float&		

Description

Polygon orientation can be thought of as the direction in which the polygon is oriented. Mathematically it is the angle between X-axis and the line passing through the polygon mass center, that rotation around this line produces the smallest torque.

Note that if the input path is not a valid polygon (i.e. it has at least one self-intersection), the computation may lead to results that are not intuitive.

Examples



Orientation of the sample region equals to 55.386, which is visualized on the second image by drawing the line of this orientation passing through the mass center of the polygon.

Errors

List of possible exceptions:

Error type	Description
DomainError	Degenerate polygon on input in PolygonOrientation.
DomainError	Open path on input in PolygonOrientation.

See Also

- RegionOrientation – Computes the orientation of a region as an angle of value in a proper range.

PolygonRectangularity

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the area of a polygon divided by the area of its bounding rectangle.

Syntax

```
void avl::PolygonRectangularity
(
 const avl::Path& inPolygon,
 float& outRectangularity
)
```

Parameters

	Name	Type	Default	Description
→	inPolygon	const Path&		
←	outRectangularity	float&		

Description

Note that if the input path is not a valid polygon (i.e. it has at least one self-intersection), the computation may lead to results that are not intuitive.

Errors

List of possible exceptions:

Error type	Description
DomainError	Degenerate polygon on input in PolygonRectangularity.
DomainError	Open path on input in PolygonRectangularity.

PolygonWithNormalizedOrientation

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Returns the same polygon but with orientation corresponding to the axes.

Syntax

```
void avl::PolygonWithNormalizedOrientation
(
 const avl::Path& inPolygon,
 avl::Path& outPolygon
)
```

Parameters

	Name	Type	Default	Description
→	inPolygon	const Path&		
←	outPolygon	Path&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Open path on input in PolygonWithNormalizedOrientation.

117. Polygon Relations

Table of content:

PolygonIntersectionArea_Convex

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the intersection area of two convex non-intersecting polygons.

Syntax

```
void avl::PolygonIntersectionArea_Convex
(
 const avl::Path& inPolygon1,
 const avl::Path& inPolygon2,
 float& outIntersectionArea
)
```

Parameters

	Name	Type	Default	Description
➡	inPolygon1	const Path&		
➡	inPolygon2	const Path&		
⬅	outIntersectionArea	float&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Concave polygon on input in PolygonIntersectionArea_Convex.
DomainError	Empty path on input in PolygonIntersectionArea_Convex.
DomainError	Open path on input in PolygonIntersectionArea_Convex.

PolygonIntersection_Convex

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Finds the intersection of two convex non-intersecting polygons.

Syntax

```
void avl::PolygonIntersection_Convex
(
 const avl::Path& inPolygon1,
 const avl::Path& inPolygon2,
 avl::Path& outIntersection
)
```

Parameters

	Name	Type	Default	Description
➡	inPolygon1	const Path&		
➡	inPolygon2	const Path&		
⬅	outIntersection	Path&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Concave polygon on input in PolygonIntersection_Convex.
DomainError	Empty path on input in PolygonIntersection_Convex.
DomainError	Open path on input in PolygonIntersection_Convex.

TestPointArrayInPolygon

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Tests which points lie inside a polygon.

Syntax

```
void avl::TestPointArrayInPolygon
(
 const atl::Array<avl::Point2D>& inPoints,
 const avl::Path& inPolygon,
 atl::Array<bool>& outIsContainedArray,
 atl::Array<avl::Point2D>& outPoints,
 atl::Optional<bool&> outAreAllContained = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inPoints	const Array<Point2D>&		Points which will be tested
→	inPolygon	const Path&		Polygon against which the points will be tested
←	outIsContainedArray	Array<bool>&		
←	outPoints	Array<Point2D>&		Points that are contained
←	outAreAllContained	Optional<bool&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAreAllContained**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty polygon on input in TestPointArrayInPolygon.
DomainError	Open path on input in TestPointArrayInPolygon.

TestPointInPolygon

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Tests whether a point lies inside a polygon.

Syntax

```
void avl::TestPointInPolygon
(
 const avl::Point2D& inPoint,
 const avl::Path& inPolygon,
 bool& outIsContained
)
```

Parameters

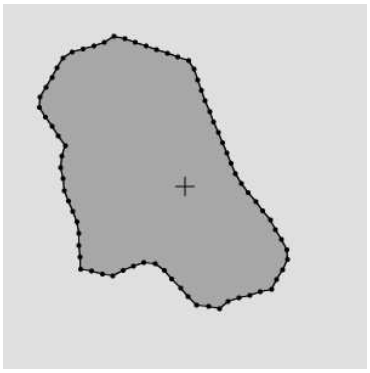
	Name	Type	Default	Description
→	inPoint	const Point2D&		Point the position of which will be tested
→	inPolygon	const Path&		Polygon against which the position will be tested
←	outIsContained	bool&		True if point lies inside the polygon, false otherwise

Description

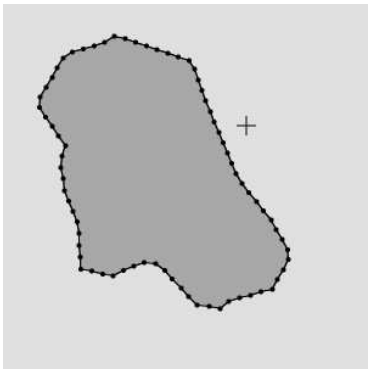
The operation tests if **inPoint** lies inside **inPolygon**.

Due to inaccuracy of floating point representation a point lying extremely close to a path may considered to be on either of the sides of the path or exactly on the path itself.

Examples



TestPointInPolygon *run on the sample data produces the* ***outIsContained*** *= true*



TestPointInPolygon *run on the sample data produces the* ***outIsContained*** *= false*

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty polygon on input in TestPointInPolygon.
DomainError	Open path on input in TestPointInPolygon.

See Also

- TestPolygonInPolygon – Tests whether a polygon lies inside another one.

TestPolygonConvex

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Tests whether a polygon is convex.

Syntax

```
void avl::TestPolygonConvex
(
 const avl::Path& inPolygon,
 bool& outIsConvex
)
```

Parameters

	Name	Type	Default	Description
→	inPolygon	const Path&		Input polygon
←	outIsConvex	bool&		True if the polygon is convex, false otherwise

TestPolygonInPolygon

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Tests whether a polygon lies inside another one.

Syntax

```
void avl::TestPolygonInPolygon
(
 const avl::Path& inSubPolygon,
 const avl::Path& inPolygon,
 bool& outIsContained
)
```

Parameters

	Name	Type	Default	Description
→	inSubPolygon	const Path&		Polygon the position of which will be tested
→	inPolygon	const Path&		Polygon against which the position will be tested
←	outIsContained	bool&		True if whole polygon lies inside the other one, false otherwise

Description

The operation tests if **inSubPolygon** lies (in its entirety) inside **inPolygon**.

Due to inaccuracy of floating point representation a point lying extremely close to a path may considered to be on either of the sides of the path or exactly on the path itself.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty path on input (inSubPolygon) in TestPolygonInPolygon.
DomainError	Open path on input in TestPolygonInPolygon.

See Also

- TestPointInPolygon – Tests whether a point lies inside a polygon.

118. Profile Metrics

Table of content:

ProfileAutocorrelation

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes the correlation between neighboring sub-profiles of different sizes and infers the most probable period length.

Syntax

```
void avl::ProfileAutocorrelation
(
 const avl::Profile& inProfile,
 int inStart,
 int inMinPeriod,
 int inMaxPeriod,
 int inMinVerifiedLength,
 bool inFlexibleVerification,
 int inMinRepeatCount,
 float inHarmonicHysteresis,
 avl::PeriodPrecisionMethod::Type inPrecisionMethod,
 atl::Array< float >& outAutocorrelationValues,
 float& outPeriod,
 float& outPeriodScore
)
```

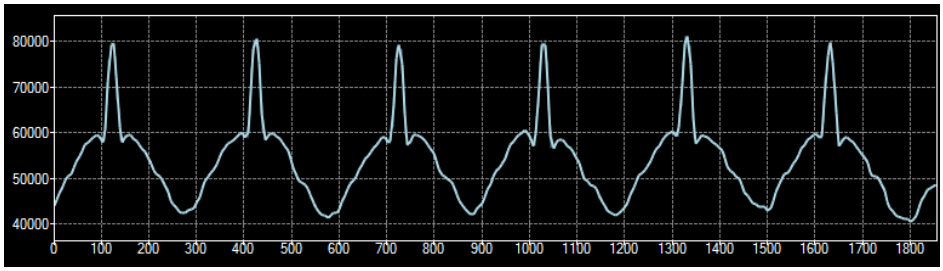
Parameters

	Name	Type	Range	Default	Description
→	inProfile	const Profile&			Input profile
→	inStart	int	0 - ∞		Beginning index of the sub-profile of interest
→	inMinPeriod	int	2 - ∞	2	Minimum period length
→	inMaxPeriod	int	2 - ∞	2	Maximum period length
→	inMinVerifiedLength	int	1 - ∞	1	Minimum number of profile points that verify single period (increases the actual RepeatCount for small periods)
→	inFlexibleVerification	bool			Compensates errors resulting from whole-pixel precision
→	inMinRepeatCount	int	1 - ∞	1	The number of repeats for sufficiently big periods
→	inHarmonicHysteresis	float	0.0 - 1.0	0.05f	Defines how much better must be the period T than T/2, T/3 etc. to be accepted
→	inPrecisionMethod	PeriodPrecisionMethod::Type			Defines if and how sub-point precision is achieved
←	outAutocorrelationValues	Array< float >&			Autocorrelation values for consecutive period values
←	outPeriod	float&			Estimated period length
←	outPeriodScore	float&			Correlation value for the estimated period length

Description

This filter is usually used to find a period within a profile. It tests all possible periods from the range **inMinPeriod** and **inMaxPeriod**. For each candidate period it creates several sub-profiles of that size and computes the normalized correlation between the first one and the others. As the final score, the minimum of the correlations is taken. The number of sub-profiles that is used is determined with the **inMinRepeatCount** and **inMinVerifiedLength** inputs. The latter is used to increase the number of sub-profiles that are used for small period candidates, which often cause false high-scoring periods when used without this parameter. Furthermore, as for any good period of the length T, the periods of the length T * 2, T * 3 etc. also score high, the **inHarmonicHysteresis** input can be used to define the minimum difference between the score of the tested period (T) and its integer partitions (T/2, T/3, ...).

Examples



On this profile the ProfileAutocorrelation filter finds a period of length 302 and score 0.998.

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

This operation is optimized for AVX2 technology.

Errors

List of possible exceptions:

Error type	Description
DomainError	inMinPeriod is larger than inMaxPeriod in ProfileAutocorrelation.
DomainError	inStart + inMinVerifiedLength is larger than the profile size in ProfileAutocorrelation.
DomainError	The range of applicable profile periods is empty in ProfileAutocorrelation.

ProfileCorrelation

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes the correlation between two sub-profiles.

Syntax

```
void avl::ProfileCorrelation
(
 const avl::Profile& inProfile1,
 int inStart1,
 const avl::Profile& inProfile2,
 int inStart2,
 atl::Optional<int> inLength,
 float& outCorrelation,
 atl::Optional<float&> outCovariance = atl::NIL,
 avl::Profile& diagProfile1,
 avl::Profile& diagProfile2
)
```

Parameters

	Name	Type	Range	Default	Description
→	inProfile1	const Profile&			First input profile
→	inStart1	int	0 - ∞		Start of the first sub-profile of interest
→	inProfile2	const Profile&			Second input profile
→	inStart2	int	0 - ∞		Start of the second sub-profile of interest
→	inLength	Optional<int>	0 - ∞	NIL	Length of the sub-profiles of interest
←	outCorrelation	float&			Pearson correlation coefficient
←	outCovariance	Optional<float&>		NIL	
🔍	diagProfile1	Profile&			First sub-profile of interest
🔍	diagProfile2	Profile&			Second sub-profile of interest

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outCovariance**.

Read more about Optional Outputs.

Hardware Acceleration

This operation is optimized for AVX2 technology.

This operation is optimized for NEON technology.

Errors

List of possible exceptions:

Error type	Description
DomainError	At least two element sub-profiles are required in ProfileCorrelation.
DomainError	Empty profiles on input in ProfileCorrelation.
DomainError	First sub-profile is out of range in ProfileCorrelation.
DomainError	Second sub-profile is out of range in ProfileCorrelation.

ProfileDistance

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes the [mean] square error between two profiles.

Syntax

```
void avl::ProfileDistance
(
 const avl::Profile& inProfile1,
 const avl::Profile& inProfile2,
 atl::Optional<const avl::Range&> inRange,
 avl::DistanceMeasure::Type inDistanceMeasure,
 float& outDistance
)
```

Parameters

	Name	Type	Default	Description
→	inProfile1	const Profile&		First input profile
→	inProfile2	const Profile&		Second input profile
→	inRange	Optional<const Range&>	NIL	
→	inDistanceMeasure	DistanceMeasure::Type		Measure of distance
←	outDistance	float&		Output distance value

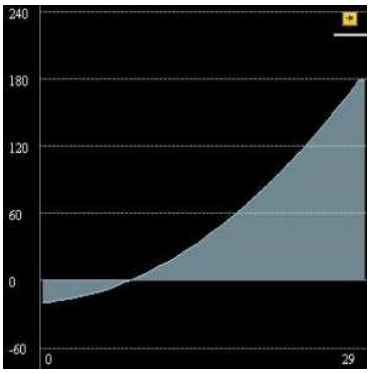
Description

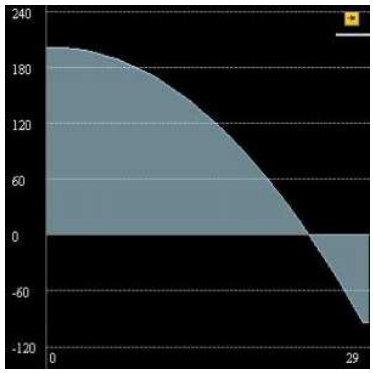
The operation computes the approximate difference between two profiles using the selected distance measure.

- If the **inDistanceMeasure** is set to **MeanError** then the resulting **outDistance** is the average difference between corresponding values of the profiles.
- If the **inDistanceMeasure** is set to **MeanSquaredError** then the resulting **outDistance** is the average squared difference between corresponding values of the profiles.

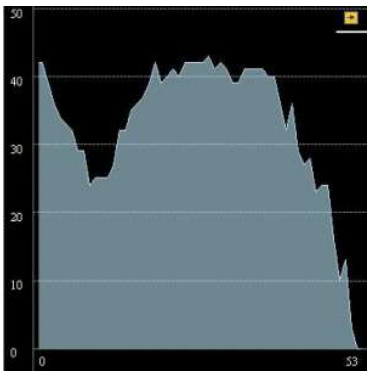
The operation requires that the profiles being compared have equal sizes, otherwise an error with appropriate description occurs.

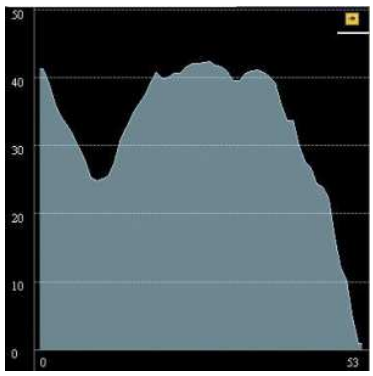
Examples





Mean Squared Error between the sample profiles equals 25245.070.





Mean Error between the sample profiles equals 0.803.

Errors

List of possible exceptions:

Error type	Description
DomainError	DistanceMeasure type not supported in ProfileDistance.
DomainError	Empty profile range in ProfileDistance.
DomainError	Empty profiles on input in ProfileDistance.
DomainError	Input profiles have different X coordinates in ProfileDistance.
DomainError	Range exceeds the input profile in ProfileDistance.
DomainError	Sizes of input profiles differ in ProfileDistance.

119. Profile Features

Table of content:

ProfileAverage

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes the average value of a profile.

Syntax

```
void avl::ProfileAverage
(
 const avl::Profile& inProfile,
 atl::Optional<const avl::Range&> inRange,
 float& outAverage
)
```

Parameters

	Name	Type	Default	Description
→	inProfile	const Profile&		Input profile
→	inRange	Optional<const Range&>	NIL	
←	outAverage	float&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty profile in ProfileAverage.
DomainError	Empty profile range in ProfileAverage.
DomainError	Range exceeds the input profile in ProfileAverage.

ProfileEdges

Header: AVL.h
Namespace: avl
Module: FoundationPro

Finds the locations at which the profile values raise or fall quickly.

Applications: Can be used for 1D edge detection when the brightness profile is extracted from an image in a non-standard way.

Syntax

```
void avl::ProfileEdges
(
 const avl::Profile& inProfile,
 atl::Optional<const avl::Range&> inRange,
 bool inCyclic,
 const avl::EdgeScanParams& inEdgeScanParams,
 float inMinDistance,
 atl::Optional<float> inMaxDistance,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Array<avl::ProfileEdge>& outEdges,
 atl::Optional<atl::Array<float>&> outDistances = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inProfile	const Profile&			Input profile
→	inRange	Optional<const Range&>		NIL	
→	inCyclic	bool			
→	inEdgeScanParams	const EdgeScanParams&		EdgeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f MinMagnitude: 5.0f EdgeTransition: BrightToDark)	Parameters controlling the edge extraction process
→	inMinDistance	float	0.0 - ∞	0.0f	Minimal distance between consecutive edges
→	inMaxDistance	Optional<float>	0.0 - ∞	NIL	Maximal distance between consecutive edges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
←	outEdges	Array<ProfileEdge>&			Found edges
←	outDistances	Optional<Array<float>&>		NIL	Output distances between consecutive edges
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the edge (derivative) operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outDistances**, **outResponseProfile**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Range exceeds the input profile in ProfileEdges.

ProfileLocalExtrema

Header: AVL.h
Namespace: avl
Module: FoundationPro

Finds the locations at which the values of the input profile are locally highest or lowest.

Syntax

```
void avl::ProfileLocalExtrema
(
 const avl::Profile& inProfile,
 atl::Optional<const avl::Range&> inRange,
 const bool inCyclic,
 const avl::ExtremumType::Type inExtremumType,
 avl::ProfileInterpolationMethod::Type inInterpolationMethod,
 const bool inConsiderPlateaus,
 atl::Optional<float> inMinValue,
 atl::Optional<float> inMaxValue,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Array<avl::Extremum1D>& outLocalExtrema
)
```

Parameters

	Name	Type	Default	Description
→	inProfile	const Profile&		Input profile
→	inRange	Optional<const Range&>	NIL	
→	inCyclic	const bool		Indicates whether the last element should be considered a neighbour of the first element
→	inExtremumType	const ExtremumType::Type		Type of extremum to find
→	inInterpolationMethod	ProfileInterpolationMethod::Type	Quadratic4	When interpolation is set to Quadratic each non-plateau extremum is located using a parabola fit
→	inConsiderPlateaus	const bool	True	Indicates whether the result should include centers of plateau extrema
→	inMinValue	Optional<float>	NIL	Minimum value of an extremum
→	inMaxValue	Optional<float>	NIL	Maximum value of an extremum
→	inLocalBlindness	Optional<const LocalBlindness&>	NIL	Defines conditions in which weaker extrema can be detected in the vicinity of stronger ones
←	outLocalExtrema	Array<Extremum1D>&		Extrema of the profile values

Description

The operation returns locations where the profile values are locally extremal, i.e. maximal or minimal, depending on the **inExtremumType** parameter. If the **inConsiderPlateaus** parameter is set to true, the same-value-ranges of the input profile will also be detected. The **inMinValue** and **inMaxValue** parameters control how strong an extremum has to be to be a part of the result. The **inLocalBlindness** parameter allows to have an even better control over the results of the filter. Its fields define a set of conditions in which weaker extrema can be detected in the vicinity of stronger extrema. When e.g. a maximum of value 50 is present at location 35 and another maximum of value 40 is present at location 28, the local blindness with the threshold bigger than 0.8 and the radius not smaller than 7 will suppress the weaker maximum. For minimum, the threshold works as a reciprocal, i.e. under a local blindness with the threshold equal to 0.8 the minimum with value 20 could suppress another minimum with value 26, but not a minimum with value 24. It should be noted that the **inLocalBlindness** parameter has no effect at all when **inExtremumType** is set to **Any**. When some profile values are negative, setting **inLocalBlindness** leads to undefined results.

Errors

List of possible exceptions:

Error type	Description
DomainError	Range exceeds the input profile in ProfileLocalExtrema.

ProfileMaximum

Header: AVL.h
Namespace: avl
Module: FoundationPro

Finds the highest value of the input profile, its precise location and the corresponding index.

Syntax

```
void avl::ProfileMaximum
(
 const avl::Profile& inProfile,
 atl::Optional<const avl::Range&> inRange,
 avl::ProfileInterpolationMethod::Type inInterpolationMethod,
 float& outMaximumPoint,
 atl::Optional<int&> outMaximumIndex = atl::NIL,
 atl::Optional<float&> outMaximumValue = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inProfile	const Profile&		Input profile
→	inRange	Optional<const Range&>	NIL	
→	inInterpolationMethod	ProfileInterpolationMethod::Type	Quadratic4	Profile points' interpolation method
←	outMaximumPoint	float&		Position of highest value with respect to profile's offset and scale
←	outMaximumIndex	Optional<int&>	NIL	Index of highest value
←	outMaximumValue	Optional<float&>	NIL	Highest value

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outMaximumIndex**, **outMaximumValue**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty profile in ProfileMaximum.
DomainError	Empty profile range in ProfileMaximum.
DomainError	Range exceeds the input profile in ProfileMaximum.
DomainError	Unsupported interpolation method in ProfileMaximum.

ProfileMinimum

Header: AVL.h
Namespace: avl
Module: FoundationPro

Finds the lowest value of the input profile, its precise location and the corresponding index.

Syntax

```
void avl::ProfileMinimum
(
 const avl::Profile& inProfile,
 atl::Optional<const avl::Range&> inRange,
 avl::ProfileInterpolationMethod::Type inInterpolationMethod,
 float& outMinimumPoint,
 atl::Optional<int&> outMinimumIndex = atl::NIL,
 atl::Optional<float&> outMinimumValue = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inProfile	const Profile&		Input profile
→	inRange	Optional<const Range&>	NIL	
→	inInterpolationMethod	ProfileInterpolationMethod::Type	Quadratic4	Profile points' interpolation method
←	outMinimumPoint	float&		Position of lowest value with respect to profile's offset and scale
←	outMinimumIndex	Optional<int&>	NIL	Index of lowest value
←	outMinimumValue	Optional<float&>	NIL	Lowest value

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outMinimumIndex**, **outMinimumValue**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty profile in ProfileMinimum.
DomainError	Empty profile range in ProfileMinimum.
DomainError	Range exceeds the input profile in ProfileMinimum.
DomainError	Unsupported interpolation method in ProfileMinimum.

ProfileRidges

Header: AVL.h
Namespace: avl
Module: FoundationPro

Finds the high or low peaks in the input profile.

Applications: Can be used for 1D ridge detection when the brightness profile is extracted from an image in a non-standard way.

Syntax

```
void avl::ProfileRidges
(
 const avl::Profile& inProfile,
 atl::Optional<const avl::Range&> inRange,
 bool inCyclic,
 const avl::RidgeScanParams& inRidgeScanParams,
 float inMinDistance,
 atl::Optional<float> inMaxDistance,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Array<avl::ProfileRidge>& outRidges,
 atl::Optional<atl::Array<float>&> outDistances = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inProfile	const Profile&			Input profile
→	inRange	Optional<const Range&>		NIL	
→	inCyclic	bool			
→	inRidgeScanParams	const RidgeScanParams&		RidgeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f RidgeWidth: 5 RidgeMargin: 2 MinMagnitude: 5.0f RidgePolarity: Bright)	Parameters controlling the ridge extraction process
→	inMinDistance	float	0.0 - ∞	0.0f	Minimal distance between consecutive ridges
→	inMaxDistance	Optional<float>	0.0 - ∞	NIL	Maximal distance between consecutive ridges
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker ridges can be detected in the vicinity of stronger ridges
←	outRidges	Array<ProfileRidge>&			Found ridges
←	outDistances	Optional<Array<float>&>		NIL	Output distances between consecutive ridges
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the ridge operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outDistances**, **outResponseProfile**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Range exceeds the input profile in ProfileRidges.

ProfileSections

Header: AVL.h
Namespace: avl
Module: FoundationPro

Finds subprofiles whose values fall into the specified range.

Applications: It may also be considered profile thresholding.

Syntax

```
void avl::ProfileSections
(
 const avl::Profile& inProfile,
 atl::Optional<const avl::Range&> inRange,
 atl::Optional<float> inMinValue,
 atl::Optional<float> inMaxValue,
 const float inMinSectionWidth,
 atl::Optional<float> inMaxSectionWidth,
 float inMinGapWidth,
 atl::Optional<float> inMaxGapWidth,
 atl::Optional<float> inMaxInnerGapWidth,
 atl::Array<avl::ProfileSection>& outSections,
 atl::Conditional<avl::ProfileSection>& outBoundingSection
)
```

Parameters

	Name	Type	Range	Default	Description
→	inProfile	const Profile&			Input profile
→	inRange	Optional<const Range&>		NIL	
→	inMinValue	Optional<float>		5.0f	Lower bound for profile values
→	inMaxValue	Optional<float>		NIL	Upper bound for profile values
→	inMinSectionWidth	const float	0.0 - ∞	0.0f	Minimal width of the found section
→	inMaxSectionWidth	Optional<float>	0.0 - ∞	NIL	Maximal width of the found section
→	inMinGapWidth	float	0.0 - ∞	0.0f	Minimal distance between consecutive sections
→	inMaxGapWidth	Optional<float>	0.0 - ∞	NIL	Maximal distance between consecutive sections
→	inMaxInnerGapWidth	Optional<float>	0.0 - ∞	0.0f	Maximal possible gap width between two sections to join them into one
←	outSections	Array<ProfileSection>&			Output profile sections
←	outBoundingSection	Conditional<ProfileSection>&			The smallest section that contains all outSections

Errors

List of possible exceptions:

Error type	Description
DomainError	Range exceeds the input profile in ProfileSections.

ProfileSize

Header: AVL.h
Namespace: avl
Module: FoundationPro

Returns the number of elements in a profile.

Syntax

```
void avl::ProfileSize
(
 const avl::Profile& inProfile,
 int& outSize
)
```

Parameters

	Name	Type	Default	Description
→	inProfile	const Profile&		Input profile
←	outSize	int&		

ProfileStripes

Header: AVL.h
Namespace: avl
Module: FoundationPro

Finds pairs of opposite (raising and falling) edges in the input profile.

Applications: Can be used for 1D stripe detection when the brightness profile is extracted from an image in a non-standard way.

Syntax

```
void avl::ProfileStripes
(
 const avl::Profile& inProfile,
 atl::Optional<const avl::Range&> inRange,
 bool inCyclic,
 const avl::StripeScanParams& inStripeScanParams,
 float inMinGapWidth,
 atl::Optional<float> inMaxGapWidth,
 atl::Optional<const avl::LocalBlindness&> inLocalBlindness,
 atl::Array<avl::ProfileStripe>& outStripes,
 atl::Optional<atl::Array<float>&> outGapWidths = atl::NIL,
 atl::Optional<avl::Profile&> outResponseProfile = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inProfile	const Profile&			Input profile
→	inRange	Optional<const Range&>		NIL	
→	inCyclic	bool			
→	inStripeScanParams	const StripeScanParams&		StripeScanParams (ProfileInterpolation: Quadratic4 SmoothingStdDev: 0.6f MinMagnitude: 5.0f MaxInnerEdgeMagnitude: Nil StripePolarity: Bright MinStripeWidth: 0.0f MaxStripeWidth: Nil)	Parameters controlling the stripe extraction process
→	inMinGapWidth	float	0.0 - ∞	0.0f	Minimal distance between consecutive stripes
→	inMaxGapWidth	Optional<float>	0.0 - ∞	NIL	Maximal distance between consecutive stripes
→	inLocalBlindness	Optional<const LocalBlindness&>		NIL	Defines conditions in which weaker edges can be detected in the vicinity of stronger edges
←	outStripes	Array<ProfileStripe>&			Found stripes
←	outGapWidths	Optional<Array<float>&>		NIL	Distances between consecutive stripes
←	outResponseProfile	Optional<Profile&>		NIL	Profile of the edge (derivative) operator response

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outGapWidths**, **outResponseProfile**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Range exceeds the input profile in ProfileStripes.

ProfileSum

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes the sum of profile values.

Syntax

```
void avl::ProfileSum
(
 const avl::Profile& inProfile,
 atl::Optional<const avl::Range&> inRange,
 float& outSum
)
```

Parameters

	Name	Type	Default	Description
→	inProfile	const Profile&		Input profile
→	inRange	Optional<const Range&>	NIL	
←	outSum	float&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Range exceeds the input profile in ProfileSum.

ProfileZeroCrossings

Header: AVL.h
Namespace: avl
Module: FoundationPro

Finds the points of a profile at which the profile intersects the x-axis.

Syntax

```
void avl::ProfileZeroCrossings
(
 const avl::Profile& inProfile,
 atl::Optional<const avl::Range&> inRange,
 atl::Array<float>& outZeroCrossings
)
```

Parameters

	Name	Type	Default	Description
→	inProfile	const Profile&		Input profile
→	inRange	Optional<const Range&>	NIL	
←	outZeroCrossings	Array<float>&		X coordinates of the locations where the profile crosses the y=0 axis

Description

The operation assumes linear interpolation between the profile values and finds precise locations at which the profile crosses the horizontal $y=0$ axis. For the constant profile sections of value 0 only the integer coordinates of the sampling points in the section are returned.

Examples

→	←
inProfile = {-1.0,3.0,0.0,0.0,0.0}	outZeroCrossings = {0.25, 2.0, 3.0, 4.0}

Errors

List of possible exceptions:

Error type	Description
DomainError	Range exceeds the input profile in ProfileZeroCrossings.

120. Region Features

Table of content:

RegionArea

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the number of pixels contained in a region.

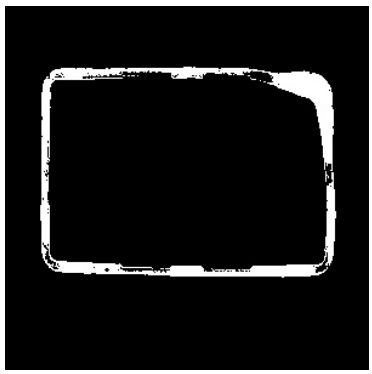
Syntax

```
void avl::RegionArea
(
 const avl::Region& inRegion,
 int& outArea
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outArea	int&		

Examples



Area of the sample region equals to 5567.

Errors

List of possible exceptions:

Error type	Description
DomainError	The area of the region exceeds the capacity of the integer type in RegionArea.

See Also

- PolygonArea – Computes the area of polygon.

RegionBoundingBox

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the smallest box containing a region.

Syntax

```
void avl::RegionBoundingBox
(
 const avl::Region& inRegion,
 avl::Box& outBoundingBox
)
```

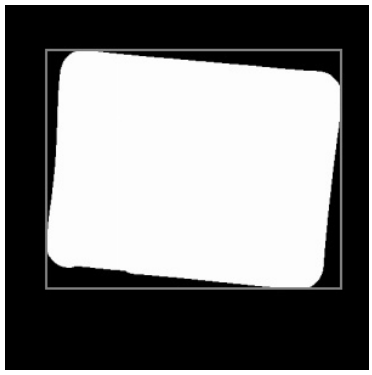
Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outBoundingBox	Box&		

Hints

- If the input region is not guaranteed to be non-empty, precede this filter with SkipEmptyRegion.

Examples



*The resulting **outBoundingBox** box drawn onto the input region.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty region on input in RegionBoundingBox.

See Also

- RegionBoundingCircle – Computes the smallest circle enclosing a region.
- PathBoundingBox – Computes the smallest box containing a path.

RegionBoundingBox_OrNil

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the smallest box containing a region; returns NIL if the region is empty.

Syntax

```
void avl::RegionBoundingBox_OrNil
(
 const avl::Region& inRegion,
 atl::Conditional<avl::Box>& outBoundingBox
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outBoundingBox	Conditional<Box>&		

RegionBoundingCircle

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the smallest circle enclosing a region.

Syntax

```
void avl::RegionBoundingCircle
(
 const avl::Region& inRegion,
 avl::Circle2D& outBoundingCircle
)
```

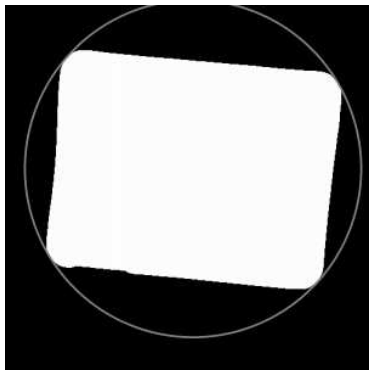
Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outBoundingCircle	Circle2D&		

Hints

- If the input region is not guaranteed to be non-empty, precede this filter with SkipEmptyRegion.

Examples



*The resulting **outBoundingCircle** circle drawn onto the input region.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty region on input in RegionBoundingCircle.

See Also

- RegionBoundingBox – Computes the smallest box containing a region.
- PathBoundingCircle – Computes the smallest circle enclosing a path.

RegionBoundingCircle_OrNil

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the smallest circle enclosing a region; returns NIL if the region is empty.

Syntax

```
void avl::RegionBoundingCircle_OrNil
(
 const avl::Region& inRegion,
 atl::Conditional<avl::Circle2D>& outBoundingCircle
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outBoundingCircle	Conditional<Circle2D>&		

RegionBoundingEllipse

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the smallest ellipse enclosing a region.

Syntax

```
void avl::RegionBoundingEllipse
(
 const avl::Region& inRegion,
 avl::Ellipse2D& outBoundingEllipse
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outBoundingEllipse	Ellipse2D&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty region on input in RegionBoundingEllipse.

RegionBoundingEllipse_OrNil

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the smallest ellipse enclosing a region; returns NIL if the region is empty.

Syntax

```
void avl::RegionBoundingEllipse_OrNil
(
 const avl::Region& inRegion,
 atl::Conditional<avl::Ellipse2D>& outBoundingEllipse
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outBoundingEllipse	Conditional<Ellipse2D>&		

RegionBoundingParallelogram

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes the smallest parallelogram containing a region.

Syntax

```
void avl::RegionBoundingParallelogram
(
 const avl::Region& inRegion,
 avl::BoundingRectangleFeature::Type inBoundingParallelogramFeature,
 avl::Path& outBoundingParallelogram,
 atl::Optional<avl::Point2D&> outCenter = atl::NIL,
 atl::Optional<float&> outLongSide = atl::NIL,
 atl::Optional<float&> outShortSide = atl::NIL,
 atl::Optional<float&> outAngle = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inBoundingParallelogramFeature	BoundingRectangleFeature::Type	MinimalArea	Determines what kind of bounding parallelogram will be computed
←	outBoundingParallelogram	Path&		Smallest bounding parallelogram of the input points
←	outCenter	Optional<Point2D&>	NIL	Center of the bounding parallelogram
←	outLongSide	Optional<float&>	NIL	Length of the bounding parallelogram long side
←	outShortSide	Optional<float&>	NIL	Length of the bounding parallelogram short side
←	outAngle	Optional<float&>	NIL	Angle of the bounding parallelogram

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outCenter**, **outLongSide**, **outShortSide**, **outAngle**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty region on input in RegionBoundingParallelogram.

RegionBoundingRectangle

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the smallest rectangle containing a region.

Syntax

```
void avl::RegionBoundingRectangle
(
 const avl::Region& inRegion,
 avl::BoundingRectangleFeature::Type inBoundingRectangleFeature,
 float inReferenceAngle,
 avl::RectangleOrientation::Type inRectangleOrientation,
 avl::Rectangle2D& outBoundingRectangle,
 atl::Optional<avl::Point2D&> outCenter = atl::NIL,
 atl::Optional<float&> outLongSide = atl::NIL,
 atl::Optional<float&> outShortSide = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inBoundingRectangleFeature	BoundingRectangleFeature::Type	MinimalArea	Determines what kind of bounding rectangle will be computed
→	inReferenceAngle	float	0.0f	The middle angle of the valid range of the output rectangle's angle
→	inRectangleOrientation	RectangleOrientation::Type	Horizontal	Orientation of the output rectangle
←	outBoundingRectangle	Rectangle2D&		The smallest bounding rectangle of the input region
←	outCenter	Optional<Point2D&>	NIL	Center of the bounding rectangle
←	outLongSide	Optional<float&>	NIL	Length of the bounding rectangle long side
←	outShortSide	Optional<float&>	NIL	Length of the bounding rectangle short side

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outCenter**, **outLongSide**, **outShortSide**.

Read more about Optional Outputs.

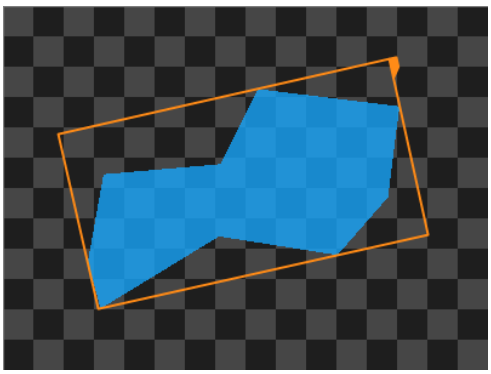
Description

The filter computes a rectangle with the smallest possible selected feature that contains all pixels belonging to the input region. The angle of the resulting rectangle is then normalized as in the NormalizeRectangleOrientation filter.

Hints

- If the input region is not guaranteed to be non-empty, precede this filter with SkipEmptyRegion.

Examples



RegionBoundingRectangle *performed on a sample region with* ***inRectangleOrientation*** *set on* ***Vertical****.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty region on input in RegionBoundingRectangle.

See Also

- NormalizeRectangleOrientation – Changes orientation of the given rectangle according to parameters.

RegionBoundingRectangle_FixedAngle

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the smallest rectangle with the given orientation angle containing a region.

Syntax

```
void avl::RegionBoundingRectangle_FixedAngle
(
 const avl::Region& inRegion,
 float inAngle,
 avl::Rectangle2D& outBoundingRectangle,
 atl::Optional<avl::Point2D&> outCenter = atl::NIL,
 atl::Optional<float&> outLongSide = atl::NIL,
 atl::Optional<float&> outShortSide = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inAngle	float		Expected angle of the resulting rectangle
←	outBoundingRectangle	Rectangle2D&		Smallest bounding rectangle of the input region
←	outCenter	Optional<Point2D&>	NIL	Center of the bounding rectangle
←	outLongSide	Optional<float&>	NIL	Length of the bounding rectangle long side
←	outShortSide	Optional<float&>	NIL	Length of the bounding rectangle short side

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outCenter**, **outLongSide**, **outShortSide**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty region on input in RegionBoundingRectangle_FixedAngle.

RegionBoundingRectangle_FixedAngle_OrNil

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the smallest rectangle with the given orientation angle containing a region; returns NIL if the region is empty.

Syntax

```
void avl::RegionBoundingRectangle_FixedAngle_OrNil
(
 const avl::Region& inRegion,
 float inAngle,
 atl::Conditional<avl::Rectangle2D>& outBoundingRectangle,
 atl::Conditional<avl::Point2D>& outCenter,
 atl::Conditional<float>& outLongSide,
 atl::Conditional<float>& outShortSide
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inAngle	float		Expected angle of the resulting rectangle
←	outBoundingRectangle	Conditional<Rectangle2D>&		Smallest bounding rectangle of the input region
←	outCenter	Conditional<Point2D>&		Center of the bounding rectangle
←	outLongSide	Conditional<float>&		Length of the bounding rectangle long side
←	outShortSide	Conditional<float>&		Length of the bounding rectangle short side

RegionBoundingRectangle_OrNil

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the smallest rectangle containing a region; returns NIL if the region is empty.

Syntax

```
void avl::RegionBoundingRectangle_OrNil
(
 const avl::Region& inRegion,
 avl::BoundingRectangleFeature::Type inBoundingRectangleFeature,
 float inReferenceAngle,
 avl::RectangleOrientation::Type inRectangleOrientation,
 atl::Conditional<avl::Rectangle2D>& outBoundingRectangle,
 atl::Conditional<avl::Point2D>& outCenter,
 atl::Conditional<float>& outLongSide,
 atl::Conditional<float>& outShortSide
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inBoundingRectangleFeature	BoundingRectangleFeature::Type	MinimalArea	Determines what kind of bounding rectangle will be computed
→	inReferenceAngle	float	0.0f	The middle angle of the valid range of the output rectangle's angle
→	inRectangleOrientation	RectangleOrientation::Type	Horizontal	Orientation of the output rectangle
←	outBoundingRectangle	Conditional<Rectangle2D>&		Smallest bounding rectangle of the input region
←	outCenter	Conditional<Point2D>&		Center of the bounding rectangle
←	outLongSide	Conditional<float>&		Length of the bounding rectangle long side
←	outShortSide	Conditional<float>&		Length of the bounding rectangle short side

RegionCaliperDiameter

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the longest and the shortest width of the input region measured as distance between parallel lines containing the whole region.

Syntax

```
void avl::RegionCaliperDiameter
(
 const avl::Region& inRegion,
 atl::Optional<avl::Segment2D&> outMinDiameter = atl::NIL,
 atl::Optional<float&> outMinDiameterLength = atl::NIL,
 atl::Optional<avl::Segment2D&> outMaxDiameter = atl::NIL,
 atl::Optional<float&> outMaxDiameterLength = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outMinDiameter	Optional<Segment2D&>	NIL	
←	outMinDiameterLength	Optional<float&>	NIL	
←	outMaxDiameter	Optional<Segment2D&>	NIL	
←	outMaxDiameterLength	Optional<float&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outMinDiameter**, **outMinDiameterLength**, **outMaxDiameter**, **outMaxDiameterLength**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty region on input in RegionCaliperDiameter.

RegionCircularity

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the area of a region divided by the area of a circular region having the same feature.

Syntax

```
void avl::RegionCircularity
(
 const avl::Region& inRegion,
 const avl::CircularityMeasure::Type inCircularityMeasure,
 float& outCircularity,
 avl::Circle2D& diagCircle
)
```

Parameters

	Name	Type	Default	Description
➡	inRegion	const Region&		Input region
➡	inCircularityMeasure	const CircularityMeasure::Type	RadiusPreserving	Which algorithm should be used to compute a circle
⬅	outCircularity	float&		
🔍	diagCircle	Circle2D&		Computed circle which area was compared.

Description

Circularity is a measure of similarity of a region shape to the perfect circle. Circular regions have circularity close to 1.0, while the more elongated the region is (or contains more holes), the closer to 0.0 is its circularity.

Mathematically, the circularity is calculated as follows:

$$Circularity(r) = \frac{Area(r)}{Area(c)}$$

Where **c** denotes a circular region having the same feature as input region **r**. The feature being considered depends on the **inCircularityMeasure** chosen and it is:

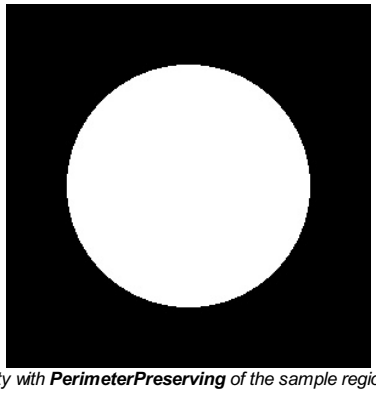
- the minimal bounding circle in case of **BoundingCirclePreserving**
- the perimeter in case of **PerimeterPreserving**
- the radius (maximal distance from mass center to any of the region pixels) in case of **RadiusPreserving**

Hints

- If the input region is not guaranteed to be non-empty, precede this filter with SkipEmptyRegion.

Examples

*Circularity with **RadiusPreserving** of the sample region equals to 0.667.*



*Circularity with **PerimeterPreserving** of the sample region equals to 1.000.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty region on input in RegionCircularity.
DomainError	Not supported circularity measure in RegionCircularity.

See Also

- RegionConvexity – Computes the area of a region divided by area of its convex hull.
- RegionElongation – Computes the elongation factor of a region (perfect circle has minimal elongation equal 1.0).
- PolygonCircularity – Computes the area of a polygon divided by the area of a circle having the same feature.

RegionComplexity

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Calculates how much the input region's perimeter is bigger than the perimeter of a circle of the same area.

Syntax

```
void avl::RegionComplexity
(
 const avl::Region& inRegion,
 float& outComplexity
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outComplexity	float&		

RegionContours

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes an array of closed paths corresponding to the contours of the input region.

Syntax

```
void avl::RegionContours
(
 const avl::Region& inRegion,
 const avl::RegionContourMode::Type inContourMode,
 avl::RegionConnectivity::Type inRegionConnectivity,
 atl::Array<avl::Path>& outContours
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inContourMode	const RegionContourMode::Type		
→	inRegionConnectivity	RegionConnectivity::Type	EightDirections	Region connectivity semantics
←	outContours	Array<Path>&		

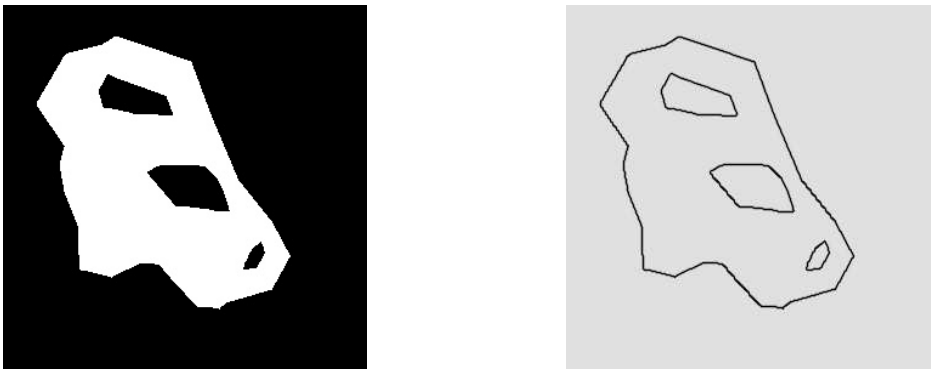
Description

The operation forms an array of paths that, depending on the value of the **inContourMode** parameter:

- Pass through the centers of boundary pixels of the region, if **inContourMode** is set to **PixelCenters**
- Run along the edges of boundary pixels of the region, if **inContourMode** is set to **PixelEdges**

To compute the approximation of the region perimeter length, one can use RegionPerimeterLength filter.

Examples



*The computed **outContours** array drawn onto an empty image.*

See Also

- RegionBoundaries – Removes interior pixels from a region.
- RegionPerimeterLength – Computes the length of the input region perimeter.

RegionConvexity

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the area of a region divided by area of its convex hull.

Syntax

```
void avl::RegionConvexity
(
 const avl::Region& inRegion,
 float& outConvexity
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outConvexity	float&		

Description

Convexity is a measure of how close a region is to being convex. Convex regions have convexity equal to 1.0, while the more concave the region is, the closer to 0.0 is its convexity.

Mathematically, the convexity is calculated as follows:

$$Convexity(r) = \frac{Area(r)}{Area(ConvexHull(r))}$$

Hints

- If the input region is not guaranteed to be non-empty, precede this filter with SkipEmptyRegion.

Examples

*Area of the sample region (on the left) equals to 2345, while area of its convex hull (on the right) equals to 40357, so the **RegionConvexity** run on the sample region would produce the result 2345/40357 = 0.058 .*

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty region on input in RegionConvexity.

See Also

- RegionConvexHull – Computes the smallest convex region containing the input region.
- PolygonConvexity – Computes the area of a polygon divided by the area of its convex hull.

RegionDiameter

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the longest segment connecting two pixels contained in region and its length.

Syntax

```
void avl::RegionDiameter
(
 const avl::Region& inRegion,
 atl::Optional<avl::Segment2D&> outDiameter = atl::NIL,
 atl::Optional<float&> outDiameterLength = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outDiameter	Optional<Segment2D&>	NIL	
←	outDiameterLength	Optional<float&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outDiameter**, **outDiameterLength**.

Read more about Optional Outputs.

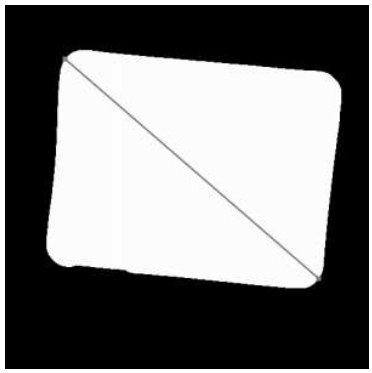
Description

The operation finds the most distant pair of region pixels and returns the distance between them (**outDiameterLength**) and the segment representing the diameter (**outDiameter**). If there is more than one pair of maximal distance, the returned segment will correspond to one of them. The orientation of the resulting **outDiameter** is always between 0 and 180 degrees.

Hints

- If the input region is not guaranteed to be non-empty, precede this filter with SkipEmptyRegion.

Examples



*The resulting **outDiameterLength** = 276.4815, **outDiameter** segment was drawn onto the region.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty region on input in RegionDiameter.

See Also

- RegionPerimeterLength – Computes the length of the input region perimeter.
- PathDiameter – Finds the longest segment connecting two characteristic points of a path.

RegionDirectionalDispersion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Directional standard deviation of the distance of the region's points to the center.

Syntax

```
void avl::RegionDirectionalDispersion
(
 const avl::Region& inRegion,
 const float inOrientation,
 const avl::ProjectionDirection::Type inDirection,
 float& outDispersion
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inOrientation	const float	.0f	
→	inDirection	const ProjectionDirection::Type	Horizontal	
←	outDispersion	float&		

RegionDispersion

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Calculates how much more the input region's points are dispersed comparing to the points of a circle of the same area.

Syntax

```
void avl::RegionDispersion
(
 const avl::Region& inRegion,
 float& outDispersion
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outDispersion	float&		

RegionEllipticAxes

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes axes of an ellipse having the same first and second order moments as the given region.

Syntax

```
void avl::RegionEllipticAxes
(
 const avl::Region& inRegion,
 avl::Segment2D& outMajorAxis,
 avl::Segment2D& outMinorAxis
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outMajorAxis	Segment2D&		
←	outMinorAxis	Segment2D&		

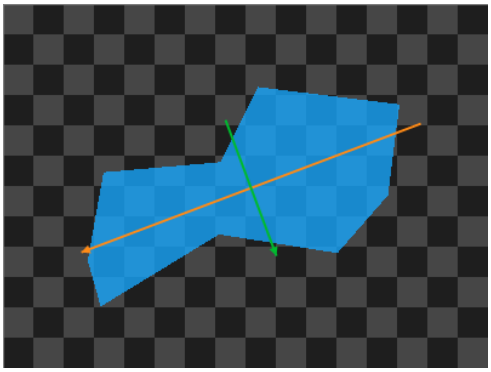
Description

The orientations of the resulting axes are always between 0 and 180 degrees.

Hints

- If the input region is not guaranteed to be non-empty, precede this filter with SkipEmptyRegion.

Examples



RegionEllipticAxes *performed on a sample region.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty region on input in RegionEllipticAxes.

RegionElongation

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the elongation factor of a region (perfect circle has minimal elongation equal 1.0).

Syntax

```
void avl::RegionElongation
(
 const avl::Region& inRegion,
 float& outElongation
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outElongation	float&		

Description

Elongation is a measure of how long the shape is in relation to its width. The perfect circle has the minimal elongation equal to 1.0, while the upper bound doesn't exist. The operation internally approximates the region with an ellipse, and then computes the result as:

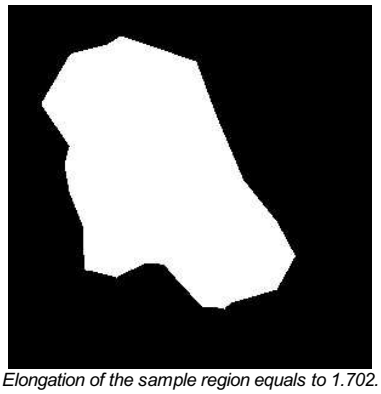
$$Elongation(r) = \frac{Length(L)}{Length(S)}$$

Where **L** denotes the longer axis of the approximating ellipse, and **S** denotes the shorter one.

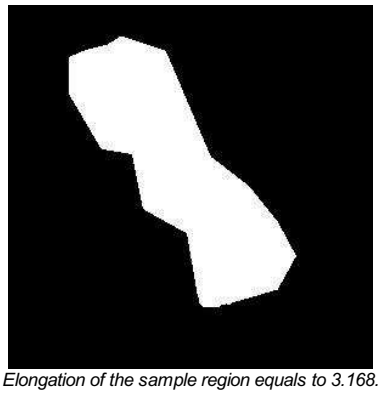
Hints

- If the input region is not guaranteed to be non-empty, precede this filter with SkipEmptyRegion.

Examples



Elongation of the sample region equals to 1.702.



Elongation of the sample region equals to 3.168.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty region on input in RegionElongation.

See Also

- RegionCircularity – Computes the area of a region divided by the area of a circular region having the same feature.
- PolygonElongation – Computes the elongation factor of a polygon (perfect circle has minimal elongation equal 1.0).

RegionFeatureValue

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the feature value of the input region.

Syntax

```
void avl::RegionFeatureValue
(
 const avl::Region& inRegion,
 avl::RegionFeature::Type inFeature,
 float& outValue
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inFeature	RegionFeature::Type		Region feature value to be computed
←	outValue	float&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect RegionFeature in RegionFeatureValue.

RegionHoles

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Creates regions of the holes of the input region.

Syntax

```
void avl::RegionHoles
(
 const avl::Region& inRegion,
 avl::RegionConnectivity::Type inConnectivity,
 int inMinHoleArea,
 atl::Optional<int> inMaxHoleArea,
 atl::Array<avl::Region> & outHoles
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inRegion	const Region&			Input region
➡	inConnectivity	RegionConnectivity::Type			Region connectivity semantics
➡	inMinHoleArea	int	0 - ∞	1	Minimal area of a resulting hole
➡	inMaxHoleArea	Optional<int>	0 - ∞	NIL	Maximal area of a resulting hole
⬅	outHoles	Array<Region> &			

Description

The operation computes regions representing holes of input region. Holes of a region are those connected areas of pixels **not** belonging to the region, that do not touch the boundary of the region frame.

Examples

RegionHoles *run on a sample region.*

See Also

- RegionNumberOfHoles – Computes the number of holes in a region.
- FillRegionHoles – Adds pixels to the input region so that it contains no holes.

RegionHoles_Elastic

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Fills a ridges region with convex basins.

Syntax

```
void avl::RegionHoles_Elastic
(
 const avl::Region& inRegion,
 atl::Optional<const avl::Region&> inRoi,
 int inDistanceThreshold,
 atl::Array<avl::Region>& outBasins
)
```

Parameters

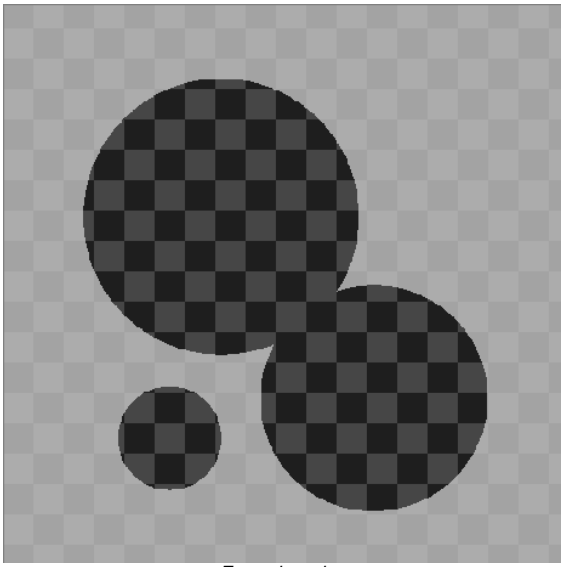
	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inDistanceThreshold	int	5	
←	outBasins	Array<Region>&		

Description

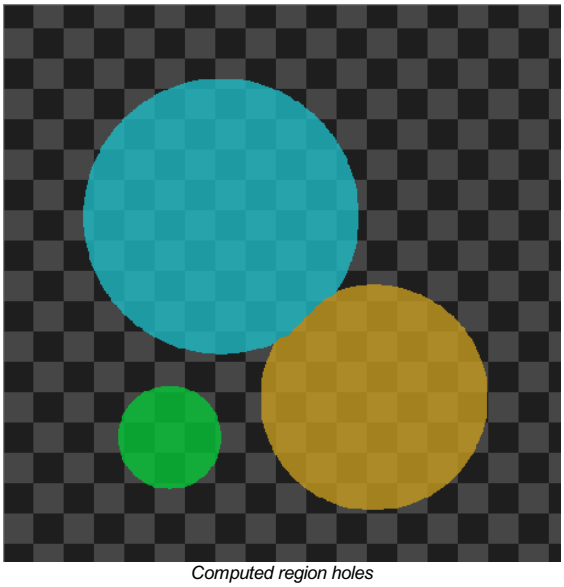
This filter fills a regions holes with convex basins based on a flooding algorithm.

The parameter **inDistanceThreshold** can be used to ignore smaller holes of the input region.

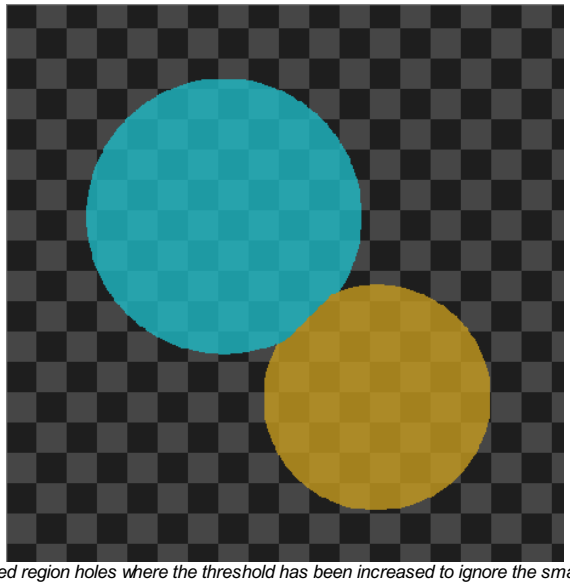
Examples



Example region



Computed region holes



Computed region holes where the threshold has been increased to ignore the smaller hole

See Also

- SegmentImage_Watersheds – Computes dark or bright watershed basins of an image.

RegionInscribedBox

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the largest box contained in a region.

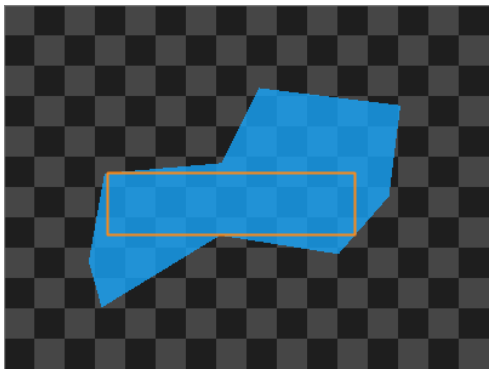
Syntax

```
void avl::RegionInscribedBox
(
 const avl::Region& inRegion,
 const float inMinAspectRatio,
 atl::Optional<float> inMaxAspectRatio,
 const int inMinWidth,
 atl::Optional<int> inMaxWidth,
 const int inMinHeight,
 atl::Optional<int> inMaxHeight,
 atl::Conditional<avl::Box>& outBox
)
```

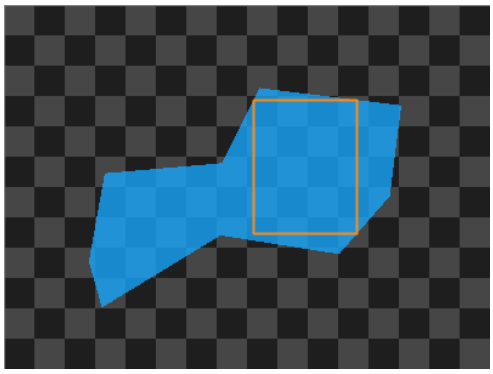
Parameters

	Name	Type	Range	Default	Description
→	inRegion	const Region&			Input region
→	inMinAspectRatio	const float	0.0 - ∞	0.0f	Minimal aspect ratio of found box
→	inMaxAspectRatio	Optional<float>	0.0 - ∞	NIL	Maximal aspect ratio of found box (reciprocal of inMinAspectRatio by default)
→	inMinWidth	const int	1 - 65535	1	Minimal width of found box
→	inMaxWidth	Optional<int>	1 - 65535	NIL	Maximal width of found box
→	inMinHeight	const int	1 - 65535	1	Minimal height of found box
→	inMaxHeight	Optional<int>	1 - 65535	NIL	Maximal height of found box
←	outBox	Conditional<Box>&			Found box with largest area

Examples



RegionInscribedBox *performed on a sample region with* ***inMaxAspectRatio*** *= Auto.*



RegionInscribedBox *performed on a sample region with* ***inMaxAspectRatio*** *= 2.0.*

RegionInscribedCircle

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the largest circle contained in a region.

Syntax

```
void avl::RegionInscribedCircle
(
 const avl::Region& inRegion,
 avl::Circle2D& outCircle
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outCircle	Circle2D&		Found circle with largest radius

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty region on input in RegionInscribedCircle.

RegionMassCenter

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes a point with coordinates equal to the average coordinates of the region's pixels.

Syntax

```
void avl::RegionMassCenter
(
 const avl::Region& inRegion,
 avl::Point2D& outMassCenter,
 atl::Optional<int&> outArea = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outMassCenter	Point2D&		
←	outArea	Optional<int&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outArea**.

Read more about Optional Outputs.

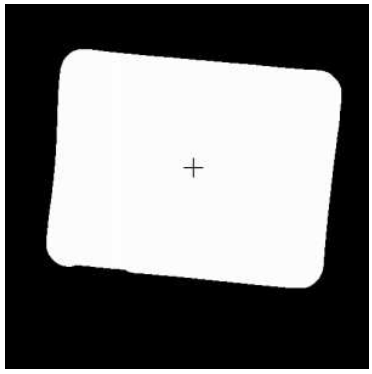
Description

The operation computes the average of all pixel locations contained in a region. Note that the result is a Point2D, not a Location as its coordinates may be not-integer.

Hints

- If the input region is not guaranteed to be non-empty, precede this filter with SkipEmptyRegion.

Examples



*The resulting **outMassCenter** point drawn onto the sample region.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Input region is empty in RegionMassCenter.

See Also

- PolygonMassCenter – Computes the mass center of polygon.

RegionMassCenter_OrNil

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes a point with coordinates equal to the average coordinates of the region's pixels; returns NIL if the region is empty.

Syntax

```
void avl::RegionMassCenter_OrNil
(
 const avl::Region& inRegion,
 atl::Conditional<avl::Point2D>& outMassCenter,
 atl::Optional<atl::Conditional<int>&> outArea = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outMassCenter	Conditional<Point2D>&		
←	outArea	Optional<Conditional<int>&>	NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outArea**.

Read more about Optional Outputs.

RegionMedialAxis

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes an array of paths corresponding to the skeleton of the input region.

Syntax

```
void avl::RegionMedialAxis
(
 const avl::Region& inRegion,
 avl::RegionSkeletonMethod::Type inRegionSkeletonMethod,
 atl::Array<avl::Path>& outSkeletonPaths
)
```

Parameters

	Name	Type	Default	Description
➡	inRegion	const Region&		Input region
➡	inRegionSkeletonMethod	RegionSkeletonMethod::Type	TwelveConnected	
⬅	outSkeletonPaths	Array<Path>&		

Description

The operation performs skeletonization presenting result as an array of paths.

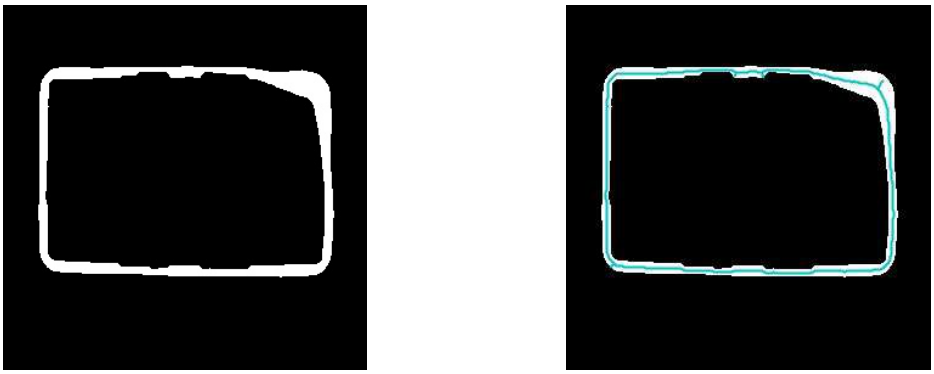
Skeleton of a region is a connected set of medial axis of its limbs. It is a useful tool when one is interested only in general structure of a shape and wants to disregard width of its limbs. Two methods are available, depending on the value of **inRegionSkeletonMethod** being chosen:

- **EightConnected**: the input region is thinned as long as the thinning procedure affects it; the structuring elements come from the well known Golay alphabet
- **TwelveConnected**: the input region is thinned also, but the structuring elements being used are slightly larger than in previous method (they consider 12-neighborhood of the given pixel instead of 8-neighborhood). The method comes from the paper of U. Eckhardt and G. Maderlechner "Invariant thinning"

The second method is slower than the first one, but its results look in most cases much better than the first one's results.

This filter is a cousin of SkeletonizeRegion which represents the results as a region instead of an array of paths.

Examples



*The resulting **outSkeletonPaths** paths drawn onto the input region, **inRegionSkeletonMethod** = **TwelveConnected**.*

See Also

- SkeletonizeRegion – Thins a region to its skeleton.

RegionMoment

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes selected second-order moment of a region in regular and normalized (divided by region area) variant.

Syntax

```
void avl::RegionMoment
(
 const avl::Region& inRegion,
 avl::RegionMomentType::Type inMomentType,
 bool inCentral,
 float& outMoment,
 float& outNormMoment
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inMomentType	RegionMomentType::Type		
→	inCentral	bool		
←	outMoment	float&		
←	outNormMoment	float&		

Description

The operation computes the mathematical features of a shape called moments. Those are sums computed as follows:

$$Moment_{2,0}(R) = \sum_{p \in R} p_x^2$$
$$Moment_{1,1}(R) = \sum_{p \in R} p_x p_y$$
$$Moment_{0,2}(R) = \sum_{p \in R} p_y^2$$

The summing is conducted over region pixels, while p_x and p_y denote, accordingly, x and y coordinate of a pixel.

When **inCentral** parameter is set, the region is shifted before computations, so that its mass center is at location (0,0).

Hints

- If the input region is not guaranteed to be non-empty, precede this filter with SkipEmptyRegion.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty region on input in RegionMoment.
DomainError	Not supported moment type in RegionMoment.

See Also

- PolygonMoment – Computes the selected second-order moment of a polygon in regular and normalized (divided by polygon area) variant.
- ImageMoment – Computes the selected moment of an image in regular and normalized (divided by sum of pixel values) variant.

RegionNumberOfHoles

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the number of holes in a region.

Syntax

```
void avl::RegionNumberOfHoles
(
 const avl::Region& inRegion,
 avl::RegionConnectivity::Type inConnectivity,
 int inMinHoleArea,
 atl::Optional<int> inMaxHoleArea,
 int& outNumberOfHoles
)
```

Parameters

	Name	Type	Range	Default	Description
→	inRegion	const Region&			Input region
→	inConnectivity	RegionConnectivity::Type			Region connectivity semantics
→	inMinHoleArea	int	0 - ∞	1	Minimal area of a hole
→	inMaxHoleArea	Optional<int>	0 - ∞	NIL	Maximal area of a hole
←	outNumberOfHoles	int&			

Description

The operation computes the number of region holes. Holes of a region are those connected areas of pixels **not** belonging to the region, that do not touch the boundary of the region frame.

Examples

The number of holes in the sample (left) region equals to 3. The holes are presented on the right image.

See Also

- FillRegionHoles – Adds pixels to the input region so that it contains no holes.

RegionOrientation

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the orientation of a region as an angle of value in a proper range.

Syntax

```
void avl::RegionOrientation
(
 const avl::Region& inRegion,
 avl::AngleRange::Type inAngleRange,
 float& outOrientationAngle
)
```

Parameters

	Name	Type	Default	Description
➡	inRegion	const Region&		Input region
➡	inAngleRange	AngleRange::Type	_0_180	Switches between ranges <0; 90), <0; 180) and <0; 360)
⬅	outOrientationAngle	float&		

Description

Region orientation can be thought of as the direction in which the region is oriented. Mathematically it is the angle between X-axis and the line passing through the region mass center, that rotation around this line produces the smallest torque.

Hints

- If the input region is not guaranteed to be non-empty, precede this filter with SkipEmptyRegion.

Examples

Orientation of the sample region equals to 53.496, which is visualized on the second image by drawing the line of this orientation passing through the region mass center.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty region on input in RegionOrientation.

See Also

- PolygonOrientation – Computes the polygon orientation as angle with value in range 0.0 - 180.0.

RegionPerimeterLength

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the length of the input region perimeter.

Syntax

```
void avl::RegionPerimeterLength
(
 const avl::Region& inRegion,
 float& outPerimeterLength
)
```

Parameters

	Name	Type	Default	Description
➡	inRegion	const Region&		Input region
⬅	outPerimeterLength	float&		

Description

The operation computes the perimeter length of a region **shape**. Because regions are pixel-precise, the literal computation of the length of the region contour would lead to the overestimation of the diagonal edges. Therefore, the filter employs the following equation to approximate the result:

$$Perimeter(r) \approx a \cdot sides - b \cdot vertices$$

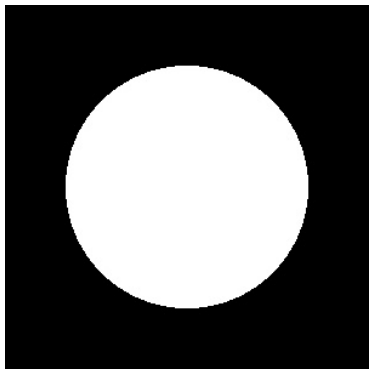
Where

- **sides** denotes the number of pixel sides adjacent to the region background
- **vertices** is the number of turns to right during clockwise walk along the pixel sides of the region boundary
- **a**, **b** are constants defined as follows:

$$a = \frac{\pi(1+\sqrt{2})}{8}$$
$$b = \frac{\pi}{4\sqrt{2}}$$

To compute the actual contour of the region, one can use RegionContours filter.

Examples



RegionPerimeterLength *run on the sample region (circular region of radius 100) computes* ***outPerimeterLength*** *= 627.382, while the actual perimeter of the perfect circle of radius 100 is 628.318 .*

See Also

- RegionContours – Computes an array of closed paths corresponding to the contours of the input region.
- RegionDiameter – Computes the longest segment connecting two pixels contained in region and its length.

RegionPointRunLengths

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Returns the lengths of horizontal sequences of locations that constitute the input region.

Syntax

```
void avl::RegionPointRunLengths
(
 const avl::Region& inRegion,
 atl::Array< int >& outPointRunLengths
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outPointRunLengths	Array< int >&		

RegionProjection

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the profile of the region pixel count in consecutive rows or columns.

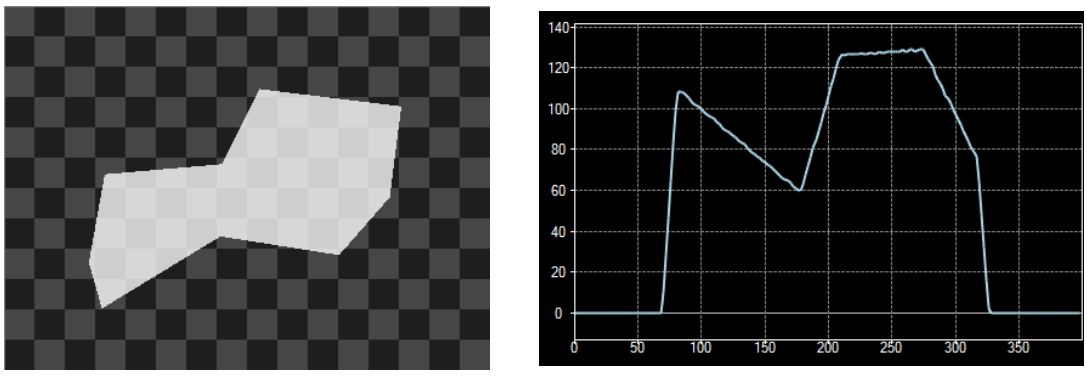
Syntax

```
void avl::RegionProjection
(
 const avl::Region& inRegion,
 const avl::ProjectionDirection::Type inProjectionDirection,
 avl::Profile& outProfile
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inProjectionDirection	const ProjectionDirection::Type		Direction in which the pixel occurrences are counted, 'horizontal' means summation row after row, 'vertical' indicates summation column after column
←	outProfile	Profile&		Profile of pixel count in consecutive rows/columns

Examples



RegionProjection *performed on a sample region with* ***inProjectionDirection*** *= Vertical.*

RegionRectangularity

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the area of a region divided by the area of its bounding rectangle.

Syntax

```
void avl::RegionRectangularity
(
 const avl::Region& inRegion,
 float& outRectangularity
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outRectangularity	float&		

Hints

- If the input region is not guaranteed to be non-empty, precede this filter with SkipEmptyRegion.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty region on input in RegionRectangularity.

RegionsFeatureValues

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes the feature values of the input regions.

Syntax

```
void avl::RegionsFeatureValues
(
 const atl::Array<avl::Region>& inRegions,
 avl::RegionFeature::Type inFeature,
 atl::Array<float>& outValues
)
```

Parameters

	Name	Type	Default	Description
→	inRegions	const Array<Region>&		Input regions
→	inFeature	RegionFeature::Type		Region feature value to be computed
←	outValues	Array<float>&		

RegionSkewness

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Directional standard deviation of the distance of the region's points to the center.

Syntax

```
void avl::RegionSkewness
(
 const avl::Region& inRegion,
 const float inOrientation,
 const avl::ProjectionDirection::Type inDirection,
 float& outSkewness
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
→	inOrientation	const float	.0f	
→	inDirection	const ProjectionDirection::Type	Horizontal	
←	outSkewness	float&		

RegionThickness

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Diameter of the biggest inscribed circle.

Syntax

```
void avl::RegionThickness
(
 const avl::Region& inRegion,
 float& outThickness
)
```

Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outThickness	float&		

121. Region Point Transforms

Table of content:

RegionComplement

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes a region of pixels not contained in the input region.

Syntax

```
void avl::RegionComplement
(
 const avl::Region& inRegion,
 avl::Region& outRegion
)
```

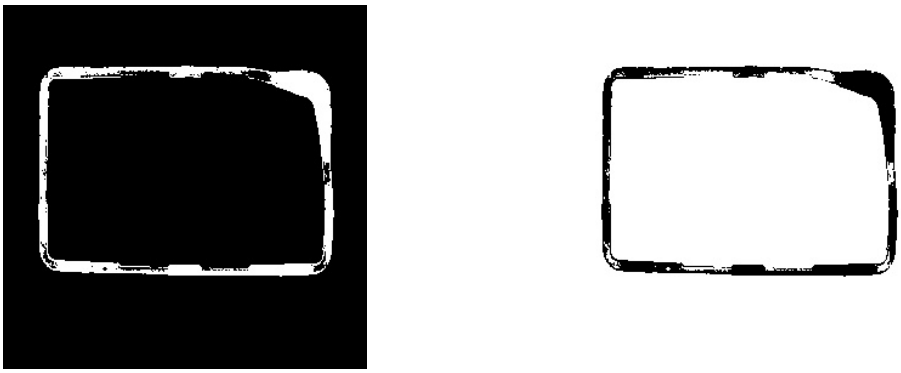
Parameters

	Name	Type	Default	Description
→	inRegion	const Region&		Input region
←	outRegion	Region&		Output region

Description

The operation computes a region of the same dimensions, containing only pixels that do not belong to the given one.

Examples



RegionComplement *run on a sample region.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Input and output regions are not distinct in RegionComplement.

122. Region Combinators

Table of content:

RegionDifference

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes a region containing pixels from the first input region but not from the second input region.

Syntax

```
void avl::RegionDifference
(
 const avl::Region& inRegion1,
 const avl::Region& inRegion2,
 avl::Region& outRegion
)
```

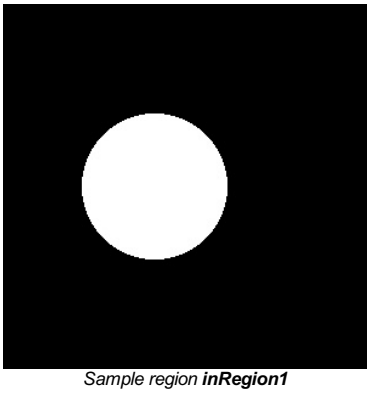
Parameters

	Name	Type	Default	Description
➡	inRegion1	const Region&		First input region
➡	inRegion2	const Region&		Second input region
⬅	outRegion	Region&		Output region

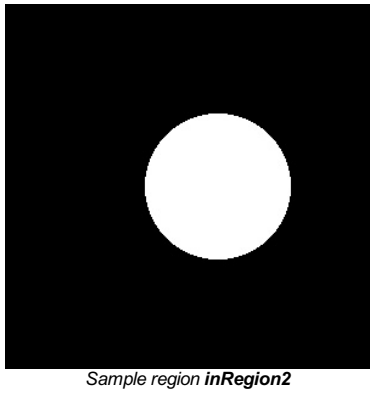
Description

The operation computes the logical difference of two given regions. That is, the resulting region contains pixel locations that belong to the first input region but do **not** belong to the second one. Both dimensions (width and height) of the resulting region are set to the corresponding dimensions of the **first** input region.

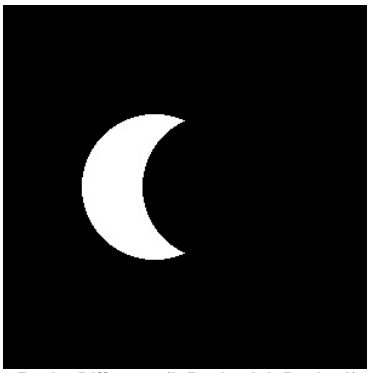
Examples



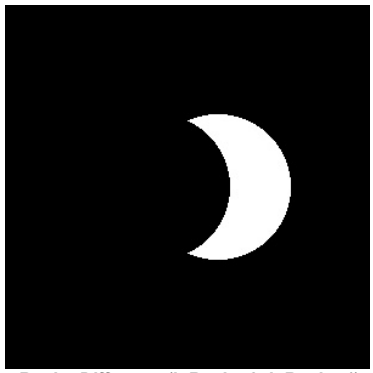
*Sample region **inRegion1***



*Sample region **inRegion2***



RegionDifference(inRegion1, inRegion2)



RegionDifference(inRegion2, inRegion1)

Errors

List of possible exceptions:

Error type	Description
DomainError	Input and output regions are not distinct in RegionDifference.

See Also

- RegionSymmetricDifference – Computes a region containing pixels from first or second input region, but not from both.

RegionIntersection

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the common part of two regions.

Syntax

```
void avl::RegionIntersection
(
 const avl::Region& inRegion1,
 const avl::Region& inRegion2,
 avl::Region& outRegion
)
```

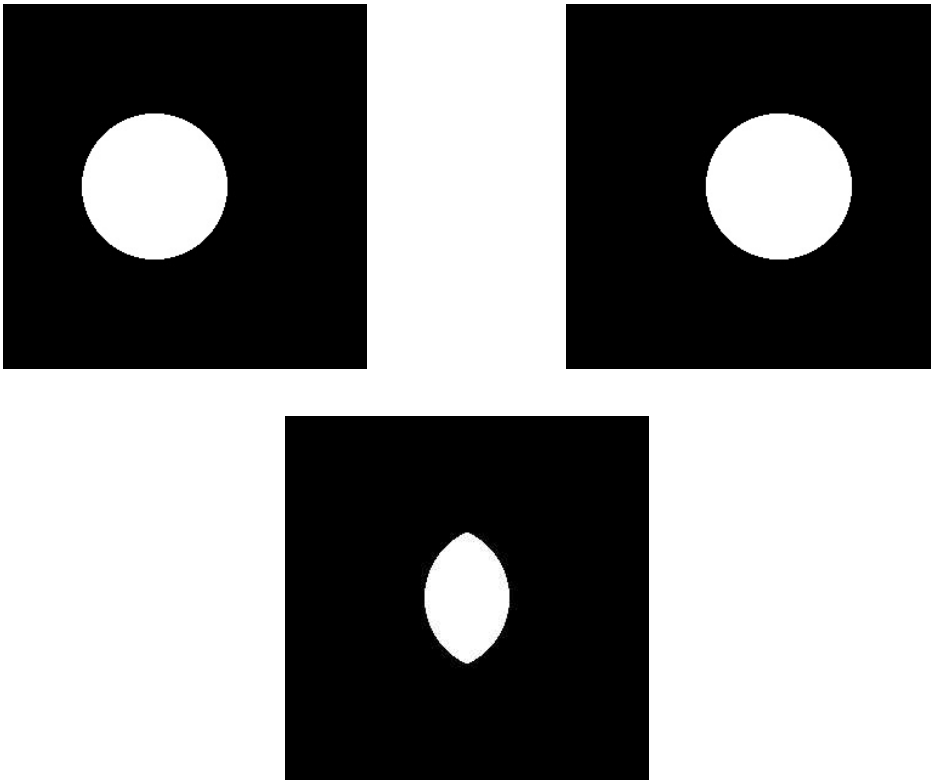
Parameters

	Name	Type	Default	Description
➡	inRegion1	const Region&		First input region
➡	inRegion2	const Region&		Second input region
⬅	outRegion	Region&		Output region

Description

The operation computes the logical intersection of two given regions. That is, the resulting region contains pixel locations that belong to both of the given regions. Both dimensions (width and height) of the resulting region are set to the **minimum** of the corresponding dimensions of the input regions.

Examples



RegionIntersection *run on the sample regions.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Input and output regions are not distinct in RegionIntersection.

See Also

- RegionUnion – Computes a region containing pixels from both input regions.

RegionIntersection_OfArray

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the common part of an array of regions.

Syntax

```
void avl::RegionIntersection_OfArray
(
 atl::Optional<const avl::Region&> inInitialRegion,
 const atl::Array<avl::Region>& inArray,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Default	Description
→	inInitialRegion	Optional<const Region&>	NIL	
→	inArray	const Array<Region>&		
←	outRegion	Region&		Output region

Description

Array version of RegionIntersection.

Errors

List of possible exceptions:

Error type	Description
DomainError	Input array is empty and inInitialRegion isn't set in RegionIntersection_OfArray.

See Also

- RegionIntersection – Computes the common part of two regions.

RegionIntersection_OfLoop

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the common part of regions appearing in consecutive iterations.

Syntax

```
void avl::RegionIntersection_OfLoop
(
 RegionCombinators_OfLoopState& ioState,
 const avl::Region& inRegion,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Default	Description
↔	ioState	RegionCombinators_OfLoopState&		Object used to maintain state of the function.
→	inRegion	const Region&		Input region
←	outRegion	Region&		Output region

Description

Loop version of RegionIntersection.

See Also

- RegionIntersection – Computes the common part of two regions.

RegionSymmetricDifference

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes a region containing pixels from first or second input region, but not from both.

Syntax

```
void avl::RegionSymmetricDifference
(
 const avl::Region& inRegion1,
 const avl::Region& inRegion2,
 avl::Region& outRegion
)
```

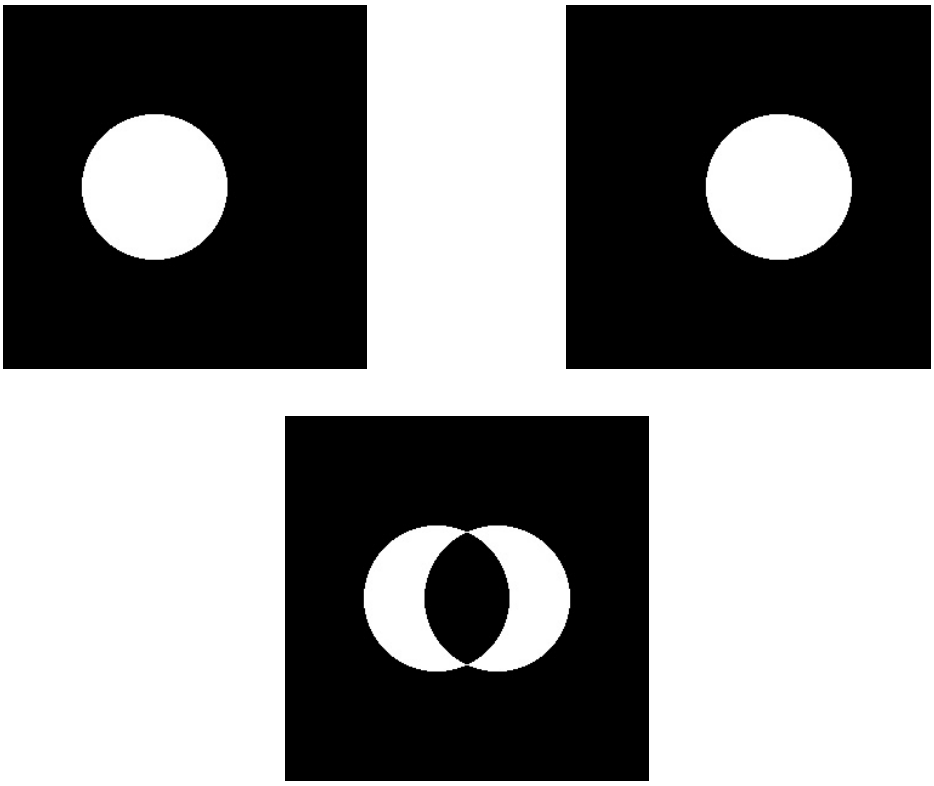
Parameters

	Name	Type	Default	Description
→	inRegion1	const Region&		First input region
→	inRegion2	const Region&		Second input region
←	outRegion	Region&		Output region

Description

The operation computes the logical symmetric difference of two given regions. That is, the resulting region contains pixel locations that belong to one of the given regions, but **not** to both of them. Both dimensions (width and height) of the resulting region are set to the maximum of the corresponding dimensions of the input regions.

Examples



RegionSymmetricDifference *run on the sample regions.*

See Also

- RegionDifference – Computes a region containing pixels from the first input region but not from the second input region.

RegionSymmetricDifference_OfArray

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes a region containing pixels belonging to an odd number of array regions.

Syntax

```
void avl::RegionSymmetricDifference_OfArray
(
 const atl::Array<avl::Region>& inRegionArray,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Default	Description
→	inRegionArray	const Array<Region>&		
←	outRegion	Region&		Output region

Description

Array version of RegionSymmetricDifference.

See Also

- RegionSymmetricDifference – Computes a region containing pixels from first or second input region, but not from both.

RegionSymmetricDifference_OfLoop

Header: AVL.h
Namespace: avl
Module: FoundationPro

Computes a region containing pixels belonging to an odd number of regions appearing in consecutive iterations.

Syntax

```
void avl::RegionSymmetricDifference_OfLoop
(
 RegionCombinators_OfLoopState& ioState,
 const avl::Region& inRegion,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Default	Description
↔	ioState	RegionCombinators_OfLoopState&		Object used to maintain state of the function.
→	inRegion	const Region&		Input region
←	outRegion	Region&		Output region

Description

Loop version of RegionSymmetricDifference.

See Also

- RegionSymmetricDifference – Computes a region containing pixels from first or second input region, but not from both.

RegionUnion

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes a region containing pixels from both input regions.

Syntax

```
void avl::RegionUnion
(
 const avl::Region& inRegion1,
 const avl::Region& inRegion2,
 avl::Region& outRegion
)
```

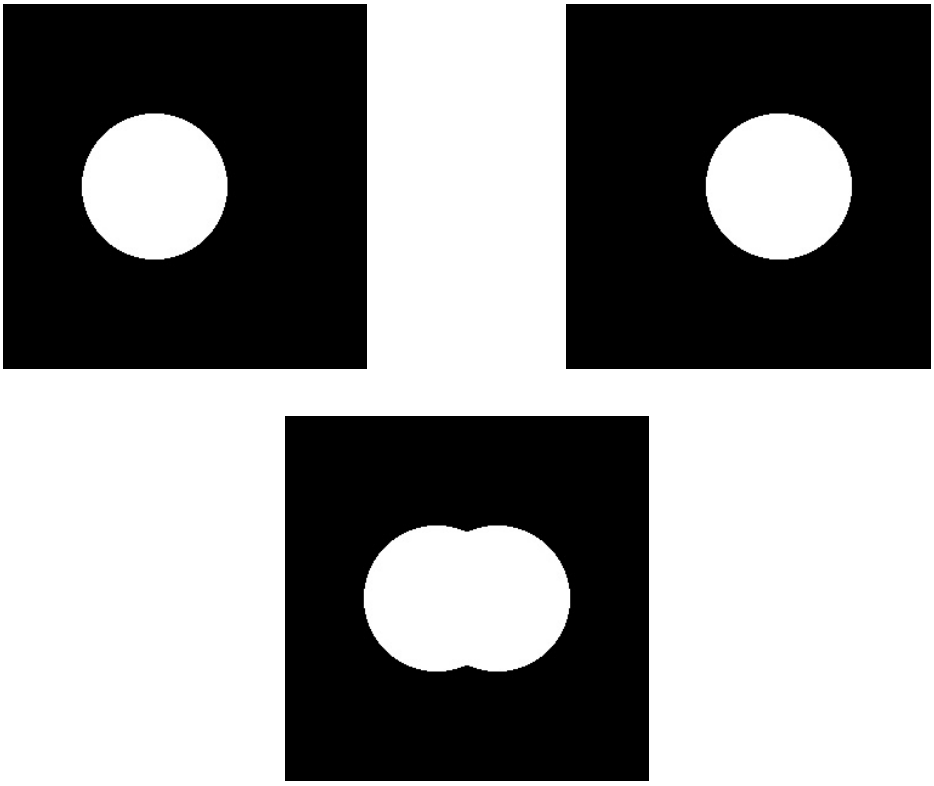
Parameters

	Name	Type	Default	Description
→	inRegion1	const Region&		First input region
→	inRegion2	const Region&		Second input region
←	outRegion	Region&		Output region

Description

The operation computes the logical union of two given regions. That is, the resulting region contains pixel locations that belong to one or both of the given regions. Both dimensions (width and height) of the resulting region are set to the **maximum** of corresponding dimensions of the input regions.

Examples



***RegionUnion** run on the sample regions.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Input and output regions are not distinct in RegionUnion.

See Also

- RegionIntersection – Computes the common part of two regions.

RegionUnion_OfArray

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes a region containing all the pixels that any of the input regions contains.

Syntax

```
void avl::RegionUnion_OfArray
(
 const atl::Array<avl::Region>& inArray,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Default	Description
→	inArray	const Array<Region>&		Array of regions
←	outRegion	Region&		Union of the input regions

Description

Array version of RegionUnion.

See Also

- RegionUnion – Computes a region containing pixels from both input regions.

RegionUnion_OfLoop

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Computes the union of regions appearing in consecutive iterations.

Syntax

```
void avl::RegionUnion_OfLoop
(
 RegionCombinators_OfLoopState& ioState,
 const avl::Region& inRegion,
 avl::Region& outRegion
)
```

Parameters

	Name	Type	Default	Description
↔	ioState	RegionCombinators_OfLoopState&		Object used to maintain state of the function.
→	inRegion	const Region&		Input region
←	outRegion	Region&		Output region

Description

Loop version of RegionUnion.

See Also

- RegionUnion – Computes a region containing pixels from both input regions.

123. Region Metrics

Table of content:

RegionToRegionDistance

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes minimal distance between one of the points of the first region with one of the points of the second region.

Syntax

```
void avl::RegionToRegionDistance
(
 const avl::Region& inRegion1,
 const avl::Region& inRegion2,
 float inResolution,
 float& outDistance,
 atl::Optional<avl::Segment2D&> outConnectingSegment = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inRegion1	const Region&			First input region
→	inRegion2	const Region&			Second input region
→	inResolution	float	0.0 - ∞	1.0f	Number of real-world units per one pixel
←	outDistance	float&			
←	outConnectingSegment	Optional<Segment2D&>		NIL	

Optional Outputs

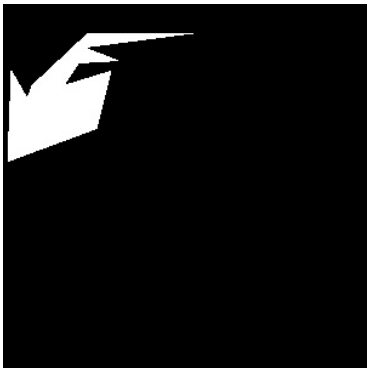
The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outConnectingSegment**.

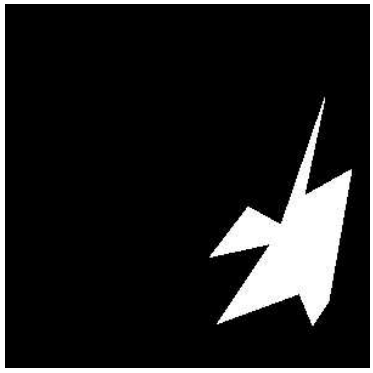
Read more about Optional Outputs.

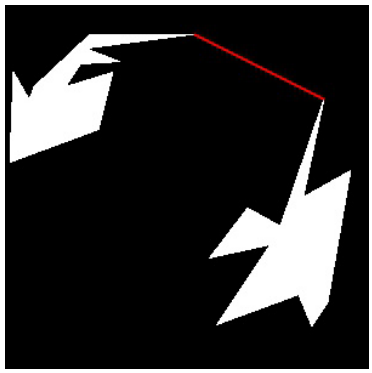
Description

Operation computes length of the shortest segment that connects center of one of **inRegion1**'s point which center of one of **inRegion2**'s point.

Examples







RegionToRegionDistance *performed on sample regions produced* ***outDistance*** *= 118,5116. The shortest segment connecting regions is presented on the bottom image*

Remarks

- **inRegion1** and **inRegion2** must not be empty, otherwise an error with appropriate description occurs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty region on input in RegionToRegionDistance.

124. Geometry 3D Normalizations

Table of content:

RemoveInvalidPoints3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Removes invalid 3D points from an array (i.e. points with NaN or INF coordinates).

Syntax

```
void avl::RemoveInvalidPoints3D
(
 const atl::Array<avl::Point3D>& inPoints3D,
 atl::Array<avl::Point3D>& outPoints3D
)
```

Parameters

	Name	Type	Default	Description
→	inPoints3D	const Array<Point3D>&		
←	outPoints3D	Array<Point3D>&		

125. Surface Features

Table of content:

ReplaceInvalidSurfacePoints

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Replaces all invalid points with a given Z value.

Syntax

```
void avl::ReplaceInvalidSurfacePoints
(
 avl::Surface& ioSurface,
 atl::Optional<const avl::Region&> inRoi,
 float inNewZ
)
```

Parameters

	Name	Type	Default	Description
↔	ioSurface	Surface&		
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inNewZ	float		

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region of interest exceeds an input surface in ReplaceInvalidSurfacePoints.

SurfaceArea

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Computes the surface area of given surface.

Syntax

```
void avl::SurfaceArea
(
 const avl::Surface& inSurface,
 float& outArea
)
```

Parameters

	Name	Type	Default	Description
→	inSurface	const Surface&		Input surface
←	outArea	float&		Area of the input surface

SurfaceBoundingBox

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Computes the bounding box 3D of given surface.

Syntax

```
void avl::SurfaceBoundingBox
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 avl::Box3D& outBoundingBox3D
)
```

Parameters

	Name	Type	Default	Description
➡	inSurface	const Surface&		Input surface
➡	inRoi	Optional<const Region&>	NIL	Region of interest
⬅	outBoundingBox3D	Box3D&		Bounding box of the surface points

Errors

List of possible exceptions:

Error type	Description
DomainError	No valid point in given region in SurfaceBoundingBox.
DomainError	Region of interest exceeds an input surface in SurfaceBoundingBox.

SurfaceBoundingBox_OrNil

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Computes the bounding box 3D of given surface; returns NIL if no valid point is present.

Syntax

```
void avl::SurfaceBoundingBox_OrNil
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 atl::Conditional<avl::Box3D>& outBoundingBox3D
)
```

Parameters

	Name	Type	Default	Description
➡	inSurface	const Surface&		Input surface
➡	inRoi	Optional<const Region&>	NIL	Region of interest
⬅	outBoundingBox3D	Conditional<Box3D>&		Bounding box of the surface points

Errors

List of possible exceptions:

Error type	Description
DomainError	Region of interest exceeds an input surface in SurfaceBoundingBox_OrNil.

SurfaceFlatness

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Computes the surface flatness i.e. how thick the surface is according to the input plane.

Syntax

```
void avl::SurfaceFlatness
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 const avl::Plane3D& inPlane,
 avl::Point3D& outMaximumPoint1,
 avl::Point3D& outMaximumPoint2,
 float& outMaximumDistance1,
 float& outMaximumDistance2,
 float& outFlatness
)
```

Parameters

	Name	Type	Default	Description
→	inSurface	const Surface&		Input surface
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inPlane	const Plane3D&		Input plane
←	outMaximumPoint1	Point3D&		Surface point with the largest distance on one side of the plane
←	outMaximumPoint2	Point3D&		Surface point with the largest distance on the other side of the plane
←	outMaximumDistance1	float&		Distance of the first maximum point from the input plane
←	outMaximumDistance2	float&		Distance of the second maximum point from the input plane
←	outFlatness	float&		Difference between two extremal distances of surface points from the input plane

Errors

List of possible exceptions:

Error type	Description
DomainError	No points available to compute surface flatness in SurfaceFlatness.
DomainError	Region of interest exceeds an input surface in SurfaceFlatness.

SurfaceLocalMaxima

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Finds surface locations characterized by locally maximal heights.

Applications: Detection of characteristic points, usually after some surface transformations.

Syntax

```
void avl::SurfaceLocalMaxima
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 bool inConsiderPlateaus,
 atl::Optional<float> inMinHeight,
 atl::Optional<float> inMaxHeight,
 float inMinDistance,
 atl::Optional<const avl::SurfaceLocalExtremaVerification&> inMaximaVerification,
 atl::Optional<atl::Array<avl::SurfaceExtremum>&> outLocalMaxima,
 atl::Optional<atl::Array<avl::Region>&> outMaximaRegions = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inConsiderPlateaus	bool			Consider multi-pixel maxima (plateaus) or not
→	inMinHeight	Optional<float>		NIL	Minimal height of maximum to be considered
→	inMaxHeight	Optional<float>		NIL	Maximal height of maximum to be considered
→	inMinDistance	float	0.0 - ∞		Minimal distance between two found maxima
→	inMaximaVerification	Optional<const SurfaceLocalExtremaVerification&>		NIL	Maxima verification structure
←	outLocalMaxima	Optional<Array<SurfaceExtremum>&>			Found local maxima
←	outMaximaRegions	Optional<Array<Region>&>		NIL	Regions of local maxima (plateaus and singletons)

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outLocalMaxima**, **outMaximaRegions**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region of interest exceeds an input surface in SurfaceLocalMaxima.

SurfaceLocalMinima

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Finds surface locations characterized by locally minimal heights.

Applications: Detection of characteristic points, usually after some surface transformations.

Syntax

```
void avl::SurfaceLocalMinima
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 bool inConsiderLowlands,
 atl::Optional<float> inMinHeight,
 atl::Optional<float> inMaxHeight,
 float inMinDistance,
 atl::Optional<const avl::SurfaceLocalExtremaVerification&> inMinimaVerification,
 atl::Optional<atl::Array<avl::SurfaceExtremum>&> outLocalMinima,
 atl::Optional<atl::Array<avl::Region>&> outMinimaRegions = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inConsiderLowlands	bool			Consider multi-pixel minima (lowlands) or not
→	inMinHeight	Optional<float>		NIL	Minimal height of minimum to be considered
→	inMaxHeight	Optional<float>		NIL	Maximal height of minimum to be considered
→	inMinDistance	float	0.0 - ∞		Minimal distance between two found minima
→	inMinimaVerification	Optional<const SurfaceLocalExtremaVerification&>		NIL	Minima verification structure
←	outLocalMinima	Optional<Array<SurfaceExtremum>&>			Found local minima
←	outMinimaRegions	Optional<Array<Region>&>		NIL	Regions of local minima (plateaus and singletons)

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outLocalMinima**, **outMinimaRegions**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region of interest exceeds an input surface in SurfaceLocalMinima.

SurfaceMassCenter

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Computes the mass center of surface points.

Syntax

```
void avl::SurfaceMassCenter
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 avl::Point3D& outMassCenter
)
```

Parameters

	Name	Type	Default	Description
➡	inSurface	const Surface&		Input surface
➡	inRoi	Optional<const Region&>	NIL	Region of interest
⬅	outMassCenter	Point3D&		Mass center of the surface points

Errors

List of possible exceptions:

Error type	Description
DomainError	No valid point in given region in SurfaceMassCenter.
DomainError	Region of interest exceeds an input surface in SurfaceMassCenter.

SurfaceMassCenter_OrNil

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Computes the mass center of surface points; returns NIL if the surface is empty in the given region.

Syntax

```
void avl::SurfaceMassCenter_OrNil
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 atl::Conditional<avl::Point3D>& outMassCenter
)
```

Parameters

	Name	Type	Default	Description
➡	inSurface	const Surface&		Input surface
➡	inRoi	Optional<const Region&>	NIL	Region of interest
⬅	outMassCenter	Conditional<Point3D>&		Mass center of the surface points

Errors

List of possible exceptions:

Error type	Description
DomainError	Region of interest exceeds an input surface in SurfaceMassCenter_OrNil.

SurfaceMaximalPoint

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Finds the surface point with maximal Z coordinate.

Syntax

```
void avl::SurfaceMaximalPoint
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 avl::Point3D& outMaximalPoint
)
```

Parameters

	Name	Type	Default	Description
→	inSurface	const Surface&		Input surface
→	inRoi	Optional<const Region&>	NIL	Region of interest
←	outMaximalPoint	Point3D&		Point with maximal Z coordinate

Errors

List of possible exceptions:

Error type	Description
DomainError	No valid point in given region while computing Surface extremal point.
DomainError	Region of interest exceeds an input surface in SurfaceMaximalPoint.

SurfaceMedian

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Finds the median Z coordinate of the surface points.

Syntax

```
void avl::SurfaceMedian
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 float& outMedian
)
```

Parameters

	Name	Type	Default	Description
→	inSurface	const Surface&		Input surface
→	inRoi	Optional<const Region&>	NIL	Region of interest
←	outMedian	float&		Median Z coordinate of the surface points

Errors

List of possible exceptions:

Error type	Description
DomainError	No valid point in given region in SurfaceMedian.
DomainError	Region of interest exceeds an input surface in SurfaceMedian.

SurfaceMedian_OrNil

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Finds the median Z coordinate of the surface points; returns NIL if the surface is empty in the given region.

Syntax

```
void avl::SurfaceMedian_OrNil
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 atl::Conditional<float>& outMedian
)
```

Parameters

	Name	Type	Default	Description
➡	inSurface	const Surface&		Input surface
➡	inRoi	Optional<const Region&>	NIL	Region of interest
⬅	outMedian	Conditional<float>&		Median Z coordinate of the surface points

Errors

List of possible exceptions:

Error type	Description
DomainError	Region of interest exceeds an input surface in SurfaceMedian_OrNil.

SurfaceMinimalPoint

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Finds the surface point with minimal Z coordinate.

Syntax

```
void avl::SurfaceMinimalPoint
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 avl::Point3D& outMinimalPoint
)
```

Parameters

	Name	Type	Default	Description
➡	inSurface	const Surface&		Input surface
➡	inRoi	Optional<const Region&>	NIL	Region of interest
⬅	outMinimalPoint	Point3D&		Point with minimal Z coordinate

Errors

List of possible exceptions:

Error type	Description
DomainError	No valid point in given region while computing Surface extremal point.
DomainError	Region of interest exceeds an input surface in SurfaceMinimalPoint.

SurfaceMultiplePointsAlongAxis

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Returns multiple arrays of surface points along X or Y axis.

Syntax

```
void avl::SurfaceMultiplePointsAlongAxis
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Surface&> inSurface2,
 avl::Axis::Type inAxis,
 atl::Optional<double> inCoordinateValueStart,
 atl::Optional<double> inCoordinateValueEnd,
 atl::Optional<double> inCoordinateValueStep,
 int inSmoothRadius,
 atl::Optional<double> inMinOutputCoordinate,
 atl::Optional<double> inMaxOutputCoordinate,
 atl::Optional<int> inMaxInterpolationLength,
 atl::Array<atl::Array<avl::Point3D>>& outPoints,
 atl::Optional<atl::Array<double>&> outCoordinateValues = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inSurface2	Optional<const Surface&>		NIL	Optional second input surface
→	inAxis	Axis::Type			Axis along which the points are extracted
→	inCoordinateValueStart	Optional<double>		NIL	Determines the coordinate the first row of points will be extracted from
→	inCoordinateValueEnd	Optional<double>		NIL	Limits the coordinate the last row of points will be extracted from
→	inCoordinateValueStep	Optional<double>	0 - ∞	NIL	Determines the distance between consecutive extracted row of points
→	inSmoothRadius	int	0 - ∞		Increases the number of neighbouring points taken into account extracting a single row of points
→	inMinOutputCoordinate	Optional<double>		NIL	Minimal second coordinate of the output points
→	inMaxOutputCoordinate	Optional<double>		NIL	Maximal second coordinate of the output points
→	inMaxInterpolationLength	Optional<int>	0 - ∞	0	Maximal number of consecutive not existing points to be interpolated
←	outPoints	Array<Array<Point3D>>&			The resulting surface points
←	outCoordinateValues	Optional<Array<double>&>		NIL	The coordinates the output points were extracted from

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outCoordinateValues**.

Read more about Optional Outputs.

SurfaceMultipleProfilesAlongAxis

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Creates the profiles of point Z values along X or Y axis.

Syntax

```
void avl::SurfaceMultipleProfilesAlongAxis
(
 const avl::Surface& inSurface,
 avl::Axis::Type inAxis,
 atl::Optional<double> inCoordinateValueStart,
 atl::Optional<double> inCoordinateValueEnd,
 atl::Optional<double> inCoordinateValueStep,
 int inSmoothRadius,
 atl::Optional<double> inProfileDomainStart,
 atl::Optional<double> inProfileDomainEnd,
 atl::Optional<int> inMaxInterpolationLength,
 float inDefaultValue,
 atl::Array<avl::Profile>& outProfiles,
 atl::Optional<atl::Array<double>&> outCoordinateValues = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inAxis	Axis::Type			Axis along which the profile is extracted
→	inCoordinateValueStart	Optional<double>		NIL	Determines the coordinate the first profile will be extracted from
→	inCoordinateValueEnd	Optional<double>		NIL	Limits the coordinate the last profile will be extracted from
→	inCoordinateValueStep	Optional<double>	0 - ∞	NIL	Determines the distance between consecutive extracted profiles
→	inSmoothRadius	int	0 - ∞		Increases the number of neighbouring profiles taken into account extracting a single profile
→	inProfileDomainStart	Optional<double>		NIL	Minimal X coordinate of the output profiles
→	inProfileDomainEnd	Optional<double>		NIL	Maximal X coordinate of the output profiles
→	inMaxInterpolationLength	Optional<int>	0 - ∞	NIL	Maximal number of consecutive not existing profile points to be interpolated
→	inDefaultValue	float			Default value of the not existing and not interpolated surface point
←	outProfiles	Array<Profile>&			The resulting profiles of the surface height
←	outCoordinateValues	Optional<Array<double>&>		NIL	The coordinates the output profiles were extracted from

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outCoordinateValues**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Coordinate value range exceeds input surface in SurfaceMultipleProfilesAlongAxis.
DomainError	Incorrect output profile domain in SurfaceMultipleProfilesAlongAxis.
DomainError	Non-positive coordinate value step in SurfaceMultipleProfilesAlongAxis.
DomainError	Unknown axis type in SurfaceMultipleProfilesAlongAxis.

SurfaceNormalsImage

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Computes a normals image for a surface

Syntax

```
void avl::SurfaceNormalsImage
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 const float inScale,
 avl::Image& outNormalsImage,
 atl::Optional<avl::Region&> outComputedRegion = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inScale	const float	0.0 - ∞	1.0f	
←	outNormalsImage	Image&			
←	outComputedRegion	Optional<Region&>		NIL	

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outComputedRegion**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region of interest exceeds an input surface in SurfaceNormalsImage.

SurfaceProfileAlongPath

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Creates a series of segments across the input path, measures the average surface height on each of the segments, and creates the final profile from those values.

Applications: This is the first step of all 1D Edge Detection 3D operations. Here available for direct use by the user.

Syntax

```
void avl::SurfaceProfileAlongPath
(
 ScanMapState& ioState,
 const avl::Surface& inSurface,
 const avl::Path& inScanPath,
 atl::Optional<const avl::CoordinateSystem2D&> inScanPathAlignment,
 atl::Optional<float> inSamplingStep,
 int inScanWidth,
 avl::InterpolationMethod::Type inSurfaceInterpolation,
 atl::Optional<int> inMaxInterpolationLength,
 float inSmoothingStdDev,
 avl::Profile& outProfile,
 avl::Path& outPath,
 atl::Optional<avl::Path&> outAlignedScanPath = atl::NIL,
 atl::Array<avl::Path>& diagSamplingPoints
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	ScanMapState&			Object used to maintain state of the function.
	inSurface	const Surface&			Input surface
	inScanPath	const Path&			Path along which the profile is extracted
	inScanPathAlignment	Optional<const CoordinateSystem2D&>		NIL	Adjusts the scan path to the position of the inspected object
	inSamplingStep	Optional<float>	0.0 - ∞	NIL	Distance between consecutive sampling points on the scan path; if Nil, the bigger of surface X and Y scales is chosen
	inScanWidth	int	1 - ∞	5	Width of the scan field
	inSurfaceInterpolation	InterpolationMethod::Type		Bilinear	Interpolation method used for extraction of surface points
	inMaxInterpolationLength	Optional<int>		NIL	Maximal number of consecutive not existing profile points
	inSmoothingStdDev	float	0.0 - ∞	0.6f	Standard deviation of the gaussian smoothing applied to the extracted profile
	outProfile	Profile&			The resulting profile of the surface height
	outPath	Path&			The path consisting of the points from which the resulting profile is extracted
	outAlignedScanPath	Optional<Path&>		NIL	Path along which the scan is performed
	diagSamplingPoints	Array<Path>&			Array of paths each one containing the sampling points that contributed to a single value of the extracted profile; in the image coordinate system

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAlignedScanPath**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Non-positive sampling step on input in SurfaceProfileAlongPath.
DomainError	Non-positive scale on input in SurfaceProfileAlongPath.

SurfaceSinglePointsAlongAxis

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Returns single array of surface points along X or Y axis.

Syntax

```
void avl::SurfaceSinglePointsAlongAxis
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Surface&> inSurface2,
 avl::Axis::Type inAxis,
 double inCoordinateValue,
 int inSmoothRadius,
 atl::Optional<double> inMinOutputCoordinate,
 atl::Optional<double> inMaxOutputCoordinate,
 atl::Optional<int> inMaxInterpolationLength,
 atl::Array<avl::Point3D>& outPoints,
 atl::Optional<double&> outCoordinateValue = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inSurface2	Optional<const Surface&>		NIL	Optional second input surface
→	inAxis	Axis::Type			Axis along which the points are extracted
→	inCoordinateValue	double			Determines the coordinate the points will be extracted from
→	inSmoothRadius	int	0 - ∞		Increases the number of neighbouring points taken into account
→	inMinOutputCoordinate	Optional<double>		NIL	Minimal second coordinate of the output points
→	inMaxOutputCoordinate	Optional<double>		NIL	Maximal second coordinate of the output points
→	inMaxInterpolationLength	Optional<int>	0 - ∞	0	Maximal number of consecutive not existing points to be interpolated
←	outPoints	Array<Point3D>&			The resulting surface points
←	outCoordinateValue	Optional<double&>		NIL	The coordinate the output points were extracted from

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outCoordinateValue**.

Read more about Optional Outputs.

SurfaceSingleProfileAlongAxis

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Creates the profile of point Z values along X or Y axis.

Syntax

```
void avl::SurfaceSingleProfileAlongAxis
(
 const avl::Surface& inSurface,
 avl::Axis::Type inAxis,
 double inCoordinateValue,
 int inSmoothRadius,
 atl::Optional<double> inProfileDomainStart,
 atl::Optional<double> inProfileDomainEnd,
 atl::Optional<int> inMaxInterpolationLength,
 float inDefaultValue,
 avl::Profile& outProfile,
 atl::Optional<double&> outCoordinateValue = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inAxis	Axis::Type			Axis along which the profile is extracted
→	inCoordinateValue	double			Determines the coordinate the profile will be extracted from
→	inSmoothRadius	int	0 - ∞		Increases the number of neighbouring profiles taken into account extracting a profile
→	inProfileDomainStart	Optional<double>		NIL	Minimal X coordinate of the output profile
→	inProfileDomainEnd	Optional<double>		NIL	Maximal X coordinate of the output profile
→	inMaxInterpolationLength	Optional<int>	0 - ∞	NIL	Maximal number of consecutive not existing profile points to be interpolated
→	inDefaultValue	float			Default value of the not existing and not interpolated surface point
←	outProfile	Profile&			The resulting profile of the surface height
←	outCoordinateValue	Optional<double&>		NIL	The coordinate the output profile was extracted from

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outCoordinateValue**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect output profile domain in SurfaceSingleProfileAlongAxis.
DomainError	Input coordinate value exceeds surface in SurfaceSingleProfileAlongAxis.
DomainError	Unknown axis type in SurfaceSingleProfileAlongAxis.

SurfaceToPlaneDistanceImage

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Computes the image of the distances of the input surface points from a given plane.

Syntax

```
void avl::SurfaceToPlaneDistanceImage
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 const avl::Plane3D& inPlane,
 float inResolution,
 bool inSignedDistance,
 avl::Image& outDistanceImage
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			Input surface
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inPlane	const Plane3D&			Input plane
→	inResolution	float	0.0 - ∞	1.0f	Number of real-world units per one pixel
→	inSignedDistance	bool		False	Flag indicating whether to compute signed distance to the plane
←	outDistanceImage	Image&			

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite plane on input in SurfaceToPlaneDistanceImage.
DomainError	Region of interest exceeds an input surface in SurfaceToPlaneDistanceImage.

SurfaceValidPointsRegion

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Computes region of locations where points are valid in a surface and where they are invalid.

Syntax

```
void avl::SurfaceValidPointsRegion
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 avl::Region& outValidPointsRegion,
 avl::Region& outInvalidPointsRegion
)
```

Parameters

	Name	Type	Default	Description
→	inSurface	const Surface&		Input surface
→	inRoi	Optional<const Region&>	NIL	Region of interest
←	outValidPointsRegion	Region&		Region of locations where points are valid
←	outInvalidPointsRegion	Region&		Region of locations where points are invalid

Errors

List of possible exceptions:

Error type	Description
DomainError	Region of interest exceeds an input surface in SurfaceValidPointsRegion.

SurfaceVolume_Double

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Computes the volume of a surface with respect to another surface.

Syntax

```
void avl::SurfaceVolume_Double
(
 const avl::Surface& inTopSurface,
 const avl::Surface& inBottomSurface,
 atl::Optional<const avl::Region&> inRoi,
 avl::VolumeCalculationMethod::Type inVolumeCalculationMethod,
 double& outVolume
)
```

Parameters

	Name	Type	Default	Description
→	inTopSurface	const Surface&		The top surface
→	inBottomSurface	const Surface&		The bottom surface
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inVolumeCalculationMethod	VolumeCalculationMethod::Type		Determines how the parts of a volume are added up
←	outVolume	double&		Volume of the input surface

Errors

List of possible exceptions:

Error type	Description
DomainError	Different surface dimensions between inTopSurface and inBottomSurface in SurfaceVolume_Double.
DomainError	Different surface scales or offsets between inTopSurface and inBottomSurface in SurfaceVolume_Double.
DomainError	Different surface types between inTopSurface and inBottomSurface in SurfaceVolume_Double.
DomainError	Region of interest exceeds an input surface in SurfaceVolume_Double.
DomainError	Unsupported inVolumeCalculationMethod in SurfaceVolume_Double.

SurfaceVolume_Single

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Computes the volume of a surface with respect to a plane.

Syntax

```
void avl::SurfaceVolume_Single
(
 const avl::Surface& inSurface,
 const double inZ,
 atl::Optional<const avl::Region&> inRoi,
 avl::VolumeCalculationMethod::Type inVolumeCalculationMethod,
 double& outVolume,
 atl::Optional<avl::Plane3D&> outPlane = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inSurface	const Surface&		Input surface
→	inZ	const double		The volume is calculated with respect to the plane Z = inZ
→	inRoi	Optional<const Region&>	NIL	Range of pixels to be processed
→	inVolumeCalculationMethod	VolumeCalculationMethod::Type		Determines how the parts of a volume are added up
←	outVolume	double&		Volume of the input surface
←	outPlane	Optional<Plane3D&>	NIL	The plane defined as Z = inZ

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outPlane**.

Read more about Optional Outputs.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region of interest exceeds an input surface in SurfaceVolume_Single.
DomainError	Unsupported inVolumeCalculationMethod in SurfaceVolume_Single.

126. Surface Interpolations

Table of content:

ResampleSurface

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Equalizes the scales format of the input surface.

Syntax

```
void avl::ResampleSurface
(
 const avl::Surface& inSurface,
 avl::ResampleSurfaceMode::Type inResampleSurfaceMode,
 atl::Optional<double> inXOffset,
 atl::Optional<double> inYOffset,
 avl::Surface& outSurface
)
```

Parameters

	Name	Type	Default	Description
→	inSurface	const Surface&		
→	inResampleSurfaceMode	ResampleSurfaceMode::Type		
→	inXOffset	Optional<double>	NIL	Offset for the X axis of the output surface
→	inYOffset	Optional<double>	NIL	Offset for the Y axis of the output surface
←	outSurface	Surface&		

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect resample mode in ResampleSurface.

ResampleSurface_AnyScales

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Changes the coordinates format of the input surface.

Syntax

```
void avl::ResampleSurface_AnyScales
(
 const avl::Surface& inSurface,
 atl::Optional<double> inXOffset,
 atl::Optional<double> inXScale,
 atl::Optional<double> inYOffset,
 atl::Optional<double> inYScale,
 avl::Surface& outSurface
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			
→	inXOffset	Optional<double>		NIL	Offset for the X axis of the output surface
→	inXScale	Optional<double>	0.000001 - ∞	NIL	Scale for the X axis of the output surface
→	inYOffset	Optional<double>		NIL	Offset for the Y axis of the output surface
→	inYScale	Optional<double>	0.000001 - ∞	NIL	Scale for the Y axis of the output surface
←	outSurface	Surface&			

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Non-positive scale on input in ResampleSurface_AnyScales.

SmoothSurface_Gauss

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Smooths a surface using a gaussian kernel.

Applications: Removal of gaussian noise from surfaces.

Syntax

```
void avl::SmoothSurface_Gauss
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 float inStdDevX,
 atl::Optional<float> inStdDevY,
 const float inKernelRelativeSize,
 avl::Surface& outSurface,
 int& diagKernelRadiusX,
 int& diagKernelRadiusY
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			
→	inRoi	Optional<const Region&>		NIL	Range of output points to be computed
→	inStdDevX	float	0.0 - ∞	1.0f	Horizontal smoothing standard deviation
→	inStdDevY	Optional<float>	0.0 - ∞	NIL	Vertical smoothing standard deviation
→	inKernelRelativeSize	const float	0.0 - 3.0	2.0f	A multiple of the standard deviation determining the size of the kernel
←	outSurface	Surface&			
🔍	diagKernelRadiusX	int&			Horizontal radius of Gaussian kernel being used
🔍	diagKernelRadiusY	int&			Vertical radius of Gaussian kernel being used

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region of interest exceeds a surface in SmoothSurface_Gauss.

SmoothSurface_Mean

Header: AVL.h
Namespace: avl
Module: Vision3DStandard

Smooths a surface by averaging heights within a rectangular kernel.

Applications: Usually used for computing features related to local surface "windows". Can be also used for noise removal, but Gauss is superior here.

Syntax

```
void avl::SmoothSurface_Mean
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 avl::KernelShape::Type inKernel,
 int inRadiusX,
 atl::Optional<int> inRadiusY,
 avl::Surface& outSurface
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			
→	inRoi	Optional<const Region&>		NIL	Range of points to be computed
→	inKernel	KernelShape::Type			Kernel shape
→	inRadiusX	int	0 - ∞	1	Nearly half of the kernel's width (2*R+1)
→	inRadiusY	Optional<int>	0 - ∞	NIL	Nearly half of the kernel's height (2*R+1), or same as inRadiusX
←	outSurface	Surface&			

Hardware Acceleration

This operation supports automatic parallelization for multicore and multiprocessor systems.

Errors

List of possible exceptions:

Error type	Description
DomainError	Region of interest exceeds an input surface in SmoothSurface_Mean.

127. Segmentation 3D

Table of content:

SegmentPoint3DGrid_PlanarCells

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes a list of planar cells of a point cloud

Syntax

```
void avl::SegmentPoint3DGrid_PlanarCells
(
 const avl::Point3DGrid& inPointCloud,
 const float inCellSize,
 const double inFlatnessNoiseLevel,
 atl::Array<atl::Array<avl::Point3D>>& outPlanarCells,
 atl::Array<avl::Segment3D>& diagPlanarCellNormals,
 atl::Array<avl::Box3D>& diagPlanarCellBoxes
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPointCloud	const Point3DGrid&			
→	inCellSize	const float	2.0 - ∞	16.0f	Size of a single cell in the real coordinate system
→	inFlatnessNoiseLevel	const double	0.0 - 1.0	0.04D	
←	outPlanarCells	Array<Array<Point3D>>&			
🔍	diagPlanarCellNormals	Array<Segment3D>&			Segments representing the normals of computed cells
🔍	diagPlanarCellBoxes	Array<Box3D>&			Bounding boxes of planar cells

SegmentPoint3DGrid_Planes

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Segments a point cloud into planes.

Syntax

```
void avl::SegmentPoint3DGrid_Planes
(
 const avl::Point3DGrid& inPointCloud,
 const float inCellSize,
 const double inFlatnessNoiseLevel,
 int inMaxIterations,
 const float inMaxAngleDeviation,
 const float inMaxDistanceDeviation,
 atl::Array<atl::Array<avl::Point3D>>& outPlanePoints,
 atl::Array<avl::Plane3D>& outPlanes,
 atl::Array<atl::Array<avl::Point3D>>& diagPlanarCells
)
```

Parameters

	Name	Type	Range	Default	Description
→	inPointCloud	const Point3DGrid&			
→	inCellSize	const float	2.0 - ∞	16.0f	Size of a single cell in the real coordinate system
→	inFlatnessNoiseLevel	const double	0.0 - 1.0	0.04D	
→	inMaxIterations	int	1 - ∞	1000	
→	inMaxAngleDeviation	const float	0.0 - 90.0	15.0f	
→	inMaxDistanceDeviation	const float	0.0 - ∞	0.1f	
←	outPlanePoints	Array<Array<Point3D>>&			
←	outPlanes	Array<Plane3D>&			
🔍	diagPlanarCells	Array<Array<Point3D>>&			Computed planar cells

SegmentSurface_PlanarCells

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Computes a list of planar cells of a surface

Syntax

```
void avl::SegmentSurface_PlanarCells
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 const int inCellWidth,
 atl::Optional<int> inCellHeight,
 const double inFlatnessNoiseLevel,
 const float inMinCellCoverage,
 atl::Array<avl::Region>& outPlanarCells,
 atl::Array<avl::Segment3D>& diagPlanarCellNormals,
 atl::Array<avl::Region>& diagCellRegions
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inCellWidth	const int	2 - ∞	16	Width of a single cell in the girds coordinate system
→	inCellHeight	Optional<int>	2 - ∞	NIL	Height of a single cell in the girds coordinate system
→	inFlatnessNoiseLevel	const double	0.0 - 1.0	0.04D	
→	inMinCellCoverage	const float	0.0 - 1.0	0.3f	Minimum percentage of valid points that a planar cell has to have
←	outPlanarCells	Array<Region>&			
🔍	diagPlanarCellNormals	Array<Segment3D>&			Segments representing the normals of computed cells,
🔍	diagCellRegions	Array<Region>&			Regions of all both planar and non-planar cells

SegmentSurface_Planes

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Segments a surface into planes.

Syntax

```
void avl::SegmentSurface_Planes
(
 const avl::Surface& inSurface,
 atl::Optional<const avl::Region&> inRoi,
 const int inCellWidth,
 atl::Optional<int> inCellHeight,
 const double inFlatnessNoiseLevel,
 int inMaxIterations,
 const double inMaxAngleDeviation,
 const double inMaxDistanceDeviation,
 const float inMinCellCoverage,
 const int inMinPlaneArea,
 atl::Array<avl::Region>& outPlaneRegions,
 atl::Array<avl::Plane3D>& outPlanes,
 atl::Array<avl::Region>& diagPlanarCells
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSurface	const Surface&			
→	inRoi	Optional<const Region&>		NIL	Range of pixels to be processed
→	inCellWidth	const int	2 - ∞	16	Width of a single cell in the girds coordinate system
→	inCellHeight	Optional<int>	2 - ∞	NIL	Height of a single cell in the girds coordinate system
→	inFlatnessNoiseLevel	const double	0.0 - 1.0	0.04D	
→	inMaxIterations	int	1 - ∞	1000	
→	inMaxAngleDeviation	const double	0.0 - 90.0	15.0D	
→	inMaxDistanceDeviation	const double	0.0 - ∞	0.1D	
→	inMinCellCoverage	const float	0.0 - 1.0	0.3f	Minimum percentage of valid points that a planar cell has to have
→	inMinPlaneArea	const int	1 - + ∞	1	Minimum number of pixels for a single plane
←	outPlaneRegions	Array<Region>&			
←	outPlanes	Array<Plane3D>&			
🔍	diagPlanarCells	Array<Region>&			Computed planar cells

128. Histogram Local Transforms

Table of content:

SmoothHistogram_Gauss

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Smooths a histogram by averaging points within a kernel using gaussian-weighted average.

Syntax

```
void avl::SmoothHistogram_Gauss
(
 const avl::Histogram& inHistogram,
 atl::Optional<const avl::Range&> inRange,
 const float inStdDev,
 const float inKernelRelativeSize,
 const bool inCyclic,
 avl::Histogram& outHistogram
)
```

Parameters

	Name	Type	Range	Default	Description
→	inHistogram	const Histogram&			Input histogram
→	inRange	Optional<const Range&>		NIL	
→	inStdDev	const float	0.0 - ∞	0.6f	Smoothing standard deviation
→	inKernelRelativeSize	const float	0.0 - ∞	3.0f	
→	inCyclic	const bool		False	
←	outHistogram	Histogram&			Output histogram

SmoothHistogram_Mean

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Smooths a histogram by averaging points within a kernel.

Syntax

```
void avl::SmoothHistogram_Mean
(
 const avl::Histogram& inHistogram,
 atl::Optional<const avl::Range&> inRange,
 const int inKernelRadius,
 const bool inCyclic,
 avl::Histogram& outHistogram
)
```

Parameters

	Name	Type	Range	Default	Description
→	inHistogram	const Histogram&			Input histogram
→	inRange	Optional<const Range&>		NIL	
→	inKernelRadius	const int	0 - ∞	3	Defines the width of the kernel as 2*R+1
→	inCyclic	const bool		False	
←	outHistogram	Histogram&			Output histogram

129. Path Local Transforms

Table of content:

SmoothPath_Gauss

Header: AVL.h
Namespace: avl
Module: FoundationPro

Smooths a path by averaging its characteristic points within a kernel using gaussian-weighted average.

Syntax

```
void avl::SmoothPath_Gauss
(
 const avl::Path& inPath,
 const float inStdDev,
 const float inKernelRelativeSize,
 avl::Path& outPath
)
```

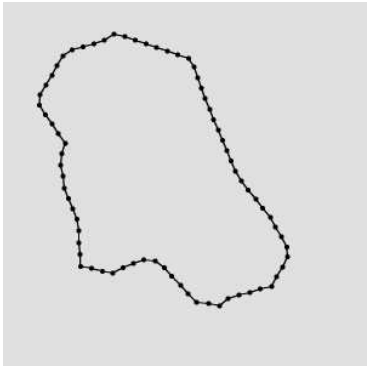
Parameters

	Name	Type	Range	Default	Description
→	inPath	const Path&			Input path
→	inStdDev	const float	0.0 - ∞	0.6f	Smoothing standard deviation
→	inKernelRelativeSize	const float	0.0 - ∞	3.0f	A multiple of the standard deviation determining the size of the kernel
←	outPath	Path&			Output path

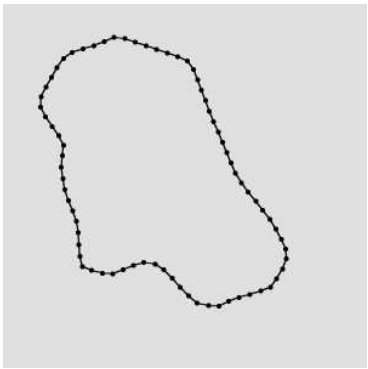
Description

The operation replaces each characteristic point of the path with the local gaussian average, thus smoothing its shape. The local average is computed as a gaussian mean of the consecutive **inStdDev*inKernelRelativeSize** characteristic points.

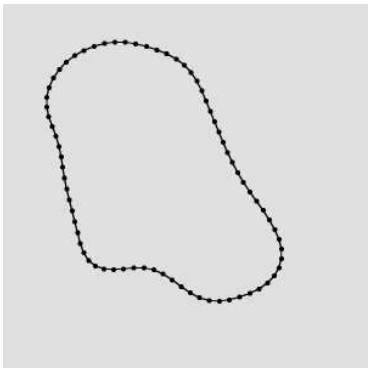
Examples



A sample path



SmoothPath_Gauss *run on the sample path with* ***inStdDev*** *= 0.6 and* ***inKernelRelativeSize*** *= 3.*



SmoothPath_Gauss *run on the sample path with* ***inStdDev*** *= 2 and* ***inKernelRelativeSize*** *= 3.*

See Also

- SmoothPath_Mean – Smooths a path by averaging its characteristic points within a kernel.

SmoothPath_Mean

Header: AVL.h
Namespace: avl
Module: FoundationPro

Smooths a path by averaging its characteristic points within a kernel.

Syntax

```
void avl::SmoothPath_Mean
(
 const avl::Path& inPath,
 int inKernelRadius,
 avl::Path& outPath
)
```

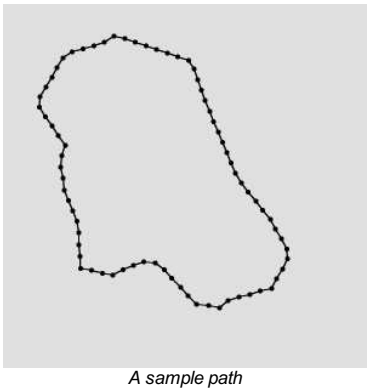
Parameters

	Name	Type	Range	Default	Description
➡	inPath	const Path&			Input path
➡	inKernelRadius	int	0 - ∞	3	
⬅	outPath	Path&			Output path

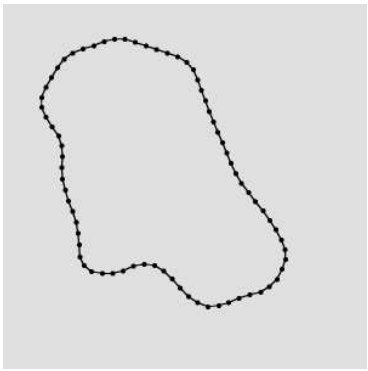
Description

The operation replaces each characteristic point of the path with the local average, thus smoothing its shape. The local average is computed as a simple arithmetic mean of the consecutive 2***inKernelRadius** + 1 characteristic points.

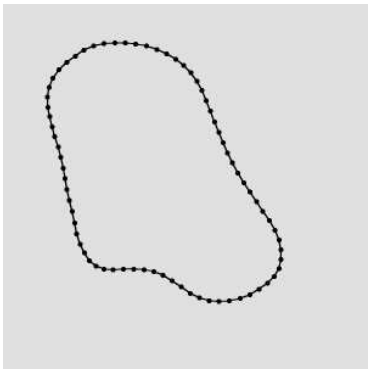
Examples



A sample path



***SmoothPath_Mean** run on the sample path with **inKernelRadius** = 1.*



***SmoothPath_Mean** run on the sample path with **inKernelRadius** = 3.*

Errors

List of possible exceptions:

Error type	Description
DomainError	Kernel size exceeds the number of points in a closed path in SmoothPath_Mean.

See Also

- SmoothPath_Gauss – Smooths a path by averaging its characteristic points within a kernel using gaussian-weighted average.

130. Support Vector Machines

Table of content:

SVM_ClassifyMultiple

Header: AVL.h
Namespace: avl
Module: FoundationPro

Classifies input points based on trained model.

Syntax

```
void avl::SVM_ClassifyMultiple
(
 const avl::SvmModel& inSvmModel,
 const atl::Array<atl::Array<float> >& inVectorArray,
 atl::Array< int >& outPredictions,
 atl::Optional<atl::Array< int >& > outModelClasses = atl::NIL,
 atl::Optional<atl::Array<atl::Array<float>>& > outClassProbabilities = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inSvmModel	const SvmModel&		Input trained model
→	inVectorArray	const Array<Array<float> >&		Data vector array of unknown classes
←	outPredictions	Array< int >&		Predicted classes
←	outModelClasses	Optional<Array< int >& >	NIL	All known model classes in order
←	outClassProbabilities	Optional<Array<Array<float>>& >	NIL	For each data vector the probability of belonging to each class

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outModelClasses**, **outClassProbabilities**.

Read more about Optional Outputs.

Description

The operation predicts classes for the given data points. It takes an array of data vectors (**inVectorArray**) as an argument. Each vector has to be of the same size as vectors used for training the model. The operation outputs predicted class (**outPredictions**) for each data vector.
outModelClasses are all class labels encountered during training.
outClassProbabilities provides, for each vector, estimated probability of this vector belonging to each class. Precisely, in each array the value under index **i** denotes probability of given vector belonging to the class **outModelClasses[i]**.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty vector array in SVM_ClassifyMultiple.
DomainError	Incorrect vector size in SVM_Classify
DomainError	Incorrect, uninitialized or not trained SvmModel in SVM_ClassifyMultiple.

See Also

- SVM_Init – Initializes an SVM model.
- SVM_Train – Trains an SVM model.
- SVM_ClassifySingle – Classifies input features based on a trained model.

SVM_ClassifySingle

Header: AVL.h
Namespace: avl
Module: FoundationPro

Classifies input features based on a trained model.

Syntax

```
void avl::SVM_ClassifySingle
(
 const avl::SvmModel& inSvmModel,
 const atl::Array<float>& inVector,
 int& outPrediction,
 atl::Optional<atl::Array< int >& > outModelClasses = atl::NIL,
 atl::Optional<atl::Array<float>& > outClassProbabilities = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inSvmModel	const SvmModel&		Input trained model
→	inVector	const Array<float>&		Data vector of unknown class
←	outPrediction	int&		Predicted classes
←	outModelClasses	Optional<Array< int >& >	NIL	All known model classes in order
←	outClassProbabilities	Optional<Array<float>& >	NIL	For each data vector the probability of belonging to each class

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outModelClasses**, **outClassProbabilities**.

Read more about Optional Outputs.

Description

The operation predicts classes for the given data points. It takes a data vector (**inVector**) as an argument. The vector has to be of the same size as vectors used for training the model. The operation outputs predicted class (**outPrediction**) for the data vector.
outModelClasses are all class labels encountered during training.
outClassProbabilities provides, for the input vector, estimated probability belonging to each class. Precisely, the value under index **i** denotes probability of given vector belonging to the class **outModelClasses[i]**.

Errors

List of possible exceptions:

Error type	Description
DomainError	Incorrect vector size in SVM_ClassifySingle
DomainError	Incorrect, uninitialized or not trained SvmModel in SVM_ClassifySingle.

See Also

- SVM_Init – Initializes an SVM model.
- SVM_Train – Trains an SVM model.
- SVM_ClassifyMultiple – Classifies input points based on trained model.

SVM_Init

Header: AVL.h
Namespace: avl
Module: FoundationPro

Initializes an SVM model.

Syntax

```
void avl::SVM_Init
(
 atl::Optional<float> inKernelGamma,
 float inKernelGammaScale,
 float inRegularizationConstant,
 atl::Optional<float> inNu,
 float inStoppingEpsilon,
 bool inUseShrinkingHeuristics,
 avl::SvmModel& outSvmModel
)
```

Parameters

	Name	Type	Range	Default	Description
→	inKernelGamma	Optional<float>	0.0001 - ∞	0.0001f	Gamma parameter for Rbf kernel
→	inKernelGammaScale	float	0.0001 - ∞	1.0f	Gamma parameter scale
→	inRegularizationConstant	float	0.0001 - ∞	1.0f	Preventing overfitting
→	inNu	Optional<float>	0.0001 - 1.0	NIL	Tradeoff between training accuracy and number of SV
→	inStoppingEpsilon	float	0.0001 - ∞	0.001f	Epsilon for stopping criterium
→	inUseShrinkingHeuristics	bool		True	May speed up computations
←	outSvmModel	SvmModel&			Output model

Description

The operation initializes a model for an SVM classifier that will be used by SVM_Train and SVM_ClassifySingle.

Support Vector Machines (SVM or C-SVC) is a classifier based on the support vector idea. Those vectors define hypersurfaces that separate data points from two different classes. The shape of those hypersurfaces is defined by the scalar product dependent on the kernel type. In the simplest case of linear kernel, they define hyperplanes.

Notice that SVM uses one-versus-all evaluation, because it is a binary classifier.

Nu-SVC is a variant of C-SVC making use of Nu parameter. It provides a tradeoff between the number of support vectors and the number of training errors. In fact, it defines a lower bound on the former and an upper bound on the latter.

There are two types of kernels:

- **Linear** is a simple kernel for easier tasks. It is also recommended when the training time is critical.
  The corresponding scalar product is $L(u,v) = u^T v$. This kernel is used when **inKernelGamma** is equal to Auto.
- **Rbf** (radial basis kernel) is the most common and the recommended kernel. Its function is $\mathrm{Rbf}_\gamma(u,v) = \exp(-\gamma\|u-v\|^2)$.

SVM_Init parameters are:

- **inKernelGamma** is the γ (gamma) parameter for Rbf kernel. When it is set to Auto, then **Linear** kernel is used.
- **inKernelGammaScale** is used to provide more precise setting the value of **inKernelGamma**.
  As the result kernel gamma value is set to $\gamma = \frac{inKernelGamma}{inKernelGammaScale}$
- **inRegularizationConstant** is the C constant of C-SVC and Nu-SVC. With greater C, the model is less likely to overfit training data.
- **inStoppingEpsilon** defines precision of the training. The smaller this parameter is, the more accurate the training is, but also the longer it takes.
- **inNu** is the Nu parameter of Nu-SVC described above. If it is Auto, then simple C-SVC classifier is used.
- If **inUseShrinkingHeuristics** is set, the algorithm will use the heuristics that shrinks the search region to speed up computation with little or none loss of accuracy.

Getting started:

At the beginning a good choice for SVM type is C-SVC (i.e. to leave **inNu** equal NIL) and the recommended kernel type is **Rbf**. The parameter **inStoppingEpsilon** may be left default (equal 0.001) and it is recommended to set **inUseShrinkingHeuristics** to true. There are two parameters to choose: **inKernelGamma** and **inRegularizationConstant**. The first one is responsible for "tightness" of the class regions and the second one for ignoring detached data points.

The best way to choose those parameters is to perform a *grid search*, i.e. to try different pairs of values, for example pairs of powers of 2. The grid search may be performed on a relatively small subset of training data to reduce training time. Moreover, it is recommended to use *cross-validation*, which means evaluating parameters not on the training data, but on a different data set, to prevent overfitting. One has to remember to choose those subsets at random.

Another thing worth trying is normalizing data before passing it to training and classification procedures. However, data has to be rescaled with the same value for both training and classification data! Normalizing data might improve performance and is useful for optimal parameter choice. If the data vectors are rescaled N times, then gamma in RBF kernel has to be rescaled by term N^{-2} to ensure similar performance.

Remarks

It is recommended there are at least two training samples for each data class in the training set.

See Also

- SVM_ClassifySingle – Classifies input features based on a trained model.
- SVM_Train – Trains an SVM model.

SVM_Train

Header: AVL.h
Namespace: avl
Module: FoundationPro

Trains an SVM model.

Syntax

```
void avl::SVM_Train
(
 const avl::SvmModel& inSvmModel,
 const atl::Array<atl::Array<float> >& inVectorArray,
 const atl::Array< int >& inAnswerArray,
 avl::SvmModel& outSvmModel,
 float& outTrainingAccuracy
)
```

Parameters

	Name	Type	Default	Description
→	inSvmModel	const SvmModel&		Initialized SVM model
→	inVectorArray	const Array<Array<float> >&		Training data vector array
→	inAnswerArray	const Array< int >&		Correct classes for data vectors
←	outSvmModel	SvmModel&		Trained model
←	outTrainingAccuracy	float&		Accuracy of prediction on training set

Description

The operation trains an SVM classifier initialized beforehand by SVM_Init function. It takes two arrays as arguments:

- **inVectorArray**, an array of data points with known classes
- **inAnswerArray**, an array of classes where the corresponding data points belong

Those two arrays have to be of the same size. Moreover, there have to be at least two classes within the training data set.

The output **outSvmModel** is an SVM_Model that may be used by SVM_ClassifySingle function.

outTrainingAccuracy is the fraction of correctly classified training data points.

Errors

List of possible exceptions:

Error type	Description
DomainError	Data vector cannot be empty in SVM_Train.
DomainError	Incompatible array sizes in SVM_Train
DomainError	Incompatible vector sizes in SVM_Train.
DomainError	Incorrect or uninitialized SvmModel in Svm_Train.
DomainError	SM model is already trained in SVM_Train.

See Also

- SVM_Init – Initializes an SVM model.
- SVM_ClassifySingle – Classifies input features based on a trained model.

131. Image Relations

Table of content:

TestBoxInImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Tests whether a box is contained in the dimensions of an image.

Syntax

```
void avl::TestBoxInImage
(
 const avl::Image& inImage,
 const avl::Box& inBox,
 bool& outIsContained
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inBox	const Box&		
←	outIsContained	bool&		

Description

Tests whether a box is contained in the dimensions of an image.

Examples

TestBoxInImage Output	Preview
True	
False	

TestImageEqualTo

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Tests whether two images equal.

Syntax

```
void avl::TestImageEqualTo
(
 const avl::Image& inImage,
 const avl::Image& inReferenceImage,
 bool& outIsEqual
)
```

Parameters

	Name	Type	Default	Description
➡	inImage	const Image&		Input image
➡	inReferenceImage	const Image&		
⬅	outIsEqual	bool&		

TestLocationInBox

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Tests whether a location is contained in the dimensions of a box.

Syntax

```
void avl::TestLocationInBox
(
 const avl::Location& inLocation,
 const avl::Box& inBox,
 bool& outIsContained
)
```

Parameters

	Name	Type	Default	Description
➡	inLocation	const Location&		
➡	inBox	const Box&		
⬅	outIsContained	bool&		

TestLocationInImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Tests whether a location is contained in the dimensions of an image.

Syntax

```
void avl::TestLocationInImage
(
 const avl::Image& inImage,
 const avl::Location& inLocation,
 bool& outIsContained
)
```

Parameters

	Name	Type	Default	Description
➡	inImage	const Image&		Input image
➡	inLocation	const Location&		
⬅	outIsContained	bool&		

TestPointInImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Tests whether a point is contained in the dimensions of an image.

Syntax

```
void avl::TestPointInImage
(
 const avl::Image& inImage,
 const avl::Point2D& inPoint,
 bool& outIsContained
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inPoint	const Point2D&		
←	outIsContained	bool&		

TestRectangleInImage

Also in AVL Lite

Header: AVL.h
Namespace: avl
Module: FoundationLite

Tests whether a rectangle is contained in the dimensions of an image.

Syntax

```
void avl::TestRectangleInImage
(
 const avl::Image& inImage,
 const avl::Rectangle2D& inRectangle,
 bool& outIsContained
)
```

Parameters

	Name	Type	Default	Description
→	inImage	const Image&		Input image
→	inRectangle	const Rectangle2D&		
←	outIsContained	bool&		

132. Histogram Relations

Table of content:

TestHistogramDominatesHistogram

Header: AVL.h
Namespace: avl
Module: FoundationBasic

Checks whether the values in the first histogram are greater or equal than corresponding values in the second histogram.

Syntax

```
void avl::TestHistogramDominatesHistogram
(
 const avl::Histogram& inHistogram1,
 const avl::Histogram& inHistogram2,
 bool& outDominates
)
```

Parameters

	Name	Type	Default	Description
→	inHistogram1	const Histogram&		Input histogram1
→	inHistogram2	const Histogram&		Input histogram2
←	outDominates	bool&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Input histograms formats are not the same in TestHistogramDominatesHistogram.

133. Geometry 3D Relations

Table of content:

TestLine3DThroughCircle3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Tests whether a line in 3D goes through a circle in 3D.

Syntax

```
void avl::TestLine3DThroughCircle3D
(
 const avl::Line3D& inLine3D,
 const avl::Circle3D& inCircle3D,
 bool& outGoesThrough
)
```

Parameters

	Name	Type	Default	Description
➡	inLine3D	const Line3D&		
➡	inCircle3D	const Circle3D&		
⬅	outGoesThrough	bool&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Indefinite circle on input in TestLine3DThroughCircle3D.
DomainError	Indefinite line on input in TestLine3DThroughCircle3D.

TestPoint3DInBox3D

Header: AVL.h
Namespace: avl
Module: Vision3DLite

Tests whether a point in 3D lies in a box in 3D.

Syntax

```
void avl::TestPoint3DInBox3D
(
 const avl::Point3D& inPoint3D,
 const avl::Box3D& inBox3D,
 bool& outIsContained
)
```

Parameters

	Name	Type	Default	Description
➡	inPoint3D	const Point3D&		
➡	inBox3D	const Box3D&		
⬅	outIsContained	bool&		

134. Profile Relations

Table of content:

TestProfileDominatesProfile

Header: AVL.h
Namespace: avl
Module: FoundationPro

Checks whether values in the first profile are greater or equal than corresponding values in the second profile.

Syntax

```
void avl::TestProfileDominatesProfile
(
 const avl::Profile& inProfile1,
 const avl::Profile& inProfile2,
 bool& outDominates
)
```

Parameters

	Name	Type	Default	Description
→	inProfile1	const Profile&		Input profile1
→	inProfile2	const Profile&		Input profile2
←	outDominates	bool&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Input profiles have different lengths in TestProfileDominatesProfile.
DomainError	Input profiles have different X coordinates in TestProfileDominatesProfile.

TestProfileEqualTo

Header: AVL.h
Namespace: avl
Module: FoundationPro

Checks whether profiles are exactly equals.

Syntax

```
void avl::TestProfileEqualTo
(
 const avl::Profile& inProfile,
 const avl::Profile& inReferenceProfile,
 bool& outIsEqual
)
```

Parameters

	Name	Type	Default	Description
→	inProfile	const Profile&		Input profile
→	inReferenceProfile	const Profile&		
←	outIsEqual	bool&		

135. Loop Generators

Table of content:

EnumerateCombinations

Header: STD.h
Namespace: avl
Module: FoundationLite

In each consecutive iteration produces a consecutive combination of integer numbers.

Syntax

```
bool avl::EnumerateCombinations
(
 EnumerateCombinationsState& ioState,
 int inCombinationSize,
 int inSetSize,
 bool inRepeat,
 atl::Array<int>& outCombination,
 int& outIndex
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	EnumerateCombinationsState&			Object used to maintain state of the function.
	inCombinationSize	int	0 - ∞	2	
	inSetSize	int	0 - ∞	3	
	inRepeat	bool			Determines whether to repeat loop
	outCombination	Array<int>&			
	outIndex	int&			

Errors

List of possible exceptions:

Error type	Description
DomainError	Combination size cannot be greater than the set size in EnumerateCombinations.

EnumerateIntegerPairs

Header: STD.h
Namespace: avl
Module: FoundationLite

Produces a total number of 'inCount1 * inCount2' of pairs of integer numbers.

Syntax

```
bool avl::EnumerateIntegerPairs
(
 Enumerate2DState& ioState,
 int inStart1,
 atl::Optional<int> inCount1,
 int inStep1,
 int inStart2,
 int inCount2,
 int inStep2,
 bool inRepeat,
 int& outValue1,
 int& outValue2
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Enumerate2DState&			Object used to maintain state of the function.
	inStart1	int			First element of first range
	inCount1	Optional<int>	0 - ∞	NIL	Length of first range
	inStep1	int			Difference between consecutive elements of first range
	inStart2	int			First element of second range
	inCount2	int	0 - ∞		Length of second range
	inStep2	int			Difference between consecutive elements of second range
	inRepeat	bool			Determines whether to repeat loop
	outValue1	int&			Elements of first range in 'AAABBBCCC' order
	outValue2	int&			Elements of second range in 'ABCABCABC' order

EnumerateIntegers

Header: STD.h
Namespace: avl
Module: FoundationLite

In each consecutive iteration produces a consecutive number from an arithmetic sequence.

Syntax

```
bool avl::EnumerateIntegers
(
 Enumerate1DState& ioState,
 int inStart,
 atl::Optional<int> inCount,
 int inStep,
 bool inRepeat,
 int& outValue,
 atl::Optional<bool> outIsFirst = atl::NIL,
 atl::Optional<bool> outIsLast = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Enumerate1DState&			Object used to maintain state of the function.
	inStart	int			First value of the generated sequence
	inCount	Optional<int>	0 - ∞	NIL	Length of the generated sequence
	inStep	int			Value added in each iteration
	inRepeat	bool			Determines whether to repeat loop
	outValue	int&			
	outIsFirst	Optional<bool>		NIL	Flag indicating the first iteration
	outIsLast	Optional<bool>		NIL	Flag indicating the last iteration

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outIsFirst**, **outIsLast**.

Read more about Optional Outputs.

EnumerateRealPairs

Header: STD.h
Namespace: avl
Module: FoundationLite

Produces a total number of 'inCount1 * inCount2' of pairs of real numbers.

Syntax

```
bool avl::EnumerateRealPairs
(
 Enumerate2DState& ioState,
 float inStart1,
 atl::Optional<int> inCount1,
 float inStep1,
 float inStart2,
 int inCount2,
 float inStep2,
 bool inRepeat,
 float& outValue1,
 float& outValue2
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Enumerate2DState&			Object used to maintain state of the function.
	inStart1	float			First element of first range
	inCount1	Optional<int>	0 - ∞	NIL	Length of first range
	inStep1	float			Difference between consecutive elements of first range
	inStart2	float			First element of second range
	inCount2	int	0 - ∞		Length of second range
	inStep2	float			Difference between consecutive elements of second range
	inRepeat	bool			Determines whether to repeat loop
	outValue1	float&			Elements of first range in 'AAABBBCCC' order
	outValue2	float&			Elements of second range in 'ABCABCABC' order

EnumerateReals

Header: STD.h
Namespace: avl
Module: FoundationLite

In each consecutive iteration produces a consecutive number from an arithmetic sequence.

Syntax

```
bool avl::EnumerateReals
(
 Enumerate1DState& ioState,
 float inStart,
 atl::Optional<int> inCount,
 float inStep,
 bool inRepeat,
 float& outValue,
 atl::Optional<bool> outIsFirst = atl::NIL,
 atl::Optional<bool> outIsLast = atl::NIL
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Enumerate1DState&			Object used to maintain state of the function.
	inStart	float			First value of the generated sequence
	inCount	Optional<int>	0 - ∞	NIL	Length of the generated sequence
	inStep	float			Value added in each iteration
	inRepeat	bool			Determines whether to repeat loop
	outValue	float&			
	outIsFirst	Optional<bool>		NIL	Flag indicating the first iteration
	outIsLast	Optional<bool>		NIL	Flag indicating the last iteration

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outIsFirst**, **outIsLast**.

Read more about Optional Outputs.

136. Matrix

Table of content:

AddMatrices

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Adds two matrices element by element.

Syntax

```
void avl::AddMatrices
(
 const avl::Matrix& inMatrix1,
 const avl::Matrix& inMatrix2,
 avl::Matrix& outMatrix
)
```

Parameters

	Name	Type	Default	Description
→	inMatrix1	const Matrix&		
→	inMatrix2	const Matrix&		
←	outMatrix	Matrix&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Matrix dimensions do not match in AddMatrices.

CombineMatrices

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Combines two matrices horizontally.

Syntax

```
void avl::CombineMatrices
(
 const avl::Matrix& inMatrix1,
 const avl::Matrix& inMatrix2,
 avl::Matrix& outMatrix
)
```

Parameters

	Name	Type	Default	Description
→	inMatrix1	const Matrix&		
→	inMatrix2	const Matrix&		
←	outMatrix	Matrix&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Cannot combine matrices which do not have equal number of rows!

ConcatenateMatrices

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Combines two matrices horizontally.

Syntax

```
void avl::ConcatenateMatrices
(
 const avl::Matrix& inMatrix1,
 const avl::Matrix& inMatrix2,
 avl::Matrix& outMatrix
)
```

Parameters

	Name	Type	Default	Description
→	inMatrix1	const Matrix&		
→	inMatrix2	const Matrix&		
←	outMatrix	Matrix&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Matrix row counts do not match in ConcatenateMatrices.

CreateIdentityMatrix

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Creates a square matrix with ones on the main diagonal, and zeros elsewhere.

Syntax

```
void avl::CreateIdentityMatrix
(
 int inSize,
 avl::Matrix& outMatrix
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSize	int	1 - ∞		
←	outMatrix	Matrix&			

CreateUniformMatrix

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Creates a matrix with the specified dimensions and filled with a uniform element value.

Syntax

```
void avl::CreateUniformMatrix
(
 int inRowCount,
 int inColumnCount,
 float inValue,
 avl::Matrix& outMatrix
)
```

Parameters

	Name	Type	Range	Default	Description
→	inRowCount	int	0 - ∞		
→	inColumnCount	int	0 - ∞		
→	inValue	float			
←	outMatrix	Matrix&			

InvertMatrix

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Finds the inverse of a square matrix.

Syntax

```
void avl::InvertMatrix
(
 const avl::Matrix& inMatrix,
 avl::Matrix& outInverseMatrix
)
```

Parameters

	Name	Type	Default	Description
→	inMatrix	const Matrix&		
←	outInverseMatrix	Matrix&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty matrix in MatrixInverse
DomainError	Non-square matrix in MatrixInverse

LoadMatrix

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Loads a serialized Matrix object from an AVDATA file.

Syntax

```
void avl::LoadMatrix
(
 const atl::File& inFilename,
 avl::Matrix& outMatrix
)
```

Parameters

	Name	Type	Default	Description
→	inFilename	const File&		Name of the source file
←	outMatrix	Matrix&		Deserialized Matrix

MultiplyMatrices

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Multiplies two matrices element by element.

Syntax

```
void avl::MultiplyMatrices
(
 const avl::Matrix& inMatrix1,
 const avl::Matrix& inMatrix2,
 avl::Matrix& outMatrix
)
```

Parameters

	Name	Type	Default	Description
→	inMatrix1	const Matrix&		
→	inMatrix2	const Matrix&		
←	outMatrix	Matrix&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Matrix dimensions incompatible in MultiplyMatrices.

MultiplyMatrixByReal

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Multiplies all elements of a matrix by a value.

Syntax

```
void avl::MultiplyMatrixByReal
(
 const avl::Matrix& inMatrix,
 float inValue,
 avl::Matrix& outMatrix
)
```

Parameters

	Name	Type	Default	Description
→	inMatrix	const Matrix&		
→	inValue	float		
←	outMatrix	Matrix&		

SaveMatrix

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Saves a serialized Matrix object as AVDATA file.

Syntax

```
void avl::SaveMatrix
(
 const avl::Matrix& inMatrix,
 const atl::File& inFilename
)
```

Parameters

	Name	Type	Default	Description
→	inMatrix	const Matrix&		Matrix to be serialized
→	inFilename	const File&		Name of the target file

StackMatrices

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Combines two matrices vertically.

Syntax

```
void avl::StackMatrices
(
 const avl::Matrix& inMatrix1,
 const avl::Matrix& inMatrix2,
 avl::Matrix& outMatrix
)
```

Parameters

	Name	Type	Default	Description
→	inMatrix1	const Matrix&		
→	inMatrix2	const Matrix&		
←	outMatrix	Matrix&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Matrix column counts do not match in StackMatrices.

SubtractMatrices

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Subtracts two matrices element by element.

Syntax

```
void avl::SubtractMatrices
(
 const avl::Matrix& inMatrix1,
 const avl::Matrix& inMatrix2,
 avl::Matrix& outMatrix
)
```

Parameters

	Name	Type	Default	Description
→	inMatrix1	const Matrix&		
→	inMatrix2	const Matrix&		
←	outMatrix	Matrix&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Matrix dimensions do not match in SubtractMatrices.

TransposeMatrix

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Swaps rows with columns in a matrix.

Syntax

```
void avl::TransposeMatrix
(
 const avl::Matrix& inMatrix,
 avl::Matrix& outTranspose
)
```

Parameters

	Name	Type	Default	Description
→	inMatrix	const Matrix&		
←	outTranspose	Matrix&		

137. String

Table of content:

FormatDoubleToString

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Creates a string from a double number using a proper format.

Applications: Useful for preparing a number for display or communication with specific number of fractional digits, sign etc.

Syntax

```
void avl::FormatDoubleToString
(
 double inDouble,
 const int inIntegerDigitCount,
 const int inFractionalDigitCount,
 const atl::String& inDecimalMark,
 const atl::String& inTrailingCharacter,
 const bool inForceSignPrinting,
 const atl::String& inSuffix,
 atl::String& outString
)
```

Parameters

	Name	Type	Range	Default	Description
→	inDouble	double			Input real
→	inIntegerDigitCount	const int	0 - 1000		How many characters the integer part of the input real should have at least
→	inFractionalDigitCount	const int	0 - 100	3	How many characters the fractional part of the input real should have
→	inDecimalMark	const String&		\".\"	The symbol used to separate the integer part from the fractional part of the number
→	inTrailingCharacter	const String&		\"0\"	Defines the trailing character
→	inForceSignPrinting	const bool		False	Forces printing the sign of the number even if the number is positive
→	inSuffix	const String&		\"\"	Defines a suffix. Generally it is an unit of value (e.g. mm)
←	outString	String&			Output string

Examples

→	←
inDouble = 2.7182818284590452 **inIntegerDigitCount** = 2 **inFractionalDigitCount** = 2 **inDecimalMark** = "," **inTrailingCharacter** = "0" **inForceSignPrinting** = False **inSuffix** = "..."	**outString** = "02,72..."

In the first example desired integer digit count equals 2 as entered in **inIntegerDigitCount**, so the filter attaches character in **inTrailingCharacter** (zero) at the beginning of the result **outString**. Decimal mark is set as a comma and the number of fractional digits as 2. The result string ends with ellipsis defined in **inSuffix** and the result is "02,72...".
Note that the second digit was rounded up to "2" due to the succeeding digit.

→	←
inDouble = 77 **inIntegerDigitCount** = 1 **inFractionalDigitCount** = 1 **inDecimalMark** = "." **inTrailingCharacter** = "#" **inForceSignPrinting** = True **inSuffix** = "°"	**outString** = "+77.0°"

The second example demonstrates formatting with one integer digit and one fractional. Note that the integer part of the formatted number is less than the value, but it doesn't affect to this part (unlike fractional part which can cuts off excessing digits).
Trailing character (**#**) doesn't affect to anything, because desired integer digit count described by **inIntegerDigitCount** is one (the same effect would be for zero).

Remarks

If you want to cut off excessing digits from a fractional part of a number (instead of rounding them), you can format a number with one extra precision digit and then use Substring to cut off the last digit. You can add a suffix manually by using AvsFilter_ConcatenateStrings.

Errors

List of possible exceptions:

Error type	Description
DomainError	inDecimalMark has to be a single character in FormatDoubleToString.
DomainError	inTrailingCharacter has to be a single character in FormatDoubleToString.

See Also

- FormatRealToString – Creates a string from a real number using a proper format.
- FormatIntegerToString – Creates a string from an integer number using a proper format.
- FormatString – Creates a string according to the given format and data.

FormatIntegerToString

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Creates a string from an integer number using a proper format.

Applications: Useful for preparing a number for display or communication with specific number of digits, sign etc.

Syntax

```
void avl::FormatIntegerToString
(
 const int inInteger,
 const int inDigitCount,
 const atl::String& inTrailingCharacter,
 const bool inForceSignPrinting,
 const atl::String& inSuffix,
 const int inSystemBase,
 atl::String& outString
)
```

Parameters

	Name	Type	Range	Default	Description
→	inInteger	const int			Input integer
→	inDigitCount	const int	0 - 100		How many characters the output string should have at least
→	inTrailingCharacter	const String&		\"0\"	Defines the trailing character
→	inForceSignPrinting	const bool		False	Forces printing the sign of the number even if the number is positive
→	inSuffix	const String&		\"\"	Defines a suffix. Generally it is an unit of value (e.g. mm)
→	inSystemBase	const int	2 - 16	10	The base of the numeral system
←	outString	String&			Output string

Examples

→	←
inInteger = 200 **inDigitCount** = 4 **inTrailingCharacter** = "0" **inForceSignPrinting** = False **inSuffix** = "" **inSystemBase** = 16	**outString** = "00C8"

In the above example the input value is 200. Formatting is configured for displaying 4-digit hexadecimal number by setting the **inDigitCount** input. The value of 200 expressed in hexadecimal numeral system is represented as **0xC8**. The final result is filled in with zeros.

→	←
inInteger = 25 **inDigitCount** = 4 **inTrailingCharacter** = "-" **inForceSignPrinting** = False **inSuffix** = "" **inSystemBase** = 10	**outString** = "--25"

The next example shows you how final representation of a number can be filled with arbitrary chosen sign. First two characters are dashes as defined in **inTrailingCharacter**.

Errors

List of possible exceptions:

Error type	Description
DomainError	inTrailingCharacter has to be a single character in FormatIntegerToString.

See Also

- FormatRealToString – Creates a string from a real number using a proper format.
- FormatDoubleToString – Creates a string from a double number using a proper format.
- FormatString – Creates a string according to the given format and data.

FormatRealToString

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Creates a string from a real number using a proper format.

Applications: Useful for preparing a number for display or communication with specific number of fractional digits, sign etc.

Syntax

```
void avl::FormatRealToString
(
 const float inReal,
 const int inIntegerDigitCount,
 const int inFractionalDigitCount,
 const atl::String& inDecimalMark,
 const atl::String& inTrailingCharacter,
 const bool inForceSignPrinting,
 const atl::String& inSuffix,
 atl::String& outString
)
```

Parameters

	Name	Type	Range	Default	Description
→	inReal	const float			Input real
→	inIntegerDigitCount	const int	0 - 1000		How many characters the integer part of the input real should have at least
→	inFractionalDigitCount	const int	0 - 100	3	How many characters the fractional part of the input real should have
→	inDecimalMark	const String&		\".\"	The symbol used to separate the integer part from the fractional part of the number
→	inTrailingCharacter	const String&		\"0\"	Defines the trailing character
→	inForceSignPrinting	const bool		False	Forces printing the sign of the number even if the number is positive
→	inSuffix	const String&		\"\"	Defines a suffix. Generally it is an unit of value (e.g. mm)
←	outString	String&			Output string

Examples

→	←
inReal = 3.14159265359 **inIntegerDigitCount** = 2 **inFractionalDigitCount** = 3 **inDecimalMark** = "." **inTrailingCharacter** = "0" **inForceSignPrinting** = False **inSuffix** = "..."	**outString** = "03.142..."

In the example above desired integer digit count defined in **inIntegerDigitCount** equals 2, so the filter attaches trailing sign (zero) at the beginning of the result **outString**. Decimal mark (**inDecimalMark**) is set as a dot and the number of fractional digits (**inFractionalDigitCount**) as 3. The result string ends with ellipsis defined in **inSuffix** and the final result is "03.142...".
Note that the third digit was rounded up to "2" due to the succeeding digit.

→	←
inReal = 10.7 **inIntegerDigitCount** = 1 **inFractionalDigitCount** = 2 **inDecimalMark** = "," **inTrailingCharacter** = " " **inForceSignPrinting** = True **inSuffix** = " °C"	**outString** = "+10,70 °C"

This example demonstrates formatting of value 10.7 with one integer digit as defined in **inIntegerDigitCount** and two fractional digits described by **inFractionalDigitCount**. Note that the integer part of formatted number (**1**) is less than digits in integral part of passed value (**10**), but it doesn't affect to either (unlike the fractional part which can cuts off excessing digits).
Trailing character (**inTrailingCharacter**) doesn't affect to anything, because desired integer digit count is one (the same effect would be for zero).

Remarks

If you want to cut off excessing digits from a fractional part of a number (instead of rounding them), you can format a number with one extra precision digit and then use Substring to cut off the last digit. You can add a suffix manually by using AvsFilter_ConcatenateStrings.

Errors

List of possible exceptions:

Error type	Description
DomainError	inDecimalMark has to be a single character in FormatRealToString.
DomainError	inTrailingCharacter has to be a single character in FormatRealToString.

See Also

- FormatDoubleToString – Creates a string from a double number using a proper format.
- FormatIntegerToString – Creates a string from an integer number using a proper format.
- FormatString – Creates a string according to the given format and data.

LoadText

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Loads a text from a file.

Syntax

```
void avl::LoadText
(
 const atl::File& inFile,
 atl::String& outText
)
```

Parameters

	Name	Type	Default	Description
→	inFile	const File&		
←	outText	String&		

LoadTextLines

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Loads text lines from a file.

Syntax

```
void avl::LoadTextLines
(
 const atl::File& inFile,
 const atl::String& inLineDelimiter,
 bool inSkipEmptyLines,
 atl::Array<atl::String>& outTextLines
)
```

Parameters

	Name	Type	Default	Description
→	inFile	const File&		
→	inLineDelimiter	const String&	\"\\\\r\\\\n\"	End of line character sequence, escaped
→	inSkipEmptyLines	bool		
←	outTextLines	Array<String>&		

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty LineDelimiter on input in LoadTextLines.

SaveText

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Saves a text to a file.

Syntax

```
void avl::SaveText
(
 const atl::String& inText,
 avl::FileAccessMode::Type inFileAccessMode,
 const atl::File& inFile,
 bool inIgnoreErrors
)
```

Parameters

	Name	Type	Default	Description
➡	inText	const String&		
➡	inFileAccessMode	FileAccessMode::Type		
➡	inFile	const File&		
➡	inIgnoreErrors	bool		

SaveTextLines

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Saves text lines to a file.

Syntax

```
void avl::SaveTextLines
(
 const atl::Array<atl::String>& inTextLines,
 avl::FileAccessMode::Type inFileAccessMode,
 const atl::File& inFile,
 const atl::String& inLineDelimiter,
 bool inIgnoreErrors
)
```

Parameters

	Name	Type	Default	Description
➡	inTextLines	const Array<String>&		
➡	inFileAccessMode	FileAccessMode::Type		
➡	inFile	const File&		
➡	inLineDelimiter	const String&	\"\\\\r\\\\n\"	End of line character sequence, escaped
➡	inIgnoreErrors	bool		

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty LineDelimiter on input in SaveTextLines.

138. Process

Table of content:

ReadEnvironmentVariable

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Reads environment variable by given name.

Syntax

```
void avl::ReadEnvironmentVariable
(
 const atl::String& inVariableName,
 atl::String& outValue
)
```

Parameters

	Name	Type	Default	Description
→	inVariableName	const String&		Variable name to be read
←	outValue	String&		Variable value

Errors

List of possible exceptions:

Error type	Description
IoError	Variable: 'Variable name' was not found.

ReadEnvironmentVariable_OrNil

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Reads environment variable by given name.

Syntax

```
void avl::ReadEnvironmentVariable_OrNil
(
 const atl::String& inVariableName,
 atl::Optional<atl::String>& outValue
)
```

Parameters

	Name	Type	Default	Description
→	inVariableName	const String&		Variable name to be read
←	outValue	Optional<String>&		Variable value or nil

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outValue**.

Read more about Optional Outputs.

139. Time

Table of content:

CurrentDateTime

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Returns a string containing the date time information in selected format and all of the date time data separately.

Syntax

```
void avl::CurrentDateTime
(
 const atl::String& inFormat,
 atl::String& outDateTimeString,
 avl::DateTime& outDateTime
)
```

Parameters

	Name	Type	Default	Description
→	inFormat	const String&	\"%x %X\"	Date time string format
←	outDateTimeString	String&		Date time string
←	outDateTime	DateTime&		

Description

inFormat string containing any combination of regular characters and special format specifiers. These format specifiers are replaced by the function to the corresponding values to represent the time. They all begin with a percentage (%) sign, and are:

specifier	Replaced by	Example
%a	Abbreviated day of the week name *	Thu
%A	Full day of the week name *	Thursday
%b	Abbreviated month name *	Aug
%B	Full month name *	August
%c	Date and time representation *	Thu Aug 23 14:55:02 2001
%d	Day of the month (01-31)	23
%H	Hour in 24h format (00-23)	14
%I	Hour in 12h format (01-12)	02
%j	Day of the year (001-366)	235
%m	Month as a decimal number (01-12)	08
%M	Minute (00-59)	55
%p	AM or PM designation	PM
%S	Second (00-61)	02
%U	Week number with the first Sunday as the first day of week one (00-53)	33
%w	Day of the week as a decimal number with Sunday as 0 (0-6)	4
%W	Week number with the first Monday as the first day of week one (00-53)	34
%x	Date representation *	08/23/01
%X	Time representation *	14:55:02
%y	Year, last two digits (00-99)	01
%Y	Year	2001
%Z	Timezone name or abbreviation	CDT
%z	Timezone offset in ±HHMM format	+0200
%%	A % sign	%

* The specifiers whose description is marked with an asterisk (*) are locale-dependent.

The other parameters are calendar date and time broken down into its components as integers.

Name	Meaning	Range
outDateTime.Milliseconds	Milliseconds	[0-999]
outDateTime.Seconds	Seconds	[0-60] *
outDateTime.Minutes	Minutes	[0-59]
outDateTime.Hour	Hour in 24h format	[0-23]
outDateTime.DayOfMonth	Day of the month	[1-31]
outDateTime.Month	Month as a decimal number	[1-12]
outDateTime.Year	Year	
outDateTime.DayOfWeek	Day of the week as a decimal number with Sunday as 0	[0-6]
outDateTime.DayOfYear	Day of the year	[1-366]
outDateTime.TimezoneOffset	Offset relative to UTC in seconds (negative value for positive time zone).	
outDateTime.DaylightSaving	Whether current local time is subject to Daylight Saving.	
outDateTime.UnixTime	Number of seconds that have elapsed since the Unix epoch.	

* Number of seconds is usually in range [0-59], value 60 is only present because of leap seconds.

Errors

List of possible exceptions:

Error type	Description
	Incorrect date time format
DomainError	Provided incorrect date time format. Please refer to the official documentation for more information.

Delay

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Suspends the program workflow for inTime milliseconds.

Syntax

```
void avl::Delay
(
 int inTime
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inTime	int	0 - 10000	100	Target time in milliseconds

Description

This filter suspends the program workflow for **inTime** milliseconds.

See Also

- DelayByPeriod – Suspends the program workflow for inTime milliseconds relative to the end of the filter's last invoke time.
- MeasurePeriod – Returns elapsed time in milliseconds from last filter call.

DelayByPeriod

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Suspends the program workflow for inTime milliseconds relative to the end of the filter's last invoke time.

Syntax

```
void avl::DelayByPeriod
(
 DelayByPeriodState& ioState,
 int inTime
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	DelayByPeriodState&			Object used to maintain state of the function.
➡	inTime	int	0 - 10000	100	Target period in milliseconds

Description

This filter suspends the program workflow for **inTime** milliseconds relative to the end of the filter's last invoke time.

See Also

- Delay – Suspends the program workflow for inTime milliseconds.
- MeasurePeriod – Returns elapsed time in milliseconds from last filter call.

GetClockTime

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Stops clock to measure performance.

Syntax

```
void avl::GetClockTime
(
 const avl::ClockTime& inStartTime,
 double& outElapsedTime
)
```

Parameters

	Name	Type	Default	Description
→	inStartTime	const ClockTime&		Start time - use StartClock filter
←	outElapsedTime	double&		Time in milliseconds from inStartTime

MeasurePeriod

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Returns elapsed time in milliseconds from last filter call.

Syntax

```
void avl::MeasurePeriod
(
 MeasureState& ioState,
 float& outTime,
 float& outFrequency
)
```

Parameters

	Name	Type	Default	Description
↔	ioState	MeasureState&		Object used to maintain state of the function.
←	outTime	float&		Elapsed time in milliseconds
←	outFrequency	float&		The resulting frequency in Hz or FPS

Description

This filter returns elapsed time **outTime** in milliseconds from it's last call.

The **outFrequency** is the momentary frequency in Hz, i.e. iterations per second.

Remarks

On the first call of this filter, the returned values of **outTime** and **outFrequency** are zero.

See Also

- DelayByPeriod – Suspends the program workflow for inTime milliseconds relative to the end of the filter's last invoke time.
- Delay – Suspends the program workflow for inTime milliseconds.

StartClock

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Starts clock to measure performance.

Syntax

```
void avl::StartClock
(
 avl::ClockTime& outStartTime
)
```

Parameters

	Name	Type	Default	Description
⬅	outStartTime	ClockTime&		Start time

140. Random

Table of content:

CreateRandomArray

Header: STD.h
Namespace: avl
Module: FoundationLite

Creates array with random values.

Syntax

```
void avl::CreateRandomArray
(
 const int inLength,
 const float inMinValue,
 const float inMaxValue,
 const float inStep,
 atl::Optional<int> inSeed,
 atl::Array<float>& outArray
)
```

Parameters

	Name	Type	Range	Default	Description
→	inLength	const int		10	Length of output array
→	inMinValue	const float			Minimal generated value
→	inMaxValue	const float		10.0f	Maximal generated value
→	inStep	const float	0.0001 - ∞	1.0f	Minimal difference between two generated values
→	inSeed	Optional<int>		NIL	Random seed used to generate values
←	outArray	Array<float>&			

Remarks

This filter should not be used for generating cryptographically secure random numbers.

Errors

List of possible exceptions:

Error type	Description
DomainError	Value of inLength is non-positive.
DomainError	Value of inMinValue is greater than value of inMaxValue.
DomainError	Value of inStep is greater than span between maximal and minimal value.
DomainError	Values inMinValue and inMaxValue are equal.

CreateRandomMatrix

Header: STD.h
Namespace: avl
Module: FoundationLite

Creates matrix with random values in closed interval.

Syntax

```
void avl::CreateRandomMatrix
(
 const int inColumnCount,
 const int inRowCount,
 const float inMinValue,
 const float inMaxValue,
 const float inStep,
 atl::Optional<int> inSeed,
 avl::Matrix& outMatrix
)
```

Parameters

	Name	Type	Range	Default	Description
→	inColumnCount	const int		10	Columns count of generated matrix
→	inRowCount	const int		10	Rows count of generated matrix
→	inMinValue	const float			Minimal generated value
→	inMaxValue	const float		10.0f	Maximal generated value
→	inStep	const float	0.0001 - ∞	1.0f	Minimal difference between two random values
→	inSeed	Optional<int>		NIL	Random seed used to generate values
←	outMatrix	Matrix&			

Remarks

This filter should not be used for generating cryptographically secure random numbers.

Errors

List of possible exceptions:

Error type	Description
DomainError	Value of inColumnCount is non-positive.
DomainError	Value of inMinValue is greater than value of inMaxValue.
DomainError	Value of inRowCount is non-positive.
DomainError	Value of inStep is greater than span between maximal and minimal value.
DomainError	Values inMinValue and inMaxValue are equal.

RandomInteger

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Creates random integer value in given closed interval.

Syntax

```
void avl::RandomInteger
(
 RandomState& ioState,
 const int inMinValue,
 const int inMaxValue,
 atl::Optional<int> inSeed,
 int& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	RandomState&		Object used to maintain state of the function.
	inMinValue	const int		Minimal generated value
	inMaxValue	const int	10	Maximal generated value
	inSeed	Optional<int>	NIL	Random seed used to generate values
	outValue	int&		

Remarks

This filter should not be used for generating cryptographically secure random numbers.

Errors

List of possible exceptions:

Error type	Description
DomainError	Value of inMinValue is greater than value of inMaxValue.
DomainError	Values inMinValue and inMaxValue are equal.

RandomReal

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Creates random real value in given closed interval.

Syntax

```
void avl::RandomReal
(
 RandomState& ioState,
 const float inMinValue,
 const float inMaxValue,
 atl::Optional<int> inSeed,
 float& outValue
)
```

Parameters

	Name	Type	Default	Description
↻	ioState	RandomState&		Object used to maintain state of the function.
→	inMinValue	const float		Minimal generated value
→	inMaxValue	const float	1.0f	Maximal generated value
→	inSeed	Optional<int>	NIL	Random seed used to generate values
←	outValue	float&		

Remarks

This filter should not be used for generating cryptographically secure random numbers.

Errors

List of possible exceptions:

Error type	Description
DomainError	Value of inMinValue is greater than value of inMaxValue.

ShuffleArray

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Creates an array by setting input array values in the random positions.

Syntax

```
void avl::ShuffleArray
(
 const atl::Array<Type>& inArray,
 atl::Optional<int> inSeed,
 atl::Array<Type>& outArray
)
```

Parameters

	Name	Type	Default	Description
→	inArray	const Array<Type>&		Array to shuffle
→	inSeed	Optional<int>	NIL	Random seed used to generate values
←	outArray	Array<Type>&		

Remarks

This filter should not be used for generating cryptographically secure random numbers.

141. Unit Conversions

Table of content:

DegreesToRadians

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Converts degrees to radians.

Syntax

```
void avl::DegreesToRadians
(
 float inDegrees,
 float& outRadians
)
```

Parameters

	Name	Type	Default	Description
→	inDegrees	float		
←	outRadians	float&		

MillimetresToPixels

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Converts a length in millimetres to pixels using the given scale [px / mm].

Syntax

```
void avl::MillimetresToPixels
(
 float inMillimetres,
 float inScale,
 float& outPixels
)
```

Parameters

	Name	Type	Default	Description
→	inMillimetres	float		
→	inScale	float		px / mm
←	outPixels	float&		

PixelsToMillimetres

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Converts a length in pixels to millimetres using the given scale [px / mm].

Syntax

```
void avl::PixelsToMillimetres
(
 float inPixels,
 float inScale,
 float& outMillimetres
)
```

Parameters

	Name	Type	Default	Description
→	inPixels	float		
→	inScale	float		px / mm
←	outMillimetres	float&		

RadiansToDegrees

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Converts radians to degrees.

Syntax

```
void avl::RadiansToDegrees
(
 float inRadians,
 float& outDegrees
)
```

Parameters

	Name	Type	Default	Description
→	inRadians	float		
←	outDegrees	float&		

142. User Input

Table of content:

GetKeyboardKeys

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Returns virtual key codes of the pressed keys or NULL if no key was pressed.

Syntax

```
void avl::GetKeyboardKeys
(
 const bool inNumKeypad,
 const bool inAlpha,
 const bool inSpecial,
 const bool inFKeys,
 atl::Array<int>& outKeyCodes
)
```

Parameters

	Name	Type	Default	Description
→	inNumKeypad	const bool	True	Determines whether to analyze keys from numeric keypad
→	inAlpha	const bool	True	Determines whether to analyze keys from alphanumeric keypad
→	inSpecial	const bool		Determines whether to analyze special keys
→	inFKeys	const bool		Determines whether to analyze functional keys (F1 to F24)
←	outKeyCodes	Array<int>&		Virtual key codes of the pressed keys. See documentation for a complete table

Remarks

For complete list of virtual key codes, please follow the link below:
https://docs.microsoft.com/windows/desktop/inputdev/virtual-key-codes

See Also

- GetKeyboardKeyState – Checks if the specified keyboard key is down and if it is toggled.

GetKeyboardKeyState

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Checks if the specified keyboard key is down and if it is toggled.

Syntax

```
void avl::GetKeyboardKeyState
(
 const int inKeyCode,
 bool& outIsDown,
 bool& outIsToggled
)
```

Parameters

	Name	Type	Default	Description
→	inKeyCode	const int		Virtual key code. See documentation for a complete table.
←	outIsDown	bool&		Informs if the specified keyboard key is being pressed at the moment.
←	outIsToggled	bool&		Informs about the state. Important for keys such as CapsLock.

Remarks

For complete list of virtual key codes, please follow the link below:
https://docs.microsoft.com/windows/desktop/inputdev/virtual-key-codes

See Also

- GetKeyboardKeys – Returns virtual key codes of the pressed keys or NULL if no key was pressed.

143. INI

Table of content:

INI_AddBoolValue

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Adds or updates an INI value in the specified section & key.

Syntax

```
void avl::INI_AddBoolValue
(
 const atl::String& inINI,
 const atl::String& inSection,
 const atl::String& inKey,
 const bool inValue,
 atl::Optional<const atl::String&> inComment,
 bool inMultipleValues,
 atl::String& outINI
)
```

Parameters

	Name	Type	Default	Description
→	inINI	const String&		INI
→	inSection	const String&		Section name
→	inKey	const String&		Key name
→	inValue	const bool		Key value
→	inComment	Optional<const String&>	NIL	Optional comment
→	inMultipleValues	bool	False	Allow multiple values in one key
←	outINI	String&		

INI_AddIntegerValue

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Adds or updates an INI value in the specified section & key.

Syntax

```
void avl::INI_AddIntegerValue
(
 const atl::String& inINI,
 const atl::String& inSection,
 const atl::String& inKey,
 const int inValue,
 atl::Optional<const atl::String&> inComment,
 bool inMultipleValues,
 atl::String& outINI
)
```

Parameters

	Name	Type	Default	Description
→	inINI	const String&		INI
→	inSection	const String&		Section name
→	inKey	const String&		Key name
→	inValue	const int		Key value
→	inComment	Optional<const String&>	NIL	Optional comment
→	inMultipleValues	bool	False	Allow multiple values in one key
←	outINI	String&		

INI_AddRealValue

Header: STD.h
Namespace: avl
Module: FoundationLite

Adds or updates an INI value in the specified section & key.s

Syntax

```
void avl::INI_AddRealValue
(
 const atl::String& inINI,
 const atl::String& inSection,
 const atl::String& inKey,
 const float inValue,
 atl::Optional<const atl::String&> inComment,
 bool inMultipleValues,
 atl::String& outINI
)
```

Parameters

	Name	Type	Default	Description
→	inINI	const String&		INI
→	inSection	const String&		Section name
→	inKey	const String&		Key name
→	inValue	const float		Key value
→	inComment	Optional<const String&>	NIL	Optional comment
→	inMultipleValues	bool	False	Allow multiple values in one key
←	outINI	String&		

INI_AddSection

Header: STD.h
Namespace: avl
Module: FoundationLite

Adds a new section.

Syntax

```
void avl::INI_AddSection
(
 const atl::String& inINI,
 const atl::String& inSection,
 atl::Optional<const atl::String&> inComment,
 atl::String& outINI
)
```

Parameters

	Name	Type	Default	Description
→	inINI	const String&		INI
→	inSection	const String&		Section name
→	inComment	Optional<const String&>	NIL	Optional comment
←	outINI	String&		

INI_AddStringValue

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Adds or updates an INI value in the specified section & key.

Syntax

```
void avl::INI_AddStringValue
(
 const atl::String& inINI,
 const atl::String& inSection,
 const atl::String& inKey,
 const atl::String& inValue,
 atl::Optional<const atl::String&> inComment,
 bool inMultipleValues,
 atl::String& outINI
)
```

Parameters

	Name	Type	Default	Description
→	inINI	const String&		INI
→	inSection	const String&		Section name
→	inKey	const String&		Key name
→	inValue	const String&		Key value
→	inComment	Optional<const String&>	NIL	Optional comment
→	inMultipleValues	bool	False	Allow multiple values in one key
←	outINI	String&		

INI_GetAllSectionKeys

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Gets all key names within a section.

Syntax

```
void avl::INI_GetAllSectionKeys
(
 const atl::String& inINI,
 const atl::String& inSectionName,
 atl::Array<atl::String>& outKeys
)
```

Parameters

	Name	Type	Default	Description
→	inINI	const String&		INI
→	inSectionName	const String&		Section name
←	outKeys	Array<String>&		

INI_GetAllSections

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Gets all section names in the INI.

Syntax

```
void avl::INI_GetAllSections
(
 const atl::String& inINI,
 atl::Array<atl::String>& outSections
)
```

Parameters

	Name	Type	Default	Description
→	inINI	const String&		INI
←	outSections	Array<String>&		

INI_GetAllValues

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Gets all values within a section & key.

Syntax

```
void avl::INI_GetAllValues
(
 const atl::String& inINI,
 const atl::String& inSection,
 const atl::String& inKey,
 atl::Array<atl::String>& outValues
)
```

Parameters

	Name	Type	Default	Description
→	inINI	const String&		INI
→	inSection	const String&		Section name
→	inKey	const String&		Key name
←	outValues	Array<String>&		

INI_GetBoolValue

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Gets value from section & key.

Syntax

```
void avl::INI_GetBoolValue
(
 const atl::String& inINI,
 const atl::String& inSection,
 const atl::String& inKey,
 const bool inDefault,
 bool& outValue
)
```

Parameters

	Name	Type	Default	Description
→	inINI	const String&		INI
→	inSection	const String&		Section name
→	inKey	const String&		Key name
→	inDefault	const bool		Value to return if the key is not found
←	outValue	bool&		

INI_GetIntegerValue

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Gets value from section & key.

Syntax

```
void avl::INI_GetIntegerValue
(
 const atl::String& inINI,
 const atl::String& inSection,
 const atl::String& inKey,
 const int inDefault,
 int& outValue
)
```

Parameters

	Name	Type	Default	Description
→	inINI	const String&		INI
→	inSection	const String&		Section name
→	inKey	const String&		Key name
→	inDefault	const int		Value to return if the key is not found
←	outValue	int&		

INI_GetRealValue

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Gets value from section & key.

Syntax

```
void avl::INI_GetRealValue
(
 const atl::String& inINI,
 const atl::String& inSection,
 const atl::String& inKey,
 const float inDefault,
 float& outValue
)
```

Parameters

	Name	Type	Default	Description
→	inINI	const String&		INI
→	inSection	const String&		Section name
→	inKey	const String&		Key name
→	inDefault	const float		Value to return if the key is not found
←	outValue	float&		

INI_GetStringValue

Header: STD.h
Namespace: avl
Module: FoundationLite

Gets value from section & key.

Syntax

```
void avl::INI_GetStringValue
(
 const atl::String& inINI,
 const atl::String& inSection,
 const atl::String& inKey,
 const atl::String& inDefault,
 atl::String& outValue
)
```

Parameters

	Name	Type	Default	Description
➡	inINI	const String&		INI
➡	inSection	const String&		Section name
➡	inKey	const String&		Key name
➡	inDefault	const String&		Value to return if the key is not found
⬅	outValue	String&		

INI_LoadFile

Header: STD.h
Namespace: avl
Module: FoundationLite

Loads an INI file.

Syntax

```
void avl::INI_LoadFile
(
 const atl::File& inFile,
 bool inMultipleValues,
 atl::String& outINI
)
```

Parameters

	Name	Type	Default	Description
➡	inFile	const File&		Path to the source file
➡	inMultipleValues	bool	True	Allow multiple values in one key
⬅	outINI	String&		

INI_SaveFile

Header: STD.h
Namespace: avl
Module: FoundationLite

Saves an INI file.

Syntax

```
void avl::INI_SaveFile
(
 const atl::String& inINI,
 bool inMultipleValues,
 const atl::File& inFile
)
```

Parameters

	Name	Type	Default	Description
➡	inINI	const String&		INI
➡	inMultipleValues	bool	False	Allow multiple values in one key
➡	inFile	const File&		Path to a file

144. Binary Data

Table of content:

LoadBuffer

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Reads bytes from a file and returns them as a bytes buffer.

Syntax

```
void avl::LoadBuffer
(
 const atl::File& inFile,
 avl::ByteBuffer& outBuffer
)
```

Parameters

	Name	Type	Default	Description
→	inFile	const File&		
←	outBuffer	ByteBuffer&		

Errors

List of possible exceptions:

Error type	Description
IoError	File is too large for byte buffer.

SaveBuffer

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Writes bytes from a byte buffer to a file.

Syntax

```
void avl::SaveBuffer
(
 const avl::ByteBuffer& inBuffer,
 avl::FileAccessMode::Type inFileAccessMode,
 const atl::File& inFile,
 bool inIgnoreErrors
)
```

Parameters

	Name	Type	Default	Description
→	inBuffer	const ByteBuffer&		
→	inFileAccessMode	FileAccessMode::Type	CreateOrErase	
→	inFile	const File&		
→	inIgnoreErrors	bool		

145. Modbus TCP

Table of content:

ModbusTCP_Close

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Closes a connected Modbus socket gracefully.

Syntax

```
void avl::ModbusTCP_Close
(
 avl::SocketId inSocket
)
```

Parameters

	Name	Type	Default	Description
→	inSocket	SocketId		Connected socket Id.

Hints

- Connect **inSocket** with the output of ModbusTCP_Connect.

See Also

- ModbusTCP_Connect – Connects as a client to a remote Modbus server socket.

ModbusTCP_Connect

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Connects as a client to a remote Modbus server socket.

Syntax

```
void avl::ModbusTCP_Connect
(
 TcpIpConnectState& ioState,
 const atl::String& inHost,
 int inPort,
 const atl::Optional<int>& inTimeout,
 const atl::Optional<int>& inKeepAliveTime,
 atl::Conditional<avl::SocketId>& outSocket
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	TcpIpConnectState&			Object used to maintain state of the function.
→	inHost	const String&		\"localhost\"	The hostname or IP address to connect to.
→	inPort	int	7 - 65535	502	Modbus port of host to connect to. 502 is the default one.
→	inTimeout	const Optional<int>&	500 - ∞	NIL	Timeout in milliseconds, block if not specified.
→	inKeepAliveTime	const Optional<int>&	2000 - ∞	NIL	When specified activates Tcp/Ip keep alive on new socket with given idle time.
←	outSocket	Conditional<SocketId>&			Connected socket ID.

Description

For more details please check help of TcpIp_Connect filter. The filters works in the same way.

See Also

- ModbusTCP_Close – Closes a connected Modbus socket gracefully.

ModbusTCP_ForceMultipleCoils

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Function Code 15. Forces each coil in a sequence of coils to either ON or OFF in a remote device

Syntax

```
void avl::ModbusTCP_ForceMultipleCoils
(
 const avl::SocketId& inSocket,
 const atl::Optional<int>& inTimeout,
 int inUnitID,
 int inStartingAddress,
 const atl::Array<bool>& inStatuses
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSocket	const SocketId&			Connected socket ID on port 502.
→	inTimeout	const Optional<int>&	10 - ∞	NIL	Timeout in milliseconds, block if not specified.
→	inUnitID	int	0 - 255	1	Default is 1.
→	inStartingAddress	int	0 - 65535		The address of the first coil.
→	inStatuses	const Array<bool>&			

Errors

List of possible exceptions:

Error type	Description
DomainError	Size of statuses array should be between 1 and 1968.

See Also

- ModbusTCP_ReadCoils – Function Code 01. Reads contiguous status of coils in a remote device
- ModbusTCP_WriteCoil – Function Code 05. Writes a single output to either ON or OFF in a remote device

ModbusTCP_ReadCoils

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Function Code 01. Reads contiguous status of coils in a remote device

Syntax

```
void avl::ModbusTCP_ReadCoils
(
 const avl::SocketId& inSocket,
 const atl::Optional<int>& inTimeout,
 int inUnitID,
 int inStartingAddress,
 int inBitCount,
 atl::Array<bool>& outBits
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSocket	const SocketId&			Connected socket ID on port 502.
→	inTimeout	const Optional<int>&	10 - ∞	NIL	Timeout in milliseconds, block if not specified.
→	inUnitID	int	0 - 255	1	Unit identifier.
→	inStartingAddress	int	0 - 65535		The address of the first coil.
→	inBitCount	int	1 - 2000	1	Requested number of bits.
←	outBits	Array<bool>&			Response.

See Also

- ModbusTCP_WriteCoil – Function Code 05. Writes a single output to either ON or OFF in a remote device
- ModbusTCP_ForceMultipleCoils – Function Code 15. Forces each coil in a sequence of coils to either ON or OFF in a remote device

ModbusTCP_ReadDiscreteInputs

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Function Code 02. Reads contiguous status of discrete inputs in a remote device

Syntax

```
void avl::ModbusTCP_ReadDiscreteInputs
(
 const avl::SocketId& inSocket,
 const atl::Optional<int>& inTimeout,
 int inUnitID,
 int inStartingAddress,
 int inBitCount,
 atl::Array<bool>& outBits
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSocket	const SocketId&			Connected socket ID on port 502.
→	inTimeout	const Optional<int>&	10 - ∞	NIL	Timeout in milliseconds, block if not specified.
→	inUnitID	int	0 - 255	1	Default is 1.
→	inStartingAddress	int	0 - 65535		The address of the first input.
→	inBitCount	int	1 - 2000	1	Requested number of bits.
←	outBits	Array<bool>&			Response.

ModbusTCP_ReadExceptionStatus

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Function Code 07. Attempts to read Exception Status

Syntax

```
void avl::ModbusTCP_ReadExceptionStatus
(
 const avl::SocketId& inSocket,
 const atl::Optional<int>& inTimeout,
 int inUnitID,
 int& outStatus
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSocket	const SocketId&			Connected socket ID on port 502.
→	inTimeout	const Optional<int>&	10 - ∞	NIL	Timeout in milliseconds, block if not specified.
→	inUnitID	int	0 - 255	1	Default is 1
←	outStatus	int&			

Errors

List of possible exceptions:

Error type	Description
IoError	Received message length is too short.

ModbusTCP_ReadInputIntegerRegisters

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Function Code 04. Reads contiguous input registers in a remote device

Syntax

```
void avl::ModbusTCP_ReadInputIntegerRegisters
(
 const avl::SocketId& inSocket,
 const atl::Optional<int>& inTimeout,
 int inUnitID,
 int inStartingAddress,
 int inCount,
 avl::ModbusDataFormat::Type inInputDataFormat,
 atl::Array<int>& outIntegerValues
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSocket	const SocketId&			Connected socket ID on port 502.
→	inTimeout	const Optional<int>&	10 - ∞	NIL	Timeout in milliseconds, block if not specified.
→	inUnitID	int	0 - 255	1	Default is 1
→	inStartingAddress	int	0 - 65535		The address of the first input register.
→	inCount	int	1 - 125	1	
→	inInputDataFormat	ModbusDataFormat::Type			
←	outIntegerValues	Array<int>&			Received integer values

See Also

- ModbusTCP_ReadInputRegisters_AsByteBuffer – Function Code 03. Reads contiguous input registers in a remote device
- ModbusTCP_ReadInputRealRegisters – Function Code 04. Reads contiguous input registers in a remote device

ModbusTCP_ReadInputRealRegisters

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Function Code 04. Reads contiguous input registers in a remote device

Syntax

```
void avl::ModbusTCP_ReadInputRealRegisters
(
 const avl::SocketId& inSocket,
 const atl::Optional<int>& inTimeout,
 int inUnitID,
 int inStartingAddress,
 int inCount,
 atl::Array<float>& outRealValues
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSocket	const SocketId&			Connected socket ID on port 502.
→	inTimeout	const Optional<int>&	10 - ∞	NIL	Timeout in milliseconds, block if not specified.
→	inUnitID	int	0 - 255	1	Default is 1.
→	inStartingAddress	int	0 - 65535		The address of the first input register.
→	inCount	int	1 - 125	1	
←	outRealValues	Array<float>&			Received real values.

See Also

- ModbusTCP_ReadInputRegisters_AsByteBuffer – Function Code 03. Reads contiguous input registers in a remote device
- ModbusTCP_ReadInputIntegerRegisters – Function Code 04. Reads contiguous input registers in a remote device

ModbusTCP_ReadInputRegisters_AsByteBuffer

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Function Code 03. Reads contiguous input registers in a remote device

Syntax

```
void avl::ModbusTCP_ReadInputRegisters_AsByteBuffer
(
 const avl::SocketId& inSocket,
 const atl::Optional<int>& inTimeout,
 int inUnitID,
 int inStartingAddress,
 int inCount,
 avl::ByteBuffer& outData
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSocket	const SocketId&			Connected socket ID on port 502.
→	inTimeout	const Optional<int>&	10 - ∞	NIL	Timeout in milliseconds, block if not specified.
→	inUnitID	int	0 - 255	1	Default is 1.
→	inStartingAddress	int	0 - 65535		The address of the first input register.
→	inCount	int	1 - 125	1	Number of registers to read. Each register is 2 Bytes in size
←	outData	ByteBuffer&			Received register values.

See Also

- ModbusTCP_ReadInputIntegerRegisters – Function Code 04. Reads contiguous input registers in a remote device
- ModbusTCP_ReadInputRealRegisters – Function Code 04. Reads contiguous input registers in a remote device

ModbusTCP_ReadMultipleIntegerRegisters

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Function Code 03. Reads the contents of a contiguous block of holding registers in a remote device

Syntax

```
void avl::ModbusTCP_ReadMultipleIntegerRegisters
(
 const avl::SocketId& inSocket,
 const atl::Optional<int>& inTimeout,
 int inUnitID,
 int inStartingAddress,
 int inCount,
 avl::ModbusDataFormat::Type inInputDataFormat,
 atl::Array<int>& outIntegerValue
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSocket	const SocketId&			Connected socket ID on port 502.
→	inTimeout	const Optional<int>&	10 - ∞	NIL	Timeout in milliseconds, block if not specified.
→	inUnitID	int	0 - 255	1	Default is 1.
→	inStartingAddress	int	0 - 65535		The address of the first holding register.
→	inCount	int	1 - 125	1	
→	inInputDataFormat	ModbusDataFormat::Type			
←	outIntegerValue	Array<int>&			Received integer value.

See Also

- ModbusTCP_ReadMultipleRegisters_AsByteBuffer – Function Code 03. Reads the contents of a contiguous block of holding registers in a remote device
- ModbusTCP_ReadMultipleRealRegisters – Function Code 03. Reads the contents of a contiguous block of holding registers in a remote device
- ModbusTCP_WriteSingleRegister – Function Code 06. Writes a single holding register in a remote device
- ModbusTCP_WriteMultipleRegisters_AsByteBuffer – Function Code 16. Writes a block of contiguous registers (1 to 123 registers) in a remote device
- ModbusTCP_WriteMultipleIntegerRegisters – Function Code 16. Writes a block of contiguous registers (1 to 123 registers) in a remote device
- ModbusTCP_WriteMultipleRealRegisters – Function Code 16. Writes a block of contiguous registers (1 to 123 registers) in a remote device

ModbusTCP_ReadMultipleRealRegisters

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Function Code 03. Reads the contents of a contiguous block of holding registers in a remote device

Syntax

```
void avl::ModbusTCP_ReadMultipleRealRegisters
(
 const avl::SocketId& inSocket,
 const atl::Optional<int>& inTimeout,
 int inUnitID,
 int inStartingAddress,
 int inCount,
 atl::Array<float>& outRealValue
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSocket	const SocketId&			Connected socket ID on port 502.
→	inTimeout	const Optional<int>&	10 - ∞	NIL	Timeout in milliseconds, block if not specified.
→	inUnitID	int	0 - 255	1	Default is 1.
→	inStartingAddress	int	0 - 65535		The address of the first holding register.
→	inCount	int	1 - 125	1	
←	outRealValue	Array<float>&			Received real value.

See Also

- ModbusTCP_ReadMultipleRegisters_AsByteBuffer – Function Code 03. Reads the contents of a contiguous block of holding registers in a remote device
- ModbusTCP_ReadMultipleIntegerRegisters – Function Code 03. Reads the contents of a contiguous block of holding registers in a remote device
- ModbusTCP_WriteSingleRegister – Function Code 06. Writes a single holding register in a remote device
- ModbusTCP_WriteMultipleRegisters_AsByteBuffer – Function Code 16. Writes a block of contiguous registers (1 to 123 registers) in a remote device
- ModbusTCP_WriteMultipleIntegerRegisters – Function Code 16. Writes a block of contiguous registers (1 to 123 registers) in a remote device
- ModbusTCP_WriteMultipleRealRegisters – Function Code 16. Writes a block of contiguous registers (1 to 123 registers) in a remote device

ModbusTCP_ReadMultipleRegisters_AsByteBuffer

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Function Code 03. Reads the contents of a contiguous block of holding registers in a remote device

Syntax

```
void avl::ModbusTCP_ReadMultipleRegisters_AsByteBuffer
(
 const avl::SocketId& inSocket,
 const atl::Optional<int>& inTimeout,
 int inUnitID,
 int inStartingAddress,
 int inCount,
 avl::ByteBuffer& outData
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSocket	const SocketId&			Connected socket ID on port 502.
→	inTimeout	const Optional<int>&	10 - ∞	NIL	Timeout in milliseconds, block if not specified.
→	inUnitID	int	0 - 255	1	Default is 1.
→	inStartingAddress	int	0 - 65535		The address of the first holding register.
→	inCount	int	1 - 125	1	Number of registers to read. Each register is 2 Bytes in size.
←	outData	ByteBuffer&			Received register values.

See Also

- ModbusTCP_ReadMultipleIntegerRegisters – Function Code 03. Reads the contents of a contiguous block of holding registers in a remote device
- ModbusTCP_ReadMultipleRealRegisters – Function Code 03. Reads the contents of a contiguous block of holding registers in a remote device
- ModbusTCP_WriteSingleRegister – Function Code 06. Writes a single holding register in a remote device
- ModbusTCP_WriteMultipleRegisters_AsByteBuffer – Function Code 16. Writes a block of contiguous registers (1 to 123 registers) in a remote device
- ModbusTCP_WriteMultipleIntegerRegisters – Function Code 16. Writes a block of contiguous registers (1 to 123 registers) in a remote device
- ModbusTCP_WriteMultipleRealRegisters – Function Code 16. Writes a block of contiguous registers (1 to 123 registers) in a remote device

ModbusTCP_SendBuffer

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Sends data using Modbus TCP frame format.

Syntax

```
void avl::ModbusTCP_SendBuffer
(
 const avl::SocketId& inSocket,
 const atl::Optional<int>& inTimeout,
 int inUnitID,
 int inFunctionCode,
 const avl::ByteBuffer& inBuffer,
 avl::ByteBuffer& outResponse
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSocket	const SocketId&			Connected socket ID on port 502.
→	inTimeout	const Optional<int>&	10 - ∞	NIL	Timeout in milliseconds, block if not specified.
→	inUnitID	int	0 - 255	1	Default is 1
→	inFunctionCode	int	0 - 255	1	Function code
→	inBuffer	const ByteBuffer&			Data to send
←	outResponse	ByteBuffer&			Response data

ModbusTCP_WriteCoil

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Function Code 05. Writes a single output to either ON or OFF in a remote device

Syntax

```
void avl::ModbusTCP_WriteCoil
(
 const avl::SocketId& inSocket,
 const atl::Optional<int>& inTimeout,
 int inUnitID,
 int inOutputAddress,
 bool inStatus
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inSocket	const SocketId&			Connected socket ID on port 502.
➡	inTimeout	const Optional<int>&	10 - ∞	NIL	Timeout in milliseconds, block if not specified.
➡	inUnitID	int	0 - 255	1	Default is 1
➡	inOutputAddress	int	0 - 65535		The address of the coil to be forced.
➡	inStatus	bool			

See Also

- ModbusTCP_ReadCoils – Function Code 01. Reads contiguous status of coils in a remote device
- ModbusTCP_ForceMultipleCoils – Function Code 15. Forces each coil in a sequence of coils to either ON or OFF in a remote device

ModbusTCP_WriteMultipleIntegerRegisters

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Function Code 16. Writes a block of contiguous registers (1 to 123 registers) in a remote device

Syntax

```
void avl::ModbusTCP_WriteMultipleIntegerRegisters
(
 const avl::SocketId& inSocket,
 const atl::Optional<int>& inTimeout,
 int inUnitID,
 int inStartingAddress,
 const atl::Array<int>& inIntegerValues,
 avl::ModbusDataFormat::Type inDataFormat
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inSocket	const SocketId&			Connected socket ID on port 502.
➡	inTimeout	const Optional<int>&	10 - ∞	NIL	Timeout in milliseconds, block if not specified.
➡	inUnitID	int	0 - 255	1	Default is 1
➡	inStartingAddress	int	0 - 65535		The address of the first register.
➡	inIntegerValues	const Array<int>&			Integer values to set
➡	inDataFormat	ModbusDataFormat::Type			

See Also

- ModbusTCP_ReadMultipleRegisters_AsByteBuffer – Function Code 03. Reads the contents of a contiguous block of holding registers in a remote device
- ModbusTCP_ReadMultipleIntegerRegisters – Function Code 03. Reads the contents of a contiguous block of holding registers in a remote device
- ModbusTCP_ReadMultipleRealRegisters – Function Code 03. Reads the contents of a contiguous block of holding registers in a remote device
- ModbusTCP_WriteSingleRegister – Function Code 06. Writes a single holding register in a remote device
- ModbusTCP_WriteMultipleRegisters_AsByteBuffer – Function Code 16. Writes a block of contiguous registers (1 to 123 registers) in a remote device
- ModbusTCP_WriteMultipleRealRegisters – Function Code 16. Writes a block of contiguous registers (1 to 123 registers) in a remote device

ModbusTCP_WriteMultipleRealRegisters

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Function Code 16. Writes a block of contiguous registers (1 to 123 registers) in a remote device

Syntax

```
void avl::ModbusTCP_WriteMultipleRealRegisters
(
 const avl::SocketId& inSocket,
 const atl::Optional<int>& inTimeout,
 int inUnitID,
 int inStartingAddress,
 const atl::Array<float>& inRealValues
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inSocket	const SocketId&			Connected socket ID on port 502.
➡	inTimeout	const Optional<int>&	10 - ∞	NIL	Timeout in milliseconds, block if not specified.
➡	inUnitID	int	0 - 255	1	Default is 1.
➡	inStartingAddress	int	0 - 65535		The address of the first register.
➡	inRealValues	const Array<float>&			Real values to set.

See Also

- ModbusTCP_ReadMultipleRegisters_AsByteBuffer – Function Code 03. Reads the contents of a contiguous block of holding registers in a remote device
- ModbusTCP_ReadMultipleIntegerRegisters – Function Code 03. Reads the contents of a contiguous block of holding registers in a remote device
- ModbusTCP_ReadMultipleRealRegisters – Function Code 03. Reads the contents of a contiguous block of holding registers in a remote device
- ModbusTCP_WriteSingleRegister – Function Code 06. Writes a single holding register in a remote device
- ModbusTCP_WriteMultipleRegisters_AsByteBuffer – Function Code 16. Writes a block of contiguous registers (1 to 123 registers) in a remote device
- ModbusTCP_WriteMultipleIntegerRegisters – Function Code 16. Writes a block of contiguous registers (1 to 123 registers) in a remote device

ModbusTCP_WriteMultipleRegisters_AsByteBuffer

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Function Code 16. Writes a block of contiguous registers (1 to 123 registers) in a remote device

Syntax

```
void avl::ModbusTCP_WriteMultipleRegisters_AsByteBuffer
(
 const avl::SocketId& inSocket,
 const atl::Optional<int>& inTimeout,
 int inUnitID,
 int inStartingAddress,
 const avl::ByteBuffer& inBuffer
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inSocket	const SocketId&			Connected socket ID on port 502.
➡	inTimeout	const Optional<int>&	10 - ∞	NIL	Timeout in milliseconds, block if not specified.
➡	inUnitID	int	0 - 255	1	Default is 1.
➡	inStartingAddress	int	0 - 65535		The address of the first register.
➡	inBuffer	const ByteBuffer&			Buffer with values to set.

See Also

- ModbusTCP_ReadMultipleRegisters_AsByteBuffer – Function Code 03. Reads the contents of a contiguous block of holding registers in a remote device
- ModbusTCP_ReadMultipleIntegerRegisters – Function Code 03. Reads the contents of a contiguous block of holding registers in a remote device
- ModbusTCP_ReadMultipleRealRegisters – Function Code 03. Reads the contents of a contiguous block of holding registers in a remote device
- ModbusTCP_WriteSingleRegister – Function Code 06. Writes a single holding register in a remote device
- ModbusTCP_WriteMultipleIntegerRegisters – Function Code 16. Writes a block of contiguous registers (1 to 123 registers) in a remote device
- ModbusTCP_WriteMultipleRealRegisters – Function Code 16. Writes a block of contiguous registers (1 to 123 registers) in a remote device

ModbusTCP_WriteSingleRegister

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Function Code 06. Writes a single holding register in a remote device

Syntax

```
void avl::ModbusTCP_WriteSingleRegister
(
 const avl::SocketId& inSocket,
 const atl::Optional<int>& inTimeout,
 int inUnitID,
 int inRegisterAddress,
 int inValue
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inSocket	const SocketId&			Connected socket ID on port 502.
➡	inTimeout	const Optional<int>&	10 - ∞	NIL	Timeout in milliseconds, block if not specified.
➡	inUnitID	int	0 - 255	1	Default is 1.
➡	inRegisterAddress	int	0 - 65535		The address of the holding register to be written.
➡	inValue	int	0 - 65535		Value to send.

See Also

- ModbusTCP_ReadMultipleRegisters_AsByteBuffer – Function Code 03. Reads the contents of a contiguous block of holding registers in a remote device
- ModbusTCP_ReadMultipleIntegerRegisters – Function Code 03. Reads the contents of a contiguous block of holding registers in a remote device
- ModbusTCP_ReadMultipleRealRegisters – Function Code 03. Reads the contents of a contiguous block of holding registers in a remote device
- ModbusTCP_WriteMultipleRegisters_AsByteBuffer – Function Code 16. Writes a block of contiguous registers (1 to 123 registers) in a remote device
- ModbusTCP_WriteMultipleIntegerRegisters – Function Code 16. Writes a block of contiguous registers (1 to 123 registers) in a remote device
- ModbusTCP_WriteMultipleRealRegisters – Function Code 16. Writes a block of contiguous registers (1 to 123 registers) in a remote device

146. Serial Port

Table of content:

SerialPort_Config

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Configures the serial port.

Syntax

```
void avl::SerialPort_Config
(
 SerialPortState& ioState,
 int inPortId,
 const atl::String& inPort,
 const int inBaudRate,
 avl::SerialPortParity::Type inParity,
 const int inDataBits,
 avl::SerialPortFlowControl::Type inFlowControl,
 avl::SerialPortStopBits::Type inStopBits
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SerialPortState&			Object used to maintain state of the function.
	inPortId	int	0 - 7	0	Identifies open port instance when working with multiple serial ports
	inPort	const String&		\"COM1\"	Serial port name
	inBaudRate	const int	1 - ∞	9600	Serial baud rate
	inParity	SerialPortParity::Type			Serial parity
	inDataBits	const int	5 - 8	8	Serial character size
	inFlowControl	SerialPortFlowControl::Type			Serial flow control
	inStopBits	SerialPortStopBits::Type			Serial stop bits

Description

Filter creates serial port connection and set its parameters.

Typically values for **inBaudRate**: 4800, 9600, 14400, 19200, 28800, 38400, 57600, 115200.

By using parameter **inPortId** up to 8 simultaneous connection can be handled. Trying to configure already created connection identified by **inPortId** will result in "Access denied" error.

Remarks

Using the SerialPort_Config filter is necessary before using other filters for serial communication.

Filter SerialPort_Config creates connection only on first execution all next execution will have no effect.

Errors

Trying to connect to port that is already opened will result in error "Could not initialize serial port. open: Access is denied".

Trying to connect to not existing port will result in error "Could not initialize serial port. open: The system cannot find the file specified".

List of possible exceptions:

Error type	Description
	Empty port name in SerialPort_Config.
DomainError	Empty port name is invalid. Typical port names are COM1, COM2, etc.

See Also

- SerialPort_WriteByte – Writes one character in binary mode to serial port.
- SerialPort_WriteBuffer – Writes raw binary data from a byte buffer to serial port.
- SerialPort_WriteString – Writes string characters to serial port.
- SerialPort_ReadByte – Reads one character in binary mode from serial port.
- SerialPort_ReadBuffer – Reads raw binary data from serial port.
- SerialPort_ReadChar – Reads single character from serial port.
- SerialPort_ReadString – Reads string characters from serial port.
- SerialPort_ReadStringUntil – Reads the string from the serial port to encounter a string delimiter.

SerialPort_ReadBuffer

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Reads raw binary data from serial port.

Syntax

```
void avl::SerialPort_ReadBuffer
(
 SerialPortState& ioState,
 int inPortId,
 atl::Conditional<avl::ByteBuffer>& outBuffer
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SerialPortState&			Object used to maintain state of the function.
	inPortId	int	0 - 7	0	Identifies open port instance when working with multiple serial ports
	outBuffer	Conditional<ByteBuffer>&			Conditionally returns non empty byte buffer with raw data, when any data available.

Description

Filter reads all data from the serial port transmission input buffer.

Filter perform non-blocking reading form buffer and will return NIL when no data was read.

Data buffer returned on the **outBuffer** output can be processed using filters from the Binary Data category.

Remarks

Using the SerialPort_Config filter is necessary before using the SerialPort_ReadBuffer filter.

Filter SerialPort_Config does not guarantee that input buffer will be empty.

Errors

Using filters reading from serial port without previous configuration will cause "SerialPort not initialized" error.

See Also

- SerialPort_Config – Configures the serial port.
- SerialPort_ReadByte – Reads one character in binary mode from serial port.
- SerialPort_ReadChar – Reads single character from serial port.
- SerialPort_ReadString – Reads string characters from serial port.
- SerialPort_ReadStringUntil – Reads the string from the serial port to encounter a string delimiter.

SerialPort_ReadByte

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Reads one character in binary mode from serial port.

Syntax

```
void avl::SerialPort_ReadByte
(
 SerialPortState& ioState,
 int inPortId,
 atl::Conditional<int>& outData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SerialPortState&			Object used to maintain state of the function.
	inPortId	int	0 - 7	0	Identifies open port instance when working with multiple serial ports
	outData	Conditional<int>&			Conditionally returns received character value, when one is available.

Description

Filter reads single byte from input buffer.

Filter perform non-blocking reading form buffer and will return NIL when no data was read.

Remarks

Using the SerialPort_Config filter is necessary before using the SerialPort_ReadByte filter.

Filter SerialPort_Config does not guarantee that input buffer will be empty.

Errors

Using filters reading from serial port without previous configuration will cause "SerialPort not initialized" error.

See Also

- SerialPort_Config – Configures the serial port.
- SerialPort_ReadBuffer – Reads raw binary data from serial port.
- SerialPort_ReadChar – Reads single character from serial port.
- SerialPort_ReadString – Reads string characters from serial port.
- SerialPort_ReadStringUntil – Reads the string from the serial port to encounter a string delimiter.

SerialPort_ReadChar

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Reads single character from serial port.

Syntax

```
void avl::SerialPort_ReadChar
(
 SerialPortState& ioState,
 int inPortId,
 atl::Conditional<atl::String>& outCharacter
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SerialPortState&			Object used to maintain state of the function.
	inPortId	int	0 - 7	0	Identifies open port instance when working with multiple serial ports
	outCharacter	Conditional<String>&			Conditionally returns string with one received character.

Description

Filter reads single byte from input buffer and return data as a single character.

Filter perform non-blocking reading form buffer and will return NIL when no data was read.

Remarks

Using the SerialPort_Config filter is necessary before using the SerialPort_ReadChar filter.

Filter SerialPort_Config does not guarantee that input buffer will be empty.

Errors

Using filters reading from serial port without previous configuration will cause "SerialPort not initialized" error.

See Also

- SerialPort_Config – Configures the serial port.
- SerialPort_ReadByte – Reads one character in binary mode from serial port.
- SerialPort_ReadBuffer – Reads raw binary data from serial port.
- SerialPort_ReadString – Reads string characters from serial port.
- SerialPort_ReadStringUntil – Reads the string from the serial port to encounter a string delimiter.

SerialPort_ReadString

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Reads string characters from serial port.

Syntax

```
void avl::SerialPort_ReadString
(
 SerialPortState& ioState,
 int inPortId,
 atl::Conditional<atl::String>& outString
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SerialPortState&			Object used to maintain state of the function.
	inPortId	int	0 - 7	0	Identifies open port instance when working with multiple serial ports
	outString	Conditional<String>&			Conditionally returns string with concatenated all characters available in input buffer.

Description

Filter reads all data from input and returns it as string. Result string will be read in UTF8 format.

Filter perform non-blocking reading form buffer and will return NIL when no data was read.

Remarks

Using the SerialPort_Config filter is necessary before using the SerialPort_ReadString filter.

Filter SerialPort_Config does not guarantee that input buffer will be empty.

Errors

Using filters reading from serial port without previous configuration will cause "SerialPort not initialized" error.

See Also

- SerialPort_Config – Configures the serial port.
- SerialPort_ReadByte – Reads one character in binary mode from serial port.
- SerialPort_ReadBuffer – Reads raw binary data from serial port.
- SerialPort_ReadChar – Reads single character from serial port.
- SerialPort_ReadStringUntil – Reads the string from the serial port to encounter a string delimiter.

SerialPort_ReadStringUntil

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Reads the string from the serial port to encounter a string delimiter.

Syntax

```
void avl::SerialPort_ReadStringUntil
(
 SerialPortState& ioState,
 int inPortId,
 const atl::String& inEndString,
 atl::Conditional<atl::String>& outString
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SerialPortState&			Object used to maintain state of the function.
	inPortId	int	0 - 7	0	Identifies open port instance when working with multiple serial ports
	inEndString	const String&			Delimiter string that will be searched in incoming stream.
	outString	Conditional<String>&			Conditionally returns received string without delimiter.

Description

Filter reads characters from input buffer until find **inEndString** delimiter is found. Returns data without delimiter.

Result string will be read in UTF8 format.

Delimiter provided in **inEndString** must contains only ASCII characters.

Filter perform non-blocking reading form buffer and will return NIL when no data was read.

Remarks

Using the SerialPort_Config filter is necessary before using the SerialPort_ReadStringUntil filter.

Filter SerialPort_Config does not guarantee that input buffer will be empty.

Errors

Using filters reading from serial port without previous configuration will cause "SerialPort not initialized" error.

List of possible exceptions:

Error type	Description
	inEndString must contains only ASCII characters.
DomainError	Termination string must contain only ASCII characters.

See Also

- SerialPort_Config – Configures the serial port.
- SerialPort_ReadByte – Reads one character in binary mode from serial port.
- SerialPort_ReadBuffer – Reads raw binary data from serial port.
- SerialPort_ReadChar – Reads single character from serial port.
- SerialPort_ReadString – Reads string characters from serial port.

SerialPort_WriteBuffer

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Writes raw binary data from a byte buffer to serial port.

Syntax

```
void avl::SerialPort_WriteBuffer
(
 SerialPortState& ioState,
 int inPortId,
 const avl::ByteBuffer& inBuffer
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SerialPortState&			Object used to maintain state of the function.
	inPortId	int	0 - 7	0	Identifies open port instance when working with multiple serial ports
	inBuffer	const ByteBuffer&			Buffer containing raw data to write

Examples

Filter writes raw data block from the **inBuffer** input to the serial port transmission output buffer.

Data buffer for the **inBuffer** input can be prepared using filters from the Binary Data category.

Remarks

Using the SerialPort_Config filter is necessary before using the SerialPort_WriteBuffer filter.

Errors

Using filters writing to serial port without previous configuration will cause "SerialPort not initialized" error.

See Also

- SerialPort_Config – Configures the serial port.
- SerialPort_WriteByte – Writes one character in binary mode to serial port.
- SerialPort_WriteString – Writes string characters to serial port.
- SerialPort_WriteChar – Writes single ASCII character to device.

SerialPort_WriteByte

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Writes one character in binary mode to serial port.

Syntax

```
void avl::SerialPort_WriteByte
(
 SerialPortState& ioState,
 int inPortId,
 int inData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SerialPortState&			Object used to maintain state of the function.
	inPortId	int	0 - 7	0	Identifies open port instance when working with multiple serial ports
	inData	int	0 - 255		Character value.

Description

Filters write single byte into output buffer.

Remarks

Using the SerialPort_Config filter is necessary before using the SerialPort_WriteByte filter.

Errors

Using filters writing to serial port without previous configuration will cause "SerialPort not initialized" error.

See Also

- SerialPort_Config – Configures the serial port.
- SerialPort_WriteBuffer – Writes raw binary data from a byte buffer to serial port.
- SerialPort_WriteString – Writes string characters to serial port.
- SerialPort_WriteChar – Writes single ASCII character to device.
- SerialPort_WriteByte – Writes one character in binary mode to serial port.

SerialPort_WriteChar

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Writes single ASCII character to device.

Syntax

```
void avl::SerialPort_WriteChar
(
 SerialPortState& ioState,
 int inPortId,
 const atl::String& inCharacter
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SerialPortState&			Object used to maintain state of the function.
	inPortId	int	0 - 7	0	Identifies open port instance when working with multiple serial ports
	inCharacter	const String&			Single character to send

Description

Filter writes single character **inCharacter** into input buffer. Input **inCharacter** must contains only single valid ASCII character.

Remarks

Using the SerialPort_Config filter is necessary before using the SerialPort_WriteChar filter.

Errors

Using filters writing to serial port without previous configuration will cause "SerialPort not initialized" error.

List of possible exceptions:

Error type	Description
DomainError	Length inCharacter is greater than 1 in SerialPort_WriteChar.
DomainError	SerialPort_WriteChar can send only ASCII characters.
DomainError	Value inCharacter is empty in SerialPort_WriteChar.

See Also

- SerialPort_Config – Configures the serial port.
- SerialPort_WriteBuffer – Writes raw binary data from a byte buffer to serial port.
- SerialPort_WriteString – Writes string characters to serial port.
- SerialPort_WriteChar – Writes single ASCII character to device.
- SerialPort_WriteByte – Writes one character in binary mode to serial port.

SerialPort_WriteString

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Writes string characters to serial port.

Syntax

```
void avl::SerialPort_WriteString
(
 SerialPortState& ioState,
 int inPortId,
 const atl::String& inString,
 bool inASCIIMode,
 const atl::Array<int>& inTerminator
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SerialPortState&			Object used to maintain state of the function.
	inPortId	int	0 - 7	0	Identifies open port instance when working with multiple serial ports
	inString	const String&			
	inASCIIMode	bool			
	inTerminator	const Array<int>&			

Examples

Filter writes **inString** characters to output buffer. By default string is send using UTF8 encoding (multi-byte).

When **inASCIIMode** is enabled filter will send only ASCII characters. Any attempt of sending non-ASCII character will result in domain error.

Remarks

Using the SerialPort_Config filter is necessary before using the SerialPort_WriteString filter.

Errors

Using filters writing to serial port without previous configuration will cause "SerialPort not initialized" error.

List of possible exceptions:

Error type	Description
	SerialPort_WriteString can send only ASCII characters when inASCIIMode is selected.
DomainError	When option inASCIIMode is enabled, filter can send only an ASCII characters. Please check ASCII table codes for more informations.

See Also

- SerialPort_Config – Configures the serial port.
- SerialPort_WriteBuffer – Writes raw binary data from a byte buffer to serial port.
- SerialPort_WriteChar – Writes single ASCII character to device.
- SerialPort_WriteByte – Writes one character in binary mode to serial port.

147. TCP IP

Table of content:

TcpIp_Accept

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Accepts a connection from a remote client.

Syntax

```
void avl::TcpIp_Accept
(
 TcpIpAcceptState& ioState,
 int inPort,
 const atl::Optional<int>& inTimeout,
 const atl::Optional<int>& inKeepAliveTime,
 bool inNoSendDelay,
 atl::Conditional<avl::SocketId>& outSocket
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	TcpIpAcceptState&			Object used to maintain state of the function.
	inPort	int	7 - 65535	12345	TCP port to listen on.
	inTimeout	const Optional<int>&	0 - ∞	NIL	Timeout in milliseconds, block if not specified.
	inKeepAliveTime	const Optional<int>&	2000 - ∞	NIL	When specified activates Tcp/Ip keep alive on new socket with given idle time.
	inNoSendDelay	bool		False	Disables the Nagle algorithm for the newly connected socket to send the outgoing packets without a delay.
	outSocket	Conditional<SocketId>&			Connected socket ID.

Description

This filter opens a server socket for listening for and accepting incoming connections, which stays open during the execution of the program.

For every execution this filter will wait for connection from a single client, accept this connection and return a new socket representing connection with the client. The **inTimeout** input specifies the longest possible waiting time, in milliseconds. When no timeout is specified (set to Auto), the execution is blocking. When filter execution is terminated by timeout condition no socket is returned (Nil is returned on **outSocket** output) and the operation can be retried later. When timeout is set to zero the filter performs a single check for incoming connection and returns immediately when no connecting clients are available.

Every socket created and returned by this filter must be explicitly closed with TcpIp_Close filter when it is not needed any more.

inKeepAliveTime can be used to enable Keep-Alive mechanism on newly established connection with the client.

Keep-Alive is a mechanism built in the TCP/IP stack, that can be helpful with detecting broken connection in systems with long communication idle periods or with communication in one direction only. When enabled, operating system will send additional control packets to check state of the connection. **inKeepAliveTime** input specifies the time, in milliseconds, of connection idle state after which connection state will be checked. The exact time after which connection broken state will be detected is system dependent.

The Aurora Vision Studio or Aurora Vision Executor process have to have the necessary permissions in local firewall to open a listening port on the executing host. The port is not bound to any specific networking interface or IP address. The port must not be listened to in another process.

This filter can raise IoError when specified port is already in use in local system, or when other unexpected network-related error occurs.

Hints

- Set **inPort** to the same port number that the other side of communication will attempt to connect to.
- Set **inTimeout** if you want the filter to abort waiting for a connection after a predefined time.

See Also

- TcpIp_Connect – Connects as a client to a remote TCP server socket.
- TcpIp_Close – Closes a connected TCP socket gracefully.

TcpIp_Close

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Closes a connected TCP socket gracefully.

Syntax

```
void avl::TcpIp_Close
(
 avl::SocketId inSocket
)
```

Parameters

	Name	Type	Default	Description
➡	inSocket	SocketId		Connected socket Id.

Description

Closes the TCP/IP socket with underlying connection and releases resources owned by the socket.

All sockets, created with TcpIp_Connect and TcpIp_Accept should be closed with this filter after their use, regardless of the state of the connection and regardless of the fact whether the connection was closed by the other side.

Hints

- Connect **inSocket** with the output of TcpIp_Connect or TcpIp_Accept.

See Also

- TcpIp_Accept – Accepts a connection from a remote client.
- TcpIp_Connect – Connects as a client to a remote TCP server socket.

TcpIp_Connect

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Connects as a client to a remote TCP server socket.

Syntax

```
void avl::TcpIp_Connect
(
 TcpIpConnectState& ioState,
 const atl::String& inHost,
 int inPort,
 const atl::Optional<int>& inTimeout,
 const atl::Optional<int>& inKeepAliveTime,
 bool inNoSendDelay,
 atl::Conditional<avl::SocketId>& outSocket
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	TcpIpConnectState&			Object used to maintain state of the function.
	inHost	const String&		\"localhost\"	The hostname or IP address to connect to.
	inPort	int	7 - 65535	12345	TCP port of host to connect to.
	inTimeout	const Optional<int>&	500 - ∞	NIL	Timeout in milliseconds, block if not specified.
	inKeepAliveTime	const Optional<int>&	2000 - ∞	NIL	When specified activates Tcp/Ip keep alive on new socket with given idle time.
	inNoSendDelay	bool		False	Disables the Nagle algorithm for the socket to send the outgoing packets without a delay.
	outSocket	Conditional<SocketId>&			Connected socket ID.

Description

This filter connects to a listening TCP/IP socket.

The **inTimeout** input specifies the longest possible waiting time, in milliseconds, for the connection to be accepted. When no timeout is specified (set to Auto), the execution is blocking. When filter execution is terminated by timeout condition no socket is returned (Nil is returned on **outSocket** output) and the operation can be retried later. When remote host can not be found or when remote host is not listening on specified port the connection attempt is repeated until timeout expires. This filter can block for significantly longer than the timeout specified because its work is not cancelled in the middle of a single connection attempt.

Every socket created and returned by this filter must be explicitly closed with TcpIp_Close filter when it is not needed any more.

inKeepAliveTime can be used to enable Keep-Alive mechanism on newly established connection with the host.

Keep-Alive is a mechanism built in the TCP/IP stack, that can be helpful with detecting broken connection in systems with long communication idle periods or with communication in one direction only. When enabled, operating system will send additional control packets to check state of the connection. **inKeepAliveTime** input specifies the time, in milliseconds, of connection idle state after which connection state will be checked. The exact time after which connection broken state will be detected is system dependent.

The **inHost** parameter can be entered as a DNS host name or IP address. The time necessary to connect can vary, for example subsequent connections to the same host may be faster because of cached DNS information.

In case of failure for reasons other than timeout, IoError is raised.

Hints

- Set **inHost** to an IP address or an url of the server you want to connect to.
- Set **inPort** to the same port number that the other side of communication will be listening on.
- Set **inTimeout** if you want the filter to abort trying to connect after a predefined time.

See Also

- TcpIp_Accept – Accepts a connection from a remote client.
- TcpIp_Close – Closes a connected TCP socket gracefully.

TcpIp_ReadAllBuffer

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Receives data from a connected socket until the other side closes connection.

Syntax

```
void avl::TcpIp_ReadAllBuffer
(
 avl::SocketId inSocket,
 const atl::Optional<int>& inTimeout,
 atl::Conditional<avl::ByteBuffer>& outBuffer,
 bool& outEof
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSocket	SocketId			Connected socket ID, will be closed if successfully read.
→	inTimeout	const Optional<int>&	0 - ∞	NIL	Timeout in milliseconds, block if not specified.
←	outBuffer	Conditional<ByteBuffer>&			Buffer with raw received data, or Nil if the operation was interrupted by timeout.
←	outEof	bool&			Indicates whether the operation successfully read data to the end of stream.

Description

Reads all data until the other side stops sending and closes the connection on its side.

The **inTimeout** input specifies the longest possible waiting time, in milliseconds, for the incoming data to be fully received. When no timeout is specified (set to Auto), the execution is blocking. When the filter execution is terminated by timeout condition no data is returned or removed from the input buffer (Nil is returned instead). In such situation the operation can be retried later. When timeout is et to zero the filter performs a single read attempt and returns immediately when no data is available.

outBuffer is a buffer containing raw binary data (a sequence of bytes). Data buffer can be processed using filters from the Binary Data category.

outEof output is set to True when data was successfully received and the connection is closed by the other side. This output can be used as a more convenient way to detect end of communication in the application.

This filter can raise IoError when the connection is disruptively broken or when other network-related error occurred.

Hints

- Connect **inSocket** with the output of TcpIp_Connect or TcpIp_Accept.
- Set **inTimeout** if you want the filter to abort waiting for data after a predefined time.

See Also

- TcpIp_WriteBuffer – Outputs a block of raw data through a connected TCP socket.
- TcpIp_WriteText – Outputs a string through a connected TCP socket.
- TcpIp_ReadLine – Reads from a connected TCP socket until receiving a specific sequence.
- TcpIp_Close – Closes a connected TCP socket gracefully.

TcpIp_ReadAllText

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Receives text from a connected socket until the other side closes connection.

Syntax

```
void avl::TcpIp_ReadAllText
(
 avl::SocketId inSocket,
 const atl::Optional<int>& inTimeout,
 atl::Conditional<atl::String>& outText,
 bool& outEof
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSocket	SocketId			Connected socket ID, will be closed if successfully read.
→	inTimeout	const Optional<int>&	0 - ∞	NIL	Timeout in milliseconds, block if not specified.
←	outText	Conditional<String>&			Received data as textual string, or Nil if the operation was interrupted by timeout.
←	outEof	bool&			Indicates whether the operation successfully read data to the end of stream.

Description

Reads all data until the other side stops sending and closes the connection on its side.

The **inTimeout** input specifies the longest possible waiting time, in milliseconds, for the incoming data to be fully received. When no timeout is specified (set to Auto), the execution is blocking. When the filter execution is terminated by timeout condition no data is returned or removed from the input buffer (Nil is returned instead). In such situation the operation can be retried later. When timeout is et to zero the filter performs a single read attempt and returns immediately when no data is available.

Received bytes are interpreted as UTF-8 encoded text and returned on **outText** output. When received data is not in a valid UTF-8 encoding an IOError is raised.

outEof output is set to True when data was successfully received and the connection is closed by the other side. This output can be used as a more convenient way to detect end of communication in the application.

This filter can raise IoError when the connection is disruptively broken or when other network-related error occurred.

Hints

- Connect **inSocket** with the output of TcpIp_Connect or TcpIp_Accept.
- Set **inTimeout** if you want the filter to abort waiting for data after a predefined time.

See Also

- TcpIp_WriteBuffer – Outputs a block of raw data through a connected TCP socket.
- TcpIp_WriteText – Outputs a string through a connected TCP socket.
- TcpIp_ReadLine – Reads from a connected TCP socket until receiving a specific sequence.
- TcpIp_Close – Closes a connected TCP socket gracefully.

TcpIp_ReadBuffer

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Receives a fixed number of bytes from a connected TCP socket.

Syntax

```
void avl::TcpIp_ReadBuffer
(
 avl::SocketId inSocket,
 int inLength,
 const atl::Optional<int>& inTimeout,
 bool inAcceptEof,
 atl::Conditional<avl::ByteBuffer>& outBuffer,
 bool& outEof
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSocket	SocketId			Connected socket ID.
→	inLength	int	1 - 20971520		Number of bytes to receive.
→	inTimeout	const Optional<int>&	0 - ∞	NIL	Timeout in milliseconds, block if not specified.
→	inAcceptEof	bool		False	When false, throws an IoError on attempt to read data from beyond the end of stream; when true interrupts operation without exception in such a case.
←	outBuffer	Conditional<ByteBuffer>&			Buffer with raw received data, or Nil if the operation was interrupted (by timeout or reading beyond the end of stream).
←	outEof	bool&			Indicates whether the operation was interrupted on attempt to get data from beyond the end of stream.

Description

This filter reads a fixed number of bytes from the socket. The number of bytes is specified by the **inLength** input.

The **inTimeout** input specifies the longest possible waiting time, in milliseconds, for the incoming data to be fully received. When no timeout is specified (set to Auto), the execution is blocking. When the filter execution is terminated by timeout condition no data is returned or removed from the input buffer (Nil is returned instead). In such situation the operation can be retried later. When timeout is et to zero the filter performs a single read attempt and returns immediately when no data is available.

outBuffer is a buffer containing raw binary data (a sequence of bytes). Data buffer can be processed using filters from the Binary Data category.

The **outEof** output can be used to detect whether the peer finished transmission and gracefully closed connection on its side. To enable this **inAcceptEof** must be set, otherwise connection closing will be treated as erroneous state.

The point in transmission when the other side closes the connection gracefully is known as the end of stream (or End of File - Eof). All TCP/IP reading filters can normally read data remaining in the input buffer even when the connection has been closed, as long as the whole required chunk of data is present in the input buffer. However, the reaction to the attempt to read from the input buffer beyond the end of the stream depends on the **inAcceptEof** input. When **inAcceptEof** is set to False the filter will raise IO_Error when required chunk of data cannot be read. When **inAcceptEof** is set to True the filter execution is terminated, no data is removed from the input buffer, Nil is returned instead of requested data and the **outEof** output is set to True. In such situation the socket remains valid and subsequent reads will perform similar data read attempts.

This filter can raise IO_Error when the connection is disruptively broken or when other network-related error occurred.

Hints

- Connect **inSocket** with the output of TcpIp_Connect or TcpIp_Accept.
- Set **inLength** to the number of bytes you want to read.
- Set **inTimeout** if you want the filter to abort waiting for data after a predefined time.

See Also

- TcpIp_WriteBuffer – Outputs a block of raw data through a connected TCP socket.
- TcpIp_ReadLine – Reads from a connected TCP socket until receiving a specific sequence.
- TcpIp_ReadAllBuffer – Receives data from a connected socket until the other side closes connection.

TcpIp_ReadLine

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Reads from a connected TCP socket until receiving a specific sequence.

Syntax

```
void avl::TcpIp_ReadLine
(
 avl::SocketId inSocket,
 const atl::String& inDelimiter,
 const atl::Optional<int>& inTimeout,
 bool inAcceptEof,
 avl::DelimiterHandling::Type inMode,
 atl::Conditional<atl::String>& outText,
 bool& outEof
)
```

Parameters

	Name	Type	Range	Default	Description
→	inSocket	SocketId			Connected socket ID.
→	inDelimiter	const String&		\"\\\\r\\\\n\"	Terminating character sequence, escaped.
→	inTimeout	const Optional<int>&	0 - ∞	NIL	Timeout in milliseconds, block if not specified.
→	inAcceptEof	bool		False	When false, throws an IoError on attempt to read data from beyond the end of stream; when true interrupts operation without exception in such a case.
→	inMode	DelimiterHandling::Type			Delimiter handling mode.
←	outText	Conditional<String>&			Received data as textual string, or Nil if the operation was interrupted (by timeout or reading beyond the end of stream).
←	outEof	bool&			Indicates whether the operation was interrupted on attempt to get data from beyond the end of stream.

Description

This filter can read a sequence of characters of variable length from an open socket. The characters are terminated by a sequence, which is specified via **inDelimiter**.

The value on the **inDelimiter** input can contain escape sequences.

An escaped string contains *escape sequences*, which are combinations of the \ (backslash) and another character, which have special meaning. The supported escape sequences are:

- \r - carriage return (ASCII 13)
- \n - new line (ASCII 10)
- \t - tab character (ASCII 9)
- \\ - verbatim backslash \

The **inTimeout** input specifies the longest possible waiting time, in milliseconds, for the incoming data to be fully received. When no timeout is specified (set to Auto), the execution is blocking. When the filter execution is terminated by timeout condition no data is returned or removed from the input buffer (Nil is returned instead). In such situation the operation can be retried later. When timeout is et to zero the filter performs a single read attempt and returns immediately when no data is available.

Received bytes are interpreted as UTF-8 encoded text and returned on **outText** output. When received data is not in a valid UTF-8 encoding IoError is raised.

The **outEof** output can be used to detect whether the peer finished transmission and gracefully closed connection on its side. To enable this **inAcceptEof** must be set, otherwise connection closing will be treated as erroneous state.

The point in transmission when the other side closes the connection gracefully is known as the end of stream (or End of File - Eof). All TCP/IP reading filters can normally read data remaining in the input buffer even when the connection has been closed, as long as the whole required chunk of data is present in the input buffer. However, the reaction to the attempt to read from the input buffer beyond the end of the stream depends on the **inAcceptEof** input. When **inAcceptEof** is set to False the filter will raise IO_Error when required chunk of data cannot be read. When **inAcceptEof** is set to True the filter execution is terminated, no data is removed from the input buffer, Nil is returned instead of requested data and the **outEof** output is set to True. In such situation the socket remains valid and subsequent reads will perform similar data read attempts.

This filter can raise IoError when the connection is disruptively broken or when other network-related error occurred.

Hints

- Connect **inSocket** with the output of TcpIp_Connect or TcpIp_Accept.
- Set **inDelimiter** to define a character sequence that defines end of line. "\r\n" is the typical value of end of line for Windows-based systems.
- Set **inTimeout** if you want the filter to abort waiting for data after a predefined time.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty delimiter in TcpIp_ReadLine

See Also

- TcpIp_WriteText – Outputs a string through a connected TCP socket.
- TcpIp_ReadBuffer – Receives a fixed number of bytes from a connected TCP socket.
- TcpIp_ReadAllText – Receives text from a connected socket until the other side closes connection.

TcpIp_WriteBuffer

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Outputs a block of raw data through a connected TCP socket.

Syntax

```
void avl::TcpIp_WriteBuffer
(
 avl::SocketId inSocket,
 const avl::ByteBuffer& inBuffer
)
```

Parameters

	Name	Type	Default	Description
➡	inSocket	SocketId		Connected socket ID.
➡	inBuffer	const ByteBuffer&		Buffer with data to send.

Description

This filter writes arbitrary bytes to a connected socket.

Data buffer for the **inBuffer** input can be prepared using filters from the Binary Data category.

This filter can raise IoError when the connection is broken or closed by the other side, when the output buffer overflows because the other side is not receiving data or receiving data too slow, or when other network-related error occurred.

Hints

- Connect **inSocket** with the output of TcpIp_Connect or TcpIp_Accept.
- Connect the data you want to send to the **inBuffer** input.

See Also

- TcpIp_ReadLine – Reads from a connected TCP socket until receiving a specific sequence.
- TcpIp_ReadBuffer – Receives a fixed number of bytes from a connected TCP socket.
- TcpIp_WriteText – Outputs a string through a connected TCP socket.

TcpIp_WriteText

Also in AVL Lite

Header: STD.h
Namespace: avl
Module: FoundationLite

Outputs a string through a connected TCP socket.

Syntax

```
void avl::TcpIp_WriteText
(
 avl::SocketId inSocket,
 const atl::String& inText,
 const atl::String& inSuffix
)
```

Parameters

	Name	Type	Default	Description
➡	inSocket	SocketId		Connected socket ID.
➡	inText	const String&		Text to send through socket.
➡	inSuffix	const String&	\"\\\\r\\\\n\"	Additional data to send, like a newline, escaped.

Description

This filter writes textual data to a connected socket.

The text from **inText** is sent through the connected socket, and if the **inSuffix** is given, it is sent after. Either parameter can be the empty string, but when both are empty, no data is sent through the socket. The text is sent using UTF-8 encoding.

The value on the **inSuffix** input can contain escape sequences.

An escaped string contains *escape sequences*, which are combinations of the \ (backslash) and another character, which have special meaning. The supported escape sequences are:

- \r - carriage return (ASCII 13)
- \n - new line (ASCII 10)
- \t - tab character (ASCII 9)
- \\ - verbatim backslash \

This filter can raise ioError when the connection is broken or closed by the other side, when the output buffer overflows because the other side is not receiving data or receiving data too slow, or when other network-related error occurred.

Hints

- Connect **inSocket** with the output of TcpIp_Connect or TcpIp_Accept.
- Connect the data you want to send to the **inText** input.
- Additional suffix defined on **inSuffix** will be added at the end of sent chunk.

See Also

- TcpIp_ReadLine – Reads from a connected TCP socket until receiving a specific sequence.
- TcpIp_ReadBuffer – Receives a fixed number of bytes from a connected TCP socket.
- TcpIp_WriteBuffer – Outputs a block of raw data through a connected TCP socket.

148. Advantech

Table of content:

AdamTCP_Connect

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Connects as a client to a remote Advantech Adam device.

Syntax

```
void avl::AdamTCP_Connect
(
 AdamTCP_State& ioState,
 const atl::String& inDeviceIP,
 const atl::Optional<int>& inTimeout,
 const atl::Optional<int>& inKeepAliveTime
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdamTCP_State&			Object used to maintain state of the function.
	inDeviceIP	const String&			The Advantech Adam device IP address to connect to.
	inTimeout	const Optional<int>&	500 - ∞	NIL	Timeout in milliseconds, block if not specified.
	inKeepAliveTime	const Optional<int>&	2000 - ∞	NIL	When specified activates Tcp/Ip keep alive on new socket with given idle time.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

AdamTCP_Function01

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Function code 01. Reads discrete output's ON/OFF status.

Syntax

```
void avl::AdamTCP_Function01
(
 AdamTCP_State& ioState,
 const atl::Optional<atl::String>& inDeviceIP,
 int inStationAddress,
 int inStartAddress,
 int inNumberOfCoils,
 atl::Array<int>& outValues,
 atl::Array<bool>& outStatuses
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdamTCP_State&			Object used to maintain state of the function.
	inDeviceIP	const Optional<String>&		NIL	The Advantech Adam device IP address.
	inStationAddress	int	0 - ∞	1	
	inStartAddress	int			The address of the first coil.
	inNumberOfCoils	int	1 - 2000		Requested number of bits.
	outValues	Array<int>&			Received values as integers.
	outStatuses	Array<bool>&			Received statuses as bits.

Remarks

You need to specify the IP address of your Adam device in the **inDeviceIP** port only in the first filter which is using the device. In the consecutive filters you can leave **inDeviceIP** set to Auto, these filters will connect to the device with the IP address specified in the first filter.

For a more detailed description of the function from this article and other Adam functions available in Aurora Vision, please refer to the Advantech documentation for your device.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- AdamTCP_Function02 – Function code 02. Reads discrete input's ON/OFF status.
- AdamTCP_Function03 – Function code 03. Reads the binary contents of input registers.
- AdamTCP_Function04 – Function code 04. Reads the binary contents of input registers.
- AdamTCP_Function05 – Function code 05. Forces a single coil to either ON or OFF.

AdamTCP_Function02

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Function code 02. Reads discrete input's ON/OFF status.

Syntax

```
void avl::AdamTCP_Function02
(
 AdamTCP_State& ioState,
 const atl::Optional<atl::String>& inDeviceIP,
 int inStationAddress,
 int inStartAddress,
 int inNumberOfCoils,
 atl::Array<int>& outValues,
 atl::Array<bool>& outStatuses
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	AdamTCP_State&			Object used to maintain state of the function.
→	inDeviceIP	const Optional<String>&		NIL	The Advantech Adam device IP address.
→	inStationAddress	int	0 - ∞	1	
→	inStartAddress	int			The address of the first input.
→	inNumberOfCoils	int	1 - ∞		Requested number of bits.
←	outValues	Array<int>&			Received values as integers.
←	outStatuses	Array<bool>&			Received statuses as bits.

Remarks

You need to specify the IP address of your Adam device in the **inDeviceIP** port only in the first filter which is using the device. In the consecutive filters you can leave **inDeviceIP** set to Auto, these filters will connect to the device with the IP address specified in the first filter.

For a more detailed description of the function from this article and other Adam functions available in Aurora Vision, please refer to the Advantech documentation for your device.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- AdamTCP_Function01 – Function code 01. Reads discrete output's ON/OFF status.
- AdamTCP_Function03 – Function code 03. Reads the binary contents of input registers.
- AdamTCP_Function04 – Function code 04. Reads the binary contents of input registers.
- AdamTCP_Function05 – Function code 05. Forces a single coil to either ON or OFF.

AdamTCP_Function03

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Function code 03. Reads the binary contents of input registers.

Syntax

```
void avl::AdamTCP_Function03
(
 AdamTCP_State& ioState,
 const atl::Optional<atl::String>& inDeviceIP,
 int inStationAddress,
 int inStartAddress,
 int inNumberOfRegister,
 atl::Array<int>& outValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdamTCP_State&			Object used to maintain state of the function.
	inDeviceIP	const Optional<String>&		NIL	The Advantech Adam device IP address.
	inStationAddress	int	0 - ∞	1	
	inStartAddress	int			The address of the first holding register.
	inNumberOfRegister	int	1 - ∞		Number of registers to read.
	outValues	Array<int>&			Received values as integers.

Remarks

You need to specify the IP address of your Adam device in the **inDeviceIP** port only in the first filter which is using the device. In the consecutive filters you can leave **inDeviceIP** set to Auto, these filters will connect to the device with the IP address specified in the first filter.

For a more detailed description of the function from this article and other Adam functions available in Aurora Vision, please refer to the Advantech documentation for your device.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- AdamTCP_Function01 – Function code 01. Reads discrete output's ON/OFF status.
- AdamTCP_Function02 – Function code 02. Reads discrete input's ON/OFF status.
- AdamTCP_Function04 – Function code 04. Reads the binary contents of input registers.
- AdamTCP_Function05 – Function code 05. Forces a single coil to either ON or OFF.

AdamTCP_Function04

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Function code 04. Reads the binary contents of input registers.

Syntax

```
void avl::AdamTCP_Function04
(
 AdamTCP_State& ioState,
 const atl::Optional<atl::String>& inDeviceIP,
 int inStationAddress,
 int inStartAddress,
 int inNumberOfRegister,
 atl::Array<int>& outValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdamTCP_State&			Object used to maintain state of the function.
	inDeviceIP	const Optional<String>&		NIL	The Advantech Adam device IP address.
	inStationAddress	int	0 - ∞	1	
	inStartAddress	int			The address of the first input register.
	inNumberOfRegister	int	1 - ∞		Number of registers to read.
	outValues	Array<int>&			Received values as integers.

Remarks

You need to specify the IP address of your Adam device in the **inDeviceIP** port only in the first filter which is using the device. In the consecutive filters you can leave **inDeviceIP** set to Auto, these filters will connect to the device with the IP address specified in the first filter.

For a more detailed description of the function from this article and other Adam functions available in Aurora Vision, please refer to the Advantech documentation for your device.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- AdamTCP_Function01 – Function code 01. Reads discrete output's ON/OFF status.
- AdamTCP_Function02 – Function code 02. Reads discrete input's ON/OFF status.
- AdamTCP_Function03 – Function code 03. Reads the binary contents of input registers.
- AdamTCP_Function05 – Function code 05. Forces a single coil to either ON or OFF.

AdamTCP_Function05

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Function code 05. Forces a single coil to either ON or OFF.

Syntax

```
void avl::AdamTCP_Function05
(
 AdamTCP_State& ioState,
 const atl::Optional<atl::String>& inDeviceIP,
 int inStationAddress,
 int inCoilAddress,
 bool inState
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdamTCP_State&			Object used to maintain state of the function.
	inDeviceIP	const Optional<String>&		NIL	The Advantech Adam device IP address.
	inStationAddress	int	0 - ∞	1	
	inCoilAddress	int	0 - ∞	16	The address of the coil to be forced.
	inState	bool			Coil state to be set.

Remarks

You need to specify the IP address of your Adam device in the **inDeviceIP** port only in the first filter which is using the device. In the consecutive filters you can leave **inDeviceIP** set to Auto, these filters will connect to the device with the IP address specified in the first filter.

For a more detailed description of the function from this article and other Adam functions available in Aurora Vision, please refer to the Advantech documentation for your device.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- AdamTCP_Function01 – Function code 01. Reads discrete output's ON/OFF status.
- AdamTCP_Function02 – Function code 02. Reads discrete input's ON/OFF status.
- AdamTCP_Function03 – Function code 03. Reads the binary contents of input registers.
- AdamTCP_Function04 – Function code 04. Reads the binary contents of input registers.

DAQNavi_GetDigitalInput

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Returns value from digital input.

Syntax

```
void avl::DAQNavi_GetDigitalInput
(
 DAQNavi_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inPort,
 int& outValue
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	DAQNavi_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Description or index of device
	inPort	int	0 - +∞		Port number
	outValue	int&			Output value

Remarks

Board driver software

This filter is intended to cooperate with digital I/O cards using its vendor SDK. To be able to connect to a card it is required to install DAQNavi SDK software. Currently Aurora Vision Studio requires **DAQNavi SDK version 4.1.3**.

DAQNavi SDK can be downloaded from the following website: https://www.advantech.com.

Device identification

When there is only one device connected to computer, **inDeviceID** field can be set to Auto. In this situation first available device will be found and connected.

inDeviceID can be used to pick one of multiple devices connected to computer. Set this field to Device Description (for example "DemoDevice, BID#000"), or to Device Number(for example "0"). Both parameters should be available in Advantech Navigator.

See Also

- DAQNavi_SetPortDirection – Sets the port direction (input or output).
- DAQNavi_GetDigitalInputs – Returns values from all digital inputs.
- DAQNavi_InitInterrupts – Initializes interrupts.
- DAQNavi_GetDigitalInterrupt – Returns interrupt values.
- DAQNavi_SetDigitalOutput – Sets digital output value.

DAQNavi_GetDigitalInputs

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Returns values from all digital inputs.

Syntax

```
void avl::DAQNavi_GetDigitalInputs
(
 DAQNavi_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 atl::Array<int>& outValues
)
```

Parameters

	Name	Type	Default	Description
	ioState	DAQNavi_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Description or index of device
	outValues	Array<int>&		Output values

Remarks

Board driver software

This filter is intended to cooperate with digital I/O cards using its vendor SDK. To be able to connect to a card it is required to install DAQNavi SDK software. Currently Aurora Vision Studio requires **DAQNavi SDK version 4.1.3**.

DAQNavi SDK can be downloaded from the following website: https://www.advantech.com.

Device identification

When there is only one device connected to computer, **inDeviceID** field can be set to Auto. In this situation first available device will be found and connected.

inDeviceID can be used to pick one of multiple devices connected to computer. Set this field to Device Description (for example "DemoDevice, BID#000"), or to Device Number(for example "0"). Both parameters should be available in Advantech Navigator.

See Also

- DAQNavi_SetPortDirection – Sets the port direction (input or output).
- DAQNavi_GetDigitalInput – Returns value from digital input.
- DAQNavi_InitInterrupts – Initializes interrupts.
- DAQNavi_GetDigitalInterrupt – Returns interrupt values.
- DAQNavi_SetDigitalOutput – Sets digital output value.

DAQNavi_GetDigitalInterrupt

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Returns interrupt values.

Syntax

```
void avl::DAQNavi_GetDigitalInterrupt
(
 DAQNavi_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inSrcChannel,
 int inPort,
 atl::Array<int>& outPortValues
)
```

Parameters

	Name	Type	Range	Default	Description
↻	ioState	DAQNavi_State&			Object used to maintain state of the function.
→	inDeviceID	Optional<const String&>		NIL	Description or index of device
→	inSrcChannel	int	0 - +∞		Channel number which snap event occurs
→	inPort	int	0 - +∞		Port number to scan when interrupt occurred
←	outPortValues	Array<int>&			Port values scanned when interrupt occurred

Remarks

Board driver software

This filter is intended to cooperate with digital I/O cards using its vendor SDK. To be able to connect to a card it is required to install DAQNavi SDK software. Currently Aurora Vision Studio requires **DAQNavi SDK version 4.1.3**.

DAQNavi SDK can be downloaded from the following website: https://www.advantech.com.

Device identification

When there is only one device connected to computer, **inDeviceID** field can be set to Auto. In this situation first available device will be found and connected.

inDeviceID can be used to pick one of multiple devices connected to computer. Set this field to Device Description (for example "DemoDevice, BID#000"), or to Device Number(for example "0"). Both parameters should be available in Advantech Navigator.

See Also

- DAQNavi_SetPortDirection – Sets the port direction (input or output).
- DAQNavi_GetDigitalInput – Returns value from digital input.
- DAQNavi_GetDigitalInputs – Returns values from all digital inputs.
- DAQNavi_InitInterrupts – Initializes interrupts.
- DAQNavi_SetDigitalOutput – Sets digital output value.

DAQNavi_GetDigitalOutput

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets digital output value.

Syntax

```
void avl::DAQNavi_GetDigitalOutput
(
 DAQNavi_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inPort,
 int& outValue
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	DAQNavi_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Description or index of device
	inPort	int	0 - +∞		Port used for outputting data
	outValue	int&			Value to output

DAQNavi_GetDigitalOutputBit

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets digital output bit value.

Syntax

```
void avl::DAQNavi_GetDigitalOutputBit
(
 DAQNavi_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inPort,
 int inBitId,
 bool& outValue
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	DAQNavi_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Description or index of device
	inPort	int	0 - +∞		Port used for outputting data
	inBitId	int	0 - 8		Bit id within the selected port
	outValue	bool&			Value to output

DAQNavi_InitInterrupts

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Initializes interrupts.

Syntax

```
void avl::DAQNavi_InitInterrupts
(
 DAQNavi_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::Array<int>& inChannels,
 avl::DAQNaviTrigger::Type inTriggerMode
)
```

Parameters

	Name	Type	Default	Description
	ioState	DAQNavi_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Description or index of device
	inChannels	const Array<int>&		Interrupted channel numbers
	inTriggerMode	DAQNaviTrigger::Type		Trigger mode

Remarks

Board driver software

This filter is intended to cooperate with digital I/O cards using its vendor SDK. To be able to connect to a card it is required to install DAQNavi SDK software. Currently Aurora Vision Studio requires **DAQNavi SDK version 4.1.3**.

DAQNavi SDK can be downloaded from the following website: https://www.advantech.com.

Device identification

When there is only one device connected to computer, **inDeviceID** field can be set to Auto. In this situation first available device will be found and connected.

inDeviceID can be used to pick one of multiple devices connected to computer. Set this field to Device Description (for example "DemoDevice, BID#000"), or to Device Number(for example "0"). Both parameters should be available in Advantech Navigator.

See Also

- DAQNavi_SetPortDirection – Sets the port direction (input or output).
- DAQNavi_GetDigitalInput – Returns value from digital input.
- DAQNavi_GetDigitalInputs – Returns values from all digital inputs.
- DAQNavi_GetDigitalInterrupt – Returns interrupt values.
- DAQNavi_SetDigitalOutput – Sets digital output value.

DAQNavi_SetDigitalOutput

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets digital output value.

Syntax

```
void avl::DAQNavi_SetDigitalOutput
(
 DAQNavi_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inPort,
 int inValue
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	DAQNavi_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Description or index of device
	inPort	int	0 - +∞		Port used for outputting data
	inValue	int	0 - 255		Value to output

Remarks

Board driver software

This filter is intended to cooperate with digital I/O cards using its vendor SDK. To be able to connect to a card it is required to install DAQNavi SDK software. Currently Aurora Vision Studio requires **DAQNavi SDK version 4.1.3**.

DAQNavi SDK can be downloaded from the following website: https://www.advantech.com.

Device identification

When there is only one device connected to computer, **inDeviceID** field can be set to Auto. In this situation first available device will be found and connected.

inDeviceID can be used to pick one of multiple devices connected to computer. Set this field to Device Description (for example "DemoDevice, BID#000"), or to Device Number(for example "0"). Both parameters should be available in Advantech Navigator.

See Also

- DAQNavi_SetPortDirection – Sets the port direction (input or output).
- DAQNavi_GetDigitalInput – Returns value from digital input.
- DAQNavi_GetDigitalInputs – Returns values from all digital inputs.
- DAQNavi_InitInterrupts – Initializes interrupts.
- DAQNavi_GetDigitalInterrupt – Returns interrupt values.

DAQNavi_SetDigitalOutputBit

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets digital output bit to specified value.

Syntax

```
void avl::DAQNavi_SetDigitalOutputBit
(
 DAQNavi_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inPort,
 int inBitId,
 bool inValue
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	DAQNavi_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Description or index of device
	inPort	int	0 - +∞		Port used for outputting data
	inBitId	int	0 - 8		Bit id within the selected port
	inValue	bool			Value to output

DAQNavi_SetPortDirection

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets the port direction (input or output).

Syntax

```
void avl::DAQNavi_SetPortDirection
(
 DAQNavi_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inPort,
 bool inPortIsOutput
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	DAQNavi_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Description or index of device
	inPort	int	0 - + ∞		Port number
	inPortIsOutput	bool			True if port should be an output, false if port should be an input

Remarks

Board driver software

This filter is intended to cooperate with digital I/O cards using its vendor SDK. To be able to connect to a card it is required to install DAQNavi SDK software. Currently Aurora Vision Studio requires **DAQNavi SDK version 4.1.3**.

DAQNavi SDK can be downloaded from the following website: https://www.advantech.com.

Device identification

When there is only one device connected to computer, **inDeviceID** field can be set to Auto. In this situation first available device will be found and connected.

inDeviceID can be used to pick one of multiple devices connected to computer. Set this field to Device Description (for example "DemoDevice, BID#000"), or to Device Number(for example "0"). Both parameters should be available in Advantech Navigator.

See Also

- DAQNavi_GetDigitalInput – Returns value from digital input.
- DAQNavi_GetDigitalInputs – Returns values from all digital inputs.
- DAQNavi_InitInterrupts – Initializes interrupts.
- DAQNavi_GetDigitalInterrupt – Returns interrupt values.
- DAQNavi_SetDigitalOutput – Sets digital output value.

149. Advantech SUSI

Table of content:

AdvantechSUSI_GetFrequencyI2C

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets frequency of I2C bus.

Syntax

```
void avl::AdvantechSUSI_GetFrequencyI2C
(
 AdvantechSUSI_State& ioState,
 atl::Optional<int> inDevice,
 int& outFrequency
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
	inDevice	Optional<int>	1 - 6	NIL	Select I2C device or empty for main host device.
	outFrequency	int&			

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_GetGPIOPortLevel_Multiple

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets GPIO level.

Syntax

```
void avl::AdvantechSUSI_GetGPIOPortLevel_Multiple
(
 AdvantechSUSI_State& ioState,
 int inBank,
 atl::Optional<const atl::Array<bool>&> inMask,
 atl::Array<bool>& outLevel
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
	inBank	int	0 - ∞		
	inMask	Optional<const Array<bool>&>		NIL	
	outLevel	Array<bool>&			

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_GetGPIOPortLevel_Single

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets GPIO level.

Syntax

```
void avl::AdvantechSUSI_GetGPIOPortLevel_Single
(
 AdvantechSUSI_State& ioState,
 int inPort,
 bool& outLevel
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
	inPort	int	0 - ∞		
	outLevel	bool&			

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_GetThermalProtection

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets thermal protection.

Syntax

```
void avl::AdvantechSUSI_GetThermalProtection
(
 AdvantechSUSI_State& ioState,
 int inProtectionZone,
 avl::AdvantechSUSIBoardTemperatureSensor::Type& outSource,
 avl::AdvantechSUSITemperatureEvent::Type& outEvent,
 float& outEventSend,
 float& outEventClear
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
	inProtectionZone	int	0 - 3		
	outSource	AdvantechSUSIBoardTemperatureSensor::Type&			
	outEvent	AdvantechSUSITemperatureEvent::Type&			
	outEventSend	float&			Unit is Kelvins. When thermal source goes over this value, SUSI will send event according event.
	outEventClear	float&			Unit is Kelvins. When thermal source goes below this value, SUSI will clear event according event.

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_LockStorageArea

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Locks storage area.

Syntax

```
void avl::AdvantechSUSI_LockStorageArea
(
 AdvantechSUSI_State& ioState,
 atl::Optional<int> inStorage,
 const avl::ByteBuffer& inKey
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
→	inStorage	Optional<int>	1 - 11	NIL	Automatic value for standard device.
→	inKey	const ByteBuffer&			Storage password.

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_ReadBlockSMB

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads block from SMBus.

Syntax

```
void avl::AdvantechSUSI_ReadBlockSMB
(
 AdvantechSUSI_State& ioState,
 atl::Optional<int> inDevice,
 int inAddress,
 int inCommand,
 int inReadSize,
 avl::ByteBuffer& outBuffer
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
→	inDevice	Optional<int>	1 - 4	NIL	Select SMBus device or empty for main host device.
→	inAddress	int	0 - 255		
→	inCommand	int	0 - 255		
→	inReadSize	int	1 - ∞		
←	outBuffer	ByteBuffer&			

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_ReadBlockSMBI2C

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads block from SMBus-I2C.

Syntax

```
void avl::AdvantechSUSI_ReadBlockSMBI2C
(
 AdvantechSUSI_State& ioState,
 atl::Optional<int> inDevice,
 int inAddress,
 int inCommand,
 int inReadSize,
 avl::ByteBuffer& outBuffer
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
	inDevice	Optional<int>	1 - 4	NIL	Select SMBus device or empty for main host device.
	inAddress	int	0 - 255		
	inCommand	int	0 - 255		
	inReadSize	int	1 - ∞		
	outBuffer	ByteBuffer&			

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_ReadByteSMB

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads byte from SMBus.

Syntax

```
void avl::AdvantechSUSI_ReadByteSMB
(
 AdvantechSUSI_State& ioState,
 atl::Optional<int> inDevice,
 int inAddress,
 int inCommand,
 int& outData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
	inDevice	Optional<int>	1 - 4	NIL	Select SMBus device or empty for main host device.
	inAddress	int	0 - 255		
	inCommand	int	0 - 255		
	outData	int&			

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_ReadI2C

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads from I2C device.

Syntax

```
void avl::AdvantechSUSI_ReadI2C
(
 AdvantechSUSI_State& ioState,
 atl::Optional<int> inDevice,
 int inAddress,
 bool inLongAddressFormat,
 avl::AdvantechSUSII2CCommandType::Type inCommandType,
 int inCommand,
 int inReadSize,
 avl::ByteBuffer& outBuffer
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
	inDevice	Optional<int>	1 - 6	NIL	Select I2C device or empty for main host device.
	inAddress	int	0 - 1023		Device address as 7 or 10 bit number
	inLongAddressFormat	bool			Activate 10 bit address format.
	inCommandType	AdvantechSUSII2CCommandType::Type			
	inCommand	int			
	inReadSize	int	1 - ∞		
	outBuffer	ByteBuffer&			

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_ReadQuickSMB

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Turns SMBus device function off (on) or disable (enable) a specific device mode.

Syntax

```
void avl::AdvantechSUSI_ReadQuickSMB
(
 AdvantechSUSI_State& ioState,
 atl::Optional<int> inDevice,
 int inAddress
)
```

Parameters

	Name	Type	Range	Default	Description
↩	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
→	inDevice	Optional<int>	1 - 4	NIL	Select SMBus device or empty for main host device.
→	inAddress	int	0 - 255		

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_ReadStorageArea

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads from storage area.

Syntax

```
void avl::AdvantechSUSI_ReadStorageArea
(
 AdvantechSUSI_State& ioState,
 atl::Optional<int> inStorage,
 int inOffset,
 int inReadSize,
 avl::ByteBuffer& outBuffer
)
```

Parameters

	Name	Type	Range	Default	Description
↩	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
→	inStorage	Optional<int>	1 - 11	NIL	
→	inOffset	int	0 - ∞		
→	inReadSize	int	1 - ∞		
←	outBuffer	ByteBuffer&			

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_ReadWordSMB

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads word from SMBus.

Syntax

```
void avl::AdvantechSUSI_ReadWordSMB
(
 AdvantechSUSI_State& ioState,
 atl::Optional<int> inDevice,
 int inAddress,
 int inCommand,
 int& outData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
	inDevice	Optional<int>	1 - 4	NIL	Select SMBus device or empty for main host device.
	inAddress	int	0 - 255		
	inCommand	int	0 - 255		
	outData	int&			

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_ReceiveByteSMB

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Receive byte from SMBus.

Syntax

```
void avl::AdvantechSUSI_ReceiveByteSMB
(
 AdvantechSUSI_State& ioState,
 atl::Optional<int> inDevice,
 int inAddress,
 int& outData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
	inDevice	Optional<int>	1 - 4	NIL	Select SMBus device or empty for main host device.
	inAddress	int	0 - 255		
	outData	int&			

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_SendByteSMB

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Send byte to SMBus.

Syntax

```
void avl::AdvantechSUSI_SendByteSMB
(
 AdvantechSUSI_State& ioState,
 atl::Optional<int> inDevice,
 int inAddress,
 int inData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
	inDevice	Optional<int>	1 - 4	NIL	Select SMBus device or empty for main host device.
	inAddress	int	0 - 255		
	inData	int	0 - 255		

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_SetFrequencyI2C

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets frequency of I2C bus.

Syntax

```
void avl::AdvantechSUSI_SetFrequencyI2C
(
 AdvantechSUSI_State& ioState,
 atl::Optional<int> inDevice,
 int inFrequency
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
	inDevice	Optional<int>	1 - 6	NIL	Select I2C device or empty for main host device.
	inFrequency	int			

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_SetGPIOPortDirection_Multiple

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets GPIO direction.

Syntax

```
void avl::AdvantechSUSI_SetGPIOPortDirection_Multiple
(
 AdvantechSUSI_State& ioState,
 int inBank,
 atl::Optional<const atl::Array<bool>&> inMask,
 atl::Array<avl::AdvantechSUSIPortDirection::Type> inDirection
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
	inBank	int	0 - ∞		
	inMask	Optional<const Array<bool>&>		NIL	
	inDirection	Array<AdvantechSUSIPortDirection::Type>			

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_SetGPIOPortDirection_Single

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets GPIO direction.

Syntax

```
void avl::AdvantechSUSI_SetGPIOPortDirection_Single
(
 AdvantechSUSI_State& ioState,
 int inPort,
 avl::AdvantechSUSIPortDirection::Type inDirection
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
	inPort	int	0 - ∞		
	inDirection	AdvantechSUSIPortDirection::Type			

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_SetGPIOPortLevel_Multiple

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets GPIO level.

Syntax

```
void avl::AdvantechSUSI_SetGPIOPortLevel_Multiple
(
 AdvantechSUSI_State& ioState,
 int inBank,
 atl::Optional<const atl::Array<bool>&> inMask,
 const atl::Array<bool>& inLevel
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
	inBank	int	0 - ∞		
	inMask	Optional<const Array<bool>&>		NIL	
	inLevel	const Array<bool>&			

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_SetGPIOPortLevel_Single

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets GPIO level.

Syntax

```
void avl::AdvantechSUSI_SetGPIOPortLevel_Single
(
 AdvantechSUSI_State& ioState,
 int inPort,
 bool inLevel
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
	inPort	int	0 - ∞		
	inLevel	bool			

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_SetThermalProtection

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets thermal protection.

Syntax

```
void avl::AdvantechSUSI_SetThermalProtection
(
 AdvantechSUSI_State& ioState,
 int inProtectionZone,
 avl::AdvantechSUSIBoardTemperatureSensor::Type inSource,
 avl::AdvantechSUSITemperatureEvent::Type inEvent,
 float inEventSend,
 float inEventClear
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
	inProtectionZone	int	0 - 3		
	inSource	AdvantechSUSIBoardTemperatureSensor::Type			
	inEvent	AdvantechSUSITemperatureEvent::Type			
	inEventSend	float	1.0 - ∞	274.0f	Unit is Kelvins. When thermal source goes over this value, SUSI will send event according event.
	inEventClear	float	1.0 - ∞	274.0f	Unit is Kelvins. When thermal source goes below this value, SUSI will clear event according event.

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_UnlockStorageArea

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Unlocks storage area.

Syntax

```
void avl::AdvantechSUSI_UnlockStorageArea
(
 AdvantechSUSI_State& ioState,
 atl::Optional<int> inStorage,
 const avl::ByteBuffer& inKey
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
	inStorage	Optional<int>	1 - 11	NIL	Automatic value for standard device.
	inKey	const ByteBuffer&			Storage password

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_WatchDogGetLimits

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets hardware watchdog configuration.

Syntax

```
void avl::AdvantechSUSI_WatchDogGetLimits
(
 AdvantechSUSI_State& ioState,
 int inWatchdogTimerId,
 avl::ValueLimits& outDelayTimeLimits,
 avl::ValueLimits& outEventTimeLimits,
 avl::ValueLimits& outResetTimeLimits,
 int& outMinimumUnit
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
	inWatchdogTimerId	int	1 - 3		
	outDelayTimeLimits	ValueLimits&			
	outEventTimeLimits	ValueLimits&			
	outResetTimeLimits	ValueLimits&			
	outMinimumUnit	int&			

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_WatchDogStart

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Starts hardware watchdog timer..

Syntax

```
void avl::AdvantechSUSI_WatchDogStart
(
 AdvantechSUSI_State& ioState,
 int inWatchdogTimerId,
 int inDelayTime,
 int inEventTime,
 int inResetTime,
 avl::AdvantechSUSIWatchdogEvent::Type inEventType
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
	inWatchdogTimerId	int	1 - 3		
	inDelayTime	int	0 - ∞		Initial delay for the watchdog timer in milliseconds.
	inEventTime	int	0 - ∞		Watchdog timeout interval in milliseconds to trigger an event.
	inResetTime	int	0 - ∞		Watchdog timeout interval in milliseconds to trigger a reset.
	inEventType	AdvantechSUSIWatchdogEvent::Type			

Description

Please check minimum unit value with AdvantechSUSI_WatchDogGetLimits. Watchdog values lower than this value will not take effect.

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_WatchDogStop

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Stops hardware watchdog timer.

Syntax

```
void avl::AdvantechSUSI_WatchDogStop
(
 AdvantechSUSI_State& ioState,
 int inWatchdogTimerId
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
	inWatchdogTimerId	int	1 - 3		

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_WatchDogTrigger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Triggers hardware watchdog timer.

Syntax

```
void avl::AdvantechSUSI_WatchDogTrigger
(
 AdvantechSUSI_State& ioState,
 int inWatchdogTimerId
)
```

Parameters

	Name	Type	Default	Description
	ioState	AdvantechSUSI_State&		Object used to maintain state of the function.
	inWatchdogTimerId	int		

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_WriteBlockSMB

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Writes block to SMBus.

Syntax

```
void avl::AdvantechSUSI_WriteBlockSMB
(
 AdvantechSUSI_State& ioState,
 atl::Optional<int> inDevice,
 int inAddress,
 int inCommand,
 const avl::ByteBuffer& inBuffer
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
	inDevice	Optional<int>	1 - 4	NIL	Select SMBus device or empty for main host device.
	inAddress	int	0 - 255		
	inCommand	int	0 - 255		
	inBuffer	const ByteBuffer&			

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_WriteBlockSMBI2C

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Writes block to SMBus-I2C.

Syntax

```
void avl::AdvantechSUSI_WriteBlockSMBI2C
(
 AdvantechSUSI_State& ioState,
 atl::Optional<int> inDevice,
 int inAddress,
 int inCommand,
 const avl::ByteBuffer& inBuffer
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
	inDevice	Optional<int>	1 - 4	NIL	Select SMBus device or empty for main host device.
	inAddress	int	0 - 255		
	inCommand	int	0 - 255		
	inBuffer	const ByteBuffer&			

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_WriteByteSMB

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Writes byte to SMBus.

Syntax

```
void avl::AdvantechSUSI_WriteByteSMB
(
 AdvantechSUSI_State& ioState,
 atl::Optional<int> inDevice,
 int inAddress,
 int inCommand,
 int inData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
	inDevice	Optional<int>	1 - 4	NIL	Select SMBus device or empty for main host device.
	inAddress	int	0 - 255		
	inCommand	int	0 - 255		
	inData	int	0 - 255		

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_WriteI2C

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Writes to I2C device.

Syntax

```
void avl::AdvantechSUSI_WriteI2C
(
 AdvantechSUSI_State& ioState,
 atl::Optional<int> inDevice,
 int inAddress,
 bool inLongAddressFormat,
 avl::AdvantechSUSII2CCommandType::Type inCommandType,
 int inCommand,
 const avl::ByteBuffer& inBuffer
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
	inDevice	Optional<int>	1 - 6	NIL	Select I2C device or empty for main host device.
	inAddress	int	0 - 1023		Device address as 7 or 10 bit number
	inLongAddressFormat	bool			Activate 10 bit address format.
	inCommandType	AdvantechSUSII2CCommandType::Type			
	inCommand	int			
	inBuffer	const ByteBuffer&			

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_WriteQuickSMB

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Turns SMBus device function off (on) or disable (enable) a specific device mode.

Syntax

```
void avl::AdvantechSUSI_WriteQuickSMB
(
 AdvantechSUSI_State& ioState,
 atl::Optional<int> inDevice,
 int inAddress
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
	inDevice	Optional<int>	1 - 4	NIL	Select SMBus device or empty for main host device.
	inAddress	int	0 - 255		

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_WriteReadI2C

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads and writes to I2C bus.

Syntax

```
void avl::AdvantechSUSI_WriteReadI2C
(
 AdvantechSUSI_State& ioState,
 atl::Optional<int> inDevice,
 int inAddress,
 const avl::ByteBuffer& inBuffer,
 int inReadSize,
 avl::ByteBuffer& outBuffer
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
	inDevice	Optional<int>	1 - 6	NIL	Select I2C device or empty for main host device.
	inAddress	int	0 - 127		Device address as 7 bit number.
	inBuffer	const ByteBuffer&			Empty buffer mean no write operation
	inReadSize	int	1 - ∞		Zero size mean no read operation
	outBuffer	ByteBuffer&			

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_WriteStorageArea

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Writes to storage area.

Syntax

```
void avl::AdvantechSUSI_WriteStorageArea
(
 AdvantechSUSI_State& ioState,
 atl::Optional<int> inStorage,
 int inOffset,
 const avl::ByteBuffer& inBuffer
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
	inStorage	Optional<int>	1 - 11	NIL	Automatic value for standard device.
	inOffset	int	0 - ∞		
	inBuffer	const ByteBuffer&			

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

AdvantechSUSI_WriteWordSMB

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Writes word to SMBus.

Syntax

```
void avl::AdvantechSUSI_WriteWordSMB
(
 AdvantechSUSI_State& ioState,
 atl::Optional<int> inDevice,
 int inAddress,
 int inCommand,
 int inData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AdvantechSUSI_State&			Object used to maintain state of the function.
	inDevice	Optional<int>	1 - 4	NIL	Select SMBus device or empty for main host device.
	inAddress	int	0 - 255		
	inCommand	int	0 - 255		
	inData	int	0 - 65535		

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Advantech SUSI SDK.

Add DLL path to system environment variable is required.

Recommended runtime version for Aurora Vision Studio usage is **4**.

Setup low level system functionality e.g. watchdog may still affect the system after the program finished.

150. AvSMART

Table of content:

AvSMART_ConfigureLEDDriverMode

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Set the driver working mode of LED.

Syntax

```
void avl::AvSMART_ConfigureLEDDriverMode
(
 AvSmart_State& ioState,
 avl::RoseekLEDDriverType::Type inType,
 avl::RoseekLEDDriverMode::Type inMode
)
```

Parameters

	Name	Type	Default	Description
	ioState	AvSmart_State&		Object used to maintain state of the function.
	inType	RoseekLEDDriverType::Type		Driver type (own or external)
	inMode	RoseekLEDDriverMode::Type		LED driver mode

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

AvSMART_GenerateSoftwareTrigger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Generates software trigger.

Syntax

```
void avl::AvSMART_GenerateSoftwareTrigger
(
 AvSmart_State& ioState
)
```

Parameters

	Name	Type	Default	Description
	ioState	AvSmart_State&		Object used to maintain state of the function.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

AvSMART_GetInputState

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Get input I/O state.

Syntax

```
void avl::AvSMART_GetInputState
(
 AvSmart_State& ioState,
 int inId,
 bool& outState
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AvSmart_State&			Object used to maintain state of the function.
	inId	int	0 - 7		Index of I/O output port
	outState	bool&			Received state

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

AvSMART_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures images from a AvSMART device.

Syntax

```
bool avl::AvSMART_GrabImage
(
 AvSmart_State& ioState,
 int inInputQueueSize,
 avl::RoseekImageFormat::Type inImageFormat,
 atl::Optional<avl::RoseekResolutionMode::Type> inResolutionMode,
 const atl::Optional<avl::Box>& inROI,
 atl::Optional<float> inFrameRate,
 atl::Optional<avl::RoseekWorkingMode::Type> inWorkingMode,
 atl::Optional<int> inSensitivityLevel,
 atl::Optional<avl::RoseekExposureMode::Type> inExposureMode,
 atl::Optional<int> inExposureTime,
 atl::Optional<float> inGain,
 avl::Image& outImage,
 int& outFrameID
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AvSmart_State&			Object used to maintain state of the function.
	inInputQueueSize	int	1 - 1000	3	Number of incoming frames that can be buffered before the application is able to process them
	inImageFormat	RoseekImageFormat::Type			Image pixel format
	inResolutionMode	Optional<RoseekResolutionMode::Type>		NIL	Set resolution of image
	inROI	const Optional<Box>&		NIL	Set resolution region. Has effect only if resolution mode is ROI.
	inFrameRate	Optional<float>	0.1 - 400.0	NIL	Requested camera frame rate in frames per second
	inWorkingMode	Optional<RoseekWorkingMode::Type>		NIL	Working mode of image acquisition
	inSensitivityLevel	Optional<int>	0 - 3	NIL	Sensitivity level of camera sensor
	inExposureMode	Optional<RoseekExposureMode::Type>		NIL	Exposure mode, should be set to Manual if you want to adjust inExposureTime manually
	inExposureTime	Optional<int>	0 - ∞	NIL	Camera frame exposition time
	inGain	Optional<float>	0.0 - 36.0	NIL	Camera exposure gain
	outImage	Image&			Captured frame
	outFrameID	int&			Captured frame ID

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

AvSMART_GrabImage_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures images from a AvSMART device.

Applications: Use this filter if the trigger may be not coming for some time, while the application should perform other operations in the main loop continuously, or when the timeout situation must be explicitly detected, or when you want to process images from multiple cameras in a single loop and the cameras are sending images asynchronously.

Syntax

```
bool avl::AvSMART_GrabImage_WithTimeout
(
 AvSmart_State& ioState,
 int inTimeout,
 int inInputQueueSize,
 avl::RoseekImageFormat::Type inImageFormat,
 atl::Optional<avl::RoseekResolutionMode::Type> inResolutionMode,
 const atl::Optional<avl::Box>& inROI,
 atl::Optional<float> inFrameRate,
 atl::Optional<avl::RoseekWorkingMode::Type> inWorkingMode,
 atl::Optional<int> inSensitivityLevel,
 atl::Optional<avl::RoseekExposureMode::Type> inExposureMode,
 atl::Optional<int> inExposureTime,
 atl::Optional<float> inGain,
 atl::Conditional<avl::Image>& outImage,
 atl::Conditional<int>& outFrameID
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AvSmart_State&			Object used to maintain state of the function.
	inTimeout	int	10 - 3600000		Maximum time to wait for frame in milliseconds
	inInputQueueSize	int	1 - 1000	3	Number of incoming frames that can be buffered before the application is able to process them
	inImageFormat	RoseekImageFormat::Type			Image pixel format
	inResolutionMode	Optional<RoseekResolutionMode::Type>		NIL	Set resolution of image
	inROI	const Optional<Box>&		NIL	Set resolution region. Has effect only if resolution mode is ROI.
	inFrameRate	Optional<float>	0.1 - 400.0	NIL	Requested camera frame rate in frames per second
	inWorkingMode	Optional<RoseekWorkingMode::Type>		NIL	Working mode of image acquisition
	inSensitivityLevel	Optional<int>	0 - 3	NIL	Sensitivity level of camera sensor
	inExposureMode	Optional<RoseekExposureMode::Type>		NIL	Exposure mode, should be set to Manual if you want to adjust inExposureTime manually
	inExposureTime	Optional<int>	0 - ∞	NIL	Camera frame exposition time
	inGain	Optional<float>	0.0 - 36.0	NIL	Camera exposure gain
	outImage	Conditional<Image>&			Captured frame
	outFrameID	Conditional<int>&			Captured frame ID

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

AvSMART_SetLEDDriverStrength

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Set the driver strength of LED.

Syntax

```
void avl::AvSMART_SetLEDDriverStrength
(
 AvSmart_State& ioState,
 avl::RoseekLEDDriverType::Type inType,
 int inStrength
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AvSmart_State&			Object used to maintain state of the function.
	inType	RoseekLEDDriverType::Type			Driver type (own or external)
	inStrength	int	0 - 1500		Strength. For OWN driver type the max value is 330 otherwise 1500

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

AvSMART_SetOutputState

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Set output I/O state.

Syntax

```
void avl::AvSMART_SetOutputState
(
 AvSmart_State& ioState,
 int inId,
 bool inState
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AvSmart_State&			Object used to maintain state of the function.
	inId	int	0 - 7		Index of I/O output port
	inState	bool			true = TURN_ON, false = TURN_OFF

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

AvSMART_SetStatusLED

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Set the status of programmable LED state indicators supported on the camera.

Syntax

```
void avl::AvSMART_SetStatusLED
(
 AvSmart_State& ioState,
 int inId,
 avl::RoseekLEDStatus::Type inState
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AvSmart_State&			Object used to maintain state of the function.
	inId	int	0 - 2		Index of I/O output port
	inState	RoseekLEDStatus::Type			Received state

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

AvSMART_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Initializes and starts image acquisition.

Syntax

```
void avl::AvSMART_StartAcquisition
(
 AvSmart_State& ioState,
 int inInputQueueSize,
 avl::RoseekImageFormat::Type inImageFormat,
 atl::Optional<avl::RoseekResolutionMode::Type> inResolutionMode,
 const atl::Optional<avl::Box>& inROI,
 atl::Optional<float> inFrameRate,
 atl::Optional<avl::RoseekWorkingMode::Type> inWorkingMode,
 atl::Optional<int> inSensitivityLevel,
 atl::Optional<avl::RoseekExposureMode::Type> inExposureMode,
 atl::Optional<int> inExposureTime,
 atl::Optional<float> inGain
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AvSmart_State&			Object used to maintain state of the function.
	inInputQueueSize	int	1 - 1000	3	Number of incoming frames that can be buffered before the application is able to process them
	inImageFormat	RoseekImageFormat::Type			Image pixel format
	inResolutionMode	Optional<RoseekResolutionMode::Type>		NIL	Set resolution of image
	inROI	const Optional<Box>&		NIL	Set resolution region. Has effect only if resolution mode is ROI.
	inFrameRate	Optional<float>	0.1 - 400.0	NIL	Requested camera frame rate in frames per second
	inWorkingMode	Optional<RoseekWorkingMode::Type>		NIL	Working mode of image acquisition
	inSensitivityLevel	Optional<int>	0 - 3	NIL	Sensitivity level of camera sensor
	inExposureMode	Optional<RoseekExposureMode::Type>		NIL	Exposure mode, should be set to Manual if you want to adjust inExposureTime manually
	inExposureTime	Optional<int>	0 - ∞	NIL	Camera frame exposition time
	inGain	Optional<float>	0.0 - 36.0	NIL	Camera exposure gain

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

AvSMART_StopAcquisition

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Stops image acquisition.

Syntax

```
void avl::AvSMART_StopAcquisition
(
 AvSmart_State& ioState
)
```

Parameters

	Name	Type	Default	Description
	ioState	AvSmart_State&		Object used to maintain state of the function.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

151. AXIS

Table of content:

AXIS_ConfigureDigitalIO

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Configures AXIS digital IO.

Syntax

```
void avl::AXIS_ConfigureDigitalIO
(
 AXIS_State& ioState,
 const atl::String& inAddress,
 int inCameraID,
 atl::Optional<const atl::String&> inLogin,
 atl::Optional<const atl::String&> inPassword,
 int inPort,
 bool inIsInput,
 bool inIsClosed
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AXIS_State&			Object used to maintain state of the function.
	inAddress	const String&			Source device host or ip address
	inCameraID	int	1 - 4		ID of camera
	inLogin	Optional<const String&>		NIL	Login for Basic authorization
	inPassword	Optional<const String&>		NIL	Login for Basic authorization
	inPort	int	0 - ∞		Port number
	inIsInput	bool			Port type. If possible, type will be forced to device
	inIsClosed	bool			Port working mode

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty inAddress in AXIS_ConfigureDigitalIO.

AXIS_GenericRequest

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Send basic GET HTTP request to AXIS device.

Syntax

```
void avl::AXIS_GenericRequest
(
 AXIS_State& ioState,
 const atl::String& inAddress,
 int inCameraID,
 atl::Optional<const atl::String&> inLogin,
 atl::Optional<const atl::String&> inPassword,
 const atl::String& inRequest,
 atl::String& outResponse
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	AXIS_State&			Object used to maintain state of the function.
→	inAddress	const String&			Source device host or ip address
→	inCameraID	int	1 - 4		ID of camera
→	inLogin	Optional<const String&>		NIL	Login for Basic authorization
→	inPassword	Optional<const String&>		NIL	Login for Basic authorization
→	inRequest	const String&			The url part that follows the device address.
←	outResponse	String&			Response limited to 1024 lines.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty inAddress in AXIS_GenericRequest.

AXIS_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a frame using AXIS.

Syntax

```
bool avl::AXIS_GrabImage
(
 AXIS_State& ioState,
 const atl::String& inAddress,
 int inCameraID,
 int inOutputQueueSize,
 atl::Optional<const atl::String&> inLogin,
 atl::Optional<const atl::String&> inPassword,
 atl::Optional<const atl::String&> inStreamProfile,
 atl::Optional<avl::AXISResolution::Type> inResolution,
 atl::Optional<avl::AXISRotation::Type> inRotation,
 atl::Optional<int> inCompression,
 atl::Optional<const atl::String&> inTextString,
 atl::Optional<bool> inUseSquarePixel,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	AXIS_State&			Object used to maintain state of the function.
→	inAddress	const String&			Source device host or ip address
→	inCameraID	int	1 - 4		ID of camera
→	inOutputQueueSize	int	5 - 200		Capacity of output frames queue
→	inLogin	Optional<const String&>		NIL	Login for Basic authorization
→	inPassword	Optional<const String&>		NIL	Login for Basic authorization
→	inStreamProfile	Optional<const String&>		NIL	Profile configuration name.
→	inResolution	Optional<AXISResolution::Type>		NIL	Resolution of the returned image
→	inRotation	Optional<AXISRotation::Type>		NIL	Rotates the image clockwise
→	inCompression	Optional<int>	0 - 100	NIL	Compression level of the image
→	inTextString	Optional<const String&>		NIL	
→	inUseSquarePixel	Optional<bool>		NIL	Square pixel (aspect ratio) correction
←	outImage	Image&			Output image

Remarks

Camera identification

Camera is identified by two inputs:

- inAddress - server name or IP address of camera. It is not a http address using to view image by website. For example:

  ```
  http://my.camera.com/folder/view/viewer_index.shtml?id=217
  ```

  inAddress is

  ```
  http://my.camera.com
  ```

- inCameraID - camera source. Applies only to video servers with more than one video input.

Changing parameters

AXIS Camera support is based on MJPEG streaming, and after changing any parameter the connection must be reset.

The simplest way to reset connection is to stop and restart the program.

Aurora Vision Studio support only basic authentication, please check your device setting.
In order to change authentication method you need to right-click device on the list in AXIS Device Manager and go to Configure Device. There you need to change "Authentication Policy" parameter to "Basic"

HTTPS is not supported.

Configuring the settings in the manufacturer's application may take priority.

New camera models use a global rotation setting.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty inAddress in AXIS_GrabImage.

See Also

- AXIS_GrabImage – Captures a frame using AXIS.
- AXIS_GrabImage_WithTimeout – Captures with timeout a frame using AXIS.
- AXIS_StartAcquisition – Starts acquisition using AXIS.

AXIS_GrabImage_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures with timeout a frame using AXIS.

Syntax

```
bool avl::AXIS_GrabImage_WithTimeout
(
 AXIS_State& ioState,
 const atl::String& inAddress,
 int inCameraID,
 int inOutputQueueSize,
 int inTimeout,
 atl::Optional<const atl::String&> inLogin,
 atl::Optional<const atl::String&> inPassword,
 atl::Optional<const atl::String&> inStreamProfile,
 atl::Optional<avl::AXISResolution::Type> inResolution,
 atl::Optional<avl::AXISRotation::Type> inRotation,
 atl::Optional<int> inCompression,
 atl::Optional<const atl::String&> inTextString,
 atl::Optional<bool> inUseSquarePixel,
 atl::Conditional<avl::Image>& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AXIS_State&			Object used to maintain state of the function.
	inAddress	const String&			Source device host or ip address
	inCameraID	int	1 - 4		ID of camera
	inOutputQueueSize	int	5 - 200		Capacity of output frames queue
	inTimeout	int	1 - ∞	100	Maximum time to wait for frame in milliseconds
	inLogin	Optional<const String&>		NIL	Login for Basic authorization
	inPassword	Optional<const String&>		NIL	Login for Basic authorization
	inStreamProfile	Optional<const String&>		NIL	Profile configuration name.
	inResolution	Optional<AXISResolution::Type>		NIL	Resolution of the returned image
	inRotation	Optional<AXISRotation::Type>		NIL	Rotates the image clockwise
	inCompression	Optional<int>	0 - 100	NIL	Compression level of the image
	inTextString	Optional<const String&>		NIL	
	inUseSquarePixel	Optional<bool>		NIL	Square pixel (aspect ratio) correction
	outImage	Conditional<Image>&			Output image

Remarks

Camera identification

Camera is identified by two inputs:

- inAddress - server name or IP address of camera. It is not a http address using to view image by website. For example:

  ```
  http://my.camera.com/folder/view/viewer_index.shtml?id=217
  ```

  inAddress is

  ```
  http://my.camera.com
  ```

- inCameraID - camera source. Applies only to video servers with more than one video input.

Changing parameters

AXIS Camera support is based on MJPEG streaming, and after changing any parameter the connection must be reset.

The simplest way to reset connection is to stop and restart the program.

Aurora Vision Studio support only basic authentication, please check your device setting.
In order to change authentication method you need to right-click device on the list in AXIS Device Manager and go to Configure Device. There you need to change "Authentication Policy" parameter to "Basic"

HTTPS is not supported.

Configuring the settings in the manufacturer's application may take priority.

New camera models use a global rotation setting.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty inAddress in AXIS_GrabImage_WithTimeout.

See Also

- AXIS_GrabImage – Captures a frame using AXIS.
- AXIS_GrabImage_WithTimeout – Captures with timeout a frame using AXIS.
- AXIS_StartAcquisition – Starts acquisition using AXIS.

AXIS_ReadDigitalPortStatus

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads AXIS digital port.

Syntax

```
void avl::AXIS_ReadDigitalPortStatus
(
 AXIS_State& ioState,
 const atl::String& inAddress,
 int inCameraID,
 atl::Optional<const atl::String&> inLogin,
 atl::Optional<const atl::String&> inPassword,
 int inPort,
 bool& outActive
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AXIS_State&			Object used to maintain state of the function.
	inAddress	const String&			Source device host or ip address
	inCameraID	int	1 - 4		ID of camera
	inLogin	Optional<const String&>		NIL	Login for Basic authorization
	inPassword	Optional<const String&>		NIL	Login for Basic authorization
	inPort	int	0 - ∞		Port number
	outActive	bool&			

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty inAddress in AXIS_ReadDigitalPortStatus.

AXIS_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Starts acquisition using AXIS.

Syntax

```
void avl::AXIS_StartAcquisition
(
 AXIS_State& ioState,
 const atl::String& inAddress,
 int inCameraID,
 int inOutputQueueSize,
 atl::Optional<const atl::String&> inLogin,
 atl::Optional<const atl::String&> inPassword,
 atl::Optional<const atl::String&> inStreamProfile,
 atl::Optional<avl::AXISResolution::Type> inResolution,
 atl::Optional<avl::AXISRotation::Type> inRotation,
 atl::Optional<int> inCompression,
 atl::Optional<const atl::String&> inTextString,
 atl::Optional<bool> inUseSquarePixel
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	AXIS_State&			Object used to maintain state of the function.
	inAddress	const String&			Source device host or ip address
	inCameraID	int	1 - 4		ID of camera
	inOutputQueueSize	int	5 - 200		Capacity of output frames queue
	inLogin	Optional<const String&>		NIL	Login for Basic authorization
	inPassword	Optional<const String&>		NIL	Login for Basic authorization
	inStreamProfile	Optional<const String&>		NIL	Profile configuration name.
	inResolution	Optional<AXISResolution::Type>		NIL	Resolution of the returned image
	inRotation	Optional<AXISRotation::Type>		NIL	Rotates the image clockwise
	inCompression	Optional<int>	0 - 100	NIL	Compression level of the image
	inTextString	Optional<const String&>		NIL	
	inUseSquarePixel	Optional<bool>		NIL	Square pixel (aspect ratio) correction

Remarks

Camera identification

Camera is identified by two inputs:

- inAddress - server name or IP address of camera. It is not a http address using to view image by website. For example:

  ```
  http://my.camera.com/folder/view/viewer_index.shtml?id=217
  ```

  inAddress is

  ```
  http://my.camera.com
  ```

- inCameraID - camera source. Applies only to video servers with more than one video input.

Changing parameters

AXIS Camera support is based on MJPEG streaming, and after changing any parameter the connection must be reset.

The simplest way to reset connection is to stop and restart the program.

Aurora Vision Studio support only basic authentication, please check your device setting.
In order to change authentication method you need to right-click device on the list in AXIS Device Manager and go to Configure Device. There you need to change "Authentication Policy" parameter to "Basic"

HTTPS is not supported.

Configuring the settings in the manufacturer's application may take priority.

New camera models use a global rotation setting.

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty inAddress in AXIS_StartAcquisition.

See Also

- AXIS_GrabImage – Captures a frame using AXIS.
- AXIS_GrabImage_WithTimeout – Captures with timeout a frame using AXIS.
- AXIS_StartAcquisition – Starts acquisition using AXIS.

AXIS_StopAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Stops acquisition using AXIS.

Syntax

```
void avl::AXIS_StopAcquisition
(
 AXIS_State& ioState,
 const atl::String& inAddress,
 int inCameraID
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	AXIS_State&			Object used to maintain state of the function.
→	inAddress	const String&			Source device host or ip address
→	inCameraID	int	1 - 4		ID of camera

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty inAddress in AXIS_StopAcquisition.

AXIS_WriteDigitalPortStatus

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Writes AXIS digital port.

Syntax

```
void avl::AXIS_WriteDigitalPortStatus
(
 AXIS_State& ioState,
 const atl::String& inAddress,
 int inCameraID,
 atl::Optional<const atl::String&> inLogin,
 atl::Optional<const atl::String&> inPassword,
 int inPort,
 bool inActive
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	AXIS_State&			Object used to maintain state of the function.
→	inAddress	const String&			Source device host or ip address
→	inCameraID	int	1 - 4		ID of camera
→	inLogin	Optional<const String&>		NIL	Login for Basic authorization
→	inPassword	Optional<const String&>		NIL	Login for Basic authorization
→	inPort	int	0 - ∞		Port number
→	inActive	bool			

Errors

List of possible exceptions:

Error type	Description
DomainError	Empty inAddress in AXIS_WriteDigitalPortStatus.

152. BitFlow

Table of content:

BitFlow_GetGPOutPin

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets R64 GPOUT pins state.

Syntax

```
void avl::BitFlow_GetGPOutPin
(
 BitFlow_BaseState& ioState,
 int inBoardNumber,
 const atl::File& inCameraFileName,
 int inOutput,
 bool& outState
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	BitFlow_BaseState&			Object used to maintain state of the function.
	inBoardNumber	int			Index of board
	inCameraFileName	const File&			Camera file
	inOutput	int	0 - 6		output
	outState	bool&			pin state

Remarks

Board driver software

This filter is intended to cooperate with board using its vendor SDK. To be able to connect with board it is required to install BitFlow SDK software. Currently Aurora Vision Studio requires **BitFlow SDK version 6.30**.

BitFlow SDK can be downloaded from following website: http://www.bitflow.com (registration may be required).

Running application

This filter uses camera configuration files, which are loaded to register. To load this file to register you should use SysReg application from BitFlow SDK.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- BitFlow_GrabImage – Captures a frame using BitFlow frame grabber.
- BitFlow_GetHardwareTriggerStatus – Gets the status of the hardware trigger.
- BitFlow_SetGPOutPin – Sets R64 GPOUT pins.
- BitFlow_SoftwareTrigger – Performs a software trigger.

BitFlow_GetHardwareTriggerStatus

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets the status of the hardware trigger.

Syntax

```
void avl::BitFlow_GetHardwareTriggerStatus
(
 BitFlow_BaseState& ioState,
 int inBoardNumber,
 const atl::File& inCameraFileName,
 avl::BitFlowTriggers::Type inTrigger,
 bool& outTriggerStatus
)
```

Parameters

	Name	Type	Default	Description
	ioState	BitFlow_BaseState&		Object used to maintain state of the function.
	inBoardNumber	int		Index of board
	inCameraFileName	const File&		Camera file
	inTrigger	BitFlowTriggers::Type		Trigger
	outTriggerStatus	bool&		Status of trigger

Remarks

Board driver software

This filter is intended to cooperate with board using its vendor SDK. To be able to connect with board it is required to install BitFlow SDK software. Currently Aurora Vision Studio requires **BitFlow SDK version 6.30**.

BitFlow SDK can be downloaded from following website: http://www.bitflow.com (registration may be required).

Running application

This filter uses camera configuration files, which are loaded to register. To load this file to register you should use SysReg application from BitFlow SDK.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- BitFlow_GrabImage – Captures a frame using BitFlow frame grabber.
- BitFlow_GetGPOutPin – Gets R64 GPOUT pins state.
- BitFlow_SetGPOutPin – Sets R64 GPOUT pins.
- BitFlow_SoftwareTrigger – Performs a software trigger.

BitFlow_GetRegisterName

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets a register name.

Syntax

```
void avl::BitFlow_GetRegisterName
(
 BitFlow_BaseState& ioState,
 int inBoardNumber,
 const atl::File& inCameraFileName,
 int inRegId,
 atl::String& outName
)
```

Parameters

	Name	Type	Default	Description
	ioState	BitFlow_BaseState&		Object used to maintain state of the function.
	inBoardNumber	int		Index of board
	inCameraFileName	const File&		Camera file
	inRegId	int		Register Id
	outName	String&		Register name

Remarks

Board driver software

This filter is intended to cooperate with board using its vendor SDK. To be able to connect with board it is required to install BitFlow SDK software. Currently Aurora Vision Studio requires **BitFlow SDK version 6.30**.

BitFlow SDK can be downloaded from following website: http://www.bitflow.com (registration may be required).

Running application

This filter uses camera configuration files, which are loaded to register. To load this file to register you should use SysReg application from BitFlow SDK.

Registers management

Register IDs are declared in "BFTabRegister.h" located in Bitflow SDK. See the corresponding Bitflow Hardware Reference Manual for a description of each bit.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- BitFlow_WriteValueToRegister – Writes a value to a register.
- BitFlow_ReadValueFromRegister – Reads a value from a register.

BitFlow_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a frame using BitFlow frame grabber.

Syntax

```
bool avl::BitFlow_GrabImage
(
 BitFlow_BaseState& ioState,
 int inBoardNumber,
 const atl::File& inCameraFileName,
 int inInputQueueSize,
 avl::BitFlowTriggerMode::Type inTriggerMode,
 avl::BitFlowTrigAssignments::Type inTriggerAssignments,
 avl::BitFlowTrigPolarity::Type inTriggerAPolarity,
 avl::BitFlowTrigPolarity::Type inTriggerBPolarity,
 avl::Image& outImage,
 atl::int64& outFrameID
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	BitFlow_BaseState&			Object used to maintain state of the function.
	inBoardNumber	int			Index of board
	inCameraFileName	const File&			Camera file
	inInputQueueSize	int	1 - 200	4	Number of incoming frames that can be buffered before the application is able to process them
	inTriggerMode	BitFlowTriggerMode::Type			Trigger mode of the current camera
	inTriggerAssignments	BitFlowTrigAssignments::Type			Assign trigger to acquisition engine
	inTriggerAPolarity	BitFlowTrigPolarity::Type			Polarity for trigger B
	inTriggerBPolarity	BitFlowTrigPolarity::Type			Polarity for trigger A
	outImage	Image&			Captured frame
	outFrameID	int64&			Captured frame ID

Remarks

Board driver software

This filter is intended to cooperate with board using its vendor SDK. To be able to connect with board it is required to install BitFlow SDK software. Currently Aurora Vision Studio requires **BitFlow SDK version 6.30**.

BitFlow SDK can be downloaded from following website: http://www.bitflow.com (registration may be required).

Running application

This filter uses camera configuration files, which are loaded to register. To load this file to register you should use SysReg application from BitFlow SDK.

Camera identification

To identify the camera, configuration file should be loaded to Aurora Vision Studio, using **inCameraFileName** input. Choosing index of board is too necessary.

Camera file must be located in configuration directory. Default configuration directory should be "C:\BitFlow SDK 5.90\Config". After that, correct directory should be used - it depends on type of board installed in system.

Camera parameters

To change trigger options or digital outputs, use Aurora Vision Studio(see "See also" paragraph). Another parameters might be changed by creating new configuration file. Configuration tool "CamEd" available with BitFlow SDK 5.90 should be used.

Supported pixel formats

- Mono8
- Mono10
- Mono12
- Mono16
- Rgb24
- Rgb30
- Rgb36
- Rgb48

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- BitFlow_GrabImage_WithTimeout – Captures a frame using BitFlow frame grabber.
- BitFlow_GetGPOutPin – Gets R64 GPOUT pins state.
- BitFlow_GetHardwareTriggerStatus – Gets the status of the hardware trigger.
- BitFlow_SetGPOutPin – Sets R64 GPOUT pins.
- BitFlow_SoftwareTrigger – Performs a software trigger.

BitFlow_GrabImage_WithTimeout

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a frame using BitFlow frame grabber.

Applications: Use this filter if the trigger may be not coming for some time, while the application should be performing other operations continuously (e.g. processing HMI events).

Syntax

```
bool avl::BitFlow_GrabImage_WithTimeout
(
 BitFlow_BaseState& ioState,
 int inBoardNumber,
 const atl::File& inCameraFileName,
 int inInputQueueSize,
 int inTimeout,
 avl::BitFlowTriggerMode::Type inTriggerMode,
 avl::BitFlowTrigAssignments::Type inTriggerAssignments,
 avl::BitFlowTrigPolarity::Type inTriggerAPolarity,
 avl::BitFlowTrigPolarity::Type inTriggerBPolarity,
 atl::Conditional<avl::Image>& outImage,
 atl::Conditional<atl::int64>& outFrameID
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	BitFlow_BaseState&			Object used to maintain state of the function.
	inBoardNumber	int			Index of board
	inCameraFileName	const File&			Camera file
	inInputQueueSize	int	1 - 200	4	Number of incoming frames that can be buffered before the application is able to process them
	inTimeout	int	1 - 3600000	5000	Capture timeout
	inTriggerMode	BitFlowTriggerMode::Type			Trigger mode of the current camera
	inTriggerAssignments	BitFlowTrigAssignments::Type			Assign trigger to acquisition engine
	inTriggerAPolarity	BitFlowTrigPolarity::Type			Polarity for trigger B
	inTriggerBPolarity	BitFlowTrigPolarity::Type			Polarity for trigger A
	outImage	Conditional<Image>&			Captured frame
	outFrameID	Conditional<int64>&			Captured frame ID

Remarks

Board driver software

This filter is intended to cooperate with board using its vendor SDK. To be able to connect with board it is required to install BitFlow SDK software. Currently Aurora Vision Studio requires **BitFlow SDK version 6.30**.

BitFlow SDK can be downloaded from following website: http://www.bitflow.com (registration may be required).

Running application

This filter uses camera configuration files, which are loaded to register. To load this file to register you should use SysReg application from BitFlow SDK.

Camera identification

To identify the camera, configuration file should be loaded to Aurora Vision Studio, using **inCameraFileName** input. Choosing index of board is too necessary.

Camera file must be located in configuration directory. Default configuration directory should be "C:\BitFlow SDK 5.90\Config". After that, correct directory should be used - it depends on type of board installed in system.

Camera parameters

To change trigger options or digital outputs, use Aurora Vision Studio(see "See also" paragraph). Another parameters might be changed by creating new configuration file. Configuration tool "CamEd" available with BitFlow SDK 5.90 should be used.

Supported pixel formats

- Mono8
- Mono10
- Mono12
- Mono16
- Rgb24
- Rgb30
- Rgb36
- Rgb48

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- BitFlow_GrabImage – Captures a frame using BitFlow frame grabber.
- BitFlow_GetGPOutPin – Gets R64 GPOUT pins state.
- BitFlow_GetHardwareTriggerStatus – Gets the status of the hardware trigger.
- BitFlow_SetGPOutPin – Sets R64 GPOUT pins.
- BitFlow_SoftwareTrigger – Performs a software trigger.

BitFlow_ReadValueFromRegister

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads a value from a register.

Syntax

```
void avl::BitFlow_ReadValueFromRegister
(
 BitFlow_BaseState& ioState,
 int inBoardNumber,
 const atl::File& inCameraFileName,
 int inRegId,
 int& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	BitFlow_BaseState&		Object used to maintain state of the function.
	inBoardNumber	int		Index of board
	inCameraFileName	const File&		Camera file
	inRegId	int		Register Id
	outValue	int&		Register value

Remarks

Board driver software

This filter is intended to cooperate with board using its vendor SDK. To be able to connect with board it is required to install BitFlow SDK software. Currently Aurora Vision Studio requires **BitFlow SDK version 6.30**.

BitFlow SDK can be downloaded from following website: http://www.bitflow.com (registration may be required).

Running application

This filter uses camera configuration files, which are loaded to register. To load this file to register you should use SysReg application from BitFlow SDK.

Registers management

Register IDs are declared in "BFTabRegister.h" located in Bitflow SDK. See the corresponding Bitflow Hardware Reference Manual for a description of each bit.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- BitFlow_WriteValueToRegister – Writes a value to a register.
- BitFlow_GetRegisterName – Gets a register name.

BitFlow_SetGPOutPin

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets R64 GPOUT pins.

Syntax

```
void avl::BitFlow_SetGPOutPin
(
 BitFlow_BaseState& ioState,
 int inBoardNumber,
 const atl::File& inCameraFileName,
 int inOutput,
 bool inValue
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	BitFlow_BaseState&			Object used to maintain state of the function.
	inBoardNumber	int			Index of board
	inCameraFileName	const File&			Camera file
	inOutput	int	0 - 6		output
	inValue	bool			pin enabled

Remarks

Board driver software

This filter is intended to cooperate with board using its vendor SDK. To be able to connect with board it is required to install BitFlow SDK software. Currently Aurora Vision Studio requires **BitFlow SDK version 6.30**.

BitFlow SDK can be downloaded from following website: http://www.bitflow.com (registration may be required).

Running application

This filter uses camera configuration files, which are loaded to register. To load this file to register you should use SysReg application from BitFlow SDK.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- BitFlow_GrabImage – Captures a frame using BitFlow frame grabber.
- BitFlow_GetHardwareTriggerStatus – Gets the status of the hardware trigger.
- BitFlow_GrabImage – Captures a frame using BitFlow frame grabber.
- BitFlow_SoftwareTrigger – Performs a software trigger.

BitFlow_SoftwareTrigger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Performs a software trigger.

Syntax

```
void avl::BitFlow_SoftwareTrigger
(
 BitFlow_BaseState& ioState,
 int inBoardNumber,
 const atl::File& inCameraFileName,
 avl::BitFlowTriggers::Type inTrigger,
 bool inTriggerEnable
)
```

Parameters

	Name	Type	Default	Description
	ioState	BitFlow_BaseState&		Object used to maintain state of the function.
	inBoardNumber	int		Index of board
	inCameraFileName	const File&		Camera file
	inTrigger	BitFlowTriggers::Type		Trigger to assert
	inTriggerEnable	bool		Trigger enabled / disabled

Remarks

Board driver software

This filter is intended to cooperate with board using its vendor SDK. To be able to connect with board it is required to install BitFlow SDK software. Currently Aurora Vision Studio requires **BitFlow SDK version 6.30**.

BitFlow SDK can be downloaded from following website: http://www.bitflow.com (registration may be required).

Running application

This filter uses camera configuration files, which are loaded to register. To load this file to register you should use SysReg application from BitFlow SDK.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- BitFlow_GrabImage – Captures a frame using BitFlow frame grabber.
- BitFlow_GetHardwareTriggerStatus – Gets the status of the hardware trigger.
- BitFlow_GrabImage – Captures a frame using BitFlow frame grabber.
- BitFlow_SetGPOutPin – Sets R64 GPOUT pins.

BitFlow_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a frame using BitFlow frame grabber.

Syntax

```
void avl::BitFlow_StartAcquisition
(
 BitFlow_BaseState& ioState,
 int inBoardNumber,
 const atl::File& inCameraFileName,
 int inInputQueueSize,
 avl::BitFlowTriggerMode::Type inTriggerMode,
 avl::BitFlowTrigAssignments::Type inTriggerAssignments,
 avl::BitFlowTrigPolarity::Type inTriggerAPolarity,
 avl::BitFlowTrigPolarity::Type inTriggerBPolarity
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	BitFlow_BaseState&			Object used to maintain state of the function.
	inBoardNumber	int			Index of board
	inCameraFileName	const File&			Camera file
	inInputQueueSize	int	1 - 200	4	Number of incoming frames that can be buffered before the application is able to process them
	inTriggerMode	BitFlowTriggerMode::Type			Trigger mode of the current camera
	inTriggerAssignments	BitFlowTrigAssignments::Type			Assign trigger to acquisition engine
	inTriggerAPolarity	BitFlowTrigPolarity::Type			Polarity for trigger B
	inTriggerBPolarity	BitFlowTrigPolarity::Type			Polarity for trigger A

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

BitFlow_StopAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a frame using BitFlow frame grabber.

Syntax

```
void avl::BitFlow_StopAcquisition
(
 BitFlow_BaseState& ioState,
 int inBoardNumber,
 const atl::File& inCameraFileName
)
```

Parameters

	Name	Type	Default	Description
	ioState	BitFlow_BaseState&		Object used to maintain state of the function.
	inBoardNumber	int		Index of board
	inCameraFileName	const File&		Camera file

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

BitFlow_WriteValueToRegister

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Writes a value to a register.

Syntax

```
void avl::BitFlow_WriteValueToRegister
(
 BitFlow_BaseState& ioState,
 int inBoardNumber,
 const atl::File& inCameraFileName,
 int inRegId,
 int inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	BitFlow_BaseState&		Object used to maintain state of the function.
	inBoardNumber	int		Index of board
	inCameraFileName	const File&		Camera file
	inRegId	int		Register Id
	inValue	int		Value to store

Remarks

Board driver software

This filter is intended to cooperate with board using its vendor SDK. To be able to connect with board it is required to install BitFlow SDK software. Currently Aurora Vision Studio requires **BitFlow SDK version 6.30**.

BitFlow SDK can be downloaded from following website: http://www.bitflow.com (registration may be required).

Running application

This filter uses camera configuration files, which are loaded to register. To load this file to register you should use SysReg application from BitFlow SDK.

Registers management

Register IDs are declared in "BFTabRegister.h" located in Bitflow SDK. See the corresponding Bitflow Hardware Reference Manual for a description of each bit.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- BitFlow_ReadValueFromRegister – Reads a value from a register.
- BitFlow_GetRegisterName – Gets a register name.

153. cxCam

Table of content:

cxCam_CalibratePointCloud

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Calibrate data channel to point clod output.

Syntax

```
void avl::cxCam_CalibratePointCloud
(
 cxCam_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const avl::Image& inRangeImage,
 const atl::Optional<atl::File>& inFile,
 const atl::Optional<atl::String>& inCalibrationId,
 atl::Optional<int> inAOI,
 avl::Point3DGrid& outPointCloud
)
```

Parameters

	Name	Type	Default	Description
	ioState	cxCam_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device URI
	inRangeImage	const Image&		Data to calibrate.
	inFile	const Optional<File>&	NIL	Calibration file. Leave blank to use the file from device memory.
	inCalibrationId	const Optional<String>&	NIL	Calibration file id.
	inAOI	Optional<int>	NIL	Number of area of interest that you are going to process. Only used to optimize the reading of calibration data.
	outPointCloud	Point3DGrid&		

Remarks

This filter converts the data from the acquisition (typically the third data channel) with a configuration file to return the 3D real coordinate data. For performance reasons, the file is loaded only once, the first time the filter is called for given parameters.

See Also

- cxCam_GrabData – Captures a frame using cxCam Support Package.
- cxCam_GrabData_WithTimeout – Captures a frame using cxCam Support Package with timeout.

cxCam_CalibrateZMap

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Calibrate data channel to z-map output.

Syntax

```
void avl::cxCam_CalibrateZMap
(
 cxCam_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const avl::Image& inRangeImage,
 const atl::Optional<atl::File>& inFile,
 const atl::Optional<atl::String>& inCalibrationId,
 atl::Optional<int> inAOI,
 avl::Surface& outZMap
)
```

Parameters

	Name	Type	Default	Description
	ioState	cxCam_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device URI
	inRangeImage	const Image&		Data to calibrate.
	inFile	const Optional<File>&	NIL	Calibration file. Leave blank to use the file from device memory.
	inCalibrationId	const Optional<String>&	NIL	Calibration file id.
	inAOI	Optional<int>	NIL	Number of area of interest that you are going to process. Only used to optimize the reading of calibration data.
	outZMap	Surface&		

Remarks

This filter converts the data from the acquisition (typically the third data channel) with a configuration file to return the 3D real coordinate data. For performance reasons, the file is loaded only once, the first time the filter is called for given parameters.

See Also

- cxCam_GrabData – Captures a frame using cxCam Support Package.
- cxCam_GrabData_WithTimeout – Captures a frame using cxCam Support Package with timeout.

cxCam_GetDoubleParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter using cxCam Support Package.

Syntax

```
void avl::cxCam_GetDoubleParameter
(
 cxCam_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 double& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	cxCam_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device URI
	inName	const String&		
	outValue	double&		

cxCam_GetIntegerParameter

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter using cxCam Support Package.

Syntax

```
void avl::cxCam_GetIntegerParameter
(
 cxCam_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 int& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	cxCam_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device URI
	inName	const String&		
	outValue	int&		

cxCam_GetStringParameter

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter using cxCam Support Package.

Syntax

```
void avl::cxCam_GetStringParameter
(
 cxCam_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 atl::String& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	cxCam_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device URI
	inName	const String&		
	outValue	String&		

cxCam_GrabData

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a frame using cxCam Support Package.

Syntax

```
bool avl::cxCam_GrabData
(
 cxCam_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inInputQueueSize,
 const avl::CxCamModeAndAlgorithmConfiguration& inModeAndAlgorithm,
 avl::CxCamImagePixel::Type inPixel,
 avl::CxCamScannerMode::Type inScannerMode,
 avl::CxCamAcquisitionMode::Type inAcquisitionMode,
 atl::Optional<avl::CxCamTriggerMode::Type> inTriggerMode,
 atl::Optional<avl::CxCamTriggerSequencerMode::Type> inSequencerMode,
 atl::Array<avl::CxCamAOIOutput>& outAOI
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	cxCam_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Device URI
	inInputQueueSize	int	3 - 200	4	Capacity of output frames queue
	inModeAndAlgorithm	const CxCamModeAndAlgorithmConfiguration&			3D algorithm configuration.
	inPixel	CxCamImagePixel::Type			Output pixel type.
	inScannerMode	CxCamScannerMode::Type			Scanner mode.
	inAcquisitionMode	CxCamAcquisitionMode::Type			AcquisitionMode.
	inTriggerMode	Optional<CxCamTriggerMode::Type>		NIL	Trigger mode.
	inSequencerMode	Optional<CxCamTriggerSequencerMode::Type>		NIL	Trigger sequencer mode.
	outAOI	Array<CxCamAOIOutput>&			Output data. Usually one item.

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install cxCam Support Package software with camera dedicated drivers.

Package can be downloaded from the following website: https://www.automationtechnology.de/cms/en/support-packages/ (registration may be required).

Add DLL path to system environment variable may be required. The SDK is divided into several directories with shared libraries - you have to add them all.

You should install the transport layer from Common Vision Box Runtime. The lack of this component may cause the application to behave unexpectedly.

Installing special network transport transport layer from vendor may be required.

Recommended cxCam Support Package version for Aurora Vision Studio usage is **01.08.2017**.

32-bit platform is not supported.

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to computer. DeviceID can be set to:

- **URI** - should be specified on device casing.

Camera parameters

Setting **inInputQueueSize** parameter to 'Auto' will select four value as input queue size.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

To change other, more advanced camera parameters, use specific filters.

See Also

- cxCam_GrabData_WithTimeout – Captures a frame using cxCam Support Package with timeout.
- cxCam_StartAcquisition – Initializes and starts image acquisition in a camera.

cxCam_GrabData_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a frame using cxCam Support Package with timeout.

Syntax

```
bool avl::cxCam_GrabData_WithTimeout
(
 cxCam_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inInputQueueSize,
 int inTimeout,
 const avl::CxCamModeAndAlgorithmConfiguration& inModeAndAlgorithm,
 avl::CxCamImagePixel::Type inPixel,
 avl::CxCamScannerMode::Type inScannerMode,
 avl::CxCamAcquisitionMode::Type inAcquisitionMode,
 atl::Optional<avl::CxCamTriggerMode::Type> inTriggerMode,
 atl::Optional<avl::CxCamTriggerSequencerMode::Type> inSequencerMode,
 atl::Optional<atl::Array<avl::CxCamAOIOutput>>& outAOI
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	cxCam_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Device URI.
	inInputQueueSize	int	3 - 200	4	Capacity of output frames queue
	inTimeout	int	1 - ∞	100	Maximum time to wait for frame in milliseconds.
	inModeAndAlgorithm	const CxCamModeAndAlgorithmConfiguration&			3D algorithm configuration.
	inPixel	CxCamImagePixel::Type			Output pixel type.
	inScannerMode	CxCamScannerMode::Type			Scanner mode.
	inAcquisitionMode	CxCamAcquisitionMode::Type			AcquisitionMode.
	inTriggerMode	Optional<CxCamTriggerMode::Type>		NIL	Trigger mode.
	inSequencerMode	Optional<CxCamTriggerSequencerMode::Type>		NIL	Trigger sequencer mode.
	outAOI	Optional<Array<CxCamAOIOutput>>&			Output data. Usually one item.

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outAOI**.

Read more about Optional Outputs.

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install cxCam Support Package software with camera dedicated drivers.

Package can be downloaded from the following website: https://www.automationtechnology.de/cms/en/support-packages/ (registration may be required).

Add DLL path to system environment variable may be required. The SDK is divided into several directories with shared libraries - you have to add them all.

You should install the transport layer from Common Vision Box Runtime. The lack of this component may cause the application to behave unexpectedly.

Installing special network transport transport layer from vendor may be required.

Recommended cxCam Support Package version for Aurora Vision Studio usage is **01.08.2017**.

32-bit platform is not supported.

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to computer. DeviceID can be set to:

- **URI** - should be specified on device casing.

Camera parameters

Setting **inInputQueueSize** parameter to 'Auto' will select four value as input queue size.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

To change other, more advanced camera parameters, use specific filters.

See Also

- cxCam_GrabData – Captures a frame using cxCam Support Package.
- cxCam_StartAcquisition – Initializes and starts image acquisition in a camera.

cxCam_SetDoubleParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter using cxCam Support Package.

Syntax

```
void avl::cxCam_SetDoubleParameter
(
 cxCam_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 double inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	cxCam_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device URI
	inName	const String&		
	inValue	double		

cxCam_SetIntegerParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter using cxCam Support Package.

Syntax

```
void avl::cxCam_SetIntegerParameter
(
 cxCam_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 int inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	cxCam_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device URI
	inName	const String&		
	inValue	int		

cxCam_SetStringParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter using cxCam Support Package.

Syntax

```
void avl::cxCam_SetStringParameter
(
 cxCam_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 const atl::String& inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	cxCam_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device URI
	inName	const String&		
	inValue	const String&		

cxCam_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Initializes and starts image acquisition in a camera.

Syntax

```
void avl::cxCam_StartAcquisition
(
 cxCam_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inInputQueueSize,
 const avl::CxCamModeAndAlgorithmConfiguration& inModeAndAlgorithm,
 avl::CxCamImagePixel::Type inPixel,
 avl::CxCamScannerMode::Type inScannerMode,
 avl::CxCamAcquisitionMode::Type inAcquisitionMode,
 atl::Optional<avl::CxCamTriggerMode::Type> inTriggerMode,
 atl::Optional<avl::CxCamTriggerSequencerMode::Type> inSequencerMode
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	cxCam_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Device URI
	inInputQueueSize	int	3 - 200	4	Capacity of output frames queue
	inModeAndAlgorithm	const CxCamModeAndAlgorithmConfiguration&			3D algorithm configuration.
	inPixel	CxCamImagePixel::Type			Output pixel type.
	inScannerMode	CxCamScannerMode::Type			Scanner mode.
	inAcquisitionMode	CxCamAcquisitionMode::Type			AcquisitionMode.
	inTriggerMode	Optional<CxCamTriggerMode::Type>		NIL	Trigger mode.
	inSequencerMode	Optional<CxCamTriggerSequencerMode::Type>		NIL	Trigger sequencer mode.

Remarks

This filter is intended for establishing connection with a Automation Technology 3D camera device using cxCam Packager interface, to initialize image streaming. It is only needed when explicit image acquisition start is required in the initial phase of a program. For example, it can be used to prepare a camera, running in triggered mode, to be able to capture trigger signals before the first invoke of cxCam_GrabData or to start multiple cameras in sync before the acquisition phase.

The use of this filter is not obligatory. cxCam_GrabData or cxCam_GrabData_WithTimeout filters will initialize and start image acquisition upon their first invoke.

This filter has no effect when invoked for the second time and when invoked after image grabbing filters.

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install cxCam Support Package software with camera dedicated drivers.

Package can be downloaded from the following website: https://www.automationtechnology.de/cms/en/support-packages/ (registration may be required).

Add DLL path to system environment variable may be required. The SDK is divided into several directories with shared libraries - you have to add them all.

You should install the transport layer from Common Vision Box Runtime. The lack of this component may cause the application to behave unexpectedly.

Installing special network transport transport layer from vendor may be required.

Recommended cxCam Support Package version for Aurora Vision Studio usage is **01.08.2017**.

32-bit platform is not supported.

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to computer. DeviceID can be set to:

- **URI** - should be specified on device casing.

Camera parameters

Setting **inInputQueueSize** parameter to 'Auto' will select four value as input queue size.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

To change other, more advanced camera parameters, use specific filters.

See Also

- cxCam_GrabData – Captures a frame using cxCam Support Package.
- cxCam_GrabData_WithTimeout – Captures a frame using cxCam Support Package with timeout.

cxCam_StopAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Initializes and starts image acquisition in a camera.

Syntax

```
void avl::cxCam_StopAcquisition
(
 cxCam_State& ioState,
 atl::Optional<const atl::String&> inDeviceID
)
```

Parameters

	Name	Type	Default	Description
	ioState	cxCam_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device URI

154. Dahua

Table of content:

Dahua_GenerateSoftwareTrigger

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Generates software trigger.

Syntax

```
void avl::Dahua_GenerateSoftwareTrigger
(
 Dahua_State& ioState,
 atl::Optional<const atl::String&> inDeviceID
)
```

Parameters

	Name	Type	Default	Description
	ioState	Dahua_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Dahua_GetBoolParameter

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type Bool.

Syntax

```
void avl::Dahua_GetBoolParameter
(
 Dahua_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameter,
 bool& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Dahua_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inParameter	const String&		Name of parameter
	outValue	bool&		Value retrieved from device

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Dahua_GetDoubleParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type Double.

Syntax

```
void avl::Dahua_GetDoubleParameter
(
 Dahua_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameter,
 double& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Dahua_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inParameter	const String&		Name of parameter
	outValue	double&		Value retrieved from device

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Dahua_GetEnumParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type Enum.

Syntax

```
void avl::Dahua_GetEnumParameter
(
 Dahua_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameter,
 atl::String& outValue,
 atl::Array<atl::String>& outValuesList
)
```

Parameters

	Name	Type	Default	Description
	ioState	Dahua_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inParameter	const String&		Name of parameter
	outValue	String&		Value retrieved from device
	outValuesList	Array<String>&		List of possible values

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Dahua_GetIntParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type Integer.

Syntax

```
void avl::Dahua_GetIntParameter
(
 Dahua_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameter,
 int& outValue
)
```

Parameters

	Name	Type	Default	Description
↔	ioState	Dahua_State&		Object used to maintain state of the function.
→	inDeviceID	Optional<const String&>	NIL	Device identifying number
→	inParameter	const String&		Name of parameter
←	outValue	int&		Value retrieved from device

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Dahua_GetStringParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type String.

Syntax

```
void avl::Dahua_GetStringParameter
(
 Dahua_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameter,
 atl::String& outValue
)
```

Parameters

	Name	Type	Default	Description
↔	ioState	Dahua_State&		Object used to maintain state of the function.
→	inDeviceID	Optional<const String&>	NIL	Device identifying number
→	inParameter	const String&		Name of parameter
←	outValue	String&		Value retrieved from device

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Dahua_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a frame using Dahua.

Syntax

```
bool avl::Dahua_GrabImage
(
 Dahua_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inInputQueueSize,
 atl::Optional<float> inFrameRate,
 atl::Optional<float> inExposureTime,
 atl::Optional<float> inGain,
 atl::Optional<int> inBlackLevel,
 atl::Optional<bool> inTriggerEnabled,
 atl::Optional<avl::DahuaTriggerSource::Type> inTriggerSource,
 atl::Optional<avl::DahuaTriggerActivation::Type> inTriggerActivation,
 avl::Image& outImage,
 atl::int64& outFrameID,
 atl::int64& outTimestamp
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Dahua_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Device identifying number
	inInputQueueSize	int	0 - 200	4	Capacity of output frames queue
	inFrameRate	Optional<float>		NIL	Requested camera frame rate in frames per second
	inExposureTime	Optional<float>		NIL	Camera frame exposition time
	inGain	Optional<float>		NIL	Analog gain of source image in device raw unit
	inBlackLevel	Optional<int>		NIL	Black level of source image
	inTriggerEnabled	Optional<bool>		NIL	Configure trigger enable
	inTriggerSource	Optional<DahuaTriggerSource::Type>		NIL	Source of acquisition trigger
	inTriggerActivation	Optional<DahuaTriggerActivation::Type>		NIL	Circumstances defining when the trigger is activated
	outImage	Image&			Captured frame
	outFrameID	int64&			Captured frame ID
	outTimestamp	int64&			Captured frame timestamp

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install MVViewer software with camera dedicated drivers.

MVViewer can be downloaded from the following website: https://www.dahuasecurity.com/ (registration may be required).

Add DLL path to system environment variable may be required.

Recommended MVViewer version for Aurora Vision Studio usage is **2.1.6**.

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to computer. DeviceID can be set to:

- **Serial number** - should be specified on device casing in format **"vendor:serial number"**.

Camera parameters

Setting **inInputQueueSize** parameter to 'Auto' will select four value as input queue size.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

To change other, more advanced camera parameters, use specific filters.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Dahua_GrabImage_WithTimeout – Captures a frame using Dahua.
- Dahua_StartAcquisition – Initializes and starts image acquisition in a camera.

Dahua_GrabImage_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a frame using Dahua.

Syntax

```
bool avl::Dahua_GrabImage_WithTimeout
(
 Dahua_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inTimeout,
 int inInputQueueSize,
 atl::Optional<float> inFrameRate,
 atl::Optional<float> inExposureTime,
 atl::Optional<float> inGain,
 atl::Optional<int> inBlackLevel,
 atl::Optional<bool> inTriggerEnabled,
 atl::Optional<avl::DahuaTriggerSource::Type> inTriggerSource,
 atl::Optional<avl::DahuaTriggerActivation::Type> inTriggerActivation,
 atl::Conditional < avl::Image>& outImage,
 atl::Conditional < atl::int64>& outFrameID,
 atl::Conditional < atl::int64>& outTimestamp
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Dahua_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Device identifying number
	inTimeout	int	1 - ∞	100	Maximum time to wait for frame in milliseconds
	inInputQueueSize	int	1 - 200	4	Capacity of output frames queue
	inFrameRate	Optional<float>		NIL	Requested camera frame rate in frames per second
	inExposureTime	Optional<float>		NIL	Camera frame exposition time
	inGain	Optional<float>		NIL	Analog gain of source image in device raw unit
	inBlackLevel	Optional<int>		NIL	Black level of source image
	inTriggerEnabled	Optional<bool>		NIL	Configure trigger enable
	inTriggerSource	Optional<DahuaTriggerSource::Type>		NIL	Source of acquisition trigger
	inTriggerActivation	Optional<DahuaTriggerActivation::Type>		NIL	Circumstances defining when the trigger is activated
	outImage	Conditional < Image>&			Captured frame
	outFrameID	Conditional < int64>&			Captured frame ID
	outTimestamp	Conditional < int64>&			Captured frame timestamp

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install MVViewer software with camera dedicated drivers.

MVViewer can be downloaded from the following website: https://www.dahuasecurity.com/ (registration may be required).

Add DLL path to system environment variable may be required.

Recommended MVViewer version for Aurora Vision Studio usage is **2.1.6**.

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to computer. DeviceID can be set to:

- **Serial number** - should be specified on device casing in format **"vendor:serial number"**.

Camera parameters

Setting **inInputQueueSize** parameter to 'Auto' will select four value as input queue size.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

To change other, more advanced camera parameters, use specific filters.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Dahua_GrabImage – Captures a frame using Dahua.
- Dahua_StartAcquisition – Initializes and starts image acquisition in a camera.

Dahua_SetBoolParameter

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type Bool.

Syntax

```
void avl::Dahua_SetBoolParameter
(
 Dahua_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameter,
 bool inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Dahua_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inParameter	const String&		Name of parameter
	inValue	bool		Value to set

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Dahua_SetDoubleParameter

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type Double.

Syntax

```
void avl::Dahua_SetDoubleParameter
(
 Dahua_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameter,
 double inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Dahua_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inParameter	const String&		Name of parameter
	inValue	double		Value to set

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Dahua_SetEnumParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type Enum.

Syntax

```
void avl::Dahua_SetEnumParameter
(
 Dahua_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameter,
 const atl::String& inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Dahua_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inParameter	const String&		Name of parameter
	inValue	const String&		Value to set

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Dahua_SetIntParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type Integer.

Syntax

```
void avl::Dahua_SetIntParameter
(
 Dahua_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameter,
 int inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Dahua_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inParameter	const String&		Name of parameter
	inValue	int		Value to sets

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Dahua_SetStringParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type String.

Syntax

```
void avl::Dahua_SetStringParameter
(
 Dahua_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameter,
 const atl::String& inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Dahua_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inParameter	const String&		Name of parameter
	inValue	const String&		Value to set

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Dahua_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Initializes and starts image acquisition in a camera.

Syntax

```
void avl::Dahua_StartAcquisition
(
 Dahua_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inInputQueueSize,
 atl::Optional<float> inFrameRate,
 atl::Optional<float> inExposureTime,
 atl::Optional<float> inGain,
 atl::Optional<int> inBlackLevel,
 atl::Optional<bool> inTriggerEnabled,
 atl::Optional<avl::DahuaTriggerSource::Type> inTriggerSource,
 atl::Optional<avl::DahuaTriggerActivation::Type> inTriggerActivation
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Dahua_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Device identifying number
	inInputQueueSize	int	1 - 200	4	Capacity of output frames queue
	inFrameRate	Optional<float>		NIL	Requested camera frame rate in frames per second
	inExposureTime	Optional<float>		NIL	Camera frame exposition time
	inGain	Optional<float>		NIL	Analog gain of source image in device raw unit
	inBlackLevel	Optional<int>		NIL	Black level of source image
	inTriggerEnabled	Optional<bool>		NIL	Configure trigger enable
	inTriggerSource	Optional<DahuaTriggerSource::Type>		NIL	Source of acquisition trigger
	inTriggerActivation	Optional<DahuaTriggerActivation::Type>		NIL	Circumstances defining when the trigger is activated

Remarks

This filter is intended for establishing connection with a Dahua camera device using MVViewer interface, to initialize image streaming. It is only needed when explicit image acquisition start is required in the initial phase of a program. For example, it can be used to prepare a camera, running in triggered mode, to be able to capture trigger signals before the first invoke of Dahua_GrabImage or to start multiple cameras in sync before the acquisition phase.

The use of this filter is not obligatory. Dahua_GrabImage or Dahua_GrabImage_WithTimeout] filters will initialize and start image acquisition upon their first invoke.

This filter has no effect when invoked for the second time and when invoked after image grabbing filters.

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install MVViewer software with camera dedicated drivers.

MVViewer can be downloaded from the following website: https://www.dahuasecurity.com/ (registration may be required).

Add DLL path to system environment variable may be required.

Recommended MVViewer version for Aurora Vision Studio usage is **2.1.6**.

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to computer. DeviceID can be set to:

- **Serial number** - should be specified on device casing in format **"vendor:serial number"**.

Camera parameters

Setting **inInputQueueSize** parameter to 'Auto' will select four value as input queue size.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

To change other, more advanced camera parameters, use specific filters.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Dahua_GrabImage – Captures a frame using Dahua.
- Dahua_GrabImage_WithTimeout – Captures a frame using Dahua.

Dahua_StopAcquisition

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Stops image acquisition in a camera.

Syntax

```
void avl::Dahua_StopAcquisition
(
 Dahua_State& ioState,
 atl::Optional<const atl::String&> inDeviceID
)
```

Parameters

	Name	Type	Default	Description
	ioState	Dahua_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

155. National Instruments

Table of content:

DAQmx_ConfigAnalogEdgeTrigger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Returns values from a digital channel.

Syntax

```
void avl::DAQmx_ConfigAnalogEdgeTrigger
(
 DAQmx_ConfigAnalogEdgeTriggerState& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::DAQmxTaskID inTaskID,
 const atl::String& inTriggerSource,
 avl::DAQmxActiveEdge::Type inTriggerSlope,
 float inTriggerLevel,
 atl::Conditional<avl::DAQmxTaskID>& outTaskID
)
```

Parameters

	Name	Type	Default	Description
	ioState	DAQmx_ConfigAnalogEdgeTriggerState&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device alias, product name or a serial number
	inTaskID	DAQmxTaskID		ID of task where triggering is used
	inTriggerSource	const String&		Source of trigger
	inTriggerSlope	DAQmxActiveEdge::Type		Slope of trigger
	inTriggerLevel	float		Threshold of trigger
	outTaskID	Conditional<DAQmxTaskID>&		ID of a created task

Remarks

Device driver software

This filter is intended to cooperate with digital I/O cards using its vendor driver software. To be able to connect to a card it is required to install NI-DAQmx driver software. Currently Aurora Vision Studio requires **NI-DAQmx version 19.5**.

NI-DAQmx driver software can be downloaded from the following website: http://www.ni.com/dataacquisition/nidaqmx.htm.

Device identification

When there is only one device connected to computer, **inDeviceID** field can be set to Auto. In this situation first available device will be found and connected.

inDeviceID can be used to pick one of multiple devices connected to computer. Set this field to Device alias (for example "Dev1"), product name or serial number. This parameters should be available in Measurement & Automation Explorer.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- DAQmx_ConfigDigitEdgeTrigger – Configures a trigger in a specified task.
- DAQmx_ConfigureTiming – Configure timing in specified task.

DAQmx_ConfigDigitEdgeTrigger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Configures a trigger in a specified task.

Syntax

```
void avl::DAQmx_ConfigDigitEdgeTrigger
(
 DAQmx_ConfigDigitEdgeTriggerState& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::DAQmxTaskID inTaskID,
 const atl::String& inTriggerSource,
 avl::DAQmxActiveEdge::Type inTriggerEdge,
 atl::Conditional<avl::DAQmxTaskID>& outTaskID
)
```

Parameters

	Name	Type	Default	Description
	ioState	DAQmx_ConfigDigitEdgeTriggerState&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device alias, product name or a serial number
	inTaskID	DAQmxTaskID		ID of task where triggering is used
	inTriggerSource	const String&		Source of trigger
	inTriggerEdge	DAQmxActiveEdge::Type		Edge of trigger
	outTaskID	Conditional<DAQmxTaskID>&		ID of a created task

Remarks

Device driver software

This filter is intended to cooperate with digital I/O cards using its vendor driver software. To be able to connect to a card it is required to install NI-DAQmx driver software. Currently Aurora Vision Studio requires **NI-DAQmx version 19.5**.

NI-DAQmx driver software can be downloaded from the following website: http://www.ni.com/dataacquisition/nidaqmx.htm.

Device identification

When there is only one device connected to computer, **inDeviceID** field can be set to Auto. In this situation first available device will be found and connected.

inDeviceID can be used to pick one of multiple devices connected to computer. Set this field to Device alias (for example "Dev1"), product name or serial number. This parameters should be available in Measurement & Automation Explorer.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- DAQmx_ConfigAnalogEdgeTrigger – Returns values from a digital channel.
- DAQmx_ConfigureTiming – Configure timing in specified task.

DAQmx_ConfigureTiming

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Configure timing in specified task.

Syntax

```
void avl::DAQmx_ConfigureTiming
(
 DAQmx_ConfigureTimingState& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::DAQmxTaskID inTaskID,
 avl::TimingMode::Type inTimingMode,
 atl::Optional<const atl::String&> inSource,
 float inRate,
 avl::DAQmxActiveEdge::Type inActiveEdge,
 avl::DAQmxSampleMode::Type inSampleMode,
 int inSampsPerChanToAcq,
 atl::Conditional<avl::DAQmxTaskID>& outTaskID
)
```

Parameters

	Name	Type	Default	Description
	ioState	DAQmx_ConfigureTimingState&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device alias, product name or a serial number
	inTaskID	DAQmxTaskID		ID of task where timing is used
	inTimingMode	TimingMode::Type		Mode of the timing
	inSource	Optional<const String&>	NIL	Source terminal of sample clock
	inRate	float		Sampling rate in samples per second
	inActiveEdge	DAQmxActiveEdge::Type		Active edge to generate or acquire samples
	inSampleMode	DAQmxSampleMode::Type		Sampling mode
	inSampsPerChanToAcq	int		Number of samples to acquire or generate for each channel
	outTaskID	Conditional<DAQmxTaskID>&		ID of a created task

Remarks

Device driver software

This filter is intended to cooperate with digital I/O cards using its vendor driver software. To be able to connect to a card it is required to install NI-DAQmx driver software. Currently Aurora Vision Studio requires **NI-DAQmx version 19.5**.

NI-DAQmx driver software can be downloaded from the following website: http://www.ni.com/dataacquisition/nidaqmx.htm.

Device identification

When there is only one device connected to computer, **inDeviceID** field can be set to Auto. In this situation first available device will be found and connected.

inDeviceID can be used to pick one of multiple devices connected to computer. Set this field to Device alias (for example "Dev1"), product name or serial number. This parameters should be available in Measurement & Automation Explorer.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- DAQmx_ConfigDigitEdgeTrigger – Configures a trigger in a specified task.
- DAQmx_ConfigAnalogEdgeTrigger – Returns values from a digital channel.

DAQmx_CreateAnalogChannel

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Creates a task and channel to measure or generate analog values.

Syntax

```
void avl::DAQmx_CreateAnalogChannel
(
 DAQmx_CreateAnalogChannelState& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 atl::Optional<const atl::String&> inChannel,
 avl::ElectricalMode::Type inCVType,
 avl::DirectionMode::Type inIOType,
 avl::DAQmxTerminalConfig::Type inTerminalConfig,
 float inMinValue,
 float inMaxValue,
 avl::DAQmxShuntResistorLoc::Type inShuntResistorLocation,
 atl::Optional<float> inExtShuntResistorValue,
 avl::DAQmxTaskID& outTaskID
)
```

Parameters

	Name	Type	Default	Description
↔	ioState	DAQmx_CreateAnalogChannelState&		Object used to maintain state of the function.
→	inDeviceID	Optional<const String&>	NIL	Device alias, product name or serial number
→	inChannel	Optional<const String&>	NIL	Name of the physical channel used to create a virtual channel
→	inCVType	ElectricalMode::Type		Electrical mode
→	inIOType	DirectionMode::Type		Direction of channel
→	inTerminalConfig	DAQmxTerminalConfig::Type		Input terminal configuration
→	inMinValue	float		Minimum value, that is expected to measure or generate
→	inMaxValue	float	5.0f	Maximum value, that is expected to measure or generate
→	inShuntResistorLocation	DAQmxShuntResistorLoc::Type		Location of the shunt resistor
→	inExtShuntResistorValue	Optional<float>	NIL	Value of the shunt resistor
←	outTaskID	DAQmxTaskID&		ID of a created task

Remarks

Device driver software

This filter is intended to cooperate with digital I/O cards using its vendor driver software. To be able to connect to a card it is required to install NI-DAQmx driver software. Currently Aurora Vision Studio requires **NI-DAQmx version 19.5**.

NI-DAQmx driver software can be downloaded from the following website: http://www.ni.com/dataacquisition/nidaqmx.htm.

Device identification

When there is only one device connected to computer, **inDeviceID** field can be set to Auto. In this situation first available device will be found and connected.

inDeviceID can be used to pick one of multiple devices connected to computer. Set this field to Device alias (for example "Dev1"), product name or serial number. This parameters should be available in Measurement & Automation Explorer.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- DAQmx_CreateDigitalPort – Creates a task and channel to measure or generate digital values.
- DAQmx_CreatePulseChannelFreq – Creates a task and channel to generate pulse.
- DAQmx_CreateCountEdgesChannel – Creates a task and channel to count number of edges.

DAQmx_CreateCountEdgesChannel

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Creates a task and channel to count number of edges.

Syntax

```
void avl::DAQmx_CreateCountEdgesChannel
(
 DAQmx_CreateCountEdgesChannelState& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::Optional<const atl::String&> inCounter,
 avl::DAQmxActiveEdge::Type inActiveEdge,
 int inInitialCount,
 avl::DAQmxCountDirection::Type inCountDirection,
 avl::DAQmxTaskID& outTaskID
)
```

Parameters

	Name	Type	Default	Description
	ioState	DAQmx_CreateCountEdgesChannelState&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device alias, product name or serial number
	inCounter	const Optional<const String&>	NIL	Name of a counter used to create virtual channel
	inActiveEdge	DAQmxActiveEdge::Type		Active edge for counting
	inInitialCount	int		Initial value
	inCountDirection	DAQmxCountDirection::Type		Type of counting (incrementing and decrementing values)
	outTaskID	DAQmxTaskID&		ID of a created task

Remarks

Device driver software

This filter is intended to cooperate with digital I/O cards using its vendor driver software. To be able to connect to a card it is required to install NI-DAQmx driver software. Currently Aurora Vision Studio requires **NI-DAQmx version 19.5**.

NI-DAQmx driver software can be downloaded from the following website: http://www.ni.com/dataacquisition/nidaqmx.htm.

Device identification

When there is only one device connected to computer, **inDeviceID** field can be set to Auto. In this situation first available device will be found and connected.

inDeviceID can be used to pick one of multiple devices connected to computer. Set this field to Device alias (for example "Dev1"), product name or serial number. This parameters should be available in Measurement & Automation Explorer.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- DAQmx_CreateAnalogChannel – Creates a task and channel to measure or generate analog values.
- DAQmx_CreatePulseChannelFreq – Creates a task and channel to generate pulse.
- DAQmx_CreateDigitalPort – Creates a task and channel to measure or generate digital values.

DAQmx_CreateDigitalPort

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Creates a task and channel to measure or generate digital values.

Syntax

```
void avl::DAQmx_CreateDigitalPort
(
 DAQmx_CreateDigitalPortState& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::Optional<const atl::String&> inPort,
 avl::DirectionMode::Type inIOType,
 atl::Optional<int> inFirstLine,
 atl::Optional<int> inLastLine,
 avl::DAQmxTaskID& outTaskID
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	DAQmx_CreateDigitalPortState&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Device alias, product name or serial number
	inPort	const Optional<const String&>		NIL	Name of a physical port used to create a virtual channel
	inIOType	DirectionMode::Type			Direction of a channel
	inFirstLine	Optional<int>	0 - ∞	NIL	First line to measure or generate
	inLastLine	Optional<int>	0 - ∞	NIL	Last line to measure or generate
	outTaskID	DAQmxTaskID&			ID of a created task

Remarks

Device driver software

This filter is intended to cooperate with digital I/O cards using its vendor driver software. To be able to connect to a card it is required to install NI-DAQmx driver software. Currently Aurora Vision Studio requires **NI-DAQmx version 19.5**.

NI-DAQmx driver software can be downloaded from the following website: http://www.ni.com/dataacquisition/nidaqmx.htm.

Device identification

When there is only one device connected to computer, **inDeviceID** field can be set to Auto. In this situation first available device will be found and connected.

inDeviceID can be used to pick one of multiple devices connected to computer. Set this field to Device alias (for example "Dev1"), product name or serial number. This parameters should be available in Measurement & Automation Explorer.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- DAQmx_CreateAnalogChannel – Creates a task and channel to measure or generate analog values.
- DAQmx_CreatePulseChannelFreq – Creates a task and channel to generate pulse.
- DAQmx_CreateCountEdgesChannel – Creates a task and channel to count number of edges.

DAQmx_CreatePulseChannelFreq

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Creates a task and channel to generate pulse.

Syntax

```
void avl::DAQmx_CreatePulseChannelFreq
(
 DAQmx_CreatePulseChannelFreqState& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 atl::Optional<const atl::String&> inCounter,
 avl::DAQmxIdleState::Type inIdleState,
 float inInitDelay,
 float inFrequency,
 float inDutyCycle,
 avl::DAQmxTaskID& outTaskID
)
```

Parameters

	Name	Type	Default	Description
	ioState	DAQmx_CreatePulseChannelFreqState&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device alias, product name or a serial number
	inCounter	Optional<const String&>	NIL	Name of a counter used to create virtual channel
	inIdleState	DAQmxIdleState::Type		Resting state of an output terminal
	inInitDelay	float		Time (in seconds) to wait before generating pulse
	inFrequency	float		Frequency of generated pulse
	inDutyCycle	float		The width of the pulse divided by the pulse period
	outTaskID	DAQmxTaskID&		ID of a created task

Remarks

Device driver software

This filter is intended to cooperate with digital I/O cards using its vendor driver software. To be able to connect to a card it is required to install NI-DAQmx driver software. Currently Aurora Vision Studio requires **NI-DAQmx version 19.5**.

NI-DAQmx driver software can be downloaded from the following website: http://www.ni.com/dataacquisition/nidaqmx.htm.

Device identification

When there is only one device connected to computer, **inDeviceID** field can be set to Auto. In this situation first available device will be found and connected.

inDeviceID can be used to pick one of multiple devices connected to computer. Set this field to Device alias (for example "Dev1"), product name or serial number. This parameters should be available in Measurement & Automation Explorer.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- DAQmx_CreateAnalogChannel – Creates a task and channel to measure or generate analog values.
- DAQmx_CreateDigitalPort – Creates a task and channel to measure or generate digital values.
- DAQmx_CreateCountEdgesChannel – Creates a task and channel to count number of edges.

DAQmx_GetDigitalChannel

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Returns array of values from a digital channels.

Syntax

```
void avl::DAQmx_GetDigitalChannel
(
 DAQmx_GetDigitalChannelState& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::DAQmxTaskID inTaskID,
 atl::Optional<int> inTimeout,
 atl::Array<bool>& outValues,
 atl::Conditional<avl::DAQmxTaskID>& outTaskID
)
```

Parameters

	Name	Type	Default	Description
	ioState	DAQmx_GetDigitalChannelState&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device alias, product name or a serial number
	inTaskID	DAQmxTaskID		ID of task to read sample from
	inTimeout	Optional<int>	NIL	Time (in milliseconds) to wait for the function to get sample
	outValues	Array<bool>&		Digital values read from task
	outTaskID	Conditional<DAQmxTaskID>&		ID of a created task

Remarks

Device driver software

This filter is intended to cooperate with digital I/O cards using its vendor driver software. To be able to connect to a card it is required to install NI-DAQmx driver software. Currently Aurora Vision Studio requires **NI-DAQmx version 19.5**.

NI-DAQmx driver software can be downloaded from the following website: http://www.ni.com/dataacquisition/nidaqmx.htm.

Device identification

When there is only one device connected to computer, **inDeviceID** field can be set to Auto. In this situation first available device will be found and connected.

inDeviceID can be used to pick one of multiple devices connected to computer. Set this field to Device alias (for example "Dev1"), product name or serial number. This parameters should be available in Measurement & Automation Explorer.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- DAQmx_SetDigitalChannel – Sets array of values to a digital channels.

DAQmx_ReadAnalogArray

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads multiple samples from a specified analog task.

Syntax

```
void avl::DAQmx_ReadAnalogArray
(
 DAQmx_ReadAnalogArrayState& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::DAQmxTaskID inTaskID,
 int inSamples,
 atl::Optional<int> inTimeout,
 atl::Array<float>& outValues,
 atl::Conditional<avl::DAQmxTaskID>& outTaskID
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	DAQmx_ReadAnalogArrayState&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Device alias, product name or a serial number
	inTaskID	DAQmxTaskID			ID of task to read samples from
	inSamples	int	1 - ∞		Number of samples to read
	inTimeout	Optional<int>		NIL	Time (in milliseconds) to wait for the function to read samples
	outValues	Array<float>&			Array of read values
	outTaskID	Conditional<DAQmxTaskID>&			ID of a created task

Remarks

Device driver software

This filter is intended to cooperate with digital I/O cards using its vendor driver software. To be able to connect to a card it is required to install NI-DAQmx driver software. Currently Aurora Vision Studio requires **NI-DAQmx version 19.5**.

NI-DAQmx driver software can be downloaded from the following website: http://www.ni.com/dataacquisition/nidaqmx.htm.

Device identification

When there is only one device connected to computer, **inDeviceID** field can be set to Auto. In this situation first available device will be found and connected.

inDeviceID can be used to pick one of multiple devices connected to computer. Set this field to Device alias (for example "Dev1"), product name or serial number. This parameters should be available in Measurement & Automation Explorer.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- DAQmx_ReadAnalogScalar – Reads a sample from a specified analog task.
- DAQmx_WriteAnalogScalar – Sets an analog input to a specified value.
- DAQmx_WriteAnalogArray – Writes multiple samples to a specified analog task.

DAQmx_ReadAnalogScalar

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads a sample from a specified analog task.

Syntax

```
void avl::DAQmx_ReadAnalogScalar
(
 DAQmx_ReadAnalogScalarState& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::DAQmxTaskID inTaskID,
 atl::Optional<int> inTimeout,
 float& outValue,
 atl::Conditional<avl::DAQmxTaskID>& outTaskID
)
```

Parameters

	Name	Type	Default	Description
	ioState	DAQmx_ReadAnalogScalarState&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device alias, product name or a serial number
	inTaskID	DAQmxTaskID		ID of task to read sample from
	inTimeout	Optional<int>	NIL	Time (in milliseconds) to wait for the function to read sample
	outValue	float&		Value read from task
	outTaskID	Conditional<DAQmxTaskID>&		ID of a created task

Remarks

Device driver software

This filter is intended to cooperate with digital I/O cards using its vendor driver software. To be able to connect to a card it is required to install NI-DAQmx driver software. Currently Aurora Vision Studio requires **NI-DAQmx version 19.5**.

NI-DAQmx driver software can be downloaded from the following website: http://www.ni.com/dataacquisition/nidaqmx.htm.

Device identification

When there is only one device connected to computer, **inDeviceID** field can be set to Auto. In this situation first available device will be found and connected.

inDeviceID can be used to pick one of multiple devices connected to computer. Set this field to Device alias (for example "Dev1"), product name or serial number. This parameters should be available in Measurement & Automation Explorer.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- DAQmx_ReadAnalogArray – Reads multiple samples from a specified analog task.
- DAQmx_WriteAnalogScalar – Sets an analog input to a specified value.
- DAQmx_WriteAnalogArray – Writes multiple samples to a specified analog task.

DAQmx_ReadCounterArray

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads multiple samples from a specified counter task.

Syntax

```
void avl::DAQmx_ReadCounterArray
(
 DAQmx_ReadCounterArrayState& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::DAQmxTaskID inTaskID,
 int inSamples,
 atl::Optional<int> inTimeout,
 atl::Array<float>& outValues,
 atl::Conditional<avl::DAQmxTaskID>& outTaskID
)
```

Parameters

	Name	Type	Range	Default	Description
↻	ioState	DAQmx_ReadCounterArrayState&			Object used to maintain state of the function.
→	inDeviceID	Optional<const String&>		NIL	Device alias, product name or a serial number
→	inTaskID	DAQmxTaskID			ID of task to read samples from
→	inSamples	int	1 - ∞		Number of samples to read
→	inTimeout	Optional<int>		NIL	Time (in milliseconds) to wait for the function to read samples
←	outValues	Array<float>&			Array of read values
←	outTaskID	Conditional<DAQmxTaskID>&			ID of a created task

Remarks

Device driver software

This filter is intended to cooperate with digital I/O cards using its vendor driver software. To be able to connect to a card it is required to install NI-DAQmx driver software. Currently Aurora Vision Studio requires **NI-DAQmx version 19.5**.

NI-DAQmx driver software can be downloaded from the following website: http://www.ni.com/dataacquisition/nidaqmx.htm.

Device identification

When there is only one device connected to computer, **inDeviceID** field can be set to Auto. In this situation first available device will be found and connected.

inDeviceID can be used to pick one of multiple devices connected to computer. Set this field to Device alias (for example "Dev1"), product name or serial number. This parameters should be available in Measurement & Automation Explorer.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- DAQmx_ReadAnalogScalar – Reads a sample from a specified analog task.
- DAQmx_WriteAnalogScalar – Sets an analog input to a specified value.
- DAQmx_WriteAnalogArray – Writes multiple samples to a specified analog task.

DAQmx_ReadCounterScalar

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads a sample from a specified counter task.

Syntax

```
void avl::DAQmx_ReadCounterScalar
(
 DAQmx_ReadCounterScalarState& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::DAQmxTaskID inTaskID,
 atl::Optional<int> inTimeout,
 float& outValue,
 atl::Conditional<avl::DAQmxTaskID>& outTaskID
)
```

Parameters

	Name	Type	Default	Description
	ioState	DAQmx_ReadCounterScalarState&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device alias, product name or a serial number
	inTaskID	DAQmxTaskID		ID of task to read sample from
	inTimeout	Optional<int>	NIL	Time (in milliseconds) to wait for the function to read sample
	outValue	float&		Value read from task
	outTaskID	Conditional<DAQmxTaskID>&		ID of a created task

Remarks

Device driver software

This filter is intended to cooperate with digital I/O cards using its vendor driver software. To be able to connect to a card it is required to install NI-DAQmx driver software. Currently Aurora Vision Studio requires **NI-DAQmx version 19.5**.

NI-DAQmx driver software can be downloaded from the following website: http://www.ni.com/dataacquisition/nidaqmx.htm.

Device identification

When there is only one device connected to computer, **inDeviceID** field can be set to Auto. In this situation first available device will be found and connected.

inDeviceID can be used to pick one of multiple devices connected to computer. Set this field to Device alias (for example "Dev1"), product name or serial number. This parameters should be available in Measurement & Automation Explorer.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- DAQmx_ReadAnalogArray – Reads multiple samples from a specified analog task.
- DAQmx_WriteAnalogScalar – Sets an analog input to a specified value.
- DAQmx_WriteAnalogArray – Writes multiple samples to a specified analog task.

DAQmx_ReadDigitalArray

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads multiple samples from a specified digital task.

Syntax

```
void avl::DAQmx_ReadDigitalArray
(
 DAQmx_ReadDigitalArrayState& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::DAQmxTaskID inTaskID,
 int inSamples,
 atl::Optional<int> inTimeout,
 atl::Array<int>& outValues,
 atl::Conditional<avl::DAQmxTaskID>& outTaskID
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	DAQmx_ReadDigitalArrayState&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Device alias, product name or a serial number
	inTaskID	DAQmxTaskID			ID of task to read samples from
	inSamples	int	1 - ∞		Number of samples to read
	inTimeout	Optional<int>		NIL	Time (in milliseconds) to wait for the function to read samples
	outValues	Array<int>&			Array of read values
	outTaskID	Conditional<DAQmxTaskID>&			ID of a created task

Remarks

Device driver software

This filter is intended to cooperate with digital I/O cards using its vendor driver software. To be able to connect to a card it is required to install NI-DAQmx driver software. Currently Aurora Vision Studio requires **NI-DAQmx version 19.5**.

NI-DAQmx driver software can be downloaded from the following website: http://www.ni.com/dataacquisition/nidaqmx.htm.

Device identification

When there is only one device connected to computer, **inDeviceID** field can be set to Auto. In this situation first available device will be found and connected.

inDeviceID can be used to pick one of multiple devices connected to computer. Set this field to Device alias (for example "Dev1"), product name or serial number. This parameters should be available in Measurement & Automation Explorer.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- DAQmx_ReadAnalogScalar – Reads a sample from a specified analog task.
- DAQmx_WriteAnalogScalar – Sets an analog input to a specified value.
- DAQmx_WriteAnalogArray – Writes multiple samples to a specified analog task.

DAQmx_ReadDigitalScalar

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads a sample from a specified digital task.

Syntax

```
void avl::DAQmx_ReadDigitalScalar
(
 DAQmx_ReadDigitalScalarState& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::DAQmxTaskID inTaskID,
 atl::Optional<int> inTimeout,
 int& outValue,
 atl::Conditional<avl::DAQmxTaskID>& outTaskID
)
```

Parameters

	Name	Type	Default	Description
	ioState	DAQmx_ReadDigitalScalarState&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device alias, product name or a serial number
	inTaskID	DAQmxTaskID		ID of task to read sample from
	inTimeout	Optional<int>	NIL	Time (in milliseconds) to wait for the function to read sample
	outValue	int&		Value read from the task
	outTaskID	Conditional<DAQmxTaskID>&		ID of a created task

Remarks

Device driver software

This filter is intended to cooperate with digital I/O cards using its vendor driver software. To be able to connect to a card it is required to install NI-DAQmx driver software. Currently Aurora Vision Studio requires **NI-DAQmx version 19.5**.

NI-DAQmx driver software can be downloaded from the following website: http://www.ni.com/dataacquisition/nidaqmx.htm.

Device identification

When there is only one device connected to computer, **inDeviceID** field can be set to Auto. In this situation first available device will be found and connected.

inDeviceID can be used to pick one of multiple devices connected to computer. Set this field to Device alias (for example "Dev1"), product name or serial number. This parameters should be available in Measurement & Automation Explorer.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- DAQmx_ReadAnalogArray – Reads multiple samples from a specified analog task.
- DAQmx_WriteAnalogScalar – Sets an analog input to a specified value.
- DAQmx_WriteAnalogArray – Writes multiple samples to a specified analog task.

DAQmx_ResetDevice

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Resets a selected device.

Syntax

```
void avl::DAQmx_ResetDevice
(
 DAQmx_ResetDeviceState& ioState,
 const atl::String& inDeviceID
)
```

Parameters

	Name	Type	Default	Description
	ioState	DAQmx_ResetDeviceState&		Object used to maintain state of the function.
	inDeviceID	const String&		Device alias, product name or a serial number

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

DAQmx_SetDigitalChannel

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets array of values to a digital channels.

Syntax

```
void avl::DAQmx_SetDigitalChannel
(
 DAQmx_SetDigitalChannelState& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::DAQmxTaskID inTaskID,
 atl::Optional<int> inTimeout,
 const atl::Array<bool>& inValues,
 atl::Conditional<avl::DAQmxTaskID>& outTaskID
)
```

Parameters

	Name	Type	Default	Description
	ioState	DAQmx_SetDigitalChannelState&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device alias, product name or a serial number
	inTaskID	DAQmxTaskID		ID of task to write sample to
	inTimeout	Optional<int>	NIL	Time (in milliseconds) to wait for the function to set sample
	inValues	const Array<bool>&		Values to write to channel
	outTaskID	Conditional<DAQmxTaskID>&		ID of a created task

Remarks

Device driver software

This filter is intended to cooperate with digital I/O cards using its vendor driver software. To be able to connect to a card it is required to install NI-DAQmx driver software. Currently Aurora Vision Studio requires **NI-DAQmx version 19.5**.

NI-DAQmx driver software can be downloaded from the following website: http://www.ni.com/dataacquisition/nidaqmx.htm.

Device identification

When there is only one device connected to computer, **inDeviceID** field can be set to Auto. In this situation first available device will be found and connected.

inDeviceID can be used to pick one of multiple devices connected to computer. Set this field to Device alias (for example "Dev1"), product name or serial number. This parameters should be available in Measurement & Automation Explorer.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- DAQmx_GetDigitalChannel – Returns array of values from a digital channels.

DAQmx_StartTask

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Starts a specified task.

Syntax

```
void avl::DAQmx_StartTask
(
 DAQmx_StartTaskState& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::DAQmxTaskID inTaskID
)
```

Parameters

	Name	Type	Default	Description
	ioState	DAQmx_StartTaskState&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device alias, product name or serial number
	inTaskID	DAQmxTaskID		ID of a task to start

Remarks

Device driver software

This filter is intended to cooperate with digital I/O cards using its vendor driver software. To be able to connect to a card it is required to install NI-DAQmx driver software. Currently Aurora Vision Studio requires **NI-DAQmx version 19.5**.

NI-DAQmx driver software can be downloaded from the following website: http://www.ni.com/dataacquisition/nidaqmx.htm.

Device identification

When there is only one device connected to computer, **inDeviceID** field can be set to Auto. In this situation first available device will be found and connected.

inDeviceID can be used to pick one of multiple devices connected to computer. Set this field to Device alias (for example "Dev1"), product name or serial number. This parameters should be available in Measurement & Automation Explorer.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- DAQmx_StopTask – Stops and clears a task.

DAQmx_StopTask

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Stops and clears a task.

Syntax

```
void avl::DAQmx_StopTask
(
 DAQmx_StopTaskState& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::DAQmxTaskID inTaskID
)
```

Parameters

	Name	Type	Default	Description
	ioState	DAQmx_StopTaskState&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device alias, product name or serial number
	inTaskID	DAQmxTaskID		ID of a task to stop

Remarks

Device driver software

This filter is intended to cooperate with digital I/O cards using its vendor driver software. To be able to connect to a card it is required to install NI-DAQmx driver software. Currently Aurora Vision Studio requires **NI-DAQmx version 19.5**.

NI-DAQmx driver software can be downloaded from the following website: http://www.ni.com/dataacquisition/nidaqmx.htm.

Device identification

When there is only one device connected to computer, **inDeviceID** field can be set to Auto. In this situation first available device will be found and connected.

inDeviceID can be used to pick one of multiple devices connected to computer. Set this field to Device alias (for example "Dev1"), product name or serial number. This parameters should be available in Measurement & Automation Explorer.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- DAQmx_StartTask – Starts a specified task.

DAQmx_WriteAnalogArray

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Writes multiple samples to a specified analog task.

Syntax

```
void avl::DAQmx_WriteAnalogArray
(
 DAQmx_WriteDigitalArrayState& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::DAQmxTaskID inTaskID,
 atl::Optional<int> inTimeout,
 const atl::Array<float>& inValues,
 atl::Conditional<avl::DAQmxTaskID>& outTaskID
)
```

Parameters

	Name	Type	Default	Description
	ioState	DAQmx_WriteDigitalArrayState&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device alias, product name or a serial number
	inTaskID	DAQmxTaskID		ID of task to write samples to
	inTimeout	Optional<int>	NIL	Time (in milliseconds) to wait for the function to write samples
	inValues	const Array<float>&		Array of samples to write
	outTaskID	Conditional<DAQmxTaskID>&		ID of a created task

Remarks

Device driver software

This filter is intended to cooperate with digital I/O cards using its vendor driver software. To be able to connect to a card it is required to install NI-DAQmx driver software. Currently Aurora Vision Studio requires **NI-DAQmx version 19.5**.

NI-DAQmx driver software can be downloaded from the following website: http://www.ni.com/dataacquisition/nidaqmx.htm.

Device identification

When there is only one device connected to computer, **inDeviceID** field can be set to Auto. In this situation first available device will be found and connected.

inDeviceID can be used to pick one of multiple devices connected to computer. Set this field to Device alias (for example "Dev1"), product name or serial number. This parameters should be available in Measurement & Automation Explorer.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- DAQmx_ReadAnalogScalar – Reads a sample from a specified analog task.
- DAQmx_ReadAnalogArray – Reads multiple samples from a specified analog task.
- DAQmx_WriteAnalogScalar – Sets an analog input to a specified value.

DAQmx_WriteAnalogScalar

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets an analog input to a specified value.

Syntax

```
void avl::DAQmx_WriteAnalogScalar
(
 DAQmx_WriteAnalogScalarState& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::DAQmxTaskID inTaskID,
 atl::Optional<int> inTimeout,
 float inValue,
 atl::Conditional<avl::DAQmxTaskID>& outTaskID
)
```

Parameters

	Name	Type	Default	Description
	ioState	DAQmx_WriteAnalogScalarState&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device alias, product name or a serial number
	inTaskID	DAQmxTaskID		ID of task to write samples to
	inTimeout	Optional<int>	NIL	Time (in milliseconds) to wait for the function to write sample
	inValue	float		Value to write
	outTaskID	Conditional<DAQmxTaskID>&		ID of a created task

Remarks

Device driver software

This filter is intended to cooperate with digital I/O cards using its vendor driver software. To be able to connect to a card it is required to install NI-DAQmx driver software. Currently Aurora Vision Studio requires **NI-DAQmx version 19.5**.

NI-DAQmx driver software can be downloaded from the following website: http://www.ni.com/dataacquisition/nidaqmx.htm.

Device identification

When there is only one device connected to computer, **inDeviceID** field can be set to Auto. In this situation first available device will be found and connected.

inDeviceID can be used to pick one of multiple devices connected to computer. Set this field to Device alias (for example "Dev1"), product name or serial number. This parameters should be available in Measurement & Automation Explorer.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- DAQmx_ReadAnalogArray – Reads multiple samples from a specified analog task.
- DAQmx_ReadAnalogScalar – Reads a sample from a specified analog task.
- DAQmx_WriteAnalogArray – Writes multiple samples to a specified analog task.

DAQmx_WriteDigitalArray

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Writes multiple samples to a specified digital task.

Syntax

```
void avl::DAQmx_WriteDigitalArray
(
 DAQmx_WriteDigitalArrayState& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::DAQmxTaskID inTaskID,
 atl::Optional<int> inTimeout,
 const atl::Array<int>& inValues,
 atl::Conditional<avl::DAQmxTaskID>& outTaskID
)
```

Parameters

	Name	Type	Default	Description
	ioState	DAQmx_WriteDigitalArrayState&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device alias, product name or a serial number
	inTaskID	DAQmxTaskID		ID of task to write samples to
	inTimeout	Optional<int>	NIL	Time (in milliseconds) to wait for the function to write samples
	inValues	const Array<int>&		Array of samples to write
	outTaskID	Conditional<DAQmxTaskID>&		ID of a created task

Remarks

Device driver software

This filter is intended to cooperate with digital I/O cards using its vendor driver software. To be able to connect to a card it is required to install NI-DAQmx driver software. Currently Aurora Vision Studio requires **NI-DAQmx version 19.5**.

NI-DAQmx driver software can be downloaded from the following website: http://www.ni.com/dataacquisition/nidaqmx.htm.

Device identification

When there is only one device connected to computer, **inDeviceID** field can be set to Auto. In this situation first available device will be found and connected.

inDeviceID can be used to pick one of multiple devices connected to computer. Set this field to Device alias (for example "Dev1"), product name or serial number. This parameters should be available in Measurement & Automation Explorer.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- DAQmx_ReadDigitalArray – Reads multiple samples from a specified digital task.
- DAQmx_ReadDigitalScalar – Reads a sample from a specified digital task.
- DAQmx_WriteDigitalArray – Writes multiple samples to a specified digital task.

DAQmx_WriteDigitalScalar

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a digital input to a specified value.

Syntax

```
void avl::DAQmx_WriteDigitalScalar
(
 DAQmx_WriteDigitalScalarState& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::DAQmxTaskID inTaskID,
 atl::Optional<int> inTimeout,
 int inValue,
 atl::Conditional<avl::DAQmxTaskID>& outTaskID
)
```

Parameters

	Name	Type	Default	Description
	ioState	DAQmx_WriteDigitalScalarState&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device alias, product name or a serial number
	inTaskID	DAQmxTaskID		ID of task to write samples to
	inTimeout	Optional<int>	NIL	Time (in milliseconds) to wait for the function to write sample
	inValue	int		Value to write
	outTaskID	Conditional<DAQmxTaskID>&		ID of a created task

Remarks

Device driver software

This filter is intended to cooperate with digital I/O cards using its vendor driver software. To be able to connect to a card it is required to install NI-DAQmx driver software. Currently Aurora Vision Studio requires **NI-DAQmx version 19.5**.

NI-DAQmx driver software can be downloaded from the following website: http://www.ni.com/dataacquisition/nidaqmx.htm.

Device identification

When there is only one device connected to computer, **inDeviceID** field can be set to Auto. In this situation first available device will be found and connected.

inDeviceID can be used to pick one of multiple devices connected to computer. Set this field to Device alias (for example "Dev1"), product name or serial number. This parameters should be available in Measurement & Automation Explorer.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- DAQmx_ReadDigitalScalar – Reads a sample from a specified digital task.
- DAQmx_ReadDigitalArray – Reads multiple samples from a specified digital task.
- DAQmx_WriteDigitalScalar – Sets a digital input to a specified value.

156. Ensenso

Table of content:

Ensenso_GenerateSoftwareTrigger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Triggers software trigger.

Syntax

```
void avl::Ensenso_GenerateSoftwareTrigger
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices

Remarks

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Ensenso_StartAcquisition – Starts capturing data from an Ensenso camera.

Ensenso_GetGlobalParameter_Bool

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Get the value of a bool parameter.

Syntax

```
void avl::Ensenso_GetGlobalParameter_Bool
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::Optional<avl::EnsensoCameraInformation> inDevice,
 const atl::String& inPath,
 bool& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices
	inDevice	const Optional<EnsensoCameraInformation>	NIL	Specific device
	inPath	const String&		Configuration path, relative
	outValue	bool&		

Examples

Configuration tree

for example path

```
Sensor/Temperature
```

in

```
GetGlobalParamerter
```

filter will give read only access to currently opened camera sensor temperature (double)

Configuration paths are relative to root or camera tree.

path

```
Parameters/Capture/PixelClock
```

in

```
GetParamerter
```

filter will give access to camera pixel clock

Remarks

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

Will throw atl::DomainError if the type is not convertible to requested type

Will throw atl::DomainError if the type does not exist or is null

Ensenso_GetGlobalParameter_BoolOrNil

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Get the value of a bool parameter.

Syntax

```
void avl::Ensenso_GetGlobalParameter_BoolOrNil
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::Optional<avl::EnsensoCameraInformation> inDevice,
 const atl::String& inPath,
 atl::Conditional<bool>& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices
	inDevice	const Optional<EnsensoCameraInformation>	NIL	Specific device
	inPath	const String&		Configuration path, relative
	outValue	Conditional<bool>&		

Examples

Configuration tree

for example path

```
Sensor/Temperature
```

in

```
GetGlobalParamerter
```

filter will give read only access to currently opened camera sensor temperature (double)

Configuration paths are relative to root or camera tree.

path

```
Parameters/Capture/PixelClock
```

in

```
GetParamerter
```

filter will give access to camera pixel clock

Remarks

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

Will throw atl::DomainError if the type is not convertible to requested type

Ensenso_GetGlobalParameter_Double

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets the value of a double parameter.

Syntax

```
void avl::Ensenso_GetGlobalParameter_Double
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::String& inPath,
 double& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices
	inPath	const String&		Configuration path, relative
	outValue	double&		

Examples

Configuration tree

for example path

```
Sensor/Temperature
```

in

```
GetGlobalParamerter
```

filter will give read only access to currently opened camera sensor temperature (double)

Configuration paths are relative to root or camera tree.

path

```
Parameters/Capture/PixelClock
```

in

```
GetParamerter
```

filter will give access to camera pixel clock

Remarks

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

Will throw atl::DomainError if the type is not convertible to requested type

Will throw atl::DomainError if the type does not exist or is null

Ensenso_GetGlobalParameter_DoubleOrNil

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets the value of a double parameter.

Syntax

```
void avl::Ensenso_GetGlobalParameter_DoubleOrNil
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::String& inPath,
 atl::Conditional<double>& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices
	inPath	const String&		Configuration path, relative
	outValue	Conditional<double>&		

Examples

Configuration tree

for example path

```
Sensor/Temperature
```

in

```
GetGlobalParamerter
```

filter will give read only access to currently opened camera sensor temperature (double)

Configuration paths are relative to root or camera tree.

path

```
Parameters/Capture/PixelClock
```

in

```
GetParamerter
```

filter will give access to camera pixel clock

Remarks

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

Will throw atl::DomainError if the type is not convertible to requested type

Ensenso_GetGlobalParameter_Int

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Get the value of an int parameter.

Syntax

```
void avl::Ensenso_GetGlobalParameter_Int
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::String& inPath,
 int& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices
	inPath	const String&		Configuration path, relative
	outValue	int&		

Examples

Configuration tree

for example path

```
Sensor/Temperature
```

in

```
GetGlobalParamerter
```

filter will give read only access to currently opened camera sensor temperature (double)

Configuration paths are relative to root or camera tree.

path

```
Parameters/Capture/PixelClock
```

in

```
GetParamerter
```

filter will give access to camera pixel clock

Remarks

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

Will throw atl::DomainError if the type is not convertible to requested type

Will throw atl::DomainError if the type does not exist or is null

Ensenso_GetGlobalParameter_IntOrNil

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets the value of an int parameter.

Syntax

```
void avl::Ensenso_GetGlobalParameter_IntOrNil
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::String& inPath,
 atl::Conditional<int>& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices
	inPath	const String&		Configuration path, relative
	outValue	Conditional<int>&		

Examples

Configuration tree

for example path

```
Sensor/Temperature
```

in

```
GetGlobalParamerter
```

filter will give read only access to currently opened camera sensor temperature (double)

Configuration paths are relative to root or camera tree.

path

```
Parameters/Capture/PixelClock
```

in

```
GetParamerter
```

filter will give access to camera pixel clock

Remarks

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

Will throw atl::DomainError if the type is not convertible to requested type

Ensenso_GetGlobalParameter_String

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets the value of a string parameter.

Syntax

```
void avl::Ensenso_GetGlobalParameter_String
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::String& inPath,
 atl::String& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices
	inPath	const String&		Configuration path, relative
	outValue	String&		

Examples

Configuration tree

for example path

```
Sensor/Temperature
```

in

```
GetGlobalParamerter
```

filter will give read only access to currently opened camera sensor temperature (double)

Configuration paths are relative to root or camera tree.

path

```
Parameters/Capture/PixelClock
```

in

```
GetParamerter
```

filter will give access to camera pixel clock

Remarks

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

Will throw atl::DomainError if the type is not convertible to requested type

Will throw atl::DomainError if the type does not exist or is null

Ensenso_GetGlobalParameter_StringOrNil

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets the value of a string parameter.

Syntax

```
void avl::Ensenso_GetGlobalParameter_StringOrNil
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::String& inPath,
 atl::Conditional<atl::String>& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices
	inPath	const String&		Configuration path, relative
	outValue	Conditional<String>&		

Examples

Configuration tree

for example path

```
Sensor/Temperature
```

in

```
GetGlobalParamerter
```

filter will give read only access to currently opened camera sensor temperature (double)

Configuration paths are relative to root or camera tree.

path

```
Parameters/Capture/PixelClock
```

in

```
GetParamerter
```

filter will give access to camera pixel clock

Remarks

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

Will throw atl::DomainError if the type is not convertible to requested type

Ensenso_GetParameter_Bool

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets the value of a bool parameter.s

Syntax

```
void avl::Ensenso_GetParameter_Bool
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::Optional<avl::EnsensoCameraInformation> inDevice,
 const atl::String& inPath,
 bool& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices
	inDevice	const Optional<EnsensoCameraInformation>	NIL	Specific device
	inPath	const String&		Configuration path, relative
	outValue	bool&		

Examples

Configuration tree

for example path

```
Sensor/Temperature
```

in

```
GetGlobalParamerter
```

filter will give read only access to currently opened camera sensor temperature (double)

Configuration paths are relative to root or camera tree.

path

```
Parameters/Capture/PixelClock
```

in

```
GetParamerter
```

filter will give access to camera pixel clock

Remarks

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

Will throw atl::DomainError if the type is not convertible to requested type

Will throw atl::DomainError if the type does not exist or is null

Ensenso_GetParameter_BoolOrNil

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets the value of a bool parameter.s

Syntax

```
void avl::Ensenso_GetParameter_BoolOrNil
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::Optional<avl::EnsensoCameraInformation> inDevice,
 const atl::String& inPath,
 atl::Conditional<bool>& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices
	inDevice	const Optional<EnsensoCameraInformation>	NIL	Specific device
	inPath	const String&		Configuration path, relative
	outValue	Conditional<bool>&		

Examples

Configuration tree

for example path

```
Sensor/Temperature
```

in

```
GetGlobalParamerter
```

filter will give read only access to currently opened camera sensor temperature (double)

Configuration paths are relative to root or camera tree.

path

```
Parameters/Capture/PixelClock
```

in

```
GetParamerter
```

filter will give access to camera pixel clock

Remarks

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

Will throw atl::DomainError if the type is not convertible to requested type

Ensenso_GetParameter_Double

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets the value of a double parameter.

Syntax

```
void avl::Ensenso_GetParameter_Double
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::Optional<avl::EnsensoCameraInformation> inDevice,
 const atl::String& inPath,
 double& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices
	inDevice	const Optional<EnsensoCameraInformation>	NIL	Specific device
	inPath	const String&		Configuration path, relative
	outValue	double&		

Examples

Configuration tree

for example path

```
Sensor/Temperature
```

in

```
GetGlobalParamerter
```

filter will give read only access to currently opened camera sensor temperature (double)

Configuration paths are relative to root or camera tree.

path

```
Parameters/Capture/PixelClock
```

in

```
GetParamerter
```

filter will give access to camera pixel clock

Remarks

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

Will throw atl::DomainError if the type is not convertible to requested type

Will throw atl::DomainError if the type does not exist or is null

Ensenso_GetParameter_DoubleOrNil

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets the value of a double parameter.

Syntax

```
void avl::Ensenso_GetParameter_DoubleOrNil
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::Optional<avl::EnsensoCameraInformation> inDevice,
 const atl::String& inPath,
 atl::Conditional<double>& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices
	inDevice	const Optional<EnsensoCameraInformation>	NIL	Specific device
	inPath	const String&		Configuration path, relative
	outValue	Conditional<double>&		

Examples

Configuration tree

for example path

```
Sensor/Temperature
```

in

```
GetGlobalParamerter
```

filter will give read only access to currently opened camera sensor temperature (double)

Configuration paths are relative to root or camera tree.

path

```
Parameters/Capture/PixelClock
```

in

```
GetParamerter
```

filter will give access to camera pixel clock

Remarks

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

Will throw atl::DomainError if the type is not convertible to requested type

Ensenso_GetParameter_Int

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets the value of an int parameter.

Syntax

```
void avl::Ensenso_GetParameter_Int
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::Optional<avl::EnsensoCameraInformation> inDevice,
 const atl::String& inPath,
 int& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices
	inDevice	const Optional<EnsensoCameraInformation>	NIL	Specific device
	inPath	const String&		Configuration path, relative
	outValue	int&		

Examples

Configuration tree

for example path

```
Sensor/Temperature
```

in

```
GetGlobalParamerter
```

filter will give read only access to currently opened camera sensor temperature (double)

Configuration paths are relative to root or camera tree.

path

```
Parameters/Capture/PixelClock
```

in

```
GetParamerter
```

filter will give access to camera pixel clock

Remarks

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

Will throw atl::DomainError if the type is not convertible to requested type

Will throw atl::DomainError if the type does not exist or is null

Ensenso_GetParameter_IntOrNil

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets the value of an int parameter.

Syntax

```
void avl::Ensenso_GetParameter_IntOrNil
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::Optional<avl::EnsensoCameraInformation> inDevice,
 const atl::String& inPath,
 atl::Conditional<int>& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices
	inDevice	const Optional<EnsensoCameraInformation>	NIL	Specific device
	inPath	const String&		Configuration path, relative
	outValue	Conditional<int>&		

Examples

Configuration tree

for example path

```
Sensor/Temperature
```

in

```
GetGlobalParamerter
```

filter will give read only access to currently opened camera sensor temperature (double)

Configuration paths are relative to root or camera tree.

path

```
Parameters/Capture/PixelClock
```

in

```
GetParamerter
```

filter will give access to camera pixel clock

Remarks

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

Will throw atl::DomainError if the type is not convertible to requested type

Ensenso_GetParameter_String

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets the value of a string parameter.

Syntax

```
void avl::Ensenso_GetParameter_String
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::Optional<avl::EnsensoCameraInformation> inDevice,
 const atl::String& inPath,
 atl::String& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices
	inDevice	const Optional<EnsensoCameraInformation>	NIL	Specific device
	inPath	const String&		Configuration path, relative
	outValue	String&		

Examples

Configuration tree

for example path

```
Sensor/Temperature
```

in

```
GetGlobalParamerter
```

filter will give read only access to currently opened camera sensor temperature (double)

Configuration paths are relative to root or camera tree.

path

```
Parameters/Capture/PixelClock
```

in

```
GetParamerter
```

filter will give access to camera pixel clock

Remarks

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

Will throw atl::DomainError if the type is not convertible to requested type

Will throw atl::DomainError if the type does not exist or is null

Ensenso_GetParameter_StringOrNil

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets the value of a string parameter.

Syntax

```
void avl::Ensenso_GetParameter_StringOrNil
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::Optional<avl::EnsensoCameraInformation> inDevice,
 const atl::String& inPath,
 atl::Conditional<atl::String>& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices
	inDevice	const Optional<EnsensoCameraInformation>	NIL	Specific device
	inPath	const String&		Configuration path, relative
	outValue	Conditional<String>&		

Examples

Configuration tree

for example path

```
Sensor/Temperature
```

in

```
GetGlobalParamerter
```

filter will give read only access to currently opened camera sensor temperature (double)

Configuration paths are relative to root or camera tree.

path

```
Parameters/Capture/PixelClock
```

in

```
GetParamerter
```

filter will give access to camera pixel clock

Remarks

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

Will throw atl::DomainError if the type is not convertible to requested type

Ensenso_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an Image using Ensenso.

Syntax

```
bool avl::Ensenso_GrabImage
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::Optional<atl::File>& inParametersFile,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices
	inParametersFile	const Optional<File>&	NIL	Initial global parameters
	outImage	Image&		Captured Surface

Remarks

Initial parameters

Initial parameters are only set during capture start. To change parameters either restart the stream, or use appropriate Set/Get filters.

Settings

To obtain settings from the camera:

- From NxView Parameters window
  - Launch NxView
  - Open camera
  - Open Parameters window (menu Capture->Parameters...)
  - Adjust settings as wanted
  - Use *Save...* button
- From NxTreeEdit application
  - Either:
    - Launch Aurora Vision Studio, add Ensenso_GrabPoint3DGrid filter, Run it
    - Launch NxView, open camera
  - Launch NxTreeEdit, connect to wanted instance
  - Adjust settings as wanted, either in NxView or NxTreeEdit
  - right click on */Cameras/BySerialNo/WantedSerialNumber* and select *Copy value as JSON string*
  - save to a simple text file using an editor

The settings include all camera parameters, including *Link*, *Calibration* and *Parameters* The saved file can be then used in *inCalibrationFile* and *inSettingsFile* arguments.

To obtain global parameters, follow previous NxTreeEdit step, but save the global */Parameters* node.

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Ensenso_LoadSettings – Loads parameter data from a file.
- Ensenso_LoadCalibration – Loads calibration data from a file.
- Ensenso_LoadGlobalParameters – Loads global parameters from a file.

Ensenso_GrabImage_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an Image using Ensenso.

Syntax

```
bool avl::Ensenso_GrabImage_WithTimeout
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::Optional<atl::File>& inParametersFile,
 int inTimeout,
 atl::Conditional<avl::Image>& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Ensenso_State&			Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&			Structures identifying devices
	inParametersFile	const Optional<File>&		NIL	Initial global parameters
	inTimeout	int	100 - 3600000	5000	Maximum time to wait for frame in milliseconds
	outImage	Conditional<Image>&			Captured Surface

Remarks

Initial parameters

Initial parameters are only set during capture start. To change parameters either restart the stream, or use appropriate Set/Get filters.

Settings

To obtain settings from the camera:

- From NxView Parameters window
  - Launch NxView
  - Open camera
  - Open Parameters window (menu Capture->Parameters...)
  - Adjust settings as wanted
  - Use *Save...* button
- From NxTreeEdit application
  - Either:
    - Launch Aurora Vision Studio, add Ensenso_GrabPoint3DGrid filter, Run it
    - Launch NxView, open camera
  - Launch NxTreeEdit, connect to wanted instance
  - Adjust settings as wanted, either in NxView or NxTreeEdit
  - right click on */Cameras/BySerialNo/WantedSerialNumber* and select *Copy value as JSON string*
  - save to a simple text file using an editor

The settings include all camera parameters, including *Link*, *Calibration* and *Parameters* The saved file can be then used in *inCalibrationFile* and *inSettingsFile* arguments.

To obtain global parameters, follow previous NxTreeEdit step, but save the global */Parameters* node.

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Ensenso_LoadSettings – Loads parameter data from a file.
- Ensenso_LoadCalibration – Loads calibration data from a file.
- Ensenso_LoadGlobalParameters – Loads global parameters from a file.

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a Point3DGrid using Ensenso.

Syntax

```
bool avl::Ensenso_GrabPoint3DGrid
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 avl::Point3DGrid& outGrid,
 const atl::Optional<atl::File>& inParametersFile
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices
	outGrid	Point3DGrid&		Captured Grid
	inParametersFile	const Optional<File>&	NIL	Initial global parameters

Remarks

Initial parameters

Initial parameters are only set during capture start. To change parameters either restart the stream, or use appropriate Set/Get filters.

Settings

To obtain settings from the camera:

- From NxView Parameters window
  - Launch NxView
  - Open camera
  - Open Parameters window (menu Capture->Parameters...)
  - Adjust settings as wanted
  - Use *Save...* button
- From NxTreeEdit application
  - Either:
    - Launch Aurora Vision Studio, add Ensenso_GrabPoint3DGrid filter, Run it
    - Launch NxView, open camera
  - Launch NxTreeEdit, connect to wanted instance
  - Adjust settings as wanted, either in NxView or NxTreeEdit
  - right click on */Cameras/BySerialNo/WantedSerialNumber* and select *Copy value as JSON string*
  - save to a simple text file using an editor

The settings include all camera parameters, including *Link*, *Calibration* and *Parameters* The saved file can be then used in *inCalibrationFile* and *inSettingsFile* arguments.

To obtain global parameters, follow previous NxTreeEdit step, but save the global */Parameters* node.

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Ensenso_LoadSettings – Loads parameter data from a file.
- Ensenso_LoadCalibration – Loads calibration data from a file.
- Ensenso_LoadGlobalParameters – Loads global parameters from a file.

Ensenso_GrabPoint3DGrid_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a Point3DGrid using Ensenso.

Syntax

```
bool avl::Ensenso_GrabPoint3DGrid_WithTimeout
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 int inTimeout,
 atl::Conditional<avl::Point3DGrid>& outGrid,
 const atl::Optional<atl::File>& inParametersFile
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Ensenso_State&			Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&			Structures identifying devices
	inTimeout	int	100 - 3600000	5000	Maximum time to wait for frame in milliseconds
	outGrid	Conditional<Point3DGrid>&			Captured Grid
	inParametersFile	const Optional<File>&		NIL	Initial global parameters

Remarks

Initial parameters

Initial parameters are only set during capture start. To change parameters either restart the stream, or use appropriate Set/Get filters.

Settings

To obtain settings from the camera:

- From NxView Parameters window
  - Launch NxView
  - Open camera
  - Open Parameters window (menu Capture->Parameters...)
  - Adjust settings as wanted
  - Use *Save...* button
- From NxTreeEdit application
  - Either:
    - Launch Aurora Vision Studio, add Ensenso_GrabPoint3DGrid filter, Run it
    - Launch NxView, open camera
  - Launch NxTreeEdit, connect to wanted instance
  - Adjust settings as wanted, either in NxView or NxTreeEdit
  - right click on */Cameras/BySerialNo/WantedSerialNumber* and select *Copy value as JSON string*
  - save to a simple text file using an editor

The settings include all camera parameters, including *Link*, *Calibration* and *Parameters* The saved file can be then used in *inCalibrationFile* and *inSettingsFile* arguments.

To obtain global parameters, follow previous NxTreeEdit step, but save the global */Parameters* node.

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Ensenso_LoadSettings – Loads parameter data from a file.
- Ensenso_LoadCalibration – Loads calibration data from a file.
- Ensenso_LoadGlobalParameters – Loads global parameters from a file.

Ensenso_GrabSurface

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a Surface using Ensenso.

Syntax

```
bool avl::Ensenso_GrabSurface
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::Optional<atl::File>& inParametersFile,
 avl::Surface& outSurface
)
```

Parameters

	Name	Type	Default	Description
↔	ioState	Ensenso_State&		Object used to maintain state of the function.
→	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices
→	inParametersFile	const Optional<File>&	NIL	Initial global parameters
←	outSurface	Surface&		Captured Surface

Remarks

Initial parameters

Initial parameters are only set during capture start. To change parameters either restart the stream, or use appropriate Set/Get filters.

Settings

To obtain settings from the camera:

- From NxView Parameters window
  - Launch NxView
  - Open camera
  - Open Parameters window (menu Capture->Parameters...)
  - Adjust settings as wanted
  - Use *Save...* button
- From NxTreeEdit application
  - Either:
    - Launch Aurora Vision Studio, add Ensenso_GrabPoint3DGrid filter, Run it
    - Launch NxView, open camera
  - Launch NxTreeEdit, connect to wanted instance
  - Adjust settings as wanted, either in NxView or NxTreeEdit
  - right click on */Cameras/BySerialNo/WantedSerialNumber* and select *Copy value as JSON string*
  - save to a simple text file using an editor

The settings include all camera parameters, including *Link*, *Calibration* and *Parameters* The saved file can be then used in *inCalibrationFile* and *inSettingsFile* arguments.

To obtain global parameters, follow previous NxTreeEdit step, but save the global */Parameters* node.

Surface, Point3DGrid, Point color map

If one properly sets the global */Parameters/RenderPointMap* settings node (e.g. by using inParametersFile parameter) output data will change.

- To obtain Surface, RenderPointMap should contain ViewPose node, Point3DGrid will be outputted (perspective projection from camera).
- To obtain Point3DGrid, do not set the ViewPose parameter, and use only one stereoscopic camera
- To obtain Images one has to use Surface output, and make sure RenderPointMap/Texture is set to true

Currently only RenderPointMap node is applied from global Parameters. Other settings may be applied in future revisions.

Mixing Grab filters

Data internally is processed in frames. A frame can contain Surface, Image and Point3DGrid at the same time, or any mix of them. Grab filters will take proper data from current frame, if no data is available in current frame it is discarded, and next frame is taken. This process is repeated until data is available or the filter times out.

If one configures camera to grab a Surface and an Image, a frame will contain Surface and an Image, no Point3DGrid. When one uses GrabPoint3DGrid in that situation it will never output data, because no frame contains a Point3DGrid. Two consecutive GrabSurface will discard one Image. Second Grab surface finds no Surface in current frame - it was processed, it will discard it and look in the next frame.

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Ensenso_LoadSettings – Loads parameter data from a file.
- Ensenso_LoadCalibration – Loads calibration data from a file.
- Ensenso_LoadGlobalParameters – Loads global parameters from a file.

Ensenso_GrabSurface_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a Surface using Ensenso.

Syntax

```
bool avl::Ensenso_GrabSurface_WithTimeout
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::Optional<atl::File>& inParametersFile,
 int inTimeout,
 atl::Conditional<avl::Surface>& outSurface
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Ensenso_State&			Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&			Structures identifying devices
	inParametersFile	const Optional<File>&		NIL	Initial global parameters
	inTimeout	int	100 - 3600000	5000	Maximum time to wait for frame in milliseconds
	outSurface	Conditional<Surface>&			Captured Surface

Remarks

Initial parameters

Initial parameters are only set during capture start. To change parameters either restart the stream, or use appropriate Set/Get filters.

Settings

To obtain settings from the camera:

- From NxView Parameters window
  - Launch NxView
  - Open camera
  - Open Parameters window (menu Capture->Parameters...)
  - Adjust settings as wanted
  - Use *Save...* button
- From NxTreeEdit application
  - Either:
    - Launch Aurora Vision Studio, add Ensenso_GrabPoint3DGrid filter, Run it
    - Launch NxView, open camera
  - Launch NxTreeEdit, connect to wanted instance
  - Adjust settings as wanted, either in NxView or NxTreeEdit
  - right click on */Cameras/BySerialNo/WantedSerialNumber* and select *Copy value as JSON string*
  - save to a simple text file using an editor

The settings include all camera parameters, including *Link*, *Calibration* and *Parameters* The saved file can be then used in *inCalibrationFile* and *inSettingsFile* arguments.

To obtain global parameters, follow previous NxTreeEdit step, but save the global */Parameters* node.

Surface, Point3DGrid, Point color map

If one properly sets the global */Parameters/RenderPointMap* settings node (e.g. by using inParametersFile parameter) output data will change.

- To obtain Surface, RenderPointMap should contain ViewPose node, Point3DGrid will be outputted (perspective projection from camera).
- To obtain Point3DGrid, do not set the ViewPose parameter, and use only one stereoscopic camera
- To obtain Images one has to use Surface output, and make sure RenderPointMap/Texture is set to true

Currently only RenderPointMap node is applied from global Parameters. Other settings may be applied in future revisions.

Mixing Grab filters

Data internally is processed in frames. A frame can contain Surface, Image and Point3DGrid at the same time, or any mix of them. Grab filters will take proper data from current frame, if no data is available in current frame it is discarded, and next frame is taken. This process is repeated until data is available or the filter times out.

If one configures camera to grab a Surface and an Image, a frame will contain Surface and an Image, no Point3DGrid. When one uses GrabPoint3DGrid in that situation it will never output data, because no frame contains a Point3DGrid. Two consecutive GrabSurface will discard one Image. Second Grab surface finds no Surface in current frame - it was processed, it will discard it and look in the next frame.

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Ensenso_LoadSettings – Loads parameter data from a file.
- Ensenso_LoadCalibration – Loads calibration data from a file.
- Ensenso_LoadGlobalParameters – Loads global parameters from a file.

Ensenso_LoadCalibration

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Loads calibration data from a file.

Syntax

```
void avl::Ensenso_LoadCalibration
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::Optional<avl::EnsensoCameraInformation>& inDevice,
 const atl::File& inCalibrationFile
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices
	inDevice	const Optional<EnsensoCameraInformation>&	NIL	Specific device
	inCalibrationFile	const File&		JSON encoded parameters for camera

Description

Loads *Link* and *Calibration* nodes from settings file.

Remarks

Settings

To obtain settings from the camera:

- From NxView Parameters window
  - Launch NxView
  - Open camera
  - Open Parameters window (menu Capture->Parameters...)
  - Adjust settings as wanted
  - Use *Save...* button
- From NxTreeEdit application
  - Either:
    - Launch Aurora Vision Studio, add Ensenso_GrabPoint3DGrid filter, Run it
    - Launch NxView, open camera
  - Launch NxTreeEdit, connect to wanted instance
  - Adjust settings as wanted, either in NxView or NxTreeEdit
  - right click on */Cameras/BySerialNo/WantedSerialNumber* and select *Copy value as JSON string*
  - save to a simple text file using an editor

The settings include all camera parameters, including *Link*, *Calibration* and *Parameters* The saved file can be then used in *inCalibrationFile* and *inSettingsFile* arguments.

To obtain global parameters, follow previous NxTreeEdit step, but save the global */Parameters* node.

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Ensenso_LoadGlobalParameters

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Loads global parameters from a file.

Syntax

```
void avl::Ensenso_LoadGlobalParameters
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::File& inParametersFile
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices
	inParametersFile	const File&		JSON encoded parameters for camera

Remarks

Settings

To obtain settings from the camera:

- From NxView Parameters window
  - Launch NxView
  - Open camera
  - Open Parameters window (menu Capture->Parameters...)
  - Adjust settings as wanted
  - Use *Save...* button
- From NxTreeEdit application
  - Either:
    - Launch Aurora Vision Studio, add Ensenso_GrabPoint3DGrid filter, Run it
    - Launch NxView, open camera
  - Launch NxTreeEdit, connect to wanted instance
  - Adjust settings as wanted, either in NxView or NxTreeEdit
  - right click on */Cameras/BySerialNo/WantedSerialNumber* and select *Copy value as JSON string*
  - save to a simple text file using an editor

The settings include all camera parameters, including *Link*, *Calibration* and *Parameters* The saved file can be then used in *inCalibrationFile* and *inSettingsFile* arguments.

To obtain global parameters, follow previous NxTreeEdit step, but save the global */Parameters* node.

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Ensenso_LoadSettings

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Loads parameter data from a file.

Syntax

```
void avl::Ensenso_LoadSettings
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::Optional<avl::EnsensoCameraInformation>& inDevice,
 const atl::File& inSettingsFile
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices
	inDevice	const Optional<EnsensoCameraInformation>&	NIL	Specific device
	inSettingsFile	const File&		JSON encoded parameters for camera

Description

Loads *Parameters* node from settings file.

Remarks

Settings

To obtain settings from the camera:

- From NxView Parameters window
  - Launch NxView
  - Open camera
  - Open Parameters window (menu Capture->Parameters...)
  - Adjust settings as wanted
  - Use *Save...* button
- From NxTreeEdit application
  - Either:
    - Launch Aurora Vision Studio, add Ensenso_GrabPoint3DGrid filter, Run it
    - Launch NxView, open camera
  - Launch NxTreeEdit, connect to wanted instance
  - Adjust settings as wanted, either in NxView or NxTreeEdit
  - right click on */Cameras/BySerialNo/WantedSerialNumber* and select *Copy value as JSON string*
  - save to a simple text file using an editor

The settings include all camera parameters, including *Link*, *Calibration* and *Parameters* The saved file can be then used in *inCalibrationFile* and *inSettingsFile* arguments.

To obtain global parameters, follow previous NxTreeEdit step, but save the global */Parameters* node.

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Ensenso_SetGlobalParameter_Bool

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a bool parameter.

Syntax

```
void avl::Ensenso_SetGlobalParameter_Bool
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::String& inPath,
 const bool inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices
	inPath	const String&		Configuration path, relative
	inValue	const bool		

Examples

Configuration tree

for example path

```
Sensor/Temperature
```

in

```
GetGlobalParamerter
```

filter will give read only access to currently opened camera sensor temperature (double)

Configuration paths are relative to root or camera tree.

path

```
Parameters/Capture/PixelClock
```

in

```
GetParamerter
```

filter will give access to camera pixel clock

Remarks

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Ensenso_SetGlobalParameter_Double

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a double parameter.

Syntax

```
void avl::Ensenso_SetGlobalParameter_Double
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::String& inPath,
 const double inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices
	inPath	const String&		Configuration path, relative
	inValue	const double		

Examples

Configuration tree

for example path

```
Sensor/Temperature
```

in

```
GetGlobalParamerter
```

filter will give read only access to currently opened camera sensor temperature (double)

Configuration paths are relative to root or camera tree.

path

```
Parameters/Capture/PixelClock
```

in

```
GetParamerter
```

filter will give access to camera pixel clock

Remarks

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Ensenso_SetGlobalParameter_Int

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets an int parameter.

Syntax

```
void avl::Ensenso_SetGlobalParameter_Int
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::String& inPath,
 const int inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices
	inPath	const String&		Configuration path, relative
	inValue	const int		

Examples

Configuration tree

for example path

```
Sensor/Temperature
```

in

```
GetGlobalParamerter
```

filter will give read only access to currently opened camera sensor temperature (double)

Configuration paths are relative to root or camera tree.

path

```
Parameters/Capture/PixelClock
```

in

```
GetParamerter
```

filter will give access to camera pixel clock

Remarks

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Ensenso_SetGlobalParameter_String

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a string parameter.

Syntax

```
void avl::Ensenso_SetGlobalParameter_String
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::String& inPath,
 const atl::String& inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices
	inPath	const String&		Configuration path, relative
	inValue	const String&		

Examples

Configuration tree

for example path

```
Sensor/Temperature
```

in

```
GetGlobalParamerter
```

filter will give read only access to currently opened camera sensor temperature (double)

Configuration paths are relative to root or camera tree.

path

```
Parameters/Capture/PixelClock
```

in

```
GetParamerter
```

filter will give access to camera pixel clock

Remarks

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Ensenso_SetParameter_Bool

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a bool parameter.

Syntax

```
void avl::Ensenso_SetParameter_Bool
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::Optional<avl::EnsensoCameraInformation> inDevice,
 const atl::String& inPath,
 const bool inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices
	inDevice	const Optional<EnsensoCameraInformation>	NIL	Specific device
	inPath	const String&		Configuration path, relative
	inValue	const bool		

Examples

Configuration tree

for example path

```
Sensor/Temperature
```

in

```
GetGlobalParamerter
```

filter will give read only access to currently opened camera sensor temperature (double)

Configuration paths are relative to root or camera tree.

path

```
Parameters/Capture/PixelClock
```

in

```
GetParamerter
```

filter will give access to camera pixel clock

Remarks

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Ensenso_SetParameter_Double

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a double parameter.

Syntax

```
void avl::Ensenso_SetParameter_Double
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::Optional<avl::EnsensoCameraInformation> inDevice,
 const atl::String& inPath,
 const double inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices
	inDevice	const Optional<EnsensoCameraInformation>	NIL	Specific device
	inPath	const String&		Configuration path, relative
	inValue	const double		

Examples

Configuration tree

for example path

```
Sensor/Temperature
```

in

```
GetGlobalParamerter
```

filter will give read only access to currently opened camera sensor temperature (double)

Configuration paths are relative to root or camera tree.

path

```
Parameters/Capture/PixelClock
```

in

```
GetParamerter
```

filter will give access to camera pixel clock

Remarks

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Ensenso_SetParameter_Int

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets an int parameter.

Syntax

```
void avl::Ensenso_SetParameter_Int
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::Optional<avl::EnsensoCameraInformation> inDevice,
 const atl::String& inPath,
 const int inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices
	inDevice	const Optional<EnsensoCameraInformation>	NIL	Specific device
	inPath	const String&		Configuration path, relative
	inValue	const int		

Examples

Configuration tree

for example path

```
Sensor/Temperature
```

in

```
GetGlobalParamerter
```

filter will give read only access to currently opened camera sensor temperature (double)

Configuration paths are relative to root or camera tree.

path

```
Parameters/Capture/PixelClock
```

in

```
GetParamerter
```

filter will give access to camera pixel clock

Remarks

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Ensenso_SetParameter_String

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a string parameter.

Syntax

```
void avl::Ensenso_SetParameter_String
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::Optional<avl::EnsensoCameraInformation> inDevice,
 const atl::String& inPath,
 const atl::String& inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices
	inDevice	const Optional<EnsensoCameraInformation>	NIL	Specific device
	inPath	const String&		Configuration path, relative
	inValue	const String&		

Examples

Configuration tree

for example path

```
Sensor/Temperature
```

in

```
GetGlobalParamerter
```

filter will give read only access to currently opened camera sensor temperature (double)

Configuration paths are relative to root or camera tree.

path

```
Parameters/Capture/PixelClock
```

in

```
GetParamerter
```

filter will give access to camera pixel clock

Remarks

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Ensenso_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Starts capturing data from an Ensenso camera.

Applications: Typically used for establishing camera connectivity before the first trigger event. Especially important for multiple-camera systems.

Syntax

```
void avl::Ensenso_StartAcquisition
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices,
 const atl::Optional<atl::File>& inParametersFile,
 const bool inSoftwareTriggered
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices
	inParametersFile	const Optional<File>&	NIL	Initial global parameters
	inSoftwareTriggered	const bool	False	When true, capture will wait for GenerateSoftwareTrigger

Remarks

Initial parameters

Initial parameters are only set during capture start. To change parameters either restart the stream, or use appropriate Set/Get filters.

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Ensenso_LoadSettings – Loads parameter data from a file.
- Ensenso_LoadCalibration – Loads calibration data from a file.
- Ensenso_LoadGlobalParameters – Loads global parameters from a file.

Ensenso_StopAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Stops data acquisition in a Ensenso device.

Syntax

```
void avl::Ensenso_StopAcquisition
(
 Ensenso_State& ioState,
 const atl::Array<avl::EnsensoCameraInformation>& inDevices
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ensenso_State&		Object used to maintain state of the function.
	inDevices	const Array<EnsensoCameraInformation>&		Structures identifying devices

Remarks

Camera identification

When there is only one Ensenso camera connected, the field **inDevices** can be set to Auto. In this situation, the first available camera will be used.

inDevices can be used to pick one or many of multiple cameras connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Ensenso SDK software, and uEye driver (same as in IDS camera filters)

Ensenso SDK can be downloaded from the following website: https://www.ensenso.com/support/sdk-download/

uEye driver can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

Recommended Ensenso SDK version for Aurora Vision Studio usage is **2.2** and uEye driver **4.90**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

157. Flir

Table of content:

FlyCapture_GenerateSoftwareTrigger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Generates software trigger.

Syntax

```
void avl::FlyCapture_GenerateSoftwareTrigger
(
 FlyCapture_State& ioState,
 atl::Optional<int> inDeviceSerialNumber
)
```

Parameters

	Name	Type	Default	Description
	ioState	FlyCapture_State&		Object used to maintain state of the function.
	inDeviceSerialNumber	Optional<int>	NIL	Source device serial number

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

FlyCapture_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures images from a Flir camera.

Syntax

```
bool avl::FlyCapture_GrabImage
(
 FlyCapture_State& ioState,
 atl::Optional<int> inDeviceSerialNumber,
 atl::Optional<const avl::Box&> inAoi,
 atl::Optional<float> inShutter,
 atl::Optional<float> inGain,
 atl::Optional<float> inExposure,
 atl::Optional<float> inFrameRate,
 avl::FlirTriggerMode::Type inTriggerMode,
 avl::FlirColorMode::Type inColorMode,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	FlyCapture_State&			Object used to maintain state of the function.
	inDeviceSerialNumber	Optional<int>		NIL	Source device serial number
	inAoi	Optional<const Box&>		NIL	Required fragment of image to stream
	inShutter	Optional<float>	0.0 - ∞	NIL	Shutter time
	inGain	Optional<float>	0.0 - ∞	NIL	Image analog gain value
	inExposure	Optional<float>		NIL	Exposure time
	inFrameRate	Optional<float>	0.1 - ∞	NIL	Requested camera frame rate
	inTriggerMode	FlirTriggerMode::Type			Camera trigger mode and source control
	inColorMode	FlirColorMode::Type			Requested grayscale or color mode of output image
	outImage	Image&			Output image

Remarks

Getting Started

This application note provides information on how to install, configure, and use Point Grey imaging cameras with Aurora Vision software: Getting Started with Aurora Vision.

Camera driver software

This filter is intended to cooperate with camera using its vendor SDK. To be able to connect to camera it is required to install Fly Capture 2 SDK software with camera dedicated drivers. Currently Aurora Vision Studio requires **Fly Capture version 2.13.3.61**.

Fly Capture 2 SDK can be downloaded from the following website: https://www.flir.com/support/browse/camera-cores--components/machine-vision-cameras/sdks (registration may be required).

For 64 bit Aurora Vision Studio it is required to install 64 bit SDK version.

Camera identification

When there is only one camera connected to computer, **inDeviceSerialNumber** field can be set to Auto. In this situation first available camera will be found and connected.

inDeviceSerialNumber can be used to pick one of multiple cameras connected to computer. Set this field to device serial number (for example "1234567", should be available on device casing as "s/n: 1234567").

Camera parameters

Most of parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameters for automatic configuration by camera driver (for example shutter time will be adjusted according to light conditions).

To change other and more advanced camera parameters use configuration tool "FlyCap2" available with Fly Capture 2 SDK. Refer to SDK documentation to find information about parameters and how to save parameters into memory channels.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- FlyCapture_GrabImage_WithTimeout – Captures images from a Flir camera.
- FlyCapture_StartAcquisition – Starts capturing images from a Flir camera.
- FlyCapture_SetStrobe – Configures Flir camera Strobe.

FlyCapture_GrabImage_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures images from a Flir camera.

Applications: Use this filter if the trigger may be not coming for some time, while the application should be performing other operations continuously (e.g. processing HMI events).

Syntax

```
bool avl::FlyCapture_GrabImage_WithTimeout
(
 FlyCapture_State& ioState,
 atl::Optional<int> inDeviceSerialNumber,
 atl::Optional<const avl::Box&> inAoi,
 atl::Optional<float> inShutter,
 atl::Optional<float> inGain,
 atl::Optional<float> inExposure,
 atl::Optional<float> inFrameRate,
 atl::Optional<int> inTimeout,
 avl::FlirTriggerMode::Type inTriggerMode,
 avl::FlirColorMode::Type inColorMode,
 atl::Conditional<avl::Image>& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	FlyCapture_State&			Object used to maintain state of the function.
	inDeviceSerialNumber	Optional<int>		NIL	Source device serial number
	inAoi	Optional<const Box&>		NIL	Required fragment of image to stream
	inShutter	Optional<float>	0.0 - ∞	NIL	Shutter time
	inGain	Optional<float>	0.0 - ∞	NIL	Image analog gain value
	inExposure	Optional<float>		NIL	Exposure time
	inFrameRate	Optional<float>	0.1 - ∞	NIL	Requested camera frame rate
	inTimeout	Optional<int>	100 - 3600000	5000	Timeout
	inTriggerMode	FlirTriggerMode::Type			Camera trigger mode and source control
	inColorMode	FlirColorMode::Type			Requested grayscale or color mode of output image
	outImage	Conditional<Image>&			Output image

Remarks

Getting Started

This application note provides information on how to install, configure, and use Point Grey imaging cameras with Aurora Vision software: Getting Started with Aurora Vision.

Camera driver software

This filter is intended to cooperate with camera using its vendor SDK. To be able to connect to camera it is required to install Fly Capture 2 SDK software with camera dedicated drivers. Currently Aurora Vision Studio requires **Fly Capture version 2.13.3.61**.

Fly Capture 2 SDK can be downloaded from the following website: https://www.flir.com/support/browse/camera-cores--components/machine-vision-cameras/sdks (registration may be required).

For 64 bit Aurora Vision Studio it is required to install 64 bit SDK version.

Camera identification

When there is only one camera connected to computer, **inDeviceSerialNumber** field can be set to Auto. In this situation first available camera will be found and connected.

inDeviceSerialNumber can be used to pick one of multiple cameras connected to computer. Set this field to device serial number (for example "1234567", should be available on device casing as "s/n: 1234567").

Camera parameters

Most of parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameters for automatic configuration by camera driver (for example shutter time will be adjusted according to light conditions).

To change other and more advanced camera parameters use configuration tool "FlyCap2" available with Fly Capture 2 SDK. Refer to SDK documentation to find information about parameters and how to save parameters into memory channels.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- FlyCapture_GrabImage – Captures images from a Flir camera.
- FlyCapture_StartAcquisition – Starts capturing images from a Flir camera.
- FlyCapture_SetStrobe – Configures Flir camera Strobe.

FlyCapture_SetStrobe

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Configures Flir camera Strobe.

Syntax

```
void avl::FlyCapture_SetStrobe
(
 FlyCapture_State& ioState,
 atl::Optional<int> inDeviceSerialNumber,
 int inSource,
 bool inEnabled,
 int inPolarity,
 float inDelay,
 float inDuration
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	FlyCapture_State&			Object used to maintain state of the function.
	inDeviceSerialNumber	Optional<int>		NIL	Source device serial number
	inSource	int	0 - +∞		Source value
	inEnabled	bool			Flag controlling on/off
	inPolarity	int			Signal polarity
	inDelay	float	0.0 - ∞		Signal delay (in ms)
	inDuration	float	0.0 - ∞		Signal duration (in ms)

Remarks

Getting Started

This application note provides information on how to install, configure, and use Point Grey imaging cameras with Aurora Vision software: Getting Started with Aurora Vision.

Camera driver software

This filter is intended to cooperate with camera using its vendor SDK. To be able to connect to camera it is required to install Fly Capture 2 SDK software with camera dedicated drivers. Currently Aurora Vision Studio requires **Fly Capture version 2.13.3.61**.

Fly Capture 2 SDK can be downloaded from the following website: https://www.flir.com/support/browse/camera-cores--components/machine-vision-cameras/sdks (registration may be required).

For 64 bit Aurora Vision Studio it is required to install 64 bit SDK version.

Camera identification

When there is only one camera connected to computer, **inDeviceSerialNumber** field can be set to Auto. In this situation first available camera will be found and connected.

inDeviceSerialNumber can be used to pick one of multiple cameras connected to computer. Set this field to device serial number (for example "1234567", should be available on device casing as "s/n: 1234567").

Camera parameters

Most of parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameters for automatic configuration by camera driver (for example shutter time will be adjusted according to light conditions).

To change other and more advanced camera parameters use configuration tool "FlyCap2" available with Fly Capture 2 SDK. Refer to SDK documentation to find information about parameters and how to save parameters into memory channels.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- FlyCapture_GrabImage – Captures images from a Flir camera.
- FlyCapture_GrabImage_WithTimeout – Captures images from a Flir camera.
- FlyCapture_StartAcquisition – Starts capturing images from a Flir camera.

FlyCapture_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Starts capturing images from a Flir camera.

Applications: Typically used for establishing camera connectivity before the first trigger event. Especially important for multiple-camera systems.

Syntax

```
void avl::FlyCapture_StartAcquisition
(
 FlyCapture_State& ioState,
 atl::Optional<int> inDeviceSerialNumber,
 atl::Optional<const avl::Box&> inAoi,
 atl::Optional<float> inShutter,
 atl::Optional<float> inGain,
 atl::Optional<float> inExposure,
 atl::Optional<float> inFrameRate,
 avl::FlirTriggerMode::Type inTriggerMode,
 avl::FlirColorMode::Type inColorMode
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	FlyCapture_State&			Object used to maintain state of the function.
	inDeviceSerialNumber	Optional<int>		NIL	Source device serial number
	inAoi	Optional<const Box&>		NIL	Required fragment of image to stream
	inShutter	Optional<float>	0.0 - ∞	NIL	Shutter time
	inGain	Optional<float>	0.0 - ∞	NIL	Image analog gain value
	inExposure	Optional<float>		NIL	Exposure time
	inFrameRate	Optional<float>	0.1 - ∞	NIL	Requested camera frame rate
	inTriggerMode	FlirTriggerMode::Type			Camera trigger mode and source control
	inColorMode	FlirColorMode::Type			Requested grayscale or color mode of output image

Remarks

Getting Started

This application note provides information on how to install, configure, and use Point Grey imaging cameras with Aurora Vision software: Getting Started with Aurora Vision.

Camera driver software

This filter is intended to cooperate with camera using its vendor SDK. To be able to connect to camera it is required to install Fly Capture 2 SDK software with camera dedicated drivers. Currently Aurora Vision Studio requires **Fly Capture version 2.13.3.61**.

Fly Capture 2 SDK can be downloaded from the following website: https://www.flir.com/support/browse/camera-cores--components/machine-vision-cameras/sdks (registration may be required).

For 64 bit Aurora Vision Studio it is required to install 64 bit SDK version.

Camera identification

When there is only one camera connected to computer, **inDeviceSerialNumber** field can be set to Auto. In this situation first available camera will be found and connected.

inDeviceSerialNumber can be used to pick one of multiple cameras connected to computer. Set this field to device serial number (for example "1234567", should be available on device casing as "s/n: 1234567").

Camera parameters

Most of parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameters for automatic configuration by camera driver (for example shutter time will be adjusted according to light conditions).

To change other and more advanced camera parameters use configuration tool "FlyCap2" available with Fly Capture 2 SDK. Refer to SDK documentation to find information about parameters and how to save parameters into memory channels.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- FlyCapture_GrabImage – Captures images from a Flir camera.
- FlyCapture_GrabImage_WithTimeout – Captures images from a Flir camera.
- FlyCapture_SetStrobe – Configures Flir camera Strobe.

158. Gocator

Table of content:

Gocator_GenerateSoftwareTrigger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Generates software trigger.

Syntax

```
void avl::Gocator_GenerateSoftwareTrigger
(
 Gocator_State& ioState,
 const atl::Optional<avl::GocatorAddress>& inAddress
)
```

Parameters

	Name	Type	Default	Description
	ioState	Gocator_State&		Object used to maintain state of the function.
	inAddress	const Optional<GocatorAddress>&	NIL	Gocator Device identifying IP address (e.g. "127.0.0.1") or serial number (e.g. "SN:17335")

Remarks

Device identification

When there is only one device connected, the field **inAddress** can be set to Auto. In this situation, the first available device will be used.

inAddress can be used to pick one of multiple devices connected to the computer. **inAddress** can be set to:

- **IP Address** - Gocator Device identifying IP address (e.g. "127.0.0.1")
- **Serial Number** - Gocator Device serial number with "SN:" prefix (e.g. "SN:17335")

Gocator emulator

This filter was tested with **Gocator emulator in version 6.1.20.8**.

Gocator_GetCurrentJobName

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads current job name.

Syntax

```
void avl::Gocator_GetCurrentJobName
(
 Gocator_State& ioState,
 const atl::Optional<avl::GocatorAddress>& inAddress,
 atl::String& outJobName
)
```

Parameters

	Name	Type	Default	Description
	ioState	Gocator_State&		Object used to maintain state of the function.
	inAddress	const Optional<GocatorAddress>&	NIL	Gocator Device identifying IP address (e.g. "127.0.0.1") or serial number (e.g. "SN:17335")
	outJobName	String&		Current Job name

Remarks

Device identification

When there is only one device connected, the field **inAddress** can be set to Auto. In this situation, the first available device will be used.

inAddress can be used to pick one of multiple devices connected to the computer. **inAddress** can be set to:

- **IP Address** - Gocator Device identifying IP address (e.g. "127.0.0.1")
- **Serial Number** - Gocator Device serial number with "SN:" prefix (e.g. "SN:17335")

Gocator emulator

This filter was tested with **Gocator emulator in version 6.1.20.8**.

Gocator_GetTriggerSource

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets current trigger source.

Syntax

```
void avl::Gocator_GetTriggerSource
(
 Gocator_State& ioState,
 const atl::Optional<avl::GocatorAddress>& inAddress,
 avl::GocatorTriggerSource::Type& outTriggerSource
)
```

Parameters

	Name	Type	Default	Description
	ioState	Gocator_State&		Object used to maintain state of the function.
	inAddress	const Optional<GocatorAddress>&	NIL	Gocator Device identifying IP address (e.g. "127.0.0.1") or serial number (e.g. "SN:17335")
	outTriggerSource	GocatorTriggerSource::Type&		Current trigger source

Remarks

Device identification

When there is only one device connected, the field **inAddress** can be set to Auto. In this situation, the first available device will be used.

inAddress can be used to pick one of multiple devices connected to the computer. **inAddress** can be set to:

- **IP Address** - Gocator Device identifying IP address (e.g. "127.0.0.1")
- **Serial Number** - Gocator Device serial number with "SN:" prefix (e.g. "SN:17335")

Gocator emulator

This filter was tested with **Gocator emulator in version 6.1.20.8**.

Gocator_GrabImages

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures images from Gocator device.

Syntax

```
bool avl::Gocator_GrabImages
(
 Gocator_State& ioState,
 const atl::Optional<avl::GocatorAddress>& inAddress,
 int inInputQueueSize,
 avl::Image& outImage1,
 avl::Image& outImage2,
 avl::GocatorFrameInfo& outFrameInfo
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Gocator_State&			Object used to maintain state of the function.
	inAddress	const Optional<GocatorAddress>&		NIL	Gocator Device identifying IP address (e.g. "127.0.0.1") or serial number (e.g. "SN:17335")
	inInputQueueSize	int	1 - 50000		Number of incoming frames that can be buffered before the application is able to process them
	outImage1	Image&			First output image
	outImage2	Image&			Second output image
	outFrameInfo	GocatorFrameInfo&			

Remarks

Device identification

When there is only one device connected, the field **inAddress** can be set to Auto. In this situation, the first available device will be used.

inAddress can be used to pick one of multiple devices connected to the computer. **inAddress** can be set to:

- **IP Address** - Gocator Device identifying IP address (e.g. "127.0.0.1")
- **Serial Number** - Gocator Device serial number with "SN:" prefix (e.g. "SN:17335")

Gocator emulator

This filter was tested with **Gocator emulator in version 6.1.20.8**.

See Also

- Gocator_StartAcquisition – Initializes and starts image acquisition in a Gocator device.
- Gocator_GrabSurface – Captures Surface from Gocator device.
- Gocator_GrabSurface_WithTimeout – Captures Surface from Gocator device with timeout; returns Nil if no data comes in the specified time.
- Gocator_GrabImages_WithTimeout – Captures images from Gocator device.
- Gocator_GrabProfile – Captures profile from Gocator device.
- Gocator_GrabProfile_WithTimeout – Captures profile from Gocator device.
- Gocator_GrabUniformProfile – Captures uniform profile from Gocator device.
- Gocator_GrabUniformProfile_WithTimeout – Captures uniform profile from Gocator device.

Gocator_GrabImages_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures images from Gocator device.

Syntax

```
bool avl::Gocator_GrabImages_WithTimeout
(
 Gocator_State& ioState,
 const atl::Optional<avl::GocatorAddress>& inAddress,
 int inInputQueueSize,
 int inTimeout,
 atl::Conditional<avl::Image>& outImage1,
 atl::Conditional<avl::Image>& outImage2,
 atl::Conditional<avl::GocatorFrameInfo>& outFrameInfo
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Gocator_State&			Object used to maintain state of the function.
	inAddress	const Optional<GocatorAddress>&		NIL	Gocator Device identifying IP address (e.g. "127.0.0.1") or serial number (e.g. "SN:17335")
	inInputQueueSize	int	1 - 50000		Number of incoming frames that can be buffered before the application is able to process them
	inTimeout	int	10 - ∞	100	Maximum time to wait for data in milliseconds
	outImage1	Conditional<Image>&			First output image
	outImage2	Conditional<Image>&			Second output image
	outFrameInfo	Conditional<GocatorFrameInfo>&			

Remarks

Device identification

When there is only one device connected, the field **inAddress** can be set to Auto. In this situation, the first available device will be used.

inAddress can be used to pick one of multiple devices connected to the computer. **inAddress** can be set to:

- **IP Address** - Gocator Device identifying IP address (e.g. "127.0.0.1")
- **Serial Number** - Gocator Device serial number with "SN:" prefix (e.g. "SN:17335")

Gocator emulator

This filter was tested with **Gocator emulator in version 6.1.20.8**.

See Also

- Gocator_StartAcquisition – Initializes and starts image acquisition in a Gocator device.
- Gocator_GrabSurface – Captures Surface from Gocator device.
- Gocator_GrabSurface_WithTimeout – Captures Surface from Gocator device with timeout; returns Nil if no data comes in the specified time.
- Gocator_GrabImages – Captures images from Gocator device.
- Gocator_GrabProfile – Captures profile from Gocator device.
- Gocator_GrabProfile_WithTimeout – Captures profile from Gocator device.
- Gocator_GrabUniformProfile – Captures uniform profile from Gocator device.
- Gocator_GrabUniformProfile_WithTimeout – Captures uniform profile from Gocator device.

Gocator_GrabMeasurement

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures Measurement data from Gocator device.

Syntax

```
bool avl::Gocator_GrabMeasurement
(
 Gocator_State& ioState,
 const atl::Optional<avl::GocatorAddress>& inAddress,
 int inInputQueueSize,
 int inMeasurementID,
 double& outValue,
 bool& outDecision,
 avl::GocatorDecisionCode::Type& outDecisionCode,
 avl::GocatorFrameInfo& outFrameInfo
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Gocator_State&			Object used to maintain state of the function.
	inAddress	const Optional<GocatorAddress>&		NIL	Gocator Device identifying IP address (e.g. "127.0.0.1") or serial number (e.g. "SN:17335")
	inInputQueueSize	int	1 - 50000		Number of incoming frames that can be buffered before the application is able to process them
	inMeasurementID	int	0 - ∞		
	outValue	double&			
	outDecision	bool&			
	outDecisionCode	GocatorDecisionCode::Type&			
	outFrameInfo	GocatorFrameInfo&			

Remarks

Device identification

When there is only one device connected, the field **inAddress** can be set to Auto. In this situation, the first available device will be used.

inAddress can be used to pick one of multiple devices connected to the computer. **inAddress** can be set to:

- **IP Address** - Gocator Device identifying IP address (e.g. "127.0.0.1")
- **Serial Number** - Gocator Device serial number with "SN:" prefix (e.g. "SN:17335")

Gocator emulator

This filter was tested with **Gocator emulator in version 6.1.20.8**.

Gocator_GrabMeasurement_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures Measurement data from Gocator device.

Syntax

```
bool avl::Gocator_GrabMeasurement_WithTimeout
(
 Gocator_State& ioState,
 const atl::Optional<avl::GocatorAddress>& inAddress,
 int inInputQueueSize,
 int inTimeout,
 int inMeasurementID,
 atl::Conditional<atl::real64>& outValue,
 atl::Conditional<bool>& outDecision,
 atl::Conditional<avl::GocatorDecisionCode::Type>& outDecisionCode,
 atl::Conditional<avl::GocatorFrameInfo>& outFrameInfo
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	Gocator_State&			Object used to maintain state of the function.
→	inAddress	const Optional<GocatorAddress>&		NIL	Gocator Device identifying IP address (e.g. "127.0.0.1") or serial number (e.g. "SN:17335")
→	inInputQueueSize	int	1 - 50000		Number of incoming frames that can be buffered before the application is able to process them
→	inTimeout	int	10 - ∞	100	
→	inMeasurementID	int	0 - ∞		
←	outValue	Conditional<real64>&			
←	outDecision	Conditional<bool>&			
←	outDecisionCode	Conditional<GocatorDecisionCode::Type>&			
←	outFrameInfo	Conditional<GocatorFrameInfo>&			

Remarks

Device identification

When there is only one device connected, the field **inAddress** can be set to Auto. In this situation, the first available device will be used.

inAddress can be used to pick one of multiple devices connected to the computer. **inAddress** can be set to:

- **IP Address** - Gocator Device identifying IP address (e.g. "127.0.0.1")
- **Serial Number** - Gocator Device serial number with "SN:" prefix (e.g. "SN:17335")

Gocator emulator

This filter was tested with **Gocator emulator in version 6.1.20.8**.

Gocator_GrabPoint3DGrid

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures Point3DGrid (non resampled surface) from Gocator device.

Syntax

```
bool avl::Gocator_GrabPoint3DGrid
(
 Gocator_State& ioState,
 const atl::Optional<avl::GocatorAddress>& inAddress,
 int inInputQueueSize,
 avl::Point3DGrid& outPoint3DGrid,
 atl::Conditional<avl::Image>& outSurfaceIntensity,
 avl::GocatorFrameInfo& outFrameInfo
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Gocator_State&			Object used to maintain state of the function.
	inAddress	const Optional<GocatorAddress>&		NIL	Gocator Device identifying IP address (e.g. "127.0.0.1") or serial number (e.g. "SN:17335")
	inInputQueueSize	int	1 - 50000		Number of incoming frames that can be buffered before the application is able to process them
	outPoint3DGrid	Point3DGrid&			
	outSurfaceIntensity	Conditional<Image>&			
	outFrameInfo	GocatorFrameInfo&			

Remarks

Device identification

When there is only one device connected, the field **inAddress** can be set to Auto. In this situation, the first available device will be used.

inAddress can be used to pick one of multiple devices connected to the computer. **inAddress** can be set to:

- **IP Address** - Gocator Device identifying IP address (e.g. "127.0.0.1")
- **Serial Number** - Gocator Device serial number with "SN:" prefix (e.g. "SN:17335")

Gocator emulator

This filter was tested with **Gocator emulator in version 6.1.20.8**.

Gocator_GrabPoint3DGrid_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures Point3DGrid (Un-Resampled surface) from Gocator device with timeout; returns Nil if no data comes in the specified time.

Syntax

```
bool avl::Gocator_GrabPoint3DGrid_WithTimeout
(
 Gocator_State& ioState,
 const atl::Optional<avl::GocatorAddress>& inAddress,
 int inInputQueueSize,
 int inTimeout,
 atl::Conditional<avl::Point3DGrid>& outPoint3DGrid,
 atl::Conditional<avl::Image>& outSurfaceIntensity,
 atl::Conditional<avl::GocatorFrameInfo>& outFrameInfo
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	Gocator_State&			Object used to maintain state of the function.
→	inAddress	const Optional<GocatorAddress>&		NIL	Gocator Device identifying IP address (e.g. "127.0.0.1") or serial number (e.g. "SN:17335")
→	inInputQueueSize	int	1 - 50000		Number of incoming frames that can be buffered before the application is able to process them
→	inTimeout	int	10 - ∞	100	Maximum time to wait for data in milliseconds
←	outPoint3DGrid	Conditional<Point3DGrid>&			
←	outSurfaceIntensity	Conditional<Image>&			
←	outFrameInfo	Conditional<GocatorFrameInfo>&			

Remarks

Device identification

When there is only one device connected, the field **inAddress** can be set to Auto. In this situation, the first available device will be used.

inAddress can be used to pick one of multiple devices connected to the computer. **inAddress** can be set to:

- **IP Address** - Gocator Device identifying IP address (e.g. "127.0.0.1")
- **Serial Number** - Gocator Device serial number with "SN:" prefix (e.g. "SN:17335")

Gocator emulator

This filter was tested with **Gocator emulator in version 6.1.20.8**.

Gocator_GrabProfile

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures profile from Gocator device.

Syntax

```
bool avl::Gocator_GrabProfile
(
 Gocator_State& ioState,
 const atl::Optional<avl::GocatorAddress>& inAddress,
 int inInputQueueSize,
 atl::Array<avl::Point2D>& outProfileData,
 atl::Conditional<avl::Profile>& outProfileIntensity,
 atl::real& outXScale,
 atl::real& outXOffset,
 atl::real& outZScale,
 atl::real& outZOffset,
 avl::GocatorFrameInfo& outFrameInfo
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Gocator_State&			Object used to maintain state of the function.
	inAddress	const Optional<GocatorAddress>&		NIL	Gocator Device identifying IP address (e.g. "127.0.0.1") or serial number (e.g. "SN:17335")
	inInputQueueSize	int	1 - 50000		Number of incoming frames that can be buffered before the application is able to process them
	outProfileData	Array<Point2D>&			
	outProfileIntensity	Conditional<Profile>&			
	outXScale	real&			Xscale in mm
	outXOffset	real&			Xoffset in mm
	outZScale	real&			Zscale in mm
	outZOffset	real&			Zoffset in mm
	outFrameInfo	GocatorFrameInfo&			

Remarks

Device identification

When there is only one device connected, the field **inAddress** can be set to Auto. In this situation, the first available device will be used.

inAddress can be used to pick one of multiple devices connected to the computer. **inAddress** can be set to:

- **IP Address** - Gocator Device identifying IP address (e.g. "127.0.0.1")
- **Serial Number** - Gocator Device serial number with "SN:" prefix (e.g. "SN:17335")

Gocator emulator

This filter was tested with **Gocator emulator in version 6.1.20.8**.

See Also

- Gocator_StartAcquisition – Initializes and starts image acquisition in a Gocator device.
- Gocator_GrabSurface – Captures Surface from Gocator device.
- Gocator_GrabSurface_WithTimeout – Captures Surface from Gocator device with timeout; returns Nil if no data comes in the specified time.
- Gocator_GrabImages – Captures images from Gocator device.
- Gocator_GrabImages_WithTimeout – Captures images from Gocator device.
- Gocator_GrabProfile_WithTimeout – Captures profile from Gocator device.
- Gocator_GrabUniformProfile – Captures uniform profile from Gocator device.
- Gocator_GrabUniformProfile_WithTimeout – Captures uniform profile from Gocator device.

Gocator_GrabProfile_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures profile from Gocator device.

Syntax

```
bool avl::Gocator_GrabProfile_WithTimeout
(
 Gocator_State& ioState,
 const atl::Optional<avl::GocatorAddress>& inAddress,
 int inInputQueueSize,
 int inTimeout,
 atl::Conditional<atl::Array<avl::Point2D>>& outProfileData,
 atl::Conditional<avl::Profile>& outProfileIntensity,
 atl::Conditional<atl::real>& outXScale,
 atl::Conditional<atl::real>& outXOffset,
 atl::Conditional<atl::real>& outZScale,
 atl::Conditional<atl::real>& outZOffset,
 atl::Conditional<avl::GocatorFrameInfo>& outFrameInfo
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Gocator_State&			Object used to maintain state of the function.
	inAddress	const Optional<GocatorAddress>&		NIL	Gocator Device identifying IP address (e.g. "127.0.0.1") or serial number (e.g. "SN:17335")
	inInputQueueSize	int	1 - 50000		Number of incoming frames that can be buffered before the application is able to process them
	inTimeout	int	10 - ∞	100	
	outProfileData	Conditional<Array<Point2D>>&			
	outProfileIntensity	Conditional<Profile>&			
	outXScale	Conditional<real>&			Xscale in mm
	outXOffset	Conditional<real>&			Xoffset in mm
	outZScale	Conditional<real>&			Zscale in mm
	outZOffset	Conditional<real>&			Zoffset in mm
	outFrameInfo	Conditional<GocatorFrameInfo>&			

Remarks

Device identification

When there is only one device connected, the field **inAddress** can be set to Auto. In this situation, the first available device will be used.

inAddress can be used to pick one of multiple devices connected to the computer. **inAddress** can be set to:

- **IP Address** - Gocator Device identifying IP address (e.g. "127.0.0.1")
- **Serial Number** - Gocator Device serial number with "SN:" prefix (e.g. "SN:17335")

Gocator emulator

This filter was tested with **Gocator emulator in version 6.1.20.8**.

See Also

- Gocator_StartAcquisition – Initializes and starts image acquisition in a Gocator device.
- Gocator_GrabSurface – Captures Surface from Gocator device.
- Gocator_GrabSurface_WithTimeout – Captures Surface from Gocator device with timeout; returns Nil if no data comes in the specified time.
- Gocator_GrabImages – Captures images from Gocator device.
- Gocator_GrabImages_WithTimeout – Captures images from Gocator device.
- Gocator_GrabProfile – Captures profile from Gocator device.
- Gocator_GrabUniformProfile – Captures uniform profile from Gocator device.
- Gocator_GrabUniformProfile_WithTimeout – Captures uniform profile from Gocator device.

Gocator_GrabSection

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures Section from Gocator device.

Syntax

```
bool avl::Gocator_GrabSection
(
 Gocator_State& ioState,
 const atl::Optional<avl::GocatorAddress>& inAddress,
 int inInputQueueSize,
 int inSectionID,
 atl::Array<avl::Point2D>& outSectionData,
 atl::Conditional<avl::Profile>& outSectionIntensity,
 atl::real& outXScale,
 atl::real& outXOffset,
 atl::real& outZScale,
 atl::real& outZOffset,
 avl::GocatorFrameInfo& outFrameInfo
)
```

Parameters

	Name	Type	Range	Default	Description
⇄	ioState	Gocator_State&			Object used to maintain state of the function.
→	inAddress	const Optional<GocatorAddress>&		NIL	Gocator Device identifying IP address (e.g. "127.0.0.1") or serial number (e.g. "SN:17335")
→	inInputQueueSize	int	1 - 50000		Number of incoming frames that can be buffered before the application is able to process them
→	inSectionID	int	0 - ∞		
←	outSectionData	Array<Point2D>&			
←	outSectionIntensity	Conditional<Profile>&			
←	outXScale	real&			X scale in mm
←	outXOffset	real&			X offset in mm
←	outZScale	real&			Z scale in mm
←	outZOffset	real&			Z offset in mm
←	outFrameInfo	GocatorFrameInfo&			

Remarks

Device identification

When there is only one device connected, the field **inAddress** can be set to Auto. In this situation, the first available device will be used.

inAddress can be used to pick one of multiple devices connected to the computer. **inAddress** can be set to:

- **IP Address** - Gocator Device identifying IP address (e.g. "127.0.0.1")
- **Serial Number** - Gocator Device serial number with "SN:" prefix (e.g. "SN:17335")

Gocator emulator

This filter was tested with **Gocator emulator in version 6.1.20.8**.

Gocator_GrabSection_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures Section from Gocator device.

Syntax

```
bool avl::Gocator_GrabSection_WithTimeout
(
 Gocator_State& ioState,
 const atl::Optional<avl::GocatorAddress>& inAddress,
 int inInputQueueSize,
 int inTimeout,
 int inSectionID,
 atl::Conditional<atl::Array<avl::Point2D>>& outSectionData,
 atl::Conditional<avl::Profile>& outSectionIntensity,
 atl::Conditional<atl::real>& outXScale,
 atl::Conditional<atl::real>& outXOffset,
 atl::Conditional<atl::real>& outZScale,
 atl::Conditional<atl::real>& outZOffset,
 atl::Conditional<avl::GocatorFrameInfo>& outFrameInfo
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	Gocator_State&			Object used to maintain state of the function.
→	inAddress	const Optional<GocatorAddress>&		NIL	Gocator Device identifying IP address (e.g. "127.0.0.1") or serial number (e.g. "SN:17335")
→	inInputQueueSize	int	1 - 50000		Number of incoming frames that can be buffered before the application is able to process them
→	inTimeout	int	10 - ∞	100	
→	inSectionID	int	0 - ∞		
←	outSectionData	Conditional<Array<Point2D>>&			
←	outSectionIntensity	Conditional<Profile>&			
←	outXScale	Conditional<real>&			X scale in mm
←	outXOffset	Conditional<real>&			X offset in mm
←	outZScale	Conditional<real>&			Z scale in mm
←	outZOffset	Conditional<real>&			Z offset in mm
←	outFrameInfo	Conditional<GocatorFrameInfo>&			

Remarks

Device identification

When there is only one device connected, the field **inAddress** can be set to Auto. In this situation, the first available device will be used.

inAddress can be used to pick one of multiple devices connected to the computer. **inAddress** can be set to:

- **IP Address** - Gocator Device identifying IP address (e.g. "127.0.0.1")
- **Serial Number** - Gocator Device serial number with "SN:" prefix (e.g. "SN:17335")

Gocator emulator

This filter was tested with **Gocator emulator in version 6.1.20.8**.

Gocator_GrabSurface

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures Surface from Gocator device.

Syntax

```
bool avl::Gocator_GrabSurface
(
 Gocator_State& ioState,
 const atl::Optional<avl::GocatorAddress>& inAddress,
 int inInputQueueSize,
 avl::Surface& outSurface,
 atl::Conditional<avl::Image>& outSurfaceIntensity,
 avl::GocatorFrameInfo& outFrameInfo
)
```

Parameters

	Name	Type	Range	Default	Description
↻	ioState	Gocator_State&			Object used to maintain state of the function.
→	inAddress	const Optional<GocatorAddress>&		NIL	Gocator Device identifying IP address (e.g. "127.0.0.1") or serial number (e.g. "SN:17335")
→	inInputQueueSize	int	1 - 50000		Number of incoming frames that can be buffered before the application is able to process them
←	outSurface	Surface&			
←	outSurfaceIntensity	Conditional<Image>&			
←	outFrameInfo	GocatorFrameInfo&			

Remarks

Device identification

When there is only one device connected, the field **inAddress** can be set to Auto. In this situation, the first available device will be used.

inAddress can be used to pick one of multiple devices connected to the computer. **inAddress** can be set to:

- **IP Address** - Gocator Device identifying IP address (e.g. "127.0.0.1")
- **Serial Number** - Gocator Device serial number with "SN:" prefix (e.g. "SN:17335")

Gocator emulator

This filter was tested with **Gocator emulator in version 6.1.20.8**.

See Also

- Gocator_StartAcquisition – Initializes and starts image acquisition in a Gocator device.
- Gocator_GrabSurface_WithTimeout – Captures Surface from Gocator device with timeout; returns Nil if no data comes in the specified time.
- Gocator_GrabImages – Captures images from Gocator device.
- Gocator_GrabImages_WithTimeout – Captures images from Gocator device.
- Gocator_GrabProfile – Captures profile from Gocator device.
- Gocator_GrabProfile_WithTimeout – Captures profile from Gocator device.
- Gocator_GrabUniformProfile – Captures uniform profile from Gocator device.
- Gocator_GrabUniformProfile_WithTimeout – Captures uniform profile from Gocator device.

Gocator_GrabSurface_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures Surface from Gocator device with timeout; returns Nil if no data comes in the specified time.

Syntax

```
bool avl::Gocator_GrabSurface_WithTimeout
(
 Gocator_State& ioState,
 const atl::Optional<avl::GocatorAddress>& inAddress,
 int inInputQueueSize,
 int inTimeout,
 atl::Conditional<avl::Surface>& outSurface,
 atl::Conditional<avl::Image>& outSurfaceIntensity,
 atl::Conditional<avl::GocatorFrameInfo>& outFrameInfo
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	Gocator_State&			Object used to maintain state of the function.
→	inAddress	const Optional<GocatorAddress>&		NIL	Gocator Device identifying IP address (e.g. "127.0.0.1") or serial number (e.g. "SN:17335")
→	inInputQueueSize	int	1 - 50000		Number of incoming frames that can be buffered before the application is able to process them
→	inTimeout	int	10 - ∞	100	Maximum time to wait for data in milliseconds
←	outSurface	Conditional<Surface>&			
←	outSurfaceIntensity	Conditional<Image>&			
←	outFrameInfo	Conditional<GocatorFrameInfo>&			

Remarks

Device identification

When there is only one device connected, the field **inAddress** can be set to Auto. In this situation, the first available device will be used.

inAddress can be used to pick one of multiple devices connected to the computer. **inAddress** can be set to:

- **IP Address** - Gocator Device identifying IP address (e.g. "127.0.0.1")
- **Serial Number** - Gocator Device serial number with "SN:" prefix (e.g. "SN:17335")

Gocator emulator

This filter was tested with **Gocator emulator in version 6.1.20.8**.

See Also

- Gocator_StartAcquisition – Initializes and starts image acquisition in a Gocator device.
- Gocator_GrabSurface – Captures Surface from Gocator device.
- Gocator_GrabImages – Captures images from Gocator device.
- Gocator_GrabImages_WithTimeout – Captures images from Gocator device.
- Gocator_GrabProfile – Captures profile from Gocator device.
- Gocator_GrabProfile_WithTimeout – Captures profile from Gocator device.
- Gocator_GrabUniformProfile – Captures uniform profile from Gocator device.
- Gocator_GrabUniformProfile_WithTimeout – Captures uniform profile from Gocator device.

Gocator_GrabUniformProfile

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures uniform profile from Gocator device.

Syntax

```
bool avl::Gocator_GrabUniformProfile
(
 Gocator_State& ioState,
 const atl::Optional<avl::GocatorAddress>& inAddress,
 int inInputQueueSize,
 avl::Profile& outProfileData,
 atl::Conditional<avl::Profile>& outProfileIntensity,
 atl::real& outZScale,
 atl::real& outZOffset,
 avl::GocatorFrameInfo& outFrameInfo
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Gocator_State&			Object used to maintain state of the function.
	inAddress	const Optional<GocatorAddress>&		NIL	Gocator Device identifying IP address (e.g. "127.0.0.1") or serial number (e.g. "SN:17335")
	inInputQueueSize	int	1 - 50000		Number of incoming frames that can be buffered before the application is able to process them
	outProfileData	Profile&			
	outProfileIntensity	Conditional<Profile>&			
	outZScale	real&			Z scale in mm
	outZOffset	real&			Z offset in mm
	outFrameInfo	GocatorFrameInfo&			

Remarks

Device identification

When there is only one device connected, the field **inAddress** can be set to Auto. In this situation, the first available device will be used.

inAddress can be used to pick one of multiple devices connected to the computer. **inAddress** can be set to:

- **IP Address** - Gocator Device identifying IP address (e.g. "127.0.0.1")
- **Serial Number** - Gocator Device serial number with "SN:" prefix (e.g. "SN:17335")

Gocator emulator

This filter was tested with **Gocator emulator in version 6.1.20.8**.

See Also

- Gocator_StartAcquisition – Initializes and starts image acquisition in a Gocator device.
- Gocator_GrabSurface – Captures Surface from Gocator device.
- Gocator_GrabSurface_WithTimeout – Captures Surface from Gocator device with timeout; returns Nil if no data comes in the specified time.
- Gocator_GrabImages – Captures images from Gocator device.
- Gocator_GrabImages_WithTimeout – Captures images from Gocator device.
- Gocator_GrabProfile – Captures profile from Gocator device.
- Gocator_GrabProfile_WithTimeout – Captures profile from Gocator device.
- Gocator_GrabUniformProfile_WithTimeout – Captures uniform profile from Gocator device.

Gocator_GrabUniformProfile_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures uniform profile from Gocator device.

Syntax

```
bool avl::Gocator_GrabUniformProfile_WithTimeout
(
 Gocator_State& ioState,
 const atl::Optional<avl::GocatorAddress>& inAddress,
 int inInputQueueSize,
 int inTimeout,
 atl::Conditional<avl::Profile>& outProfileData,
 atl::Conditional<avl::Profile>& outProfileIntensity,
 atl::Conditional<atl::real>& outZScale,
 atl::Conditional<atl::real>& outZOffset,
 atl::Conditional<avl::GocatorFrameInfo>& outFrameInfo
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	Gocator_State&			Object used to maintain state of the function.
→	inAddress	const Optional<GocatorAddress>&		NIL	Gocator Device identifying IP address (e.g. "127.0.0.1") or serial number (e.g. "SN:17335")
→	inInputQueueSize	int	1 - 50000		Number of incoming frames that can be buffered before the application is able to process them
→	inTimeout	int	10 - ∞	100	
←	outProfileData	Conditional<Profile>&			
←	outProfileIntensity	Conditional<Profile>&			
←	outZScale	Conditional<real>&			Z scale in mm
←	outZOffset	Conditional<real>&			Z offset in mm
←	outFrameInfo	Conditional<GocatorFrameInfo>&			

Remarks

Device identification

When there is only one device connected, the field **inAddress** can be set to Auto. In this situation, the first available device will be used.

inAddress can be used to pick one of multiple devices connected to the computer. **inAddress** can be set to:

- **IP Address** - Gocator Device identifying IP address (e.g. "127.0.0.1")
- **Serial Number** - Gocator Device serial number with "SN:" prefix (e.g. "SN:17335")

Gocator emulator

This filter was tested with **Gocator emulator in version 6.1.20.8**.

See Also

- Gocator_StartAcquisition – Initializes and starts image acquisition in a Gocator device.
- Gocator_GrabSurface – Captures Surface from Gocator device.
- Gocator_GrabSurface_WithTimeout – Captures Surface from Gocator device with timeout; returns Nil if no data comes in the specified time.
- Gocator_GrabImages – Captures images from Gocator device.
- Gocator_GrabImages_WithTimeout – Captures images from Gocator device.
- Gocator_GrabProfile – Captures profile from Gocator device.
- Gocator_GrabProfile_WithTimeout – Captures profile from Gocator device.
- Gocator_GrabUniformProfile – Captures uniform profile from Gocator device.

Gocator_LoadJob

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Loads new job.

Syntax

```
void avl::Gocator_LoadJob
(
 Gocator_State& ioState,
 const atl::Optional<avl::GocatorAddress>& inAddress,
 const atl::String& inJobName
)
```

Parameters

	Name	Type	Default	Description
	ioState	Gocator_State&		Object used to maintain state of the function.
	inAddress	const Optional<GocatorAddress>&	NIL	Gocator Device identifying IP address (e.g. "127.0.0.1") or serial number (e.g. "SN:17335")
	inJobName	const String&		Job name. The name must end with: ".job"

Remarks

Device identification

When there is only one device connected, the field **inAddress** can be set to Auto. In this situation, the first available device will be used.

inAddress can be used to pick one of multiple devices connected to the computer. **inAddress** can be set to:

- **IP Address** - Gocator Device identifying IP address (e.g. "127.0.0.1")
- **Serial Number** - Gocator Device serial number with "SN:" prefix (e.g. "SN:17335")

Gocator emulator

This filter was tested with **Gocator emulator in version 6.1.20.8**.

Gocator_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Initializes and starts image acquisition in a Gocator device.

Syntax

```
void avl::Gocator_StartAcquisition
(
 Gocator_State& ioState,
 const atl::Optional<avl::GocatorAddress>& inAddress,
 int inInputQueueSize,
 atl::Optional<avl::GocatorTriggerSource::Type> inTriggerSource
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Gocator_State&			Object used to maintain state of the function.
	inAddress	const Optional<GocatorAddress>&		NIL	Gocator Device identifying IP address (e.g. "127.0.0.1") or serial number (e.g. "SN:17335")
	inInputQueueSize	int	1 - 50000		Number of incoming frames that can be buffered before the application is able to process them
	inTriggerSource	Optional<GocatorTriggerSource::Type>		NIL	Configure Gocator trigger source

Remarks

Device identification

When there is only one device connected, the field **inAddress** can be set to Auto. In this situation, the first available device will be used.

inAddress can be used to pick one of multiple devices connected to the computer. **inAddress** can be set to:

- **IP Address** - Gocator Device identifying IP address (e.g. "127.0.0.1")
- **Serial Number** - Gocator Device serial number with "SN:" prefix (e.g. "SN:17335")

Gocator emulator

This filter was tested with **Gocator emulator in version 6.1.20.8**.

See Also

- Gocator_GrabSurface – Captures Surface from Gocator device.
- Gocator_GrabSurface_WithTimeout – Captures Surface from Gocator device with timeout; returns Nil if no data comes in the specified time.
- Gocator_GrabImages – Captures images from Gocator device.
- Gocator_GrabImages_WithTimeout – Captures images from Gocator device.
- Gocator_GrabProfile – Captures profile from Gocator device.
- Gocator_GrabProfile_WithTimeout – Captures profile from Gocator device.
- Gocator_GrabUniformProfile – Captures uniform profile from Gocator device.
- Gocator_GrabUniformProfile_WithTimeout – Captures uniform profile from Gocator device.

Gocator_StopAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Stops image acquisition in a Gocator device.

Syntax

```
void avl::Gocator_StopAcquisition
(
 Gocator_State& ioState,
 const atl::Optional<avl::GocatorAddress>& inAddress
)
```

Parameters

	Name	Type	Default	Description
	ioState	Gocator_State&		Object used to maintain state of the function.
	inAddress	const Optional<GocatorAddress>&	NIL	Gocator Device identifying IP address (e.g. "127.0.0.1") or serial number (e.g. "SN:17335")

Remarks

Device identification

When there is only one device connected, the field **inAddress** can be set to Auto. In this situation, the first available device will be used.

inAddress can be used to pick one of multiple devices connected to the computer. **inAddress** can be set to:

- **IP Address** - Gocator Device identifying IP address (e.g. "127.0.0.1")
- **Serial Number** - Gocator Device serial number with "SN:" prefix (e.g. "SN:17335")

Gocator emulator

This filter was tested with **Gocator emulator in version 6.1.20.8**.

159. Hikrobot

Table of content:

Hikrobot_GetFirmwareVersion

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Get Hikrobot SDK version.

Syntax

```
void avl::Hikrobot_GetFirmwareVersion
(
 Hikrobot_State& ioState,
 atl::Optional<int> inDeviceID,
 atl::String& outFirmwareVersion,
 atl::String& outFirmwareDate
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hikrobot_State&			Object used to maintain state of the function.
	inDeviceID	Optional<int>	1 - ∞	NIL	Device identification number - COM port number.
	outFirmwareVersion	String&			Firmware version.
	outFirmwareDate	String&			Firmware date formatted as YYYY-mm-dd.

Remarks

This filter allows to check the firmware version of the device.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Hikrobot VC4000 SDK.

Add DLL path to system environment variable is required.

In some computer models, digital inputs and outputs may operate in reverse logic (true is low signal level). Please check your device.

Recommended Hikrobot VC4000 SDK version for Aurora Vision Studio usage is **1.2.0**.

Device identification

inDeviceID field can be used to pick one of multiple I/Os connected to computer. DeviceID can be set to:

- **COM port number** - the number of the COM port used to communicate with the I/O.

See Also

- Hikrobot_GetSdkVersion – Get Hikrobot IO firmware version.

Hikrobot_GetInputLevel

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Get input level information. When edge detection is on, the interface cannot be used to obtain the level.

Syntax

```
void avl::Hikrobot_GetInputLevel
(
 Hikrobot_State& ioState,
 atl::Optional<int> inDeviceID,
 avl::HikrobotPortNumber::Type inPortNumber,
 bool& outLevel
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hikrobot_State&			Object used to maintain state of the function.
	inDeviceID	Optional<int>	1 - ∞	NIL	Device identification number - COM port number.
	inPortNumber	HikrobotPortNumber::Type			Device port number reference.
	outLevel	bool&			Logic level information for the specified port.

Remarks

This filter allows to get the input level information of a specific port of the device. When edge detection is on, the interface cannot be used to obtain the level.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Hikrobot VC4000 SDK.

Add DLL path to system environment variable is required.

In some computer models, digital inputs and outputs may operate in reverse logic (true is low signal level). Please check your device.

Recommended Hikrobot VC4000 SDK version for Aurora Vision Studio usage is **1.2.0**.

Device identification

inDeviceID field can be used to pick one of multiple I/Os connected to computer. DeviceID can be set to:

- **COM port number** - the number of the COM port used to communicate with the I/O.

Hikrobot_GetInputParameters

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Get input parameters.

Syntax

```
void avl::Hikrobot_GetInputParameters
(
 Hikrobot_State& ioState,
 atl::Optional<int> inDeviceID,
 avl::HikrobotPortNumber::Type inPortNumber,
 int& outEnableState,
 avl::HikrobotEdgeType::Type& outTriggerEdge,
 int& outTriggerDelay,
 int& outDebounceTime
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hikrobot_State&			Object used to maintain state of the function.
	inDeviceID	Optional<int>	1 - ∞	NIL	Device identification number - COM port number.
	inPortNumber	HikrobotPortNumber::Type			Device port number reference.
	outEnableState	int&			Enable state (1 is on, 0 is off).
	outTriggerEdge	HikrobotEdgeType::Type&			Trigger edge.
	outTriggerDelay	int&			Trigger delay [ms].
	outDebounceTime	int&			Contact debounce time [ms].

Remarks

This filter allows to get the input parameters of a specific port of the device. The parameters are enable state, trigger edge, trigger delay and contact debounce time.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Hikrobot VC4000 SDK.

Add DLL path to system environment variable is required.

In some computer models, digital inputs and outputs may operate in reverse logic (true is low signal level). Please check your device.

Recommended Hikrobot VC4000 SDK version for Aurora Vision Studio usage is **1.2.0**.

Device identification

inDeviceID field can be used to pick one of multiple I/Os connected to computer. DeviceID can be set to:

- **COM port number** - the number of the COM port used to communicate with the I/O.

See Also

- Hikrobot_SetInputParameters – Set input parameters.
- Hikrobot_GetOutputParameters – Get output parameters.

Hikrobot_GetLightParameters

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Get light parameters.

Syntax

```
void avl::Hikrobot_GetLightParameters
(
 Hikrobot_State& ioState,
 atl::Optional<int> inDeviceID,
 avl::HikrobotLightPortNumber::Type inPortNumber,
 int& outLightValue,
 avl::HikrobotLightState::Type& outLightState,
 avl::HikrobotEdgeType::Type& outTriggerEdge,
 int& outDurationTime
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hikrobot_State&			Object used to maintain state of the function.
	inDeviceID	Optional<int>	1 - ∞	NIL	Device identification number - COM port number.
	inPortNumber	HikrobotLightPortNumber::Type			Device light port number reference.
	outLightValue	int&			Light brightness value [0-100].
	outLightState	HikrobotLightState::Type&			Light state after triggering.
	outTriggerEdge	HikrobotEdgeType::Type&			Trigger edge.
	outDurationTime	int&			Duration time [ms].

Remarks

This filter allows to get the parameters of a specific light port of the device. The parameters are brightness value, state after triggering, trigger edge and duration time.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Hikrobot VC4000 SDK.

Add DLL path to system environment variable is required.

In some computer models, digital inputs and outputs may operate in reverse logic (true is low signal level). Please check your device.

Recommended Hikrobot VC4000 SDK version for Aurora Vision Studio usage is **1.2.0**.

Device identification

inDeviceID field can be used to pick one of multiple I/Os connected to computer. DeviceID can be set to:

- **COM port number** - the number of the COM port used to communicate with the I/O.

Hikrobot_GetOutputParameters

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Get output parameters.

Syntax

```
void avl::Hikrobot_GetOutputParameters
(
 Hikrobot_State& ioState,
 atl::Optional<int> inDeviceID,
 avl::HikrobotPortNumber::Type inPortNumber,
 avl::HikrobotPatternOut::Type& outMode,
 int& outPulseWidth,
 int& outPulsePeriod,
 int& outPulseDuration,
 bool& outLevel
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hikrobot_State&			Object used to maintain state of the function.
	inDeviceID	Optional<int>	1 - ∞	NIL	Device identification number - COM port number.
	inPortNumber	HikrobotPortNumber::Type			Device port number reference.
	outMode	HikrobotPatternOut::Type&			Output mode: Single pulse or PWM.
	outPulseWidth	int&			PWM pulse width [ms].
	outPulsePeriod	int&			PWM pulse period [ms].
	outPulseDuration	int&			PWM pulse duration [ms].
	outLevel	bool&			Effective level (there may be reversed logic).

Remarks

This filter allows to get the output parameters of a specific port of the device. The parameters are output mode (Single pulse or PWM), PWM pulse width, PWM pulse period, PWM pulse duration and effective level.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Hikrobot VC4000 SDK.

Add DLL path to system environment variable is required.

In some computer models, digital inputs and outputs may operate in reverse logic (true is low signal level). Please check your device.

Recommended Hikrobot VC4000 SDK version for Aurora Vision Studio usage is **1.2.0**.

Device identification

inDeviceID field can be used to pick one of multiple I/Os connected to computer. DeviceID can be set to:

- **COM port number** - the number of the COM port used to communicate with the I/O.

See Also

- Hikrobot_SetOutputParameters – Set output parameters.
- Hikrobot_GetInputParameters – Get input parameters.

Hikrobot_GetSdkVersion

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Get Hikrobot IO firmware version.

Syntax

```
void avl::Hikrobot_GetSdkVersion
(
 Hikrobot_State& ioState,
 atl::Optional<int> inDeviceID,
 atl::String& outSdkVersion,
 atl::String& outSdkDate
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hikrobot_State&			Object used to maintain state of the function.
	inDeviceID	Optional<int>	1 - ∞	NIL	Device identification number - COM port number.
	outSdkVersion	String&			SDK version.
	outSdkDate	String&			SDK date formatted as YYYY-mm-dd.

Remarks

This filter allows to check the installed SDK version. It can also be used for testing purposes, e.g. to see whether the SDK has been properly configured in the operating system.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Hikrobot VC4000 SDK.

Add DLL path to system environment variable is required.

In some computer models, digital inputs and outputs may operate in reverse logic (true is low signal level). Please check your device.

Recommended Hikrobot VC4000 SDK version for Aurora Vision Studio usage is **1.2.0**.

Device identification

inDeviceID field can be used to pick one of multiple I/Os connected to computer. DeviceID can be set to:

- **COM port number** - the number of the COM port used to communicate with the I/O.

See Also

- Hikrobot_GetFirmwareVersion – Get Hikrobot SDK version.

Hikrobot_Reboot

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reboot the I/O device.

Syntax

```
void avl::Hikrobot_Reboot
(
 Hikrobot_State& ioState,
 atl::Optional<int> inDeviceID
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hikrobot_State&			Object used to maintain state of the function.
	inDeviceID	Optional<int>	1 - ∞	NIL	Device identification number - COM port number.

Remarks

This filter allows to reboot the I/O hardware of the device.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Hikrobot VC4000 SDK.

Add DLL path to system environment variable is required.

In some computer models, digital inputs and outputs may operate in reverse logic (true is low signal level). Please check your device.

Recommended Hikrobot VC4000 SDK version for Aurora Vision Studio usage is **1.2.0**.

Device identification

inDeviceID field can be used to pick one of multiple I/Os connected to computer. DeviceID can be set to:

- **COM port number** - the number of the COM port used to communicate with the I/O.

Hikrobot_ResetAllParameters

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reset all I/O parameters to default values.

Syntax

```
void avl::Hikrobot_ResetAllParameters
(
 Hikrobot_State& ioState,
 atl::Optional<int> inDeviceID
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hikrobot_State&			Object used to maintain state of the function.
	inDeviceID	Optional<int>	1 - ∞	NIL	Device identification number - COM port number.

Remarks

This filter allows to reset all parameters of the I/O, including the configuration of inputs, outputs and lights.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Hikrobot VC4000 SDK.

Add DLL path to system environment variable is required.

In some computer models, digital inputs and outputs may operate in reverse logic (true is low signal level). Please check your device.

Recommended Hikrobot VC4000 SDK version for Aurora Vision Studio usage is **1.2.0**.

Device identification

inDeviceID field can be used to pick one of multiple I/Os connected to computer. DeviceID can be set to:

- **COM port number** - the number of the COM port used to communicate with the I/O.

Hikrobot_SetInputParameters

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Set input parameters.

Syntax

```
void avl::Hikrobot_SetInputParameters
(
 Hikrobot_State& ioState,
 atl::Optional<int> inDeviceID,
 avl::HikrobotPortNumber::Type inPortNumber,
 int inEnableState,
 avl::HikrobotEdgeType::Type inTriggerEdge,
 int inTriggerDelay,
 int inDebounceTime
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hikrobot_State&			Object used to maintain state of the function.
	inDeviceID	Optional<int>	1 - ∞	NIL	Device identification number - COM port number.
	inPortNumber	HikrobotPortNumber::Type			Device port number reference.
	inEnableState	int	0 - 1	0	Enable state (1 is on, 0 is off).
	inTriggerEdge	HikrobotEdgeType::Type		MMoEdgeRising	Trigger edge.
	inTriggerDelay	int	0 - 1000	20	Trigger delay [ms].
	inDebounceTime	int	0 - 1000	20	Contact debounce time [ms].

Remarks

This filter allows to set the input parameters of a specific port of the device. The parameters are enable state, trigger edge, trigger delay and contact debounce time.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Hikrobot VC4000 SDK.

Add DLL path to system environment variable is required.

In some computer models, digital inputs and outputs may operate in reverse logic (true is low signal level). Please check your device.

Recommended Hikrobot VC4000 SDK version for Aurora Vision Studio usage is **1.2.0**.

Device identification

inDeviceID field can be used to pick one of multiple I/Os connected to computer. DeviceID can be set to:

- **COM port number** - the number of the COM port used to communicate with the I/O.

See Also

- Hikrobot_GetInputParameters – Get input parameters.
- Hikrobot_SetOutputParameters – Set output parameters.

Hikrobot_SetLightParameters

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Set light parameters.

Syntax

```
void avl::Hikrobot_SetLightParameters
(
 Hikrobot_State& ioState,
 atl::Optional<int> inDeviceID,
 avl::HikrobotLightPortNumber::Type inPortNumber,
 int inLightValue,
 avl::HikrobotLightState::Type inLightState,
 avl::HikrobotEdgeType::Type inTriggerEdge,
 int inDurationTime
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hikrobot_State&			Object used to maintain state of the function.
	inDeviceID	Optional<int>	1 - ∞	NIL	Device identification number - COM port number.
	inPortNumber	HikrobotLightPortNumber::Type			Device light port number reference.
	inLightValue	int	0 - 100	20	Light brightness value [0-100].
	inLightState	HikrobotLightState::Type		MvoLightStateOff	Light state after triggering.
	inTriggerEdge	HikrobotEdgeType::Type		MvoEdgeRising	Trigger edge.
	inDurationTime	int	0 - 65535	200	Duration time [ms].

Remarks

This filter allows to set the parameters of a specific light port of the device. The parameters are brightness value, state after triggering, trigger edge and duration time.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Hikrobot VC4000 SDK.

Add DLL path to system environment variable is required.

In some computer models, digital inputs and outputs may operate in reverse logic (true is low signal level). Please check your device.

Recommended Hikrobot VC4000 SDK version for Aurora Vision Studio usage is **1.2.0**.

Device identification

inDeviceID field can be used to pick one of multiple I/Os connected to computer. DeviceID can be set to:

- **COM port number** - the number of the COM port used to communicate with the I/O.

Hikrobot_SetOutputEnable

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Turn on output enable.

Syntax

```
void avl::Hikrobot_SetOutputEnable
(
 Hikrobot_State& ioState,
 atl::Optional<int> inDeviceID,
 avl::HikrobotPortNumber::Type inPortNumber,
 avl::HikrobotEnableType::Type inEnableType
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hikrobot_State&			Object used to maintain state of the function.
	inDeviceID	Optional<int>	1 - ∞	NIL	Device identification number - COM port number.
	inPortNumber	HikrobotPortNumber::Type			Device port number reference.
	inEnableType	HikrobotEnableType::Type			Start or stop enabling.

Remarks

This filter allows to enable or disable output of a specific port of the device.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Hikrobot VC4000 SDK.

Add DLL path to system environment variable is required.

In some computer models, digital inputs and outputs may operate in reverse logic (true is low signal level). Please check your device.

Recommended Hikrobot VC4000 SDK version for Aurora Vision Studio usage is **1.2.0**.

Device identification

inDeviceID field can be used to pick one of multiple I/Os connected to computer. DeviceID can be set to:

- **COM port number** - the number of the COM port used to communicate with the I/O.

Hikrobot_SetOutputParameters

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Set output parameters.

Syntax

```
void avl::Hikrobot_SetOutputParameters
(
 Hikrobot_State& ioState,
 atl::Optional<int> inDeviceID,
 avl::HikrobotPortNumber::Type inPortNumber,
 avl::HikrobotPatternOut::Type inMode,
 int inPulseWidth,
 int inPulsePeriod,
 int inPulseDuration,
 bool inLevel
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hikrobot_State&			Object used to maintain state of the function.
	inDeviceID	Optional<int>	1 - ∞	NIL	Device identification number - COM port number.
	inPortNumber	HikrobotPortNumber::Type			Device port number reference.
	inMode	HikrobotPatternOut::Type		MoPatternSingle	Output mode: Single pulse or PWM.
	inPulseWidth	int	1 - 65535		PWM pulse width [ms].
	inPulsePeriod	int	1 - 65535	1	PWM pulse period [ms].
	inPulseDuration	int	1 - 65535	20	PWM pulse duration [ms].
	inLevel	bool		True	Effective level (there may be reversed logic).

Remarks

This filter allows to set the output parameters of a specific port of the device. The parameters are output mode (Single pulse or PWM), PWM pulse width, PWM pulse period, PWM pulse duration and effective level.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Hikrobot VC4000 SDK.

Add DLL path to system environment variable is required.

In some computer models, digital inputs and outputs may operate in reverse logic (true is low signal level). Please check your device.

Recommended Hikrobot VC4000 SDK version for Aurora Vision Studio usage is **1.2.0**.

Device identification

inDeviceID field can be used to pick one of multiple I/Os connected to computer. DeviceID can be set to:

- **COM port number** - the number of the COM port used to communicate with the I/O.

See Also

- Hikrobot_GetOutputParameters – Get output parameters.
- Hikrobot_SetInputParameters – Set input parameters.

160. Hikvision

Table of content:

Hikvision_ConfigureStrobe

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets strobe parameters.

Syntax

```
void avl::Hikvision_ConfigureStrobe
(
 Hikvision_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::HikvisionLine::Type inLineSelector,
 avl::HikvisionEvent::Type inLineSource,
 atl::Optional<bool> inLineInverter,
 bool inStrobeEnabled,
 atl::Optional<int> inStrobeDuration,
 atl::Optional<int> inStrobeDelay
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hikvision_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Camera serial number, name or IP address, Auto = first camera
	inLineSelector	HikvisionLine::Type			Strobe output line
	inLineSource	HikvisionEvent::Type		ExposureActive	Strobe source
	inLineInverter	Optional<bool>		NIL	True if inverting output signal
	inStrobeEnabled	bool			True if using strobe on selected line
	inStrobeDuration	Optional<int>	0 - ∞	NIL	Duration of strobe pulse in microseconds
	inStrobeDelay	Optional<int>	0 - ∞	NIL	Delay of strobe pulse in microseconds

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install Hikvision Machine Vision Software with camera dedicated drivers.

Hikvision Machine Vision Software can be downloaded from the following website:
https://www.hikrobotics.com/en/machinevision/service/download?module=0 (registration may be required).

Add DLL path to system environment variable may be required.

Recommended Hikvision Machine Vision Software version for Aurora Vision Studio usage is **4.0.1** (DLL version **4.0.0.5**).

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to computer. DeviceID can be set to:

- **Serial number** - serial number of the device which can be viewed in the MVS software from Hikvision under Feature Tree / Device Control / Device Serial Number. Please note that this serial number may differ from the one printed on the device housing.
- **Camera name** - user-defined ID which can be set in the MVS software from Hikvision under Feature Tree / Device Control / Device User ID.
- **IP address** - ip address of the camera on the local network.

Setting bandwidth for GigEVision cameras

Bandwidth of Hikvision GigEVision cameras needs to be configured before use, otherwise packet loss and image artifacts may occur.

Open MVS software from Hikvision and connect to the camera. Open Settings / Options menu and go to Network tab. Bandwidth Control slider will be shown. To preserve new value of bandwidth after disconnecting power from the camera, you will need to save it using User Set Control parameters in Features tab.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Hikvision_GrabImage – Captures an image from a Hikvision device.
- Hikvision_GrabImage_WithTimeout – Captures an image from a Hikvision device.
- Hikvision_ConfigureTrigger – Sets triggering parameters.
- Hikvision_GenerateSoftwareTrigger – Generates software trigger in Hikvision device, trigger source should be set to 'Software'.

Hikvision_ConfigureTrigger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets triggering parameters.

Syntax

```
void avl::Hikvision_ConfigureTrigger
(
 Hikvision_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 atl::Optional<bool> inTriggerEnabled,
 atl::Optional<avl::HikvisionEvent::Type> inTriggerSelector,
 atl::Optional<avl::HikvisionTriggerSource::Type> inTriggerSource,
 atl::Optional<avl::HikvisionTriggerActivation::Type> inTriggerActivation,
 atl::Optional<float> inTriggerDelay
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hikvision_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Camera serial number, name or IP address, Auto = first camera
	inTriggerEnabled	Optional<bool>		False	True if using trigger
	inTriggerSelector	Optional<HikvisionEvent::Type>		FrameBurstStart	Triggering event selection
	inTriggerSource	Optional<HikvisionTriggerSource::Type>		Line0	Trigger source
	inTriggerActivation	Optional<HikvisionTriggerActivation::Type>		RisingEdge	Trigger polarity
	inTriggerDelay	Optional<float>	0.0 - ∞	0.0f	Delay of trigger in microseconds

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install Hikvision Machine Vision Software with camera dedicated drivers.

Hikvision Machine Vision Software can be downloaded from the following website:
https://www.hikrobotics.com/en/machinevision/service/download?module=0 (registration may be required).

Add DLL path to system environment variable may be required.

Recommended Hikvision Machine Vision Software version for Aurora Vision Studio usage is **4.0.1** (DLL version **4.0.0.5**).

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to computer. DeviceID can be set to:

- **Serial number** - serial number of the device which can be viewed in the MVS software from Hikvision under Feature Tree / Device Control / Device Serial Number. Please note that this serial number may differ from the one printed on the device housing.
- **Camera name** - user-defined ID which can be set in the MVS software from Hikvision under Feature Tree / Device Control / Device User ID.
- **IP address** - ip address of the camera on the local network.

Setting bandwidth for GigEVision cameras

Bandwidth of Hikvision GigEVision cameras needs to be configured before use, otherwise packet loss and image artifacts may occur.

Open MVS software from Hikvision and connect to the camera. Open Settings / Options menu and go to Network tab. Bandwidth Control slider will be shown. To preserve new value of bandwidth after disconnecting power from the camera, you will need to save it using User Set Control parameters in Features tab.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Hikvision_GrabImage – Captures an image from a Hikvision device.
- Hikvision_GrabImage_WithTimeout – Captures an image from a Hikvision device.
- Hikvision_GenerateSoftwareTrigger – Generates software trigger in Hikvision device, trigger source should be set to 'Software'.
- Hikvision_ConfigureStrobe – Sets strobe parameters.

Hikvision_GenerateSoftwareTrigger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Generates software trigger in Hikvision device, trigger source should be set to 'Software'.

Syntax

```
void avl::Hikvision_GenerateSoftwareTrigger
(
 Hikvision_State& ioState,
 atl::Optional<const atl::String&> inDeviceID
)
```

Parameters

	Name	Type	Default	Description
	ioState	Hikvision_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Camera serial number, name or IP address, Auto = first camera

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install Hikvision Machine Vision Software with camera dedicated drivers.

Hikvision Machine Vision Software can be downloaded from the following website:
https://www.hikrobotics.com/en/machinevision/service/download?module=0 (registration may be required).

Add DLL path to system environment variable may be required.

Recommended Hikvision Machine Vision Software version for Aurora Vision Studio usage is **4.0.1** (DLL version **4.0.0.5**).

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to computer. DeviceID can be set to:

- **Serial number** - serial number of the device which can be viewed in the MVS software from Hikvision under Feature Tree / Device Control / Device Serial Number. Please note that this serial number may differ from the one printed on the device housing.
- **Camera name** - user-defined ID which can be set in the MVS software from Hikvision under Feature Tree / Device Control / Device User ID.
- **IP address** - ip address of the camera on the local network.

Setting bandwidth for GigEVision cameras

Bandwidth of Hikvision GigEVision cameras needs to be configured before use, otherwise packet loss and image artifacts may occur.

Open MVS software from Hikvision and connect to the camera. Open Settings / Options menu and go to Network tab. Bandwidth Control slider will be shown. To preserve new value of bandwidth after disconnecting power from the camera, you will need to save it using User Set Control parameters in Features tab.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Hikvision_GrabImage – Captures an image from a Hikvision device.
- Hikvision_GrabImage_WithTimeout – Captures an image from a Hikvision device.
- Hikvision_ConfigureTrigger – Sets triggering parameters.
- Hikvision_ConfigureStrobe – Sets strobe parameters.

Also in AVL Lite

Hikvision_GetBoolParameter

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type Bool.

Syntax

```
void avl::Hikvision_GetBoolParameter
(
 Hikvision_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 bool& outParameterValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Hikvision_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Camera serial number, name or IP address, Auto = first camera
	inParameterName	const String&		Name of parameter
	outParameterValue	bool&		Value of parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Also in AVL Lite

Hikvision_GetEnumParameter

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type Enum.

Syntax

```
void avl::Hikvision_GetEnumParameter
(
 Hikvision_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 atl::String& outParameterValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Hikvision_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Camera serial number, name or IP address, Auto = first camera
	inParameterName	const String&		Name of parameter
	outParameterValue	String&		Value of parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Also in AVL Lite

HIK Hikvision_GetIntegerParameter

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type Integer.

Syntax

```
void avl::Hikvision_GetIntegerParameter
(
 Hikvision_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 int& outParameterValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Hikvision_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Camera serial number, name or IP address, Auto = first camera
	inParameterName	const String&		Name of parameter
	outParameterValue	int&		Value of parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Also in AVL Lite

HIK Hikvision_GetRealParameter

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type Real.

Syntax

```
void avl::Hikvision_GetRealParameter
(
 Hikvision_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 float& outParameterValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Hikvision_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Camera serial number, name or IP address, Auto = first camera
	inParameterName	const String&		Name of parameter
	outParameterValue	float&		Value of parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Also in AVL Lite

Hikvision_GetStringParameter

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type String.

Syntax

```
void avl::Hikvision_GetStringParameter
(
 Hikvision_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 atl::String& outParameterValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Hikvision_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Camera serial number, name or IP address, Auto = first camera
	inParameterName	const String&		Name of parameter
	outParameterValue	String&		Value of parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Also in AVL Lite

Hikvision_GrabImage

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an image from a Hikvision device.

Syntax

```
bool avl::Hikvision_GrabImage
(
 Hikvision_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inInputQueueSize,
 avl::HikvisionPixelFormat::Type inPixelFormat,
 atl::Optional<const avl::Box&> inRoi,
 atl::Optional<float> inFrameRate,
 bool inFrameRateAuto,
 atl::Optional<float> inExposureTime,
 bool inExposureTimeAuto,
 atl::Optional<float> inGain,
 bool inGainAuto,
 avl::Image& outImage,
 atl::Conditional<int>& outFrameID
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hikvision_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Camera serial number, name or IP address, Auto = first camera
	inInputQueueSize	int	1 - ∞	8	Sets size of image queue (on acquisition start)
	inPixelFormat	HikvisionPixelFormat::Type			Image pixel format (set on acquisition start)
	inRoi	Optional<const Box&>		NIL	Region of interest (set on acquisition start)
	inFrameRate	Optional<float>	0.0 - ∞	NIL	Sets the frame rate limit
	inFrameRateAuto	bool		True	True if frame rate is not limited
	inExposureTime	Optional<float>	0.0 - ∞	10000.0f	Sets the target exposure time in microseconds
	inExposureTimeAuto	bool			True if using automatic exposure time
	inGain	Optional<float>	0.0 - ∞	0.0f	Sets gain in dB
	inGainAuto	bool			True if using automatic gain
	outImage	Image&			Captured frame
	outFrameID	Conditional<int>&			Captured frame ID

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install Hikvision Machine Vision Software with camera dedicated drivers.

Hikvision Machine Vision Software can be downloaded from the following website:
https://www.hikrobotics.com/en/machinevision/service/download?module=0 (registration may be required).

Add DLL path to system environment variable may be required.

Recommended Hikvision Machine Vision Software version for Aurora Vision Studio usage is **4.0.1** (DLL version **4.0.0.5**).

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to computer. DeviceID can be set to:

- **Serial number** - serial number of the device which can be viewed in the MVS software from Hikvision under Feature Tree / Device Control / Device Serial Number. Please note that this serial number may differ from the one printed on the device housing.
- **Camera name** - user-defined ID which can be set in the MVS software from Hikvision under Feature Tree / Device Control / Device User ID.
- **IP address** - ip address of the camera on the local network.

Setting bandwidth for GigEVision cameras

Bandwidth of Hikvision GigEVision cameras needs to be configured before use, otherwise packet loss and image artifacts may occur.

Open MVS software from Hikvision and connect to the camera. Open Settings / Options menu and go to Network tab. Bandwidth Control slider will be shown. To preserve new value of bandwidth after disconnecting power from the camera, you will need to save it using User Set Control parameters in Features tab.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Hikvision_GrabImage_WithTimeout – Captures an image from a Hikvision device.
- Hikvision_StartAcquisition – Initializes and starts image acquisition in a device.
- Hikvision_ConfigureTrigger – Sets triggering parameters.

Hikvision_GrabImage_WithTimeout

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an image from a Hikvision device.

Syntax

```
bool avl::Hikvision_GrabImage_WithTimeout
(
 Hikvision_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inTimeout,
 int inInputQueueSize,
 avl::HikvisionPixelFormat::Type inPixelFormat,
 atl::Optional<const avl::Box&> inRoi,
 atl::Optional<float> inFrameRate,
 bool inFrameRateAuto,
 atl::Optional<float> inExposureTime,
 bool inExposureTimeAuto,
 atl::Optional<float> inGain,
 bool inGainAuto,
 atl::Conditional<avl::Image>& outImage,
 atl::Conditional<int>& outFrameID
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hikvision_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Camera serial number, name or IP address, Auto = first camera
	inTimeout	int			Maximum time to wait for frame in milliseconds
	inInputQueueSize	int	1 - ∞	8	Sets size of image queue (on acquisition start)
	inPixelFormat	HikvisionPixelFormat::Type			Image pixel format (set on acquisition start)
	inRoi	Optional<const Box&>		NIL	Region of interest (set on acquisition start)
	inFrameRate	Optional<float>	0.0 - ∞	NIL	Sets the frame rate limit
	inFrameRateAuto	bool		True	True if frame rate is not limited
	inExposureTime	Optional<float>	0.0 - ∞	10000.0f	Sets the target exposure time in microseconds
	inExposureTimeAuto	bool			True if using automatic exposure time
	inGain	Optional<float>	0.0 - ∞	0.0f	Sets gain in dB
	inGainAuto	bool			True if using automatic gain
	outImage	Conditional<Image>&			Captured image
	outFrameID	Conditional<int>&			Captured frame ID

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install Hikvision Machine Vision Software with camera dedicated drivers.

Hikvision Machine Vision Software can be downloaded from the following website:
https://www.hikrobotics.com/en/machinevision/service/download?module=0 (registration may be required).

Add DLL path to system environment variable may be required.

Recommended Hikvision Machine Vision Software version for Aurora Vision Studio usage is **4.0.1** (DLL version **4.0.0.5**).

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to computer. DeviceID can be set to:

- **Serial number** - serial number of the device which can be viewed in the MVS software from Hikvision under Feature Tree / Device Control / Device Serial Number. Please note that this serial number may differ from the one printed on the device housing.
- **Camera name** - user-defined ID which can be set in the MVS software from Hikvision under Feature Tree / Device Control / Device User ID.
- **IP address** - ip address of the camera on the local network.

Setting bandwidth for GigEVision cameras

Bandwidth of Hikvision GigEVision cameras needs to be configured before use, otherwise packet loss and image artifacts may occur.

Open MVS software from Hikvision and connect to the camera. Open Settings / Options menu and go to Network tab. Bandwidth Control slider will be shown. To preserve new value of bandwidth after disconnecting power from the camera, you will need to save it using User Set Control parameters in Features tab.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Hikvision_GrabImage – Captures an image from a Hikvision device.
- Hikvision_StartAcquisition – Initializes and starts image acquisition in a device.
- Hikvision_ConfigureTrigger – Sets triggering parameters.

Hikvision_SetBoolParameter

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type Bool.

Syntax

```
void avl::Hikvision_SetBoolParameter
(
 Hikvision_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 bool inParameterValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Hikvision_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Camera serial number, name or IP address, Auto = first camera
	inParameterName	const String&		Name of parameter
	inParameterValue	bool		Value to set to parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Also in AVL Lite

Hikvision_SetEnumParameter

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type Enum.

Syntax

```
void avl::Hikvision_SetEnumParameter
(
 Hikvision_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 const atl::String& inParameterValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Hikvision_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Camera serial number, name or IP address, Auto = first camera
	inParameterName	const String&		Name of parameter
	inParameterValue	const String&		Value to set to parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Also in AVL Lite

HIK Hikvision_SetIntegerParameter

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type Integer.

Syntax

```
void avl::Hikvision_SetIntegerParameter
(
 Hikvision_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 int inParameterValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Hikvision_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Camera serial number, name or IP address, Auto = first camera
	inParameterName	const String&		Name of parameter
	inParameterValue	int		Value to set to parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Also in AVL Lite

HIK Hikvision_SetRealParameter

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type Real.

Syntax

```
void avl::Hikvision_SetRealParameter
(
 Hikvision_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 float inParameterValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Hikvision_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Camera serial number, name or IP address, Auto = first camera
	inParameterName	const String&		Name of parameter
	inParameterValue	float		Value to set to parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Also in AVL Lite

HIK Hikvision_SetStringParameter

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type String.

Syntax

```
void avl::Hikvision_SetStringParameter
(
 Hikvision_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 const atl::String& inParameterValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Hikvision_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Camera serial number, name or IP address, Auto = first camera
	inParameterName	const String&		Name of parameter
	inParameterValue	const String&		Value to set to parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Also in AVL Lite

HIK Hikvision_StartAcquisition

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Initializes and starts image acquisition in a device.

Syntax

```
void avl::Hikvision_StartAcquisition
(
 Hikvision_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inInputQueueSize,
 avl::HikvisionPixelFormat::Type inPixelFormat,
 atl::Optional<const avl::Box&> inRoi,
 atl::Optional<float> inFrameRate,
 bool inFrameRateAuto,
 atl::Optional<float> inExposureTime,
 bool inExposureTimeAuto,
 atl::Optional<float> inGain,
 bool inGainAuto
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hikvision_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Camera serial number, name or IP address, Auto = first camera
	inInputQueueSize	int	1 - ∞	8	Sets size of image queue (on acquisition start)
	inPixelFormat	HikvisionPixelFormat::Type			Image pixel format (set on acquisition start)
	inRoi	Optional<const Box&>		NIL	Region of interest (set on acquisition start)
	inFrameRate	Optional<float>	0.0 - ∞	NIL	Sets the frame rate limit
	inFrameRateAuto	bool		True	True if frame rate is not limited
	inExposureTime	Optional<float>	0.0 - ∞	10000.0f	Sets the target exposure time in microseconds
	inExposureTimeAuto	bool			True if using automatic exposure time
	inGain	Optional<float>	0.0 - ∞	0.0f	Sets gain in dB
	inGainAuto	bool			True if using automatic gain

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install Hikvision Machine Vision Software with camera dedicated drivers.

Hikvision Machine Vision Software can be downloaded from the following website:
https://www.hikrobotics.com/en/machinevision/service/download?module=0 (registration may be required).

Add DLL path to system environment variable may be required.

Recommended Hikvision Machine Vision Software version for Aurora Vision Studio usage is **4.0.1** (DLL version **4.0.0.5**).

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to computer. DeviceID can be set to:

- **Serial number** - serial number of the device which can be viewed in the MVS software from Hikvision under Feature Tree / Device Control / Device Serial Number. Please note that this serial number may differ from the one printed on the device housing.
- **Camera name** - user-defined ID which can be set in the MVS software from Hikvision under Feature Tree / Device Control / Device User ID.
- **IP address** - ip address of the camera on the local network.

Setting bandwidth for GigEVision cameras

Bandwidth of Hikvision GigEVision cameras needs to be configured before use, otherwise packet loss and image artifacts may occur.

Open MVS software from Hikvision and connect to the camera. Open Settings / Options menu and go to Network tab. Bandwidth Control slider will be shown. To preserve new value of bandwidth after disconnecting power from the camera, you will need to save it using User Set Control parameters in Features tab.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Hikvision_GrabImage – Captures an image from a Hikvision device.
- Hikvision_GrabImage_WithTimeout – Captures an image from a Hikvision device.
- Hikvision_StopAcquisition – Stops image acquisition in a device.

Hikvision_StopAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Stops image acquisition in a device.

Syntax

```
void avl::Hikvision_StopAcquisition
(
 Hikvision_State& ioState,
 atl::Optional<const atl::String&> inDeviceID
)
```

Parameters

	Name	Type	Default	Description
	ioState	Hikvision_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Camera serial number, name or IP address, Auto = first camera

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install Hikvision Machine Vision Software with camera dedicated drivers.

Hikvision Machine Vision Software can be downloaded from the following website:
https://www.hikrobotics.com/en/machinevision/service/download?module=0 (registration may be required).

Add DLL path to system environment variable may be required.

Recommended Hikvision Machine Vision Software version for Aurora Vision Studio usage is **4.0.1** (DLL version **4.0.0.5**).

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to computer. DeviceID can be set to:

- **Serial number** - serial number of the device which can be viewed in the MVS software from Hikvision under Feature Tree / Device Control / Device Serial Number. Please note that this serial number may differ from the one printed on the device housing.
- **Camera name** - user-defined ID which can be set in the MVS software from Hikvision under Feature Tree / Device Control / Device User ID.
- **IP address** - ip address of the camera on the local network.

Setting bandwidth for GigEVision cameras

Bandwidth of Hikvision GigEVision cameras needs to be configured before use, otherwise packet loss and image artifacts may occur.

Open MVS software from Hikvision and connect to the camera. Open Settings / Options menu and go to Network tab. Bandwidth Control slider will be shown. To preserve new value of bandwidth after disconnecting power from the camera, you will need to save it using User Set Control parameters in Features tab.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Hikvision_GrabImage – Captures an image from a Hikvision device.
- Hikvision_GrabImage_WithTimeout – Captures an image from a Hikvision device.
- Hikvision_StartAcquisition – Initializes and starts image acquisition in a device.

161. Hilscher

Table of content:

- Hilscher_Driver_GetBoardInformation
- Hilscher_Driver_GetChannelInformation
- Hilscher_Driver_GetInformation
- Hilscher_Driver_Open
- Hilscher_Driver_Restart

Also in AVL Lite

Hilscher_Channel_Close

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Closes a Hilscher device channel connection.

Syntax

```
void avl::Hilscher_Channel_Close
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const bool inCloseCommunication
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inCloseCommunication	const bool		False	Close communication between card and PLC

Remarks

Driver

- Windows driver (recommended **1.5.0.0**) https://kb.hilscher.com/display/CIFXDRV/Versions+-+NXDRV-WIN
- SYCON.net for configuring slots, generating configuration files: https://kb.hilscher.com/display/SYCON/Version+History

For profinet: Firmware (recommended **3.13** series) https://kb.hilscher.com/display/PNS3V5/Version+History+-+V3.5+to+V3.x

Firmware needs to be uploaded to the card prior to opening any type of channel.
In case of failing to update the firmware, try to restart your computer.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Also in AVL Lite

Hilscher_Channel_GetSignals

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Retrieves the current slot configuration.

Syntax

```
void avl::Hilscher_Channel_GetSignals
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 atl::Array<avl::HilscherSignal>& outInputSignals,
 atl::Array<avl::HilscherSignal>& outOutputSignals
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	outInputSignals	Array<HilscherSignal>&			
	outOutputSignals	Array<HilscherSignal>&			

Hilscher_Channel_GetSlots

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Retrieves the current slot configuration.

Syntax

```
void avl::Hilscher_Channel_GetSlots
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 atl::Array<avl::HilscherSlot>& outSlots
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	Hilscher_State&			Object used to maintain state of the function.
→	inBoardName	const Optional<String>&		NIL	
→	inChannelNumber	const int	0 - 10		
←	outSlots	Array<HilscherSlot>&			

Hilscher_Channel_IndexRead_ByteBuffer

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Receives generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_IndexRead_ByteBuffer
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const int inIndex,
 const int inSize,
 avl::ByteBuffer& outData
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	Hilscher_State&			Object used to maintain state of the function.
→	inBoardName	const Optional<String>&		NIL	
→	inChannelNumber	const int	0 - 10		
→	inIndex	const int	0 - 1000		
→	inSize	const int	1 - ∞		
←	outData	ByteBuffer&			

Also in AVL Lite

Hilscher_Channel_IndexRead_SInt16

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Receives generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_IndexRead_SInt16
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const int inIndex,
 const avl::DataEndianness::Type inDataEndianness,
 int& outData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inIndex	const int	0 - 1000		
	inDataEndianness	const DataEndianness::Type		BigEndian	
	outData	int&			

Also in AVL Lite

Hilscher_Channel_IndexRead_SInt32

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Receives generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_IndexRead_SInt32
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const int inIndex,
 const avl::DataEndianness::Type inDataEndianness,
 int& outData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inIndex	const int	0 - 1000		
	inDataEndianness	const DataEndianness::Type		BigEndian	
	outData	int&			

Also in AVL Lite

Hilscher_Channel_IndexRead_SInt64

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Receives generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_IndexRead_SInt64
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const int inIndex,
 const avl::DataEndianness::Type inDataEndianness,
 atl::sint64& outData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inIndex	const int	0 - 1000		
	inDataEndianness	const DataEndianness::Type		BigEndian	
	outData	sint64&			

Also in AVL Lite

Hilscher_Channel_IndexRead_SInt8

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Receives generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_IndexRead_SInt8
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const int inIndex,
 int& outData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inIndex	const int	0 - 1000		
	outData	int&			

Also in AVL Lite

Hilscher_Channel_IndexWrite_ByteBuffer

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sends generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_IndexWrite_ByteBuffer
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const int inIndex,
 const avl::ByteBuffer& inData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inIndex	const int	0 - 1000		
	inData	const ByteBuffer&			

Also in AVL Lite

Hilscher_Channel_IndexWrite_SInt16

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sends generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_IndexWrite_SInt16
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const int inIndex,
 const avl::DataEndianness::Type inDataEndianness,
 const int inData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inIndex	const int	0 - 1000		
	inDataEndianness	const DataEndianness::Type		BigEndian	
	inData	const int	-32768 - 32767		

Hilscher_Channel_IndexWrite_SInt32

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sends generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_IndexWrite_SInt32
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const int inIndex,
 const avl::DataEndianness::Type inDataEndianness,
 const int inData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inIndex	const int	0 - 1000		
	inDataEndianness	const DataEndianness::Type		BigEndian	
	inData	const int			

Also in AVL Lite

Hilscher_Channel_IndexWrite_SInt64

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sends generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_IndexWrite_SInt64
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const int inIndex,
 const avl::DataEndianness::Type inDataEndianness,
 const atl::sint64 inData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inIndex	const int	0 - 1000		
	inDataEndianness	const DataEndianness::Type		BigEndian	
	inData	const sint64			

Hilscher_Channel_IndexWrite_SInt8

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sends generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_IndexWrite_SInt8
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const int inIndex,
 const int inData
)
```

Parameters

	Name	Type	Range	Default	Description
↻	ioState	Hilscher_State&			Object used to maintain state of the function.
→	inBoardName	const Optional<String>&		NIL	
→	inChannelNumber	const int	0 - 10		
→	inIndex	const int	0 - 1000		
→	inData	const int	-128 - 127		

Hilscher_Channel_IORead_ByteBuffer

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Receives generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_IORead_ByteBuffer
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const int inOffset,
 const int inSize,
 avl::ByteBuffer& outData
)
```

Parameters

	Name	Type	Range	Default	Description
↻	ioState	Hilscher_State&			Object used to maintain state of the function.
→	inBoardName	const Optional<String>&		NIL	
→	inChannelNumber	const int	0 - 10		
→	inOffset	const int	0 - 10000		
→	inSize	const int	1 - ∞		
←	outData	ByteBuffer&			

Hilscher_Channel_IORead_SInt16

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Receives generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_IORead_SInt16
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const int inOffset,
 const avl::DataEndianness::Type inDataEndianness,
 int& outData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inOffset	const int	0 - 10000		
	inDataEndianness	const DataEndianness::Type		BigEndian	
	outData	int&			

Remarks

Driver

- Windows driver (recommended **1.5.0.0**) https://kb.hilscher.com/display/CIFXDRV/Versions+-+NXDRV-WIN
- SYCON.net for configuring slots, generating configuration files: https://kb.hilscher.com/display/SYCON/Version+History

For profinet: Firmware (recommended **3.13** series) https://kb.hilscher.com/display/PNS3V5/Version+History+-+V3.5+to+V3.x

Firmware needs to be uploaded to the card prior to opening any type of channel.
In case of failing to update the firmware, try to restart your computer.

IO Offset (inOffset)

Offset as displayed in Hilscher software

Data direction

Slot direction is from the application perspective, different than in SYCON software.
IN: in SYCON.net is marked as output
OUT: in SYCON.net is marked as input

Data endianness

Remember to check endianness on controller and device sides, most controllers will use big endian. Data written to the card will be converted by default to network order. (to big endian)
Data read from the card will be converted to native byte order. (by default from big endian)

Hilscher_Channel_IORead_SInt32

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Receives generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_IORead_SInt32
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const int inOffset,
 const avl::DataEndianness::Type inDataEndianness,
 int& outData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inOffset	const int	0 - 10000		
	inDataEndianness	const DataEndianness::Type		BigEndian	
	outData	int&			

Remarks

Driver

- Windows driver (recommended **1.5.0.0**) https://kb.hilscher.com/display/CIFXDRV/Versions+-+NXDRV-WIN
- SYCON.net for configuring slots, generating configuration files: https://kb.hilscher.com/display/SYCON/Version+History

For profinet: Firmware (recommended **3.13** series) https://kb.hilscher.com/display/PNS3V5/Version+History+-+V3.5+to+V3.x

Firmware needs to be uploaded to the card prior to opening any type of channel.
In case of failing to update the firmware, try to restart your computer.

IO Offset (inOffset)

Offset as displayed in Hilscher software

Data direction

Slot direction is from the application perspective, different than in SYCON software.
IN: in SYCON.net is marked as output
OUT: in SYCON.net is marked as input

Data endianness

Remember to check endianness on controller and device sides, most controllers will use big endian. Data written to the card will be converted by default to network order. (to big endian)
Data read from the card will be converted to native byte order. (by default from big endian)

Hilscher_Channel_IORead_SInt64

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Receives generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_IORead_SInt64
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const int inOffset,
 const avl::DataEndianness::Type inDataEndianness,
 atl::int64& outData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inOffset	const int	0 - 10000		
	inDataEndianness	const DataEndianness::Type		BigEndian	
	outData	int64&			

Remarks

Driver

- Windows driver (recommended **1.5.0.0**) https://kb.hilscher.com/display/CIFXDRV/Versions+-+NXDRV-WIN
- SYCON.net for configuring slots, generating configuration files: https://kb.hilscher.com/display/SYCON/Version+History

For profinet: Firmware (recommended **3.13** series) https://kb.hilscher.com/display/PNS3V5/Version+History+-+V3.5+to+V3.x

Firmware needs to be uploaded to the card prior to opening any type of channel.
In case of failing to update the firmware, try to restart your computer.

IO Offset (inOffset)

Offset as displayed in Hilscher software

Data direction

Slot direction is from the application perspective, different than in SYCON software.
IN: in SYCON.net is marked as output
OUT: in SYCON.net is marked as input

Data endianness

Remember to check endianness on controller and device sides, most controllers will use big endian. Data written to the card will be converted by default to network order. (to big endian)
Data read from the card will be converted to native byte order. (by default from big endian)

Hilscher_Channel_IORead_SInt8

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Receives generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_IORead_SInt8
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const int inOffset,
 int& outData
)
```

Parameters

	Name	Type	Range	Default	Description
↻	ioState	Hilscher_State&			Object used to maintain state of the function.
→	inBoardName	const Optional<String>&		NIL	
→	inChannelNumber	const int	0 - 10		
→	inOffset	const int	0 - 10000		
←	outData	int&			

Remarks

Driver

- Windows driver (recommended **1.5.0.0**) https://kb.hilscher.com/display/CIFXDRV/Versions+-+NXDRV-WIN
- SYCON.net for configuring slots, generating configuration files: https://kb.hilscher.com/display/SYCON/Version+History

For profinet: Firmware (recommended **3.13** series) https://kb.hilscher.com/display/PNS3V5/Version+History+-+V3.5+to+V3.x

Firmware needs to be uploaded to the card prior to opening any type of channel.
In case of failing to update the firmware, try to restart your computer.

IO Offset (inOffset)

Offset as displayed in Hilscher software

Data direction

Slot direction is from the application perspective, different than in SYCON software.
IN: in SYCON.net is marked as output
OUT: in SYCON.net is marked as input

Data endianness

Remember to check endianness on controller and device sides, most controllers will use big endian. Data written to the card will be converted by default to network order. (to big endian)
Data read from the card will be converted to native byte order. (by default from big endian)

Hilscher_Channel_IOWrite_ByteBuffer

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sends generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_IOWrite_ByteBuffer
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const int inOffset,
 const avl::ByteBuffer& inData
)
```

Parameters

	Name	Type	Range	Default	Description
↻	ioState	Hilscher_State&			Object used to maintain state of the function.
→	inBoardName	const Optional<String>&		NIL	
→	inChannelNumber	const int	0 - 10		
→	inOffset	const int	0 - 10000		
→	inData	const ByteBuffer&			

Hilscher_Channel_IOWrite_SInt16

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sends generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_IOWrite_SInt16
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const int inOffset,
 const avl::DataEndianness::Type inDataEndianness,
 const int inData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inOffset	const int	0 - 10000		
	inDataEndianness	const DataEndianness::Type		BigEndian	
	inData	const int	-32768 - 32767		

Remarks

Driver

- Windows driver (recommended **1.5.0.0**) https://kb.hilscher.com/display/CIFXDRV/Versions+-+NXDRV-WIN
- SYCON.net for configuring slots, generating configuration files: https://kb.hilscher.com/display/SYCON/Version+History

For profinet: Firmware (recommended **3.13** series) https://kb.hilscher.com/display/PNS3V5/Version+History+-+V3.5+to+V3.x

Firmware needs to be uploaded to the card prior to opening any type of channel.
In case of failing to update the firmware, try to restart your computer.

IO Offset (inOffset)

Offset as displayed in Hilscher software

Data direction

Slot direction is from the application perspective, different than in SYCON software.
IN: in SYCON.net is marked as output
OUT: in SYCON.net is marked as input

Data endianness

Remember to check endianness on controller and device sides, most controllers will use big endian. Data written to the card will be converted by default to network order. (to big endian)
Data read from the card will be converted to native byte order. (by default from big endian)

Hilscher_Channel_IOWrite_SInt32

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sends generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_IOWrite_SInt32
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const int inOffset,
 const avl::DataEndianness::Type inDataEndianness,
 const int inData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inOffset	const int	0 - 10000		
	inDataEndianness	const DataEndianness::Type		BigEndian	
	inData	const int			

Remarks

Driver

- Windows driver (recommended **1.5.0.0**) https://kb.hilscher.com/display/CIFXDRV/Versions+-+NXDRV-WIN
- SYCON.net for configuring slots, generating configuration files: https://kb.hilscher.com/display/SYCON/Version+History

For profinet: Firmware (recommended **3.13** series) https://kb.hilscher.com/display/PNS3V5/Version+History+-+V3.5+to+V3.x

Firmware needs to be uploaded to the card prior to opening any type of channel.
In case of failing to update the firmware, try to restart your computer.

IO Offset (inOffset)

Offset as displayed in Hilscher software

Data direction

Slot direction is from the application perspective, different than in SYCON software.
IN: in SYCON.net is marked as output
OUT: in SYCON.net is marked as input

Data endianness

Remember to check endianness on controller and device sides, most controllers will use big endian. Data written to the card will be converted by default to network order. (to big endian)
Data read from the card will be converted to native byte order. (by default from big endian)

Hilscher_Channel_IOWrite_SInt64

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sends generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_IOWrite_SInt64
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const int inOffset,
 const avl::DataEndianness::Type inDataEndianness,
 const atl::int64 inData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inOffset	const int	0 - 10000		
	inDataEndianness	const DataEndianness::Type		BigEndian	
	inData	const int64			

Remarks

Driver

- Windows driver (recommended **1.5.0.0**) https://kb.hilscher.com/display/CIFXDRV/Versions+-+NXDRV-WIN
- SYCON.net for configuring slots, generating configuration files: https://kb.hilscher.com/display/SYCON/Version+History

For profinet: Firmware (recommended **3.13** series) https://kb.hilscher.com/display/PNS3V5/Version+History+-+V3.5+to+V3.x

Firmware needs to be uploaded to the card prior to opening any type of channel.
In case of failing to update the firmware, try to restart your computer.

IO Offset (inOffset)

Offset as displayed in Hilscher software

Data direction

Slot direction is from the application perspective, different than in SYCON software.
IN: in SYCON.net is marked as output
OUT: in SYCON.net is marked as input

Data endianness

Remember to check endianness on controller and device sides, most controllers will use big endian. Data written to the card will be converted by default to network order. (to big endian)
Data read from the card will be converted to native byte order. (by default from big endian)

Hilscher_Channel_IOWrite_SInt8

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sends generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_IOWrite_SInt8
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const int inOffset,
 const int inData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inOffset	const int	0 - 10000		
	inData	const int	-128 - 127		

Remarks

Driver

- Windows driver (recommended **1.5.0.0**) https://kb.hilscher.com/display/CIFXDRV/Versions+-+NXDRV-WIN
- SYCON.net for configuring slots, generating configuration files: https://kb.hilscher.com/display/SYCON/Version+History

For profinet: Firmware (recommended **3.13** series) https://kb.hilscher.com/display/PNS3V5/Version+History+-+V3.5+to+V3.x

Firmware needs to be uploaded to the card prior to opening any type of channel.
In case of failing to update the firmware, try to restart your computer.

IO Offset (inOffset)

Offset as displayed in Hilscher software

Data direction

Slot direction is from the application perspective, different than in SYCON software.
IN: in SYCON.net is marked as output
OUT: in SYCON.net is marked as input

Data endianness

Remember to check endianness on controller and device sides, most controllers will use big endian. Data written to the card will be converted by default to network order. (to big endian)
Data read from the card will be converted to native byte order. (by default from big endian)

Also in AVL Lite

Hilscher_Channel_NameRead_ByteBuffer

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Receives generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_NameRead_ByteBuffer
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const atl::String& inName,
 const int inSize,
 avl::ByteBuffer& outData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inName	const String&			
	inSize	const int	1 - ∞		
	outData	ByteBuffer&			

Also in AVL Lite

Hilscher_Channel_NameRead_SInt16

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Receives generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_NameRead_SInt16
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const atl::String& inName,
 const avl::DataEndianness::Type inDataEndianness,
 int& outData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inName	const String&			
	inDataEndianness	const DataEndianness::Type		BigEndian	
	outData	int&			

Also in AVL Lite

Hilscher_Channel_NameRead_SInt32

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Receives generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_NameRead_SInt32
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const atl::String& inName,
 const avl::DataEndianness::Type inDataEndianness,
 int& outData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inName	const String&			
	inDataEndianness	const DataEndianness::Type		BigEndian	
	outData	int&			

Also in AVL Lite

Hilscher_Channel_NameRead_SInt64

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Receives generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_NameRead_SInt64
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const atl::String& inName,
 const avl::DataEndianness::Type inDataEndianness,
 atl::sint64& outData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inName	const String&			
	inDataEndianness	const DataEndianness::Type		BigEndian	
	outData	sint64&			

Also in AVL Lite

Hilscher_Channel_NameRead_SInt8

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Receives generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_NameRead_SInt8
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const atl::String& inName,
 int& outData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inName	const String&			
	outData	int&			

Also in AVL Lite

Hilscher_Channel_NameWrite_ByteBuffer

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sends generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_NameWrite_ByteBuffer
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const atl::String& inName,
 const avl::ByteBuffer& inData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inName	const String&			
	inData	const ByteBuffer&			

Hilscher_Channel_NameWrite_SInt16

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sends generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_NameWrite_SInt16
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const atl::String& inName,
 const avl::DataEndianness::Type inDataEndianness,
 const int inData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inName	const String&			
	inDataEndianness	const DataEndianness::Type		BigEndian	
	inData	const int	-32768 - 32767		

Also in AVL Lite

Hilscher_Channel_NameWrite_SInt32

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sends generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_NameWrite_SInt32
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const atl::String& inName,
 const avl::DataEndianness::Type inDataEndianness,
 const int inData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inName	const String&			
	inDataEndianness	const DataEndianness::Type		BigEndian	
	inData	const int			

Also in AVL Lite

Hilscher_Channel_NameWrite_SInt64

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sends generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_NameWrite_SInt64
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const atl::String& inName,
 const avl::DataEndianness::Type inDataEndianness,
 const atl::sint64 inData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inName	const String&			
	inDataEndianness	const DataEndianness::Type		BigEndian	
	inData	const sint64			

Also in AVL Lite

Hilscher_Channel_NameWrite_SInt8

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sends generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_NameWrite_SInt8
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const atl::String& inName,
 const int inData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inName	const String&			
	inData	const int	-128 - 127		

Hilscher_Channel_Open_EtherCAT

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Opens a Hilscher device channel connection.

Syntax

```
void avl::Hilscher_Channel_Open_EtherCAT
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const atl::Optional<atl::File> inConfig,
 const atl::Optional<atl::File> inNwid,
 const atl::Optional<atl::Array<avl::HilscherSignal>>& inInputSignalConfiguration,
 const atl::Optional<atl::Array<avl::HilscherSignal>>& inOutputSignalConfiguration
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inConfig	const Optional<File>		NIL	Configuration file generated in SYCON.net (xxx.nxd)
	inNwid	const Optional<File>		NIL	Configuration file generated in SYCON.net (xxx_nwid.nxd)
	inInputSignalConfiguration	const Optional<Array<HilscherSignal>>&		NIL	
	inOutputSignalConfiguration	const Optional<Array<HilscherSignal>>&		NIL	

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Hilscher_Channel_Open_EthernetIP

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Opens a Hilscher device channel connection.

Syntax

```
void avl::Hilscher_Channel_Open_EthernetIP
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const atl::Optional<atl::File> inConfig,
 const atl::Optional<atl::File> inNwid,
 const atl::Optional<atl::Array<avl::HilscherSignal>>& inInputSignalConfiguration,
 const atl::Optional<atl::Array<avl::HilscherSignal>>& inOutputSignalConfiguration
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inConfig	const Optional<File>		NIL	Configuration file generated in SYCON.net (xxx.nxd)
	inNwid	const Optional<File>		NIL	Configuration file generated in SYCON.net (xxx_nwid.nxd)
	inInputSignalConfiguration	const Optional<Array<HilscherSignal>>&		NIL	
	inOutputSignalConfiguration	const Optional<Array<HilscherSignal>>&		NIL	

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Hilscher_Channel_Open_Profinet

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Opens a Hilscher device channel connection.

Syntax

```
void avl::Hilscher_Channel_Open_Profinet
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const atl::Optional<atl::File> inConfig,
 const atl::Optional<atl::File> inNwid,
 const atl::Optional<atl::Array<avl::HilscherSignal>>& inInputSignalConfiguration,
 const atl::Optional<atl::Array<avl::HilscherSignal>>& inOutputSignalConfiguration
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inConfig	const Optional<File>		NIL	Configuration file generated in SYCON.net (xxx.nxd)
	inNwid	const Optional<File>		NIL	Configuration file generated in SYCON.net (xxx_nwid.nxd)
	inInputSignalConfiguration	const Optional<Array<HilscherSignal>>&		NIL	
	inOutputSignalConfiguration	const Optional<Array<HilscherSignal>>&		NIL	

Remarks

Initial state

Opening an not configured channel sets all output slots to 0, configured channel data is left as it was.

Channel configuration

The channel is only configured with provided configuration files, if it is in a not configured state - no connection between card and master device (for example a PLC). If it is configured, the filter assumes proper operation, and leaves card configuration in existing condition.

Driver

- Windows driver (recommended **1.5.0.0**) https://kb.hilscher.com/display/CIFXDRV/Versions+-+NXDRV-WIN
- SYCON.net for configuring slots, generating configuration files: https://kb.hilscher.com/display/SYCON/Version+History

For profinet: Firmware (recommended **3.13** series) https://kb.hilscher.com/display/PNS3V5/Version+History+-+V3.5+to+V3.x

Firmware needs to be uploaded to the card prior to opening any type of channel.
In case of failing to update the firmware, try to restart your computer.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Hilscher_Channel_SetSignalConfiguration

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets the slot configuration.

Syntax

```
void avl::Hilscher_Channel_SetSignalConfiguration
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const atl::Array<avl::HilscherSignal>& inInputSignalConfiguration,
 const atl::Array<avl::HilscherSignal>& inOutputSignalConfiguration
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inInputSignalConfiguration	const Array<HilscherSignal>&			
	inOutputSignalConfiguration	const Array<HilscherSignal>&			

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Hilscher_Channel_SlotRead_ByteBuffer

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Receives generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_SlotRead_ByteBuffer
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const int inSlot,
 const int inSize,
 avl::ByteBuffer& outData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inSlot	const int	0 - 1000		
	inSize	const int	1 - ∞		
	outData	ByteBuffer&			

Hilscher_Channel_SlotRead_SInt16

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Receives generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_SlotRead_SInt16
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const int inSlot,
 const avl::DataEndianness::Type inDataEndianness,
 int& outData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inSlot	const int	0 - 1000		
	inDataEndianness	const DataEndianness::Type		BigEndian	
	outData	int&			

Remarks

Driver

- Windows driver (recommended **1.5.0.0**) https://kb.hilscher.com/display/CIFXDRV/Versions+-+NXDRV-WIN
- SYCON.net for configuring slots, generating configuration files: https://kb.hilscher.com/display/SYCON/Version+History

For profinet: Firmware (recommended **3.13** series) https://kb.hilscher.com/display/PNS3V5/Version+History+-+V3.5+to+V3.x

Firmware needs to be uploaded to the card prior to opening any type of channel.
In case of failing to update the firmware, try to restart your computer.

inSlot

Slot as configured in Profinet/Hilscher software

Data direction

Slot direction is from the application perspective, different than in SYCON software.
IN: in SYCON.net is marked as output
OUT: in SYCON.net is marked as input

Data endianness

Remember to check endianness on controller and device sides, most controllers will use big endian. Data written to the card will be converted by default to network order. (to big endian)
Data read from the card will be converted to native byte order. (by default from big endian)

Hilscher_Channel_SlotRead_SInt32

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Receives generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_SlotRead_SInt32
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const int inSlot,
 const avl::DataEndianness::Type inDataEndianness,
 int& outData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inSlot	const int	0 - 1000		
	inDataEndianness	const DataEndianness::Type		BigEndian	
	outData	int&			

Remarks

Driver

- Windows driver (recommended **1.5.0.0**) https://kb.hilscher.com/display/CIFXDRV/Versions+-+NXDRV-WIN
- SYCON.net for configuring slots, generating configuration files: https://kb.hilscher.com/display/SYCON/Version+History

For profinet: Firmware (recommended **3.13** series) https://kb.hilscher.com/display/PNS3V5/Version+History+-+V3.5+to+V3.x

Firmware needs to be uploaded to the card prior to opening any type of channel.
In case of failing to update the firmware, try to restart your computer.

inSlot

Slot as configured in Profinet/Hilscher software

Data direction

Slot direction is from the application perspective, different than in SYCON software.
IN: in SYCON.net is marked as output
OUT: in SYCON.net is marked as input

Data endianness

Remember to check endianness on controller and device sides, most controllers will use big endian. Data written to the card will be converted by default to network order. (to big endian)
Data read from the card will be converted to native byte order. (by default from big endian)

Hilscher_Channel_SlotRead_SInt64

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Receives generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_SlotRead_SInt64
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const int inSlot,
 const avl::DataEndianness::Type inDataEndianness,
 atl::sint64& outData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inSlot	const int	0 - 1000		
	inDataEndianness	const DataEndianness::Type		BigEndian	
	outData	sint64&			

Remarks

Driver

- Windows driver (recommended **1.5.0.0**) https://kb.hilscher.com/display/CIFXDRV/Versions+-+NXDRV-WIN
- SYCON.net for configuring slots, generating configuration files: https://kb.hilscher.com/display/SYCON/Version+History

For profinet: Firmware (recommended **3.13** series) https://kb.hilscher.com/display/PNS3V5/Version+History+-+V3.5+to+V3.x

Firmware needs to be uploaded to the card prior to opening any type of channel.
In case of failing to update the firmware, try to restart your computer.

inSlot

Slot as configured in Profinet/Hilscher software

Data direction

Slot direction is from the application perspective, different than in SYCON software.
IN: in SYCON.net is marked as output
OUT: in SYCON.net is marked as input

Data endianness

Remember to check endianness on controller and device sides, most controllers will use big endian. Data written to the card will be converted by default to network order. (to big endian)
Data read from the card will be converted to native byte order. (by default from big endian)

Hilscher_Channel_SlotRead_SInt8

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Receives generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_SlotRead_SInt8
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const int inSlot,
 int& outData
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	Hilscher_State&			Object used to maintain state of the function.
→	inBoardName	const Optional<String>&		NIL	
→	inChannelNumber	const int	0 - 10		
→	inSlot	const int	0 - 1000		
←	outData	int&			

Remarks

Driver

- Windows driver (recommended **1.5.0.0**) https://kb.hilscher.com/display/CIFXDRV/Versions+-+NXDRV-WIN
- SYCON.net for configuring slots, generating configuration files: https://kb.hilscher.com/display/SYCON/Version+History

For profinet: Firmware (recommended **3.13** series) https://kb.hilscher.com/display/PNS3V5/Version+History+-+V3.5+to+V3.x

Firmware needs to be uploaded to the card prior to opening any type of channel.
In case of failing to update the firmware, try to restart your computer.

inSlot

Slot as configured in Profinet/Hilscher software

Data direction

Slot direction is from the application perspective, different than in SYCON software.
IN: in SYCON.net is marked as output
OUT: in SYCON.net is marked as input

Data endianness

Remember to check endianness on controller and device sides, most controllers will use big endian. Data written to the card will be converted by default to network order. (to big endian)
Data read from the card will be converted to native byte order. (by default from big endian)

Hilscher_Channel_SlotWrite_ByteBuffer

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sends generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_SlotWrite_ByteBuffer
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const int inSlot,
 const avl::ByteBuffer& inData
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	Hilscher_State&			Object used to maintain state of the function.
→	inBoardName	const Optional<String>&		NIL	
→	inChannelNumber	const int	0 - 10		
→	inSlot	const int	0 - 1000		
→	inData	const ByteBuffer&			

Hilscher_Channel_SlotWrite_SInt16

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sends generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_SlotWrite_SInt16
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const int inSlot,
 const avl::DataEndianness::Type inDataEndianness,
 const int inData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inSlot	const int	0 - 1000		
	inDataEndianness	const DataEndianness::Type		BigEndian	
	inData	const int	-32768 - 32767		

Remarks

Driver

- Windows driver (recommended **1.5.0.0**) https://kb.hilscher.com/display/CIFXDRV/Versions+-+NXDRV-WIN
- SYCON.net for configuring slots, generating configuration files: https://kb.hilscher.com/display/SYCON/Version+History

For profinet: Firmware (recommended **3.13** series) https://kb.hilscher.com/display/PNS3V5/Version+History+-+V3.5+to+V3.x

Firmware needs to be uploaded to the card prior to opening any type of channel.
In case of failing to update the firmware, try to restart your computer.

inSlot

Slot as configured in Profinet/Hilscher software

Data direction

Slot direction is from the application perspective, different than in SYCON software.
IN: in SYCON.net is marked as output
OUT: in SYCON.net is marked as input

Data endianness

Remember to check endianness on controller and device sides, most controllers will use big endian. Data written to the card will be converted by default to network order. (to big endian)
Data read from the card will be converted to native byte order. (by default from big endian)

Hilscher_Channel_SlotWrite_SInt32

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sends generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_SlotWrite_SInt32
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const int inSlot,
 const avl::DataEndianness::Type inDataEndianness,
 const int inData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inSlot	const int	0 - 1000		
	inDataEndianness	const DataEndianness::Type		BigEndian	
	inData	const int			

Remarks

Driver

- Windows driver (recommended **1.5.0.0**) https://kb.hilscher.com/display/CIFXDRV/Versions+-+NXDRV-WIN
- SYCON.net for configuring slots, generating configuration files: https://kb.hilscher.com/display/SYCON/Version+History

For profinet: Firmware (recommended **3.13** series) https://kb.hilscher.com/display/PNS3V5/Version+History+-+V3.5+to+V3.x

Firmware needs to be uploaded to the card prior to opening any type of channel.
In case of failing to update the firmware, try to restart your computer.

inSlot

Slot as configured in Profinet/Hilscher software

Data direction

Slot direction is from the application perspective, different than in SYCON software.
IN: in SYCON.net is marked as output
OUT: in SYCON.net is marked as input

Data endianness

Remember to check endianness on controller and device sides, most controllers will use big endian. Data written to the card will be converted by default to network order. (to big endian)
Data read from the card will be converted to native byte order. (by default from big endian)

Hilscher_Channel_SlotWrite_SInt64

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sends generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_SlotWrite_SInt64
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const int inSlot,
 const avl::DataEndianness::Type inDataEndianness,
 const atl::sint64 inData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inSlot	const int	0 - 1000		
	inDataEndianness	const DataEndianness::Type		BigEndian	
	inData	const sint64			

Remarks

Driver

- Windows driver (recommended **1.5.0.0**) https://kb.hilscher.com/display/CIFXDRV/Versions+-+NXDRV-WIN
- SYCON.net for configuring slots, generating configuration files: https://kb.hilscher.com/display/SYCON/Version+History

For profinet: Firmware (recommended **3.13** series) https://kb.hilscher.com/display/PNS3V5/Version+History+-+V3.5+to+V3.x

Firmware needs to be uploaded to the card prior to opening any type of channel.
In case of failing to update the firmware, try to restart your computer.

inSlot

Slot as configured in Profinet/Hilscher software

Data direction

Slot direction is from the application perspective, different than in SYCON software.
IN: in SYCON.net is marked as output
OUT: in SYCON.net is marked as input

Data endianness

Remember to check endianness on controller and device sides, most controllers will use big endian. Data written to the card will be converted by default to network order. (to big endian)
Data read from the card will be converted to native byte order. (by default from big endian)

Hilscher_Channel_SlotWrite_SInt8

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sends generic packet that can be filled with any data through Hilscher device.

Syntax

```
void avl::Hilscher_Channel_SlotWrite_SInt8
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 const int inChannelNumber,
 const int inSlot,
 const int inData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Hilscher_State&			Object used to maintain state of the function.
	inBoardName	const Optional<String>&		NIL	
	inChannelNumber	const int	0 - 10		
	inSlot	const int	0 - 1000		
	inData	const int	-128 - 127		

Remarks

Driver

- Windows driver (recommended **1.5.0.0**) https://kb.hilscher.com/display/CIFXDRV/Versions+-+NXDRV-WIN
- SYCON.net for configuring slots, generating configuration files: https://kb.hilscher.com/display/SYCON/Version+History

For profinet: Firmware (recommended **3.13** series) https://kb.hilscher.com/display/PNS3V5/Version+History+-+V3.5+to+V3.x

Firmware needs to be uploaded to the card prior to opening any type of channel.
In case of failing to update the firmware, try to restart your computer.

inSlot

Slot as configured in Profinet/Hilscher software

Data direction

Slot direction is from the application perspective, different than in SYCON software.
IN: in SYCON.net is marked as output
OUT: in SYCON.net is marked as input

Data endianness

Remember to check endianness on controller and device sides, most controllers will use big endian. Data written to the card will be converted by default to network order. (to big endian)
Data read from the card will be converted to native byte order. (by default from big endian)

Also in AVL Lite

Hilscher_Driver_Close

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Closes a Hilscher device driver connection.

Syntax

```
void avl::Hilscher_Driver_Close
(
 avl::Hilscher_State& ioState
)
```

Parameters

	Name	Type	Default	Description
	ioState	Hilscher_State&		Object used to maintain state of the function.

Remarks

Driver

- Windows driver (recommended **1.5.0.0**) https://kb.hilscher.com/display/CIFXDRV/Versions+-+NXDRV-WIN
- SYCON.net for configuring slots, generating configuration files: https://kb.hilscher.com/display/SYCON/Version+History

For profinet: Firmware (recommended **3.13** series) https://kb.hilscher.com/display/PNS3V5/Version+History+-+V3.5+to+V3.x

Firmware needs to be uploaded to the card prior to opening any type of channel.
In case of failing to update the firmware, try to restart your computer.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Also in AVL Lite

Hilscher_Driver_GetBoardInformation

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets Hilscher device board information.

Syntax

```
void avl::Hilscher_Driver_GetBoardInformation
(
 avl::Hilscher_State& ioState,
 atl::Array<avl::HilscherDriverBoardInformation>& outBoardInformation
)
```

Parameters

	Name	Type	Default	Description
	ioState	Hilscher_State&		Object used to maintain state of the function.
	outBoardInformation	Array<HilscherDriverBoardInformation>&		

Remarks

Driver

- Windows driver (recommended **1.5.0.0**) https://kb.hilscher.com/display/CIFXDRV/Versions+-+NXDRV-WIN
- SYCON.net for configuring slots, generating configuration files: https://kb.hilscher.com/display/SYCON/Version+History

For profinet: Firmware (recommended **3.13** series) https://kb.hilscher.com/display/PNS3V5/Version+History+-+V3.5+to+V3.x

Firmware needs to be uploaded to the card prior to opening any type of channel.
In case of failing to update the firmware, try to restart your computer.

Also in AVL Lite

Hilscher_Driver_GetChannelInformation

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets Hilscher device channel information.

Syntax

```
void avl::Hilscher_Driver_GetChannelInformation
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName,
 atl::Array<avl::HilscherDriverChannelInformation>& outChannelInformation
)
```

Parameters

	Name	Type	Default	Description
	ioState	Hilscher_State&		Object used to maintain state of the function.
	inBoardName	const Optional<String>&	NIL	
	outChannelInformation	Array<HilscherDriverChannelInformation>&		

Remarks

Driver

- Windows driver (recommended **1.5.0.0**) https://kb.hilscher.com/display/CIFXDRV/Versions+-+NXDRV-WIN
- SYCON.net for configuring slots, generating configuration files: https://kb.hilscher.com/display/SYCON/Version+History

For profinet: Firmware (recommended **3.13** series) https://kb.hilscher.com/display/PNS3V5/Version+History+-+V3.5+to+V3.x

Firmware needs to be uploaded to the card prior to opening any type of channel.
In case of failing to update the firmware, try to restart your computer.

Data direction

Slot direction is from the application perspective, different than in SYCON software.
IN: in SYCON.net is marked as output
OUT: in SYCON.net is marked as input

Also in AVL Lite

Hilscher_Driver_GetInformation

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets Hilscher device driver information.

Syntax

```
void avl::Hilscher_Driver_GetInformation
(
 avl::Hilscher_State& ioState,
 avl::HilscherDriverInformation& outDriverInformation
)
```

Parameters

	Name	Type	Default	Description
	ioState	Hilscher_State&		Object used to maintain state of the function.
	outDriverInformation	HilscherDriverInformation&		

Remarks

Driver

- Windows driver (recommended **1.5.0.0**) https://kb.hilscher.com/display/CIFXDRV/Versions+-+NXDRV-WIN
- SYCON.net for configuring slots, generating configuration files: https://kb.hilscher.com/display/SYCON/Version+History

For profinet: Firmware (recommended **3.13** series) https://kb.hilscher.com/display/PNS3V5/Version+History+-+V3.5+to+V3.x

Firmware needs to be uploaded to the card prior to opening any type of channel.
In case of failing to update the firmware, try to restart your computer.

Also in AVL Lite

Hilscher_Driver_Open

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Opens a Hilscher device driver connection.

Syntax

```
void avl::Hilscher_Driver_Open
(
 avl::Hilscher_State& ioState
)
```

Parameters

	Name	Type	Default	Description
	ioState	Hilscher_State&		Object used to maintain state of the function.

Remarks

Driver

- Windows driver (recommended **1.5.0.0**) https://kb.hilscher.com/display/CIFXDRV/Versions+-+NXDRV-WIN
- SYCON.net for configuring slots, generating configuration files: https://kb.hilscher.com/display/SYCON/Version+History

For profinet: Firmware (recommended **3.13** series) https://kb.hilscher.com/display/PNS3V5/Version+History+-+V3.5+to+V3.x

Firmware needs to be uploaded to the card prior to opening any type of channel.
In case of failing to update the firmware, try to restart your computer.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Also in AVL Lite

Hilscher_Driver_Restart

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Restarts Hilscher device.

Syntax

```
void avl::Hilscher_Driver_Restart
(
 avl::Hilscher_State& ioState,
 const atl::Optional<atl::String>& inBoardName
)
```

Parameters

	Name	Type	Default	Description
	ioState	Hilscher_State&		Object used to maintain state of the function.
	inBoardName	const Optional<String>&	NIL	

Remarks

Driver

- Windows driver (recommended **1.5.0.0**) https://kb.hilscher.com/display/CIFXDRV/Versions+-+NXDRV-WIN
- SYCON.net for configuring slots, generating configuration files: https://kb.hilscher.com/display/SYCON/Version+History

For profinet: Firmware (recommended **3.13** series) https://kb.hilscher.com/display/PNS3V5/Version+History+-+V3.5+to+V3.x

Firmware needs to be uploaded to the card prior to opening any type of channel.
In case of failing to update the firmware, try to restart your computer.

162. The Imaging Source

Table of content:

ICImagingControl_EnableExternalTrigger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Controls camera external trigger.

Syntax

```
void avl::ICImagingControl_EnableExternalTrigger
(
 ICImagingControlState& ioState,
 const atl::Optional<const atl::String&>& inDeviceID,
 bool inEnable
)
```

Parameters

	Name	Type	Default	Description
	ioState	ICImagingControlState&		Object used to maintain state of the function.
	inDeviceID	const Optional<const String&>&	NIL	Serial number, display name, unique name or base name of the camera
	inEnable	bool		External trigger state

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install ICImagingControl SDK software. Currently Aurora Vision Studio requires **ICImagingControl SDK version 3.5.7**.

Proper camera drivers and ICImagingControl SDK (IC Imaging Control .NET Component, C++ Class Library) can be downloaded from the following website: https://www.theimagingsource.com/en-us/support/download/. To work with the Imaging Source SDK you need to install SDK on the target machine or copy manually the .dll files from the `Documents\IC Imaging Control 3.5\redist\c++\win32\` directory on your computer (or `Documents\IC Imaging Control 3.5\redist\c++\x64\` if you use Aurora Vision Studio x64) and add them to the Aurora Vision Studio directory in Program Files.

Camera identification

When there is only one camera connected to the computer, field **inDeviceID** can be set to *Auto*. In such case, the first available camera will be found and connected.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** can be set to:

- serial number,
- display name,
- unique name,
- base name of the camera.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- ICImagingControl_GetDigitalIOState – Gets digital IO lines state.
- ICImagingControl_SetDigitalOutputState – Sets digital IO line state.
- ICImagingControl_GrabImage – Captures a frame from The Imaging Source cameras using ICImagingControl.

ICImagingControl_GetDigitalIOState

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets digital IO lines state.

Syntax

```
void avl::ICImagingControl_GetDigitalIOState
(
 ICImagingControlState& ioState,
 const atl::Optional<const atl::String&>& inDeviceID,
 bool& outInputState,
 bool& outOutputState
)
```

Parameters

	Name	Type	Default	Description
	ioState	ICImagingControlState&		Object used to maintain state of the function.
	inDeviceID	const Optional<const String&>&	NIL	Serial number, display name, unique name or base name of the camera
	outInputState	bool&		Digital input state
	outOutputState	bool&		Digital output state

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install ICImagingControl SDK software. Currently Aurora Vision Studio requires **ICImagingControl SDK version 3.5.7**.

Proper camera drivers and ICImagingControl SDK (IC Imaging Control .NET Component, C++ Class Library) can be downloaded from the following website: https://www.theimagingsource.com/en-us/support/download/. To work with the Imaging Source SDK you need to install SDK on the target machine or copy manually the .dll files from the `Documents\IC Imaging Control 3.5\redist\c++\win32\` directory on your computer (or `Documents\IC Imaging Control 3.5\redist\c++\x64\` if you use Aurora Vision Studio x64) and add them to the Aurora Vision Studio directory in Program Files.

Camera identification

When there is only one camera connected to the computer, field **inDeviceID** can be set to *Auto*. In such case, the first available camera will be found and connected.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** can be set to:

- serial number,
- display name,
- unique name,
- base name of the camera.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- ICImagingControl_EnableExternalTrigger – Controls camera external trigger.
- ICImagingControl_SetDigitalOutputState – Sets digital IO line state.
- ICImagingControl_GrabImage – Captures a frame from The Imaging Source cameras using ICImagingControl.

ICImagingControl_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a frame from The Imaging Source cameras using ICImagingControl.

Syntax

```
bool avl::ICImagingControl_GrabImage
(
 ICImagingControlState& ioState,
 const atl::Optional<const atl::String&>& inDeviceID,
 int inInputQueueSize,
 const avl::ICImagingControlColorFormat::Type inPixelFormat,
 const atl::Optional<int> inBinning,
 const atl::Optional<int> inSkipping,
 const atl::Optional<avl::Box> inRoi,
 const atl::Optional<float> inFrameRate,
 const atl::Optional<int> inGain,
 const atl::Optional<bool> inGainAuto,
 const atl::Optional<double> inExposure,
 const atl::Optional<bool> inExposureAuto,
 const atl::Optional<int> inGamma,
 const atl::Optional<int> inSharpness,
 const atl::Optional<int> inFocus,
 avl::Image& outImage,
 atl::int64& outFrameID
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	ICImagingControlState&			Object used to maintain state of the function.
	inDeviceID	const Optional<const String&>&		NIL	Serial number, display name, unique name or base name of the camera
	inInputQueueSize	int	1 - +∞	3	
	inPixelFormat	const ICImagingControlColorFormat::Type		RGB24	Pixel format
	inBinning	const Optional<int>	2 - 64	NIL	Skipping factor
	inSkipping	const Optional<int>	2 - 64	NIL	Binning factor
	inRoi	const Optional<Box>		NIL	Hardware Region of Interest
	inFrameRate	const Optional<float>	0.5 - ∞	NIL	Camera frame rate
	inGain	const Optional<int>	0 - +∞	NIL	Gain value
	inGainAuto	const Optional<bool>		NIL	Enable auto gain
	inExposure	const Optional<double>	0 - +∞	NIL	Exposure time
	inExposureAuto	const Optional<bool>		NIL	Enable auto exposure
	inGamma	const Optional<int>	0 - +∞	NIL	Gamma value
	inSharpness	const Optional<int>	0 - +∞	NIL	Sharpness value
	inFocus	const Optional<int>	0 - +∞	NIL	Focus value
	outImage	Image&			Captured frame
	outFrameID	int64&			Captured frame ID

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install ICImagingControl SDK software. Currently Aurora Vision Studio requires **ICImagingControl SDK version 3.5.7**.

Proper camera drivers and ICImagingControl SDK (IC Imaging Control .NET Component, C++ Class Library) can be downloaded from the following website: https://www.theimagingsource.com/en-us/support/download/. To work with the Imaging Source SDK you need to install SDK on the target machine or copy manually the .dll files from the `Documents\IC Imaging Control 3.5\redist\c++\win32\` directory on your computer (or `Documents\IC Imaging Control 3.5\redist\c++\x64\` if you use Aurora Vision Studio x64) and add them to the Aurora Vision Studio directory in Program Files.

Camera identification

When there is only one camera connected to the computer, field **inDeviceID** can be set to *Auto*. In such case, the first available camera will be found and connected.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** can be set to:

- serial number,
- display name,
- unique name,
- base name of the camera.

Camera parameters

Most of the parameters exposed by camera filters are optional, setting them to *Auto* leaves related parameter for automatic configuration by the camera driver.

To change other and more advanced camera parameters, use the VCD Property Inspector available with ICImagingControl SDK. Refer to SDK documentation for information about parameters and how to save them into memory channels.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- ICImagingControl_GrabImage_WithTimeout – Captures a frame from The Imaging Source cameras using ICImagingControl.
- ICImagingControl_StartAcquisition – Initializes and starts image acquisition in a camera.
- ICImagingControl_GetDigitalIOState – Gets digital IO lines state.
- ICImagingControl_SetDigitalOutputState – Sets digital IO line state.
- ICImagingControl_EnableExternalTrigger – Controls camera external trigger.

ICImagingControl_GrabImage_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a frame from The Imaging Source cameras using ICImagingControl.

Syntax

```
bool avl::ICImagingControl_GrabImage_WithTimeout
(
 ICImagingControlState& ioState,
 const atl::Optional<const atl::String&>& inDeviceID,
 int inInputQueueSize,
 int inTimeout,
 const avl::ICImagingControlColorFormat::Type inPixelFormat,
 const atl::Optional<int> inBinning,
 const atl::Optional<int> inSkipping,
 const atl::Optional<avl::Box> inRoi,
 const atl::Optional<float> inFrameRate,
 const atl::Optional<int> inGain,
 const atl::Optional<bool> inGainAuto,
 const atl::Optional<double> inExposure,
 const atl::Optional<bool> inExposureAuto,
 const atl::Optional<int> inGamma,
 const atl::Optional<int> inSharpness,
 const atl::Optional<int> inFocus,
 atl::Conditional<avl::Image>& outImage,
 atl::Conditional<atl::int64>& outFrameID
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	ICImagingControlState&			Object used to maintain state of the function.
	inDeviceID	const Optional<const String&>&		NIL	Serial number, display name, unique name or base name of the camera
	inInputQueueSize	int	1 - +∞	3	
	inTimeout	int			Maximum time to wait for frame in milliseconds
	inPixelFormat	const ICImagingControlColorFormat::Type		RGB24	Pixel format
	inBinning	const Optional<int>	2 - 64	NIL	Skipping factor
	inSkipping	const Optional<int>	2 - 64	NIL	Binning factor
	inRoi	const Optional<Box>		NIL	Hardware Region of Interest
	inFrameRate	const Optional<float>	0.5 - ∞	NIL	Camera frame rate
	inGain	const Optional<int>	0 - +∞	NIL	Gain value
	inGainAuto	const Optional<bool>		NIL	Enable auto gain
	inExposure	const Optional<double>	0 - +∞	NIL	Exposure time
	inExposureAuto	const Optional<bool>		NIL	Enable auto exposure
	inGamma	const Optional<int>	0 - +∞	NIL	Gamma value
	inSharpness	const Optional<int>	0 - +∞	NIL	Sharpness value
	inFocus	const Optional<int>	0 - +∞	NIL	Focus value
	outImage	Conditional<Image>&			Captured frame
	outFrameID	Conditional<int64>&			Captured frame ID

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install ICImagingControl SDK software. Currently Aurora Vision Studio requires **ICImagingControl SDK version 3.5.7**.

Proper camera drivers and ICImagingControl SDK (IC Imaging Control .NET Component, C++ Class Library) can be downloaded from the following website: https://www.theimagingsource.com/en-us/support/download/. To work with the Imaging Source SDK you need to install SDK on the target machine or copy manually the .dll files from the `Documents\IC Imaging Control 3.5\redist\c++\win32\` directory on your computer (or `Documents\IC Imaging Control 3.5\redist\c++\x64\` if you use Aurora Vision Studio x64) and add them to the Aurora Vision Studio directory in Program Files.

Camera identification

When there is only one camera connected to the computer, field **inDeviceID** can be set to *Auto*. In such case, the first available camera will be found and connected.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** can be set to:

- serial number,
- display name,
- unique name,
- base name of the camera.

Camera parameters

Most of the parameters exposed by camera filters are optional, setting them to *Auto* leaves related parameter for automatic configuration by the camera driver.

To change other and more advanced camera parameters, use the VCD Property Inspector available with ICImagingControl SDK. Refer to SDK documentation for information about parameters and how to save them into memory channels.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- ICImagingControl_GrabImage – Captures a frame from The Imaging Source cameras using ICImagingControl.
- ICImagingControl_StartAcquisition – Initializes and starts image acquisition in a camera.
- ICImagingControl_GetDigitalIOState – Gets digital IO lines state.
- ICImagingControl_SetDigitalOutputState – Sets digital IO line state.
- ICImagingControl_EnableExternalTrigger – Controls camera external trigger.

ICImagingControl_SetDigitalOutputState

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets digital IO line state.

Syntax

```
void avl::ICImagingControl_SetDigitalOutputState
(
 ICImagingControlState& ioState,
 const atl::Optional<const atl::String&>& inDeviceID,
 bool inOutputState
)
```

Parameters

	Name	Type	Default	Description
	ioState	ICImagingControlState&		Object used to maintain state of the function.
	inDeviceID	const Optional<const String&>&	NIL	Serial number, display name, unique name or base name of the camera
	inOutputState	bool		Digital output state

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install ICImagingControl SDK software. Currently Aurora Vision Studio requires **ICImagingControl SDK version 3.5.7**.

Proper camera drivers and ICImagingControl SDK (IC Imaging Control .NET Component, C++ Class Library) can be downloaded from the following website: https://www.theimagingsource.com/en-us/support/download/. To work with the Imaging Source SDK you need to install SDK on the target machine or copy manually the .dll files from the `Documents\IC Imaging Control 3.5\redist\c++\win32\` directory on your computer (or `Documents\IC Imaging Control 3.5\redist\c++\x64\` if you use Aurora Vision Studio x64) and add them to the Aurora Vision Studio directory in Program Files.

Camera identification

When there is only one camera connected to the computer, field **inDeviceID** can be set to *Auto*. In such case, the first available camera will be found and connected.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** can be set to:

- serial number,
- display name,
- unique name,
- base name of the camera.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- ICImagingControl_GetDigitalIOState – Gets digital IO lines state.
- ICImagingControl_EnableExternalTrigger – Controls camera external trigger.
- ICImagingControl_GrabImage – Captures a frame from The Imaging Source cameras using ICImagingControl.

ICImagingControl_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Initializes and starts image acquisition in a camera.

Syntax

```
void avl::ICImagingControl_StartAcquisition
(
 ICImagingControlState& ioState,
 const atl::Optional<const atl::String&>& inDeviceID,
 int inInputQueueSize,
 const avl::ICImagingControlColorFormat::Type inPixelFormat,
 const atl::Optional<int> inBinning,
 const atl::Optional<int> inSkipping,
 const atl::Optional<avl::Box> inRoi,
 const atl::Optional<float> inFrameRate,
 const atl::Optional<int> inGain,
 const atl::Optional<bool> inGainAuto,
 const atl::Optional<double> inExposure,
 const atl::Optional<bool> inExposureAuto,
 const atl::Optional<int> inGamma,
 const atl::Optional<int> inSharpness,
 const atl::Optional<int> inFocus
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	ICImagingControlState&			Object used to maintain state of the function.
	inDeviceID	const Optional<const String&>&		NIL	Serial number, display name, unique name or base name of the camera
	inInputQueueSize	int	1 - +∞	3	
	inPixelFormat	const ICImagingControlColorFormat::Type		RGB24	Pixel format
	inBinning	const Optional<int>	2 - 64	NIL	Skipping factor
	inSkipping	const Optional<int>	2 - 64	NIL	Binning factor
	inRoi	const Optional<Box>		NIL	Hardware Region of Interest
	inFrameRate	const Optional<float>	0.5 - ∞	NIL	Camera frame rate
	inGain	const Optional<int>	0 - +∞	NIL	Gain value
	inGainAuto	const Optional<bool>		NIL	Enable auto gain
	inExposure	const Optional<double>	0 - +∞	NIL	Exposure time
	inExposureAuto	const Optional<bool>		NIL	Enable auto exposure
	inGamma	const Optional<int>	0 - +∞	NIL	Gamma value
	inSharpness	const Optional<int>	0 - +∞	NIL	Sharpness value
	inFocus	const Optional<int>	0 - +∞	NIL	Focus value

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install ICImagingControl SDK software. Currently Aurora Vision Studio requires **ICImagingControl SDK version 3.5.7**.

Proper camera drivers and ICImagingControl SDK (IC Imaging Control .NET Component, C++ Class Library) can be downloaded from the following website: https://www.theimagingsource.com/en-us/support/download/. To work with the Imaging Source SDK you need to install SDK on the target machine or copy manually the .dll files from the `Documents\IC Imaging Control 3.5\redist\c++\win32\` directory on your computer (or `Documents\IC Imaging Control 3.5\redist\c++\x64\` if you use Aurora Vision Studio x64) and add them to the Aurora Vision Studio directory in Program Files.

Camera identification

When there is only one camera connected to the computer, field **inDeviceID** can be set to *Auto*. In such case, the first available camera will be found and connected.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** can be set to:

- serial number,
- display name,
- unique name,
- base name of the camera.

Camera parameters

Most of the parameters exposed by camera filters are optional, setting them to *Auto* leaves related parameter for automatic configuration by the camera driver.

To change other and more advanced camera parameters, use the VCD Property Inspector available with ICImagingControl SDK. Refer to SDK documentation for information about parameters and how to save them into memory channels.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- ICImagingControl_GrabImage – Captures a frame from The Imaging Source cameras using ICImagingControl.
- ICImagingControl_GrabImage_WithTimeout – Captures a frame from The Imaging Source cameras using ICImagingControl.
- ICImagingControl_GetDigitalIOState – Gets digital IO lines state.
- ICImagingControl_SetDigitalOutputState – Sets digital IO line state.
- ICImagingControl_EnableExternalTrigger – Controls camera external trigger.

ICImagingControl_StopAcquisition

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Stops image acquisition in a camera.

Syntax

```
void avl::ICImagingControl_StopAcquisition
(
 ICImagingControlState& ioState,
 const atl::Optional<const atl::String&>& inDeviceID
)
```

Parameters

	Name	Type	Default	Description
	ioState	ICImagingControlState&		Object used to maintain state of the function.
	inDeviceID	const Optional<const String&>&	NIL	Serial number, display name, unique name or base name of the camera

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

163. NET ICube

Table of content:

ICube_GenerateSoftwareTrigger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Generates software trigger in ICube device.

Syntax

```
void avl::ICube_GenerateSoftwareTrigger
(
 ICube_State& ioState,
 atl::Optional<int> inDeviceID
)
```

Parameters

	Name	Type	Default	Description
	ioState	ICube_State&		Object used to maintain state of the function.
	inDeviceID	Optional<int>	NIL	Index of a camera

Description

Generates software trigger. This filter executes the ICubeSDK_SetTrigger(camIndex, TRIG_SW_DO) ICube SDK function.

Remarks

To be able to use an ICube camera, you need to install the camera driver. You can find it at the following address (select binaries):

https://net-gmbh.com/en/machine-vision/products/cameras/usb2-icube/ or https://net-gmbh.com/en/machine-vision/products/cameras/usb3-vision-3icube/

Please make sure that the ICube SDK is installed properly on your computer. To verify the driver installation, you can run iControl.exe. If the camera was detected and you can see the image from it in this application, you can use your ICube camera in Aurora Vision Studio.

Recommended ICube SDK version for Aurora Vision Studio usage is **v2.0.4.8**.

The full description of the camera parameters can be found in the ICube SDK documentation.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- ICube_GrabImage – Captures an image from an ICube camera.
- ICube_GrabImage_WithTimeout – Captures an image from an ICube camera; returns Nil if no frame comes in the specified time.
- ICube_SetParameter – Sets a parameter of type Integer in an ICube device.
- ICube_SetTriggerMode – Sets a trigger mode in an ICube device.
- ICube_SetParamAuto – Sets a parameter of type auto in an ICube device.
- ICube_SetParamOnePush – Sets a parameter of type One Push in an ICube device.

ICube_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an image from an ICube camera.

Syntax

```
bool avl::ICube_GrabImage
(
 ICube_State& ioState,
 atl::Optional<int> inDeviceID,
 int inInputQueueSize,
 avl::ICubeGrabMode::Type inGrabbingMode,
 atl::Optional<avl::ICubeResolutionMode::Type> inResolutionMode,
 atl::Optional<avl::ICubeBinSkip::Type> inSkippingMode,
 atl::Optional<avl::ICubeBinSkip::Type> inBinningMode,
 atl::Optional<float> inExposureTime,
 atl::Optional<avl::Box> inRoi,
 avl::Image& outImage,
 int& outFrameID
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	ICube_State&			Object used to maintain state of the function.
	inDeviceID	Optional<int>	0 - ∞	NIL	Index of a camera
	inInputQueueSize	int	1 - 1000	10	
	inGrabbingMode	ICubeGrabMode::Type			
	inResolutionMode	Optional<ICubeResolutionMode::Type>		NIL	
	inSkippingMode	Optional<ICubeBinSkip::Type>		NIL	
	inBinningMode	Optional<ICubeBinSkip::Type>		NIL	
	inExposureTime	Optional<float>	0.0 - ∞	NIL	
	inRoi	Optional<Box>		NIL	Range of pixels to be processed
	outImage	Image&			Captured frame
	outFrameID	int&			Captured frame ID

Description

All the Auto parameters (those with checkboxes in the Property window) are initially set to the default values of the camera (they are initially not changed by Aurora Vision Studio). You can modify them by checking a checkbox and entering your own value.

Add DLL path to system environment variable may be required.

Recommended ICube version for usage with Aurora Vision Studio is **2.0.5.1**.

Changing parameters **inGrabbingMode** and **inResolutionMode** requires restarting acquisition. It is time consuming and shouldn't be done in each iteration.

The **inDeviceID** parameter is set only once when the filter is executed for the first time (if it is set to Auto, the first available camera in the system will be used). All the other parameters are set during the first filter execution or when the parameter is changed (they are not set in every iteration when the value is not changed).

To set more complex parameters, please use one of the following filters: ICube_SetParameter, ICube_SetParamAuto, ICube_SetParamOnePush, ICube_SetTriggerMode. Please note that if you set some parameter value in the ICube_GrabImage filter you should not modify it in any other ICube filter - it may lead to problems.

Remarks

To be able to use an ICube camera, you need to install the camera driver. You can find it at the following address (select binaries):

https://net-gmbh.com/en/machine-vision/products/cameras/usb2-icube/ or https://net-gmbh.com/en/machine-vision/products/cameras/usb3-vision-3icube/

Please make sure that the ICube SDK is installed properly on your computer. To verify the driver installation, you can run iControl.exe. If the camera was detected and you can see the image from it in this application, you can use your ICube camera in Aurora Vision Studio.

Recommended ICube SDK version for Aurora Vision Studio usage is **v2.0.4.8**.

The full description of the camera parameters can be found in the ICube documentation.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- ICube_GrabImage_WithTimeout – Captures an image from an ICube camera; returns Nil if no frame comes in the specified time.
- ICube_SetParameter – Sets a parameter of type Integer in an ICube device.
- ICube_SetTriggerMode – Sets a trigger mode in an ICube device.
- ICube_GenerateSoftwareTrigger – Generates software trigger in ICube device.
- ICube_SetParamAuto – Sets a parameter of type auto in an ICube device.
- ICube_SetParamOnePush – Sets a parameter of type One Push in an ICube device.

ICube_GrabImage_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an image from an ICube camera; returns Nil if no frame comes in the specified time.

Syntax

```
bool avl::ICube_GrabImage_WithTimeout
(
 ICube_State& ioState,
 atl::Optional<int> inDeviceID,
 int inInputQueueSize,
 avl::ICubeGrabMode::Type inGrabbingMode,
 int inTimeout,
 atl::Optional<avl::ICubeResolutionMode::Type> inResolutionMode,
 atl::Optional<avl::ICubeBinSkip::Type> inSkippingMode,
 atl::Optional<avl::ICubeBinSkip::Type> inBinningMode,
 atl::Optional<float> inExposureTime,
 atl::Optional<avl::Box> inRoi,
 atl::Conditional<avl::Image>& outImage,
 atl::Conditional<int>& outFrameID
)
```

Parameters

	Name	Type	Range	Default	Description
↻	ioState	ICube_State&			Object used to maintain state of the function.
→	inDeviceID	Optional<int>	0 - ∞	NIL	Index of a camera
→	inInputQueueSize	int	1 - 1000	10	
→	inGrabbingMode	ICubeGrabMode::Type			
→	inTimeout	int	0 - ∞	100	Maximum time to wait for frame in milliseconds
→	inResolutionMode	Optional<ICubeResolutionMode::Type>		NIL	
→	inSkippingMode	Optional<ICubeBinSkip::Type>		NIL	
→	inBinningMode	Optional<ICubeBinSkip::Type>		NIL	
→	inExposureTime	Optional<float>	0.0 - ∞	NIL	
→	inRoi	Optional<Box>		NIL	Range of pixels to be processed
←	outImage	Conditional<Image>&			Captured frame
←	outFrameID	Conditional<int>&			Captured frame ID

Description

All the Auto parameters (those with checkboxes in the Property window) are initially set to the default values of the camera (they are initially not changed by Aurora Vision Studio). You can modify them by checking a checkbox and entering your own value.

Add DLL path to system environment variable may be required.

Recommended ICube version for usage with Aurora Vision Studio is **2.0.5.1**.

Changing parameters **inGrabbingMode** and **inResolutionMode** requires restarting acquisition. It is time consuming and shouldn't be done in each iteration.

The **inDeviceID** parameter is set only once when the filter is executed for the first time (if it is set to Auto, the first available camera in the system will be used). All the other parameters are set during the first filter execution or when the parameter is changed (they are not set in every iteration when the value is not changed).

To set more complex parameters, please use one of the following filters: ICube_SetParameter, ICube_SetParamAuto, ICube_SetParamOnePush, ICube_SetTriggerMode. Please note that if you set some parameter value in the ICube_GrabImage filter you should not modify it in any other ICube filter - it may lead to problems.

Remarks

To be able to use an ICube camera, you need to install the camera driver. You can find it at the following address (select binaries):

https://net-gmbh.com/en/machine-vision/products/cameras/usb2-icube/ or https://net-gmbh.com/en/machine-vision/products/cameras/usb3-vision-3icube/

Please make sure that the ICube SDK is installed properly on your computer. To verify the driver installation, you can run iControl.exe. If the camera was detected and you can see the image from it in this application, you can use your ICube camera in Aurora Vision Studio.

Recommended ICube SDK version for Aurora Vision Studio usage is **v2.0.4.8**.

The full description of the camera parameters can be found in the ICube documentation.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- ICube_GrabImage – Captures an image from an ICube camera.
- ICube_SetParameter – Sets a parameter of type Integer in an ICube device.
- ICube_SetTriggerMode – Sets a trigger mode in an ICube device.
- ICube_GenerateSoftwareTrigger – Generates software trigger in ICube device.
- ICube_SetParamAuto – Sets a parameter of type auto in an ICube device.
- ICube_SetParamOnePush – Sets a parameter of type One Push in an ICube device.

ICube_SetParamAuto

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a parameter of type auto in an ICube device.

Syntax

```
void avl::ICube_SetParamAuto
(
 ICube_State& ioState,
 atl::Optional<int> inDeviceID,
 int inType,
 bool inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	ICube_State&		Object used to maintain state of the function.
	inDeviceID	Optional<int>	NIL	Index of a camera
	inType	int		Type of the parameter to set (see Parameter Definitions in the API Header)
	inValue	bool		1 == set auto, 0 == unset auto.

Description

If auto mode is supported, this filter sets/unsets auto mode of the parameter specified by **inType**. It executes the ICubeSDK_SetParamAuto ICube SDK function.

Remarks

To be able to use an ICube camera, you need to install the camera driver. You can find it at the following address (select binaries):

https://net-gmbh.com/en/machine-vision/products/cameras/usb2-icube/ or https://net-gmbh.com/en/machine-vision/products/cameras/usb3-vision-3icube/

Please make sure that the ICube SDK is installed properly on your computer. To verify the driver installation, you can run iControl.exe. If the camera was detected and you can see the image from it in this application, you can use your ICube camera in Aurora Vision Studio.

Recommended ICube SDK version for Aurora Vision Studio usage is **v2.0.4.8**.

The full description of the camera parameters can be found in the ICube SDK documentation.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- ICube_GrabImage – Captures an image from an ICube camera.
- ICube_GrabImage_WithTimeout – Captures an image from an ICube camera; returns Nil if no frame comes in the specified time.
- ICube_SetParameter – Sets a parameter of type Integer in an ICube device.
- ICube_SetTriggerMode – Sets a trigger mode in an ICube device.
- ICube_GenerateSoftwareTrigger – Generates software trigger in ICube device.
- ICube_SetParamOnePush – Sets a parameter of type One Push in an ICube device.

ICube_SetParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a parameter of type Integer in an ICube device.

Syntax

```
void avl::ICube_SetParameter
(
 ICube_State& ioState,
 atl::Optional<int> inDeviceID,
 int inType,
 int inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	ICube_State&		Object used to maintain state of the function.
	inDeviceID	Optional<int>	NIL	Index of a camera
	inType	int		Type of the parameter to set (see Parameter Definitions in the API Header)
	inValue	int		Value to set

Description

Sets value of the parameter specified by **inType**. This filter executes the ICubeSDK_SetCamParameter ICube SDK function.

Remarks

To be able to use an ICube camera, you need to install the camera driver. You can find it at the following address (select binaries):

https://net-gmbh.com/en/machine-vision/products/cameras/usb2-icube/ or https://net-gmbh.com/en/machine-vision/products/cameras/usb3-vision-3icube/

Please make sure that the ICube SDK is installed properly on your computer. To verify the driver installation, you can run iControl.exe. If the camera was detected and you can see the image from it in this application, you can use your ICube camera in Aurora Vision Studio.

Recommended ICube SDK version for Aurora Vision Studio usage is **v2.0.4.8**.

The full description of the camera parameters can be found in the ICube SDK documentation.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- ICube_GrabImage – Captures an image from an ICube camera.
- ICube_GrabImage_WithTimeout – Captures an image from an ICube camera; returns Nil if no frame comes in the specified time.
- ICube_SetTriggerMode – Sets a trigger mode in an ICube device.
- ICube_GenerateSoftwareTrigger – Generates software trigger in ICube device.
- ICube_SetParamAuto – Sets a parameter of type auto in an ICube device.
- ICube_SetParamOnePush – Sets a parameter of type One Push in an ICube device.

ICube_SetParamOnePush

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a parameter of type One Push in an ICube device.

Syntax

```
void avl::ICube_SetParamOnePush
(
 ICube_State& ioState,
 atl::Optional<int> inDeviceID,
 int inType
)
```

Parameters

	Name	Type	Default	Description
	ioState	ICube_State&		Object used to maintain state of the function.
	inDeviceID	Optional<int>	NIL	Index of a camera
	inType	int		Type of the parameter to set (see Parameter Definitions in the API Header)

Description

If one push mode is supported, this filter sets/unsets one push mode of the parameter specified by **inType**. This filter executes the ICubeSDK_SetParamOnePush ICube SDK function.

Remarks

To be able to use an ICube camera, you need to install the camera driver. You can find it at the following address (select binaries):

https://net-gmbh.com/en/machine-vision/products/cameras/usb2-icube/ or https://net-gmbh.com/en/machine-vision/products/cameras/usb3-vision-3icube/

Please make sure that the ICube SDK is installed properly on your computer. To verify the driver installation, you can run iControl.exe. If the camera was detected and you can see the image from it in this application, you can use your ICube camera in Aurora Vision Studio.

Recommended ICube SDK version for Aurora Vision Studio usage is **v2.0.4.8**.

The full description of the camera parameters can be found in the ICube SDK documentation.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- ICube_GrabImage – Captures an image from an ICube camera.
- ICube_GrabImage_WithTimeout – Captures an image from an ICube camera; returns Nil if no frame comes in the specified time.
- ICube_SetParameter – Sets a parameter of type Integer in an ICube device.
- ICube_SetTriggerMode – Sets a trigger mode in an ICube device.
- ICube_GenerateSoftwareTrigger – Generates software trigger in ICube device.
- ICube_SetParamAuto – Sets a parameter of type auto in an ICube device.

ICube_SetTriggerMode

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a trigger mode in an ICube device.

Syntax

```
void avl::ICube_SetTriggerMode
(
 ICube_State& ioState,
 atl::Optional<int> inDeviceID,
 avl::ICubeTriggerMode::Type inTriggerMode
)
```

Parameters

	Name	Type	Default	Description
	ioState	ICube_State&		Object used to maintain state of the function.
	inDeviceID	Optional<int>	NIL	Index of a camera
	inTriggerMode	ICubeTriggerMode::Type		Type of trigger mode (see Parameter Definitions in the API Header)

Description

Sets camera Trigger mode.

Remarks

To be able to use an ICube camera, you need to install the camera driver. You can find it at the following address (select binaries):

https://net-gmbh.com/en/machine-vision/products/cameras/usb2-icube/ or https://net-gmbh.com/en/machine-vision/products/cameras/usb3-vision-3icube/

Please make sure that the ICube SDK is installed properly on your computer. To verify the driver installation, you can run iControl.exe. If the camera was detected and you can see the image from it in this application, you can use your ICube camera in Aurora Vision Studio.

Recommended ICube SDK version for Aurora Vision Studio usage is **v2.0.4.8**.

The full description of the camera parameters can be found in the ICube SDK documentation.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- ICube_GrabImage – Captures an image from an ICube camera.
- ICube_GrabImage_WithTimeout – Captures an image from an ICube camera; returns Nil if no frame comes in the specified time.
- ICube_SetParameter – Sets a parameter of type Integer in an ICube device.
- ICube_GenerateSoftwareTrigger – Generates software trigger in ICube device.
- ICube_SetParamAuto – Sets a parameter of type auto in an ICube device.
- ICube_SetParamOnePush – Sets a parameter of type One Push in an ICube device.

ICube_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Initializes and starts image acquisition in a camera.

Syntax

```
void avl::ICube_StartAcquisition
(
 ICube_State& ioState,
 atl::Optional<int> inDeviceID,
 int inInputQueueSize,
 avl::ICubeGrabMode::Type inGrabbingMode,
 atl::Optional<avl::ICubeResolutionMode::Type> inResolutionMode,
 atl::Optional<avl::ICubeBinSkip::Type> inSkippingMode,
 atl::Optional<avl::ICubeBinSkip::Type> inBinningMode,
 atl::Optional<float> inExposureTime,
 atl::Optional<avl::Box> inRoi
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	ICube_State&			Object used to maintain state of the function.
	inDeviceID	Optional<int>	0 - ∞	NIL	Index of a camera
	inInputQueueSize	int	1 - 1000	10	
	inGrabbingMode	ICubeGrabMode::Type			
	inResolutionMode	Optional<ICubeResolutionMode::Type>		NIL	
	inSkippingMode	Optional<ICubeBinSkip::Type>		NIL	
	inBinningMode	Optional<ICubeBinSkip::Type>		NIL	
	inExposureTime	Optional<float>	0.0 - ∞	NIL	
	inRoi	Optional<Box>		NIL	Range of pixels to be processed

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

ICube_StopAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Stops image acquisition in a camera.

Syntax

```
void avl::ICube_StopAcquisition
(
 ICube_State& ioState,
 atl::Optional<int> inDeviceID
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	ICube_State&			Object used to maintain state of the function.
	inDeviceID	Optional<int>	0 - ∞	NIL	Index of a camera

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

164. IDS

Table of content:

IDS_CaptureStatus

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

The function returns information on errors that occurred during an image capture. All errors are listed that occurred since the last reset of the function.

Syntax

```
void avl::IDS_CaptureStatus
(
 IDS_BaseState& ioState,
 const atl::Optional<atl::String>& inDeviceID,
 bool inReset,
 int& outNumberOfErrors,
 int& outNoDestMem,
 int& outConversionFailed,
 int& outImageLocked,
 int& outOutOfBuffers,
 int& outDeviceNotReady,
 int& outUsbTransferFailed,
 int& outDevTimeout,
 int& outEthBufferOverrun,
 int& outEthDevMissedImages,
 int& outFrameCaptureFailed
)
```

Parameters

	Name	Type	Default	Description
	ioState	IDS_BaseState&		Object used to maintain state of the function.
	inDeviceID	const Optional<String>&	NIL	Device serial number or user definable camera ID
	inReset	bool		Resets the CaptureStatus information
	outNumberOfErrors	int&		Total number of errors
	outNoDestMem	int&		There is no destination memory for copying the finished image.
	outConversionFailed	int&		The current image could not be processed correctly.
	outImageLocked	int&		The destination buffers are locked and could not be written to.
	outOutOfBuffers	int&		No free internal image memory is available to the driver. The image was discarded.
	outDeviceNotReady	int&		The camera is no longer available. It is not possible to access images that have already been transferred.
	outUsbTransferFailed	int&		The image was not transferred over the USB bus.
	outDevTimeout	int&		The maximum allowable time for image capturing in the camera was exceeded.
	outEthBufferOverrun	int&		The sensor transfers more data than the internal camera memory of the GigE uEye camera can accommodate.
	outEthDevMissedImages	int&		Freerun mode: The camera could neither process nor output an image captured by the sensor. Hardware trigger mode : The camera received a hardware trigger signal which could not be processed because the sensor was still busy.
	outFrameCaptureFailed	int&		USB 3 uEye CP Rev. 2 with activated image memory only: The image was not transferred.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

IDS_ConfigureGPIO

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Configures digital inputs/outputs of IDS camera.

Syntax

```
void avl::IDS_ConfigureGPIO
(
 IDS_BaseState& ioState,
 const atl::Optional<atl::String>& inDeviceID,
 int inGPIO,
 avl::IDSGPIOConfiguration::Type inConfiguration,
 bool inState
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	IDS_BaseState&			Object used to maintain state of the function.
	inDeviceID	const Optional<String>&		NIL	Device serial number or user definable camera ID
	inGPIO	int	1 - 6		GPIO id
	inConfiguration	IDSGPIOConfiguration::Type			GPIO Configuration
	inState	bool			GPIO state (true = High, false = Low)

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install IDS SDK software.

IDS SDK can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

To verify the driver installation, you can run IDS Camera Manager. If the camera was detected and you can see the view from the camera, you can use IDS SDK in Aurora Vision Studio.

Recommended IDS SDK version for Aurora Vision Studio usage is **4.94**.

Camera identification

When there is only one IDS camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Source code

In Professional edition this filter is open source. You can use this filter as reference when implementing support for your specific hardware. You can also modify this filter and add some additional functionality.

The source code is located in the directory:

```
Public Documents\Aurora Vision Studio 4.x Professional\Sources\UserFilters\IDS
```

Typically it is:

```
C:\Users\Public Documents\Aurora Vision Studio 4.x Professional\Sources\UserFilters\IDS
```

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- IDS_GrabImage – Captures an image from an IDS camera.
- IDS_GrabImage_WithTimeout – Captures an image from an IDS camera.
- IDS_StartAcquisition – Initialize2s and starts image acquisition in a camera.
- IDS_LoadParameterSet – Loads specific camera parameters.

IDS_ForceTrigger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Forces software-controlled captures of an image while a capturing process triggered by hardware is in progress.

Syntax

```
void avl::IDS_ForceTrigger
(
 IDS_BaseState& ioState,
 const atl::Optional<atl::String>& inDeviceID
)
```

Parameters

	Name	Type	Default	Description
	ioState	IDS_BaseState&		Object used to maintain state of the function.
	inDeviceID	const Optional<String>&	NIL	Device serial number or user definable camera ID

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install IDS SDK software.

IDS SDK can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

To verify the driver installation, you can run IDS Camera Manager. If the camera was detected and you can see the view from the camera, you can use IDS SDK in Aurora Vision Studio.

Recommended IDS SDK version for Aurora Vision Studio usage is **4.94**.

Camera identification

When there is only one IDS camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Source code

In Professional edition this filter is open source. You can use this filter as reference when implementing support for your specific hardware. You can also modify this filter and add some additional functionality.

The source code is located in the directory:

```
Public Documents\Aurora Vision Studio 4.x Professional\Sources\UserFilters\IDS
```

Typically it is:

```
C:\Users\Public Documents\Aurora Vision Studio 4.x Professional\Sources\UserFilters\IDS
```

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- IDS_GrabImage – Captures an image from an IDS camera.
- IDS_GrabImage_WithTimeout – Captures an image from an IDS camera.
- IDS_StartAcquisition – Initialize2s and starts image acquisition in a camera.
- IDS_LoadParameterSet – Loads specific camera parameters.

IDS_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an image from an IDS camera.

Syntax

```
bool avl::IDS_GrabImage
(
 IDS_BaseState& ioState,
 const atl::Optional<atl::String>& inDeviceID,
 const atl::String& inPixelFormat,
 int inInputQueueSize,
 bool inAutoReconnect,
 const atl::Optional<avl::IDSTriggerMode::Type>& inTriggerMode,
 const atl::Optional<avl::Box>& inAoi,
 const atl::Optional<float>& inFrameRate,
 const atl::Optional<float>& inExposureTime,
 const atl::Optional<avl::IDSBinning::Type>& inHorizontalBinning,
 const atl::Optional<avl::IDSBinning::Type>& inVerticalBinning,
 const atl::Optional<avl::IDSMirror::Type>& inMirror,
 const atl::Optional<bool>& inAutoBlackLevel,
 const atl::Optional<int>& inBlackLevelOffset,
 const atl::Optional<int>& inGamma,
 const atl::Optional<bool>& inGainBoost,
 const atl::Optional<int>& inGainMaster,
 const atl::Optional<int>& inGainRed,
 const atl::Optional<int>& inGainGreen,
 const atl::Optional<int>& inGainBlue,
 avl::Image& outImage,
 atl::int64& outFrameNumber,
 atl::int64& outTimestamp
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	IDS_BaseState&			Object used to maintain state of the function.
	inDeviceID	const Optional<String>&		NIL	Device serial number or user definable camera ID
	inPixelFormat	const String&		\"Mono8\"	Pixel format. Supported values: Raw8, Mono8, Rgb8, Bgr8, Rgba8, Bgra8
	inInputQueueSize	int	1 - ∞	3	Capacity of input frames queue
	inAutoReconnect	bool		True	Automatically reconnect with the camera
	inTriggerMode	const Optional<IDSTriggerMode::Type>&		NIL	Camera trigger mode
	inAoi	const Optional<Box>&		NIL	Required fragment of image to stream
	inFrameRate	const Optional<float>&	0.1 - 400.0	NIL	Frame rate
	inExposureTime	const Optional<float>&	0.0 - ∞	NIL	Exposure time in microseconds
	inHorizontalBinning	const Optional<IDSBinning::Type>&		NIL	Horizontal binning
	inVerticalBinning	const Optional<IDSBinning::Type>&		NIL	Vertical binning
	inMirror	const Optional<IDSMirror::Type>&		NIL	Mirror effect
	inAutoBlackLevel	const Optional<bool>&		NIL	Enable auto black level
	inBlackLevelOffset	const Optional<int>&	0 - ∞	NIL	Black level offset
	inGamma	const Optional<int>&	1 - 1000	NIL	Set gamma value
	inGainBoost	const Optional<bool>&		NIL	Set gain boost mode
	inGainMaster	const Optional<int>&	0 - 100	NIL	Set gain master value
	inGainRed	const Optional<int>&	0 - 100	NIL	Set gain red value
	inGainGreen	const Optional<int>&	0 - 100	NIL	Set gain green value
	inGainBlue	const Optional<int>&	0 - 100	NIL	Set gain blue value
	outImage	Image&			Output image
	outFrameNumber	int64&			Output frame number
	outTimestamp	int64&			Output image timestamp

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install IDS SDK software.

IDS SDK can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

To verify the driver installation, you can run IDS Camera Manager. If the camera was detected and you can see the view from the camera, you can use IDS SDK in Aurora Vision Studio.

Recommended IDS SDK version for Aurora Vision Studio usage is **4.94**.

Camera identification

When there is only one IDS camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Source code

In Professional edition this filter is open source. You can use this filter as reference when implementing support for your specific hardware. You can also modify this filter and add some additional functionality.

The source code is located in the directory:

```
Public Documents\Aurora Vision Studio 4.x Professional\Sources\UserFilters\IDS
```

Typically it is:

```
C:\Users\Public Documents\Aurora Vision Studio 4.x Professional\Sources\UserFilters\IDS
```

Working with camera

If you work with USB cameras, then **inAutoReconnect** parameter as well as camera disconnection detection may not work. To enable these features in USB cameras, IDS SDK requires the user to manually add the following registry key:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\ueye\Parameters
DWORD DevChangeHandlerMode = 1
```

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- IDS_GrabImage_WithTimeout – Captures an image from an IDS camera.
- IDS_StartAcquisition – Initialize2s and starts image acquisition in a camera.
- IDS_LoadParameterSet – Loads specific camera parameters.
- IDS_ForceTrigger – Forces software-controlled captures of an image while a capturing process triggered by hardware is in progress.

IDS_GrabImage_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an image from an IDS camera.

Syntax

```
bool avl::IDS_GrabImage_WithTimeout
(
 IDS_BaseState& ioState,
 const atl::Optional<atl::String>& inDeviceID,
 const atl::String& inPixelFormat,
 int inInputQueueSize,
 bool inAutoReconnect,
 const atl::Optional<avl::IDSTriggerMode::Type>& inTriggerMode,
 int inTimeout,
 const atl::Optional<avl::Box>& inAoi,
 const atl::Optional<float>& inFrameRate,
 const atl::Optional<float>& inExposureTime,
 const atl::Optional<avl::IDSBinning::Type>& inHorizontalBinning,
 const atl::Optional<avl::IDSBinning::Type>& inVerticalBinning,
 const atl::Optional<avl::IDSMirror::Type>& inMirror,
 const atl::Optional<bool>& inAutoBlackLevel,
 const atl::Optional<int>& inBlackLevelOffset,
 const atl::Optional<int>& inGamma,
 const atl::Optional<bool>& inGainBoost,
 const atl::Optional<int>& inGainMaster,
 const atl::Optional<int>& inGainRed,
 const atl::Optional<int>& inGainGreen,
 const atl::Optional<int>& inGainBlue,
 atl::Conditional<avl::Image>& outImage,
 atl::Conditional<atl::int64>& outFrameNumber,
 atl::Conditional<atl::int64>& outTimestamp
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	IDS_BaseState&			Object used to maintain state of the function.
	inDeviceID	const Optional<String>&		NIL	Device serial number or user definable camera ID
	inPixelFormat	const String&		\"Mono8\"	Pixel format. Supported values: Raw8, Mono8, Rgb8, Bgr8, Rgba8, Bgra8
	inInputQueueSize	int	1 - ∞	3	Capacity of input frames queue
	inAutoReconnect	bool		True	Automatically reconnect with the camera
	inTriggerMode	const Optional<IDSTriggerMode::Type>&		NIL	Camera trigger mode
	inTimeout	int	10 - ∞	100	Maximum time to wait for frame in milliseconds
	inAoi	const Optional<Box>&		NIL	Required fragment of image to stream
	inFrameRate	const Optional<float>&	0.1 - 400.0	NIL	Frame rate
	inExposureTime	const Optional<float>&	0.0 - ∞	NIL	Exposure time in microseconds
	inHorizontalBinning	const Optional<IDSBinning::Type>&		NIL	Horizontal binning
	inVerticalBinning	const Optional<IDSBinning::Type>&		NIL	Vertical binning
	inMirror	const Optional<IDSMirror::Type>&		NIL	Mirror effect
	inAutoBlackLevel	const Optional<bool>&		NIL	Enable auto black level
	inBlackLevelOffset	const Optional<int>&	0 - ∞	NIL	Black level offset
	inGamma	const Optional<int>&	1 - 1000	NIL	Set gamma value
	inGainBoost	const Optional<bool>&		NIL	Set gain boost mode
	inGainMaster	const Optional<int>&	0 - 100	NIL	Set gain master value
	inGainRed	const Optional<int>&	0 - 100	NIL	Set gain red value
	inGainGreen	const Optional<int>&	0 - 100	NIL	Set gain green value
	inGainBlue	const Optional<int>&	0 - 100	NIL	Set gain blue value
	outImage	Conditional<Image>&			Output image
	outFrameNumber	Conditional<int64>&			Output frame number
	outTimestamp	Conditional<int64>&			Output image timestamp

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install IDS SDK software.

IDS SDK can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

To verify the driver installation, you can run IDS Camera Manager. If the camera was detected and you can see the view from the camera, you can use IDS SDK in Aurora Vision Studio.

Recommended IDS SDK version for Aurora Vision Studio usage is **4.94**.

Camera identification

When there is only one IDS camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Source code

In Professional edition this filter is open source. You can use this filter as reference when implementing support for your specific hardware. You can also modify this filter and add some additional functionality.

The source code is located in the directory:

```
Public Documents\Aurora Vision Studio 4.x Professional\Sources\UserFilters\IDS
```

Typically it is:

```
C:\Users\Public Documents\Aurora Vision Studio 4.x Professional\Sources\UserFilters\IDS
```

Working with camera

If you work with USB cameras, then **inAutoReconnect** parameter as well as camera disconnection detection may not work. To enable these features in USB cameras, IDS SDK requires the user to manually add the following registry key:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\ueye\Parameters
DWORD DevChangeHandlerMode = 1
```

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- IDS_GrabImage – Captures an image from an IDS camera.
- IDS_StartAcquisition – Initialize2s and starts image acquisition in a camera.
- IDS_LoadParameterSet – Loads specific camera parameters.
- IDS_ForceTrigger – Forces software-controlled captures of an image while a capturing process triggered by hardware is in progress.

IDS_LoadParameterSet

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Loads specific camera parameters.

Syntax

```
void avl::IDS_LoadParameterSet
(
 IDS_BaseState& ioState,
 const atl::Optional<atl::String>& inDeviceID,
 bool inLoadFromEEPROM,
 const atl::File& inFile
)
```

Parameters

	Name	Type	Default	Description
	ioState	IDS_BaseState&		Object used to maintain state of the function.
	inDeviceID	const Optional<String>&	NIL	Device serial number or user definable camera ID
	inLoadFromEEPROM	bool		If true loads configuration from EEPROM
	inFile	const File&		Parameters file path

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install IDS SDK software.

IDS SDK can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

To verify the driver installation, you can run IDS Camera Manager. If the camera was detected and you can see the view from the camera, you can use IDS SDK in Aurora Vision Studio.

Recommended IDS SDK version for Aurora Vision Studio usage is **4.94**.

Camera identification

When there is only one IDS camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Source code

In Professional edition this filter is open source. You can use this filter as reference when implementing support for your specific hardware. You can also modify this filter and add some additional functionality.

The source code is located in the directory:

```
Public Documents\Aurora Vision Studio 4.x Professional\Sources\UserFilters\IDS
```

Typically it is:

```
C:\Users\Public Documents\Aurora Vision Studio 4.x Professional\Sources\UserFilters\IDS
```

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- IDS_GrabImage – Captures an image from an IDS camera.
- IDS_GrabImage_WithTimeout – Captures an image from an IDS camera.
- IDS_StartAcquisition – Initialize2s and starts image acquisition in a camera.
- IDS_ForceTrigger – Forces software-controlled captures of an image while a capturing process triggered by hardware is in progress.

IDS_SetParameters

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets camera parameters.

Syntax

```
void avl::IDS_SetParameters
(
 IDS_BaseState& ioState,
 const atl::Optional<atl::String>& inDeviceID,
 const atl::Optional<avl::Box>& inAoi,
 const atl::Optional<float>& inFrameRate,
 const atl::Optional<float>& inExposureTime,
 const atl::Optional<avl::IDSBinning::Type>& inHorizontalBinning,
 const atl::Optional<avl::IDSBinning::Type>& inVerticalBinning,
 const atl::Optional<avl::IDSMirror::Type>& inMirror,
 const atl::Optional<bool>& inAutoBlackLevel,
 const atl::Optional<int>& inBlackLevelOffset,
 const atl::Optional<int>& inGamma,
 const atl::Optional<bool>& inGainBoost,
 const atl::Optional<int>& inGainMaster,
 const atl::Optional<int>& inGainRed,
 const atl::Optional<int>& inGainGreen,
 const atl::Optional<int>& inGainBlue
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	IDS_BaseState&			Object used to maintain state of the function.
	inDeviceID	const Optional<String>&		NIL	Device serial number or user definable camera ID
	inAoi	const Optional<Box>&		NIL	Required fragment of image to stream
	inFrameRate	const Optional<float>&	0.1 - 400.0	NIL	Frame rate
	inExposureTime	const Optional<float>&	0.0 - ∞	NIL	Exposure time in microseconds
	inHorizontalBinning	const Optional<IDSBinning::Type>&		NIL	Horizontal binning
	inVerticalBinning	const Optional<IDSBinning::Type>&		NIL	Vertical binning
	inMirror	const Optional<IDSMirror::Type>&		NIL	Mirror effect
	inAutoBlackLevel	const Optional<bool>&		NIL	Enable auto black level
	inBlackLevelOffset	const Optional<int>&	0 - ∞	NIL	Black level offset
	inGamma	const Optional<int>&	1 - 1000	NIL	Set gamma value
	inGainBoost	const Optional<bool>&		NIL	Set gain boost mode
	inGainMaster	const Optional<int>&	0 - 100	NIL	Set gain master value
	inGainRed	const Optional<int>&	0 - 100	NIL	Set gain red value
	inGainGreen	const Optional<int>&	0 - 100	NIL	Set gain green value
	inGainBlue	const Optional<int>&	0 - 100	NIL	Set gain blue value

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install IDS SDK software.

IDS SDK can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

To verify the driver installation, you can run IDS Camera Manager. If the camera was detected and you can see the view from the camera, you can use IDS SDK in Aurora Vision Studio.

Recommended IDS SDK version for Aurora Vision Studio usage is **4.94**.

Camera identification

When there is only one IDS camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Source code

In Professional edition this filter is open source. You can use this filter as reference when implementing support for your specific hardware. You can also modify this filter and add some additional functionality.

The source code is located in the directory:

```
Public Documents\Aurora Vision Studio 4.x Professional\Sources\UserFilters\IDS
```

Typically it is:

```
C:\Users\Public Documents\Aurora Vision Studio 4.x Professional\Sources\UserFilters\IDS
```

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- IDS_GrabImage – Captures an image from an IDS camera.
- IDS_GrabImage_WithTimeout – Captures an image from an IDS camera.
- IDS_StartAcquisition – Initialize2s and starts image acquisition in a camera.
- IDS_ForceTrigger – Forces software-controlled captures of an image while a capturing process triggered by hardware is in progress.

IDS_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Initialize2s and starts image acquisition in a camera.

Syntax

```
void avl::IDS_StartAcquisition
(
 IDS_BaseState& ioState,
 const atl::Optional<atl::String>& inDeviceID,
 const atl::String& inPixelFormat,
 int inInputQueueSize,
 bool inAutoReconnect,
 const atl::Optional<avl::IDSTriggerMode::Type>& inTriggerMode,
 const atl::Optional<avl::Box>& inAoi,
 const atl::Optional<float>& inFrameRate,
 const atl::Optional<float>& inExposureTime,
 const atl::Optional<avl::IDSBinning::Type>& inHorizontalBinning,
 const atl::Optional<avl::IDSBinning::Type>& inVerticalBinning,
 const atl::Optional<avl::IDSMirror::Type>& inMirror,
 const atl::Optional<bool>& inAutoBlackLevel,
 const atl::Optional<int>& inBlackLevelOffset,
 const atl::Optional<int>& inGamma,
 const atl::Optional<bool>& inGainBoost,
 const atl::Optional<int>& inGainMaster,
 const atl::Optional<int>& inGainRed,
 const atl::Optional<int>& inGainGreen,
 const atl::Optional<int>& inGainBlue
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	IDS_BaseState&			Object used to maintain state of the function.
	inDeviceID	const Optional<String>&		NIL	Device serial number or user definable camera ID
	inPixelFormat	const String&		\"Mono8\"	Pixel format. Supported values: Raw8, Mono8, Rgb8, Bgr8, Rgba8, Bgra8
	inInputQueueSize	int	1 - ∞	3	Capacity of input frames queue
	inAutoReconnect	bool		True	Automatically reconnect with the camera
	inTriggerMode	const Optional<IDSTriggerMode::Type>&		NIL	Camera trigger mode
	inAoi	const Optional<Box>&		NIL	Required fragment of image to stream
	inFrameRate	const Optional<float>&	0.1 - 400.0	NIL	Frame rate
	inExposureTime	const Optional<float>&	0.0 - ∞	NIL	Exposure time in microseconds
	inHorizontalBinning	const Optional<IDSBinning::Type>&		NIL	Horizontal binning
	inVerticalBinning	const Optional<IDSBinning::Type>&		NIL	Vertical binning
	inMirror	const Optional<IDSMirror::Type>&		NIL	Mirror effect
	inAutoBlackLevel	const Optional<bool>&		NIL	Enable auto black level
	inBlackLevelOffset	const Optional<int>&	0 - ∞	NIL	Black level offset
	inGamma	const Optional<int>&	1 - 1000	NIL	Set gamma value
	inGainBoost	const Optional<bool>&		NIL	Set gain boost mode
	inGainMaster	const Optional<int>&	0 - 100	NIL	Set gain master value
	inGainRed	const Optional<int>&	0 - 100	NIL	Set gain red value
	inGainGreen	const Optional<int>&	0 - 100	NIL	Set gain green value
	inGainBlue	const Optional<int>&	0 - 100	NIL	Set gain blue value

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install IDS SDK software.

IDS SDK can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

To verify the driver installation, you can run IDS Camera Manager. If the camera was detected and you can see the view from the camera, you can use IDS SDK in Aurora Vision Studio.

Recommended IDS SDK version for Aurora Vision Studio usage is **4.94**.

Camera identification

When there is only one IDS camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Source code

In Professional edition this filter is open source. You can use this filter as reference when implementing support for your specific hardware. You can also modify this filter and add some additional functionality.

The source code is located in the directory:

```
Public Documents\Aurora Vision Studio 4.x Professional\Sources\UserFilters\IDS
```

Typically it is:

```
C:\Users\Public Documents\Aurora Vision Studio 4.x Professional\Sources\UserFilters\IDS
```

Working with camera

If you work with USB cameras then parameter **inAutoReconnect** doesn't work. Camera disconnection also can't be detected. To enable these features in USB cameras please add new register key:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\ueye\Parameters
DWORD DevChangeHandlerMode = 1
```

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- IDS_GrabImage – Captures an image from an IDS camera.
- IDS_GrabImage_WithTimeout – Captures an image from an IDS camera.
- IDS_LoadParameterSet – Loads specific camera parameters.
- IDS_ForceTrigger – Forces software-controlled captures of an image while a capturing process triggered by hardware is in progress.

IDS_StopAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Stops image acquisition in a camera.

Syntax

```
void avl::IDS_StopAcquisition
(
 IDS_BaseState& ioState,
 const atl::Optional<atl::String>& inDeviceID
)
```

Parameters

	Name	Type	Default	Description
	ioState	IDS_BaseState&		Object used to maintain state of the function.
	inDeviceID	const Optional<String>&	NIL	Device serial number or user definable camera ID

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install IDS SDK software.

IDS SDK can be downloaded from the following website: https://en.ids-imaging.com/download-ueye.html

To verify the driver installation, you can run IDS Camera Manager. If the camera was detected and you can see the view from the camera, you can use IDS SDK in Aurora Vision Studio.

Recommended IDS SDK version for Aurora Vision Studio usage is **4.94**.

Camera identification

When there is only one IDS camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Source code

In Professional edition this filter is open source. You can use this filter as reference when implementing support for your specific hardware. You can also modify this filter and add some additional functionality.

The source code is located in the directory:

```
Public Documents\Aurora Vision Studio 4.x Professional\Sources\UserFilters\IDS
```

Typically it is:

```
C:\Users\Public Documents\Aurora Vision Studio 4.x Professional\Sources\UserFilters\IDS
```

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- IDS_GrabImage – Captures an image from an IDS camera.
- IDS_StartAcquisition – Initialize2s and starts image acquisition in a camera.
- IDS_GrabImage_WithTimeout – Captures an image from an IDS camera.
- IDS_LoadParameterSet – Loads specific camera parameters.
- IDS_ForceTrigger – Forces software-controlled captures of an image while a capturing process triggered by hardware is in progress.

165. IFM

Table of content:

IFM_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an image from an ifm device.

Syntax

```
bool avl::IFM_GrabImage
(
 Ifm_State& ioState,
 const avl::IFMCameraAddress& inAddress,
 bool inGrab3dCoordinates,
 const atl::Optional<float>& inFrameRate,
 const atl::Optional<avl::IFMResolution::Type>& inResolution,
 const atl::Optional<avl::IFMTriggerMode::Type>& inTriggerMode,
 avl::Image& outDistancesImage,
 avl::Image& outAmplitudeImage,
 atl::Array<avl::Region>& outInvalidRois,
 atl::Conditional<avl::Point3DGrid>& out3dCoordinates
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	Ifm_State&			Object used to maintain state of the function.
→	inAddress	const IFMCameraAddress&			Device identifying address
→	inGrab3dCoordinates	bool			Enable 3d coordinates capturing
→	inFrameRate	const Optional<float>&	0.017 - 300.0	NIL	Sets the target frame rate in fps at which device should capture images
→	inResolution	const Optional<IFMResolution::Type>&		NIL	Sets the resolution of output image
→	inTriggerMode	const Optional<IFMTriggerMode::Type>&		NIL	Sets the trigger mode on device for capturing image
←	outDistancesImage	Image&			Captured distance image
←	outAmplitudeImage	Image&			Captured amplitude image
←	outInvalidRois	Array<Region>&			Region array with invalid pixels: INVALID, SATURATED, INCONSISTENT and LOW_SIGNAL.
←	out3dCoordinates	Conditional<Point3DGrid>&			Captured 3d coordinates

Remarks

Device driver software

This filter is intended to cooperate with a device using PMDSDK 2. In order to connect with the device, it is required to download PMDSDK 2.

PMDSDK 2 can be downloaded from the following website: https://www.ifm.com/us/en/search#!/infomaterialanddownloads?_type=infomaterialanddownloads&docType=download. Please download "O3D303 Software Development Kit"

After download you have to manually copy pmdaccess2.dll file to Aurora Vision Studio main directory (usually located in Program Files). Alternatively you can add directory with this dll file to PATH environment variable

Additionally in **inAddress** input you will need to provide camera plugin and processing plugin file path.

It is recommended to disable DHCP in the device and using static IP address.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- IFM_GrabImage_WithTimeout – Captures an image from an ifm device.
- IFM_SourceCommand – Issues a device-dependent command.
- IFM_SetIntegrationTime – Sets the integration time of the device.
- IFM_StartAcquisition – Initializes and starts image acquisition in a device.
- IFM_SetDigitalOutput – Sets the logic state of a specific io.

IFM_GrabImage_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an image from an ifm device.

Syntax

```
bool avl::IFM_GrabImage_WithTimeout
(
 Ifm_State& ioState,
 const avl::IFMCameraAddress& inAddress,
 int inTimeout,
 bool inGrab3dCoordinates,
 const atl::Optional<float>& inFrameRate,
 const atl::Optional<avl::IFMResolution::Type>& inResolution,
 const atl::Optional<avl::IFMTriggerMode::Type>& inTriggerMode,
 atl::Conditional<avl::Image>& outDistancesImage,
 atl::Conditional<avl::Image>& outAmplitudeImage,
 atl::Conditional<atl::Array<avl::Region>>& outInvalidRois,
 atl::Conditional<avl::Point3DGrid>& out3dCoordinates
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Ifm_State&			Object used to maintain state of the function.
	inAddress	const IFMCameraAddress&			Device identifying address
	inTimeout	int	0 - ∞	100	Maximum time to wait for frame in milliseconds
	inGrab3dCoordinates	bool			Enable 3d coordinates capturing
	inFrameRate	const Optional<float>&	0.017 - 300.0	NIL	Sets the target frame rate in fps at which device should capture images
	inResolution	const Optional<IFMResolution::Type>&		NIL	Sets the resolution of output image
	inTriggerMode	const Optional<IFMTriggerMode::Type>&		NIL	Sets the trigger mode on device for capturing image
	outDistancesImage	Conditional<Image>&			Captured distance image
	outAmplitudeImage	Conditional<Image>&			Captured amplitude image
	outInvalidRois	Conditional<Array<Region>>&			Region array with invalid pixels: INVALID, SATURATED, INCONSISTENT and LOW_SIGNAL.
	out3dCoordinates	Conditional<Point3DGrid>&			Captured 3d coordinates

Remarks

Device driver software

This filter is intended to cooperate with a device using PMDSDK 2. In order to connect with the device, it is required to download PMDSDK 2.

PMDSDK 2 can be downloaded from the following website: https://www.ifm.com/us/en/search#!/infomaterialanddownloads?_type=infomaterialanddownloads&docType=download. Please download "O3D303 Software Development Kit"

After download you have to manually copy pmdaccess2.dll file to Aurora Vision Studio main directory (usually located in Program Files). Alternatively you can add directory with this dll file to PATH environment variable

Additionally in **inAddress** input you will need to provide camera plugin and processing plugin file path.

It is recommended to disable DHCP in the device and using static IP address.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- IFM_GrabImage – Captures an image from an ifm device.
- IFM_SourceCommand – Issues a device-dependent command.
- IFM_SetIntegrationTime – Sets the integration time of the device.
- IFM_StartAcquisition – Initializes and starts image acquisition in a device.
- IFM_SetDigitalOutput – Sets the logic state of a specific io.

IFM_SetDigitalOutput

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets the logic state of a specific io.

Syntax

```
void avl::IFM_SetDigitalOutput
(
 Ifm_State& ioState,
 const avl::IFMCameraAddress& inAddress,
 const int inNumber,
 const bool inState
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ifm_State&		Object used to maintain state of the function.
	inAddress	const IFMCameraAddress&		Device identifying address
	inNumber	const int		Digital output number
	inState	const bool		Output state

Remarks

Device driver software

This filter is intended to cooperate with a device using PMDSDK 2. In order to connect with the device, it is required to download PMDSDK 2.

PMDSDK 2 can be downloaded from the following website: https://www.ifm.com/us/en/search#!/infomaterialanddownloads?_type=infomaterialanddownloads&docType=download. Please download "O3D303 Software Development Kit"

After download you have to manually copy pmdaccess2.dll file to Aurora Vision Studio main directory (usually located in Program Files). Alternatively you can add directory with this dll file to PATH environment variable

Additionally in **inAddress** input you will need to provide camera plugin and processing plugin file path.

It is recommended to disable DHCP in the device and using static IP address.

Setting digital outputs

This function doesn't work with ifm sensors. It works only with ifm cameras.

The filter is equivalent to execution of IFM_SourceCommand with **inCommand** = "SendPCICommands o<IO-ID><IO-state>"

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- IFM_GrabImage – Captures an image from an ifm device.
- IFM_GrabImage_WithTimeout – Captures an image from an ifm device.
- IFM_SourceCommand – Issues a device-dependent command.
- IFM_SetIntegrationTime – Sets the integration time of the device.
- IFM_StartAcquisition – Initializes and starts image acquisition in a device.

IFM_SetIntegrationTime

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets the integration time of the device.

Syntax

```
void avl::IFM_SetIntegrationTime
(
 Ifm_State& ioState,
 const avl::IFMCameraAddress& inAddress,
 const int inIndex,
 const int inTime
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Ifm_State&			Object used to maintain state of the function.
	inAddress	const IFMCameraAddress&			Device identifying address
	inIndex	const int	0 - +∞		Index of the integration time
	inTime	const int	0 - +∞		Integration time in microseconds

Remarks

Device driver software

This filter is intended to cooperate with a device using PMDSDK 2. In order to connect with the device, it is required to download PMDSDK 2.

PMDSDK 2 can be downloaded from the following website: https://www.ifm.com/us/en/search#!/infomaterialanddownloads?_type=infomaterialanddownloads&docType=download. Please download "O3D303 Software Development Kit"

After download you have to manually copy pmdaccess2.dll file to Aurora Vision Studio main directory (usually located in Program Files). Alternatively you can add directory with this dll file to PATH environment variable

Additionally in **inAddress** input you will need to provide camera plugin and processing plugin file path.

It is recommended to disable DHCP in the device and using static IP address.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- IFM_GrabImage – Captures an image from an ifm device.
- IFM_GrabImage_WithTimeout – Captures an image from an ifm device.
- IFM_SourceCommand – Issues a device-dependent command.
- IFM_StartAcquisition – Initializes and starts image acquisition in a device.
- IFM_SetDigitalOutput – Sets the logic state of a specific io.

IFM_SourceCommand

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Issues a device-dependent command.

Syntax

```
void avl::IFM_SourceCommand
(
 Ifm_State& ioState,
 const avl::IFMCameraAddress& inAddress,
 const atl::String& inCommand,
 atl::String& outResult
)
```

Parameters

	Name	Type	Default	Description
	ioState	Ifm_State&		Object used to maintain state of the function.
	inAddress	const IFMCameraAddress&		Device identifying address
	inCommand	const String&		The command to be executed
	outResult	String&		Result string

Remarks

Device driver software

This filter is intended to cooperate with a device using PMDSDK 2. In order to connect with the device, it is required to download PMDSDK 2.

PMDSDK 2 can be downloaded from the following website: https://www.ifm.com/us/en/search#!/infomaterialanddownloads?_type=infomaterialanddownloads&docType=download. Please download "O3D303 Software Development Kit"

After download you have to manually copy pmdaccess2.dll file to Aurora Vision Studio main directory (usually located in Program Files). Alternatively you can add directory with this dll file to PATH environment variable

Additionally in **inAddress** input you will need to provide camera plugin and processing plugin file path.

It is recommended to disable DHCP in the device and using static IP address.

Source Commands

Detailed description of available commands are in O3D3xxCamera_PMD_SDK_Manual_ifm.pdf document distributed with "O3D303 Software Development Kit".

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- IFM_GrabImage – Captures an image from an ifm device.
- IFM_GrabImage_WithTimeout – Captures an image from an ifm device.
- IFM_SetIntegrationTime – Sets the integration time of the device.
- IFM_StartAcquisition – Initializes and starts image acquisition in a device.
- IFM_SetDigitalOutput – Sets the logic state of a specific io.

IFM_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Initializes and starts image acquisition in a device.

Syntax

```
void avl::IFM_StartAcquisition
(
 Ifm_State& ioState,
 const avl::IFMCameraAddress& inAddress,
 const atl::Optional<float>& inFrameRate,
 const atl::Optional<avl::IFMResolution::Type>& inResolution,
 const atl::Optional<avl::IFMTriggerMode::Type>& inTriggerMode
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Ifm_State&			Object used to maintain state of the function.
	inAddress	const IFMCameraAddress&			Device identifying address
	inFrameRate	const Optional<float>&	0.017 - 300.0	NIL	Sets the target frame rate in fps at which device should capture images
	inResolution	const Optional<IFMResolution::Type>&		NIL	Sets the resolution of output image
	inTriggerMode	const Optional<IFMTriggerMode::Type>&		NIL	Sets the trigger mode on device for capturing image

Remarks

Device driver software

This filter is intended to cooperate with a device using PMDSDK 2. In order to connect with the device, it is required to download PMDSDK 2.

PMDSDK 2 can be downloaded from the following website: https://www.ifm.com/us/en/search#!/infomaterialanddownloads?_type=infomaterialanddownloads&docType=download. Please download "O3D303 Software Development Kit"

After download you have to manually copy pmdaccess2.dll file to Aurora Vision Studio main directory (usually located in Program Files). Alternatively you can add directory with this dll file to PATH environment variable

Additionally in **inAddress** input you will need to provide camera plugin and processing plugin file path.

It is recommended to disable DHCP in the device and using static IP address.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- IFM_GrabImage – Captures an image from an ifm device.
- IFM_GrabImage_WithTimeout – Captures an image from an ifm device.
- IFM_SourceCommand – Issues a device-dependent command.
- IFM_SetIntegrationTime – Sets the integration time of the device.
- IFM_SetDigitalOutput – Sets the logic state of a specific io.

166. JAI

Table of content:

JAI_GenerateSoftwareTrigger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Generates a software trigger.

Syntax

```
void avl::JAI_GenerateSoftwareTrigger
(
 JAI_State& ioState,
 atl::Optional<const atl::String&> inDeviceID
)
```

Parameters

	Name	Type	Default	Description
↔	ioState	JAI_State&		Object used to maintain state of the function.
→	inDeviceID	Optional<const String&>	NIL	Device identifying number

JAI_GetBooleanParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets a boolean parameter of the JAI device.

Syntax

```
void avl::JAI_GetBooleanParameter
(
 JAI_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 bool& outValue
)
```

Parameters

	Name	Type	Default	Description
↔	ioState	JAI_State&		Object used to maintain state of the function.
→	inDeviceID	Optional<const String&>	NIL	Device identifying number
→	inName	const String&		Parameter name
←	outValue	bool&		Parameter value

JAI_GetEnumParameterAsInteger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets an enum parameter of the JAI device.

Syntax

```
void avl::JAI_GetEnumParameterAsInteger
(
 JAI_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 atl::int64& outValue
)
```

Parameters

	Name	Type	Default	Description
↔	ioState	JAI_State&		Object used to maintain state of the function.
→	inDeviceID	Optional<const String&>	NIL	Device identifying number
→	inName	const String&		Parameter name
←	outValue	int64&		Parameter value

JAI_GetEnumParameterAsString

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets an enum parameter of the JAI device.

Syntax

```
void avl::JAI_GetEnumParameterAsString
(
 JAI_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 atl::String& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	JAI_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inName	const String&		Parameter name
	outValue	String&		Parameter value

JAI_GetFloatParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets a float parameter of the JAI device.

Syntax

```
void avl::JAI_GetFloatParameter
(
 JAI_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 double& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	JAI_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inName	const String&		Parameter name
	outValue	double&		Parameter value

JAI_GetIntegerParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets an integer parameter of the JAI device.

Syntax

```
void avl::JAI_GetIntegerParameter
(
 JAI_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 atl::int64& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	JAI_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inName	const String&		Parameter name
	outValue	int64&		Parameter value

JAI_GetStringParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets a string parameter of the JAI device.

Syntax

```
void avl::JAI_GetStringParameter
(
 JAI_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 atl::String& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	JAI_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inName	const String&		Parameter name
	outValue	String&		Parameter value

JAI_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an image using a JAI device.

Syntax

```
bool avl::JAI_GrabImage
(
 JAI_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inInputQueueSize,
 atl::Optional<avl::EBUSPixelFormat::Type> inPixelFormat,
 atl::Optional<const avl::Box&> inAoi,
 atl::Optional<avl::EBUSAcquisitionMode::Type> inAcquisitionMode,
 atl::Optional<int> inAcquisitionFrameCount,
 atl::Optional<double> inFrameRate,
 atl::Optional<avl::EBUSAutoExposureMode::Type> inExposureAuto,
 atl::Optional<double> inExposureTime,
 atl::Optional<avl::EBUSAutoGainMode::Type> inGainAuto,
 atl::Optional<double> inGain,
 atl::Optional<avl::EBUSTriggerSource::Type> inTriggerSource,
 atl::Optional<avl::EBUSTriggerActivation::Type> inTriggerActivation,
 bool inSkipInvalidFrames,
 avl::Image& outImage,
 avl::EBUSFrameData& outFrameData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	JAI_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Device identifying number
	inInputQueueSize	int	1 - ∞	12	Capacity of output frames queue
	inPixelFormat	Optional<EBUSPixelFormat::Type>		NIL	Image pixel format
	inAoi	Optional<const Box&>		NIL	Area of interest
	inAcquisitionMode	Optional<EBUSAcquisitionMode::Type>		NIL	Acquisition mode
	inAcquisitionFrameCount	Optional<int>	1 - 65535	NIL	Number of frames to acquire in MultiFrame acquisition mode
	inFrameRate	Optional<double>	0.125 - ∞	NIL	Acquisition frame rate
	inExposureAuto	Optional<EBUSAutoExposureMode::Type>		NIL	Automatic exposure mode
	inExposureTime	Optional<double>	1 - ∞	NIL	Exposure time in us
	inGainAuto	Optional<EBUSAutoGainMode::Type>		NIL	Automatic gain mode
	inGain	Optional<double>	1 - ∞	NIL	Gain as an absolute physical value
	inTriggerSource	Optional<EBUSTriggerSource::Type>		NIL	Trigger source
	inTriggerActivation	Optional<EBUSTriggerActivation::Type>		NIL	Trigger activation mode
	inSkipInvalidFrames	bool		True	Skipping invalid images
	outImage	Image&			Captured frame
	outFrameData	EBUSFrameData&			Captured frame data

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install eBUS SDK for JAI software with camera dedicated drivers.

eBUS SDK for JAI can be downloaded from the following website: https://www.jai.com/support-software/jai-software.

Recommended eBUS SDK for JAI version for Aurora Vision Studio usage is **6.3.0**.

Add DLL path to system environment variable may be required.

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, the first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to the computer. DeviceID can be set to:

- **Serial Number** - should be printed on the device housing.
- **Device ID** - unique device identifier.
- **IP Address** - network IP address of device.
- **MAC Address** - MAC address of device.

Camera parameters

To change other and more advanced internal camera parameters use the "eBUS Player for JAI" tool which could be installed with the eBUS SDK for JAI.

See Also

- JAI_GrabImage_WithTimeout – Captures an image using a JAI device with timeout.
- JAI_StartAcquisition – Initializes and starts image acquisition in a camera.

JAI_GrabImage_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an image using a JAI device with timeout.

Syntax

```
bool avl::JAI_GrabImage_WithTimeout
(
 JAI_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inTimeout,
 int inInputQueueSize,
 atl::Optional<avl::EBUSPixelFormat::Type> inPixelFormat,
 atl::Optional<const avl::Box&> inAoi,
 atl::Optional<avl::EBUSAcquisitionMode::Type> inAcquisitionMode,
 atl::Optional<int> inAcquisitionFrameCount,
 atl::Optional<double> inFrameRate,
 atl::Optional<avl::EBUSAutoExposureMode::Type> inExposureAuto,
 atl::Optional<double> inExposureTime,
 atl::Optional<avl::EBUSAutoGainMode::Type> inGainAuto,
 atl::Optional<double> inGain,
 atl::Optional<avl::EBUSTriggerSource::Type> inTriggerSource,
 atl::Optional<avl::EBUSTriggerActivation::Type> inTriggerActivation,
 bool inSkipInvalidFrames,
 atl::Conditional<avl::Image>& outImage,
 atl::Conditional<avl::EBUSFrameData>& outFrameData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	JAI_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Device identifying number
	inTimeout	int	100 - ∞	100	Maximum time to wait for frame in milliseconds
	inInputQueueSize	int	1 - ∞	12	Capacity of output frames queue
	inPixelFormat	Optional<EBUSPixelFormat::Type>		NIL	Image pixel format
	inAoi	Optional<const Box&>		NIL	Area of interest
	inAcquisitionMode	Optional<EBUSAcquisitionMode::Type>		NIL	Acquisition mode
	inAcquisitionFrameCount	Optional<int>	1 - 65535	NIL	Number of frames to acquire in MultiFrame acquisition mode
	inFrameRate	Optional<double>	0.125 - ∞	NIL	Acquisition frame rate
	inExposureAuto	Optional<EBUSAutoExposureMode::Type>		NIL	Automatic exposure mode
	inExposureTime	Optional<double>	1 - ∞	NIL	Exposure time in us
	inGainAuto	Optional<EBUSAutoGainMode::Type>		NIL	Automatic gain mode
	inGain	Optional<double>	1 - ∞	NIL	Gain as an absolute physical value
	inTriggerSource	Optional<EBUSTriggerSource::Type>		NIL	Trigger source
	inTriggerActivation	Optional<EBUSTriggerActivation::Type>		NIL	Trigger activation mode
	inSkipInvalidFrames	bool		True	Skipping invalid images
	outImage	Conditional<Image>&			Captured frame
	outFrameData	Conditional<EBUSFrameData>&			Captured frame data

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install eBUS SDK for JAI software with camera dedicated drivers.

eBUS SDK for JAI can be downloaded from the following website: https://www.jai.com/support-software/jai-software.

Recommended eBUS SDK for JAI version for Aurora Vision Studio usage is **6.3.0**.

Add DLL path to system environment variable may be required.

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, the first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to the computer. DeviceID can be set to:

- **Serial Number** - should be printed on the device housing.
- **Device ID** - unique device identifier.
- **IP Address** - network IP address of device.
- **MAC Address** - MAC address of device.

Camera parameters

To change other and more advanced internal camera parameters use the "eBUS Player for JAI" tool which could be installed with the eBUS SDK for JAI.

See Also

- JAI_GrabImage – Captures an image using a JAI device.
- JAI_StartAcquisition – Initializes and starts image acquisition in a camera.

JAI_SetBooleanParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a boolean parameter of the JAI device.

Syntax

```
void avl::JAI_SetBooleanParameter
(
 JAI_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 bool inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	JAI_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inName	const String&		Parameter name
	inValue	bool		Parameter value

JAI_SetEnumParameterAsInteger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets an enum parameter of the JAI device.

Syntax

```
void avl::JAI_SetEnumParameterAsInteger
(
 JAI_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 atl::int64 inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	JAI_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inName	const String&		Parameter name
	inValue	int64		Parameter value

JAI_SetEnumParameterAsString

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets an enum parameter of the JAI device.

Syntax

```
void avl::JAI_SetEnumParameterAsString
(
 JAI_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 const atl::String& inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	JAI_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inName	const String&		Parameter name
	inValue	const String&		Parameter value

JAI_SetFloatParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a float parameter of the JAI device.

Syntax

```
void avl::JAI_SetFloatParameter
(
 JAI_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 double inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	JAI_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inName	const String&		Parameter name
	inValue	double		Parameter value

JAI_SetIntegerParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets an integer parameter of the JAI device.

Syntax

```
void avl::JAI_SetIntegerParameter
(
 JAI_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 atl::int64 inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	JAI_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inName	const String&		Parameter name
	inValue	int64		Parameter value

JAI_SetStringParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a string parameter of the JAI device.

Syntax

```
void avl::JAI_SetStringParameter
(
 JAI_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 const atl::String& inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	JAI_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inName	const String&		Parameter name
	inValue	const String&		Parameter value

JAI_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Initializes and starts image acquisition in a camera.

Syntax

```
void avl::JAI_StartAcquisition
(
 JAI_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inInputQueueSize,
 atl::Optional<avl::EBUSPixelFormat::Type> inPixelFormat,
 atl::Optional<const avl::Box&> inAoi,
 atl::Optional<avl::EBUSAcquisitionMode::Type> inAcquisitionMode,
 atl::Optional<int> inAcquisitionFrameCount,
 atl::Optional<double> inFrameRate,
 atl::Optional<avl::EBUSAutoExposureMode::Type> inExposureAuto,
 atl::Optional<double> inExposureTime,
 atl::Optional<avl::EBUSAutoGainMode::Type> inGainAuto,
 atl::Optional<double> inGain,
 atl::Optional<avl::EBUSTriggerSource::Type> inTriggerSource,
 atl::Optional<avl::EBUSTriggerActivation::Type> inTriggerActivation
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	JAI_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Device identifying number
	inInputQueueSize	int	1 - ∞	12	Capacity of output frames queue
	inPixelFormat	Optional<EBUSPixelFormat::Type>		NIL	Image pixel format
	inAoi	Optional<const Box&>		NIL	Area of interest
	inAcquisitionMode	Optional<EBUSAcquisitionMode::Type>		NIL	Acquisition mode
	inAcquisitionFrameCount	Optional<int>	1 - 65535	NIL	Number of frames to acquire in MultiFrame acquisition mode
	inFrameRate	Optional<double>	0.125 - ∞	NIL	Acquisition frame rate
	inExposureAuto	Optional<EBUSAutoExposureMode::Type>		NIL	Automatic exposure mode
	inExposureTime	Optional<double>	1 - ∞	NIL	Exposure time in us
	inGainAuto	Optional<EBUSAutoGainMode::Type>		NIL	Automatic gain mode
	inGain	Optional<double>	1 - ∞	NIL	Gain as an absolute physical value
	inTriggerSource	Optional<EBUSTriggerSource::Type>		NIL	Trigger source
	inTriggerActivation	Optional<EBUSTriggerActivation::Type>		NIL	Trigger activation mode

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install eBUS SDK for JAI software with camera dedicated drivers.

eBUS SDK for JAI can be downloaded from the following website: https://www.jai.com/support-software/jai-software.

Recommended eBUS SDK for JAI version for Aurora Vision Studio usage is **6.3.0**.

Add DLL path to system environment variable may be required.

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, the first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to the computer. DeviceID can be set to:

- **Serial Number** - should be printed on the device housing.
- **Device ID** - unique device identifier.
- **IP Address** - network IP address of device.
- **MAC Address** - MAC address of device.

Camera parameters

To change other and more advanced internal camera parameters use the "eBUS Player for JAI" tool which could be installed with the eBUS SDK for JAI.

See Also

- JAI_GrabImage – Captures an image using a JAI device.
- JAI_GrabImage_WithTimeout – Captures an image using a JAI device with timeout.

JAI_StopAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Stops image acquisition in a camera.

Syntax

```
void avl::JAI_StopAcquisition
(
 JAI_State& ioState,
 atl::Optional<const atl::String&> inDeviceID
)
```

Parameters

	Name	Type	Default	Description
	ioState	JAI_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number

167. Kontron GPIO Board

Table of content:

KontronGPIOBoard_Connect

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Connect to Kontron GPIO board.

Syntax

```
void avl::KontronGPIOBoard_Connect
(
 KontronGPIOBoard_State& ioState,
 int inComPort
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	KontronGPIOBoard_State&			Object used to maintain state of the function.
	inComPort	int	1 - ∞	1	

Description

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Kontron GPIO SDK.

Add DLL path to system environment variable is required.

Use of connect filter is required before performing other operations.

See Also

- KontronGPIOBoard_Connect – Connect to Kontron GPIO board.
- KontronGPIOBoard_Disconnect – Disconnect from Kontron GPIO board.

KontronGPIOBoard_Disconnect

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Disconnect from Kontron GPIO board.

Syntax

```
void avl::KontronGPIOBoard_Disconnect
(
 KontronGPIOBoard_State& ioState
)
```

Parameters

	Name	Type	Default	Description
	ioState	KontronGPIOBoard_State&		Object used to maintain state of the function.

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Kontron GPIO SDK.

Add DLL path to system environment variable is required.

Use of connect filter is required before performing other operations.

See Also

- KontronGPIOBoard_Connect – Connect to Kontron GPIO board.
- KontronGPIOBoard_Disconnect – Disconnect from Kontron GPIO board.

KontronGPIOBoard_GetBoardInfo

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets Kontron GPIO board information.

Syntax

```
void avl::KontronGPIOBoard_GetBoardInfo
(
 KontronGPIOBoard_State& ioState,
 atl::String& outBoardNumber,
 bool& outIsActive
)
```

Parameters

	Name	Type	Default	Description
	ioState	KontronGPIOBoard_State&		Object used to maintain state of the function.
	outBoardNumber	String&		
	outIsActive	bool&		True when bootloader is active

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Kontron GPIO SDK.

Add DLL path to system environment variable is required.

Use of connect filter is required before performing other operations.

See Also

- KontronGPIOBoard_Connect – Connect to Kontron GPIO board.
- KontronGPIOBoard_Disconnect – Disconnect from Kontron GPIO board.

KontronGPIOBoard_GetInputState

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets input level.

Syntax

```
void avl::KontronGPIOBoard_GetInputState
(
 KontronGPIOBoard_State& ioState,
 int inPort,
 bool& outState
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	KontronGPIOBoard_State&			Object used to maintain state of the function.
	inPort	int	1 - 8		
	outState	bool&			

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Kontron GPIO SDK.

Add DLL path to system environment variable is required.

Use of connect filter is required before performing other operations.

See Also

- KontronGPIOBoard_Connect – Connect to Kontron GPIO board.
- KontronGPIOBoard_Disconnect – Disconnect from Kontron GPIO board.

KontronGPIOBoard_GetOutputCurrent

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets output current.

Syntax

```
void avl::KontronGPIOBoard_GetOutputCurrent
(
 KontronGPIOBoard_State& ioState,
 int inPort,
 float& outCurrent
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	KontronGPIOBoard_State&			Object used to maintain state of the function.
	inPort	int	1 - 8		
	outCurrent	float&			Current value in mA

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Kontron GPIO SDK.

Add DLL path to system environment variable is required.

Use of connect filter is required before performing other operations.

See Also

- KontronGPIOBoard_Connect – Connect to Kontron GPIO board.
- KontronGPIOBoard_Disconnect – Disconnect from Kontron GPIO board.

KontronGPIOBoard_GetOutputState

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets output level.

Syntax

```
void avl::KontronGPIOBoard_GetOutputState
(
 KontronGPIOBoard_State& ioState,
 int inPort,
 bool& outState
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	KontronGPIOBoard_State&			Object used to maintain state of the function.
	inPort	int	1 - 8		
	outState	bool&			

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Kontron GPIO SDK.

Add DLL path to system environment variable is required.

Use of connect filter is required before performing other operations.

See Also

- KontronGPIOBoard_Connect – Connect to Kontron GPIO board.
- KontronGPIOBoard_Disconnect – Disconnect from Kontron GPIO board.

KontronGPIOBoard_GetVersionInfo

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets firmware version from Kontron GPIO board.

Syntax

```
void avl::KontronGPIOBoard_GetVersionInfo
(
 KontronGPIOBoard_State& ioState,
 atl::String& outFirmwareVersion,
 atl::String& outBootloaderVersion
)
```

Parameters

	Name	Type	Default	Description
	ioState	KontronGPIOBoard_State&		Object used to maintain state of the function.
	outFirmwareVersion	String&		
	outBootloaderVersion	String&		

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Kontron GPIO SDK.

Add DLL path to system environment variable is required.

Use of connect filter is required before performing other operations.

See Also

- KontronGPIOBoard_Connect – Connect to Kontron GPIO board.
- KontronGPIOBoard_Disconnect – Disconnect from Kontron GPIO board.

KontronGPIOBoard_SetOutputState

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets input level.

Syntax

```
void avl::KontronGPIOBoard_SetOutputState
(
 KontronGPIOBoard_State& ioState,
 int inPort,
 bool inState
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	KontronGPIOBoard_State&			Object used to maintain state of the function.
	inPort	int	1 - 8		
	inState	bool			

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install Kontron GPIO SDK.

Add DLL path to system environment variable is required.

Use of connect filter is required before performing other operations.

See Also

- KontronGPIOBoard_Connect – Connect to Kontron GPIO board.
- KontronGPIOBoard_Disconnect – Disconnect from Kontron GPIO board.

168. LEX

Table of content:

Lex_GetDigitalInput

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets a digital input state of LEX computers.

Syntax

```
void avl::Lex_GetDigitalInput
(
 Lex_GetDigitalInputState& ioState,
 const int inInput,
 bool& outState
)
```

Parameters

	Name	Type	Default	Description
	ioState	Lex_GetDigitalInputState&		Object used to maintain state of the function.
	inInput	const int		Number of pin to read
	outState	bool&		Read state of chosen pin

Description

This filter can read state of digital inputs in LEX computers. Maximal number of input vary between different models.

Remarks

Running LEX related filters

Mechanism of Digital IO control of LEX computers demands to be run by Administrator account. LEX related filters will not work if Aurora Vision Studio or any program based on Aurora Vision Library is not run by Administrator. One can achieve that by right clicking on program to be ran and selecting "Run as administrator".

This filter is available only on 64-bit platform.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

List of possible exceptions:

Error type	Description
RuntimeError	Lex filters are available only on 64-bit platform.
SystemError	Could not initialize F75111 DIO controller. LEX needs Administrator privileges to run.

See Also

- Lex_SetDigitalOutput – Sets a digital output state of LEX computers.

Lex_SetDigitalOutput

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a digital output state of LEX computers.

Syntax

```
void avl::Lex_SetDigitalOutput
(
 Lex_SetDigitalOutputState& ioState,
 const int inOutput,
 bool inState
)
```

Parameters

	Name	Type	Default	Description
	ioState	Lex_SetDigitalOutputState&		Object used to maintain state of the function.
	inOutput	const int		Number of output to change
	inState	bool		State of pin output to set

Description

This filter can change state of digital outputs in LEX computers. Maximal number of output vary between different models.

Remarks

Running LEX related filters

Mechanism of Digital IO control of LEX computers demands to be run by Administrator account. LEX related filters will not work if Aurora Vision Studio or any program based on Aurora Vision Library is not run by Administrator. One can achieve that by right clicking on program to be ran and selecting "Run as administrator".

This filter is available only on 64-bit platform.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

List of possible exceptions:

Error type	Description
RuntimeError	Lex filters are available only on 64-bit platform.
SystemError	Could not initialize F75111 DIO controller. LEX needs Administrator privileges to run.

See Also

- Lex_GetDigitalInput – Gets a digital input state of LEX computers.

169. Lumenera

Table of content:

Lumenera_GetParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets a parameter.

Syntax

```
void avl::Lumenera_GetParameter
(
 Lumenera_State& ioState,
 const atl::Optional<int>& inDeviceID,
 const avl::LumeneraProperties::Type inType,
 float& outParameter
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Lumenera_State&			Object used to maintain state of the function.
	inDeviceID	const Optional<int>&	1 - 10000	NIL	Device identifying number
	inType	const LumeneraProperties::Type			
	outParameter	float&			

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Lumenera SDK software.

Lumenera SDK can be downloaded from the following website: https://www.lumenera.com/sdk.html

Recommended Lumenera SDK version for Aurora Vision Studio usage is 6.3.

Remember to install camera drivers from Lumenera SDK, in directory *%LUMENERA_SDK%\Redist*. The driver for USB is Redist LU, LM, LW, LC, LT, USB. The best is to use *All Driver Install*
Follow these instructions for quick install:

- open cmd.exe
- copy to opened window (enters driver directory):

```
cd "%LUMENERA_SDK%\Redist\Redist LU, LM, LW, LC, LT, USB\All Driver Install"
```

- copy to opened window (installs the driver):
  - For 64 bit Windows

```
luihlp64.exe -inf -dll -ax
```

  - For 32 bit Windows

```
luihlp.exe -inf -dll -ax
```

Camera identification

When there is only one Lumenera camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Lumenera_SetParameter – Sets a parameter.

Lumenera_GpioConfigure

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Configures pin direction, true for output.

Syntax

```
void avl::Lumenera_GpioConfigure
(
 Lumenera_State& ioState,
 const atl::Optional<int>& inDeviceID,
 const bool inGpio0,
 const bool inGpio1,
 const bool inGpio2,
 const bool inGpio3
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Lumenera_State&			Object used to maintain state of the function.
	inDeviceID	const Optional<int>&	1 - 10000	NIL	Device identifying number
	inGpio0	const bool			
	inGpio1	const bool			
	inGpio2	const bool			
	inGpio3	const bool			

Remarks

GPIO pin identification

Some cameras enumerate pins starting from 1, so outGpi0 might be GPI1 and inGpo0 might be GPO1. Similarly for other pins.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Lumenera SDK software.

Lumenera SDK can be downloaded from the following website: https://www.lumenera.com/sdk.html

Recommended Lumenera SDK version for Aurora Vision Studio usage is 6.3.

Remember to install camera drivers from Lumenera SDK, in directory *%LUMENERA_SDK%\Redist*. The driver for USB is Redist LU, LM, LW, LC, LT, USB. The best is to use *All Driver Install*
Follow these instructions for quick install:

- open cmd.exe
- copy to opened window (enters driver directory):
  ```
  cd "%LUMENERA_SDK%\Redist\Redist LU, LM, LW, LC, LT, USB\All Driver Install"
  ```
- copy to opened window (installs the driver):
  - For 64 bit Windows
    ```
    luihlp64.exe -inf -dll -ax
    ```
  - For 32 bit Windows
    ```
    luihlp.exe -inf -dll -ax
    ```

Camera identification

When there is only one Lumenera camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Lumenera_GpioRead – Reads GPI pins.
- Lumenera_GpioWrite – Writes GPO pins.
- Lumenera_GpioSelect – Selects GPO functionality, true for manual toggling using GpioWrite filter (see LucamGpoSelect in Lumenera API for default mode).

Lumenera_GpioRead

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads GPI pins.

Syntax

```
void avl::Lumenera_GpioRead
(
 Lumenera_State& ioState,
 const atl::Optional<int>& inDeviceID,
 bool& outGpi0,
 bool& outGpi1,
 bool& outGpi2,
 bool& outGpi3
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Lumenera_State&			Object used to maintain state of the function.
	inDeviceID	const Optional<int>&	1 - 10000	NIL	Device identifying number
	outGpi0	bool&			
	outGpi1	bool&			
	outGpi2	bool&			
	outGpi3	bool&			

Remarks

GPIO pin identification

Some cameras enumerate pins starting from 1, so outGpi0 might be GPI1 and inGpo0 might be GPO1. Similarly for other pins.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Lumenera SDK software.

Lumenera SDK can be downloaded from the following website: https://www.lumenera.com/sdk.html

Recommended Lumenera SDK version for Aurora Vision Studio usage is 6.3.

Remember to install camera drivers from Lumenera SDK, in directory *%LUMENERA_SDK%\Redist*. The driver for USB is Redist LU, LM, LW, LC, LT, USB. The best is to use *All Driver Install*
Follow these instructions for quick install:

- open cmd.exe
- copy to opened window (enters driver directory):
  ```
  cd "%LUMENERA_SDK%\Redist\Redist LU, LM, LW, LC, LT, USB\All Driver Install"
  ```
- copy to opened window (installs the driver):
  - For 64 bit Windows
    ```
    luihlp64.exe -inf -dll -ax
    ```
  - For 32 bit Windows
    ```
    luihlp.exe -inf -dll -ax
    ```

Camera identification

When there is only one Lumenera camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Lumenera_GpioWrite – Writes GPO pins.
- Lumenera_GpioConfigure – Configures pin direction, true for output.
- Lumenera_GpioSelect – Selects GPO functionality, true for manual toggling using GpioWrite filter (see LucamGpoSelect in Lumenera API for default mode).

Lumenera_GpioSelect

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Selects GPO functionality, true for manual toggling using GpioWrite filter (see LucamGpoSelect in Lumenera API for default mode).

Syntax

```
void avl::Lumenera_GpioSelect
(
 Lumenera_State& ioState,
 const atl::Optional<int>& inDeviceID,
 const bool inGpio0,
 const bool inGpio1,
 const bool inGpio2,
 const bool inGpio3
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Lumenera_State&			Object used to maintain state of the function.
	inDeviceID	const Optional<int>&	1 - 10000	NIL	Device identifying number
	inGpio0	const bool			
	inGpio1	const bool			
	inGpio2	const bool			
	inGpio3	const bool			

Remarks

GPIO pin identification

Some cameras enumerate pins starting from 1, so outGpi0 might be GPI1 and inGpo0 might be GPO1. Similarly for other pins.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Lumenera SDK software.

Lumenera SDK can be downloaded from the following website: https://www.lumenera.com/sdk.html

Recommended Lumenera SDK version for Aurora Vision Studio usage is 6.3.

Remember to install camera drivers from Lumenera SDK, in directory *%LUMENERA_SDK%\Redist*. The driver for USB is Redist LU, LM, LW, LC, LT, USB. The best is to use *All Driver Install*
Follow these instructions for quick install:

- open cmd.exe
- copy to opened window (enters driver directory):
  ```
  cd "%LUMENERA_SDK%\Redist\Redist LU, LM, LW, LC, LT, USB\All Driver Install"
  ```
- copy to opened window (installs the driver):
  - For 64 bit Windows
    ```
    luihlp64.exe -inf -dll -ax
    ```
  - For 32 bit Windows
    ```
    luihlp.exe -inf -dll -ax
    ```

Camera identification

When there is only one Lumenera camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Lumenera_GpioRead – Reads GPI pins.
- Lumenera_GpioWrite – Writes GPO pins.
- Lumenera_GpioConfigure – Configures pin direction, true for output.

Lumenera_GpioWrite

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Writes GPO pins.

Syntax

```
void avl::Lumenera_GpioWrite
(
 Lumenera_State& ioState,
 const atl::Optional<int>& inDeviceID,
 const bool inGpo0,
 const bool inGpo1,
 const bool inGpo2,
 const bool inGpo3
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Lumenera_State&			Object used to maintain state of the function.
	inDeviceID	const Optional<int>&	1 - 10000	NIL	Device identifying number
	inGpo0	const bool			
	inGpo1	const bool			
	inGpo2	const bool			
	inGpo3	const bool			

Remarks

GPIO pin identification

Some cameras enumerate pins starting from 1, so outGpi0 might be GPI1 and inGpo0 might be GPO1. Similarly for other pins.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Lumenera SDK software.

Lumenera SDK can be downloaded from the following website: https://www.lumenera.com/sdk.html

Recommended Lumenera SDK version for Aurora Vision Studio usage is 6.3.

Remember to install camera drivers from Lumenera SDK, in directory *%LUMENERA_SDK%\Redist*. The driver for USB is Redist LU, LM, LW, LC, LT, USB. The best is to use *All Driver Install*
Follow these instructions for quick install:

- open cmd.exe
- copy to opened window (enters driver directory):
  ```
  cd "%LUMENERA_SDK%\Redist\Redist LU, LM, LW, LC, LT, USB\All Driver Install"
  ```
- copy to opened window (installs the driver):
  - For 64 bit Windows
    ```
    luihlp64.exe -inf -dll -ax
    ```
  - For 32 bit Windows
    ```
    luihlp.exe -inf -dll -ax
    ```

Camera identification

When there is only one Lumenera camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Lumenera_GpioRead – Reads GPI pins.
- Lumenera_GpioConfigure – Configures pin direction, true for output.
- Lumenera_GpioSelect – Selects GPO functionality, true for manual toggling using GpioWrite filter (see LucamGpoSelect in Lumenera API for default mode).

Lumenera_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a frame using Lumenera.

Syntax

```
bool avl::Lumenera_GrabImage
(
 Lumenera_State& ioState,
 const atl::Optional<int> inDeviceID,
 const avl::LumeneraImageFormatParams& inImageFormat,
 const avl::LumeneraAcquisitionControlParams& inAcquisitionControl,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Lumenera_State&			Object used to maintain state of the function.
	inDeviceID	const Optional<int>	1 - 10000	NIL	Device identifying number
	inImageFormat	const LumeneraImageFormatParams&			Image format parameters
	inAcquisitionControl	const LumeneraAcquisitionControlParams&			Acquisition control parameters
	outImage	Image&			Captured frame

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Lumenera SDK software.

Lumenera SDK can be downloaded from the following website: https://www.lumenera.com/sdk.html

Recommended Lumenera SDK version for Aurora Vision Studio usage is 6.3.

Remember to install camera drivers from Lumenera SDK, in directory *%LUMENERA_SDK%\Redist*. The driver for USB is Redist LU, LM, LW, LC, LT, USB. The best is to use *All Driver Install*
Follow these instructions for quick install:

- open cmd.exe
- copy to opened window (enters driver directory):
  ```
  cd "%LUMENERA_SDK%\Redist\Redist LU, LM, LW, LC, LT, USB\All Driver Install"
  ```
- copy to opened window (installs the driver):
  - For 64 bit Windows
    ```
    luihlp64.exe -inf -dll -ax
    ```
  - For 32 bit Windows
    ```
    luihlp.exe -inf -dll -ax
    ```

Camera identification

When there is only one Lumenera camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Lumenera_GrabImage_WithTimeout – Captures a frame using Lumenera.

Lumenera_GrabImage_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a frame using Lumenera.

Syntax

```
bool avl::Lumenera_GrabImage_WithTimeout
(
 Lumenera_State& ioState,
 const atl::Optional<int> inDeviceID,
 int inTimeout,
 const avl::LumeneraImageFormatParams& inImageFormat,
 const avl::LumeneraAcquisitionControlParams& inAcquisitionControl,
 atl::Conditional<avl::Image>& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Lumenera_State&			Object used to maintain state of the function.
	inDeviceID	const Optional<int>	1 - 10000	NIL	Device identifying number
	inTimeout	int	100 - 3600000	5000	Maximum time to wait for frame in milliseconds
	inImageFormat	const LumeneraImageFormatParams&			Image format parameters
	inAcquisitionControl	const LumeneraAcquisitionControlParams&			Acquisition control parameters
	outImage	Conditional<Image>&			Captured frame

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Lumenera SDK software.

Lumenera SDK can be downloaded from the following website: https://www.lumenera.com/sdk.html

Recommended Lumenera SDK version for Aurora Vision Studio usage is 6.3.

Remember to install camera drivers from Lumenera SDK, in directory *%LUMENERA_SDK%\Redist*. The driver for USB is Redist LU, LM, LW, LC, LT, USB. The best is to use *All Driver Install*
Follow these instructions for quick install:

- open cmd.exe
- copy to opened window (enters driver directory):

```
cd "%LUMENERA_SDK%\Redist\Redist LU, LM, LW, LC, LT, USB\All Driver Install"
```

- copy to opened window (installs the driver):
  - For 64 bit Windows

```
luihlp64.exe -inf -dll -ax
```

  - For 32 bit Windows

```
luihlp.exe -inf -dll -ax
```

Camera identification

When there is only one Lumenera camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Lumenera_GrabImage – Captures a frame using Lumenera.

Lumenera_SetParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a parameter.

Syntax

```
void avl::Lumenera_SetParameter
(
 Lumenera_State& ioState,
 const atl::Optional<int>& inDeviceID,
 const avl::LumeneraProperties::Type inType,
 const float inParameter
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Lumenera_State&			Object used to maintain state of the function.
	inDeviceID	const Optional<int>&	1 - 10000	NIL	Device identifying number
	inType	const LumeneraProperties::Type			
	inParameter	const float			

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Lumenera SDK software.

Lumenera SDK can be downloaded from the following website: https://www.lumenera.com/sdk.html

Recommended Lumenera SDK version for Aurora Vision Studio usage is 6.3.

Remember to install camera drivers from Lumenera SDK, in directory *%LUMENERA_SDK%\Redist*. The driver for USB is Redist LU, LM, LW, LC, LT, USB. The best is to use *All Driver Install*
Follow these instructions for quick install:

- open cmd.exe
- copy to opened window (enters driver directory):

```
cd "%LUMENERA_SDK%\Redist\Redist LU, LM, LW, LC, LT, USB\All Driver Install"
```

- copy to opened window (installs the driver):
  - For 64 bit Windows

```
luihlp64.exe -inf -dll -ax
```

  - For 32 bit Windows

```
luihlp.exe -inf -dll -ax
```

Camera identification

When there is only one Lumenera camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Lumenera_GetParameter – Gets a parameter.

Lumenera_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Starts capturing images from an Lumenera camera.

Applications: Typically used for establishing camera connectivity before the first trigger event. Especially important for multiple-camera systems.

Syntax

```
void avl::Lumenera_StartAcquisition
(
 Lumenera_State& ioState,
 const atl::Optional<int> inDeviceID,
 const avl::LumeneraImageFormatParams& inImageFormat,
 const avl::LumeneraAcquisitionControlParams& inAcquisitionControl
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Lumenera_State&			Object used to maintain state of the function.
	inDeviceID	const Optional<int>	1 - 10000	NIL	Device identifying number
	inImageFormat	const LumeneraImageFormatParams&			Image format parameters
	inAcquisitionControl	const LumeneraAcquisitionControlParams&			Acquisition control parameters

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Lumenera SDK software.

Lumenera SDK can be downloaded from the following website: https://www.lumenera.com/sdk.html

Recommended Lumenera SDK version for Aurora Vision Studio usage is 6.3.

Remember to install camera drivers from Lumenera SDK, in directory *%LUMENERA_SDK%\Redist*. The driver for USB is Redist LU, LM, LW, LC, LT, USB. The best is to use *All Driver Install*
Follow these instructions for quick install:

- open cmd.exe
- copy to opened window (enters driver directory):
  ```
  cd "%LUMENERA_SDK%\Redist\Redist LU, LM, LW, LC, LT, USB\All Driver Install"
  ```
- copy to opened window (installs the driver):
  - For 64 bit Windows
    ```
    luihlp64.exe -inf -dll -ax
    ```
  - For 32 bit Windows
    ```
    luihlp.exe -inf -dll -ax
    ```

Camera identification

When there is only one Lumenera camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Lumenera_StopAcquisition – Stops image acquisition in a Lumenera device.
- Lumenera_GrabImage – Captures a frame using Lumenera.
- Lumenera_GrabImage_WithTimeout – Captures a frame using Lumenera.

Lumenera_StopAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Stops image acquisition in a Lumenera device.

Syntax

```
void avl::Lumenera_StopAcquisition
(
 Lumenera_State& ioState,
 const atl::Optional<int>& inDeviceID
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Lumenera_State&			Object used to maintain state of the function.
	inDeviceID	const Optional<int>&	1 - 10000	NIL	Device identifying number

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Lumenera SDK software.

Lumenera SDK can be downloaded from the following website: https://www.lumenera.com/sdk.html

Recommended Lumenera SDK version for Aurora Vision Studio usage is 6.3.

Remember to install camera drivers from Lumenera SDK, in directory *%LUMENERA_SDK%\Redist*. The driver for USB is Redist LU, LM, LW, LC, LT, USB. The best is to use *All Driver Install*
Follow these instructions for quick install:

- open cmd.exe
- copy to opened window (enters driver directory):
  ```
  cd "%LUMENERA_SDK%\Redist\Redist LU, LM, LW, LC, LT, USB\All Driver Install"
  ```
- copy to opened window (installs the driver):
  - For 64 bit Windows
    ```
    luihlp64.exe -inf -dll -ax
    ```
  - For 32 bit Windows
    ```
    luihlp.exe -inf -dll -ax
    ```

Camera identification

When there is only one Lumenera camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Lumenera_StartAcquisition – Starts capturing images from an Lumenera camera.
- Lumenera_GrabImage – Captures a frame using Lumenera.
- Lumenera_GrabImage_WithTimeout – Captures a frame using Lumenera.

170. Matrox

Table of content:

- Matrox_Profinet_Write_uint16
- Matrox_RegisterEdge
- Matrox_RegisterImpulse
- Matrox_RegisterPulse
- Matrox_ResetEncoderCounter
- Matrox_SetOutputLevel
- Matrox_SetOutputSource

Matrox_ConfigureEncoder

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Configures the encoder.

Syntax

```
void avl::Matrox_ConfigureEncoder
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 int inEncoder,
 int inBit0Source,
 int inBit1Source,
 avl::MatroxEncoderOutputMode::Type inOutputMode,
 int inPositionTrigger,
 avl::MatroxEncoderResetSource::Type inResetSource,
 bool inResetCounter,
 bool inEnable
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inEncoder	int	1 - 2		Encoder.
	inBit0Source	int	1 - 8	1	Auxiliary input signal on which to receive bit 0 of the 2-bit Gray code.
	inBit1Source	int	1 - 8	2	Auxiliary input signal on which to receive bit 1 of the 2-bit Gray code.
	inOutputMode	MatroxEncoderOutputMode::Type			Circumstances in which the rotary decoder should output a pulse.
	inPositionTrigger	int			Rotary decoder's counter value upon which a trigger is generated.
	inResetSource	MatroxEncoderResetSource::Type			Signal source to use to reset the rotary decoder's counter to 0.
	inResetCounter	bool			Reset the encoder counter value to 0 at the end of configuration.
	inEnable	bool		True	Enable or disable the encoder.

Remarks

I/O device driver software

This filter is intended to cooperate with a Matrox device using its vendor SDK. To be able to connect to the device, it is required to install Matrox MIL X SDK. Matrox industrial PCs usually come with a preinstalled runtime license.

Add DLL path to system environment variable may be required.

Recommended Matrox MIL X SDK version for Aurora Vision Studio usage is **V22H1 (10.50.924)**.

Device identification

inDeviceID field can be used to specify the number (rank) of the target board of the specified system type. DeviceID can be set to:

- **DEFAULT** - Specifies the default board. The default board is set in the MILConfig utility.
- **DEVn** - Specifies the device number (rank) of the board (where 0 <= n <=15).

This filter allows to configure one of the built-in quadrature decoders.

See Also

- Matrox_EnableEncoder – Enables or disables the encoder.

Matrox_ConfigureIOCommandList

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Configures an I/O command list.

Syntax

```
void avl::Matrox_ConfigureIOCommandList
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 int inCommandList,
 avl::MatroxIOCommandListCounterSource::Type inCounterSource
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inCommandList	int	1 - 2		I/O command list.
	inCounterSource	MatroxIOCommandListCounterSource::Type			Counter source for the command list.

Remarks

I/O device driver software

This filter is intended to cooperate with a Matrox device using its vendor SDK. To be able to connect to the device, it is required to install Matrox MIL X SDK. Matrox industrial PCs usually come with a preinstalled runtime license.

Add DLL path to system environment variable may be required.

Recommended Matrox MIL X SDK version for Aurora Vision Studio usage is **V22H1 (10.50.924)**.

Device identification

inDeviceID field can be used to specify the number (rank) of the target board of the specified system type. DeviceID can be set to:

- **DEFAULT** - Specifies the default board. The default board is set in the MILConfig utility.
- **DEVn** - Specifies the device number (rank) of the board (where 0 <= n <=15).

This filter allows to configure one of the I/O command lists.

Matrox_ConfigureTimer

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Configures the timer.

Syntax

```
void avl::Matrox_ConfigureTimer
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 int inTimer,
 avl::MatroxTimerTriggerSource::Type inTriggerSource,
 atl::int64 inDelay,
 atl::int64 inDuration,
 bool inInvert,
 avl::MatroxTimerTriggerActivation::Type inActivation,
 bool inEnable
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inTimer	int	1 - 16		Timer.
	inTriggerSource	MatroxTimerTriggerSource::Type		Continuous	Trigger source for the timer.
	inDelay	int64	0 - ∞	1000000L	Delay between the timer trigger and the active portion of the timer output signal (in ns).
	inDuration	int64	0 - ∞	1000000L	Duration for the active portion of the timer output signal (in ns).
	inInvert	bool		False	Whether to invert the output signal or not.
	inActivation	MatroxTimerTriggerActivation::Type			Signal variation upon which to generate a timer trigger.
	inEnable	bool		True	Enable or disable the timer.

Remarks

I/O device driver software

This filter is intended to cooperate with a Matrox device using its vendor SDK. To be able to connect to the device, it is required to install Matrox MIL X SDK. Matrox industrial PCs usually come with a preinstalled runtime license.

Add DLL path to system environment variable may be required.

Recommended Matrox MIL X SDK version for Aurora Vision Studio usage is **V22H1 (10.50.924)**.

Device identification

inDeviceID field can be used to specify the number (rank) of the target board of the specified system type. DeviceID can be set to:

- **DEFAULT** - Specifies the default board. The default board is set in the MILConfig utility.
- **DEVn** - Specifies the device number (rank) of the board (where 0 <= n <=15).

This filter allows to configure one of the built-in timers.

See Also

- Matrox_EnableTimer – Enables or disables the timer.

Matrox_EnableEncoder

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Enables or disables the encoder.

Syntax

```
void avl::Matrox_EnableEncoder
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 int inEncoder,
 bool inEnable
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inEncoder	int	1 - 2		Encoder.
	inEnable	bool		True	Enable or disable the timer.

Remarks

I/O device driver software

This filter is intended to cooperate with a Matrox device using its vendor SDK. To be able to connect to the device, it is required to install Matrox MIL X SDK. Matrox industrial PCs usually come with a preinstalled runtime license.

Add DLL path to system environment variable may be required.

Recommended Matrox MIL X SDK version for Aurora Vision Studio usage is **V22H1 (10.50.924)**.

Device identification

inDeviceID field can be used to specify the number (rank) of the target board of the specified system type. DeviceID can be set to:

- **DEFAULT** - Specifies the default board. The default board is set in the MILConfig utility.
- **DEVn** - Specifies the device number (rank) of the board (where 0 <= n <=15).

This filter allows to enable or disable the operation of one of the built-in quadrature decoders.

See Also

- Matrox_ConfigureEncoder – Configures the encoder.

Matrox_EnableTimer

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Enables or disables the timer.

Syntax

```
void avl::Matrox_EnableTimer
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 int inTimer,
 bool inEnable
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inTimer	int	1 - 16		Timer.
	inEnable	bool		True	Enable or disable the timer.

Remarks

I/O device driver software

This filter is intended to cooperate with a Matrox device using its vendor SDK. To be able to connect to the device, it is required to install Matrox MIL X SDK. Matrox industrial PCs usually come with a preinstalled runtime license.

Add DLL path to system environment variable may be required.

Recommended Matrox MIL X SDK version for Aurora Vision Studio usage is **V22H1 (10.50.924)**.

Device identification

inDeviceID field can be used to specify the number (rank) of the target board of the specified system type. DeviceID can be set to:

- **DEFAULT** - Specifies the default board. The default board is set in the MILConfig utility.
- **DEVn** - Specifies the device number (rank) of the board (where 0 <= n <=15).

This filter allows to enable or disable the operation of one of the built-in timers.

See Also

- Matrox_ConfigureTimer – Configures the timer.

Matrox_EtherNetIP_Read_ByteBuffer

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads the assembly data as a ByteBuffer.

Syntax

```
void avl::Matrox_EtherNetIP_Read_ByteBuffer
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 avl::MatroxEtherNetIPAssembly::Type inAssembly,
 int inOffset,
 int inCount,
 avl::ByteBuffer& outValues
)
```

Parameters

	Name	Type	Range	Default	Description
↻	ioState	Matrox_State&			Object used to maintain state of the function.
→	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
→	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
→	inAssembly	MatroxEtherNetIPAssembly::Type		Consumer	Assembly in local memory from which to read the data.
→	inOffset	int	0 - ∞		Offset from which to start reading the data in the specified assembly.
→	inCount	int	1 - ∞		Amount of the data to read.
←	outValues	ByteBuffer&			Received values.

Matrox_EtherNetIP_Read_float32

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads the assembly data as 32-bit floating point numbers.

Syntax

```
void avl::Matrox_EtherNetIP_Read_float32
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 avl::MatroxEtherNetIPAssembly::Type inAssembly,
 int inOffset,
 int inCount,
 bool inUseLittleEndian,
 atl::Array<float>& outValues
)
```

Parameters

	Name	Type	Range	Default	Description
↻	ioState	Matrox_State&			Object used to maintain state of the function.
→	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
→	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
→	inAssembly	MatroxEtherNetIPAssembly::Type		Consumer	Assembly in local memory from which to read the data.
→	inOffset	int	0 - ∞		Offset from which to start reading the data in the specified assembly.
→	inCount	int	1 - ∞		Amount of the data to read.
→	inUseLittleEndian	bool			Use little endian order for data in the memory.
←	outValues	Array<float>&			Received values.

Matrox_EtherNetIP_Read_sint16

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads the assembly data as signed 16-bit integers.

Syntax

```
void avl::Matrox_EtherNetIP_Read_sint16
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 avl::MatroxEtherNetIPAssembly::Type inAssembly,
 int inOffset,
 int inCount,
 bool inUseLittleEndian,
 atl::Array<int>& outValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inAssembly	MatroxEtherNetIPAssembly::Type		Consumer	Assembly in local memory from which to read the data.
	inOffset	int	0 - ∞		Offset from which to start reading the data in the specified assembly.
	inCount	int	1 - ∞		Amount of the data to read.
	inUseLittleEndian	bool			Use little endian order for data in the memory.
	outValues	Array<int>&			Received values.

Matrox_EtherNetIP_Read_sint32

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads the assembly data as signed 32-bit integers.

Syntax

```
void avl::Matrox_EtherNetIP_Read_sint32
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 avl::MatroxEtherNetIPAssembly::Type inAssembly,
 int inOffset,
 int inCount,
 bool inUseLittleEndian,
 atl::Array<int>& outValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inAssembly	MatroxEtherNetIPAssembly::Type		Consumer	Assembly in local memory from which to read the data.
	inOffset	int	0 - ∞		Offset from which to start reading the data in the specified assembly.
	inCount	int	1 - ∞		Amount of the data to read.
	inUseLittleEndian	bool			Use little endian order for data in the memory.
	outValues	Array<int>&			Received values.

Matrox_EtherNetIP_Read_uint16

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads the assembly data as unsigned 16-bit integers.

Syntax

```
void avl::Matrox_EtherNetIP_Read_uint16
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 avl::MatroxEtherNetIPAssembly::Type inAssembly,
 int inOffset,
 int inCount,
 bool inUseLittleEndian,
 atl::Array<int>& outValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inAssembly	MatroxEtherNetIPAssembly::Type		Consumer	Assembly in local memory from which to read the data.
	inOffset	int	0 - ∞		Offset from which to start reading the data in the specified assembly.
	inCount	int	1 - ∞		Amount of the data to read.
	inUseLittleEndian	bool			Use little endian order for data in the memory.
	outValues	Array<int>&			Received values.

Matrox_EtherNetIP_Write_ByteBuffer

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Writes the assembly data as a ByteBuffer.

Syntax

```
void avl::Matrox_EtherNetIP_Write_ByteBuffer
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 avl::MatroxEtherNetIPAssembly::Type inAssembly,
 int inOffset,
 const avl::ByteBuffer& inValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inAssembly	MatroxEtherNetIPAssembly::Type		Consumer	Assembly in local memory to which to write the data.
	inOffset	int	0 - ∞		Offset to which to start writing the data in the specified assembly.
	inValues	const ByteBuffer&			Values to write.

Matrox_EtherNetIP_Write_float32

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Writes the assembly data as 32-bit floating point numbers.

Syntax

```
void avl::Matrox_EtherNetIP_Write_float32
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 avl::MatroxEtherNetIPAssembly::Type inAssembly,
 int inOffset,
 bool inUseLittleEndian,
 const atl::Array<float>& inValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inAssembly	MatroxEtherNetIPAssembly::Type		Consumer	Assembly in local memory to which to write the data.
	inOffset	int	0 - ∞		Offset to which to start writing the data in the specified assembly.
	inUseLittleEndian	bool			Use little endian order for data in the memory.
	inValues	const Array<float>&			Values to write.

Matrox_EtherNetIP_Write_sint16

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Writes the assembly data as signed 16-bit integers.

Syntax

```
void avl::Matrox_EtherNetIP_Write_sint16
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 avl::MatroxEtherNetIPAssembly::Type inAssembly,
 int inOffset,
 bool inUseLittleEndian,
 const atl::Array<int>& inValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inAssembly	MatroxEtherNetIPAssembly::Type		Consumer	Assembly in local memory to which to write the data.
	inOffset	int	0 - ∞		Offset to which to start writing the data in the specified assembly.
	inUseLittleEndian	bool			Use little endian order for data in the memory.
	inValues	const Array<int>&			Values to write.

Matrox_EtherNetIP_Write_sint32

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Writes the assembly data as signed 32-bit integers.

Syntax

```
void avl::Matrox_EtherNetIP_Write_sint32
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 avl::MatroxEtherNetIPAssembly::Type inAssembly,
 int inOffset,
 bool inUseLittleEndian,
 const atl::Array<int>& inValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inAssembly	MatroxEtherNetIPAssembly::Type		Consumer	Assembly in local memory to which to write the data.
	inOffset	int	0 - ∞		Offset to which to start writing the data in the specified assembly.
	inUseLittleEndian	bool			Use little endian order for data in the memory.
	inValues	const Array<int>&			Values to write.

Matrox_EtherNetIP_Write_uint16

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Writes the assembly data as unsigned 16-bit integers.

Syntax

```
void avl::Matrox_EtherNetIP_Write_uint16
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 avl::MatroxEtherNetIPAssembly::Type inAssembly,
 int inOffset,
 bool inUseLittleEndian,
 const atl::Array<int>& inValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inAssembly	MatroxEtherNetIPAssembly::Type		Consumer	Assembly in local memory to which to write the data.
	inOffset	int	0 - ∞		Offset to which to start writing the data in the specified assembly.
	inUseLittleEndian	bool			Use little endian order for data in the memory.
	inValues	const Array<int>&			Values to write.

Matrox_GenerateSoftwareTrigger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Generates a software trigger for the timer.

Syntax

```
void avl::Matrox_GenerateSoftwareTrigger
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 int inTimer
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inTimer	int	1 - 16		Timer.

Remarks

I/O device driver software

This filter is intended to cooperate with a Matrox device using its vendor SDK. To be able to connect to the device, it is required to install Matrox MIL X SDK. Matrox industrial PCs usually come with a preinstalled runtime license.

Add DLL path to system environment variable may be required.

Recommended Matrox MIL X SDK version for Aurora Vision Studio usage is **V22H1 (10.50.924)**.

Device identification

inDeviceID field can be used to specify the number (rank) of the target board of the specified system type. DeviceID can be set to:

- **DEFAULT** - Specifies the default board. The default board is set in the MILConfig utility.
- **DEVn** - Specifies the device number (rank) of the board (where 0 <= n <=15).

This filter allows to generate a software trigger for one of the timers. The specified timer must be configured to use the **Software** trigger source using the **Matrox_ConfigureTimer** filter.

See Also

- Matrox_ConfigureTimer – Configures the timer.

Matrox_GetAllInputLevels

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets the state of all the inputs on the device.

Syntax

```
void avl::Matrox_GetAllInputLevels
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 atl::Array<bool>& outLevels
)
```

Parameters

	Name	Type	Default	Description
	ioState	Matrox_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>	NIL	Device identification number.
	outLevels	Array<bool>&		Logic levels of the I/Os.

Remarks

I/O device driver software

This filter is intended to cooperate with a Matrox device using its vendor SDK. To be able to connect to the device, it is required to install Matrox MIL X SDK. Matrox industrial PCs usually come with a preinstalled runtime license.

Add DLL path to system environment variable may be required.

Recommended Matrox MIL X SDK version for Aurora Vision Studio usage is **V22H1 (10.50.924)**.

Device identification

inDeviceID field can be used to specify the number (rank) of the target board of the specified system type. DeviceID can be set to:

- **DEFAULT** - Specifies the default board. The default board is set in the MILConfig utility.
- **DEVn** - Specifies the device number (rank) of the board (where 0 <= n <=15).

This filter allows to get the information about the logical level of all input ports of the device.

See Also

- Matrox_GetInputLevel – Gets input signal level.

Matrox_GetCounterReferenceValue

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets the I/O command list's internal counter value at the current moment.

Syntax

```
void avl::Matrox_GetCounterReferenceValue
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 int inCommandList,
 atl::int64& outReferenceValue
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inCommandList	int	1 - 2		I/O command list.
	outReferenceValue	int64&			Command list's internal counter value.

Remarks

I/O device driver software

This filter is intended to cooperate with a Matrox device using its vendor SDK. To be able to connect to the device, it is required to install Matrox MIL X SDK. Matrox industrial PCs usually come with a preinstalled runtime license.

Add DLL path to system environment variable may be required.

Recommended Matrox MIL X SDK version for Aurora Vision Studio usage is **V22H1 (10.50.924)**.

Device identification

inDeviceID field can be used to specify the number (rank) of the target board of the specified system type. DeviceID can be set to:

- **DEFAULT** - Specifies the default board. The default board is set in the MILConfig utility.
- **DEVn** - Specifies the device number (rank) of the board (where 0 <= n <=15).

This filter allows to obtain the current value of an I/O command list's counter. This value can later be used to register events at precise moments of time in relation to that value.

See Also

- Matrox_RegisterEdge – Registers an edge in a command list.
- Matrox_RegisterPulse – Registers a pulse in a command list.
- Matrox_RegisterImpulse – Registers an impulse in a command list.

Matrox_GetEncoderCounter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets the current value of an encoder counter.

Syntax

```
void avl::Matrox_GetEncoderCounter
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 int inEncoder,
 int& outCounterValue
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inEncoder	int	1 - 2		Encoder.
	outCounterValue	int&			Value of the encoder counter.

Remarks

I/O device driver software

This filter is intended to cooperate with a Matrox device using its vendor SDK. To be able to connect to the device, it is required to install Matrox MIL X SDK. Matrox industrial PCs usually come with a preinstalled runtime license.

Add DLL path to system environment variable may be required.

Recommended Matrox MIL X SDK version for Aurora Vision Studio usage is **V22H1 (10.50.924)**.

Device identification

inDeviceID field can be used to specify the number (rank) of the target board of the specified system type. DeviceID can be set to:

- **DEFAULT** - Specifies the default board. The default board is set in the MILConfig utility.
- **DEVn** - Specifies the device number (rank) of the board (where 0 <= n <=15).

This filter allows to obtain the current value of the encoder counter for one of the quadrature decoders.

See Also

- Matrox_ResetEncoderCounter – Resets the value of an encoder counter.

Matrox_GetInputLevel

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets input signal level.

Syntax

```
void avl::Matrox_GetInputLevel
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 int inInput,
 int inDebounceTime,
 bool& outLevel
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInput	int	1 - 8		Input signal.
	inDebounceTime	int	0 - ∞		Amount of time the input signal is debounced (in ns).
	outLevel	bool&			Logic level of the input signal.

Remarks

I/O device driver software

This filter is intended to cooperate with a Matrox device using its vendor SDK. To be able to connect to the device, it is required to install Matrox MIL X SDK. Matrox industrial PCs usually come with a preinstalled runtime license.

Add DLL path to system environment variable may be required.

Recommended Matrox MIL X SDK version for Aurora Vision Studio usage is **V22H1 (10.50.924)**.

Device identification

inDeviceID field can be used to specify the number (rank) of the target board of the specified system type. DeviceID can be set to:

- **DEFAULT** - Specifies the default board. The default board is set in the MILConfig utility.
- **DEVn** - Specifies the device number (rank) of the board (where 0 <= n <=15).

This filter allows to get the information about the logical level of a specific input port of the device.

See Also

- Matrox_GetAllInputLevels – Gets the state of all the inputs on the device.

Matrox_Modbus_ReadCoils

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads the coil data table in local memory for data received from a Modbus controller.

Syntax

```
void avl::Matrox_Modbus_ReadCoils
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 int inOffset,
 int inCount,
 atl::Array<bool>& outValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MLConfig utility.
	inOffset	int	0 - 65535		Offset from which to start reading the data in the specified data table.
	inCount	int	1 - 2000		Amount of the data to read.
	outValues	Array<bool>&			Received values.

Matrox_Modbus_ReadDiscreteInputs

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads the discrete input data table in local memory for data received from a Modbus controller.

Syntax

```
void avl::Matrox_Modbus_ReadDiscreteInputs
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 int inOffset,
 int inCount,
 atl::Array<bool>& outValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MLConfig utility.
	inOffset	int	0 - 65535		Offset from which to start reading the data in the specified data table.
	inCount	int	1 - 2000		Amount of the data to read.
	outValues	Array<bool>&			Received values.

Matrox_Modbus_ReadHoldingRegisters_ByteBuffer

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads the holding register data table as a ByteBuffer.

Syntax

```
void avl::Matrox_Modbus_ReadHoldingRegisters_ByteBuffer
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 int inOffset,
 int inCount,
 avl::ByteBuffer& outValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inOffset	int	0 - 65535		Offset from which to start reading the data in the specified data table.
	inCount	int	1 - 125		Amount of the data to read.
	outValues	ByteBuffer&			Received values.

Matrox_Modbus_ReadHoldingRegisters_float32

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads the holding register data table as 32-bit floating point numbers.

Syntax

```
void avl::Matrox_Modbus_ReadHoldingRegisters_float32
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 int inOffset,
 int inCount,
 bool inUseLittleEndian,
 atl::Array<float>& outValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inOffset	int	0 - 65534		Offset from which to start reading the data in the specified data table.
	inCount	int	1 - 62		Amount of the data to read.
	inUseLittleEndian	bool			Use little endian order for words in the registers.
	outValues	Array<float>&			Received values.

Matrox_Modbus_ReadHoldingRegisters_sint16

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads the holding register data table as signed 16-bit integers.

Syntax

```
void avl::Matrox_Modbus_ReadHoldingRegisters_sint16
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 int inOffset,
 int inCount,
 atl::Array<int>& outValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inOffset	int	0 - 65535		Offset from which to start reading the data in the specified data table.
	inCount	int	1 - 125		Amount of the data to read.
	outValues	Array<int>&			Received values.

Matrox_Modbus_ReadHoldingRegisters_sint32

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads the holding register data table as signed 32-bit integers.

Syntax

```
void avl::Matrox_Modbus_ReadHoldingRegisters_sint32
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 int inOffset,
 int inCount,
 bool inUseLittleEndian,
 atl::Array<int>& outValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inOffset	int	0 - 65534		Offset from which to start reading the data in the specified data table.
	inCount	int	1 - 62		Amount of the data to read.
	inUseLittleEndian	bool			Use little endian order for words in the registers.
	outValues	Array<int>&			Received values.

Matrox_Modbus_ReadHoldingRegisters_uint16

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads the holding register data table as unsigned 16-bit integers.

Syntax

```
void avl::Matrox_Modbus_ReadHoldingRegisters_uint16
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 int inOffset,
 int inCount,
 atl::Array<int>& outValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MLConfig utility.
	inOffset	int	0 - 65535		Offset from which to start reading the data in the specified data table.
	inCount	int	1 - 125		Amount of the data to read.
	outValues	Array<int>&			Received values.

Matrox_Modbus_ReadInputRegisters_ByteBuffer

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads the input register data table as a ByteBuffer.

Syntax

```
void avl::Matrox_Modbus_ReadInputRegisters_ByteBuffer
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 int inOffset,
 int inCount,
 avl::ByteBuffer& outValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MLConfig utility.
	inOffset	int	0 - 65535		Offset from which to start reading the data in the specified data table.
	inCount	int	1 - 125		Amount of the data to read.
	outValues	ByteBuffer&			Received values.

Matrox_Modbus_ReadInputRegisters_float32

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads the input register data table as 32-bit floating point numbers.

Syntax

```
void avl::Matrox_Modbus_ReadInputRegisters_float32
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 int inOffset,
 int inCount,
 bool inUseLittleEndian,
 atl::Array<float>& outValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inOffset	int	0 - 65534		Offset from which to start reading the data in the specified data table.
	inCount	int	1 - 62		Amount of the data to read.
	inUseLittleEndian	bool			Use little endian order for words in the registers.
	outValues	Array<float>&			Received values.

Matrox_Modbus_ReadInputRegisters_sint16

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads the input register data table as signed 16-bit integers.

Syntax

```
void avl::Matrox_Modbus_ReadInputRegisters_sint16
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 int inOffset,
 int inCount,
 atl::Array<int>& outValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inOffset	int	0 - 65535		Offset from which to start reading the data in the specified data table.
	inCount	int	1 - 125		Amount of the data to read.
	outValues	Array<int>&			Received values.

Matrox_Modbus_ReadInputRegisters_sint32

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads the input register data table as signed 32-bit integers.

Syntax

```
void avl::Matrox_Modbus_ReadInputRegisters_sint32
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 int inOffset,
 int inCount,
 bool inUseLittleEndian,
 atl::Array<int>& outValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inOffset	int	0 - 65534		Offset from which to start reading the data in the specified data table.
	inCount	int	1 - 62		Amount of the data to read.
	inUseLittleEndian	bool			Use little endian order for words in the registers.
	outValues	Array<int>&			Received values.

Matrox_Modbus_ReadInputRegisters_uint16

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads the input register data table as unsigned 16-bit integers.

Syntax

```
void avl::Matrox_Modbus_ReadInputRegisters_uint16
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 int inOffset,
 int inCount,
 atl::Array<int>& outValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inOffset	int	0 - 65535		Offset from which to start reading the data in the specified data table.
	inCount	int	1 - 125		Amount of the data to read.
	outValues	Array<int>&			Received values.

Matrox_Modbus_WriteCoils

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Writes the coil data table in local memory to send to the Modbus controller.

Syntax

```
void avl::Matrox_Modbus_WriteCoils
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 int inOffset,
 const atl::Array<bool>& inValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inOffset	int	0 - 65535		Offset to which to start writing the data in the specified data table.
	inValues	const Array<bool>&			Values to write.

Matrox_Modbus_WriteHoldingRegisters_ByteBuffer

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Writes the holding register data table as a ByteBuffer.

Syntax

```
void avl::Matrox_Modbus_WriteHoldingRegisters_ByteBuffer
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 int inOffset,
 const avl::ByteBuffer& inValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inOffset	int	0 - 65535		Offset to which to start writing the data in the specified data table.
	inValues	const ByteBuffer&			Values to write.

Matrox_Modbus_WriteHoldingRegisters_float32

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Writes the holding register data table as 32-bit floating point numbers.

Syntax

```
void avl::Matrox_Modbus_WriteHoldingRegisters_float32
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 int inOffset,
 bool inUseLittleEndian,
 const atl::Array<float>& inValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inOffset	int	0 - 65534		Offset to which to start writing the data in the specified data table.
	inUseLittleEndian	bool			Use little endian order for words in the registers.
	inValues	const Array<float>&			Values to write.

Matrox_Modbus_WriteHoldingRegisters_sint16

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Writes the holding register data table as signed 16-bit integers.

Syntax

```
void avl::Matrox_Modbus_WriteHoldingRegisters_sint16
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 int inOffset,
 const atl::Array<int>& inValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inOffset	int	0 - 65535		Offset to which to start writing the data in the specified data table.
	inValues	const Array<int>&			Values to write.

Matrox_Modbus_WriteHoldingRegisters_sint32

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Writes the holding register data table as signed 32-bit integers.

Syntax

```
void avl::Matrox_Modbus_WriteHoldingRegisters_sint32
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 int inOffset,
 bool inUseLittleEndian,
 const atl::Array<int>& inValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inOffset	int	0 - 65534		Offset to which to start writing the data in the specified data table.
	inUseLittleEndian	bool			Use little endian order for words in the registers.
	inValues	const Array<int>&			Values to write.

Matrox_Modbus_WriteHoldingRegisters_uint16

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Writes the holding register data table as unsigned 16-bit integers.

Syntax

```
void avl::Matrox_Modbus_WriteHoldingRegisters_uint16
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 int inOffset,
 const atl::Array<int>& inValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inOffset	int	0 - 65535		Offset to which to start writing the data in the specified data table.
	inValues	const Array<int>&			Values to write.

Matrox_Profinet_GetControllerState

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets the state of the PROFINET controller (PLC) connected to the local computer.

Syntax

```
void avl::Matrox_Profinet_GetControllerState
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 avl::MatroxProfinetPlcStatus::Type& outStatus,
 avl::MatroxProfinetPlcMode::Type& outMode
)
```

Parameters

	Name	Type	Default	Description
	ioState	Matrox_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>	NIL	Device identification number.
	inInstanceName	const Optional<String>&	NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	outStatus	MatroxProfinetPlcStatus::Type&		Status of the PROFINET controller.
	outMode	MatroxProfinetPlcMode::Type&		Mode of the PROFINET controller.

Matrox_Profinet_Read_ByteBuffer

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads the module data as a ByteBuffer.

Syntax

```
void avl::Matrox_Profinet_Read_ByteBuffer
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 int inModuleIndex,
 int inOffset,
 int inCount,
 avl::ByteBuffer& outValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inModuleIndex	int	0 - 255		Index of the module in local memory from which to read the data.
	inOffset	int	0 - ∞		Offset from which to start reading the data in the specified module.
	inCount	int	1 - ∞		Amount of the data to read.
	outValues	ByteBuffer&			Received values.

Matrox_Profinet_Read_float32

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads the module data as 32-bit floating point numbers.

Syntax

```
void avl::Matrox_Profinet_Read_float32
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 int inModuleIndex,
 int inOffset,
 int inCount,
 bool inUseLittleEndian,
 atl::Array<float>& outValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inModuleIndex	int	0 - 255		Index of the module in local memory from which to read the data.
	inOffset	int	0 - ∞		Offset from which to start reading the data in the specified module.
	inCount	int	1 - ∞		Amount of the data to read.
	inUseLittleEndian	bool			Use little endian order for data in the memory.
	outValues	Array<float>&			Received values.

Matrox_Profinet_Read_sint16

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads the module data as signed 16-bit integers.

Syntax

```
void avl::Matrox_Profinet_Read_sint16
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 int inModuleIndex,
 int inOffset,
 int inCount,
 bool inUseLittleEndian,
 atl::Array<int>& outValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inModuleIndex	int	0 - 255		Index of the module in local memory from which to read the data.
	inOffset	int	0 - ∞		Offset from which to start reading the data in the specified module.
	inCount	int	1 - ∞		Amount of the data to read.
	inUseLittleEndian	bool			Use little endian order for data in the memory.
	outValues	Array<int>&			Received values.

Matrox_Profinet_Read_sint32

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads the module data as signed 32-bit integers.

Syntax

```
void avl::Matrox_Profinet_Read_sint32
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 int inModuleIndex,
 int inOffset,
 int inCount,
 bool inUseLittleEndian,
 atl::Array<int>& outValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inModuleIndex	int	0 - 255		Index of the module in local memory from which to read the data.
	inOffset	int	0 - ∞		Offset from which to start reading the data in the specified module.
	inCount	int	1 - ∞		Amount of the data to read.
	inUseLittleEndian	bool			Use little endian order for data in the memory.
	outValues	Array<int>&			Received values.

Matrox_Profinet_Read_uint16

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads the module data as unsigned 16-bit integers.

Syntax

```
void avl::Matrox_Profinet_Read_uint16
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 int inModuleIndex,
 int inOffset,
 int inCount,
 bool inUseLittleEndian,
 atl::Array<int>& outValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inModuleIndex	int	0 - 255		Index of the module in local memory from which to read the data.
	inOffset	int	0 - ∞		Offset from which to start reading the data in the specified module.
	inCount	int	1 - ∞		Amount of the data to read.
	inUseLittleEndian	bool			Use little endian order for data in the memory.
	outValues	Array<int>&			Received values.

Matrox_Profinet_Write_ByteBuffer

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Writes the module data as a ByteBuffer.

Syntax

```
void avl::Matrox_Profinet_Write_ByteBuffer
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 int inModuleIndex,
 int inOffset,
 const avl::ByteBuffer& inValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inModuleIndex	int	0 - 255		Index of the module in local memory to which to write the data.
	inOffset	int	0 - ∞		Offset to which to start writing the data in the specified module.
	inValues	const ByteBuffer&			Values to write.

Matrox_Profinet_Write_float32

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Writes the module data as 32-bit floating point numbers.

Syntax

```
void avl::Matrox_Profinet_Write_float32
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 int inModuleIndex,
 int inOffset,
 bool inUseLittleEndian,
 const atl::Array<float>& inValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inModuleIndex	int	0 - 255		Index of the module in local memory to which to write the data.
	inOffset	int	0 - ∞		Offset to which to start writing the data in the specified module.
	inUseLittleEndian	bool			Use little endian order for data in the memory.
	inValues	const Array<float>&			Values to write.

Matrox_Profinet_Write_sint16

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Writes the module data as signed 16-bit integers.

Syntax

```
void avl::Matrox_Profinet_Write_sint16
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 int inModuleIndex,
 int inOffset,
 bool inUseLittleEndian,
 const atl::Array<int>& inValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inModuleIndex	int	0 - 255		Index of the module in local memory to which to write the data.
	inOffset	int	0 - ∞		Offset to which to start writing the data in the specified module.
	inUseLittleEndian	bool			Use little endian order for data in the memory.
	inValues	const Array<int>&			Values to write.

Matrox_Profinet_Write_sint32

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Writes the module data as signed 32-bit integers.

Syntax

```
void avl::Matrox_Profinet_Write_sint32
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 int inModuleIndex,
 int inOffset,
 bool inUseLittleEndian,
 const atl::Array<int>& inValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inModuleIndex	int	0 - 255		Index of the module in local memory to which to write the data.
	inOffset	int	0 - ∞		Offset to which to start writing the data in the specified module.
	inUseLittleEndian	bool			Use little endian order for data in the memory.
	inValues	const Array<int>&			Values to write.

Matrox_Profinet_Write_uint16

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Writes the module data as unsigned 16-bit integers.

Syntax

```
void avl::Matrox_Profinet_Write_uint16
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 const atl::Optional<atl::String>& inInstanceName,
 int inModuleIndex,
 int inOffset,
 bool inUseLittleEndian,
 const atl::Array<int>& inValues
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inInstanceName	const Optional<String>&		NIL	Instance name of a given Industrial Communication protocol set in the MILConfig utility.
	inModuleIndex	int	0 - 255		Index of the module in local memory to which to write the data.
	inOffset	int	0 - ∞		Offset to which to start writing the data in the specified module.
	inUseLittleEndian	bool			Use little endian order for data in the memory.
	inValues	const Array<int>&			Values to write.

Matrox_RegisterEdge

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Registers an edge in a command list.

Syntax

```
void avl::Matrox_RegisterEdge
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 int inCommandList,
 int inCommandBit,
 avl::MatroxEdgeType::Type inEdgeType,
 atl::Optional<atl::int64> inReferenceMoment,
 float inDelay
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inCommandList	int	1 - 2		I/O command list.
	inCommandBit	int	0 - 7		Bit of the command list.
	inEdgeType	MatroxEdgeType::Type			Type of the edge to register.
	inReferenceMoment	Optional<int64>		NIL	Value of counter for which the command is to be registered (in seconds or counts).
	inDelay	float	0.0 - ∞	0.0f	Delay from the specified counter value.

Remarks

I/O device driver software

This filter is intended to cooperate with a Matrox device using its vendor SDK. To be able to connect to the device, it is required to install Matrox MIL X SDK. Matrox industrial PCs usually come with a preinstalled runtime license.

Add DLL path to system environment variable may be required.

Recommended Matrox MIL X SDK version for Aurora Vision Studio usage is **V22H1 (10.50.924)**.

Device identification

inDeviceID field can be used to specify the number (rank) of the target board of the specified system type. DeviceID can be set to:

- **DEFAULT** - Specifies the default board. The default board is set in the MILConfig utility.
- **DEVn** - Specifies the device number (rank) of the board (where 0 <= n <=15).

This filter allows to register an edge in one of the command lists. The specified command list must be configured using **Matrox_ConfigureIOCommandList** prior to using this filter.

See Also

- Matrox_ConfigureIOCommandList – Configures an I/O command list.
- Matrox_GetCounterReferenceValue – Gets the I/O command list's internal counter value at the current moment.
- Matrox_RegisterPulse – Registers a pulse in a command list.
- Matrox_RegisterImpulse – Registers an impulse in a command list.

Matrox_RegisterImpulse

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Registers an impulse in a command list.

Syntax

```
void avl::Matrox_RegisterImpulse
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 int inCommandList,
 int inCommandBit,
 atl::Optional<atl::int64> inReferenceMoment,
 float inDelay
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inCommandList	int	1 - 2		I/O command list.
	inCommandBit	int	0 - 7		Bit of the command list.
	inReferenceMoment	Optional<int64>		NIL	Value of counter for which the command is to be registered (in seconds or counts).
	inDelay	float	0.0 - ∞	0.0f	Delay from the specified counter value.

Remarks

I/O device driver software

This filter is intended to cooperate with a Matrox device using its vendor SDK. To be able to connect to the device, it is required to install Matrox MIL X SDK. Matrox industrial PCs usually come with a preinstalled runtime license.

Add DLL path to system environment variable may be required.

Recommended Matrox MIL X SDK version for Aurora Vision Studio usage is **V22H1 (10.50.924)**.

Device identification

inDeviceID field can be used to specify the number (rank) of the target board of the specified system type. DeviceID can be set to:

- **DEFAULT** - Specifies the default board. The default board is set in the MILConfig utility.
- **DEVn** - Specifies the device number (rank) of the board (where 0 <= n <=15).

This filter allows to register an impulse in one of the command lists. The specified command list must be configured using **Matrox_ConfigureIOCommandList** prior to using this filter.

See Also

- Matrox_ConfigureIOCommandList – Configures an I/O command list.
- Matrox_GetCounterReferenceValue – Gets the I/O command list's internal counter value at the current moment.
- Matrox_RegisterEdge – Registers an edge in a command list.
- Matrox_RegisterPulse – Registers a pulse in a command list.

Matrox_RegisterPulse

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Registers a pulse in a command list.

Syntax

```
void avl::Matrox_RegisterPulse
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 int inCommandList,
 int inCommandBit,
 avl::MatroxPulseType::Type inPulseType,
 atl::Optional<atl::int64> inReferenceMoment,
 float inDelay,
 float inDuration
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inCommandList	int	1 - 2		I/O command list.
	inCommandBit	int	0 - 7		Bit of the command list.
	inPulseType	MatroxPulseType::Type			Type of the pulse to register.
	inReferenceMoment	Optional<int64>		NIL	Value of counter for which the command is to be registered (in seconds or counts).
	inDelay	float	0.0 - ∞	0.0f	Delay from the specified counter value.
	inDuration	float	0.0 - ∞	0.01f	Duration of the pulse.

Remarks

I/O device driver software

This filter is intended to cooperate with a Matrox device using its vendor SDK. To be able to connect to the device, it is required to install Matrox MIL X SDK. Matrox industrial PCs usually come with a preinstalled runtime license.

Add DLL path to system environment variable may be required.

Recommended Matrox MIL X SDK version for Aurora Vision Studio usage is **V22H1 (10.50.924)**.

Device identification

inDeviceID field can be used to specify the number (rank) of the target board of the specified system type. DeviceID can be set to:

- **DEFAULT** - Specifies the default board. The default board is set in the MILConfig utility.
- **DEVn** - Specifies the device number (rank) of the board (where 0 <= n <=15).

This filter allows to register a pulse in one of the command lists. The specified command list must be configured using **Matrox_ConfigureIOCommandList** prior to using this filter.

See Also

- Matrox_ConfigureIOCommandList – Configures an I/O command list.
- Matrox_GetCounterReferenceValue – Gets the I/O command list's internal counter value at the current moment.
- Matrox_RegisterEdge – Registers an edge in a command list.
- Matrox_RegisterImpulse – Registers an impulse in a command list.

Matrox_ResetEncoderCounter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Resets the value of an encoder counter.

Syntax

```
void avl::Matrox_ResetEncoderCounter
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 int inEncoder
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inEncoder	int	1 - 2		Encoder.

Remarks

I/O device driver software

This filter is intended to cooperate with a Matrox device using its vendor SDK. To be able to connect to the device, it is required to install Matrox MIL X SDK. Matrox industrial PCs usually come with a preinstalled runtime license.

Add DLL path to system environment variable may be required.

Recommended Matrox MIL X SDK version for Aurora Vision Studio usage is **V22H1 (10.50.924)**.

Device identification

inDeviceID field can be used to specify the number (rank) of the target board of the specified system type. DeviceID can be set to:

- **DEFAULT** - Specifies the default board. The default board is set in the MILConfig utility.
- **DEVn** - Specifies the device number (rank) of the board (where 0 <= n <=15).

This filter allows to reset the value of the encoder counter to 0.

See Also

- Matrox_GetEncoderCounter – Gets the current value of an encoder counter.

Matrox_SetOutputLevel

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets output signal level.

Syntax

```
void avl::Matrox_SetOutputLevel
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 int inOutput,
 bool inLevel,
 bool inForceUserOutputSource
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inOutput	int	1 - 8		Output signal.
	inLevel	bool			Logic level of the output signal.
	inForceUserOutputSource	bool			Whether to set the output source to user bit or not.

Remarks

I/O device driver software

This filter is intended to cooperate with a Matrox device using its vendor SDK. To be able to connect to the device, it is required to install Matrox MIL X SDK. Matrox industrial PCs usually come with a preinstalled runtime license.

Add DLL path to system environment variable may be required.

Recommended Matrox MIL X SDK version for Aurora Vision Studio usage is **V22H1 (10.50.924)**.

Device identification

inDeviceID field can be used to specify the number (rank) of the target board of the specified system type. DeviceID can be set to:

- **DEFAULT** - Specifies the default board. The default board is set in the MILConfig utility.
- **DEVn** - Specifies the device number (rank) of the board (where 0 <= n <=15).

This filter allows to set the logical level of a specific output port of the device.

Setting the output source

The source of the specific output port must be set to **User** using Matrox_SetOutputSource filter first for this filter to take effect. It is also possible to force an automatic change of the output source by setting the **inForceUserOutputSource** field value to **true**.

See Also

- Matrox_SetOutputSource – Sets output signal source.

Matrox_SetOutputSource

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets output signal source.

Syntax

```
void avl::Matrox_SetOutputSource
(
 Matrox_State& ioState,
 atl::Optional<const avl::MatroxDeviceID::Type&> inDeviceID,
 int inOutput,
 avl::MatroxOutputSource::Type inSource
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Matrox_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const MatroxDeviceID::Type&>		NIL	Device identification number.
	inOutput	int	1 - 8		Output signal.
	inSource	MatroxOutputSource::Type			Source signal routed to the output signal.

Remarks

I/O device driver software

This filter is intended to cooperate with a Matrox device using its vendor SDK. To be able to connect to the device, it is required to install Matrox MIL X SDK. Matrox industrial PCs usually come with a preinstalled runtime license.

Add DLL path to system environment variable may be required.

Recommended Matrox MIL X SDK version for Aurora Vision Studio usage is **V22H1 (10.50.924)**.

Device identification

inDeviceID field can be used to specify the number (rank) of the target board of the specified system type. DeviceID can be set to:

- **DEFAULT** - Specifies the default board. The default board is set in the MILConfig utility.
- **DEVn** - Specifies the device number (rank) of the board (where 0 <= n <=15).

This filter allows to route a source to a specific output port of the device.

See Also

- Matrox_SetOutputLevel – Sets output signal level.

171. Microview

Table of content:

Microview_GenerateSoftwareTrigger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Generates software trigger.

Syntax

```
void avl::Microview_GenerateSoftwareTrigger
(
 Microview_State& ioState,
 const atl::Optional<int> inDeviceID,
 bool inJumboFrameSupport,
 bool inRescanNeeded
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Microview_State&			Object used to maintain state of the function.
	inDeviceID	const Optional<int>	0 - 10000	NIL	Device identifying number
	inJumboFrameSupport	bool		True	
	inRescanNeeded	bool			

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Microview_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a frame using Microview.

Syntax

```
bool avl::Microview_GrabImage
(
 Microview_State& ioState,
 const atl::Optional<int> inDeviceID,
 int inInputQueueSize,
 bool inJumboFrameSupport,
 bool inRescanNeeded,
 const avl::MicroviewUsersSettingsParams& inUsersSettingsParams,
 const avl::MicroviewImageFormatParams& inImageFormat,
 const avl::MicroviewAcquisitionParams& inAcquisitionControl,
 const avl::MicroviewTriggerParams& inTriggerControl,
 avl::Image& outImage,
 int& outFrameId
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	Microview_State&			Object used to maintain state of the function.
→	inDeviceID	const Optional<int>	0 - 10000	NIL	Device identifying number
→	inInputQueueSize	int	1 - 200	4	Capacity of output frames queue
→	inJumboFrameSupport	bool		True	
→	inRescanNeeded	bool			
→	inUsersSettingsParams	const MicroviewUsersSettingsParams&			
→	inImageFormat	const MicroviewImageFormatParams&			
→	inAcquisitionControl	const MicroviewAcquisitionParams&			
→	inTriggerControl	const MicroviewTriggerParams&			
←	outImage	Image&			Captured frame
←	outFrameId	int&			Frame id

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install MVGESDK software with camera dedicated drivers.

MVGESDK can be downloaded from the following website: http://www.microview.com.cn (registration may be required).

Recommended MVGESDK version for Aurora Vision Studio usage is **2.8**.

Add DLL path to system environment variable may be required.

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to computer. DeviceID can be set to:

- **Device number** - should be specified on device casing.

Camera parameters

Setting **inJumboFrameSupport** parameter to 'true' will select maximum possible camera package size.

Setting **inRescanNeeded** parameter to 'true' will rescan network to search new cameras.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Microview_GrabImage_WithTimeout – Captures with timeout a frame using Microview.
- Microview_StartAcquisition – Starts acquisition using Microview.

Microview_GrabImage_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures with timeout a frame using Microview.

Syntax

```
bool avl::Microview_GrabImage_WithTimeout
(
 Microview_State& ioState,
 const atl::Optional<int> inDeviceID,
 int inInputQueueSize,
 bool inJumboFrameSupport,
 bool inRescanNeeded,
 const avl::MicroviewUsersSettingsParams& inUsersSettingsParams,
 const avl::MicroviewImageFormatParams& inImageFormat,
 const avl::MicroviewAcquisitionParams& inAcquisitionControl,
 const avl::MicroviewTriggerParams& inTriggerControl,
 const atl::Optional<int> inTimeout,
 atl::Conditional<avl::Image>& outImage,
 atl::Conditional<int>& outFrameId
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Microview_State&			Object used to maintain state of the function.
	inDeviceID	const Optional<int>	0 - 10000	NIL	Device identifying number
	inInputQueueSize	int	1 - 200	4	Capacity of output frames queue
	inJumboFrameSupport	bool		True	
	inRescanNeeded	bool			
	inUsersSettingsParams	const MicroviewUsersSettingsParams&			
	inImageFormat	const MicroviewImageFormatParams&			
	inAcquisitionControl	const MicroviewAcquisitionParams&			
	inTriggerControl	const MicroviewTriggerParams&			
	inTimeout	const Optional<int>	1 - ∞	100	
	outImage	Conditional<Image>&			Captured frame
	outFrameId	Conditional<int>&			Frame id

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install MVGESDK software with camera dedicated drivers.

MVGESDK can be downloaded from the following website: http://www.microview.com.cn (registration may be required).

Recommended MVGESDK version for Aurora Vision Studio usage is **2.8**.

Add DLL path to system environment variable may be required.

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to computer. DeviceID can be set to:

- **Device number** - should be specified on device casing.

Camera parameters

Setting **inJumboFrameSupport** parameter to 'true' will select maximum possible camera package size.

Setting **inRescanNeeded** parameter to 'true' will rescan network to search new cameras.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Microview_GrabImage – Captures a frame using Microview.
- Microview_StartAcquisition – Starts acquisition using Microview.

Microview_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Starts acquisition using Microview.

Syntax

```
void avl::Microview_StartAcquisition
(
 Microview_State& ioState,
 const atl::Optional<int> inDeviceID,
 int inInputQueueSize,
 bool inJumboFrameSupport,
 bool inRescanNeeded,
 const avl::MicroviewUsersSettingsParams& inUsersSettingsParams,
 const avl::MicroviewImageFormatParams& inImageFormat,
 const avl::MicroviewAcquisitionParams& inAcquisitionControl,
 const avl::MicroviewTriggerParams& inTriggerControl
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Microview_State&			Object used to maintain state of the function.
	inDeviceID	const Optional<int>	0 - 10000	NIL	Device identifying number
	inInputQueueSize	int	1 - 200	4	Capacity of output frames queue
	inJumboFrameSupport	bool		True	
	inRescanNeeded	bool			
	inUsersSettingsParams	const MicroviewUsersSettingsParams&			
	inImageFormat	const MicroviewImageFormatParams&			
	inAcquisitionControl	const MicroviewAcquisitionParams&			
	inTriggerControl	const MicroviewTriggerParams&			

Remarks

This filter is intended for establishing connection with a Microview camera device using MVGESDK interface, to initialize image streaming. It is only needed when explicit image acquisition start is required in the initial phase of a program. For example, it can be used to prepare a camera, running in triggered mode, to be able to capture trigger signals before the first invoke of Microview_GrabImage or to start multiple cameras in sync before the acquisition phase.

The use of this filter is not obligatory. Microview_GrabImage or Microview_GrabImage_WithTimeout] filters will initialize and start image acquisition upon their first invoke.

This filter has no effect when invoked for the second time and when invoked after image grabbing filters.

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install MVGESDK software with camera dedicated drivers.

MVGESDK can be downloaded from the following website: http://www.microview.com.cn (registration may be required).

Recommended MVGESDK version for Aurora Vision Studio usage is **2.8**.

Add DLL path to system environment variable may be required.

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to computer. DeviceID can be set to:

- **Device number** - should be specified on device casing.

Camera parameters

Setting **inJumboFrameSupport** parameter to 'true' will select maximum possible camera package size.

Setting **inRescanNeeded** parameter to 'true' will rescan network to search new cameras.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Microview_GrabImage – Captures a frame using Microview.
- Microview_GrabImage_WithTimeout – Captures with timeout a frame using Microview.

Microview_StopAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Stops acquisition using Microview.

Syntax

```
void avl::Microview_StopAcquisition
(
 Microview_State& ioState,
 const atl::Optional<int> inDeviceID,
 bool inJumboFrameSupport,
 bool inRescanNeeded
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Microview_State&			Object used to maintain state of the function.
	inDeviceID	const Optional<int>	0 - 10000	NIL	Device identifying number
	inJumboFrameSupport	bool		True	
	inRescanNeeded	bool			

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

172. Euresys

Table of content:

MultiCam_GetDigitalInput

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reporting the logic state of I/O lines used as inputs.

Syntax

```
void avl::MultiCam_GetDigitalInput
(
 MultiCam_State& ioState,
 atl::Optional<atl::String> inDeviceID,
 int inInput,
 const atl::Optional<avl::MultiCamInputStyle::Type>& inInputStyle,
 avl::MultiCamInputState::Type& outState
)
```

Parameters

	Name	Type	Default	Description
	ioState	MultiCam_State&		Object used to maintain state of the function.
	inDeviceID	Optional<String>	NIL	Board name, board identifier or serial number
	inInput	int		Input pin index
	inInputStyle	const Optional<MultiCamInputStyle::Type>&	NIL	Controlling the electrical style of I/O lines used as inputs. If Nil then lease style as it is
	outState	MultiCamInputState::Type&		Returns the current logic state of I/O line

Remarks

Board driver software

This filter is intended to cooperate with board using its vendor SDK. To be able to connect with board it is required to install MultiCam SDK software. Currently Aurora Vision Studio requires **MultiCam version 6.18**.

MultiCam drivers can be downloaded from following website: https://www.euresys.com (registration may be required).

Supported frame grabbers:

- GRABLINK series
- DOMINO series
- Picolo series

Board identification

When there is only one board connected to the computer, field **inDeviceID** can be set to *Auto*. In such case, first available board will be found and connected.

inDeviceID can be used to pick one of multiple boards connected to the computer. **inDeviceID** can be set to:

- board name,
- board identifier,
- driver index,
- pci position.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- MultiCam_GrabImage – Captures a frame using Euresys frame grabber.
- MultiCam_SetDigitalOutput – Sets digital output.

MultiCam_GetIntegerParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type integer from euresys device.

Syntax

```
void avl::MultiCam_GetIntegerParameter
(
 MultiCam_State& ioState,
 atl::Optional<atl::String> inDeviceID,
 atl::Optional<atl::String> inConnector,
 avl::MultiCamDestinationClass::Type inDestinationClass,
 const atl::String& inName,
 int& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	MultiCam_State&		Object used to maintain state of the function.
	inDeviceID	Optional<String>	NIL	Board name, board identifier or serial number
	inConnector	Optional<String>	NIL	Connector
	inDestinationClass	MultiCamDestinationClass::Type		Class where the value should be set (board or channel)
	inName	const String&		Parameter name
	outValue	int&		Parameter value

Remarks

Board driver software

This filter is intended to cooperate with board using its vendor SDK. To be able to connect with board it is required to install MultiCam SDK software. Currently Aurora Vision Studio requires **MultiCam version 6.18**.

MultiCam drivers can be downloaded from following website: https://www.euresys.com (registration may be required).

Supported frame grabbers:

- GRABLINK series
- DOMINO series
- Picolo series

Board identification

When there is only one board connected to the computer, field **inDeviceID** can be set to *Auto*. In such case, first available board will be found and connected.

inDeviceID can be used to pick one of multiple boards connected to the computer. **inDeviceID** can be set to:

- board name,
- board identifier,
- driver index,
- pci position.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- MultiCam_GrabImage – Captures a frame using Euresys frame grabber.

MultiCam_GetRealParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type real from euresys device.

Syntax

```
void avl::MultiCam_GetRealParameter
(
 MultiCam_State& ioState,
 atl::Optional<atl::String> inDeviceID,
 atl::Optional<atl::String> inConnector,
 avl::MultiCamDestinationClass::Type inDestinationClass,
 const atl::String& inName,
 float& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	MultiCam_State&		Object used to maintain state of the function.
	inDeviceID	Optional<String>	NIL	Board name, board identifier or serial number
	inConnector	Optional<String>	NIL	Connector
	inDestinationClass	MultiCamDestinationClass::Type		Class where the value should be set (board or channel)
	inName	const String&		Parameter name
	outValue	float&		Parameter value

Remarks

Board driver software

This filter is intended to cooperate with board using its vendor SDK. To be able to connect with board it is required to install MultiCam SDK software. Currently Aurora Vision Studio requires **MultiCam version 6.18**.

MultiCam drivers can be downloaded from following website: https://www.euresys.com (registration may be required).

Supported frame grabbers:

- GRABLINK series
- DOMINO series
- Picolo series

Board identification

When there is only one board connected to the computer, field **inDeviceID** can be set to *Auto*. In such case, first available board will be found and connected.

inDeviceID can be used to pick one of multiple boards connected to the computer. **inDeviceID** can be set to:

- board name,
- board identifier,
- driver index,
- pci position.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- MultiCam_GrabImage – Captures a frame using Euresys frame grabber.

MultiCam_GetStringParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type String from euresys device.

Syntax

```
void avl::MultiCam_GetStringParameter
(
 MultiCam_State& ioState,
 atl::Optional<atl::String> inDeviceID,
 atl::Optional<atl::String> inConnector,
 avl::MultiCamDestinationClass::Type inDestinationClass,
 const atl::String& inName,
 atl::String& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	MultiCam_State&		Object used to maintain state of the function.
	inDeviceID	Optional<String>	NIL	Board name, board identifier or serial number
	inConnector	Optional<String>	NIL	Connector
	inDestinationClass	MultiCamDestinationClass::Type		Class where the value should be set (board or channel)
	inName	const String&		Parameter name
	outValue	String&		Parameter value

Remarks

Board driver software

This filter is intended to cooperate with board using its vendor SDK. To be able to connect with board it is required to install MultiCam SDK software. Currently Aurora Vision Studio requires **MultiCam version 6.18**.

MultiCam drivers can be downloaded from following website: https://www.euresys.com (registration may be required).

Supported frame grabbers:

- GRABLINK series
- DOMINO series
- Picolo series

Board identification

When there is only one board connected to the computer, field **inDeviceID** can be set to *Auto*. In such case, first available board will be found and connected.

inDeviceID can be used to pick one of multiple boards connected to the computer. **inDeviceID** can be set to:

- board name,
- board identifier,
- driver index,
- pci position.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- MultiCam_GrabImage – Captures a frame using Euresys frame grabber.

MultiCam_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a frame using Euresys frame grabber.

Syntax

```
bool avl::MultiCam_GrabImage
(
 MultiCam_State& ioState,
 atl::Optional<atl::String> inDeviceID,
 atl::Optional<atl::String> inTopology,
 atl::Optional<atl::String> inCameraTapConfiguration,
 const atl::String& inConnector,
 avl::MultiCamAcquisitionMode::Type inAcquisitionMode,
 atl::Optional<const atl::File&> inCameraFile,
 atl::Optional<int> inPageLengthLn,
 avl::MultiCamColorFormat::Type inColorFormat,
 atl::Optional<int> inGain,
 atl::Optional<int> inSurfaceCount,
 avl::Image& outImage,
 atl::int64& outFrameID,
 atl::int64& outTimestamp
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	MultiCam_State&			Object used to maintain state of the function.
	inDeviceID	Optional<String>		NIL	Board name, board identifier or serial number
	inTopology	Optional<String>		NIL	Board topology
	inCameraTapConfiguration	Optional<String>		NIL	Tap configuration of camera
	inConnector	const String&			Indication of connector used by channel
	inAcquisitionMode	MultiCamAcquisitionMode::Type			Acquisition Mode
	inCameraFile	Optional<const File&>		NIL	Camera configuration file
	inPageLengthLn	Optional<int>	1 - 65535	NIL	Page Length
	inColorFormat	MultiCamColorFormat::Type			Color format
	inGain	Optional<int>		NIL	Linear control of gain for all digitizing units
	inSurfaceCount	Optional<int>	1 - 4096	(3)	Number of allocated surfaces
	outImage	Image&			Captured frame
	outFrameID	int64&			Captured frame ID
	outTimestamp	int64&			Captured frame timestamp in microseconds

Remarks

Board driver software

This filter is intended to cooperate with board using its vendor SDK. To be able to connect with board it is required to install MultiCam SDK software. Currently Aurora Vision Studio requires **MultiCam version 6.18**.

MultiCam drivers can be downloaded from following website: https://www.euresys.com (registration may be required).

Supported frame grabbers:

- GRABLINK series
- DOMINO series
- Picolo series

Board identification

When there is only one board connected to the computer, field **inDeviceID** can be set to *Auto*. In such case, first available board will be found and connected.

inDeviceID can be used to pick one of multiple boards connected to the computer. **inDeviceID** can be set to:

- board name,
- board identifier,
- driver index,
- pci position.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- MultiCam_StartAcquisition – Initializes and starts image acquisition by frame grabber.
- MultiCam_GetDigitalInput – Reporting the logic state of I/O lines used as inputs.
- MultiCam_SetDigitalOutput – Sets digital output.

MultiCam_GrabImage_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a frame using Euresys frame grabber.

Syntax

```
bool avl::MultiCam_GrabImage_WithTimeout
(
 MultiCam_State& ioState,
 atl::Optional<atl::String> inDeviceID,
 atl::Optional<atl::String> inTopology,
 atl::Optional<int> inTimeout,
 atl::Optional<atl::String> inCameraTapConfiguration,
 const atl::String& inConnector,
 avl::MultiCamAcquisitionMode::Type inAcquisitionMode,
 atl::Optional<const atl::File&> inCameraFile,
 atl::Optional<int> inPageLengthLn,
 avl::MultiCamColorFormat::Type inColorFormat,
 atl::Optional<int> inGain,
 atl::Optional<int> inSurfaceCount,
 atl::Conditional<avl::Image>& outImage,
 atl::Conditional<atl::int64>& outFrameID,
 atl::Conditional<atl::int64>& outTimestamp
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	MultiCam_State&			Object used to maintain state of the function.
	inDeviceID	Optional<String>		NIL	Board name, board identifier or serial number
	inTopology	Optional<String>		NIL	Board topology
	inTimeout	Optional<int>	10 - ∞	NIL	Timeout in milliseconds.
	inCameraTapConfiguration	Optional<String>		NIL	Tap configuration of camera
	inConnector	const String&			Indication of connector used by channel
	inAcquisitionMode	MultiCamAcquisitionMode::Type			Acquisition Mode
	inCameraFile	Optional<const File&>		NIL	Camera configuration file
	inPageLengthLn	Optional<int>	1 - 65535	NIL	Page Length
	inColorFormat	MultiCamColorFormat::Type			Color format
	inGain	Optional<int>		NIL	Linear control of gain for all digitizing units
	inSurfaceCount	Optional<int>	1 - 4096	(3)	Number of allocated surfaces
	outImage	Conditional<Image>&			Captured frame
	outFrameID	Conditional<int64>&			Captured frame ID
	outTimestamp	Conditional<int64>&			Captured frame timestamp in microseconds

Remarks

Board driver software

This filter is intended to cooperate with board using its vendor SDK. To be able to connect with board it is required to install MultiCam SDK software. Currently Aurora Vision Studio requires **MultiCam version 6.18**.

MultiCam drivers can be downloaded from following website: https://www.euresys.com (registration may be required).

Supported frame grabbers:

- GRABLINK series
- DOMINO series
- Picolo series

Board identification

When there is only one board connected to the computer, field **inDeviceID** can be set to *Auto*. In such case, first available board will be found and connected.

inDeviceID can be used to pick one of multiple boards connected to the computer. **inDeviceID** can be set to:

- board name,
- board identifier,
- driver index,
- pci position.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- MultiCam_StartAcquisition – Initializes and starts image acquisition by frame grabber.
- MultiCam_GrabImage – Captures a frame using Euresys frame grabber.

MultiCam_LoadConfigurationFile

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Loads camera configuration file.

Syntax

```
void avl::MultiCam_LoadConfigurationFile
(
 MultiCam_State& ioState,
 atl::Optional<atl::String> inDeviceID,
 atl::Optional<atl::String> inTopology,
 const atl::String& inConnector,
 atl::Optional<const atl::File&> inCameraFile
)
```

Parameters

	Name	Type	Default	Description
	ioState	MultiCam_State&		Object used to maintain state of the function.
	inDeviceID	Optional<String>	NIL	Board name, board identifier or serial number
	inTopology	Optional<String>	NIL	Board topology
	inConnector	const String&		Indication of connector used by channel
	inCameraFile	Optional<const File&>	NIL	Camera configuration file

Remarks

Board driver software

This filter is intended to cooperate with board using its vendor SDK. To be able to connect with board it is required to install MultiCam SDK software. Currently Aurora Vision Studio requires **MultiCam version 6.18**.

MultiCam drivers can be downloaded from following website: https://www.euresys.com (registration may be required).

Supported frame grabbers:

- GRABLINK series
- DOMINO series
- Picolo series

Board identification

When there is only one board connected to the computer, field **inDeviceID** can be set to *Auto*. In such case, first available board will be found and connected.

inDeviceID can be used to pick one of multiple boards connected to the computer. **inDeviceID** can be set to:

- board name,
- board identifier,
- driver index,
- pci position.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- MultiCam_GrabImage – Captures a frame using Euresys frame grabber.

MultiCam_SetDigitalOutput

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets digital output.

Syntax

```
void avl::MultiCam_SetDigitalOutput
(
 MultiCam_State& ioState,
 atl::Optional<atl::String> inDeviceID,
 int inOutput,
 const atl::Optional<avl::MultiCamOutputStyle::Type>& inOutputStyle,
 avl::MultiCamOutputState::Type inOutputState
)
```

Parameters

	Name	Type	Default	Description
	ioState	MultiCam_State&		Object used to maintain state of the function.
	inDeviceID	Optional<String>	NIL	Board name, board identifier or serial number
	inOutput	int		Output pin index
	inOutputStyle	const Optional<MultiCamOutputStyle::Type>&	NIL	Controlling the electrical style of I/O lines used as outputs. If Nil then lease style as it is
	inOutputState	MultiCamOutputState::Type		Issuing the logic state of I/O lines used as outputs

Remarks

Board driver software

This filter is intended to cooperate with board using its vendor SDK. To be able to connect with board it is required to install MultiCam SDK software. Currently Aurora Vision Studio requires **MultiCam version 6.18**.

MultiCam drivers can be downloaded from following website: https://www.euresys.com (registration may be required).

Supported frame grabbers:

- GRABLINK series
- DOMINO series
- Picolo series

Board identification

When there is only one board connected to the computer, field **inDeviceID** can be set to *Auto*. In such case, first available board will be found and connected.

inDeviceID can be used to pick one of multiple boards connected to the computer. **inDeviceID** can be set to:

- board name,
- board identifier,
- driver index,
- pci position.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- MultiCam_GetDigitalInput – Reporting the logic state of I/O lines used as inputs.
- MultiCam_GrabImage – Captures a frame using Euresys frame grabber.

MultiCam_SetIntegerParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type integer into euresys device.

Syntax

```
void avl::MultiCam_SetIntegerParameter
(
 MultiCam_State& ioState,
 atl::Optional<atl::String> inDeviceID,
 atl::Optional<atl::String> inConnector,
 avl::MultiCamDestinationClass::Type inDestinationClass,
 const atl::String& inName,
 int inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	MultiCam_State&		Object used to maintain state of the function.
	inDeviceID	Optional<String>	NIL	Board name, board identifier or serial number
	inConnector	Optional<String>	NIL	Connector
	inDestinationClass	MultiCamDestinationClass::Type		Class where the value should be set (board or channel)
	inName	const String&		Parameter name
	inValue	int		Parameter value

Remarks

Board driver software

This filter is intended to cooperate with board using its vendor SDK. To be able to connect with board it is required to install MultiCam SDK software. Currently Aurora Vision Studio requires **MultiCam version 6.18**.

MultiCam drivers can be downloaded from following website: https://www.euresys.com (registration may be required).

Supported frame grabbers:

- GRABLINK series
- DOMINO series
- Picolo series

Board identification

When there is only one board connected to the computer, field **inDeviceID** can be set to *Auto*. In such case, first available board will be found and connected.

inDeviceID can be used to pick one of multiple boards connected to the computer. **inDeviceID** can be set to:

- board name,
- board identifier,
- driver index,
- pci position.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- MultiCam_GrabImage – Captures a frame using Euresys frame grabber.

MultiCam_SetRealParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type real into euresys device.

Syntax

```
void avl::MultiCam_SetRealParameter
(
 MultiCam_State& ioState,
 atl::Optional<atl::String> inDeviceID,
 atl::Optional<atl::String> inConnector,
 avl::MultiCamDestinationClass::Type inDestinationClass,
 const atl::String& inName,
 float inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	MultiCam_State&		Object used to maintain state of the function.
	inDeviceID	Optional<String>	NIL	Board name, board identifier or serial number
	inConnector	Optional<String>	NIL	Connector
	inDestinationClass	MultiCamDestinationClass::Type		Class where the value should be set (board or channel)
	inName	const String&		Parameter name
	inValue	float		Parameter value

Remarks

Board driver software

This filter is intended to cooperate with board using its vendor SDK. To be able to connect with board it is required to install MultiCam SDK software. Currently Aurora Vision Studio requires **MultiCam version 6.18**.

MultiCam drivers can be downloaded from following website: https://www.euresys.com (registration may be required).

Supported frame grabbers:

- GRABLINK series
- DOMINO series
- Picolo series

Board identification

When there is only one board connected to the computer, field **inDeviceID** can be set to *Auto*. In such case, first available board will be found and connected.

inDeviceID can be used to pick one of multiple boards connected to the computer. **inDeviceID** can be set to:

- board name,
- board identifier,
- driver index,
- pci position.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- MultiCam_GrabImage – Captures a frame using Euresys frame grabber.

MultiCam_SetStringParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type String into euresys device.

Syntax

```
void avl::MultiCam_SetStringParameter
(
 MultiCam_State& ioState,
 atl::Optional<atl::String> inDeviceID,
 atl::Optional<atl::String> inConnector,
 avl::MultiCamDestinationClass::Type inDestinationClass,
 const atl::String& inName,
 const atl::String& inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	MultiCam_State&		Object used to maintain state of the function.
	inDeviceID	Optional<String>	NIL	Board name, board identifier or serial number
	inConnector	Optional<String>	NIL	Connector
	inDestinationClass	MultiCamDestinationClass::Type		Class where the value should be set (board or channel)
	inName	const String&		Parameter name
	inValue	const String&		Parameter value

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

MultiCam_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Initializes and starts image acquisition by frame grabber.

Syntax

```
void avl::MultiCam_StartAcquisition
(
 MultiCam_State& ioState,
 atl::Optional<atl::String> inDeviceID,
 atl::Optional<atl::String> inTopology,
 atl::Optional<atl::String> inCameraTapConfiguration,
 const atl::String& inConnector,
 avl::MultiCamAcquisitionMode::Type inAcquisitionMode,
 atl::Optional<const atl::File&> inCameraFile,
 atl::Optional<int> inPageLengthLn,
 avl::MultiCamColorFormat::Type inColorFormat,
 atl::Optional<int> inGain,
 atl::Optional<int> inSurfaceCount
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	MultiCam_State&			Object used to maintain state of the function.
	inDeviceID	Optional<String>		NIL	Board name, board identifier or serial number
	inTopology	Optional<String>		NIL	Board topology
	inCameraTapConfiguration	Optional<String>		NIL	Tap configuration of camera
	inConnector	const String&			Indication of connector used by channel
	inAcquisitionMode	MultiCamAcquisitionMode::Type			Acquisition Mode
	inCameraFile	Optional<const File&>		NIL	Camera configuration file
	inPageLengthLn	Optional<int>	1 - 65535	NIL	Page Length
	inColorFormat	MultiCamColorFormat::Type			Color format
	inGain	Optional<int>		NIL	Linear control of gain for all digitizing units
	inSurfaceCount	Optional<int>	1 - 4096	(3)	Number of allocated surfaces

Remarks

Board driver software

This filter is intended to cooperate with board using its vendor SDK. To be able to connect with board it is required to install MultiCam SDK software. Currently Aurora Vision Studio requires **MultiCam version 6.18**.

MultiCam drivers can be downloaded from following website: https://www.euresys.com (registration may be required).

Supported frame grabbers:

- GRABLINK series
- DOMINO series
- Picolo series

Board identification

When there is only one board connected to the computer, field **inDeviceID** can be set to *Auto*. In such case, first available board will be found and connected.

inDeviceID can be used to pick one of multiple boards connected to the computer. **inDeviceID** can be set to:

- board name,
- board identifier,
- driver index,
- pci position.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- MultiCam_GrabImage – Captures a frame using Euresys frame grabber.
- MultiCam_GrabImage_WithTimeout – Captures a frame using Euresys frame grabber.

173. MATRIX VISION

Table of content:

mvGenTLAcquire_GenerateSoftwareTrigger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Generates software trigger.

Syntax

```
void avl::mvGenTLAcquire_GenerateSoftwareTrigger
(
 mvGenTLAcquireState& ioState,
 atl::Optional<const atl::String&> inDeviceID
)
```

Parameters

	Name	Type	Default	Description
	ioState	mvGenTLAcquireState&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device name or serial number of device

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

mvGenTLAcquire_GetDigitalIOState

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Returns the state of digital inputs or outputs from matrix vision cameras.

Syntax

```
void avl::mvGenTLAcquire_GetDigitalIOState
(
 mvGenTLAcquireState& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inIOIndex,
 bool& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	mvGenTLAcquireState&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device name or serial number of device
	inIOIndex	int		Index of input or output line
	outValue	bool&		State of IO line

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install mvGenTL Acquire SDK software with camera dedicated drivers.

mvGenTL Acquire SDK software can be downloaded from the following website: www.matrix-vision.com. For Aurora Vision Studio 32-bit, mvGenTL Acquire x86 version is needed. For Aurora Vision Studio 64-bit, mvGenTL Acquire x86_x64 version of SDK is required.

Recommended mvGenTL Acquire SDK version for usage with Aurora Vision Studio is **2.49.0**.

Camera identification

When there is only one camera connected to the computer, field **inDeviceID** can be set to *Auto*. In such case, first available camera will be found and connected.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** can be set to:

- **serial number** - it should be specified on the device casing(e.g. *GX001559*),
- **product name** - it finds camera by name (might be the same for many cameras). The product name is a bit more specific than the family name, but less specific than the serial number(e.g. *mvBlueCOUGAR-X100wG*),
- **product number** - it finds camera by family name(e.g. *mvBlueCOUGAR*).

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- mvGenTLAcquire_GrabImage – Captures a frame from MATRIX VISION cameras using mvGenTLAcquire SDK.
- mvGenTLAcquire_SetDigitalOutput – Sets matrix vision camera digital outputs.

mvGenTLAcquire_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a frame from MATRIX VISION cameras using mvGenTLAcquire SDK.

Syntax

```
bool avl::mvGenTLAcquire_GrabImage
(
 mvGenTLAcquireState& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 atl::Optional<avl::MVGenTLPixelFormat::Type> inPixelFormat,
 atl::Optional<int> inExposureTime,
 atl::Optional<float> inGain,
 atl::Optional<float> inFrameRate,
 atl::Optional<const avl::Box&> inAoi,
 avl::MvIMPACTTriggerActivation::Type inTriggerMode,
 avl::MvIMPACTTriggerSource::Type inTriggerSource,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	mvGenTLAcquireState&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Device name or serial number of device
	inPixelFormat	Optional<MVGenTLPixelFormat::Type>		NIL	Pixel format of output image
	inExposureTime	Optional<int>	1 - +∞	NIL	Exposure time value
	inGain	Optional<float>		NIL	Gain value
	inFrameRate	Optional<float>	1.0 - ∞	NIL	Frame rate in FPS
	inAoi	Optional<const Box&>		NIL	Required fragment of image to stream
	inTriggerMode	MvIMPACTTriggerActivation::Type			Trigger mode
	inTriggerSource	MvIMPACTTriggerSource::Type			Source of trigger
	outImage	Image&			Captured frame

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install mvGenTL Acquire SDK software with camera dedicated drivers.

mvGenTL Acquire SDK software can be downloaded from the following website: www.matrix-vision.com. For Aurora Vision Studio 32-bit, mvGenTL Acquire x86 version is needed. For Aurora Vision Studio 64-bit, mvGenTL Acquire x86_x64 version of SDK is required.

Recommended mvGenTL Acquire SDK version for usage with Aurora Vision Studio is **2.49.0**.

Camera identification

When there is only one camera connected to the computer, field **inDeviceID** can be set to *Auto*. In such case, first available camera will be found and connected.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** can be set to:

- **serial number** - it should be specified on the device casing(e.g. *GX001559*),
- **product name** - it finds camera by name (might be the same for many cameras). The product name is a bit more specific than the family name, but less specific than the serial number(e.g. *mvBlueCOUGAR-X100wG*),
- **product number** - it finds camera by family name(e.g. *mvBlueCOUGAR*).

Camera parameters

Most of the parameters exposed by camera filters are optional, setting them to *Auto* leaves related parameter for automatic configuration by the camera driver.

To change other and more advanced camera parameters, use the wxPropView available with mvGenTL Acquire SDK. Refer to SDK documentation for information about parameters and how to save them into memory channels.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- mvGenTLAcquire_GetDigitalIOState – Returns the state of digital inputs or outputs from matrix vision cameras.
- mvGenTLAcquire_SetDigitalOutput – Sets matrix vision camera digital outputs.

mvGenTLAcquire_GrabImage_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a frame from MATRIX VISION cameras using mvGenTLAcquire SDK.

Syntax

```
bool avl::mvGenTLAcquire_GrabImage_WithTimeout
(
 mvGenTLAcquireState& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inTimeout,
 atl::Optional<avl::MVGenTLPixelFormat::Type> inPixelFormat,
 atl::Optional<int> inExposureTime,
 atl::Optional<float> inGain,
 atl::Optional<float> inFrameRate,
 atl::Optional<const avl::Box&> inAoi,
 avl::MvIMPACTTriggerActivation::Type inTriggerMode,
 avl::MvIMPACTTriggerSource::Type inTriggerSource,
 atl::Conditional<avl::Image>& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	mvGenTLAcquireState&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Device name or serial number of device
	inTimeout	int	10 - ∞	100	Maximum time to wait for frame in milliseconds
	inPixelFormat	Optional<MVGenTLPixelFormat::Type>		NIL	Pixel format of output image
	inExposureTime	Optional<int>	1 - + ∞	NIL	Exposure time value
	inGain	Optional<float>		NIL	Gain value
	inFrameRate	Optional<float>	1.0 - ∞	NIL	Frame rate in FPS
	inAoi	Optional<const Box&>		NIL	Required fragment of image to stream
	inTriggerMode	MvIMPACTTriggerActivation::Type			Trigger mode
	inTriggerSource	MvIMPACTTriggerSource::Type			Source of trigger
	outImage	Conditional<Image>&			Captured frame

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install mvGenTL Acquire SDK software with camera dedicated drivers.

mvGenTL Acquire SDK software can be downloaded from the following website: www.matrix-vision.com. For Aurora Vision Studio 32-bit, mvGenTL Acquire x86 version is needed. For Aurora Vision Studio 64-bit, mvGenTL Acquire x86_x64 version of SDK is required.

Recommended mvGenTL Acquire SDK version for usage with Aurora Vision Studio is **2.49.0**.

Camera identification

When there is only one camera connected to the computer, field **inDeviceID** can be set to *Auto*. In such case, first available camera will be found and connected.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** can be set to:

- **serial number** - it should be specified on the device casing(e.g. *GX001559*),
- **product name** - it finds camera by name (might be the same for many cameras). The product name is a bit more specific than the family name, but less specific than the serial number(e.g. *mvBlueCOUGAR-X100wG*),
- **product number** - it finds camera by family name(e.g. *mvBlueCOUGAR*).

Camera parameters

Most of the parameters exposed by camera filters are optional, setting them to *Auto* leaves related parameter for automatic configuration by the camera driver.

To change other and more advanced camera parameters, use the wxPropView available with mvGenTL Acquire SDK. Refer to SDK documentation for information about parameters and how to save them into memory channels.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- mvGenTLAcquire_GetDigitalIOState – Returns the state of digital inputs or outputs from matrix vision cameras.
- mvGenTLAcquire_SetDigitalOutput – Sets matrix vision camera digital outputs.

mvGenTLAcquire_SetDigitalOutput

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets matrix vision camera digital outputs.

Syntax

```
void avl::mvGenTLAcquire_SetDigitalOutput
(
 mvGenTLAcquireState& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inOutput,
 bool inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	mvGenTLAcquireState&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device name or serial number of device
	inOutput	int		Index of output line
	inValue	bool		Logic value of digital output

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install mvGenTL Acquire SDK software with camera dedicated drivers.

mvGenTL Acquire SDK software can be downloaded from the following website: www.matrix-vision.com. For Aurora Vision Studio 32-bit, mvGenTL Acquire x86 version is needed. For Aurora Vision Studio 64-bit, mvGenTL Acquire x86_x64 version of SDK is required.

Recommended mvGenTL Acquire SDK version for usage with Aurora Vision Studio is **2.49.0**.

Camera identification

When there is only one camera connected to the computer, field **inDeviceID** can be set to *Auto*. In such case, first available camera will be found and connected.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** can be set to:

- **serial number** - it should be specified on the device casing(e.g. *GX001559*),
- **product name** - it finds camera by name (might be the same for many cameras). The product name is a bit more specific than the family name, but less specific than the serial number(e.g. *mvBlueCOUGAR-X100wG*),
- **product number** - it finds camera by family name(e.g. *mvBlueCOUGAR*).

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- mvGenTLAcquire_GrabImage – Captures a frame from MATRIX VISION cameras using mvGenTLAcquire SDK.
- mvGenTLAcquire_GetDigitalIOState – Returns the state of digital inputs or outputs from matrix vision cameras.

174. Neousys

Table of content:

Neousys_GetDigitalInput_Multiple

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets values of all digital input channels at once on a Neousys computer.

Syntax

```
void avl::Neousys_GetDigitalInput_Multiple
(
 Neousys_State& ioState,
 bool& outValue0,
 bool& outValue1,
 bool& outValue2,
 bool& outValue3,
 bool& outValue4,
 bool& outValue5,
 bool& outValue6,
 bool& outValue7,
 bool& outValue8,
 bool& outValue9,
 bool& outValue10,
 bool& outValue11,
 bool& outValue12,
 bool& outValue13,
 bool& outValue14,
 bool& outValue15
)
```

Parameters

	Name	Type	Default	Description
	ioState	Neousys_State&		Object used to maintain state of the function.
	outValue0	bool&		Value to get from input channel 0
	outValue1	bool&		Value to get from input channel 1
	outValue2	bool&		Value to get from input channel 2
	outValue3	bool&		Value to get from input channel 3
	outValue4	bool&		Value to get from input channel 4
	outValue5	bool&		Value to get from input channel 5
	outValue6	bool&		Value to get from input channel 6
	outValue7	bool&		Value to get from input channel 7
	outValue8	bool&		Value to get from input channel 8
	outValue9	bool&		Value to get from input channel 9
	outValue10	bool&		Value to get from input channel 10
	outValue11	bool&		Value to get from input channel 11
	outValue12	bool&		Value to get from input channel 12
	outValue13	bool&		Value to get from input channel 13
	outValue14	bool&		Value to get from input channel 14
	outValue15	bool&		Value to get from input channel 15

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it may be required to install WDT_DIO drivers and SDK.

WDT_DIO can be downloaded from the following website: https://www.neousys-tech.com/en/support-service/download-resources.

Add DLL path to system environment variable may be required.

Recommended WDT_DIO version for Aurora Vision Studio usage is **2.3.1**.

Availability

Before you use this filter make sure how many digital input and output channels does your Neousys device provide.

Compatibility

Neousys Watchdog Timer and Digital I/O support is not provided when using 32-bit Aurora Vision on a 64-bit operating system. Please use 64-bit Aurora Vision in such a case.

See Also

- Neousys_SetDigitalOutput_Multiple – Sets values to all digital output channels at once on a Neousys computer.

Neousys_GetDigitalInput_MultipleAsArray

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets values of all digital input channels at once on a Neousys computer passed in an array form.

Syntax

```
void avl::Neousys_GetDigitalInput_MultipleAsArray
(
 Neousys_State& ioState,
 atl::Array<bool>& outValues
)
```

Parameters

	Name	Type	Default	Description
	ioState	Neousys_State&		Object used to maintain state of the function.
	outValues	Array<bool>&		Values to get from input channels (16 items)

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it may be required to install WDT_DIO drivers and SDK.

WDT_DIO can be downloaded from the following website: https://www.neousys-tech.com/en/support-service/download-resources.

Add DLL path to system environment variable may be required.

Recommended WDT_DIO version for Aurora Vision Studio usage is **2.3.1**.

Availability

Before you use this filter make sure how many digital input and output channels does your Neousys device provide.

Compatibility

Neousys Watchdog Timer and Digital I/O support is not provided when using 32-bit Aurora Vision on a 64-bit operating system. Please use 64-bit Aurora Vision in such a case.

See Also

- Neousys_SetDigitalOutput_MultipleAsArray – Sets values to all digital output channels at once on a Neousys computer passed in an array form.

Neousys_GetDigitalInput_Single

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets value of single digital input channel on a Neousys computer.

Syntax

```
void avl::Neousys_GetDigitalInput_Single
(
 Neousys_State& ioState,
 int inChannel,
 bool& outValue
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Neousys_State&			Object used to maintain state of the function.
	inChannel	int	0 - 15		Desired input channel identifier
	outValue	bool&			Value to get from desired input channel

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it may be required to install WDT_DIO drivers and SDK.

WDT_DIO can be downloaded from the following website: https://www.neousys-tech.com/en/support-service/download-resources.

Add DLL path to system environment variable may be required.

Recommended WDT_DIO version for Aurora Vision Studio usage is **2.3.1**.

Availability

Before you use this filter make sure how many digital input and output channels does your Neousys device provide.

Compatibility

Neousys Watchdog Timer and Digital I/O support is not provided when using 32-bit Aurora Vision on a 64-bit operating system. Please use 64-bit Aurora Vision in such a case.

See Also

- Neousys_SetDigitalOutput_Single – Sets value to single digital output channel on a Neousys computer.

Neousys_ResetWatchdogTimer

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Resets watchdog timer countdown on a Neousys computer.

Syntax

```
void avl::Neousys_ResetWatchdogTimer
(
 Neousys_State& ioState
)
```

Parameters

	Name	Type	Default	Description
	ioState	Neousys_State&		Object used to maintain state of the function.

Description

Sets watchdog timer back to the reference value in order to prevent it from reaching 0 and restarting the device.

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it may be required to install WDT_DIO drivers and SDK.

WDT_DIO can be downloaded from the following website: https://www.neousys-tech.com/en/support-service/download-resources.

Add DLL path to system environment variable may be required.

Recommended WDT_DIO version for Aurora Vision Studio usage is **2.3.1**.

Availability

Before you use this filter make sure that your Neousys device has a watchdog unit.

Initialization

When the watchdog unit is initialized it's default countdown value is set to 255 seconds.

Operation

When the program exits watchdog unit is always stopped and its parameters are set to default, so make sure to keep the program running when it encounters an error that should trigger the watchdog unit.

Compatibility

Neousys Watchdog Timer and Digital I/O support is not provided when using 32-bit Aurora Vision on a 64-bit operating system. Please use 64-bit Aurora Vision in such a case.

Errors

List of possible exceptions:

Error type	Description
IoError	Could not reset watchdog timer.

See Also

- Neousys_SetWatchdogTimer – Sets watchdog timer properties on a Neousys computer.
- Neousys_StartWatchdogTimer – Starts watchdog timer countdown on a Neousys computer.
- Neousys_StopWatchdogTimer – Stops watchdog timer countdown on a Neousys computer.

Neousys_SetDigitalOutput_Multiple

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets values to all digital output channels at once on a Neousys computer.

Syntax

```
void avl::Neousys_SetDigitalOutput_Multiple
(
 Neousys_State& ioState,
 bool inValue0,
 bool inValue1 = false,
 bool inValue2 = false,
 bool inValue3 = false,
 bool inValue4 = false,
 bool inValue5 = false,
 bool inValue6 = false,
 bool inValue7 = false,
 bool inValue8 = false,
 bool inValue9 = false,
 bool inValue10 = false,
 bool inValue11 = false,
 bool inValue12 = false,
 bool inValue13 = false,
 bool inValue14 = false,
 bool inValue15 = false
)
```

Parameters

	Name	Type	Default	Description
	ioState	Neousys_State&		Object used to maintain state of the function.
	inValue0	bool		Value to set to output channel 0
	inValue1	bool	false	Value to set to output channel 1
	inValue2	bool	false	Value to set to output channel 2
	inValue3	bool	false	Value to set to output channel 3
	inValue4	bool	false	Value to set to output channel 4
	inValue5	bool	false	Value to set to output channel 5
	inValue6	bool	false	Value to set to output channel 6
	inValue7	bool	false	Value to set to output channel 7
	inValue8	bool	false	Value to set to output channel 8
	inValue9	bool	false	Value to set to output channel 9
	inValue10	bool	false	Value to set to output channel 10
	inValue11	bool	false	Value to set to output channel 11
	inValue12	bool	false	Value to set to output channel 12
	inValue13	bool	false	Value to set to output channel 13
	inValue14	bool	false	Value to set to output channel 14
	inValue15	bool	false	Value to set to output channel 15

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it may be required to install WDT_DIO drivers and SDK.

WDT_DIO can be downloaded from the following website: https://www.neousys-tech.com/en/support-service/download-resources.

Add DLL path to system environment variable may be required.

Recommended WDT_DIO version for Aurora Vision Studio usage is **2.3.1**.

Availability

Before you use this filter make sure how many digital input and output channels does your Neousys device provide.

Compatibility

Neousys Watchdog Timer and Digital I/O support is not provided when using 32-bit Aurora Vision on a 64-bit operating system. Please use 64-bit Aurora Vision in such a case.

See Also

- Neousys_SetDigitalOutput_Multiple_Checked – Sets values to all digital output channels at once on a Neousys computer. Additionally, does a read-back of the values to make sure they're identical to the written values.
- Neousys_GetDigitalInput_Multiple – Gets values of all digital input channels at once on a Neousys computer.

Neousys_SetDigitalOutput_MultipleAsArray

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets values to all digital output channels at once on a Neousys computer passed in an array form.

Syntax

```
void avl::Neousys_SetDigitalOutput_MultipleAsArray
(
 Neousys_State& ioState,
 const atl::Array<bool>& inValues
)
```

Parameters

	Name	Type	Default	Description
	ioState	Neousys_State&		Object used to maintain state of the function.
	inValues	const Array<bool>&		Values to set to output channels; max 16

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it may be required to install WDT_DIO drivers and SDK.

WDT_DIO can be downloaded from the following website: https://www.neousys-tech.com/en/support-service/download-resources.

Add DLL path to system environment variable may be required.

Recommended WDT_DIO version for Aurora Vision Studio usage is **2.3.1**.

Availability

Before you use this filter make sure how many digital input and output channels does your Neousys device provide.

Compatibility

Neousys Watchdog Timer and Digital I/O support is not provided when using 32-bit Aurora Vision on a 64-bit operating system. Please use 64-bit Aurora Vision in such a case.

Errors

List of possible exceptions:

Error type	Description
DomainError	inValues array size too big in Neousys_SetDigitalOutput_MultipleAsArray.

See Also

- Neousys_SetDigitalOutput_MultipleAsArray_Checked – Sets values to all digital output channels at once on a Neousys computer passed in an array form. Additionally, does a read-back of the values to make sure they're identical to the written values.
- Neousys_GetDigitalInput_MultipleAsArray – Gets values of all digital input channels at once on a Neousys computer passed in an array form.

Neousys_SetDigitalOutput_MultipleAsArray_Checked

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets values to all digital output channels at once on a Neousys computer passed in an array form. Additionally, does a read-back of the values to make sure they're identical to the written values.

Syntax

```
void avl::Neousys_SetDigitalOutput_MultipleAsArray_Checked
(
 Neousys_State& ioState,
 const atl::Array<bool>& inValues
)
```

Parameters

	Name	Type	Default	Description
	ioState	Neousys_State&		Object used to maintain state of the function.
	inValues	const Array<bool>&		Values to set to output channels; max 8

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it may be required to install WDT_DIO drivers and SDK.

WDT_DIO can be downloaded from the following website: https://www.neousys-tech.com/en/support-service/download-resources.

Add DLL path to system environment variable may be required.

Recommended WDT_DIO version for Aurora Vision Studio usage is **2.3.1**.

Availability

Before you use this filter make sure how many digital input and output channels does your Neousys device provide.

Compatibility

Neousys Watchdog Timer and Digital I/O support is not provided when using 32-bit Aurora Vision on a 64-bit operating system. Please use 64-bit Aurora Vision in such a case.

Errors

List of possible exceptions:

Error type	Description
DomainError	inValues array size too big in Neousys_SetDigitalOutput_MultipleAsArray_Checked.

See Also

- Neousys_SetDigitalOutput_MultipleAsArray – Sets values to all digital output channels at once on a Neousys computer passed in an array form.
- Neousys_GetDigitalInput_MultipleAsArray – Gets values of all digital input channels at once on a Neousys computer passed in an array form.

Neousys_SetDigitalOutput_Multiple_Checked

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets values to all digital output channels at once on a Neousys computer. Additionally, does a read-back of the values to make sure they're identical to the written values.

Syntax

```
void avl::Neousys_SetDigitalOutput_Multiple_Checked
(
 Neousys_State& ioState,
 bool inValue0,
 bool inValue1 = false,
 bool inValue2 = false,
 bool inValue3 = false,
 bool inValue4 = false,
 bool inValue5 = false,
 bool inValue6 = false,
 bool inValue7 = false,
 bool inValue8 = false,
 bool inValue9 = false,
 bool inValue10 = false,
 bool inValue11 = false,
 bool inValue12 = false,
 bool inValue13 = false,
 bool inValue14 = false,
 bool inValue15 = false
)
```

Parameters

Name	Type	Default	Description
ioState	Neousys_State&		Object used to maintain state of the function.
inValue0	bool		Value to set to output channel 0
inValue1	bool	false	Value to set to output channel 1
inValue2	bool	false	Value to set to output channel 2
inValue3	bool	false	Value to set to output channel 3
inValue4	bool	false	Value to set to output channel 4
inValue5	bool	false	Value to set to output channel 5
inValue6	bool	false	Value to set to output channel 6
inValue7	bool	false	Value to set to output channel 7
inValue8	bool	false	Value to set to output channel 8
inValue9	bool	false	Value to set to output channel 9
inValue10	bool	false	Value to set to output channel 10
inValue11	bool	false	Value to set to output channel 11
inValue12	bool	false	Value to set to output channel 12
inValue13	bool	false	Value to set to output channel 13
inValue14	bool	false	Value to set to output channel 14
inValue15	bool	false	Value to set to output channel 15

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it may be required to install WDT_DIO drivers and SDK.

WDT_DIO can be downloaded from the following website: https://www.neousys-tech.com/en/support-service/download-resources.

Add DLL path to system environment variable may be required.

Recommended WDT_DIO version for Aurora Vision Studio usage is **2.3.1**.

Availability

Before you use this filter make sure how many digital input and output channels does your Neousys device provide.

Compatibility

Neousys Watchdog Timer and Digital I/O support is not provided when using 32-bit Aurora Vision on a 64-bit operating system. Please use 64-bit Aurora Vision in such a case.

See Also

- Neousys_SetDigitalOutput_Multiple – Sets values to all digital output channels at once on a Neousys computer.
- Neousys_GetDigitalInput_Multiple – Gets values of all digital input channels at once on a Neousys computer.

Neousys_SetDigitalOutput_Single

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets value to single digital output channel on a Neousys computer.

Syntax

```
void avl::Neousys_SetDigitalOutput_Single
(
 Neousys_State& ioState,
 int inChannel,
 bool inValue
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Neousys_State&			Object used to maintain state of the function.
	inChannel	int	0 - 15		Desired output channel identifier
	inValue	bool			Value to set to desired output channel

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it may be required to install WDT_DIO drivers and SDK.

WDT_DIO can be downloaded from the following website: https://www.neousys-tech.com/en/support-service/download-resources.

Add DLL path to system environment variable may be required.

Recommended WDT_DIO version for Aurora Vision Studio usage is **2.3.1**.

Availability

Before you use this filter make sure how many digital input and output channels does your Neousys device provide.

Compatibility

Neousys Watchdog Timer and Digital I/O support is not provided when using 32-bit Aurora Vision on a 64-bit operating system. Please use 64-bit Aurora Vision in such a case.

See Also

- Neousys_SetDigitalOutput_Single_Checked – Sets value to single digital output channel on a Neousys computer. Additionally, does a read-back of the value to make sure it's identical to the written value.
- Neousys_GetDigitalInput_Single – Gets value of single digital input channel on a Neousys computer.

Neousys_SetDigitalOutput_Single_Checked

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets value to single digital output channel on a Neousys computer. Additionally, does a read-back of the value to make sure it's identical to the written value.

Syntax

```
void avl::Neousys_SetDigitalOutput_Single_Checked
(
 Neousys_State& ioState,
 int inChannel,
 bool inValue
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Neousys_State&			Object used to maintain state of the function.
	inChannel	int	0 - 15		Desired output channel identifier
	inValue	bool			Value to set to desired output channel

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it may be required to install WDT_DIO drivers and SDK.

WDT_DIO can be downloaded from the following website: https://www.neousys-tech.com/en/support-service/download-resources.

Add DLL path to system environment variable may be required.

Recommended WDT_DIO version for Aurora Vision Studio usage is **2.3.1**.

Availability

Before you use this filter make sure how many digital input and output channels does your Neousys device provide.

Compatibility

Neousys Watchdog Timer and Digital I/O support is not provided when using 32-bit Aurora Vision on a 64-bit operating system. Please use 64-bit Aurora Vision in such a case.

See Also

- Neousys_SetDigitalOutput_Single – Sets value to single digital output channel on a Neousys computer.
- Neousys_GetDigitalInput_Single – Gets value of single digital input channel on a Neousys computer.

Neousys_SetWatchdogTimer

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets watchdog timer properties on a Neousys computer.

Syntax

```
void avl::Neousys_SetWatchdogTimer
(
 Neousys_State& ioState,
 int inTicks,
 avl::NeousysWatchdogTimerUnit::Type inUnit
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Neousys_State&			Object used to maintain state of the function.
	inTicks	int	1 - 65535	255	Number of ticks to set the watchdog timer to
	inUnit	NeousysWatchdogTimerUnit::Type		Second	Unit of the ticks

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it may be required to install WDT_DIO drivers and SDK.

WDT_DIO can be downloaded from the following website: https://www.neousys-tech.com/en/support-service/download-resources.

Add DLL path to system environment variable may be required.

Recommended WDT_DIO version for Aurora Vision Studio usage is **2.3.1**.

Availability

Before you use this filter make sure that your Neousys device has a watchdog unit.

Initialization

When the watchdog unit is initialized it's default countdown value is set to 255 seconds.

Operation

When the program exits watchdog unit is always stopped and its parameters are set to default, so make sure to keep the program running when it encounters an error that should trigger the watchdog unit.

Limits

Depending on the device there might be limits on the **inTicks** value depending on the **inUnit** value. Please refer to your Neousys device manual when in doubt.

Compatibility

Neousys Watchdog Timer and Digital I/O support is not provided when using 32-bit Aurora Vision on a 64-bit operating system. Please use 64-bit Aurora Vision in such a case.

Errors

List of possible exceptions:

Error type	Description
IoError	Could not set watchdog timer.

See Also

- Neousys_StartWatchdogTimer – Starts watchdog timer countdown on a Neousys computer.
- Neousys_ResetWatchdogTimer – Resets watchdog timer countdown on a Neousys computer.
- Neousys_StopWatchdogTimer – Stops watchdog timer countdown on a Neousys computer.

Neousys_StartWatchdogTimer

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Starts watchdog timer countdown on a Neousys computer.

Syntax

```
void avl::Neousys_StartWatchdogTimer
(
 Neousys_State& ioState
)
```

Parameters

	Name	Type	Default	Description
	ioState	Neousys_State&		Object used to maintain state of the function.

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it may be required to install WDT_DIO drivers and SDK.

WDT_DIO can be downloaded from the following website: https://www.neousys-tech.com/en/support-service/download-resources.

Add DLL path to system environment variable may be required.

Recommended WDT_DIO version for Aurora Vision Studio usage is **2.3.1**.

Availability

Before you use this filter make sure that your Neousys device has a watchdog unit.

Initialization

When the watchdog unit is initialized it's default countdown value is set to 255 seconds.

Operation

When the program exits watchdog unit is always stopped and its parameters are set to default, so make sure to keep the program running when it encounters an error that should trigger the watchdog unit.

Compatibility

Neousys Watchdog Timer and Digital I/O support is not provided when using 32-bit Aurora Vision on a 64-bit operating system. Please use 64-bit Aurora Vision in such a case.

Errors

List of possible exceptions:

Error type	Description
IoError	Could not start watchdog timer.

See Also

- Neousys_SetWatchdogTimer – Sets watchdog timer properties on a Neousys computer.
- Neousys_ResetWatchdogTimer – Resets watchdog timer countdown on a Neousys computer.
- Neousys_StopWatchdogTimer – Stops watchdog timer countdown on a Neousys computer.

Neousys_StopWatchdogTimer

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Stops watchdog timer countdown on a Neousys computer.

Syntax

```
void avl::Neousys_StopWatchdogTimer
(
 Neousys_State& ioState
)
```

Parameters

	Name	Type	Default	Description
	ioState	Neousys_State&		Object used to maintain state of the function.

Remarks

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it may be required to install WDT_DIO drivers and SDK.

WDT_DIO can be downloaded from the following website: https://www.neousys-tech.com/en/support-service/download-resources.

Add DLL path to system environment variable may be required.

Recommended WDT_DIO version for Aurora Vision Studio usage is **2.3.1**.

Availability

Before you use this filter make sure that your Neousys device has a watchdog unit.

Initialization

When the watchdog unit is initialized it's default countdown value is set to 255 seconds.

Operation

When the program exits watchdog unit is always stopped and its parameters are set to default, so make sure to keep the program running when it encounters an error that should trigger the watchdog unit.

Compatibility

Neousys Watchdog Timer and Digital I/O support is not provided when using 32-bit Aurora Vision on a 64-bit operating system. Please use 64-bit Aurora Vision in such a case.

Errors

List of possible exceptions:

Error type	Description
IoError	Could not stop watchdog timer.

See Also

- Neousys_SetWatchdogTimer – Sets watchdog timer properties on a Neousys computer.
- Neousys_StartWatchdogTimer – Starts watchdog timer countdown on a Neousys computer.
- Neousys_ResetWatchdogTimer – Resets watchdog timer countdown on a Neousys computer.

175. Nodka

Table of content:

Nodka_GetGeneralParam

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets general parameters of the controller.

Syntax

```
void avl::Nodka_GetGeneralParam
(
 Nodka_State& ioState,
 atl::Optional<int> inServerPort,
 atl::Optional<const atl::File&> inDeviceConfigPath,
 atl::Optional<int> inDeviceLightControllerPort,
 int inDevId,
 avl::NodkaGeneralParameter::Type inParamId,
 int inParamLen,
 int inTimeout,
 int& outParamValue
)
```

Parameters

	Name	Type	Range	Default	Description
⇆	ioState	Nodka_State&			Object used to maintain state of the function.
→	inServerPort	Optional<int>	1 - 65535	NIL	Port of NodkaServer service.
→	inDeviceConfigPath	Optional<const File&>		NIL	Path to nkio_config.ini file with I/O configuration.
→	inDeviceLightControllerPort	Optional<int>	1 - ∞	NIL	COM port used by the light controller (if supported by the device).
→	inDevId	int	0 - 255		ID of the device (0x01 for the light controller).
→	inParamId	NodkaGeneralParameter::Type			ID of the parameter.
→	inParamLen	int	0 - 255	1	Length of the parameter (refer to NKIO_API user manual).
→	inTimeout	int	0 - ∞	100	Timeout for the callback function.
←	outParamValue	int&			Value of the parameter.

Remarks

This filter can be used to get the value of general parameters of the controller (refer to NKIO_API user manual for more information).

I/O device driver software

This filter is intended to cooperate with a Nodka device using its vendor SDK and Nodka_Server.exe program. To be able to connect to the device, it is required to install Nodka NKDIO SDK.

Add DLL path to system environment variable may be required.

Recommended Nodka NKDIO SDK version for Aurora Vision Studio usage is **v4.1.9**. The implementation of all SDK functionality is done in Nodka_Server.exe program. The filters in Aurora Vision Studio work as a client which communicates with the server and they don't use the SDK directly. Only localhost connections are allowed.

It is mandatory to run the Nodka_Server.exe program **with administrative privileges** and keep it running during the execution of this and other Nodka filters.

For the correct operation of the program, it is required to copy the WinRing0.sys and WinRing0x64.sys files from the SDK directory to the directory where the Nodka_Server.exe file is located. These files are placed in C:\NODKA\NKDIO_SDK\Lib\x86 and C:\NODKA\NKDIO_SDK\Lib\x64 directories by default. Choose those that match your operating system version. Adding the directory to the PATH system variable will likely not work.

To run the program, open the Command Prompt as administrator, navigate to the directory where Nodka_Server.exe file is situated and then run the command `Nodka_Server.exe --port PORT` where 'PORT' is a TCP port on which the server will be listening (an integer value from the range 1-65535). Aurora Vision uses port 44100 by default so it is recommended to choose this one.

The Nodka_Server.exe program can be found in Aurora Vision installation path.

Some ports may operate in reverse logic. For details, refer to the electronic input and output diagram documentation of the device.

Device identification

inServerPort field is used to select a TCP port on which the Nodka_Server.exe is listening. Setting this parameter to 'Auto' will select port 44100 which is the default value. ServerPort can be set to:

- **TCP port** - a TCP port on which the Nodka_Server.exe is listening.

inDeviceConfigPath field is used to select a path to the configuration file for a specific Nodka device model. This field cannot be empty for the first Nodka filter used. For each of the following filters with the same value of **inServerPort**, the value of this field can be set to 'Auto' which would result in using the previously specified configuration file. A new configuration file must be provided for each distinct value of **inServerPort**. The Nodka NKDIO SDK installer puts the configuration files in C:\NODKA\NKDIO_SDK\ConfigFile directory by default. The SDK does not check if the specified configuration file is correct, so any errors in the configuration file may be ignored by the SDK. DeviceConfigPath can be set to:

- **File path** - a path to the configuration file for a specific Nodka device model.

inDeviceLightControllerPort field is used to select the COM port of the on-board light controller. Please note that not all Nodka devices have an on-board light controller. For more information refer to the documentation of the specific model. This field is optional and for devices without the light controller should be set to 'Auto'. DeviceLightControllerPort can be set to:

- **COM port** - a COM port of the on-board light controller.

See Also

- Nodka_SetGeneralParam – Sets general parameters of the controller.

Nodka_GetPwmParams

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameters of the light controller.

Syntax

```
void avl::Nodka_GetPwmParams
(
 Nodka_State& ioState,
 atl::Optional<int> inServerPort,
 atl::Optional<const atl::File&> inDeviceConfigPath,
 atl::Optional<int> inDeviceLightControllerPort,
 int inDevId,
 avl::NodkaLightControlChannel::Type inChIdx,
 int inTimeout,
 avl::NodkaPwmMode::Type& outPwmMode,
 int& outPwmValue,
 int& outPwmHoldingTime,
 int& outPwmOnOff
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Nodka_State&			Object used to maintain state of the function.
	inServerPort	Optional<int>	1 - 65535	NIL	Port of NodkaServer service.
	inDeviceConfigPath	Optional<const File&>		NIL	Path to nkio_config.ini file with I/O configuration.
	inDeviceLightControllerPort	Optional<int>	1 - ∞	NIL	COM port used by the light controller (if supported by the device).
	inDevId	int	0 - 255		ID of the device (0x01 for the light controller).
	inChIdx	NodkaLightControlChannel::Type			Light controller channel ID.
	inTimeout	int	0 - ∞	100	Timeout for the callback function.
	outPwmMode	NodkaPwmMode::Type&			Trigger mode of the light controller.
	outPwmValue	int&			Brightness level of the light controller.
	outPwmHoldingTime	int&			Duration of the 'on' state of the light controller (see also: Nodka_SetHoldingTimeUnit).
	outPwmOnOff	int&			Parameter to turn the light on or off.

Remarks

This filter can be used to get the parameters of the specified light control channel.

I/O device driver software

This filter is intended to cooperate with a Nodka device using its vendor SDK and Nodka_Server.exe program. To be able to connect to the device, it is required to install Nodka NKDIO SDK.

Add DLL path to system environment variable may be required.

Recommended Nodka NKDIO SDK version for Aurora Vision Studio usage is **v4.1.9**. The implementation of all SDK functionality is done in Nodka_Server.exe program. The filters in Aurora Vision Studio work as a client which communicates with the server and they don't use the SDK directly. Only localhost connections are allowed.

It is mandatory to run the Nodka_Server.exe program **with administrative privileges** and keep it running during the execution of this and other Nodka filters.

For the correct operation of the program, it is required to copy the WinRing0.sys and WinRing0x64.sys files from the SDK directory to the directory where the Nodka_Server.exe file is located. These files are placed in C:\NODKA\NKDIO_SDK\Lib\x86 and C:\NODKA\NKDIO_SDK\Lib\x64 directories by default. Choose those that match your operating system version. Adding the directory to the PATH system variable will likely not work.

To run the program, open the Command Prompt as administrator, navigate to the directory where Nodka_Server.exe file is situated and then run the command `Nodka_Server.exe --port PORT` where 'PORT' is a TCP port on which the server will be listening (an integer value from the range 1-65535). Aurora Vision uses port 44100 by default so it is recommended to choose this one.

The Nodka_Server.exe program can be found in Aurora Vision installation path.

Some ports may operate in reverse logic. For details, refer to the electronic input and output diagram documentation of the device.

Device identification

inServerPort field is used to select a TCP port on which the Nodka_Server.exe is listening. Setting this parameter to 'Auto' will select port 44100 which is the default value. ServerPort can be set to:

- **TCP port** - a TCP port on which the Nodka_Server.exe is listening.

inDeviceConfigPath field is used to select a path to the configuration file for a specific Nodka device model. This field cannot be empty for the first Nodka filter used. For each of the following filters with the same value of **inServerPort**, the value of this field can be set to 'Auto' which would result in using the previously specified configuration file. A new configuration file must be provided for each distinct value of **inServerPort**. The Nodka NKDIO SDK installer puts the configuration files in C:\NODKA\NKDIO_SDK\ConfigFile directory by default. The SDK does not check if the specified configuration file is correct, so any errors in the configuration file may be ignored by the SDK. DeviceConfigPath can be set to:

- **File path** - a path to the configuration file for a specific Nodka device model.

inDeviceLightControllerPort field is used to select the COM port of the on-board light controller. Please note that not all Nodka devices have an on-board light controller. For more information refer to the documentation of the specific model. This field is optional and for devices without the light controller should be set to 'Auto'. DeviceLightControllerPort can be set to:

- **COM port** - a COM port of the on-board light controller.

See Also

- Nodka_SetPwmParams – Sets parameters of the light controller.

Nodka_GetVersionInfo

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets information about the hardware and the firmware of the device.

Syntax

```
void avl::Nodka_GetVersionInfo
(
 Nodka_State& ioState,
 atl::Optional<int> inServerPort,
 atl::Optional<const atl::File&> inDeviceConfigPath,
 atl::Optional<int> inDeviceLightControllerPort,
 int inDevId,
 int inTimeout,
 int& outHardwareMajorVer,
 int& outHardwareMinorVer,
 int& outHardwareRevVer,
 int& outFirmwareMajorVer,
 int& outFirmwareMinorVer,
 int& outFirmwareRevVer
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Nodka_State&			Object used to maintain state of the function.
	inServerPort	Optional<int>	1 - 65535	NIL	Port of NodkaServer service.
	inDeviceConfigPath	Optional<const File&>		NIL	Path to nkio_config.ini file with I/O configuration.
	inDeviceLightControllerPort	Optional<int>	1 - ∞	NIL	COM port used by the light controller (if supported by the device).
	inDevId	int	0 - 255		ID of the device (0x01 for the light controller).
	inTimeout	int	0 - ∞	100	Timeout for the callback function.
	outHardwareMajorVer	int&			Device hardware major version.
	outHardwareMinorVer	int&			Device hardware minor version.
	outHardwareRevVer	int&			Device hardware revision version.
	outFirmwareMajorVer	int&			Device firmware major version.
	outFirmwareMinorVer	int&			Device firmware minor version.
	outFirmwareRevVer	int&			Device firmware revision version.

Remarks

This filter can be used to get the information about the hardware and the firmware version of the device.

I/O device driver software

This filter is intended to cooperate with a Nodka device using its vendor SDK and Nodka_Server.exe program. To be able to connect to the device, it is required to install Nodka NKDIO SDK.

Add DLL path to system environment variable may be required.

Recommended Nodka NKDIO SDK version for Aurora Vision Studio usage is **v4.1.9**. The implementation of all SDK functionality is done in Nodka_Server.exe program. The filters in Aurora Vision Studio work as a client which communicates with the server and they don't use the SDK directly. Only localhost connections are allowed.

It is mandatory to run the Nodka_Server.exe program **with administrative privileges** and keep it running during the execution of this and other Nodka filters.

For the correct operation of the program, it is required to copy the WinRing0.sys and WinRing0x64.sys files from the SDK directory to the directory where the Nodka_Server.exe file is located. These files are placed in C:\NODKA\NKDIO_SDK\Lib\x86 and C:\NODKA\NKDIO_SDK\Lib\x64 directories by default. Choose those that match your operating system version. Adding the directory to the PATH system variable will likely not work.

To run the program, open the Command Prompt as administrator, navigate to the directory where Nodka_Server.exe file is situated and then run the command `Nodka_Server.exe --port PORT` where 'PORT' is a TCP port on which the server will be listening (an integer value from the range 1-65535). Aurora Vision uses port 44100 by default so it is recommended to choose this one.

The Nodka_Server.exe program can be found in Aurora Vision installation path.

Some ports may operate in reverse logic. For details, refer to the electronic input and output diagram documentation of the device.

Device identification

inServerPort field is used to select a TCP port on which the Nodka_Server.exe is listening. Setting this parameter to 'Auto' will select port 44100 which is the default value. ServerPort can be set to:

- **TCP port** - a TCP port on which the Nodka_Server.exe is listening.

inDeviceConfigPath field is used to select a path to the configuration file for a specific Nodka device model. This field cannot be empty for the first Nodka filter used. For each of the following filters with the same value of **inServerPort**, the value of this field can be set to 'Auto' which would result in using the previously specified configuration file. A new configuration file must be provided for each distinct value of **inServerPort**. The Nodka NKDIO SDK installer puts the configuration files in C:\NODKA\NKDIO_SDK\ConfigFile directory by default. The SDK does not check if the specified configuration file is correct, so any errors in the configuration file may be ignored by the SDK. DeviceConfigPath can be set to:

- **File path** - a path to the configuration file for a specific Nodka device model.

inDeviceLightControllerPort field is used to select the COM port of the on-board light controller. Please note that not all Nodka devices have an on-board light controller. For more information refer to the documentation of the specific model. This field is optional and for devices without the light controller should be set to 'Auto'. DeviceLightControllerPort can be set to:

- **COM port** - a COM port of the on-board light controller.

Nodka_Initialize

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Initializes Nodka NKIO_API SDK.

Syntax

```
void avl::Nodka_Initialize
(
 Nodka_State& ioState,
 atl::Optional<int> inServerPort,
 const atl::File& inDeviceConfigPath,
 atl::Optional<int> inDeviceLightControllerPort
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Nodka_State&			Object used to maintain state of the function.
	inServerPort	Optional<int>	1 - 65535	44100	Port of NodkaServer service.
	inDeviceConfigPath	const File&			Path to nkio_config.ini file with I/O configuration.
	inDeviceLightControllerPort	Optional<int>	1 - ∞	NIL	COM port used by the light controller (if supported by the device).

Remarks

This filter can be used to explicitly initialize device parameters before using other Nodka filters.

I/O device driver software

This filter is intended to cooperate with a Nodka device using its vendor SDK and Nodka_Server.exe program. To be able to connect to the device, it is required to install Nodka NKDIO SDK.

Add DLL path to system environment variable may be required.

Recommended Nodka NKDIO SDK version for Aurora Vision Studio usage is **v4.1.9**. The implementation of all SDK functionality is done in Nodka_Server.exe program. The filters in Aurora Vision Studio work as a client which communicates with the server and they don't use the SDK directly. Only localhost connections are allowed.

It is mandatory to run the Nodka_Server.exe program **with administrative privileges** and keep it running during the execution of this and other Nodka filters.

For the correct operation of the program, it is required to copy the WinRing0.sys and WinRing0x64.sys files from the SDK directory to the directory where the Nodka_Server.exe file is located. These files are placed in C:\NODKA\NKDIO_SDK\Lib\x86 and C:\NODKA\NKDIO_SDK\Lib\x64 directories by default. Choose those that match your operating system version. Adding the directory to the PATH system variable will likely not work.

To run the program, open the Command Prompt as administrator, navigate to the directory where Nodka_Server.exe file is situated and then run the command `Nodka_Server.exe --port PORT` where 'PORT' is a TCP port on which the server will be listening (an integer value from the range 1-65535). Aurora Vision uses port 44100 by default so it is recommended to choose this one.

The Nodka_Server.exe program can be found in Aurora Vision installation path.

Some ports may operate in reverse logic. For details, refer to the electronic input and output diagram documentation of the device.

Device identification

inServerPort field is used to select a TCP port on which the Nodka_Server.exe is listening. Setting this parameter to 'Auto' will select port 44100 which is the default value. ServerPort can be set to:

- **TCP port** - a TCP port on which the Nodka_Server.exe is listening.

inDeviceConfigPath field is used to select a path to the configuration file for a specific Nodka device model. This field cannot be empty for the first Nodka filter used. For each of the following filters with the same value of **inServerPort**, the value of this field can be set to 'Auto' which would result in using the previously specified configuration file. A new configuration file must be provided for each distinct value of **inServerPort**. The Nodka NKDIO SDK installer puts the configuration files in C:\NODKA\NKDIO_SDK\ConfigFile directory by default. The SDK does not check if the specified configuration file is correct, so any errors in the configuration file may be ignored by the SDK. DeviceConfigPath can be set to:

- **File path** - a path to the configuration file for a specific Nodka device model.

inDeviceLightControllerPort field is used to select the COM port of the on-board light controller. Please note that not all Nodka devices have an on-board light controller. For more information refer to the documentation of the specific model. This field is optional and for devices without the light controller should be set to 'Auto'. DeviceLightControllerPort can be set to:

- **COM port** - a COM port of the on-board light controller.

Nodka_ReadInputLevel_Channel

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads a single input channel.

Syntax

```
void avl::Nodka_ReadInputLevel_Channel
(
 Nodka_State& ioState,
 atl::Optional<int> inServerPort,
 atl::Optional<const atl::File&> inDeviceConfigPath,
 atl::Optional<int> inDeviceLightControllerPort,
 int inPortIndex,
 int inChannelIndex,
 bool& outLevel
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Nodka_State&			Object used to maintain state of the function.
	inServerPort	Optional<int>	1 - 65535	NIL	Port of NodkaServer service.
	inDeviceConfigPath	Optional<const File&>		NIL	Path to nkio_config.ini file with I/O configuration.
	inDeviceLightControllerPort	Optional<int>	1 - ∞	NIL	COM port used by the light controller (if supported by the device).
	inPortIndex	int	0 - 1		Input port number.
	inChannelIndex	int	0 - 7		Input port channel.
	outLevel	bool&			Input channel level.

Remarks

This filter allows to get the input level information of a single input channel of the device.

I/O device driver software

This filter is intended to cooperate with a Nodka device using its vendor SDK and Nodka_Server.exe program. To be able to connect to the device, it is required to install Nodka NKDIO SDK.

Add DLL path to system environment variable may be required.

Recommended Nodka NKDIO SDK version for Aurora Vision Studio usage is **v4.1.9**. The implementation of all SDK functionality is done in Nodka_Server.exe program. The filters in Aurora Vision Studio work as a client which communicates with the server and they don't use the SDK directly. Only localhost connections are allowed.

It is mandatory to run the Nodka_Server.exe program **with administrative privileges** and keep it running during the execution of this and other Nodka filters.

For the correct operation of the program, it is required to copy the WinRing0.sys and WinRing0x64.sys files from the SDK directory to the directory where the Nodka_Server.exe file is located. These files are placed in C:\NODKA\NKDIO_SDK\Lib\x86 and C:\NODKA\NKDIO_SDK\Lib\x64 directories by default. Choose those that match your operating system version. Adding the directory to the PATH system variable will likely not work.

To run the program, open the Command Prompt as administrator, navigate to the directory where Nodka_Server.exe file is situated and then run the command `Nodka_Server.exe --port PORT` where 'PORT' is a TCP port on which the server will be listening (an integer value from the range 1-65535). Aurora Vision uses port 44100 by default so it is recommended to choose this one.

The Nodka_Server.exe program can be found in Aurora Vision installation path.

Some ports may operate in reverse logic. For details, refer to the electronic input and output diagram documentation of the device.

Device identification

inServerPort field is used to select a TCP port on which the Nodka_Server.exe is listening. Setting this parameter to 'Auto' will select port 44100 which is the default value. ServerPort can be set to:

- **TCP port** - a TCP port on which the Nodka_Server.exe is listening.

inDeviceConfigPath field is used to select a path to the configuration file for a specific Nodka device model. This field cannot be empty for the first Nodka filter used. For each of the following filters with the same value of **inServerPort**, the value of this field can be set to 'Auto' which would result in using the previously specified configuration file. A new configuration file must be provided for each distinct value of **inServerPort**. The Nodka NKDIO SDK installer puts the configuration files in C:\NODKA\NKDIO_SDK\ConfigFile directory by default. The SDK does not check if the specified configuration file is correct, so any errors in the configuration file may be ignored by the SDK. DeviceConfigPath can be set to:

- **File path** - a path to the configuration file for a specific Nodka device model.

inDeviceLightControllerPort field is used to select the COM port of the on-board light controller. Please note that not all Nodka devices have an on-board light controller. For more information refer to the documentation of the specific model. This field is optional and for devices without the light controller should be set to 'Auto'. DeviceLightControllerPort can be set to:

- **COM port** - a COM port of the on-board light controller.

See Also

- Nodka_ReadInputLevel_Port – Reads an entire input port.

Nodka_ReadInputLevel_Port

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads an entire input port.

Syntax

```
void avl::Nodka_ReadInputLevel_Port
(
 Nodka_State& ioState,
 atl::Optional<int> inServerPort,
 atl::Optional<const atl::File&> inDeviceConfigPath,
 atl::Optional<int> inDeviceLightControllerPort,
 int inPortIndex,
 atl::Array<bool>& outLevels
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Nodka_State&			Object used to maintain state of the function.
	inServerPort	Optional<int>	1 - 65535	NIL	Port of NodkaServer service.
	inDeviceConfigPath	Optional<const File&>		NIL	Path to nkio_config.ini file with I/O configuration.
	inDeviceLightControllerPort	Optional<int>	1 - ∞	NIL	COM port used by the light controller (if supported by the device).
	inPortIndex	int	0 - 1		Input port number.
	outLevels	Array<bool>&			Input port levels.

Remarks

This filter allows to get the input level information of a specific input port of the device (8 input channels).

I/O device driver software

This filter is intended to cooperate with a Nodka device using its vendor SDK and Nodka_Server.exe program. To be able to connect to the device, it is required to install Nodka NKDIO SDK.

Add DLL path to system environment variable may be required.

Recommended Nodka NKDIO SDK version for Aurora Vision Studio usage is **v4.1.9**. The implementation of all SDK functionality is done in Nodka_Server.exe program. The filters in Aurora Vision Studio work as a client which communicates with the server and they don't use the SDK directly. Only localhost connections are allowed.

It is mandatory to run the Nodka_Server.exe program **with administrative privileges** and keep it running during the execution of this and other Nodka filters.

For the correct operation of the program, it is required to copy the WinRing0.sys and WinRing0x64.sys files from the SDK directory to the directory where the Nodka_Server.exe file is located. These files are placed in C:\NODKA\NKDIO_SDK\Lib\x86 and C:\NODKA\NKDIO_SDK\Lib\x64 directories by default. Choose those that match your operating system version. Adding the directory to the PATH system variable will likely not work.

To run the program, open the Command Prompt as administrator, navigate to the directory where Nodka_Server.exe file is situated and then run the command `Nodka_Server.exe --port PORT` where 'PORT' is a TCP port on which the server will be listening (an integer value from the range 1-65535). Aurora Vision uses port 44100 by default so it is recommended to choose this one.

The Nodka_Server.exe program can be found in Aurora Vision installation path.

Some ports may operate in reverse logic. For details, refer to the electronic input and output diagram documentation of the device.

Device identification

inServerPort field is used to select a TCP port on which the Nodka_Server.exe is listening. Setting this parameter to 'Auto' will select port 44100 which is the default value. ServerPort can be set to:

- **TCP port** - a TCP port on which the Nodka_Server.exe is listening.

inDeviceConfigPath field is used to select a path to the configuration file for a specific Nodka device model. This field cannot be empty for the first Nodka filter used. For each of the following filters with the same value of **inServerPort**, the value of this field can be set to 'Auto' which would result in using the previously specified configuration file. A new configuration file must be provided for each distinct value of **inServerPort**. The Nodka NKDIO SDK installer puts the configuration files in C:\NODKA\NKDIO_SDK\ConfigFile directory by default. The SDK does not check if the specified configuration file is correct, so any errors in the configuration file may be ignored by the SDK. DeviceConfigPath can be set to:

- **File path** - a path to the configuration file for a specific Nodka device model.

inDeviceLightControllerPort field is used to select the COM port of the on-board light controller. Please note that not all Nodka devices have an on-board light controller. For more information refer to the documentation of the specific model. This field is optional and for devices without the light controller should be set to 'Auto'. DeviceLightControllerPort can be set to:

- **COM port** - a COM port of the on-board light controller.

See Also

- Nodka_ReadInputLevel_Channel – Reads a single input channel.

Nodka_ReadOutputLevel_Channel

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads a single output channel.

Syntax

```
void avl::Nodka_ReadOutputLevel_Channel
(
 Nodka_State& ioState,
 atl::Optional<int> inServerPort,
 atl::Optional<const atl::File&> inDeviceConfigPath,
 atl::Optional<int> inDeviceLightControllerPort,
 int inPortIndex,
 int inChannelIndex,
 bool& outLevel
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Nodka_State&			Object used to maintain state of the function.
	inServerPort	Optional<int>	1 - 65535	NIL	Port of NodkaServer service.
	inDeviceConfigPath	Optional<const File&>		NIL	Path to nkio_config.ini file with I/O configuration.
	inDeviceLightControllerPort	Optional<int>	1 - ∞	NIL	COM port used by the light controller (if supported by the device).
	inPortIndex	int	0 - 1		Output port number.
	inChannelIndex	int	0 - 7		Output port channel.
	outLevel	bool&			Output channel level.

Remarks

This filter allows to get the output level information of a single output channel of the device.

I/O device driver software

This filter is intended to cooperate with a Nodka device using its vendor SDK and Nodka_Server.exe program. To be able to connect to the device, it is required to install Nodka NKDIO SDK.

Add DLL path to system environment variable may be required.

Recommended Nodka NKDIO SDK version for Aurora Vision Studio usage is **v4.1.9**. The implementation of all SDK functionality is done in Nodka_Server.exe program. The filters in Aurora Vision Studio work as a client which communicates with the server and they don't use the SDK directly. Only localhost connections are allowed.

It is mandatory to run the Nodka_Server.exe program **with administrative privileges** and keep it running during the execution of this and other Nodka filters.

For the correct operation of the program, it is required to copy the WinRing0.sys and WinRing0x64.sys files from the SDK directory to the directory where the Nodka_Server.exe file is located. These files are placed in C:\NODKA\NKDIO_SDK\Lib\x86 and C:\NODKA\NKDIO_SDK\Lib\x64 directories by default. Choose those that match your operating system version. Adding the directory to the PATH system variable will likely not work.

To run the program, open the Command Prompt as administrator, navigate to the directory where Nodka_Server.exe file is situated and then run the command `Nodka_Server.exe --port PORT` where 'PORT' is a TCP port on which the server will be listening (an integer value from the range 1-65535). Aurora Vision uses port 44100 by default so it is recommended to choose this one.

The Nodka_Server.exe program can be found in Aurora Vision installation path.

Some ports may operate in reverse logic. For details, refer to the electronic input and output diagram documentation of the device.

Device identification

inServerPort field is used to select a TCP port on which the Nodka_Server.exe is listening. Setting this parameter to 'Auto' will select port 44100 which is the default value. ServerPort can be set to:

- **TCP port** - a TCP port on which the Nodka_Server.exe is listening.

inDeviceConfigPath field is used to select a path to the configuration file for a specific Nodka device model. This field cannot be empty for the first Nodka filter used. For each of the following filters with the same value of **inServerPort**, the value of this field can be set to 'Auto' which would result in using the previously specified configuration file. A new configuration file must be provided for each distinct value of **inServerPort**. The Nodka NKDIO SDK installer puts the configuration files in C:\NODKA\NKDIO_SDK\ConfigFile directory by default. The SDK does not check if the specified configuration file is correct, so any errors in the configuration file may be ignored by the SDK. DeviceConfigPath can be set to:

- **File path** - a path to the configuration file for a specific Nodka device model.

inDeviceLightControllerPort field is used to select the COM port of the on-board light controller. Please note that not all Nodka devices have an on-board light controller. For more information refer to the documentation of the specific model. This field is optional and for devices without the light controller should be set to 'Auto'. DeviceLightControllerPort can be set to:

- **COM port** - a COM port of the on-board light controller.

See Also

- Nodka_ReadOutputLevel_Port – Reads an entire output port.

Nodka_ReadOutputLevel_Port

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads an entire output port.

Syntax

```
void avl::Nodka_ReadOutputLevel_Port
(
 Nodka_State& ioState,
 atl::Optional<int> inServerPort,
 atl::Optional<const atl::File&> inDeviceConfigPath,
 atl::Optional<int> inDeviceLightControllerPort,
 int inPortIndex,
 atl::Array<bool>& outLevels
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Nodka_State&			Object used to maintain state of the function.
	inServerPort	Optional<int>	1 - 65535	NIL	Port of NodkaServer service.
	inDeviceConfigPath	Optional<const File&>		NIL	Path to nkio_config.ini file with I/O configuration.
	inDeviceLightControllerPort	Optional<int>	1 - ∞	NIL	COM port used by the light controller (if supported by the device).
	inPortIndex	int	0 - 1		Output port number.
	outLevels	Array<bool>&			Output port levels.

Remarks

This filter allows to get the output level information of a specific output port of the device (8 output channels).

I/O device driver software

This filter is intended to cooperate with a Nodka device using its vendor SDK and Nodka_Server.exe program. To be able to connect to the device, it is required to install Nodka NKDIO SDK.

Add DLL path to system environment variable may be required.

Recommended Nodka NKDIO SDK version for Aurora Vision Studio usage is **v4.1.9**. The implementation of all SDK functionality is done in Nodka_Server.exe program. The filters in Aurora Vision Studio work as a client which communicates with the server and they don't use the SDK directly. Only localhost connections are allowed.

It is mandatory to run the Nodka_Server.exe program **with administrative privileges** and keep it running during the execution of this and other Nodka filters.

For the correct operation of the program, it is required to copy the WinRing0.sys and WinRing0x64.sys files from the SDK directory to the directory where the Nodka_Server.exe file is located. These files are placed in C:\NODKA\NKDIO_SDK\Lib\x86 and C:\NODKA\NKDIO_SDK\Lib\x64 directories by default. Choose those that match your operating system version. Adding the directory to the PATH system variable will likely not work.

To run the program, open the Command Prompt as administrator, navigate to the directory where Nodka_Server.exe file is situated and then run the command `Nodka_Server.exe --port PORT` where 'PORT' is a TCP port on which the server will be listening (an integer value from the range 1-65535). Aurora Vision uses port 44100 by default so it is recommended to choose this one.

The Nodka_Server.exe program can be found in Aurora Vision installation path.

Some ports may operate in reverse logic. For details, refer to the electronic input and output diagram documentation of the device.

Device identification

inServerPort field is used to select a TCP port on which the Nodka_Server.exe is listening. Setting this parameter to 'Auto' will select port 44100 which is the default value. ServerPort can be set to:

- **TCP port** - a TCP port on which the Nodka_Server.exe is listening.

inDeviceConfigPath field is used to select a path to the configuration file for a specific Nodka device model. This field cannot be empty for the first Nodka filter used. For each of the following filters with the same value of **inServerPort**, the value of this field can be set to 'Auto' which would result in using the previously specified configuration file. A new configuration file must be provided for each distinct value of **inServerPort**. The Nodka NKDIO SDK installer puts the configuration files in C:\NODKA\NKDIO_SDK\ConfigFile directory by default. The SDK does not check if the specified configuration file is correct, so any errors in the configuration file may be ignored by the SDK. DeviceConfigPath can be set to:

- **File path** - a path to the configuration file for a specific Nodka device model.

inDeviceLightControllerPort field is used to select the COM port of the on-board light controller. Please note that not all Nodka devices have an on-board light controller. For more information refer to the documentation of the specific model. This field is optional and for devices without the light controller should be set to 'Auto'. DeviceLightControllerPort can be set to:

- **COM port** - a COM port of the on-board light controller.

See Also

- Nodka_ReadOutputLevel_Channel – Reads a single output channel.

Nodka_SetGeneralParam

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets general parameters of the controller.

Syntax

```
void avl::Nodka_SetGeneralParam
(
 Nodka_State& ioState,
 atl::Optional<int> inServerPort,
 atl::Optional<const atl::File&> inDeviceConfigPath,
 atl::Optional<int> inDeviceLightControllerPort,
 int inDevId,
 avl::NodkaGeneralParameter::Type inParamId,
 int inParamLen,
 int inParamValue,
 int inTimeout
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Nodka_State&			Object used to maintain state of the function.
	inServerPort	Optional<int>	1 - 65535	NIL	Port of NodkaServer service.
	inDeviceConfigPath	Optional<const File&>		NIL	Path to nkio_config.ini file with I/O configuration.
	inDeviceLightControllerPort	Optional<int>	1 - ∞	NIL	COM port used by the light controller (if supported by the device).
	inDevId	int	0 - 255		ID of the device (0x01 for the light controller).
	inParamId	NodkaGeneralParameter::Type			ID of the parameter.
	inParamLen	int	0 - 255	1	Length of the parameter (refer to NKIO_API user manual).
	inParamValue	int	0 - ∞		Value of the parameter (refer to NKIO_API user manual).
	inTimeout	int	0 - ∞	100	Timeout for the callback function.

Remarks

This filter can be used to set the value of general parameters of the controller (refer to NKIO_API user manual for more information).

I/O device driver software

This filter is intended to cooperate with a Nodka device using its vendor SDK and Nodka_Server.exe program. To be able to connect to the device, it is required to install Nodka NKDIO SDK.

Add DLL path to system environment variable may be required.

Recommended Nodka NKDIO SDK version for Aurora Vision Studio usage is **v4.1.9**. The implementation of all SDK functionality is done in Nodka_Server.exe program. The filters in Aurora Vision Studio work as a client which communicates with the server and they don't use the SDK directly. Only localhost connections are allowed.

It is mandatory to run the Nodka_Server.exe program **with administrative privileges** and keep it running during the execution of this and other Nodka filters.

For the correct operation of the program, it is required to copy the WinRing0.sys and WinRing0x64.sys files from the SDK directory to the directory where the Nodka_Server.exe file is located. These files are placed in C:\NODKA\NKDIO_SDK\Lib\x86 and C:\NODKA\NKDIO_SDK\Lib\x64 directories by default. Choose those that match your operating system version. Adding the directory to the PATH system variable will likely not work.

To run the program, open the Command Prompt as administrator, navigate to the directory where Nodka_Server.exe file is situated and then run the command `Nodka_Server.exe --port PORT` where 'PORT' is a TCP port on which the server will be listening (an integer value from the range 1-65535). Aurora Vision uses port 44100 by default so it is recommended to choose this one.

The Nodka_Server.exe program can be found in Aurora Vision installation path.

Some ports may operate in reverse logic. For details, refer to the electronic input and output diagram documentation of the device.

Device identification

inServerPort field is used to select a TCP port on which the Nodka_Server.exe is listening. Setting this parameter to 'Auto' will select port 44100 which is the default value. ServerPort can be set to:

- **TCP port** - a TCP port on which the Nodka_Server.exe is listening.

inDeviceConfigPath field is used to select a path to the configuration file for a specific Nodka device model. This field cannot be empty for the first Nodka filter used. For each of the following filters with the same value of **inServerPort**, the value of this field can be set to 'Auto' which would result in using the previously specified configuration file. A new configuration file must be provided for each distinct value of **inServerPort**. The Nodka NKDIO SDK installer puts the configuration files in C:\NODKA\NKDIO_SDK\ConfigFile directory by default. The SDK does not check if the specified configuration file is correct, so any errors in the configuration file may be ignored by the SDK. DeviceConfigPath can be set to:

- **File path** - a path to the configuration file for a specific Nodka device model.

inDeviceLightControllerPort field is used to select the COM port of the on-board light controller. Please note that not all Nodka devices have an on-board light controller. For more information refer to the documentation of the specific model. This field is optional and for devices without the light controller should be set to 'Auto'. DeviceLightControllerPort can be set to:

- **COM port** - a COM port of the on-board light controller.

See Also

- Nodka_GetGeneralParam – Gets general parameters of the controller.

Nodka_SetHoldingTimeUnit

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a holding time unit for a specific channel of the light controller.

Syntax

```
void avl::Nodka_SetHoldingTimeUnit
(
 Nodka_State& ioState,
 atl::Optional<int> inServerPort,
 atl::Optional<const atl::File&> inDeviceConfigPath,
 atl::Optional<int> inDeviceLightControllerPort,
 int inDevId,
 avl::NodkaLightControlChannel::Type inChannel,
 avl::NodkaHoldingTime::Type inHoldingTime,
 int inTimeout
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Nodka_State&			Object used to maintain state of the function.
	inServerPort	Optional<int>	1 - 65535	NIL	Port of NodkaServer service.
	inDeviceConfigPath	Optional<const File&>		NIL	Path to nkio_config.ini file with I/O configuration.
	inDeviceLightControllerPort	Optional<int>	1 - ∞	NIL	COM port used by the light controller (if supported by the device).
	inDevId	int	0 - 255		ID of the device (0x01 for the light controller).
	inChannel	NodkaLightControlChannel::Type			Light controller channel ID.
	inHoldingTime	NodkaHoldingTime::Type			Holding time unit.
	inTimeout	int	0 - ∞	100	Timeout for the callback function.

Remarks

This filter can be used to set the value of the holding time unit for a specific channel of the light controller (refer to NKIO_API user manual for more information).

I/O device driver software

This filter is intended to cooperate with a Nodka device using its vendor SDK and Nodka_Server.exe program. To be able to connect to the device, it is required to install Nodka NKDIO SDK.

Add DLL path to system environment variable may be required.

Recommended Nodka NKDIO SDK version for Aurora Vision Studio usage is **v4.1.9**. The implementation of all SDK functionality is done in Nodka_Server.exe program. The filters in Aurora Vision Studio work as a client which communicates with the server and they don't use the SDK directly. Only localhost connections are allowed.

It is mandatory to run the Nodka_Server.exe program **with administrative privileges** and keep it running during the execution of this and other Nodka filters.

For the correct operation of the program, it is required to copy the WinRing0.sys and WinRing0x64.sys files from the SDK directory to the directory where the Nodka_Server.exe file is located. These files are placed in C:\NODKA\NKDIO_SDK\Lib\x86 and C:\NODKA\NKDIO_SDK\Lib\x64 directories by default. Choose those that match your operating system version. Adding the directory to the PATH system variable will likely not work.

To run the program, open the Command Prompt as administrator, navigate to the directory where Nodka_Server.exe file is situated and then run the command `Nodka_Server.exe --port PORT` where 'PORT' is a TCP port on which the server will be listening (an integer value from the range 1-65535). Aurora Vision uses port 44100 by default so it is recommended to choose this one.

The Nodka_Server.exe program can be found in Aurora Vision installation path.

Some ports may operate in reverse logic. For details, refer to the electronic input and output diagram documentation of the device.

Device identification

inServerPort field is used to select a TCP port on which the Nodka_Server.exe is listening. Setting this parameter to 'Auto' will select port 44100 which is the default value. ServerPort can be set to:

- **TCP port** - a TCP port on which the Nodka_Server.exe is listening.

inDeviceConfigPath field is used to select a path to the configuration file for a specific Nodka device model. This field cannot be empty for the first Nodka filter used. For each of the following filters with the same value of **inServerPort**, the value of this field can be set to 'Auto' which would result in using the previously specified configuration file. A new configuration file must be provided for each distinct value of **inServerPort**. The Nodka NKDIO SDK installer puts the configuration files in C:\NODKA\NKDIO_SDK\ConfigFile directory by default. The SDK does not check if the specified configuration file is correct, so any errors in the configuration file may be ignored by the SDK. DeviceConfigPath can be set to:

- **File path** - a path to the configuration file for a specific Nodka device model.

inDeviceLightControllerPort field is used to select the COM port of the on-board light controller. Please note that not all Nodka devices have an on-board light controller. For more information refer to the documentation of the specific model. This field is optional and for devices without the light controller should be set to 'Auto'. DeviceLightControllerPort can be set to:

- **COM port** - a COM port of the on-board light controller.

See Also

- Nodka_SetGeneralParam – Sets general parameters of the controller.

Nodka_SetPwmParams

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameters of the light controller.

Syntax

```
void avl::Nodka_SetPwmParams
(
 Nodka_State& ioState,
 atl::Optional<int> inServerPort,
 atl::Optional<const atl::File&> inDeviceConfigPath,
 atl::Optional<int> inDeviceLightControllerPort,
 int inDevId,
 avl::NodkaLightControlChannel::Type inChIdx,
 avl::NodkaPwmMode::Type inPwmMode,
 int inPwmValue,
 int inPwmHoldingTime,
 int inPwmOnOff,
 int inTimeout
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Nodka_State&			Object used to maintain state of the function.
	inServerPort	Optional<int>	1 - 65535	NIL	Port of NodkaServer service.
	inDeviceConfigPath	Optional<const File&>		NIL	Path to nkio_config.ini file with I/O configuration.
	inDeviceLightControllerPort	Optional<int>	1 - ∞	NIL	COM port used by the light controller (if supported by the device).
	inDevId	int	0 - 255		ID of the device (0x01 for the light controller).
	inChIdx	NodkaLightControlChannel::Type			Light controller channel ID.
	inPwmMode	NodkaPwmMode::Type			Trigger mode of the light controller.
	inPwmValue	int	0 - 100		Brightness level of the light controller.
	inPwmHoldingTime	int	0 - 255		Duration of the 'on' state of the light controller (see also: Nodka_SetHoldingTimeUnit).
	inPwmOnOff	int	0 - 1		Parameter to turn the light on or off.
	inTimeout	int	0 - ∞	100	Timeout for the callback function.

Remarks

This filter can be used to set the parameters of the specified light control channel.

I/O device driver software

This filter is intended to cooperate with a Nodka device using its vendor SDK and Nodka_Server.exe program. To be able to connect to the device, it is required to install Nodka NKDIO SDK.

Add DLL path to system environment variable may be required.

Recommended Nodka NKDIO SDK version for Aurora Vision Studio usage is **v4.1.9**. The implementation of all SDK functionality is done in Nodka_Server.exe program. The filters in Aurora Vision Studio work as a client which communicates with the server and they don't use the SDK directly. Only localhost connections are allowed.

It is mandatory to run the Nodka_Server.exe program **with administrative privileges** and keep it running during the execution of this and other Nodka filters.

For the correct operation of the program, it is required to copy the WinRing0.sys and WinRing0x64.sys files from the SDK directory to the directory where the Nodka_Server.exe file is located. These files are placed in C:\NODKA\NKDIO_SDK\Lib\x86 and C:\NODKA\NKDIO_SDK\Lib\x64 directories by default. Choose those that match your operating system version. Adding the directory to the PATH system variable will likely not work.

To run the program, open the Command Prompt as administrator, navigate to the directory where Nodka_Server.exe file is situated and then run the command `Nodka_Server.exe --port PORT` where 'PORT' is a TCP port on which the server will be listening (an integer value from the range 1-65535). Aurora Vision uses port 44100 by default so it is recommended to choose this one.

The Nodka_Server.exe program can be found in Aurora Vision installation path.

Some ports may operate in reverse logic. For details, refer to the electronic input and output diagram documentation of the device.

Device identification

inServerPort field is used to select a TCP port on which the Nodka_Server.exe is listening. Setting this parameter to 'Auto' will select port 44100 which is the default value. ServerPort can be set to:

- **TCP port** - a TCP port on which the Nodka_Server.exe is listening.

inDeviceConfigPath field is used to select a path to the configuration file for a specific Nodka device model. This field cannot be empty for the first Nodka filter used. For each of the following filters with the same value of **inServerPort**, the value of this field can be set to 'Auto' which would result in using the previously specified configuration file. A new configuration file must be provided for each distinct value of **inServerPort**. The Nodka NKDIO SDK installer puts the configuration files in C:\NODKA\NKDIO_SDK\ConfigFile directory by default. The SDK does not check if the specified configuration file is correct, so any errors in the configuration file may be ignored by the SDK. DeviceConfigPath can be set to:

- **File path** - a path to the configuration file for a specific Nodka device model.

inDeviceLightControllerPort field is used to select the COM port of the on-board light controller. Please note that not all Nodka devices have an on-board light controller. For more information refer to the documentation of the specific model. This field is optional and for devices without the light controller should be set to 'Auto'. DeviceLightControllerPort can be set to:

- **COM port** - a COM port of the on-board light controller.

See Also

- Nodka_SetPwmParamsNoRetentive – Sets parameters of the light controller. The set parameters will be lost after reboot.
- Nodka_GetPwmParams – Gets parameters of the light controller.

Nodka_SetPwmParamsNoRetentive

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameters of the light controller. The set parameters will be lost after reboot.

Syntax

```
void avl::Nodka_SetPwmParamsNoRetentive
(
 Nodka_State& ioState,
 atl::Optional<int> inServerPort,
 atl::Optional<const atl::File&> inDeviceConfigPath,
 atl::Optional<int> inDeviceLightControllerPort,
 int inDevId,
 avl::NodkaLightControlChannel::Type inChIdx,
 avl::NodkaPwmMode::Type inPwmMode,
 int inPwmValue,
 int inPwmHoldingTime,
 int inPwmOnOff,
 int inTimeout
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Nodka_State&			Object used to maintain state of the function.
	inServerPort	Optional<int>	1 - 65535	NIL	Port of NodkaServer service.
	inDeviceConfigPath	Optional<const File&>		NIL	Path to nkio_config.ini file with I/O configuration.
	inDeviceLightControllerPort	Optional<int>	1 - ∞	NIL	COM port used by the light controller (if supported by the device).
	inDevId	int	0 - 255		ID of the device (0x01 for the light controller).
	inChIdx	NodkaLightControlChannel::Type			Light controller channel ID.
	inPwmMode	NodkaPwmMode::Type			Trigger mode of the light controller.
	inPwmValue	int	0 - 100		Brightness level of the light controller.
	inPwmHoldingTime	int	0 - 255		Duration of the 'on' state of the light controller (see also: Nodka_SetHoldingTimeUnit).
	inPwmOnOff	int	0 - 1		Parameter to turn the light on or off.
	inTimeout	int	0 - ∞	100	Timeout for the callback function.

Remarks

This filter can be used to set the parameters of the specified light control channel without saving the parameters. Please note that the set parameters will be lost after reboot when using this filter.

I/O device driver software

This filter is intended to cooperate with a Nodka device using its vendor SDK and Nodka_Server.exe program. To be able to connect to the device, it is required to install Nodka NKDIO SDK.

Add DLL path to system environment variable may be required.

Recommended Nodka NKDIO SDK version for Aurora Vision Studio usage is **v4.1.9**. The implementation of all SDK functionality is done in Nodka_Server.exe program. The filters in Aurora Vision Studio work as a client which communicates with the server and they don't use the SDK directly. Only localhost connections are allowed.

It is mandatory to run the Nodka_Server.exe program **with administrative privileges** and keep it running during the execution of this and other Nodka filters.

For the correct operation of the program, it is required to copy the WinRing0.sys and WinRing0x64.sys files from the SDK directory to the directory where the Nodka_Server.exe file is located. These files are placed in C:\NODKA\NKDIO_SDK\Lib\x86 and C:\NODKA\NKDIO_SDK\Lib\x64 directories by default. Choose those that match your operating system version. Adding the directory to the PATH system variable will likely not work.

To run the program, open the Command Prompt as administrator, navigate to the directory where Nodka_Server.exe file is situated and then run the command `Nodka_Server.exe --port PORT` where 'PORT' is a TCP port on which the server will be listening (an integer value from the range 1-65535). Aurora Vision uses port 44100 by default so it is recommended to choose this one.

The Nodka_Server.exe program can be found in Aurora Vision installation path.

Some ports may operate in reverse logic. For details, refer to the electronic input and output diagram documentation of the device.

Device identification

inServerPort field is used to select a TCP port on which the Nodka_Server.exe is listening. Setting this parameter to 'Auto' will select port 44100 which is the default value. ServerPort can be set to:

- **TCP port** - a TCP port on which the Nodka_Server.exe is listening.

inDeviceConfigPath field is used to select a path to the configuration file for a specific Nodka device model. This field cannot be empty for the first Nodka filter used. For each of the following filters with the same value of **inServerPort**, the value of this field can be set to 'Auto' which would result in using the previously specified configuration file. A new configuration file must be provided for each distinct value of **inServerPort**. The Nodka NKDIO SDK installer puts the configuration files in C:\NODKA\NKDIO_SDK\ConfigFile directory by default. The SDK does not check if the specified configuration file is correct, so any errors in the configuration file may be ignored by the SDK. DeviceConfigPath can be set to:

- **File path** - a path to the configuration file for a specific Nodka device model.

inDeviceLightControllerPort field is used to select the COM port of the on-board light controller. Please note that not all Nodka devices have an on-board light controller. For more information refer to the documentation of the specific model. This field is optional and for devices without the light controller should be set to 'Auto'. DeviceLightControllerPort can be set to:

- **COM port** - a COM port of the on-board light controller.

See Also

- Nodka_SetPwmParams – Sets parameters of the light controller.
- Nodka_GetPwmParams – Gets parameters of the light controller.

Nodka_WriteOutputLevel_Channel

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Writes a single output channel.

Syntax

```
void avl::Nodka_WriteOutputLevel_Channel
(
 Nodka_State& ioState,
 atl::Optional<int> inServerPort,
 atl::Optional<const atl::File&> inDeviceConfigPath,
 atl::Optional<int> inDeviceLightControllerPort,
 int inPortIndex,
 int inChannelIndex,
 bool inLevel
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Nodka_State&			Object used to maintain state of the function.
	inServerPort	Optional<int>	1 - 65535	NIL	Port of NodkaServer service.
	inDeviceConfigPath	Optional<const File&>		NIL	Path to nkio_config.ini file with I/O configuration.
	inDeviceLightControllerPort	Optional<int>	1 - ∞	NIL	COM port used by the light controller (if supported by the device).
	inPortIndex	int	0 - 1		Output port number.
	inChannelIndex	int	0 - 7		Output port channel.
	inLevel	bool			Output channel level.

Remarks

This filter allows to set the output level for a single output channel of the device.

I/O device driver software

This filter is intended to cooperate with a Nodka device using its vendor SDK and Nodka_Server.exe program. To be able to connect to the device, it is required to install Nodka NKDIO SDK.

Add DLL path to system environment variable may be required.

Recommended Nodka NKDIO SDK version for Aurora Vision Studio usage is **v4.1.9**. The implementation of all SDK functionality is done in Nodka_Server.exe program. The filters in Aurora Vision Studio work as a client which communicates with the server and they don't use the SDK directly. Only localhost connections are allowed.

It is mandatory to run the Nodka_Server.exe program **with administrative privileges** and keep it running during the execution of this and other Nodka filters.

For the correct operation of the program, it is required to copy the WinRing0.sys and WinRing0x64.sys files from the SDK directory to the directory where the Nodka_Server.exe file is located. These files are placed in C:\NODKA\NKDIO_SDK\Lib\x86 and C:\NODKA\NKDIO_SDK\Lib\x64 directories by default. Choose those that match your operating system version. Adding the directory to the PATH system variable will likely not work.

To run the program, open the Command Prompt as administrator, navigate to the directory where Nodka_Server.exe file is situated and then run the command `Nodka_Server.exe --port PORT` where 'PORT' is a TCP port on which the server will be listening (an integer value from the range 1-65535). Aurora Vision uses port 44100 by default so it is recommended to choose this one.

The Nodka_Server.exe program can be found in Aurora Vision installation path.

Some ports may operate in reverse logic. For details, refer to the electronic input and output diagram documentation of the device.

Device identification

inServerPort field is used to select a TCP port on which the Nodka_Server.exe is listening. Setting this parameter to 'Auto' will select port 44100 which is the default value. ServerPort can be set to:

- **TCP port** - a TCP port on which the Nodka_Server.exe is listening.

inDeviceConfigPath field is used to select a path to the configuration file for a specific Nodka device model. This field cannot be empty for the first Nodka filter used. For each of the following filters with the same value of **inServerPort**, the value of this field can be set to 'Auto' which would result in using the previously specified configuration file. A new configuration file must be provided for each distinct value of **inServerPort**. The Nodka NKDIO SDK installer puts the configuration files in C:\NODKA\NKDIO_SDK\ConfigFile directory by default. The SDK does not check if the specified configuration file is correct, so any errors in the configuration file may be ignored by the SDK. DeviceConfigPath can be set to:

- **File path** - a path to the configuration file for a specific Nodka device model.

inDeviceLightControllerPort field is used to select the COM port of the on-board light controller. Please note that not all Nodka devices have an on-board light controller. For more information refer to the documentation of the specific model. This field is optional and for devices without the light controller should be set to 'Auto'. DeviceLightControllerPort can be set to:

- **COM port** - a COM port of the on-board light controller.

See Also

- Nodka_WriteOutputLevel_Port – Writes an entire output port.

Nodka_WriteOutputLevel_Port

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Writes an entire output port.

Syntax

```
void avl::Nodka_WriteOutputLevel_Port
(
 Nodka_State& ioState,
 atl::Optional<int> inServerPort,
 atl::Optional<const atl::File&> inDeviceConfigPath,
 atl::Optional<int> inDeviceLightControllerPort,
 int inPortIndex,
 const atl::Array<bool>& inLevels
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Nodka_State&			Object used to maintain state of the function.
	inServerPort	Optional<int>	1 - 65535	NIL	Port of NodkaServer service.
	inDeviceConfigPath	Optional<const File&>		NIL	Path to nkio_config.ini file with I/O configuration.
	inDeviceLightControllerPort	Optional<int>	1 - ∞	NIL	COM port used by the light controller (if supported by the device).
	inPortIndex	int	0 - 1		Output port number.
	inLevels	const Array<bool>&			Output port levels.

Remarks

This filter allows to set the output level for a specific output port of the device (8 output channels).

I/O device driver software

This filter is intended to cooperate with a Nodka device using its vendor SDK and Nodka_Server.exe program. To be able to connect to the device, it is required to install Nodka NKDIO SDK.

Add DLL path to system environment variable may be required.

Recommended Nodka NKDIO SDK version for Aurora Vision Studio usage is **v4.1.9**. The implementation of all SDK functionality is done in Nodka_Server.exe program. The filters in Aurora Vision Studio work as a client which communicates with the server and they don't use the SDK directly. Only localhost connections are allowed.

It is mandatory to run the Nodka_Server.exe program **with administrative privileges** and keep it running during the execution of this and other Nodka filters.

For the correct operation of the program, it is required to copy the WinRing0.sys and WinRing0x64.sys files from the SDK directory to the directory where the Nodka_Server.exe file is located. These files are placed in C:\NODKA\NKDIO_SDK\Lib\x86 and C:\NODKA\NKDIO_SDK\Lib\x64 directories by default. Choose those that match your operating system version. Adding the directory to the PATH system variable will likely not work.

To run the program, open the Command Prompt as administrator, navigate to the directory where Nodka_Server.exe file is situated and then run the command `Nodka_Server.exe --port PORT` where 'PORT' is a TCP port on which the server will be listening (an integer value from the range 1-65535). Aurora Vision uses port 44100 by default so it is recommended to choose this one.

The Nodka_Server.exe program can be found in Aurora Vision installation path.

Some ports may operate in reverse logic. For details, refer to the electronic input and output diagram documentation of the device.

Device identification

inServerPort field is used to select a TCP port on which the Nodka_Server.exe is listening. Setting this parameter to 'Auto' will select port 44100 which is the default value. ServerPort can be set to:

- **TCP port** - a TCP port on which the Nodka_Server.exe is listening.

inDeviceConfigPath field is used to select a path to the configuration file for a specific Nodka device model. This field cannot be empty for the first Nodka filter used. For each of the following filters with the same value of **inServerPort**, the value of this field can be set to 'Auto' which would result in using the previously specified configuration file. A new configuration file must be provided for each distinct value of **inServerPort**. The Nodka NKDIO SDK installer puts the configuration files in C:\NODKA\NKDIO_SDK\ConfigFile directory by default. The SDK does not check if the specified configuration file is correct, so any errors in the configuration file may be ignored by the SDK. DeviceConfigPath can be set to:

- **File path** - a path to the configuration file for a specific Nodka device model.

inDeviceLightControllerPort field is used to select the COM port of the on-board light controller. Please note that not all Nodka devices have an on-board light controller. For more information refer to the documentation of the specific model. This field is optional and for devices without the light controller should be set to 'Auto'. DeviceLightControllerPort can be set to:

- **COM port** - a COM port of the on-board light controller.

See Also

- Nodka_WriteOutputLevel_Channel – Writes a single output channel.

176. Omron

Table of content:

Omron_GenerateSoftwareTrigger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Generates a software trigger.

Syntax

```
void avl::Omron_GenerateSoftwareTrigger
(
 Omron_State& ioState,
 atl::Optional<const atl::String&> inDeviceID
)
```

Parameters

	Name	Type	Default	Description
	ioState	Omron_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number

Omron_GetBoolParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets a parameter of type Bool.

Syntax

```
void avl::Omron_GetBoolParameter
(
 Omron_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameter,
 bool& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Omron_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inParameter	const String&		Name of parameter
	outValue	bool&		Value retrieved from device

Omron_GetDoubleParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets a parameter of type Double.

Syntax

```
void avl::Omron_GetDoubleParameter
(
 Omron_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameter,
 double& outValue,
 double& outMinValue,
 double& outMaxValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Omron_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inParameter	const String&		Name of parameter
	outValue	double&		Value retrieved from device
	outMinValue	double&		Minimum value of the parameter
	outMaxValue	double&		Maximum value of the parameter

Omron_GetEnumParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets a parameter of type Enum.

Syntax

```
void avl::Omron_GetEnumParameter
(
 Omron_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameter,
 atl::String& outValue,
 atl::Array<atl::String>& outValuesList
)
```

Parameters

	Name	Type	Default	Description
	ioState	Omron_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inParameter	const String&		Name of parameter
	outValue	String&		Value retrieved from device
	outValuesList	Array<String>&		List of possible values of the parameter

Omron_GetIntParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets a parameter of type Integer.

Syntax

```
void avl::Omron_GetIntParameter
(
 Omron_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameter,
 atl::int64& outValue,
 atl::int64& outMinValue,
 atl::int64& outMaxValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Omron_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inParameter	const String&		Name of parameter
	outValue	int64&		Value retrieved from device
	outMinValue	int64&		Minimum value of the parameter
	outMaxValue	int64&		Maximum value of the parameter

Omron_GetStringParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets a parameter of type String.

Syntax

```
void avl::Omron_GetStringParameter
(
 Omron_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameter,
 atl::String& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Omron_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inParameter	const String&		Name of parameter
	outValue	String&		Value retrieved from device

Omron_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a frame using an Omron device.

Syntax

```
bool avl::Omron_GrabImage
(
 Omron_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inInputQueueSize,
 atl::Optional<avl::OmronPixelFormat::Type> inPixelFormat,
 atl::Optional<const avl::Box&> inRoi,
 atl::Optional<avl::OmronAcquisitionMode::Type> inAcquisitionMode,
 atl::Optional<int> inAcquisitionFrameCount,
 atl::Optional<double> inFrameRate,
 atl::Optional<bool> inExposureAuto,
 atl::Optional<double> inExposureTime,
 atl::Optional<bool> inGainAuto,
 atl::Optional<double> inGain,
 atl::Optional<double> inGamma,
 atl::Optional<double> inBlackLevel,
 atl::Optional<bool> inTriggerEnable,
 atl::Optional<avl::OmronTriggerSource::Type> inTriggerSource,
 atl::Optional<avl::OmronTriggerActivation::Type> inTriggerActivation,
 atl::Optional<double> inTriggerDelay,
 avl::Image& outImage,
 atl::int64& outFrameID,
 atl::int64& outTimestamp
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Omron_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Device identifying number
	inInputQueueSize	int	1 - 200	4	Capacity of output frames queue
	inPixelFormat	Optional<OmronPixelFormat::Type>		NIL	Format of the pixels provided by the device
	inRoi	Optional<const Box&>		NIL	Region of interest
	inAcquisitionMode	Optional<OmronAcquisitionMode::Type>		NIL	Acquisition mode of the device
	inAcquisitionFrameCount	Optional<int>		NIL	Number of frames to acquire in MultiFrame Acquisition mode
	inFrameRate	Optional<double>		NIL	Acquisition rate in Hertz at which the frames are captured
	inExposureAuto	Optional<bool>		NIL	Enables or disables the automatic exposure time control
	inExposureTime	Optional<double>		NIL	Exposure time in microseconds
	inGainAuto	Optional<bool>		NIL	Enables or disables the automatic gain control
	inGain	Optional<double>		NIL	Gain as an absolute physical value
	inGamma	Optional<double>		NIL	Gamma correction of pixel intensity
	inBlackLevel	Optional<double>		NIL	Black level as an absolute physical value
	inTriggerEnable	Optional<bool>		NIL	Controls if the selected trigger is active
	inTriggerSource	Optional<OmronTriggerSource::Type>		NIL	Source of the acquisition trigger
	inTriggerActivation	Optional<OmronTriggerActivation::Type>		NIL	Activation mode of the trigger
	inTriggerDelay	Optional<double>		NIL	Delay in microseconds to apply after the trigger reception before activating it
	outImage	Image&			Captured frame
	outFrameID	int64&			Captured frame ID
	outTimestamp	int64&			Captured frame timestamp

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install Sentech SDK Package software with camera dedicated drivers.

Sentech SDK Package can be downloaded from the following website: https://sentech.co.jp/en/data/software/ (registration may be required).

Add DLL path to system environment variable may be required.

Recommended Sentech SDK Package version for Aurora Vision Studio usage is **1.1.2 Update 3**.

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to computer. DeviceID can be set to:

- **Serial Number** - should be printed on the device housing.
- **Device ID** - unique device identifier that can be viewed in StViewer utility.
- **Device User ID** - user-programmable device identifier that can be set in StViewer utility.

Camera parameters

All parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

To change other, more advanced camera parameters, use specific filters.

See Also

- Omron_GrabImage_WithTimeout – Captures a frame using an Omron device.
- Omron_StartAcquisition – Initializes and starts image acquisition in device.

Omron_GrabImage_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a frame using an Omron device.

Syntax

```
bool avl::Omron_GrabImage_WithTimeout
(
 Omron_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inTimeout,
 int inInputQueueSize,
 atl::Optional<avl::OmronPixelFormat::Type> inPixelFormat,
 atl::Optional<const avl::Box&> inRoi,
 atl::Optional<avl::OmronAcquisitionMode::Type> inAcquisitionMode,
 atl::Optional<int> inAcquisitionFrameCount,
 atl::Optional<double> inFrameRate,
 atl::Optional<bool> inExposureAuto,
 atl::Optional<double> inExposureTime,
 atl::Optional<bool> inGainAuto,
 atl::Optional<double> inGain,
 atl::Optional<double> inGamma,
 atl::Optional<double> inBlackLevel,
 atl::Optional<bool> inTriggerEnable,
 atl::Optional<avl::OmronTriggerSource::Type> inTriggerSource,
 atl::Optional<avl::OmronTriggerActivation::Type> inTriggerActivation,
 atl::Optional<double> inTriggerDelay,
 atl::Conditional<avl::Image>& outImage,
 atl::Conditional<atl::int64>& outFrameID,
 atl::Conditional<atl::int64>& outTimestamp
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Omron_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Device identifying number
	inTimeout	int	1 - ∞	100	Maximum time to wait for frame in milliseconds
	inInputQueueSize	int	1 - 200	4	Capacity of output frames queue
	inPixelFormat	Optional<OmronPixelFormat::Type>		NIL	Format of the pixels provided by the device
	inRoi	Optional<const Box&>		NIL	Region of interest
	inAcquisitionMode	Optional<OmronAcquisitionMode::Type>		NIL	Acquisition mode of the device
	inAcquisitionFrameCount	Optional<int>		NIL	Number of frames to acquire in MultiFrame Acquisition mode
	inFrameRate	Optional<double>		NIL	Acquisition rate in Hertz at which the frames are captured
	inExposureAuto	Optional<bool>		NIL	Enables or disables the automatic exposure time control
	inExposureTime	Optional<double>		NIL	Exposure time in microseconds
	inGainAuto	Optional<bool>		NIL	Enables or disables the automatic gain control
	inGain	Optional<double>		NIL	Gain as an absolute physical value
	inGamma	Optional<double>		NIL	Gamma correction of pixel intensity
	inBlackLevel	Optional<double>		NIL	Black level as an absolute physical value
	inTriggerEnable	Optional<bool>		NIL	Controls if the selected trigger is active
	inTriggerSource	Optional<OmronTriggerSource::Type>		NIL	Source of the acquisition trigger
	inTriggerActivation	Optional<OmronTriggerActivation::Type>		NIL	Activation mode of the trigger
	inTriggerDelay	Optional<double>		NIL	Delay in microseconds to apply after the trigger reception before activating it
	outImage	Conditional<Image>&			Captured frame
	outFrameID	Conditional<int64>&			Captured frame ID
	outTimestamp	Conditional<int64>&			Captured frame timestamp

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install Sentech SDK Package software with camera dedicated drivers.

Sentech SDK Package can be downloaded from the following website: https://sentech.co.jp/en/data/software/ (registration may be required).

Add DLL path to system environment variable may be required.

Recommended Sentech SDK Package version for Aurora Vision Studio usage is **1.1.2 Update 3**.

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to computer. DeviceID can be set to:

- **Serial Number** - should be printed on the device housing.
- **Device ID** - unique device identifier that can be viewed in StViewer utility.
- **Device User ID** - user-programmable device identifier that can be set in StViewer utility.

Camera parameters

All parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

To change other, more advanced camera parameters, use specific filters.

See Also

- Omron_GrabImage – Captures a frame using an Omron device.
- Omron_StartAcquisition – Initializes and starts image acquisition in device.

Also in AVL Lite

Omron_SetBoolParameter

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a parameter of type Bool.

Syntax

```
void avl::Omron_SetBoolParameter
(
 Omron_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameter,
 bool inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Omron_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inParameter	const String&		Name of parameter
	inValue	bool		Value to set

Omron_SetDoubleParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a parameter of type Double.

Syntax

```
void avl::Omron_SetDoubleParameter
(
 Omron_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameter,
 double inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Omron_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inParameter	const String&		Name of parameter
	inValue	double		Value to set

Omron_SetEnumParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a parameter of type Enum.

Syntax

```
void avl::Omron_SetEnumParameter
(
 Omron_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameter,
 const atl::String& inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Omron_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inParameter	const String&		Name of parameter
	inValue	const String&		Value to set

Omron_SetIntParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a parameter of type Integer.

Syntax

```
void avl::Omron_SetIntParameter
(
 Omron_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameter,
 atl::int64 inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Omron_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inParameter	const String&		Name of parameter
	inValue	int64		Value to sets

Omron_SetStringParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a parameter of type String.

Syntax

```
void avl::Omron_SetStringParameter
(
 Omron_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameter,
 const atl::String& inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Omron_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inParameter	const String&		Name of parameter
	inValue	const String&		Value to set

Omron_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Initializes and starts image acquisition in device.

Syntax

```
void avl::Omron_StartAcquisition
(
 Omron_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inInputQueueSize,
 atl::Optional<avl::OmronPixelFormat::Type> inPixelFormat,
 atl::Optional<const avl::Box&> inRoi,
 atl::Optional<avl::OmronAcquisitionMode::Type> inAcquisitionMode,
 atl::Optional<int> inAcquisitionFrameCount,
 atl::Optional<double> inFrameRate,
 atl::Optional<bool> inExposureAuto,
 atl::Optional<double> inExposureTime,
 atl::Optional<bool> inGainAuto,
 atl::Optional<double> inGain,
 atl::Optional<double> inGamma,
 atl::Optional<double> inBlackLevel,
 atl::Optional<bool> inTriggerEnable,
 atl::Optional<avl::OmronTriggerSource::Type> inTriggerSource,
 atl::Optional<avl::OmronTriggerActivation::Type> inTriggerActivation,
 atl::Optional<double> inTriggerDelay
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Omron_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Device identifying number
	inInputQueueSize	int	1 - 200	4	Capacity of output frames queue
	inPixelFormat	Optional<OmronPixelFormat::Type>		NIL	Format of the pixels provided by the device
	inRoi	Optional<const Box&>		NIL	Region of interest
	inAcquisitionMode	Optional<OmronAcquisitionMode::Type>		NIL	Acquisition mode of the device
	inAcquisitionFrameCount	Optional<int>		NIL	Number of frames to acquire in MultiFrame Acquisition mode
	inFrameRate	Optional<double>		NIL	Acquisition rate in Hertz at which the frames are captured
	inExposureAuto	Optional<bool>		NIL	Enables or disables the automatic exposure time control
	inExposureTime	Optional<double>		NIL	Exposure time in microseconds
	inGainAuto	Optional<bool>		NIL	Enables or disables the automatic gain control
	inGain	Optional<double>		NIL	Gain as an absolute physical value
	inGamma	Optional<double>		NIL	Gamma correction of pixel intensity
	inBlackLevel	Optional<double>		NIL	Black level as an absolute physical value
	inTriggerEnable	Optional<bool>		NIL	Controls if the selected trigger is active
	inTriggerSource	Optional<OmronTriggerSource::Type>		NIL	Source of the acquisition trigger
	inTriggerActivation	Optional<OmronTriggerActivation::Type>		NIL	Activation mode of the trigger
	inTriggerDelay	Optional<double>		NIL	Delay in microseconds to apply after the trigger reception before activating it

Remarks

This filter is intended for establishing connection with an Omron camera device using Sentech SDK Package interface, to initialize image streaming. It is only needed when explicit image acquisition start is required in the initial phase of a program. For example, it can be used to prepare a camera, running in triggered mode, to be able to capture trigger signals before the first invoke of Omron_GrabImage or to start multiple cameras in sync before the acquisition phase.

The use of this filter is not obligatory. Omron_GrabImage or Omron_GrabImage_WithTimeout] filters will initialize and start image acquisition upon their first invoke.

This filter has no effect when invoked for the second time and when invoked after image grabbing filters.

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install Sentech SDK Package software with camera dedicated drivers.

Sentech SDK Package can be downloaded from the following website: https://sentech.co.jp/en/data/software/ (registration may be required).

Add DLL path to system environment variable may be required.

Recommended Sentech SDK Package version for Aurora Vision Studio usage is **1.1.2 Update 3**.

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to computer. DeviceID can be set to:

- **Serial Number** - should be printed on the device housing.
- **Device ID** - unique device identifier that can be viewed in StViewer utility.
- **Device User ID** - user-programmable device identifier that can be set in StViewer utility.

Camera parameters

All parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

To change other, more advanced camera parameters, use specific filters.

See Also

- Omron_GrabImage – Captures a frame using an Omron device.
- Omron_GrabImage_WithTimeout – Captures a frame using an Omron device.

Omron_StopAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Stops image acquisition in device.

Syntax

```
void avl::Omron_StopAcquisition
(
 Omron_State& ioState,
 atl::Optional<const atl::String&> inDeviceID
)
```

Parameters

	Name	Type	Default	Description
	ioState	Omron_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number

177. Kinect

Table of content:

OpenNI_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a frame using Kinect.

Syntax

```
bool avl::OpenNI_GrabImage
(
 OpenNI_GrabImageState& ioState,
 avl::Image& outImage,
 avl::Image& outDepthImage
)
```

Parameters

	Name	Type	Default	Description
	ioState	OpenNI_GrabImageState&		Object used to maintain state of the function.
	outImage	Image&		Captured frame
	outDepthImage	Image&		

Description

To be able to grab images using Kinect you need to install:

- Kinect SDK 1.8
- OpenNI 2.2
- NITE 2.2

Please contact our support to get OpenNI / NITE installers.

Installed software must match the Aurora Vision Studio platform (x86 or x64).

After installing these files you have to manually copy all the files and directories from *OpenNI2\Redist* directory and all the dll files from the *PrimeSense\NiTE2\Redist* directory to your Aurora Vision Studio main directory.

Additionally, if you save the project in which you are using the skeleton tracing feature, you should copy the `PrimeSense\NiTE2\Redist\NiTE2` directory also to the project directory.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- OpenNI_SkeletonTracing – Captures a skeleton using Kinect.

OpenNI_SkeletonTracing

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a skeleton using Kinect.

Syntax

```
bool avl::OpenNI_SkeletonTracing
(
 OpenNI_SkeletonTracingState& ioState,
 int& outNumberOfPersons,
 atl::Array<atl::Array<avl::Point2D>>& outSkeletons,
 avl::Region& outPlayerRegion
)
```

Parameters

	Name	Type	Default	Description
	ioState	OpenNI_SkeletonTracingState&		Object used to maintain state of the function.
	outNumberOfPersons	int&		
	outSkeletons	Array<Array<Point2D>>&		
	outPlayerRegion	Region&		

Description

To be able to grab images using Kinect you need to install:

- Kinect SDK 1.8
- OpenNI 2.2
- NITE 2.2

Please contact our support to get OpenNI / NITE installers.

Installed software must match the Aurora Vision Studio platform (x86 or x64).

After installing these files you have to manually copy all the files and directories from *OpenNI2\Redist* directory and all the dll files from the *PrimeSense\NiTE2\Redist* directory to your Aurora Vision Studio main directory.

Additionally, if you save the project in which you are using the skeleton tracing feature, you should copy the *PrimeSense\NiTE2\Redist\NiTE2* directory also to the project directory.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- OpenNI_GrabImage – Captures a frame using Kinect.

178. OptoEngineeringHRAS

Table of content:

OptoEngineeringHRAS_COE71_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an image using COE-71 Opto Engineering device.

Syntax

```
bool avl::OptoEngineeringHRAS_COE71_GrabImage
(
 OptoEngineering_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inInputQueueSize,
 avl::OptoEngineeringBitDepth::Type inBitDepth,
 atl::Optional<avl::OptoEngineering_COE71_StrobePolarity::Type> inStrobePolarity,
 avl::OptoEngineering_COE71_TriggerMode::Type inTriggerMode,
 atl::Optional<float> inDigitalGain,
 atl::Optional<int> inDigitalOffset,
 atl::Optional<int> inExposureTime,
 avl::OptoEngineeringExposureUnit::Type inExposureUnit,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	OptoEngineering_State&			Object used to maintain state of the function.
→	inDeviceID	Optional<const String&>		NIL	Device identifying number
→	inInputQueueSize	int	1 - ∞	12	Capacity of output frames queue
→	inBitDepth	OptoEngineeringBitDepth::Type			Number of pixel bits
→	inStrobePolarity	Optional<OptoEngineering_COE71_StrobePolarity::Type>		NIL	Polarity of strobe output signal
→	inTriggerMode	OptoEngineering_COE71_TriggerMode::Type			Device triggering mode
→	inDigitalGain	Optional<float>	0.0 - 16.0	NIL	Gain of source image in device raw unit
→	inDigitalOffset	Optional<int>	-32767 - 32767	NIL	Offset of source image in device raw unit
→	inExposureTime	Optional<int>	1 - 65535	NIL	Camera frame exposition time
→	inExposureUnit	OptoEngineeringExposureUnit::Type			Camera frame exposition time unit
←	outImage	Image&			Captured frame

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install eBUS software with camera dedicated drivers.

eBUS SDK can be downloaded from the following website: https://supportcenter.pleora.com/s/topic/0TO340000004X6dGAE/ebus-sdk?tabset-0c866=2&tabset-25adb=81d66 (registration may be required).

Recommended eBUS SDK version for Aurora Vision Studio usage is **5.1.10.4642**.

Add DLL path to system environment variable may be required.

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to computer. DeviceID can be set to:

- **Device unique id number** - should be specified on device casing.

Camera parameters

To change other and more advanced internal camera parameters use configuration tool "Camera Control Application" available from the following website: https://www.opto-e.com/products/coe-area-scan#Downloads (registration may be required).

Other device settings can be changed also with eBUs Player available from the following website: https://www.pleora.com/products/ebus-sdk/ (registration may be required).

Supported devices

This filter support only COE HR AS 50, 71 and 29 series with USB interface. COE-71 devices have dedicated support.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- OptoEngineeringHRAS_COE71_GrabImage_WithTimeout – Captures an image using COE-71 Opto Engineering device.
- OptoEngineeringHRAS_COE71_StartAcquisition – Initializes and starts image acquisition in a camera.
- OptoEngineeringHRAS_GrabImage – Captures an image using Opto Engineering device.

OptoEngineeringHRAS_COE71_GrabImage_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an image using COE-71 Opto Engineering device.

Syntax

```
bool avl::OptoEngineeringHRAS_COE71_GrabImage_WithTimeout
(
 OptoEngineering_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inTimeout,
 int inInputQueueSize,
 avl::OptoEngineeringBitDepth::Type inBitDepth,
 atl::Optional<avl::OptoEngineering_COE71_StrobePolarity::Type> inStrobePolarity,
 avl::OptoEngineering_COE71_TriggerMode::Type inTriggerMode,
 atl::Optional<float> inDigitalGain,
 atl::Optional<int> inDigitalOffset,
 atl::Optional<int> inExposureTime,
 avl::OptoEngineeringExposureUnit::Type inExposureUnit,
 atl::Conditional<avl::Image>& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	OptoEngineering_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Device identifying number
	inTimeout	int	100 - ∞	100	Maximum time to wait for frame in milliseconds
	inInputQueueSize	int	1 - ∞	12	Capacity of output frames queue
	inBitDepth	OptoEngineeringBitDepth::Type			Number of pixel bits
	inStrobePolarity	Optional<OptoEngineering_COE71_StrobePolarity::Type>		NIL	Polarity of strobe output signal
	inTriggerMode	OptoEngineering_COE71_TriggerMode::Type			Device triggering mode
	inDigitalGain	Optional<float>	0.0 - 16.0	NIL	Gain of source image in device raw unit
	inDigitalOffset	Optional<int>	-32767 - 32767	NIL	Offset of source image in device raw unit
	inExposureTime	Optional<int>	1 - 65535	NIL	Camera frame exposition time
	inExposureUnit	OptoEngineeringExposureUnit::Type			Camera frame exposition time unit
	outImage	Conditional<Image>&			Captured frame

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install eBUS software with camera dedicated drivers.

eBUS SDK can be downloaded from the following website: https://supportcenter.pleora.com/s/topic/0TO340000004X6dGAE/ebus-sdk?tabset-0c866=2&tabset-25adb=81d66 (registration may be required).

Recommended eBUS SDK version for Aurora Vision Studio usage is **5.1.10.4642**.

Add DLL path to system environment variable may be required.

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to computer. DeviceID can be set to:

- **Device unique id number** - should be specified on device casing.

Camera parameters

To change other and more advanced internal camera parameters use configuration tool "Camera Control Application" available from the following website: https://www.opto-e.com/products/coe-area-scan#Downloads (registration may be required).

Other device settings can be changed also with eBUs Player available from the following website: https://www.pleora.com/products/ebus-sdk/ (registration may be required).

Supported devices

This filter support only COE HR AS 50, 71 and 29 series with USB interface. COE-71 devices have dedicated support.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- OptoEngineeringHRAS_COE71_GrabImage – Captures an image using COE-71 Opto Engineering device.
- OptoEngineeringHRAS_COE71_StartAcquisition – Initializes and starts image acquisition in a camera.
- OptoEngineeringHRAS_GrabImage – Captures an image using Opto Engineering device.

OptoEngineeringHRAS_COE71_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Initializes and starts image acquisition in a camera.

Syntax

```
void avl::OptoEngineeringHRAS_COE71_StartAcquisition
(
 OptoEngineering_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inInputQueueSize,
 avl::OptoEngineeringBitDepth::Type inBitDepth,
 atl::Optional<avl::OptoEngineering_COE71_StrobePolarity::Type> inStrobePolarity,
 avl::OptoEngineering_COE71_TriggerMode::Type inTriggerMode,
 atl::Optional<float> inDigitalGain,
 atl::Optional<int> inDigitalOffset,
 atl::Optional<int> inExposureTime,
 avl::OptoEngineeringExposureUnit::Type inExposureUnit
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	OptoEngineering_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Device identifying number
	inInputQueueSize	int	1 - ∞	12	Capacity of output frames queue
	inBitDepth	OptoEngineeringBitDepth::Type			Number of pixel bits
	inStrobePolarity	Optional<OptoEngineering_COE71_StrobePolarity::Type>		NIL	Polarity of strobe output signal
	inTriggerMode	OptoEngineering_COE71_TriggerMode::Type			Device triggering mode
	inDigitalGain	Optional<float>	0.0 - 16.0	NIL	Gain of source image in device raw unit
	inDigitalOffset	Optional<int>	-32767 - 32767	NIL	Offset of source image in device raw unit
	inExposureTime	Optional<int>	1 - 65535	NIL	Camera frame exposition time
	inExposureUnit	OptoEngineeringExposureUnit::Type			Camera frame exposition time unit

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install eBUS software with camera dedicated drivers.

eBUS SDK can be downloaded from the following website: https://supportcenter.pleora.com/s/topic/0TO340000004X6dGAE/ebus-sdk?tabset-0c866=2&tabset-25adb=81d66 (registration may be required).

Recommended eBUS SDK version for Aurora Vision Studio usage is **5.1.10.4642**.

Add DLL path to system environment variable may be required.

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to computer. DeviceID can be set to:

- **Device unique id number** - should be specified on device casing.

Camera parameters

To change other and more advanced internal camera parameters use configuration tool "Camera Control Application" available from the following website: https://www.opto-e.com/products/coe-area-scan#Downloads (registration may be required).

Other device settings can be changed also with eBUs Player available from the following website: https://www.pleora.com/products/ebus-sdk/ (registration may be required).

Supported devices

This filter support only COE HR AS 50, 71 and 29 series with USB interface. COE-71 devices have dedicated support.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- OptoEngineeringHRAS_GrabImage – Captures an image using Opto Engineering device.
- OptoEngineeringHRAS_GrabImage_WithTimeout – Captures with timeout an image using Opto Engineering device.
- OptoEngineeringHRAS_COE71_StartAcquisition – Initializes and starts image acquisition in a camera.

OptoEngineeringHRAS_GenerateSoftwareTrigger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Generates software trigger.

Syntax

```
void avl::OptoEngineeringHRAS_GenerateSoftwareTrigger
(
 OptoEngineering_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inTime
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	OptoEngineering_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Device identifying number
	inTime	int	1 - 65535	100	Duration of trigger in milliseconds

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

OptoEngineeringHRAS_GetBooleanParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets Genicam parameter of type Bool from internal frame grabber.

Syntax

```
void avl::OptoEngineeringHRAS_GetBooleanParameter
(
 OptoEngineering_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 bool& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	OptoEngineering_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inName	const String&		Parameter name
	outValue	bool&		

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

OptoEngineeringHRAS_GetEnumParameterAsInteger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets Genicam parameter of type Enum from internal frame grabber.

Syntax

```
void avl::OptoEngineeringHRAS_GetEnumParameterAsInteger
(
 OptoEngineering_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 atl::int64& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	OptoEngineering_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inName	const String&		Parameter name
	outValue	int64&		

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

OptoEngineeringHRAS_GetEnumParameterAsString

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets Genicam parameter of type Enum from internal frame grabber.

Syntax

```
void avl::OptoEngineeringHRAS_GetEnumParameterAsString
(
 OptoEngineering_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 atl::String& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	OptoEngineering_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inName	const String&		Parameter name
	outValue	String&		

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

OptoEngineeringHRAS_GetFloatParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets Genicam parameter of type Float from internal frame grabber.

Syntax

```
void avl::OptoEngineeringHRAS_GetFloatParameter
(
 OptoEngineering_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 double& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	OptoEngineering_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inName	const String&		Parameter name
	outValue	double&		

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

OptoEngineeringHRAS_GetIntegerParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets Genicam parameter of type Integer from internal frame grabber.

Syntax

```
void avl::OptoEngineeringHRAS_GetIntegerParameter
(
 OptoEngineering_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 atl::int64& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	OptoEngineering_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inName	const String&		Parameter name
	outValue	int64&		

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

OptoEngineeringHRAS_GetStringParameter

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets Genicam parameter of type String from internal frame grabber.

Syntax

```
void avl::OptoEngineeringHRAS_GetStringParameter
(
 OptoEngineering_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 atl::String& outValue
)
```

Parameters

	Name	Type	Default	Description
↔	ioState	OptoEngineering_State&		Object used to maintain state of the function.
→	inDeviceID	Optional<const String&>	NIL	Device identifying number
→	inName	const String&		Parameter name
←	outValue	String&		

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

OptoEngineeringHRAS_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an image using Opto Engineering device.

Syntax

```
bool avl::OptoEngineeringHRAS_GrabImage
(
 OptoEngineering_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inInputQueueSize,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	OptoEngineering_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Device identifying number
	inInputQueueSize	int	1 - ∞	12	Capacity of output frames queue
	outImage	Image&			Captured frame

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install eBUS software with camera dedicated drivers.

eBUS SDK can be downloaded from the following website: https://supportcenter.pleora.com/s/topic/0TO340000004X6dGAE/ebus-sdk?tabset-0c866=2&tabset-25adb=81d66 (registration may be required).

Recommended eBUS SDK version for Aurora Vision Studio usage is **5.1.10.4642**.

Add DLL path to system environment variable may be required.

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to computer. DeviceID can be set to:

- **Device unique id number** - should be specified on device casing.

Camera parameters

To change other and more advanced internal camera parameters use configuration tool "Camera Control Application" available from the following website: https://www.opto-e.com/products/coe-area-scan#Downloads (registration may be required).

Other device settings can be changed also with eBUs Player available from the following website: https://www.pleora.com/products/ebus-sdk/ (registration may be required).

Supported devices

This filter support only COE HR AS 50, 71 and 29 series with USB interface. COE-71 devices have dedicated support.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- OptoEngineeringHRAS_GrabImage_WithTimeout – Captures with timeout an image using Opto Engineering device.
- OptoEngineeringHRAS_StartAcquisition – Initializes and starts image acquisition in a camera.
- OptoEngineeringHRAS_COE71_GrabImage – Captures an image using COE-71 Opto Engineering device.

OptoEngineeringHRAS_GrabImage_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures with timeout an image using Opto Engineering device.

Syntax

```
bool avl::OptoEngineeringHRAS_GrabImage_WithTimeout
(
 OptoEngineering_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inTimeout,
 int inInputQueueSize,
 atl::Conditional<avl::Image>& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
↻	ioState	OptoEngineering_State&			Object used to maintain state of the function.
→	inDeviceID	Optional<const String&>		NIL	Device identifying number
→	inTimeout	int	100 - ∞	100	Maximum time to wait for frame in milliseconds
→	inInputQueueSize	int	1 - ∞	12	Capacity of output frames queue
←	outImage	Conditional<Image>&			Captured frame

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install eBUS software with camera dedicated drivers.

eBUS SDK can be downloaded from the following website: https://supportcenter.pleora.com/s/topic/0TO340000004X6dGAE/ebus-sdk?tabset-0c866=2&tabset-25adb=81d66 (registration may be required).

Recommended eBUS SDK version for Aurora Vision Studio usage is **5.1.10.4642**.

Add DLL path to system environment variable may be required.

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to computer. DeviceID can be set to:

- **Device unique id number** - should be specified on device casing.

Camera parameters

To change other and more advanced internal camera parameters use configuration tool "Camera Control Application" available from the following website: https://www.opto-e.com/products/coe-area-scan#Downloads (registration may be required).

Other device settings can be changed also with eBUs Player available from the following website: https://www.pleora.com/products/ebus-sdk/ (registration may be required).

Supported devices

This filter support only COE HR AS 50, 71 and 29 series with USB interface. COE-71 devices have dedicated support.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- OptoEngineeringHRAS_GrabImage – Captures an image using Opto Engineering device.
- OptoEngineeringHRAS_StartAcquisition – Initializes and starts image acquisition in a camera.
- OptoEngineeringHRAS_COE71_GrabImage_WithTimeout – Captures an image using COE-71 Opto Engineering device.

OptoEngineeringHRAS_SendSerialRawCommand

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets serial port internal protocol frame to device.

Syntax

```
void avl::OptoEngineeringHRAS_SendSerialRawCommand
(
 OptoEngineering_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const avl::OptoEngineeringSerialFrame& inData,
 atl::Conditional<avl::OptoEngineeringSerialFrame>& outResponse
)
```

Parameters

	Name	Type	Default	Description
	ioState	OptoEngineering_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inData	const OptoEngineeringSerialFrame&		Input frame command
	outResponse	Conditional<OptoEngineeringSerialFrame>&		Output frame, set only for read mode

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install eBUS software with camera dedicated drivers.

eBUS SDK can be downloaded from the following website: https://supportcenter.pleora.com/s/topic/0TO340000004X6dGAE/ebus-sdk?tabset-0c866=2&tabset-25adb=81d66 (registration may be required).

Recommended eBUS SDK version for Aurora Vision Studio usage is **5.1.10.4642**.

Add DLL path to system environment variable may be required.

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to computer. DeviceID can be set to:

- **Device unique id number** - should be specified on device casing.

Camera parameters

To change other and more advanced internal camera parameters use configuration tool "Camera Control Application" available from the following website: https://www.opto-e.com/products/coe-area-scan#Downloads (registration may be required).

Other device settings can be changed also with eBUs Player available from the following website: https://www.pleora.com/products/ebus-sdk/ (registration may be required).

Supported devices

This filter support only COE HR AS 50, 71 and 29 series with USB interface. COE-71 devices have dedicated support.

Application example

This filter provide implementation of internal protocol based on serial port.

Parameters must be formatted as hexadecimal number in string. Data field can be set empty - it will be autofill as zero.

Parameters list for each device can be find here: https://www.opto-e.com/products/coe-area-scan#Downloads (registration may be required).

For example, to read camera temperature use target: 04, index: 07, data: 0000. Checksum is automatic calculated.

This filter can fail you retry send command every loop iteration.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- OptoEngineeringHRAS_GrabImage – Captures an image using Opto Engineering device.
- OptoEngineeringHRAS_GrabImage_WithTimeout – Captures with timeout an image using Opto Engineering device.
- OptoEngineeringHRAS_StartAcquisition – Initializes and starts image acquisition in a camera.

OptoEngineeringHRAS_SetBooleanParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets Genicam parameter of type Bool to internal frame grabber.

Syntax

```
void avl::OptoEngineeringHRAS_SetBooleanParameter
(
 OptoEngineering_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 bool inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	OptoEngineering_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inName	const String&		Parameter name
	inValue	bool		

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

OptoEngineeringHRAS_SetDigitalGain

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets digital gain.

Syntax

```
void avl::OptoEngineeringHRAS_SetDigitalGain
(
 OptoEngineering_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 float inGain
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	OptoEngineering_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Device identifying number
	inGain	float	0.0 - 16.0	1.0f	Gain of source image in device raw unit

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

OptoEngineeringHRAS_SetDigitalOffset

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets digital offset.

Syntax

```
void avl::OptoEngineeringHRAS_SetDigitalOffset
(
 OptoEngineering_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inOffset
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	OptoEngineering_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Device identifying number
	inOffset	int	-32767 - 32767	0	Offset of source image in device raw unit

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

OptoEngineeringHRAS_SetEnumParameterAsInteger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets Genicam parameter of type Enum to internal frame grabber.

Syntax

```
void avl::OptoEngineeringHRAS_SetEnumParameterAsInteger
(
 OptoEngineering_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 atl::int64 inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	OptoEngineering_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inName	const String&		Parameter name
	inValue	int64		

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

OptoEngineeringHRAS_SetEnumParameterAsString

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets Genicam parameter of type Enum to internal frame grabber.

Syntax

```
void avl::OptoEngineeringHRAS_SetEnumParameterAsString
(
 OptoEngineering_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 const atl::String& inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	OptoEngineering_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inName	const String&		Parameter name
	inValue	const String&		

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

OptoEngineeringHRAS_SetFloatParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets Genicam parameter of type Float to internal frame grabber.

Syntax

```
void avl::OptoEngineeringHRAS_SetFloatParameter
(
 OptoEngineering_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 double inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	OptoEngineering_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inName	const String&		Parameter name
	inValue	double		

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

OptoEngineeringHRAS_SetIntegerParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets Genicam parameter of type Integer to internal frame grabber.

Syntax

```
void avl::OptoEngineeringHRAS_SetIntegerParameter
(
 OptoEngineering_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 atl::int64 inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	OptoEngineering_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inName	const String&		Parameter name
	inValue	int64		

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

OptoEngineeringHRAS_SetStringParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets Genicam parameter of type String to internal frame grabber.

Syntax

```
void avl::OptoEngineeringHRAS_SetStringParameter
(
 OptoEngineering_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 const atl::String& inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	OptoEngineering_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inName	const String&		Parameter name
	inValue	const String&		

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

OptoEngineeringHRAS_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Initializes and starts image acquisition in a camera.

Syntax

```
void avl::OptoEngineeringHRAS_StartAcquisition
(
 OptoEngineering_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inInputQueueSize
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	OptoEngineering_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Device identifying number
	inInputQueueSize	int	1 - ∞	12	Capacity of output frames queue

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install eBUS software with camera dedicated drivers.

eBUS SDK can be downloaded from the following website: https://supportcenter.pleora.com/s/topic/0TO340000004X6dGAE/ebus-sdk?tabset-0c866=2&tabset-25adb=81d66 (registration may be required).

Recommended eBUS SDK version for Aurora Vision Studio usage is **5.1.10.4642**.

Add DLL path to system environment variable may be required.

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to computer. DeviceID can be set to:

- **Device unique id number** - should be specified on device casing.

Camera parameters

To change other and more advanced internal camera parameters use configuration tool "Camera Control Application" available from the following website: https://www.opto-e.com/products/coe-area-scan#Downloads (registration may be required).

Other device settings can be changed also with eBUs Player available from the following website: https://www.pleora.com/products/ebus-sdk/ (registration may be required).

Supported devices

This filter support only COE HR AS 50, 71 and 29 series with USB interface. COE-71 devices have dedicated support.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- OptoEngineeringHRAS_GrabImage – Captures an image using Opto Engineering device.
- OptoEngineeringHRAS_GrabImage_WithTimeout – Captures with timeout an image using Opto Engineering device.
- OptoEngineeringHRAS_COE71_StartAcquisition – Initializes and starts image acquisition in a camera.

OptoEngineeringHRAS_StopAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Stops image acquisition in a camera.

Syntax

```
void avl::OptoEngineeringHRAS_StopAcquisition
(
 OptoEngineering_State& ioState,
 atl::Optional<const atl::String&> inDeviceID
)
```

Parameters

	Name	Type	Default	Description
	ioState	OptoEngineering_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

179. Photoneo

Table of content:

Photoneo_GenerateSoftwareTrigger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Triggers software trigger.

Syntax

```
void avl::Photoneo_GenerateSoftwareTrigger
(
 Photoneo_State& ioState,
 const atl::Optional<atl::String> inDeviceID
)
```

Parameters

	Name	Type	Default	Description
	ioState	Photoneo_State&		Object used to maintain state of the function.
	inDeviceID	const Optional<String>	NIL	Device identification string

Remarks

PhoXi Control

PhoXi Control process needs to run for the filters to work.

Camera Setup

It is advisable to use Photoneo_StartAcquisition filter and setup inCaptureSettings with proper values. Some values might not be set to values the user expects and acquisition will fail or wait infinitely, when wrong trigger source or output type is selected.

Camera identification

When there is only one Photoneo camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one camera connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Photoneo PhoXi Control software.

Photoneo PhoXi Control can be downloaded from the following website: https://www.photoneo.com/3d-scanning-software/

Supported PhoXi Control version for Aurora Vision Studio usage is **1.2.35**.

Environment variable PHOXI_CONTROL_PATH which should point to the installation folder of PhoXi Control. PHO_LOG_FILES_DIR environment variable must contain a path to accessible folder. PHO_LOG_FILES_DIR is not set automatically after SDk installation. Manual configuration of this variable may be a requirement. Failure to meet these requirements may result in unexpected termination of the application.

Rebooting computer after SDK installation is mandatory.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Photoneo_StartAcquisition – Starts capturing data from a Photoneo camera.
- Photoneo_GrabPoint3DGrid – Captures PointCloud structure using Photoneo.

Photoneo_GetParameters

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets the camera parameters.

Syntax

```
void avl::Photoneo_GetParameters
(
 Photoneo_State& ioState,
 const atl::Optional<atl::String> inDeviceID,
 avl::PhotoneoCaptureSettings& outCaptureSettings
)
```

Parameters

	Name	Type	Default	Description
↔	ioState	Photoneo_State&		Object used to maintain state of the function.
→	inDeviceID	const Optional<String>	NIL	Device identification string
←	outCaptureSettings	PhotoneoCaptureSettings&		

Remarks

PhoXi Control

PhoXi Control process needs to run for the filters to work.

Camera Setup

It is advisable to use Photoneo_StartAcquisition filter and setup inCaptureSettings with proper values. Some values might not be set to values the user expects and acquisition will fail or wait infinitely, when wrong trigger source or output type is selected.

Camera identification

When there is only one Photoneo camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one camera connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Photoneo PhoXi Control software.

Photoneo PhoXi Control can be downloaded from the following website: https://www.photoneo.com/3d-scanning-software/

Supported PhoXi Control version for Aurora Vision Studio usage is **1.2.35**.

Environment variable PHOXI_CONTROL_PATH which should point to the installation folder of PhoXi Control. PHO_LOG_FILES_DIR environment variable must contain a path to accessible folder. PHO_LOG_FILES_DIR is not set automatically after SDk installation. Manual configuration of this variable may be a requirement. Failure to meet these requirements may result in unexpected termination of the application.

Rebooting computer after SDK installation is mandatory.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Photoneo_SetParameters – Sets the camera parameters.
- Photoneo_GrabPoint3DGrid – Captures PointCloud structure using Photoneo.

Photoneo_GrabConfidence

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures ConfidenceMap structure using Photoneo.

Syntax

```
bool avl::Photoneo_GrabConfidence
(
 Photoneo_State& ioState,
 const atl::Optional<atl::String> inDeviceID,
 const avl::PhotoneoCaptureSettings& inCaptureSettings,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
	ioState	Photoneo_State&		Object used to maintain state of the function.
	inDeviceID	const Optional<String>	NIL	Device identification string
	inCaptureSettings	const PhotoneoCaptureSettings&		
	outImage	Image&		Output image

Remarks

PhoXi Control

PhoXi Control process needs to run for the filters to work.

Camera Setup

It is advisable to use Photoneo_StartAcquisition filter and setup inCaptureSettings with proper values. Some values might not be set to values the user expects and acquisition will fail or wait infinitely, when wrong trigger source or output type is selected.

Camera identification

When there is only one Photoneo camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one camera connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Photoneo PhoXi Control software.

Photoneo PhoXi Control can be downloaded from the following website: https://www.photoneo.com/3d-scanning-software/

Supported PhoXi Control version for Aurora Vision Studio usage is **1.2.35**.

Environment variable PHOXI_CONTROL_PATH which should point to the installation folder of PhoXi Control. PHO_LOG_FILES_DIR environment variable must contain a path to accessible folder. PHO_LOG_FILES_DIR is not set automatically after SDk installation. Manual configuration of this variable may be a requirement. Failure to meet these requirements may result in unexpected termination of the application.

Rebooting computer after SDK installation is mandatory.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Photoneo_StartAcquisition – Starts capturing data from a Photoneo camera.
- Photoneo_GrabTexture – Captures Texture structure using Photoneo.

Photoneo_GrabConfidence_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures ConfidenceMap structure using Photoneo.

Syntax

```
bool avl::Photoneo_GrabConfidence_WithTimeout
(
 Photoneo_State& ioState,
 const atl::Optional<atl::String> inDeviceID,
 const atl::Optional<int> inTimeout,
 const avl::PhotoneoCaptureSettings& inCaptureSettings,
 atl::Conditional<avl::Image>& outImage
)
```

Parameters

	Name	Type	Default	Description
	ioState	Photoneo_State&		Object used to maintain state of the function.
	inDeviceID	const Optional<String>	NIL	Device identification string
	inTimeout	const Optional<int>	NIL	Timeout [ms]
	inCaptureSettings	const PhotoneoCaptureSettings&		
	outImage	Conditional<Image>&		Output image

Remarks

PhoXi Control

PhoXi Control process needs to run for the filters to work.

Camera Setup

It is advisable to use Photoneo_StartAcquisition filter and setup inCaptureSettings with proper values. Some values might not be set to values the user expects and acquisition will fail or wait infinitely, when wrong trigger source or output type is selected.

Camera identification

When there is only one Photoneo camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one camera connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Photoneo PhoXi Control software.

Photoneo PhoXi Control can be downloaded from the following website: https://www.photoneo.com/3d-scanning-software/

Supported PhoXi Control version for Aurora Vision Studio usage is **1.2.35**.

Environment variable PHOXI_CONTROL_PATH which should point to the installation folder of PhoXi Control. PHO_LOG_FILES_DIR environment variable must contain a path to accessible folder. PHO_LOG_FILES_DIR is not set automatically after SDk installation. Manual configuration of this variable may be a requirement. Failure to meet these requirements may result in unexpected termination of the application.

Rebooting computer after SDK installation is mandatory.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Photoneo_StartAcquisition – Starts capturing data from a Photoneo camera.
- Photoneo_GrabTexture – Captures Texture structure using Photoneo.

Photoneo_GrabDepthMap

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures DepthMap structure using Photoneo.

Syntax

```
bool avl::Photoneo_GrabDepthMap
(
 Photoneo_State& ioState,
 const atl::Optional<atl::String> inDeviceID,
 const avl::PhotoneoCaptureSettings& inCaptureSettings,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
	ioState	Photoneo_State&		Object used to maintain state of the function.
	inDeviceID	const Optional<String>	NIL	Device identification string
	inCaptureSettings	const PhotoneoCaptureSettings&		
	outImage	Image&		Output image

Remarks

PhoXi Control

PhoXi Control process needs to run for the filters to work.

Camera Setup

It is advisable to use Photoneo_StartAcquisition filter and setup inCaptureSettings with proper values. Some values might not be set to values the user expects and acquisition will fail or wait infinitely, when wrong trigger source or output type is selected.

Camera identification

When there is only one Photoneo camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one camera connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Photoneo PhoXi Control software.

Photoneo PhoXi Control can be downloaded from the following website: https://www.photoneo.com/3d-scanning-software/

Supported PhoXi Control version for Aurora Vision Studio usage is **1.2.35**.

Environment variable PHOXI_CONTROL_PATH which should point to the installation folder of PhoXi Control. PHO_LOG_FILES_DIR environment variable must contain a path to accessible folder. PHO_LOG_FILES_DIR is not set automatically after SDk installation. Manual configuration of this variable may be a requirement. Failure to meet these requirements may result in unexpected termination of the application.

Rebooting computer after SDK installation is mandatory.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Photoneo_StartAcquisition – Starts capturing data from a Photoneo camera.
- Photoneo_GrabPoint3DGrid – Captures PointCloud structure using Photoneo.

Photoneo_GrabDepthMap_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures DepthMap structure using Photoneo.

Syntax

```
bool avl::Photoneo_GrabDepthMap_WithTimeout
(
 Photoneo_State& ioState,
 const atl::Optional<atl::String> inDeviceID,
 const atl::Optional<int> inTimeout,
 const avl::PhotoneoCaptureSettings& inCaptureSettings,
 atl::Conditional<avl::Image>& outImage
)
```

Parameters

	Name	Type	Default	Description
	ioState	Photoneo_State&		Object used to maintain state of the function.
	inDeviceID	const Optional<String>	NIL	Device identification string
	inTimeout	const Optional<int>	NIL	Timeout [ms]
	inCaptureSettings	const PhotoneoCaptureSettings&		
	outImage	Conditional<Image>&		Output image

Remarks

PhoXi Control

PhoXi Control process needs to run for the filters to work.

Camera Setup

It is advisable to use Photoneo_StartAcquisition filter and setup inCaptureSettings with proper values. Some values might not be set to values the user expects and acquisition will fail or wait infinitely, when wrong trigger source or output type is selected.

Camera identification

When there is only one Photoneo camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one camera connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Photoneo PhoXi Control software.

Photoneo PhoXi Control can be downloaded from the following website: https://www.photoneo.com/3d-scanning-software/

Supported PhoXi Control version for Aurora Vision Studio usage is **1.2.35**.

Environment variable PHOXI_CONTROL_PATH which should point to the installation folder of PhoXi Control. PHO_LOG_FILES_DIR environment variable must contain a path to accessible folder. PHO_LOG_FILES_DIR is not set automatically after SDk installation. Manual configuration of this variable may be a requirement. Failure to meet these requirements may result in unexpected termination of the application.

Rebooting computer after SDK installation is mandatory.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Photoneo_StartAcquisition – Starts capturing data from a Photoneo camera.
- Photoneo_GrabPoint3DGrid – Captures PointCloud structure using Photoneo.

Photoneo_GrabNormals

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures NormalMap structure using Photoneo.

Syntax

```
bool avl::Photoneo_GrabNormals
(
 Photoneo_State& ioState,
 const atl::Optional<atl::String> inDeviceID,
 const avl::PhotoneoCaptureSettings& inCaptureSettings,
 avl::Point3DGrid& outGrid
)
```

Parameters

	Name	Type	Default	Description
	ioState	Photoneo_State&		Object used to maintain state of the function.
	inDeviceID	const Optional<String>	NIL	Device identification string
	inCaptureSettings	const PhotoneoCaptureSettings&		
	outGrid	Point3DGrid&		

Remarks

PhoXi Control

PhoXi Control process needs to run for the filters to work.

Camera Setup

It is advisable to use Photoneo_StartAcquisition filter and setup inCaptureSettings with proper values. Some values might not be set to values the user expects and acquisition will fail or wait infinitely, when wrong trigger source or output type is selected.

Camera identification

When there is only one Photoneo camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one camera connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Photoneo PhoXi Control software.

Photoneo PhoXi Control can be downloaded from the following website: https://www.photoneo.com/3d-scanning-software/

Supported PhoXi Control version for Aurora Vision Studio usage is **1.2.35**.

Environment variable PHOXI_CONTROL_PATH which should point to the installation folder of PhoXi Control. PHO_LOG_FILES_DIR environment variable must contain a path to accessible folder. PHO_LOG_FILES_DIR is not set automatically after SDk installation. Manual configuration of this variable may be a requirement. Failure to meet these requirements may result in unexpected termination of the application.

Rebooting computer after SDK installation is mandatory.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Photoneo_StartAcquisition – Starts capturing data from a Photoneo camera.
- Photoneo_GrabPoint3DGrid – Captures PointCloud structure using Photoneo.

Photoneo_GrabNormals_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures NormalMap structure using Photoneo.

Syntax

```
bool avl::Photoneo_GrabNormals_WithTimeout
(
 Photoneo_State& ioState,
 const atl::Optional<atl::String> inDeviceID,
 const atl::Optional<int> inTimeout,
 const avl::PhotoneoCaptureSettings& inCaptureSettings,
 atl::Conditional<avl::Point3DGrid>& outGrid
)
```

Parameters

	Name	Type	Default	Description
	ioState	Photoneo_State&		Object used to maintain state of the function.
	inDeviceID	const Optional<String>	NIL	Device identification string
	inTimeout	const Optional<int>	NIL	Timeout [ms]
	inCaptureSettings	const PhotoneoCaptureSettings&		
	outGrid	Conditional<Point3DGrid>&		

Remarks

PhoXi Control

PhoXi Control process needs to run for the filters to work.

Camera Setup

It is advisable to use Photoneo_StartAcquisition filter and setup inCaptureSettings with proper values. Some values might not be set to values the user expects and acquisition will fail or wait infinitely, when wrong trigger source or output type is selected.

Camera identification

When there is only one Photoneo camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one camera connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Photoneo PhoXi Control software.

Photoneo PhoXi Control can be downloaded from the following website: https://www.photoneo.com/3d-scanning-software/

Supported PhoXi Control version for Aurora Vision Studio usage is **1.2.35**.

Environment variable PHOXI_CONTROL_PATH which should point to the installation folder of PhoXi Control. PHO_LOG_FILES_DIR environment variable must contain a path to accessible folder. PHO_LOG_FILES_DIR is not set automatically after SDk installation. Manual configuration of this variable may be a requirement. Failure to meet these requirements may result in unexpected termination of the application.

Rebooting computer after SDK installation is mandatory.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Photoneo_StartAcquisition – Starts capturing data from a Photoneo camera.
- Photoneo_GrabPoint3DGrid – Captures PointCloud structure using Photoneo.

Photoneo_GrabPoint3DGrid

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures PointCloud structure using Photoneo.

Syntax

```
bool avl::Photoneo_GrabPoint3DGrid
(
 Photoneo_State& ioState,
 const atl::Optional<atl::String> inDeviceID,
 const avl::PhotoneoCaptureSettings& inCaptureSettings,
 avl::Point3DGrid& outGrid
)
```

Parameters

	Name	Type	Default	Description
	ioState	Photoneo_State&		Object used to maintain state of the function.
	inDeviceID	const Optional<String>	NIL	Device identification string
	inCaptureSettings	const PhotoneoCaptureSettings&		
	outGrid	Point3DGrid&		

Remarks

PhoXi Control

PhoXi Control process needs to run for the filters to work.

Camera Setup

It is advisable to use Photoneo_StartAcquisition filter and setup inCaptureSettings with proper values. Some values might not be set to values the user expects and acquisition will fail or wait infinitely, when wrong trigger source or output type is selected.

Camera identification

When there is only one Photoneo camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one camera connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Photoneo PhoXi Control software.

Photoneo PhoXi Control can be downloaded from the following website: https://www.photoneo.com/3d-scanning-software/

Supported PhoXi Control version for Aurora Vision Studio usage is **1.2.35**.

Environment variable PHOXI_CONTROL_PATH which should point to the installation folder of PhoXi Control. PHO_LOG_FILES_DIR environment variable must contain a path to accessible folder. PHO_LOG_FILES_DIR is not set automatically after SDk installation. Manual configuration of this variable may be a requirement. Failure to meet these requirements may result in unexpected termination of the application.

Rebooting computer after SDK installation is mandatory.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Photoneo_StartAcquisition – Starts capturing data from a Photoneo camera.
- Photoneo_GrabTexture – Captures Texture structure using Photoneo.

Photoneo_GrabPoint3DGrid_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures PointCloud structure using Photoneo.

Syntax

```
bool avl::Photoneo_GrabPoint3DGrid_WithTimeout
(
 Photoneo_State& ioState,
 const atl::Optional<atl::String> inDeviceID,
 const atl::Optional<int> inTimeout,
 const avl::PhotoneoCaptureSettings& inCaptureSettings,
 atl::Conditional<avl::Point3DGrid>& outGrid
)
```

Parameters

	Name	Type	Default	Description
	ioState	Photoneo_State&		Object used to maintain state of the function.
	inDeviceID	const Optional<String>	NIL	Device identification string
	inTimeout	const Optional<int>	NIL	Timeout [ms]
	inCaptureSettings	const PhotoneoCaptureSettings&		
	outGrid	Conditional<Point3DGrid>&		

Remarks

PhoXi Control

PhoXi Control process needs to run for the filters to work.

Camera Setup

It is advisable to use Photoneo_StartAcquisition filter and setup inCaptureSettings with proper values. Some values might not be set to values the user expects and acquisition will fail or wait infinitely, when wrong trigger source or output type is selected.

Camera identification

When there is only one Photoneo camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one camera connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Photoneo PhoXi Control software.

Photoneo PhoXi Control can be downloaded from the following website: https://www.photoneo.com/3d-scanning-software/

Supported PhoXi Control version for Aurora Vision Studio usage is **1.2.35**.

Environment variable PHOXI_CONTROL_PATH which should point to the installation folder of PhoXi Control. PHO_LOG_FILES_DIR environment variable must contain a path to accessible folder. PHO_LOG_FILES_DIR is not set automatically after SDk installation. Manual configuration of this variable may be a requirement. Failure to meet these requirements may result in unexpected termination of the application.

Rebooting computer after SDK installation is mandatory.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Photoneo_StartAcquisition – Starts capturing data from a Photoneo camera.
- Photoneo_GrabTexture – Captures Texture structure using Photoneo.

Photoneo_GrabTexture

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures Texture structure using Photoneo.

Syntax

```
bool avl::Photoneo_GrabTexture
(
 Photoneo_State& ioState,
 const atl::Optional<atl::String> inDeviceID,
 const avl::PhotoneoCaptureSettings& inCaptureSettings,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
	ioState	Photoneo_State&		Object used to maintain state of the function.
	inDeviceID	const Optional<String>	NIL	Device identification string
	inCaptureSettings	const PhotoneoCaptureSettings&		
	outImage	Image&		Output image

Remarks

PhoXi Control

PhoXi Control process needs to run for the filters to work.

Camera Setup

It is advisable to use Photoneo_StartAcquisition filter and setup inCaptureSettings with proper values. Some values might not be set to values the user expects and acquisition will fail or wait infinitely, when wrong trigger source or output type is selected.

Camera identification

When there is only one Photoneo camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one camera connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Photoneo PhoXi Control software.

Photoneo PhoXi Control can be downloaded from the following website: https://www.photoneo.com/3d-scanning-software/

Supported PhoXi Control version for Aurora Vision Studio usage is **1.2.35**.

Environment variable PHOXI_CONTROL_PATH which should point to the installation folder of PhoXi Control. PHO_LOG_FILES_DIR environment variable must contain a path to accessible folder. PHO_LOG_FILES_DIR is not set automatically after SDk installation. Manual configuration of this variable may be a requirement. Failure to meet these requirements may result in unexpected termination of the application.

Rebooting computer after SDK installation is mandatory.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Photoneo_StartAcquisition – Starts capturing data from a Photoneo camera.
- Photoneo_GrabPoint3DGrid – Captures PointCloud structure using Photoneo.

Photoneo_GrabTexture_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures Texture structure using Photoneo.

Syntax

```
bool avl::Photoneo_GrabTexture_WithTimeout
(
 Photoneo_State& ioState,
 const atl::Optional<atl::String> inDeviceID,
 const atl::Optional<int> inTimeout,
 const avl::PhotoneoCaptureSettings& inCaptureSettings,
 atl::Conditional<avl::Image>& outImage
)
```

Parameters

	Name	Type	Default	Description
	ioState	Photoneo_State&		Object used to maintain state of the function.
	inDeviceID	const Optional<String>	NIL	Device identification string
	inTimeout	const Optional<int>	NIL	Timeout [ms]
	inCaptureSettings	const PhotoneoCaptureSettings&		
	outImage	Conditional<Image>&		Output image

Remarks

PhoXi Control

PhoXi Control process needs to run for the filters to work.

Camera Setup

It is advisable to use Photoneo_StartAcquisition filter and setup inCaptureSettings with proper values. Some values might not be set to values the user expects and acquisition will fail or wait infinitely, when wrong trigger source or output type is selected.

Camera identification

When there is only one Photoneo camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one camera connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Photoneo PhoXi Control software.

Photoneo PhoXi Control can be downloaded from the following website: https://www.photoneo.com/3d-scanning-software/

Supported PhoXi Control version for Aurora Vision Studio usage is **1.2.35**.

Environment variable PHOXI_CONTROL_PATH which should point to the installation folder of PhoXi Control. PHO_LOG_FILES_DIR environment variable must contain a path to accessible folder. PHO_LOG_FILES_DIR is not set automatically after SDk installation. Manual configuration of this variable may be a requirement. Failure to meet these requirements may result in unexpected termination of the application.

Rebooting computer after SDK installation is mandatory.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Photoneo_StartAcquisition – Starts capturing data from a Photoneo camera.
- Photoneo_GrabPoint3DGrid – Captures PointCloud structure using Photoneo.

Photoneo_SetParameters

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets the camera parameters.

Syntax

```
void avl::Photoneo_SetParameters
(
 Photoneo_State& ioState,
 const atl::Optional<atl::String> inDeviceID,
 const avl::PhotoneoCaptureSettings& inCaptureSettings
)
```

Parameters

	Name	Type	Default	Description
	ioState	Photoneo_State&		Object used to maintain state of the function.
	inDeviceID	const Optional<String>	NIL	Device identification string
	inCaptureSettings	const PhotoneoCaptureSettings&		

Remarks

PhoXi Control

PhoXi Control process needs to run for the filters to work.

Camera Setup

It is advisable to use Photoneo_StartAcquisition filter and setup inCaptureSettings with proper values. Some values might not be set to values the user expects and acquisition will fail or wait infinitely, when wrong trigger source or output type is selected.

Camera identification

When there is only one Photoneo camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one camera connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Photoneo PhoXi Control software.

Photoneo PhoXi Control can be downloaded from the following website: https://www.photoneo.com/3d-scanning-software/

Supported PhoXi Control version for Aurora Vision Studio usage is **1.2.35**.

Environment variable PHOXI_CONTROL_PATH which should point to the installation folder of PhoXi Control. PHO_LOG_FILES_DIR environment variable must contain a path to accessible folder. PHO_LOG_FILES_DIR is not set automatically after SDk installation. Manual configuration of this variable may be a requirement. Failure to meet these requirements may result in unexpected termination of the application.

Rebooting computer after SDK installation is mandatory.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Photoneo_GetParameters – Gets the camera parameters.
- Photoneo_GrabPoint3DGrid – Captures PointCloud structure using Photoneo.

Photoneo_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Starts capturing data from a Photoneo camera.

Applications: Typically used for establishing camera connectivity before the first trigger event. Especially important for multiple-camera systems.

Syntax

```
void avl::Photoneo_StartAcquisition
(
 Photoneo_State& ioState,
 const atl::Optional<atl::String> inDeviceID,
 const avl::PhotoneoCaptureSettings& inCaptureSettings
)
```

Parameters

	Name	Type	Default	Description
	ioState	Photoneo_State&		Object used to maintain state of the function.
	inDeviceID	const Optional<String>	NIL	Device identification string
	inCaptureSettings	const PhotoneoCaptureSettings&		

Remarks

PhoXi Control

PhoXi Control process needs to run for the filters to work.

Camera Setup

It is advisable to use Photoneo_StartAcquisition filter and setup inCaptureSettings with proper values. Some values might not be set to values the user expects and acquisition will fail or wait infinitely, when wrong trigger source or output type is selected.

Camera identification

When there is only one Photoneo camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one camera connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Photoneo PhoXi Control software.

Photoneo PhoXi Control can be downloaded from the following website: https://www.photoneo.com/3d-scanning-software/

Supported PhoXi Control version for Aurora Vision Studio usage is **1.2.35**.

Environment variable PHOXI_CONTROL_PATH which should point to the installation folder of PhoXi Control. PHO_LOG_FILES_DIR environment variable must contain a path to accessible folder. PHO_LOG_FILES_DIR is not set automatically after SDk installation. Manual configuration of this variable may be a requirement. Failure to meet these requirements may result in unexpected termination of the application.

Rebooting computer after SDK installation is mandatory.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Photoneo_StopAcquisition – Starts capturing data from a Photoneo camera.
- Photoneo_GrabPoint3DGrid – Captures PointCloud structure using Photoneo.

Photoneo_StopAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Starts capturing data from a Photoneo camera.

Applications: Typically used for establishing camera connectivity before the first trigger event. Especially important for multiple-camera systems.

Syntax

```
void avl::Photoneo_StopAcquisition
(
 Photoneo_State& ioState,
 const atl::Optional<atl::String> inDeviceID
)
```

Parameters

	Name	Type	Default	Description
	ioState	Photoneo_State&		Object used to maintain state of the function.
	inDeviceID	const Optional<String>	NIL	Device identification string

Remarks

PhoXi Control

PhoXi Control process needs to run for the filters to work.

Camera Setup

It is advisable to use Photoneo_StartAcquisition filter and setup inCaptureSettings with proper values. Some values might not be set to values the user expects and acquisition will fail or wait infinitely, when wrong trigger source or output type is selected.

Camera identification

When there is only one Photoneo camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one camera connected to the computer.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Photoneo PhoXi Control software.

Photoneo PhoXi Control can be downloaded from the following website: https://www.photoneo.com/3d-scanning-software/

Supported PhoXi Control version for Aurora Vision Studio usage is **1.2.35**.

Environment variable PHOXI_CONTROL_PATH which should point to the installation folder of PhoXi Control. PHO_LOG_FILES_DIR environment variable must contain a path to accessible folder. PHO_LOG_FILES_DIR is not set automatically after SDk installation. Manual configuration of this variable may be a requirement. Failure to meet these requirements may result in unexpected termination of the application.

Rebooting computer after SDK installation is mandatory.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Photoneo_StartAcquisition – Starts capturing data from a Photoneo camera.
- Photoneo_GrabPoint3DGrid – Captures PointCloud structure using Photoneo.

180. Basler

Table of content:

Pylon_ExecuteCommand

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Executes command in Basler device.

Syntax

```
void avl::Pylon_ExecuteCommand
(
 Pylon_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inCommandName
)
```

Parameters

	Name	Type	Default	Description
	ioState	Pylon_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inCommandName	const String&		Name of feature parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Pylon_GenerateSoftwareTrigger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Generates software trigger.

Syntax

```
void avl::Pylon_GenerateSoftwareTrigger
(
 Pylon_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 bool inIgnoreWaiting,
 atl::Optional<int> inTimeout,
 bool& outExecuted
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Pylon_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Serial number, model name, user defined name or full name of camera device to be opened
	inIgnoreWaiting	bool			Waiting for previous trigger doesn't work with all cameras. E.g. for models A600 this value should be set to true.
	inTimeout	Optional<int>	100 - ∞	NIL	
	outExecuted	bool&			Use only if inTimeout is set

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Pylon_GetBoolParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type Bool.

Syntax

```
void avl::Pylon_GetBoolParameter
(
 Pylon_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 bool& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Pylon_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inParameterName	const String&		Name of feature parameter
	outValue	bool&		Value retrieved from device parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Pylon_GetEnumParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type Enumeration.

Syntax

```
void avl::Pylon_GetEnumParameter
(
 Pylon_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 atl::String& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Pylon_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inParameterName	const String&		Name of feature parameter
	outValue	String&		Value retrieved from device parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Pylon_GetGigEStatistics

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets GigeVision device statistics data.

Syntax

```
void avl::Pylon_GetGigEStatistics
(
 Pylon_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 atl::int64& outTotalBufferCount,
 atl::int64& outFailedBufferCount,
 atl::int64& outBufferUnderrunCount,
 atl::int64& outTotalPacketCount,
 atl::int64& outFailedPacketCount,
 atl::int64& outResendRequestCount,
 atl::int64& outResendPacketCount,
 atl::int64& outNumEmptyBuffers,
 atl::int64& outNumQueuedBuffers,
 atl::int64& outNumReadyBuffers,
 atl::int64& outOutputQueueSize
)
```

Parameters

	Name	Type	Default	Description
	ioState	Pylon_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	outTotalBufferCount	int64&		Counts the number of received frames
	outFailedBufferCount	int64&		Counts the number of buffers with at least one failed packet (status != success)
	outBufferUnderrunCount	int64&		Counts the number of frames lost because there were no buffers queued to the driver
	outTotalPacketCount	int64&		Counts the number of received packets
	outFailedPacketCount	int64&		Counts the number of failed packets (status != success)
	outResendRequestCount	int64&		Counts the number of emitted PACKETRESEND commands
	outResendPacketCount	int64&		Counts the number of packets requested by PACKETRESEND commands
	outNumEmptyBuffers	int64&		The number of empty buffers that are not used for grabbing yet
	outNumQueuedBuffers	int64&		The number of buffers queued at Low Level API stream grabber
	outNumReadyBuffers	int64&		The number of grab result buffers in the output queue that are ready for retrieval
	outOutputQueueSize	int64&		The size of the grab result buffer output queue

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Pylon_GetIntegerParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type Integer.

Syntax

```
void avl::Pylon_GetIntegerParameter
(
 Pylon_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 int& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Pylon_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inParameterName	const String&		Name of feature parameter
	outValue	int&		Value retrieved from device parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Pylon_GetRealParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type Real.

Syntax

```
void avl::Pylon_GetRealParameter
(
 Pylon_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 float& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Pylon_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inParameterName	const String&		Name of feature parameter
	outValue	float&		Value retrieved from device parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Pylon_GetStringParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type String.

Syntax

```
void avl::Pylon_GetStringParameter
(
 Pylon_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 atl::String& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Pylon_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inParameterName	const String&		Name of feature parameter
	outValue	String&		Value retrieved from device parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Pylon_GetUsbStatistics

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets GigeVision device statistics data.

Syntax

```
void avl::Pylon_GetUsbStatistics
(
 Pylon_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 atl::int64& outTotalBufferCount,
 atl::int64& outFailedBufferCount,
 atl::int64& outLastFailedBufferStatus,
 atl::String& outLastFailedBufferStatusText,
 atl::int64& outMissedFrameCount,
 atl::int64& outResynchronizationCount,
 atl::int64& outLastBlockId,
 atl::int64& outNumEmptyBuffers,
 atl::int64& outNumQueuedBuffers,
 atl::int64& outNumReadyBuffers,
 atl::int64& outOutputQueueSize
)
```

Parameters

	Name	Type	Default	Description
	ioState	Pylon_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	outTotalBufferCount	int64&		The total count of processed buffers
	outFailedBufferCount	int64&		The count of buffers that returned with an error status
	outLastFailedBufferStatus	int64&		The status code of the last failed buffer
	outLastFailedBufferStatusText	String&		The message text of the status code of the last failed buffer
	outMissedFrameCount	int64&		The count of bad or missed frames between successfully grabbed images
	outResynchronizationCount	int64&		The count of stream resynchronizations
	outLastBlockId	int64&		The last grabbed block ID
	outNumEmptyBuffers	int64&		The number of empty buffers that are not used for grabbing yet
	outNumQueuedBuffers	int64&		The number of buffers queued at Low Level API stream grabber
	outNumReadyBuffers	int64&		The number of grab result buffers in the output queue that are ready for retrieval
	outOutputQueueSize	int64&		The size of the grab result buffer output queue

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Pylon_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an image stream from a camera using Pylon library.

Syntax

```
bool avl::Pylon_GrabImage
(
 Pylon_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inOutputQueueSize,
 atl::Optional<avl::PylonImageFormat::Type> inPixelFormat,
 atl::Optional<float> inFrameRate,
 atl::Optional<const avl::Box&> inAoi,
 atl::Optional<float> inExposureTime,
 atl::Optional<float> inGain,
 atl::Optional<float> inBlackLevel,
 atl::Optional<bool> inTriggerEnabled,
 atl::Optional<avl::PylonTriggerSource::Type> inTriggerSource,
 atl::Optional<avl::PylonTriggerActivation::Type> inTriggerActivation,
 avl::Image& outImage,
 atl::int64& outFrameID,
 atl::int64& outTimestamp
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Pylon_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Serial number, model name, user defined name or full name of camera device to be opened
	inOutputQueueSize	int	1 - 200	3	Capacity of output frames queue
	inPixelFormat	Optional<PylonImageFormat::Type>		NIL	Image pixel format
	inFrameRate	Optional<float>	0.1 - 400.0	NIL	Requested camera frame rate in frames per second
	inAoi	Optional<const Box&>		NIL	Required fragment of image to stream. To reset AOI set Box(0,0,0,0).
	inExposureTime	Optional<float>	0.0 - ∞	NIL	Camera frame exposition time
	inGain	Optional<float>	0.0 - ∞	NIL	Analog gain of source image in device raw unit
	inBlackLevel	Optional<float>	0.0 - ∞	NIL	Black level of source image
	inTriggerEnabled	Optional<bool>		NIL	Configure trigger enable
	inTriggerSource	Optional<PylonTriggerSource::Type>		NIL	Source of acquisition trigger
	inTriggerActivation	Optional<PylonTriggerActivation::Type>		NIL	Circumstances defining when the trigger is activated
	outImage	Image&			Captured frame
	outFrameID	int64&			Captured frame ID
	outTimestamp	int64&			Captured frame timestamp

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install Pylon SDK software with camera dedicated drivers.

Pylon SDK can be downloaded from the following website: https://www.baslerweb.com/ (registration may be required).

Recommended Pylon SDK version for Aurora Vision Studio usage is **7.1**.

Other SDK versions from the same family (for example 7.1.1) can be compatible with Aurora Vision Studio but correct operation is not guaranteed.

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to computer. DeviceID can be set to:

- **Serial number** - should be specified on device casing, for example "12345678".
- **User defined name** - user defined device name.
- **Friendly name** - human readable name of the device.
- **Device full name** - full name identifying the device.
- **Model name** - model name of the device.
- **Ip Address** - camera IP address. Only for GigEVision cameras.

Camera parameters

Setting **inAoi** parameter to 'Auto' will select maximum possible camera frame size.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

To change other, more advanced camera parameters, use configuration tool "Pylon Viewer" available with Pylon SDK or Pylon_SetXXXParameter filters. Refer to Pylon documentation to find information about parameters and how to create user parameters sets.

Output outFrameId

- **IEEE 1394 Camera Devices** - The value of block ID is always -1.
- **GigE Camera Devices** - The sequence number starts with 1 and wraps at 65535. The value 0 has a special meaning and indicates that this feature is not supported by the camera.
- **USB Camera Devices** - The sequence number starts with 0 and uses the full 64 Bit range.

A block ID of value -1 indicates that the Block ID is invalid and must not be used.

Output outTimeStamp

In case of GigE-Vision this describes when the image exposure was started. Cameras that do not support this feature return zero. If supported this may be used to determine which ROIs were acquired simultaneously.

In case of FireWire this value describes the cycle time when the first packet arrives.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Pylon_GrabImage_WithTimeout – Captures an image stream from a camera using Pylon library; returns Nil if no frame comes in the specified time.
- Pylon_StartAcquisition – Initializes and starts image acquisition in a camera.

Pylon_GrabImage_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an image stream from a camera using Pylon library; returns Nil if no frame comes in the specified time.

Syntax

```
bool avl::Pylon_GrabImage_WithTimeout
(
 Pylon_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inOutputQueueSize,
 int inTimeout,
 atl::Optional<avl::PylonImageFormat::Type> inPixelFormat,
 atl::Optional<float> inFrameRate,
 atl::Optional<const avl::Box&> inAoi,
 atl::Optional<float> inExposureTime,
 atl::Optional<float> inGain,
 atl::Optional<float> inBlackLevel,
 atl::Optional<bool> inTriggerEnabled,
 atl::Optional<avl::PylonTriggerSource::Type> inTriggerSource,
 atl::Optional<avl::PylonTriggerActivation::Type> inTriggerActivation,
 atl::Conditional<avl::Image>& outImage,
 atl::Conditional<atl::int64>& outFrameID,
 atl::Conditional<atl::int64>& outTimestamp
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Pylon_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Serial number, model name, user defined name or full name of camera device to be opened
	inOutputQueueSize	int	1 - 200	3	Capacity of output frames queue
	inTimeout	int	100 - ∞		Maximum time to wait for frame in milliseconds
	inPixelFormat	Optional<PylonImageFormat::Type>		NIL	Image pixel format
	inFrameRate	Optional<float>	0.1 - 400.0	NIL	Requested camera frame rate in frames per second
	inAoi	Optional<const Box&>		NIL	Required fragment of image to stream. To reset AOI set Box(0,0,0,0)
	inExposureTime	Optional<float>	0.0 - ∞	NIL	Camera frame exposition time
	inGain	Optional<float>	0.0 - ∞	NIL	Analog gain of source image in device raw unit
	inBlackLevel	Optional<float>	0.0 - ∞	NIL	Black level of source image
	inTriggerEnabled	Optional<bool>		NIL	Configure trigger enable
	inTriggerSource	Optional<PylonTriggerSource::Type>		NIL	Source of acquisition trigger
	inTriggerActivation	Optional<PylonTriggerActivation::Type>		NIL	Circumstances defining when the trigger is activated
	outImage	Conditional<Image>&			Captured frame
	outFrameID	Conditional<int64>&			Captured frame ID
	outTimestamp	Conditional<int64>&			Captured frame timestamp

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install Pylon SDK software with camera dedicated drivers.

Pylon SDK can be downloaded from the following website: https://www.baslerweb.com/ (registration may be required).

Recommended Pylon SDK version for Aurora Vision Studio usage is **7.1**.

Other SDK versions from the same family (for example 7.1.1) can be compatible with Aurora Vision Studio but correct operation is not guaranteed.

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to computer. DeviceID can be set to:

- **Serial number** - should be specified on device casing, for example "12345678".
- **User defined name** - user defined device name.
- **Friendly name** - human readable name of the device.
- **Device full name** - full name identifying the device.
- **Model name** - model name of the device.
- **Ip Address** - camera IP address. Only for GigEVision cameras.

Camera parameters

Setting **inAoi** parameter to 'Auto' will select maximum possible camera frame size.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

To change other, more advanced camera parameters, use configuration tool "Pylon Viewer" available with Pylon SDK or Pylon_SetXXXParameter filters. Refer to Pylon documentation to find information about parameters and how to create user parameters sets.

Output outFrameId

- **IEEE 1394 Camera Devices** - The value of block ID is always -1.
- **GigE Camera Devices** - The sequence number starts with 1 and wraps at 65535. The value 0 has a special meaning and indicates that this feature is not supported by the camera.
- **USB Camera Devices** - The sequence number starts with 0 and uses the full 64 Bit range.

A block ID of value -1 indicates that the Block ID is invalid and must not be used.

Output outTimeStamp

In case of GigE-Vision this describes when the image exposure was started. Cameras that do not support this feature return zero. If supported this may be used to determine which ROIs were acquired simultaneously.

In case of FireWire this value describes the cycle time when the first packet arrives.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Pylon_GrabImage – Captures an image stream from a camera using Pylon library.
- Pylon_StartAcquisition – Initializes and starts image acquisition in a camera.

Pylon_SetBoolParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type Bool.

Syntax

```
void avl::Pylon_SetBoolParameter
(
 Pylon_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 bool inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Pylon_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inParameterName	const String&		Name of feature parameter
	inValue	bool		New value to be set into device parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Pylon_SetEnumParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type Enumeration.

Syntax

```
void avl::Pylon_SetEnumParameter
(
 Pylon_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 const atl::String& inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Pylon_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inParameterName	const String&		Name of feature parameter
	inValue	const String&		New value to be set into device parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Pylon_SetIntegerParameter

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type Integer.

Syntax

```
void avl::Pylon_SetIntegerParameter
(
 Pylon_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 int inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Pylon_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inParameterName	const String&		Name of feature parameter
	inValue	int		New value to be set into device parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Pylon_SetRealParameter

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type Real.

Syntax

```
void avl::Pylon_SetRealParameter
(
 Pylon_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 float inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Pylon_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inParameterName	const String&		Name of feature parameter
	inValue	float		New value to be set into device parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Pylon_SetStringParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type String.

Syntax

```
void avl::Pylon_SetStringParameter
(
 Pylon_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 const atl::String& inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Pylon_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inParameterName	const String&		Name of feature parameter
	inValue	const String&		New value to be set into device parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Pylon_SetUserOutput

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets Basler camera user output.

Syntax

```
void avl::Pylon_SetUserOutput
(
 Pylon_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::PylonUserOutput::Type inUserOutput,
 bool inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Pylon_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Serial number, model name, user defined name or full name of camera device to be opened
	inUserOutput	PylonUserOutput::Type		
	inValue	bool		

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Pylon_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Initializes and starts image acquisition in a camera.

Syntax

```
void avl::Pylon_StartAcquisition
(
 Pylon_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inOutputQueueSize,
 atl::Optional<avl::PylonImageFormat::Type> inPixelFormat,
 atl::Optional<float> inFrameRate,
 atl::Optional<const avl::Box&> inAoi,
 atl::Optional<float> inExposureTime,
 atl::Optional<float> inGain,
 atl::Optional<float> inBlackLevel,
 atl::Optional<bool> inTriggerEnabled,
 atl::Optional<avl::PylonTriggerSource::Type> inTriggerSource,
 atl::Optional<avl::PylonTriggerActivation::Type> inTriggerActivation
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Pylon_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Serial number, model name, user defined name or full name of camera device to be opened
	inOutputQueueSize	int	1 - 200	3	Capacity of output frames queue
	inPixelFormat	Optional<PylonImageFormat::Type>		NIL	Image pixel format
	inFrameRate	Optional<float>	0.1 - 400.0	NIL	Requested camera frame rate in frames per second
	inAoi	Optional<const Box&>		NIL	Required fragment of image to stream. To reset AOI set Box(0,0,0,0).
	inExposureTime	Optional<float>	0.0 - ∞	NIL	Camera frame exposition time
	inGain	Optional<float>	0.0 - ∞	NIL	Analog gain of source image in device raw unit
	inBlackLevel	Optional<float>	0.0 - ∞	NIL	Black level of source image
	inTriggerEnabled	Optional<bool>		NIL	Configure trigger enable
	inTriggerSource	Optional<PylonTriggerSource::Type>		NIL	Source of acquisition trigger
	inTriggerActivation	Optional<PylonTriggerActivation::Type>		NIL	Circumstances defining when the trigger is activated

Remarks

This filter is intended for establishing connection with a Basler camera device using Pylon interface, to initialize image streaming. It is only needed when explicit image acquisition start is required in the initial phase of a program. For example, it can be used to prepare a camera, running in triggered mode, to be able to capture trigger signals before the first invoke of Pylon_GrabImage or to start multiple cameras in sync before the acquisition phase.

The use of this filter is not obligatory. Pylon_GrabImage or Pylon_GrabImage_WithTimeout] filters will initialize and start image acquisition upon their first invoke. When this filter is used, the **inPixelFormat** parameter of subsequent Pylon_GrabImage and Pylon_GrabImage_WithTimeout] filters has no effect.

This filter has no effect when invoked for the second time and when invoked after image grabbing filters.

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install Pylon SDK software with camera dedicated drivers.

Pylon SDK can be downloaded from the following website: https://www.baslerweb.com/ (registration may be required).

Recommended Pylon SDK version for Aurora Vision Studio usage is **7.1**.

Other SDK versions from the same family (for example 7.1.1) can be compatible with Aurora Vision Studio but correct operation is not guaranteed.

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to computer. DeviceID can be set to:

- **Serial number** - should be specified on device casing, for example "12345678".
- **User defined name** - user defined device name.
- **Friendly name** - human readable name of the device.
- **Device full name** - full name identifying the device.
- **Model name** - model name of the device.
- **Ip Address** - camera IP address. Only for GigEVision cameras.

Camera parameters

Setting **inAoi** parameter to 'Auto' will select maximum possible camera frame size.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

To change other, more advanced camera parameters, use configuration tool "Pylon Viewer" available with Pylon SDK or Pylon_SetXXXParameter filters. Refer to Pylon documentation to find information about parameters and how to create user parameters sets.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Pylon_GrabImage – Captures an image stream from a camera using Pylon library.
- Pylon_GrabImage_WithTimeout – Captures an image stream from a camera using Pylon library; returns Nil if no frame comes in the specified time.

Pylon_StopAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Stops image acquisition in a camera.

Syntax

```
void avl::Pylon_StopAcquisition
(
 Pylon_State& ioState,
 atl::Optional<const atl::String&> inDeviceID
)
```

Parameters

	Name	Type	Default	Description
	ioState	Pylon_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Serial number, model name, user defined name or full name of camera device to be opened

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

181. Roseek

Table of content:

Roseek_ConfigureLEDDriverMode

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets the driver working mode of LED.

Syntax

```
void avl::Roseek_ConfigureLEDDriverMode
(
 Roseek_State& ioState,
 avl::RoseekLEDDriverType::Type inType,
 avl::RoseekLEDDriverMode::Type inMode
)
```

Parameters

	Name	Type	Default	Description
	ioState	Roseek_State&		Object used to maintain state of the function.
	inType	RoseekLEDDriverType::Type		Driver type (own or external)
	inMode	RoseekLEDDriverMode::Type		LED driver mode

Remarks

Camera driver software

This filter is intended to cooperate with camera using its vendor Roseek SDK. To be able to connect to camera it is required to install Roseek SDK software with camera dedicated drivers. Currently Aurora Vision uses **Roseek version v1.32**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Roseek_GenerateSoftwareTrigger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Generates software trigger.

Syntax

```
void avl::Roseek_GenerateSoftwareTrigger
(
 Roseek_State& ioState
)
```

Parameters

	Name	Type	Default	Description
	ioState	Roseek_State&		Object used to maintain state of the function.

Remarks

Camera driver software

This filter is intended to cooperate with camera using its vendor Roseek SDK. To be able to connect to camera it is required to install Roseek SDK software with camera dedicated drivers. Currently Aurora Vision uses **Roseek version v1.32**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Roseek_GetInputState

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets input I/O state.

Syntax

```
void avl::Roseek_GetInputState
(
 Roseek_State& ioState,
 int inId,
 bool& outState
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Roseek_State&			Object used to maintain state of the function.
	inId	int	0 - 7		Index of I/O output port
	outState	bool&			Received state

Remarks

Camera driver software

This filter is intended to cooperate with camera using its vendor Roseek SDK. To be able to connect to camera it is required to install Roseek SDK software with camera dedicated drivers. Currently Aurora Vision uses **Roseek version v1.32**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Roseek_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures images from a Roseek device.

Syntax

```
bool avl::Roseek_GrabImage
(
 Roseek_State& ioState,
 int inInputQueueSize,
 avl::RoseekImageFormat::Type inImageFormat,
 atl::Optional<avl::RoseekResolutionMode::Type> inResolutionMode,
 const atl::Optional<avl::Box>& inROI,
 atl::Optional<float> inFrameRate,
 atl::Optional<avl::RoseekWorkingMode::Type> inWorkingMode,
 atl::Optional<int> inSensitivityLevel,
 atl::Optional<avl::RoseekExposureMode::Type> inExposureMode,
 atl::Optional<int> inExposureTime,
 atl::Optional<float> inGain,
 avl::Image& outImage,
 int& outFrameID
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Roseek_State&			Object used to maintain state of the function.
	inInputQueueSize	int	1 - 1000	3	Number of incoming frames that can be buffered before the application is able to process them
	inImageFormat	RoseekImageFormat::Type			Image pixel format
	inResolutionMode	Optional<RoseekResolutionMode::Type>		NIL	Set resolution of image
	inROI	const Optional<Box>&		NIL	Set resolution region. Has effect only if resolution mode is ROI.
	inFrameRate	Optional<float>	0.1 - 400.0	NIL	Requested camera frame rate in frames per second
	inWorkingMode	Optional<RoseekWorkingMode::Type>		NIL	Working mode of image acquisition
	inSensitivityLevel	Optional<int>	0 - 3	NIL	Sensitivity level of camera sensor
	inExposureMode	Optional<RoseekExposureMode::Type>		NIL	Exposure mode, should be set to Manual if you want to adjust inExposureTime manually
	inExposureTime	Optional<int>	0 - ∞	NIL	Camera frame exposition time
	inGain	Optional<float>	0.0 - 36.0	NIL	Camera exposure gain
	outImage	Image&			Captured frame
	outFrameID	int&			Captured frame ID

Remarks

Camera driver software

This filter is intended to cooperate with camera using its vendor Roseek SDK. To be able to connect to camera it is required to install Roseek SDK software with camera dedicated drivers. Currently Aurora Vision uses **Roseek version v1.32**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Roseek_GrabImage_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures images from a Roseek device.

Syntax

```
bool avl::Roseek_GrabImage_WithTimeout
(
 Roseek_State& ioState,
 int inTimeout,
 int inInputQueueSize,
 avl::RoseekImageFormat::Type inImageFormat,
 atl::Optional<avl::RoseekResolutionMode::Type> inResolutionMode,
 const atl::Optional<avl::Box>& inROI,
 atl::Optional<float> inFrameRate,
 atl::Optional<avl::RoseekWorkingMode::Type> inWorkingMode,
 atl::Optional<int> inSensitivityLevel,
 atl::Optional<avl::RoseekExposureMode::Type> inExposureMode,
 atl::Optional<int> inExposureTime,
 atl::Optional<float> inGain,
 atl::Conditional<avl::Image>& outImage,
 atl::Conditional<int>& outFrameID
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Roseek_State&			Object used to maintain state of the function.
	inTimeout	int	10 - 3600000		Maximum time to wait for frame in milliseconds
	inInputQueueSize	int	1 - 1000	3	Number of incoming frames that can be buffered before the application is able to process them
	inImageFormat	RoseekImageFormat::Type			Image pixel format
	inResolutionMode	Optional<RoseekResolutionMode::Type>		NIL	Set resolution of image
	inROI	const Optional<Box>&		NIL	Set resolution region. Has effect only if resolution mode is ROI.
	inFrameRate	Optional<float>	0.1 - 400.0	NIL	Requested camera frame rate in frames per second
	inWorkingMode	Optional<RoseekWorkingMode::Type>		NIL	Working mode of image acquisition
	inSensitivityLevel	Optional<int>	0 - 3	NIL	Sensitivity level of camera sensor
	inExposureMode	Optional<RoseekExposureMode::Type>		NIL	Exposure mode, should be set to Manual if you want to adjust inExposureTime manually
	inExposureTime	Optional<int>	0 - ∞	NIL	Camera frame exposition time
	inGain	Optional<float>	0.0 - 36.0	NIL	Camera exposure gain
	outImage	Conditional<Image>&			Captured frame
	outFrameID	Conditional<int>&			Captured frame ID

Remarks

Camera driver software

This filter is intended to cooperate with camera using its vendor Roseek SDK. To be able to connect to camera it is required to install Roseek SDK software with camera dedicated drivers. Currently Aurora Vision uses **Roseek version v1.32**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Roseek_SetLEDDriverStrength

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets the driver strength of LED.

Syntax

```
void avl::Roseek_SetLEDDriverStrength
(
 Roseek_State& ioState,
 avl::RoseekLEDDriverType::Type inType,
 int inStrength
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Roseek_State&			Object used to maintain state of the function.
	inType	RoseekLEDDriverType::Type			Driver type (own or external)
	inStrength	int	0 - 1500		Strength. For OWN driver type the max value is 330 otherwise 1500

Remarks

Camera driver software

This filter is intended to cooperate with camera using its vendor Roseek SDK. To be able to connect to camera it is required to install Roseek SDK software with camera dedicated drivers. Currently Aurora Vision uses **Roseek version v1.32**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Roseek_SetOutputState

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets output I/O state.

Syntax

```
void avl::Roseek_SetOutputState
(
 Roseek_State& ioState,
 int inId,
 bool inState
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Roseek_State&			Object used to maintain state of the function.
	inId	int	0 - 7		Index of I/O output port
	inState	bool			true = TURN_ON, false = TURN_OFF

Remarks

Camera driver software

This filter is intended to cooperate with camera using its vendor Roseek SDK. To be able to connect to camera it is required to install Roseek SDK software with camera dedicated drivers. Currently Aurora Vision uses **Roseek version v1.32**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Roseek_SetStatusLED

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets the status of programmable LED state indicators supported on the camera.

Syntax

```
void avl::Roseek_SetStatusLED
(
 Roseek_State& ioState,
 int inId,
 avl::RoseekLEDStatus::Type inState
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Roseek_State&			Object used to maintain state of the function.
	inId	int	0 - 2		Index of I/O output port
	inState	RoseekLEDStatus::Type			Received state

Remarks

Camera driver software

This filter is intended to cooperate with camera using its vendor Roseek SDK. To be able to connect to camera it is required to install Roseek SDK software with camera dedicated drivers. Currently Aurora Vision uses **Roseek version v1.32**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Roseek_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Initializes and starts image acquisition.

Syntax

```
void avl::Roseek_StartAcquisition
(
 Roseek_State& ioState,
 int inInputQueueSize,
 avl::RoseekImageFormat::Type inImageFormat,
 atl::Optional<avl::RoseekResolutionMode::Type> inResolutionMode,
 const atl::Optional<avl::Box>& inROI,
 atl::Optional<float> inFrameRate,
 atl::Optional<avl::RoseekWorkingMode::Type> inWorkingMode,
 atl::Optional<int> inSensitivityLevel,
 atl::Optional<avl::RoseekExposureMode::Type> inExposureMode,
 atl::Optional<int> inExposureTime,
 atl::Optional<float> inGain
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Roseek_State&			Object used to maintain state of the function.
	inInputQueueSize	int	1 - 1000	3	Number of incoming frames that can be buffered before the application is able to process them
	inImageFormat	RoseekImageFormat::Type			Image pixel format
	inResolutionMode	Optional<RoseekResolutionMode::Type>		NIL	Set resolution of image
	inROI	const Optional<Box>&		NIL	Set resolution region. Has effect only if resolution mode is ROI
	inFrameRate	Optional<float>	0.1 - 400.0	NIL	Requested camera frame rate in frames per second
	inWorkingMode	Optional<RoseekWorkingMode::Type>		NIL	Working mode of image acquisition
	inSensitivityLevel	Optional<int>	0 - 3	NIL	Sensitivity level of camera sensor
	inExposureMode	Optional<RoseekExposureMode::Type>		NIL	Exposure mode, should be set to Manual if you want to adjust inExposureTime manually
	inExposureTime	Optional<int>	0 - ∞	NIL	Camera frame exposition time
	inGain	Optional<float>	0.0 - 36.0	NIL	Camera exposure gain

Remarks

Camera driver software

This filter is intended to cooperate with camera using its vendor Roseek SDK. To be able to connect to camera it is required to install Roseek SDK software with camera dedicated drivers. Currently Aurora Vision uses **Roseek version v1.32**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Roseek_StopAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Stops image acquisition.

Syntax

```
void avl::Roseek_StopAcquisition
(
 Roseek_State& ioState
)
```

Parameters

	Name	Type	Default	Description
	ioState	Roseek_State&		Object used to maintain state of the function.

Remarks

Camera driver software

This filter is intended to cooperate with camera using its vendor Roseek SDK. To be able to connect to camera it is required to install Roseek SDK software with camera dedicated drivers. Currently Aurora Vision uses **Roseek version v1.32**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

182. SiliconSoftware

Table of content:

SiliconSoftware_GenerateSoftwareTrigger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Generates software trigger. This function depends on current applet.

Syntax

```
void avl::SiliconSoftware_GenerateSoftwareTrigger
(
 SiliconSoftware_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	SiliconSoftware_State&			Object used to maintain state of the function.
→	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
→	inPort	int	0 - ∞		DMA port

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

SiliconSoftware_GetDoubleParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type Double.

Syntax

```
void avl::SiliconSoftware_GetDoubleParameter
(
 SiliconSoftware_State& ioState,
 atl::Optional<int> inDeviceIndex,
 atl::Optional<const atl::String&> inAppletPath,
 int inPort,
 const atl::String& inName,
 double& outValue
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	SiliconSoftware_State&			Object used to maintain state of the function.
→	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
→	inAppletPath	Optional<const String&>		NIL	Applet to load
→	inPort	int	0 - ∞		DMA port
→	inName	const String&			Parameter name
←	outValue	double&			Parameter value

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

SiliconSoftware_GetIntegerParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type Integer.

Syntax

```
void avl::SiliconSoftware_GetIntegerParameter
(
 SiliconSoftware_State& ioState,
 atl::Optional<int> inDeviceIndex,
 atl::Optional<const atl::String&> inAppletPath,
 int inPort,
 const atl::String& inName,
 int& outValue
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SiliconSoftware_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inAppletPath	Optional<const String&>		NIL	Applet to load
	inPort	int	0 - ∞		DMA port
	inName	const String&			Parameter name
	outValue	int&			Parameter value

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

SiliconSoftware_GetParametersList

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Returns list of all parameter available on current applet.

Syntax

```
void avl::SiliconSoftware_GetParametersList
(
 SiliconSoftware_State& ioState,
 atl::Optional<int> inDeviceIndex,
 atl::Optional<const atl::String&> inAppletPath,
 atl::Array<atl::String>& outNames,
 atl::Array<atl::String>& outTypes
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SiliconSoftware_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inAppletPath	Optional<const String&>		NIL	Applet to load
	outNames	Array<String>&			Parameters name
	outTypes	Array<String>&			Parameters type

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

SiliconSoftware_GetStringParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type String.

Syntax

```
void avl::SiliconSoftware_GetStringParameter
(
 SiliconSoftware_State& ioState,
 atl::Optional<int> inDeviceIndex,
 atl::Optional<const atl::String&> inAppletPath,
 int inPort,
 const atl::String& inName,
 atl::String& outValue
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SiliconSoftware_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inAppletPath	Optional<const String&>		NIL	Applet to load
	inPort	int	0 - ∞		DMA port
	inName	const String&			Parameter name
	outValue	String&			Parameter value

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

SiliconSoftware_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a frame using Silicon Software board.

Syntax

```
bool avl::SiliconSoftware_GrabImage
(
 SiliconSoftware_State& ioState,
 atl::Optional<int> inDeviceIndex,
 atl::Optional<const atl::String&> inAppletPath,
 int inPort,
 int inInputQueueSize,
 atl::Optional<const atl::File&> inConfigurationFile,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SiliconSoftware_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inAppletPath	Optional<const String&>		NIL	Applet to load
	inPort	int	0 - ∞		DMA port
	inInputQueueSize	int	4 - 200	5	Capacity of output frames queue
	inConfigurationFile	Optional<const File&>		NIL	Configuration file path
	outImage	Image&			Captured frame

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - Silicon Software board identifying internal index.

Frame grabber driver software

This filter is intended to cooperate with a microEnable using its vendor SDK. To be able to connect to a device, it is required to install Silicon Software Runtime with frame grabber dedicated drivers. We also recommended install version with applets.

Silicon Software Runtime can be downloaded from the following website: https://silicon.software/product/microenable-5-acl-marathon/.

Add DLL path to system environment variable may be required.

Recommended Runtime version for Aurora Vision Studio usage is **5.7.0**.

Load applets to device memory may be needed for some devices. Please use RT microDisplay software for this purpose.

Each frame grabber is initializing master mode. Acquisition is started in non blocking mode.

Device parameters

Setting **inInputQueueSize** parameter to 'Auto' will select four value as input queue size.

Setting **inAppletPath** parameter to 'Auto' will select currently loaded and activated applet.

Setting **inConfigurationFile** is recommended way to configure acquisition properties instead of SiliconSoftware_SetParameter function.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

To change other, more advanced camera parameters, use specific filters.

List of available parameters, including acquisition triggering, depends on applet type and will contains custom options. You can find it in device vendor documentation and current selected applet documentation.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- SiliconSoftware_GrabImage – Captures a frame using Silicon Software board.
- SiliconSoftware_GrabImage_WithTimeout – Captures a frame using Silicon Software board.
- SiliconSoftware_StartAcquisition – Initializes and starts image acquisition in Silicon Software board.

SiliconSoftware_GrabImage_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a frame using Silicon Software board.

Syntax

```
bool avl::SiliconSoftware_GrabImage_WithTimeout
(
 SiliconSoftware_State& ioState,
 atl::Optional<int> inDeviceIndex,
 atl::Optional<const atl::String&> inAppletPath,
 int inPort,
 int inTimeout,
 int inInputQueueSize,
 atl::Optional<const atl::File&> inConfigurationFile,
 atl::Conditional<avl::Image>& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SiliconSoftware_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inAppletPath	Optional<const String&>		NIL	Applet to load
	inPort	int	0 - ∞		DMA port
	inTimeout	int	1 - ∞	2	Maximum time to wait for frame in seconds
	inInputQueueSize	int	4 - 200	5	Capacity of output frames queue
	inConfigurationFile	Optional<const File&>		NIL	Configuration file path
	outImage	Conditional<Image>&			Captured frame

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - Silicon Software board identifying internal index.

Frame grabber driver software

This filter is intended to cooperate with a microEnable using its vendor SDK. To be able to connect to a device, it is required to install Silicon Software Runtime with frame grabber dedicated drivers. We also recommended install version with applets.

Silicon Software Runtime can be downloaded from the following website: https://silicon.software/product/microenable-5-acl-marathon/.

Add DLL path to system environment variable may be required.

Recommended Runtime version for Aurora Vision Studio usage is **5.7.0**.

Load applets to device memory may be needed for some devices. Please use RT microDisplay software for this purpose.

Each frame grabber is initializing master mode. Acquisition is started in non blocking mode.

Device parameters

Setting **inInputQueueSize** parameter to 'Auto' will select four value as input queue size.

Setting **inAppletPath** parameter to 'Auto' will select currently loaded and activated applet.

Setting **inConfigurationFile** is recommended way to configure acquisition properties instead of SiliconSoftware_SetParameter function.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

To change other, more advanced camera parameters, use specific filters.

List of available parameters, including acquisition triggering, depends on applet type and will contains custom options. You can find it in device vendor documentation and current selected applet documentation.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- SiliconSoftware_GrabImage – Captures a frame using Silicon Software board.
- SiliconSoftware_StartAcquisition – Initializes and starts image acquisition in Silicon Software board.

SiliconSoftware_SetDoubleParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type Double.

Syntax

```
void avl::SiliconSoftware_SetDoubleParameter
(
 SiliconSoftware_State& ioState,
 atl::Optional<int> inDeviceIndex,
 atl::Optional<const atl::String&> inAppletPath,
 int inPort,
 const atl::String& inName,
 double inValue
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SiliconSoftware_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inAppletPath	Optional<const String&>		NIL	Applet to load
	inPort	int	0 - ∞		DMA port
	inName	const String&			Parameter name
	inValue	double			Parameter value

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

SiliconSoftware_SetIntegerParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type Integer.

Syntax

```
void avl::SiliconSoftware_SetIntegerParameter
(
 SiliconSoftware_State& ioState,
 atl::Optional<int> inDeviceIndex,
 atl::Optional<const atl::String&> inAppletPath,
 int inPort,
 const atl::String& inName,
 int inValue
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SiliconSoftware_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inAppletPath	Optional<const String&>		NIL	Applet to load
	inPort	int	0 - ∞		DMA port
	inName	const String&			Parameter name
	inValue	int			Parameter value

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

SiliconSoftware_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Initializes and starts image acquisition in Silicon Software board.

Syntax

```
void avl::SiliconSoftware_StartAcquisition
(
 SiliconSoftware_State& ioState,
 atl::Optional<int> inDeviceIndex,
 atl::Optional<const atl::String&> inAppletPath,
 int inPort,
 int inInputQueueSize,
 atl::Optional<const atl::File&> inConfigurationFile
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SiliconSoftware_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inAppletPath	Optional<const String&>		NIL	Applet to load
	inPort	int	0 - ∞		DMA port
	inInputQueueSize	int	4 - 200	5	Capacity of output frames queue
	inConfigurationFile	Optional<const File&>		NIL	Configuration file path

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - Silicon Software board identifying internal index.

Frame grabber driver software

This filter is intended to cooperate with a microEnable using its vendor SDK. To be able to connect to a device, it is required to install Silicon Software Runtime with frame grabber dedicated drivers. We also recommended install version with applets.

Silicon Software Runtime can be downloaded from the following website: https://silicon.software/product/microenable-5-acl-marathon/.

Add DLL path to system environment variable may be required.

Recommended Runtime version for Aurora Vision Studio usage is **5.7.0**.

Load applets to device memory may be needed for some devices. Please use RT microDisplay software for this purpose.

Each frame grabber is initializing master mode. Acquisition is started in non blocking mode.

Device parameters

Setting **inInputQueueSize** parameter to 'Auto' will select four value as input queue size.

Setting **inAppletPath** parameter to 'Auto' will select currently loaded and activated applet.

Setting **inConfigurationFile** is recommended way to configure acquisition properties instead of SiliconSoftware_SetParameter function.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

To change other, more advanced camera parameters, use specific filters.

List of available parameters, including acquisition triggering, depends on applet type and will contains custom options. You can find it in device vendor documentation and current selected applet documentation.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- SiliconSoftware_GrabImage – Captures a frame using Silicon Software board.
- SiliconSoftware_GrabImage_WithTimeout – Captures a frame using Silicon Software board.
- SiliconSoftware_StartAcquisition – Initializes and starts image acquisition in Silicon Software board.

SiliconSoftware_StopAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Initializes and starts image acquisition in Silicon Software board.

Syntax

```
void avl::SiliconSoftware_StopAcquisition
(
 SiliconSoftware_State& ioState,
 atl::Optional<int> inDeviceIndex,
 atl::Optional<int> inPort
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SiliconSoftware_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inPort	Optional<int>	0 - ∞	NIL	DMA port

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

183. SmartRay

Table of content:

SmartRay_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a live image using SmartRay.

Syntax

```
bool avl::SmartRay_GrabImage
(
 SmartRay_State& ioState,
 const atl::String& inAddress,
 int inPort,
 int inConnectionTimeout,
 int inInputQueueSize,
 const avl::SmartRayDataTrigger& inDataTrigger,
 atl::Optional<bool> inSmartXccelerate,
 atl::Optional<const avl::SmartRayRegionOfInterest&> inRegionOfInterest,
 atl::Optional<const avl::SmartRayExposure&> inExposure,
 atl::Optional<const avl::SmartRayLaser&> inLaser,
 atl::Optional<const avl::SmartRayStartTrigger&> inStartTrigger,
 atl::Optional<const avl::SmartRayReflectionFilter&> inReflectionFilter,
 atl::Optional<const atl::Array<int>&> inLaserLineThreshold,
 atl::Optional<int> inMinLaserLineThicknessLimit,
 atl::Optional<int> inMaxLaserLineThicknessLimit,
 atl::Optional<bool> inSmartXpress,
 atl::Optional<bool> inSmartXtract,
 atl::Optional<avl::SmartRaySmartXtractAlgorithm::Type> inSmartXtractAlgorithm,
 atl::Optional<avl::SmartRaySmartXactMode::Type> inSmartXactMode,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SmartRay_State&			Object used to maintain state of the function.
	inAddress	const String&		\"192.168.178.200\"	Device identifying address
	inPort	int		40	Device identifying network port
	inConnectionTimeout	int	1 - 1000	60	Timeout in seconds
	inInputQueueSize	int	1 - 200	4	Capacity of output frames queue
	inDataTrigger	const SmartRayDataTrigger&			
	inSmartXccelerate	Optional<bool>		NIL	
	inRegionOfInterest	Optional<const SmartRayRegionOfInterest&>		NIL	
	inExposure	Optional<const SmartRayExposure&>		NIL	
	inLaser	Optional<const SmartRayLaser&>		NIL	
	inStartTrigger	Optional<const SmartRayStartTrigger&>		NIL	
	inReflectionFilter	Optional<const SmartRayReflectionFilter&>		NIL	
	inLaserLineThreshold	Optional<const Array<int>&>		NIL	
	inMinLaserLineThicknessLimit	Optional<int>		NIL	
	inMaxLaserLineThicknessLimit	Optional<int>		NIL	
	inSmartXpress	Optional<bool>		NIL	
	inSmartXtract	Optional<bool>		NIL	
	inSmartXtractAlgorithm	Optional<SmartRaySmartXtractAlgorithm::Type>		NIL	
	inSmartXactMode	Optional<SmartRaySmartXactMode::Type>		NIL	
	outImage	Image&			Captured live image

Remarks

Device identification

inAddress can be used to pick one of multiple devices connected to the computer. **inAddress** can be set to:

- **IP Address** - SmartRay Device identifying IP address (e.g. "127.0.0.1")

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install SmartRay SDK software with camera dedicated drivers.

Add DLL path to system environment variable may be required.

Recommended SmartRay SDK version for Aurora Vision Studio usage is **5.5.1.41**.

Disable firewall software and set 192.168.178.100 static IP in your network interface may be needed.

ECCO 35 and ECCO 55 series

Using SmartRay_LoadParameterSet before start acquisition or grab filter for this devices may be mandatory.

Camera parameters

Setting **inInputQueueSize** parameter to 'Auto' will select four value as input queue size.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

To change other, more advanced camera parameters, use specific filters.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- SmartRay_GrabImage – Captures a live image using SmartRay.
- SmartRay_GrabImage_WithTimeout – Captures with timeout a live image using SmartRay.
- SmartRay_GrabPILImages – Captures a PIL images using SmartRay.
- SmartRay_GrabZILImages – Captures a ZIL images using SmartRay.
- SmartRay_GrabPointCloud – Captures a point cloud using SmartRay.
- SmartRay_GrabPointCloud_WithTimeout – Captures a point cloud with timeout using SmartRay.
- SmartRay_StartAcquisition – Starts acquisition using SmartRay.

SmartRay_GrabImage_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures with timeout a live image using SmartRay.

Syntax

```
bool avl::SmartRay_GrabImage_WithTimeout
(
 SmartRay_State& ioState,
 const atl::String& inAddress,
 int inPort,
 int inConnectionTimeout,
 int inInputQueueSize,
 const atl::Optional<int> inTimeout,
 const avl::SmartRayDataTrigger& inDataTrigger,
 atl::Optional<bool> inSmartXccelerate,
 atl::Optional<const avl::SmartRayRegionOfInterest&> inRegionOfInterest,
 atl::Optional<const avl::SmartRayExposure&> inExposure,
 atl::Optional<const avl::SmartRayLaser&> inLaser,
 atl::Optional<const avl::SmartRayStartTrigger&> inStartTrigger,
 atl::Optional<const avl::SmartRayReflectionFilter&> inReflectionFilter,
 atl::Optional<const atl::Array<int>&> inLaserLineThreshold,
 atl::Optional<int> inMinLaserLineThicknessLimit,
 atl::Optional<int> inMaxLaserLineThicknessLimit,
 atl::Optional<bool> inSmartXpress,
 atl::Optional<bool> inSmartXtract,
 atl::Optional<avl::SmartRaySmartXtractAlgorithm::Type> inSmartXtractAlgorithm,
 atl::Optional<avl::SmartRaySmartXactMode::Type> inSmartXactMode,
 atl::Conditional<avl::Image>& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SmartRay_State&			Object used to maintain state of the function.
	inAddress	const String&		\"192.168.178.200\"	Device identifying address
	inPort	int		40	Device identifying network port
	inConnectionTimeout	int	1 - 1000	60	Timeout in seconds
	inInputQueueSize	int	1 - 200	4	Capacity of output frames queue
	inTimeout	const Optional<int>		100	Frame timeout in milliseconds
	inDataTrigger	const SmartRayDataTrigger&			
	inSmartXccelerate	Optional<bool>		NIL	
	inRegionOfInterest	Optional<const SmartRayRegionOfInterest&>		NIL	
	inExposure	Optional<const SmartRayExposure&>		NIL	
	inLaser	Optional<const SmartRayLaser&>		NIL	
	inStartTrigger	Optional<const SmartRayStartTrigger&>		NIL	
	inReflectionFilter	Optional<const SmartRayReflectionFilter&>		NIL	
	inLaserLineThreshold	Optional<const Array<int>&>		NIL	
	inMinLaserLineThicknessLimit	Optional<int>		NIL	
	inMaxLaserLineThicknessLimit	Optional<int>		NIL	
	inSmartXpress	Optional<bool>		NIL	
	inSmartXtract	Optional<bool>		NIL	
	inSmartXtractAlgorithm	Optional<SmartRaySmartXtractAlgorithm::Type>		NIL	
	inSmartXactMode	Optional<SmartRaySmartXactMode::Type>		NIL	
	outImage	Conditional<Image>&			Captured live image

Remarks

Device identification

inAddress can be used to pick one of multiple devices connected to the computer. **inAddress** can be set to:

- **IP Address** - SmartRay Device identifying IP address (e.g. "127.0.0.1")

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install SmartRay SDK software with camera dedicated drivers.

Add DLL path to system environment variable may be required.

Recommended SmartRay SDK version for Aurora Vision Studio usage is **5.5.1.41**.

Disable firewall software and set 192.168.178.100 static IP in your network interface may be needed.

ECCO 35 and ECCO 55 series

Using SmartRay_LoadParameterSet before start acquisition or grab filter for this devices may be mandatory.

Camera parameters

Setting **inInputQueueSize** parameter to 'Auto' will select four value as input queue size.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

To change other, more advanced camera parameters, use specific filters.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- SmartRay_GrabImage – Captures a live image using SmartRay.
- SmartRay_GrabImage_WithTimeout – Captures with timeout a live image using SmartRay.
- SmartRay_GrabPILImages – Captures a PIL images using SmartRay.
- SmartRay_GrabZILImages – Captures a ZIL images using SmartRay.
- SmartRay_GrabPointCloud – Captures a point cloud using SmartRay.
- SmartRay_GrabPointCloud_WithTimeout – Captures a point cloud with timeout using SmartRay.
- SmartRay_StartAcquisition – Starts acquisition using SmartRay.

SmartRay_GrabPILImages

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a PIL images using SmartRay.

Syntax

```
bool avl::SmartRay_GrabPILImages
(
 SmartRay_State& ioState,
 const atl::String& inAddress,
 int inPort,
 int inConnectionTimeout,
 int inInputQueueSize,
 const atl::Optional<int> inTimeout,
 int inNumberOfProfiles,
 const avl::SmartRayDataTrigger& inDataTrigger,
 atl::Optional<bool> inSmartXccelerate,
 atl::Optional<const avl::SmartRayRegionOfInterest&> inRegionOfInterest,
 atl::Optional<const avl::SmartRayExposure&> inExposure,
 atl::Optional<const avl::SmartRayLaser&> inLaser,
 atl::Optional<const avl::SmartRayStartTrigger&> inStartTrigger,
 atl::Optional<const avl::SmartRayReflectionFilter&> inReflectionFilter,
 atl::Optional<const atl::Array<int>&> inLaserLineThreshold,
 atl::Optional<int> inMinLaserLineThicknessLimit,
 atl::Optional<int> inMaxLaserLineThicknessLimit,
 atl::Optional<bool> inSmartXpress,
 atl::Optional<bool> inSmartXtract,
 atl::Optional<avl::SmartRaySmartXtractAlgorithm::Type> inSmartXtractAlgorithm,
 atl::Optional<avl::SmartRaySmartXactMode::Type> inSmartXactMode,
 atl::Conditional<avl::Image>& outProfileImage,
 atl::Conditional<avl::Image>& outIntensityImage,
 atl::Conditional<avl::Image>& outLaserLineThicknessImage
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SmartRay_State&			Object used to maintain state of the function.
	inAddress	const String&		\"192.168.178.200\"	Device identifying address
	inPort	int		40	Device identifying network port
	inConnectionTimeout	int	1 - 1000	60	Timeout in seconds
	inInputQueueSize	int	1 - 200	4	Capacity of output frames queue
	inTimeout	const Optional<int>		NIL	Grab image timeout. If NIl no timeout
	inNumberOfProfiles	int		10	Number of profiles to capture
	inDataTrigger	const SmartRayDataTrigger&			
	inSmartXccelerate	Optional<bool>		NIL	
	inRegionOfInterest	Optional<const SmartRayRegionOfInterest&>		NIL	
	inExposure	Optional<const SmartRayExposure&>		NIL	
	inLaser	Optional<const SmartRayLaser&>		NIL	
	inStartTrigger	Optional<const SmartRayStartTrigger&>		NIL	
	inReflectionFilter	Optional<const SmartRayReflectionFilter&>		NIL	
	inLaserLineThreshold	Optional<const Array<int>&>		NIL	
	inMinLaserLineThicknessLimit	Optional<int>		NIL	
	inMaxLaserLineThicknessLimit	Optional<int>		NIL	
	inSmartXpress	Optional<bool>		NIL	
	inSmartXtract	Optional<bool>		NIL	
	inSmartXtractAlgorithm	Optional<SmartRaySmartXtractAlgorithm::Type>		NIL	
	inSmartXactMode	Optional<SmartRaySmartXactMode::Type>		NIL	
	outProfileImage	Conditional<Image>&			Captured profile image
	outIntensityImage	Conditional<Image>&			Captured intensity image
	outLaserLineThicknessImage	Conditional<Image>&			

Remarks

Device identification

inAddress can be used to pick one of multiple devices connected to the computer. **inAddress** can be set to:

- **IP Address** - SmartRay Device identifying IP address (e.g. "127.0.0.1")

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install SmartRay SDK software with camera dedicated drivers.

Add DLL path to system environment variable may be required.

Recommended SmartRay SDK version for Aurora Vision Studio usage is **5.5.1.41**.

Disable firewall software and set 192.168.178.100 static IP in your network interface may be needed.

ECCO 35 and ECCO 55 series

Using SmartRay_LoadParameterSet before start acquisition or grab filter for this devices may be mandatory.

Camera parameters

Setting **inInputQueueSize** parameter to 'Auto' will select four value as input queue size.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

To change other, more advanced camera parameters, use specific filters.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- SmartRay_GrabImage – Captures a live image using SmartRay.
- SmartRay_GrabImage_WithTimeout – Captures with timeout a live image using SmartRay.
- SmartRay_GrabPILImages – Captures a PIL images using SmartRay.
- SmartRay_GrabZILImages – Captures a ZIL images using SmartRay.
- SmartRay_GrabPointCloud – Captures a point cloud using SmartRay.
- SmartRay_GrabPointCloud_WithTimeout – Captures a point cloud with timeout using SmartRay.
- SmartRay_StartAcquisition – Starts acquisition using SmartRay.

SmartRay_GrabPointCloud

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a point cloud using SmartRay.

Syntax

```
bool avl::SmartRay_GrabPointCloud
(
 SmartRay_State& ioState,
 const atl::String& inAddress,
 int inPort,
 int inConnectionTimeout,
 int inInputQueueSize,
 float inTransportResolution,
 int inNumberOfProfiles,
 atl::Optional<const atl::File&> inCalibrationFile,
 const avl::SmartRayDataTrigger& inDataTrigger,
 atl::Optional<bool> inSmartXccelerate,
 atl::Optional<const avl::SmartRayRegionOfInterest&> inRegionOfInterest,
 atl::Optional<const avl::SmartRayExposure&> inExposure,
 atl::Optional<const avl::SmartRayLaser&> inLaser,
 atl::Optional<const avl::SmartRayStartTrigger&> inStartTrigger,
 atl::Optional<const avl::SmartRayReflectionFilter&> inReflectionFilter,
 atl::Optional<const atl::Array<int>&> inLaserLineThreshold,
 atl::Optional<int> inMinLaserLineThicknessLimit,
 atl::Optional<int> inMaxLaserLineThicknessLimit,
 atl::Optional<bool> inSmartXpress,
 atl::Optional<bool> inSmartXtract,
 atl::Optional<avl::SmartRaySmartXtractAlgorithm::Type> inSmartXtractAlgorithm,
 atl::Optional<avl::SmartRaySmartXactMode::Type> inSmartXactMode,
 avl::Point3DGrid& outPointCloud,
 avl::Image& outIntensity,
 avl::Image& outLaserLineThickness
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SmartRay_State&			Object used to maintain state of the function.
	inAddress	const String&		\"192.168.178.200\"	Device identifying address
	inPort	int		40	Device identifying network port
	inConnectionTimeout	int	1 - 1000	60	Timeout in seconds
	inInputQueueSize	int	1 - 200	4	Capacity of output frames queue
	inTransportResolution	float	0.0 - ∞	1.0f	Defines x-axis step for every profile
	inNumberOfProfiles	int		10	Number of profiles to capture
	inCalibrationFile	Optional<const File&>		NIL	Do not set file to load data from device internal memory
	inDataTrigger	const SmartRayDataTrigger&			
	inSmartXccelerate	Optional<bool>		NIL	
	inRegionOfInterest	Optional<const SmartRayRegionOfInterest&>		NIL	
	inExposure	Optional<const SmartRayExposure&>		NIL	
	inLaser	Optional<const SmartRayLaser&>		NIL	
	inStartTrigger	Optional<const SmartRayStartTrigger&>		NIL	
	inReflectionFilter	Optional<const SmartRayReflectionFilter&>		NIL	
	inLaserLineThreshold	Optional<const Array<int>&>		NIL	
	inMinLaserLineThicknessLimit	Optional<int>		NIL	
	inMaxLaserLineThicknessLimit	Optional<int>		NIL	
	inSmartXpress	Optional<bool>		NIL	
	inSmartXtract	Optional<bool>		NIL	
	inSmartXtractAlgorithm	Optional<SmartRaySmartXtractAlgorithm::Type>		NIL	
	inSmartXactMode	Optional<SmartRaySmartXactMode::Type>		NIL	
	outPointCloud	Point3DGrid&			
	outIntensity	Image&			
	outLaserLineThickness	Image&			

Remarks

Device identification

inAddress can be used to pick one of multiple devices connected to the computer. **inAddress** can be set to:

- **IP Address** - SmartRay Device identifying IP address (e.g. "127.0.0.1")

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install SmartRay SDK software with camera dedicated drivers.

Add DLL path to system environment variable may be required.

Recommended SmartRay SDK version for Aurora Vision Studio usage is **5.5.1.41**.

Disable firewall software and set 192.168.178.100 static IP in your network interface may be needed.

ECCO 35 and ECCO 55 series

Using SmartRay_LoadParameterSet before start acquisition or grab filter for this devices may be mandatory.

Camera parameters

Setting **inInputQueueSize** parameter to 'Auto' will select four value as input queue size.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

To change other, more advanced camera parameters, use specific filters.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- SmartRay_GrabImage – Captures a live image using SmartRay.
- SmartRay_GrabImage_WithTimeout – Captures with timeout a live image using SmartRay.
- SmartRay_GrabPILImages – Captures a PIL images using SmartRay.
- SmartRay_GrabZILImages – Captures a ZIL images using SmartRay.
- SmartRay_GrabPointCloud – Captures a point cloud using SmartRay.
- SmartRay_GrabPointCloud_WithTimeout – Captures a point cloud with timeout using SmartRay.
- SmartRay_StartAcquisition – Starts acquisition using SmartRay.

SmartRay_GrabPointCloud_WithTimeout

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a point cloud with timeout using SmartRay.

Syntax

```
bool avl::SmartRay_GrabPointCloud_WithTimeout
(
 SmartRay_State& ioState,
 const atl::String& inAddress,
 int inPort,
 int inConnectionTimeout,
 int inInputQueueSize,
 const atl::Optional<int> inTimeout,
 float inTransportResolution,
 int inNumberOfProfiles,
 atl::Optional<const atl::File&> inCalibrationFile,
 const avl::SmartRayDataTrigger& inDataTrigger,
 atl::Optional<bool> inSmartXccelerate,
 atl::Optional<const avl::SmartRayRegionOfInterest&> inRegionOfInterest,
 atl::Optional<const avl::SmartRayExposure&> inExposure,
 atl::Optional<const avl::SmartRayLaser&> inLaser,
 atl::Optional<const avl::SmartRayStartTrigger&> inStartTrigger,
 atl::Optional<const avl::SmartRayReflectionFilter&> inReflectionFilter,
 atl::Optional<const atl::Array<int>&> inLaserLineThreshold,
 atl::Optional<int> inMinLaserLineThicknessLimit,
 atl::Optional<int> inMaxLaserLineThicknessLimit,
 atl::Optional<bool> inSmartXpress,
 atl::Optional<bool> inSmartXtract,
 atl::Optional<avl::SmartRaySmartXtractAlgorithm::Type> inSmartXtractAlgorithm,
 atl::Optional<avl::SmartRaySmartXactMode::Type> inSmartXactMode,
 atl::Conditional<avl::Point3DGrid>& outPointCloud,
 atl::Conditional<avl::Image>& outIntensity,
 atl::Conditional<avl::Image>& outLaserLineThickness
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SmartRay_State&			Object used to maintain state of the function.
	inAddress	const String&		\"192.168.178.200\"	Device identifying address
	inPort	int		40	Device identifying network port
	inConnectionTimeout	int	1 - 1000	60	Timeout in seconds
	inInputQueueSize	int	1 - 200	4	Capacity of output frames queue
	inTimeout	const Optional<int>		100	Frame timeout in milliseconds
	inTransportResolution	float	0.0 - ∞	1.0f	Defines x-axis step for every profile
	inNumberOfProfiles	int		10	Number of profiles to capture
	inCalibrationFile	Optional<const File&>		NIL	Do not set file to load data from device internal memory
	inDataTrigger	const SmartRayDataTrigger&			
	inSmartXccelerate	Optional<bool>		NIL	
	inRegionOfInterest	Optional<const SmartRayRegionOfInterest&>		NIL	
	inExposure	Optional<const SmartRayExposure&>		NIL	
	inLaser	Optional<const SmartRayLaser&>		NIL	
	inStartTrigger	Optional<const SmartRayStartTrigger&>		NIL	
	inReflectionFilter	Optional<const SmartRayReflectionFilter&>		NIL	
	inLaserLineThreshold	Optional<const Array<int>&>		NIL	
	inMinLaserLineThicknessLimit	Optional<int>		NIL	
	inMaxLaserLineThicknessLimit	Optional<int>		NIL	
	inSmartXpress	Optional<bool>		NIL	
	inSmartXtract	Optional<bool>		NIL	
	inSmartXtractAlgorithm	Optional<SmartRaySmartXtractAlgorithm::Type>		NIL	
	inSmartXactMode	Optional<SmartRaySmartXactMode::Type>		NIL	
	outPointCloud	Conditional<Point3DGrid>&			
	outIntensity	Conditional<Image>&			
	outLaserLineThickness	Conditional<Image>&			

Remarks

Device identification

inAddress can be used to pick one of multiple devices connected to the computer. **inAddress** can be set to:

- **IP Address** - SmartRay Device identifying IP address (e.g. "127.0.0.1")

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install SmartRay SDK software with camera dedicated drivers.

Add DLL path to system environment variable may be required.

Recommended SmartRay SDK version for Aurora Vision Studio usage is **5.5.1.41**.

Disable firewall software and set 192.168.178.100 static IP in your network interface may be needed.

ECCO 35 and ECCO 55 series

Using SmartRay_LoadParameterSet before start acquisition or grab filter for this devices may be mandatory.

Camera parameters

Setting **inInputQueueSize** parameter to 'Auto' will select four value as input queue size.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

To change other, more advanced camera parameters, use specific filters.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- SmartRay_GrabImage – Captures a live image using SmartRay.
- SmartRay_GrabImage_WithTimeout – Captures with timeout a live image using SmartRay.
- SmartRay_GrabPILImages – Captures a PIL images using SmartRay.
- SmartRay_GrabZILImages – Captures a ZIL images using SmartRay.
- SmartRay_GrabPointCloud – Captures a point cloud using SmartRay.
- SmartRay_GrabPointCloud_WithTimeout – Captures a point cloud with timeout using SmartRay.
- SmartRay_StartAcquisition – Starts acquisition using SmartRay.

SmartRay_GrabZILImages

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a ZIL images using SmartRay.

Syntax

```
bool avl::SmartRay_GrabZILImages
(
 SmartRay_State& ioState,
 const atl::String& inAddress,
 int inPort,
 int inConnectionTimeout,
 int inInputQueueSize,
 const atl::Optional<int> inTimeout,
 int inNumberOfProfiles,
 atl::Optional<const atl::File&> inCalibrationFile,
 const avl::SmartRayDataTrigger& inDataTrigger,
 atl::Optional<bool> inSmartXccelerate,
 atl::Optional<const avl::SmartRayRegionOfInterest&> inRegionOfInterest,
 atl::Optional<const avl::SmartRayExposure&> inExposure,
 atl::Optional<const avl::SmartRayLaser&> inLaser,
 atl::Optional<const avl::SmartRayStartTrigger&> inStartTrigger,
 atl::Optional<const avl::SmartRayReflectionFilter&> inReflectionFilter,
 atl::Optional<const atl::Array<int>&> inLaserLineThreshold,
 atl::Optional<const avl::SmartRayZmapResolution&> inZmapResolution,
 atl::Optional<int> inMinLaserLineThicknessLimit,
 atl::Optional<int> inMaxLaserLineThicknessLimit,
 atl::Optional<bool> inSmartXpress,
 atl::Optional<bool> inSmartXtract,
 atl::Optional<avl::SmartRaySmartXtractAlgorithm::Type> inSmartXtractAlgorithm,
 atl::Optional<avl::SmartRaySmartXactMode::Type> inSmartXactMode,
 atl::Conditional<avl::Surface>& outZMap,
 atl::Conditional<avl::Image>& outIntensityImage,
 atl::Conditional<avl::Image>& outLaserLineThicknessImage,
 atl::Conditional<float>& outVerticalRes,
 atl::Conditional<float>& outHorizontalRes,
 atl::Conditional<float>& outOriginYMillimeters
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SmartRay_State&			Object used to maintain state of the function.
	inAddress	const String&		\"192.168.178.200\"	Device identifying address
	inPort	int		40	Device identifying network port
	inConnectionTimeout	int	1 - 1000	60	Timeout in seconds
	inInputQueueSize	int	1 - 200	4	Capacity of output frames queue
	inTimeout	const Optional<int>		NIL	Grab image timeout. If NIl no timeout
	inNumberOfProfiles	int		10	Number of profiles to capture
	inCalibrationFile	Optional<const File&>		NIL	Do not set file to load data from device internal memory
	inDataTrigger	const SmartRayDataTrigger&			
	inSmartXccelerate	Optional<bool>		NIL	
	inRegionOfInterest	Optional<const SmartRayRegionOfInterest&>		NIL	
	inExposure	Optional<const SmartRayExposure&>		NIL	
	inLaser	Optional<const SmartRayLaser&>		NIL	
	inStartTrigger	Optional<const SmartRayStartTrigger&>		NIL	
	inReflectionFilter	Optional<const SmartRayReflectionFilter&>		NIL	
	inLaserLineThreshold	Optional<const Array<int>&>		NIL	
	inZmapResolution	Optional<const SmartRayZmapResolution&>		NIL	
	inMnLaserLineThicknessLimit	Optional<int>		NIL	
	inMaxLaserLineThicknessLimit	Optional<int>		NIL	
	inSmartXpress	Optional<bool>		NIL	
	inSmartXtract	Optional<bool>		NIL	
	inSmartXtractAlgorithm	Optional<SmartRaySmartXtractAlgorithm::Type>		NIL	
	inSmartXactMode	Optional<SmartRaySmartXactMode::Type>		NIL	
	outZMap	Conditional<Surface>&			Captured z-map
	outIntensityImage	Conditional<Image>&			Captured intensity image
	outLaserLineThicknessImage	Conditional<Image>&			Captured laser line thickness image
	outVerticalRes	Conditional<float>&			
	outHorizontalRes	Conditional<float>&			
	outOriginYMillimeters	Conditional<float>&			

Remarks

Device identification

inAddress can be used to pick one of multiple devices connected to the computer. **inAddress** can be set to:

- **IP Address** - SmartRay Device identifying IP address (e.g. "127.0.0.1")

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install SmartRay SDK software with camera dedicated drivers.

Add DLL path to system environment variable may be required.

Recommended SmartRay SDK version for Aurora Vision Studio usage is **5.5.1.41**.

Disable firewall software and set 192.168.178.100 static IP in your network interface may be needed.

ECCO 35 and ECCO 55 series

Using SmartRay_LoadParameterSet before start acquisition or grab filter for this devices may be mandatory.

Camera parameters

Setting **inInputQueueSize** parameter to 'Auto' will select four value as input queue size.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

To change other, more advanced camera parameters, use specific filters.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- SmartRay_GrabImage – Captures a live image using SmartRay.
- SmartRay_GrabImage_WithTimeout – Captures with timeout a live image using SmartRay.
- SmartRay_GrabPILImages – Captures a PIL images using SmartRay.
- SmartRay_GrabZILImages – Captures a ZIL images using SmartRay.
- SmartRay_GrabPointCloud – Captures a point cloud using SmartRay.
- SmartRay_GrabPointCloud_WithTimeout – Captures a point cloud with timeout using SmartRay.
- SmartRay_StartAcquisition – Starts acquisition using SmartRay.

SmartRay_LoadParameterSet

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Loads parameter set from file to SmartRay device.

Syntax

```
void avl::SmartRay_LoadParameterSet
(
 SmartRay_State& ioState,
 const atl::String& inAddress,
 int inPort,
 int inConnectionTimeout,
 const atl::File& inFile
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SmartRay_State&			Object used to maintain state of the function.
	inAddress	const String&		\"192.168.178.200\"	Device identifying address
	inPort	int		40	Device identifying network port
	inConnectionTimeout	int	1 - 1000	60	Timeout in seconds
	inFile	const File&			

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Also in AVL Lite

SmartRay_LoadSmartXpressConfiguration

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Loads SmartXpress configuration from file to SmartRay device.

Syntax

```
void avl::SmartRay_LoadSmartXpressConfiguration
(
 SmartRay_State& ioState,
 const atl::String& inAddress,
 int inPort,
 int inConnectionTimeout,
 const atl::File& inFile
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SmartRay_State&			Object used to maintain state of the function.
	inAddress	const String&		\"192.168.178.200\"	Device identifying address
	inPort	int		40	Device identifying network port
	inConnectionTimeout	int	1 - 1000	60	Timeout in seconds
	inFile	const File&			

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Also in AVL Lite

SmartRay_LoadSmartXtractPreset

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Loads SmartXtract preset data from file to SmartRay device.

Syntax

```
void avl::SmartRay_LoadSmartXtractPreset
(
 SmartRay_State& ioState,
 const atl::String& inAddress,
 int inPort,
 int inConnectionTimeout,
 const atl::File& inFile
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SmartRay_State&			Object used to maintain state of the function.
	inAddress	const String&		\"192.168.178.200\"	Device identifying address
	inPort	int		40	Device identifying network port
	inConnectionTimeout	int	1 - 1000	60	Timeout in second
	inFile	const File&			

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Also in AVL Lite

SmartRay_MeanFilter

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

SmartRay mean filter.

Syntax

```
void avl::SmartRay_MeanFilter
(
 SmartRay_UtilityState& ioState,
 int inKernelSizeX,
 int inKernelSizeY,
 int inExcludeZeros,
 const avl::Image& inImage,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
	ioState	SmartRay_UtilityState&		Object used to maintain state of the function.
	inKernelSizeX	int	5	
	inKernelSizeY	int	5	
	inExcludeZeros	int		
	inImage	const Image&		Input image
	outImage	Image&		Output image

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Also in AVL Lite

SmartRay_MedianFilter

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

SmartRay median filter.

Syntax

```
void avl::SmartRay_MedianFilter
(
 SmartRay_UtilityState& ioState,
 int inKernelSizeX,
 int inKernelSizeY,
 int inExcludeZeros,
 const avl::Image& inImage,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
	ioState	SmartRay_UtilityState&		Object used to maintain state of the function.
	inKernelSizeX	int	5	
	inKernelSizeY	int	5	
	inExcludeZeros	int		
	inImage	const Image&		Input image
	outImage	Image&		Output image

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

SmartRay_SetDigitalOutput

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets digital output of SmartRay device.

Syntax

```
void avl::SmartRay_SetDigitalOutput
(
 SmartRay_State& ioState,
 const atl::String& inAddress,
 int inPort,
 int inConnectionTimeout,
 int inChannel,
 bool inEnable
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SmartRay_State&			Object used to maintain state of the function.
	inAddress	const String&		\"192.168.178.200\"	Device identifying address
	inPort	int		40	Device identifying network port
	inConnectionTimeout	int	1 - 1000	60	Timeout in seconds
	inChannel	int	1 - 2		
	inEnable	bool			

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

SmartRay_SmoothImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

SmartRay smooth filter dedicated to 3D vision .

Syntax

```
void avl::SmartRay_SmoothImage
(
 SmartRay_UtilityState& ioState,
 avl::SmartRaySmoothImageAlgorithm::Type inSmoothAlgorithm,
 const avl::Image& inImage,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
	ioState	SmartRay_UtilityState&		Object used to maintain state of the function.
	inSmoothAlgorithm	SmartRaySmoothImageAlgorithm::Type		
	inImage	const Image&		Input image
	outImage	Image&		Output image

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

SmartRay_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Starts acquisition using SmartRay.

Syntax

```
void avl::SmartRay_StartAcquisition
(
 SmartRay_State& ioState,
 const atl::String& inAddress,
 int inPort,
 int inConnectionTimeout,
 int inInputQueueSize,
 float inTransportResolution,
 int inNumberOfProfiles,
 atl::Optional<const atl::File&> inCalibrationFile,
 avl::SmartRayImageAcquisitionType::Type inAcquisitionType,
 const avl::SmartRayDataTrigger& inDataTrigger,
 atl::Optional<bool> inSmartXccelerate,
 atl::Optional<const avl::SmartRayRegionOfInterest&> inRegionOfInterest,
 atl::Optional<const avl::SmartRayExposure&> inExposure,
 atl::Optional<const avl::SmartRayLaser&> inLaser,
 atl::Optional<const avl::SmartRayStartTrigger&> inStartTrigger,
 atl::Optional<const avl::SmartRayReflectionFilter&> inReflectionFilter,
 atl::Optional<const atl::Array<int>&> inLaserLineThreshold,
 atl::Optional<const avl::SmartRayZmapResolution&> inZmapResolution,
 atl::Optional<int> inMinLaserLineThicknessLimit,
 atl::Optional<int> inMaxLaserLineThicknessLimit,
 atl::Optional<bool> inSmartXpress,
 atl::Optional<bool> inSmartXtract,
 atl::Optional<avl::SmartRaySmartXtractAlgorithm::Type> inSmartXtractAlgorithm,
 atl::Optional<avl::SmartRaySmartXactMode::Type> inSmartXactMode
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SmartRay_State&			Object used to maintain state of the function.
	inAddress	const String&		\"192.168.178.200\"	Device identifying address
	inPort	int		40	Device identifying network port
	inConnectionTimeout	int	1 - 1000	60	Timeout in seconds
	inInputQueueSize	int	1 - 200	4	Capacity of output frames queue
	inTransportResolution	float	0.0 - ∞	1.0f	Defines x-axis step for every profile
	inNumberOfProfiles	int		10	Number of profiles to capture
	inCalibrationFile	Optional<const File&>		NIL	Do not set file to load data from device internal memory
	inAcquisitionType	SmartRayImageAcquisitionType::Type			
	inDataTrigger	const SmartRayDataTrigger&			
	inSmartXccelerate	Optional<bool>		NIL	
	inRegionOfInterest	Optional<const SmartRayRegionOfInterest&>		NIL	
	inExposure	Optional<const SmartRayExposure&>		NIL	
	inLaser	Optional<const SmartRayLaser&>		NIL	
	inStartTrigger	Optional<const SmartRayStartTrigger&>		NIL	
	inReflectionFilter	Optional<const SmartRayReflectionFilter&>		NIL	
	inLaserLineThreshold	Optional<const Array<int>&>		NIL	
	inZmapResolution	Optional<const SmartRayZmapResolution&>		NIL	
	inMinLaserLineThicknessLimit	Optional<int>		NIL	
	inMaxLaserLineThicknessLimit	Optional<int>		NIL	
	inSmartXpress	Optional<bool>		NIL	
	inSmartXtract	Optional<bool>		NIL	
	inSmartXtractAlgorithm	Optional<SmartRaySmartXtractAlgorithm::Type>		NIL	
	inSmartXactMode	Optional<SmartRaySmartXactMode::Type>		NIL	

Remarks

Device identification

inAddress can be used to pick one of multiple devices connected to the computer. **inAddress** can be set to:

- **IP Address** - SmartRay Device identifying IP address (e.g. "127.0.0.1")

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install SmartRay SDK software with camera dedicated drivers.

Add DLL path to system environment variable may be required.

Recommended SmartRay SDK version for Aurora Vision Studio usage is **5.5.1.41**.

Disable firewall software and set 192.168.178.100 static IP in your network interface may be needed.

ECCO 35 and ECCO 55 series

Using SmartRay_LoadParameterSet before start acquisition or grab filter for this devices may be mandatory.

Camera parameters

Setting **inInputQueueSize** parameter to 'Auto' will select four value as input queue size.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

To change other, more advanced camera parameters, use specific filters.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- SmartRay_GrabImage – Captures a live image using SmartRay.
- SmartRay_GrabImage_WithTimeout – Captures with timeout a live image using SmartRay.
- SmartRay_GrabPILImages – Captures a PIL images using SmartRay.
- SmartRay_GrabZILImages – Captures a ZIL images using SmartRay.
- SmartRay_GrabPointCloud – Captures a point cloud using SmartRay.
- SmartRay_GrabPointCloud_WithTimeout – Captures a point cloud with timeout using SmartRay.
- SmartRay_StartAcquisition – Starts acquisition using SmartRay.

SmartRay_StopAcquisition

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Stops acquisition using SmartRay.

Syntax

```
void avl::SmartRay_StopAcquisition
(
 SmartRay_State& ioState,
 const atl::String& inAddress,
 int inPort,
 int inConnectionTimeout
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SmartRay_State&			Object used to maintain state of the function.
	inAddress	const String&		\"192.168.178.200\"	Device identifying address
	inPort	int		40	Device identifying network port
	inConnectionTimeout	int	1 - 1000	60	

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

184. Smart

Table of content:

Smart_ConfigureDigitalIO_SynView

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Configure SynView digital IO.

Syntax

```
void avl::Smart_ConfigureDigitalIO_SynView
(
 Smart_State& ioState,
 const avl::SmartIPAddress& inIpAddress,
 atl::Optional<const atl::String&> inDeviceID,
 int inLineNumber,
 atl::Optional<bool> inLineInverter,
 atl::Optional<float> inLineDebounceDuration,
 atl::Optional<avl::SynViewLineSource::Type> inLineSource,
 bool& outIsRemote
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Smart_State&			Object used to maintain state of the function.
	inIpAddress	const SmartIPAddress&			
	inDeviceID	Optional<const String&>		NIL	Tries to find the camera in all available IDs (UserID, VendorName, ModelName...)
	inLineNumber	int	1 - 32		I/O line for configuration
	inLineInverter	Optional<bool>		NIL	Invert the signal on the selected line
	inLineDebounceDuration	Optional<float>		NIL	Input line debounce duration in us (0 for disable debouncer)
	inLineSource	Optional<SynViewLineSource::Type>		NIL	Selects a device internal signal that should drive the output signal of the selected line.
	outIsRemote	bool&			

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

List of possible exceptions:

Error type	Description
RuntimeError	Smart_ConfigureDigitalIO_SynView is not supported in the Lite edition.

Smart_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an image from a smart camera using AvSMART interface. Allows for remote access to the runtime system.

Syntax

```
bool avl::Smart_GrabImage
(
 Smart_State& ioState,
 const avl::SmartIPAddress& inIpAddress,
 avl::RoseekImageFormat::Type inPixelFormat,
 atl::Optional<float> inFrameRate,
 atl::Optional<avl::RoseekWorkingMode::Type> inWorkingMode,
 atl::Optional<int> inSensitivityLevel,
 atl::Optional<avl::RoseekExposureMode::Type> inExposureMode,
 atl::Optional<int> inExposureTime,
 atl::Optional<float> inGain,
 avl::Image& outImage,
 bool& outIsRemote
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Smart_State&			Object used to maintain state of the function.
	inIpAddress	const SmartIPAddress&			IP address of the runtime system (e.g. a smart camera)
	inPixelFormat	RoseekImageFormat::Type			Image color format
	inFrameRate	Optional<float>		NIL	Requested camera frame rate in frames per second
	inWorkingMode	Optional<RoseekWorkingMode::Type>		NIL	Working mode of image acquisition
	inSensitivityLevel	Optional<int>	0 - 3	NIL	Sensitivity level of camera sensor
	inExposureMode	Optional<RoseekExposureMode::Type>		NIL	Exposure mode, should be set to Manual if you want to adjust inExposureTime manually
	inExposureTime	Optional<int>	0 - ∞	NIL	Camera frame exposition time
	inGain	Optional<float>	0.0 - 36.0	NIL	Camera exposure gain
	outImage	Image&			Captured frame
	outIsRemote	bool&			

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

List of possible exceptions:

Error type	Description
RuntimeError	Smart_GrabImage is not supported in the Lite edition.

Smart_GrabImage_GenICam

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an image from a smart camera. Allows for remote access to the runtime system.

Syntax

```
bool avl::Smart_GrabImage_GenICam
(
 Smart_State& ioState,
 const avl::SmartIPAddress& inIpAddress,
 atl::Optional<const avl::GenAddress&> inDeviceID,
 atl::Optional<avl::RemoteGrabberPixelFormat::Type> inPixelFormat,
 avl::Image& outImage,
 bool& outIsRemote
)
```

Parameters

	Name	Type	Default	Description
	ioState	Smart_State&		Object used to maintain state of the function.
	inIpAddress	const SmartIPAddress&		IP address of the runtime system (e.g. a smart camera)
	inDeviceID	Optional<const GenAddress&>	NIL	Tries to find the camera in all available IDs (UserID, VendorName, ModelName...)
	inPixelFormat	Optional<RemoteGrabberPixelFormat::Type>	NIL	
	outImage	Image&		Captured frame
	outIsRemote	bool&		

Hints

- Check the IP address of your runtime system (e.g. using the "ipconfig" command) and set it to the **inIpAddress** input.
- Set an appropriate **inPixelFormat**.
- Run the program on you runtime system and the camera filter will acquire images directly on it.
- Run the program on your development machine (e.g. your laptop) connected to the same Ethernet and the camera filter will receive images through a connection with the runtime system.

Troubleshooting:

- Turn off the firewall on the device where the runtime system is launched.
- Check the **inDeviceID** parameter and either select the GenTL device from the manager ([...] icon on the right) or manually set the parameters if the GenTL is not available (plus icon on the left). In this case you have to fill in all the positions on the list (**inDeviceID.VendorName**, **inDeviceID.TLType**, **inDeviceID.DeviceID**).

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

List of possible exceptions:

Error type	Description
RuntimeError	Smart_GrabImage_GenICam is not supported in the Lite edition.

Smart_GrabImage_Roseek

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an image from a smart camera using Roseek interface. Allows for remote access to the runtime system.

Syntax

```
bool avl::Smart_GrabImage_Roseek
(
 Smart_State& ioState,
 const avl::SmartIPAddress& inIpAddress,
 avl::RoseekImageFormat::Type inPixelFormat,
 atl::Optional<float> inFrameRate,
 atl::Optional<avl::RoseekWorkingMode::Type> inWorkingMode,
 atl::Optional<int> inSensitivityLevel,
 atl::Optional<avl::RoseekExposureMode::Type> inExposureMode,
 atl::Optional<int> inExposureTime,
 atl::Optional<float> inGain,
 avl::Image& outImage,
 bool& outIsRemote
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Smart_State&			Object used to maintain state of the function.
	inIpAddress	const SmartIPAddress&			IP address of the runtime system (e.g. a smart camera)
	inPixelFormat	RoseekImageFormat::Type			Image color format
	inFrameRate	Optional<float>		NIL	Requested camera frame rate in frames per second
	inWorkingMode	Optional<RoseekWorkingMode::Type>		NIL	Working mode of image acquisition
	inSensitivityLevel	Optional<int>	0 - 3	NIL	Sensitivity level of camera sensor
	inExposureMode	Optional<RoseekExposureMode::Type>		NIL	Exposure mode, should be set to Manual if you want to adjust inExposureTime manually
	inExposureTime	Optional<int>	0 - ∞	NIL	Camera frame exposition time
	inGain	Optional<float>	0.0 - 36.0	NIL	Camera exposure gain
	outImage	Image&			Captured frame
	outIsRemote	bool&			

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

List of possible exceptions:

Error type	Description
RuntimeError	Smart_GrabImage_Roseek is not supported in the Lite edition.

Smart_GrabImage_SynView

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an image from a smart camera using SynView interface. Allows for remote access to the runtime system.

Syntax

```
bool avl::Smart_GrabImage_SynView
(
 Smart_State& ioState,
 const avl::SmartIPAddress& inIpAddress,
 atl::Optional<const atl::String&> inDeviceID,
 avl::SynViewColorFormat::Type inPixelFormat,
 const avl::SynViewAcquisitionParams& inAcquisitionParams,
 const avl::SynViewImageFormatParams& inImageFormatParams,
 const avl::SynViewAnalogParams& inAnalogParams,
 avl::Image& outImage,
 bool& outIsRemote
)
```

Parameters

	Name	Type	Default	Description
	ioState	Smart_State&		Object used to maintain state of the function.
	inIpAddress	const SmartIPAddress&		IP address of the runtime system (e.g. a smart camera)
	inDeviceID	Optional<const String&>	NIL	Tries to find the camera in all available IDs (UserID, VendorName, ModelName...)
	inPixelFormat	SynViewColorFormat::Type		Image color format
	inAcquisitionParams	const SynViewAcquisitionParams&		
	inImageFormatParams	const SynViewImageFormatParams&		
	inAnalogParams	const SynViewAnalogParams&		
	outImage	Image&		Captured frame
	outIsRemote	bool&		

Hints

- Check the IP address of your runtime system (e.g. using the "ipconfig" command) and set it to the **inIpAddress** input.
- Set an appropriate **inPixelFormat**.
- Run the program on you runtime system and the camera filter will acquire images directly on it.
- Run the program on your development machine (e.g. your laptop) connected to the same Ethernet and the camera filter will receive images through a connection with the runtime system.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

List of possible exceptions:

Error type	Description
RuntimeError	Smart_GrabImage_SynView is not supported in the Lite edition.

Smart_GrabImage_WebCamera

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an image from a smart camera. Allows for remote access to the runtime system.

Syntax

```
bool avl::Smart_GrabImage_WebCamera
(
 Smart_State& ioState,
 const avl::SmartIPAddress& inIpAddress,
 atl::Optional<int> inDeviceID,
 avl::Image& outImage,
 bool& outIsRemote
)
```

Parameters

	Name	Type	Default	Description
	ioState	Smart_State&		Object used to maintain state of the function.
	inIpAddress	const SmartIPAddress&		IP address of the runtime system
	inDeviceID	Optional<int>	NIL	Tries to find the camera in all available IDs (UserID, VendorName, ModelName...)
	outImage	Image&		Captured frame
	outIsRemote	bool&		Specifies whether the program is currently executed on the runtime (True) or development (False) system

Hints

- Check the IP address of your runtime system (e.g. using the "ipconfig" command) and set it to the **inIpAddress** input.
- Run the program on you runtime system and the camera filter will acquire images directly on it.
- Run the program on your development machine (e.g. your laptop) connected to the same Ethernet and the camera filter will receive images through a connection with the runtime system.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

List of possible exceptions:

Error type	Description
RuntimeError	Smart_GrabImage_WebCamera is not supported in the Lite edition.

Smart_SetDigitalOutputs

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets digital outputs using AvSMART interface.

Syntax

```
void avl::Smart_SetDigitalOutputs
(
 Smart_State& ioState,
 const avl::SmartIPAddress& inIpAddress,
 bool inVal1,
 bool inVal2,
 bool inVal3,
 bool inVal4,
 bool& outIsRemote
)
```

Parameters

	Name	Type	Default	Description
	ioState	Smart_State&		Object used to maintain state of the function.
	inIpAddress	const SmartIPAddress&		
	inVal1	bool		
	inVal2	bool		
	inVal3	bool		
	inVal4	bool		
	outIsRemote	bool&		

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

List of possible exceptions:

Error type	Description
RuntimeError	Smart_SetDigitalOutputs is not supported in the Lite edition.

Smart_SetDigitalOutputs_GenICam

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Set digital outputs.

Syntax

```
void avl::Smart_SetDigitalOutputs_GenICam
(
 Smart_State& ioState,
 const avl::SmartIPAddress& inIpAddress,
 atl::Optional<const avl::GenAddress&> inDeviceID,
 bool inVal1,
 bool inVal2,
 bool inVal3,
 bool inVal4,
 bool& outIsRemote
)
```

Parameters

	Name	Type	Default	Description
	ioState	Smart_State&		Object used to maintain state of the function.
	inIpAddress	const SmartIPAddress&		
	inDeviceID	Optional<const GenAddress&>	NIL	Tries to find the camera in all available IDs (UserID, VendorName, ModelName...)
	inVal1	bool		
	inVal2	bool		
	inVal3	bool		
	inVal4	bool		
	outIsRemote	bool&		

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

List of possible exceptions:

Error type	Description
RuntimeError	Smart_SetDigitalOutputs_GenICam is not supported in the Lite edition.

Smart_SetDigitalOutputs_Roseek

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Set digital outputs.

Syntax

```
void avl::Smart_SetDigitalOutputs_Roseek
(
 Smart_State& ioState,
 const avl::SmartIPAddress& inIpAddress,
 bool inVal1,
 bool inVal2,
 bool inVal3,
 bool inVal4,
 bool& outIsRemote
)
```

Parameters

	Name	Type	Default	Description
	ioState	Smart_State&		Object used to maintain state of the function.
	inIpAddress	const SmartIPAddress&		
	inVal1	bool		
	inVal2	bool		
	inVal3	bool		
	inVal4	bool		
	outIsRemote	bool&		

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

List of possible exceptions:

Error type	Description
RuntimeError	Smart_SetDigitalOutputs_Roseek is not supported in the Lite edition.

185. Flir Spinnaker

Table of content:

Spinnaker_BayerToRgb

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Convert image to RGB format.

Syntax

```
void avl::Spinnaker_BayerToRgb
(
 Spinnaker_State& ioState,
 const avl::Image& inImage,
 avl::SpinnakerBayerConverter::Type inBayerConverter,
 avl::BayerType::Type inBayerType,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
	ioState	Spinnaker_State&		Object used to maintain state of the function.
	inImage	const Image&		Input image
	inBayerConverter	SpinnakerBayerConverter::Type		
	inBayerType	BayerType::Type		
	outImage	Image&		Output image

Spinnaker_ExecuteCommand

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Executes command in GenICam device.

Syntax

```
void avl::Spinnaker_ExecuteCommand
(
 Spinnaker_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inCommandName,
 bool inVerify
)
```

Parameters

	Name	Type	Default	Description
	ioState	Spinnaker_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inCommandName	const String&		Name of command node to access
	inVerify	bool	True	Enables AccessMode and Range verification (default = true)

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Spinnaker_GetBoolParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type Bool from Spinnaker device.

Syntax

```
void avl::Spinnaker_GetBoolParameter
(
 Spinnaker_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 bool inVerify,
 bool inIgnoreCache,
 bool& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Spinnaker_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inParameterName	const String&		Name of parameter node to access
	inVerify	bool		Enables Range verification
	inIgnoreCache	bool		If true the value is read ignoring any caches
	outValue	bool&		Value retrieved from device parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Spinnaker_GetEnumParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type Enumeration from Spinnaker device.

Syntax

```
void avl::Spinnaker_GetEnumParameter
(
 Spinnaker_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 bool inVerify,
 bool inIgnoreCache,
 atl::String& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Spinnaker_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inParameterName	const String&		Name of parameter node to access
	inVerify	bool		Enables Range verification
	inIgnoreCache	bool		If true the value is read ignoring any caches
	outValue	String&		Value retrieved from device parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Spinnaker_GetIntegerParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type Integer from Spinnaker device.

Syntax

```
void avl::Spinnaker_GetIntegerParameter
(
 Spinnaker_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 bool inVerify,
 bool inIgnoreCache,
 int& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Spinnaker_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inParameterName	const String&		Name of parameter node to access
	inVerify	bool		Enables Range verification
	inIgnoreCache	bool		If true the value is read ignoring any caches
	outValue	int&		Value retrieved from device parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Spinnaker_GetLineStatus

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads line status.

Syntax

```
void avl::Spinnaker_GetLineStatus
(
 Spinnaker_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inLineSelector,
 bool inVerify,
 bool inIgnoreCache,
 bool& outStatus
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Spinnaker_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Device identifying address
	inLineSelector	int	0 - 3		Selects which bit of the User Output register is set by UserOutputValue.
	inVerify	bool		True	Enables Range verification
	inIgnoreCache	bool			If true the value is read ignoring any caches
	outStatus	bool&			Returns the current status of the selected input or output Line

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Spinnaker_GetRealParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type Real from Spinnaker device.

Syntax

```
void avl::Spinnaker_GetRealParameter
(
 Spinnaker_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 bool inVerify,
 bool inIgnoreCache,
 float& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Spinnaker_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inParameterName	const String&		Name of parameter node to access
	inVerify	bool		Enables Range verification
	inIgnoreCache	bool		If true the value is read ignoring any caches
	outValue	float&		Value retrieved from device parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Spinnaker_GetStringParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type String from Spinnaker device.

Syntax

```
void avl::Spinnaker_GetStringParameter
(
 Spinnaker_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 bool inVerify,
 bool inIgnoreCache,
 atl::String& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Spinnaker_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inParameterName	const String&		Name of parameter node to access
	inVerify	bool		Enables Range verification
	inIgnoreCache	bool		If true the value is read ignoring any caches
	outValue	String&		Value retrieved from device parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Spinnaker_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures images from a Flir camera using Spinnaker interface.

Syntax

```
bool avl::Spinnaker_GrabImage
(
 Spinnaker_State& ioState,
 const atl::Optional<atl::String>& inDeviceSerialNumber,
 atl::Optional<int> inBufferCount,
 atl::Optional<avl::SpinnakerPixelFormat::Type> inPixelFormat,
 avl::SpinnakerBayerConverter::Type inBayerConverter,
 atl::Optional<avl::SpinnakerAutoSettings::Type> inExposureAuto,
 atl::Optional<double> inExposureTime,
 atl::Optional<bool> inFrameRateEnable,
 atl::Optional<double> inFrameRate,
 atl::Optional<avl::SpinnakerAutoSettings::Type> inGainAuto,
 atl::Optional<double> inGain,
 avl::Image& outImage,
 atl::int64& outFrameID,
 atl::int64& outTimestamp
)
```

Parameters

	Name	Type	Range	Default	Description
↻	ioState	Spinnaker_State&			Object used to maintain state of the function.
→	inDeviceSerialNumber	const Optional<String>&		NIL	Source device serial number
→	inBufferCount	Optional<int>	1 - ∞	10	Number of image buffers
→	inPixelFormat	Optional<SpinnakerPixelFormat::Type>		NIL	Image pixel format
→	inBayerConverter	SpinnakerBayerConverter::Type			Image bayer converter. Used only with Bayer pixel format
→	inExposureAuto	Optional<SpinnakerAutoSettings::Type>		NIL	Sets the automatic exposure mode
→	inExposureTime	Optional<double>	6 - ∞	NIL	Exposure time in microseconds
→	inFrameRateEnable	Optional<bool>		NIL	If enabled, inFrameRate can be used to manually control the frame rate
→	inFrameRate	Optional<double>	1 - ∞	NIL	Frame rate in Hertz
→	inGainAuto	Optional<SpinnakerAutoSettings::Type>		NIL	Sets the automatic gain mode
→	inGain	Optional<double>	0 - ∞	NIL	Controls the amplification of the video signal in dB
←	outImage	Image&			Captured frame
←	outFrameID	int64&			Captured frame ID
←	outTimestamp	int64&			Captured frame timestamp

Description

This filter is intended for establishing connection with camera device using Spinnaker SDK, for streaming images out of it in continuous mode.

Waiting time for new, valid image frame is not limited. For explicitly limiting timeout value and handling timeout condition, use Spinnaker_GrabImage_WithTimeout filter.

Remarks

Camera driver software

This filter is intended to cooperate with camera using its vendor Spinnaker SDK. To be able to connect to camera it is required to install Spinnaker SDK software with camera dedicated drivers. Currently Aurora Vision Studio uses **Spinnaker version v2.7.0.128**.

Spinnaker SDK can be downloaded from the following website: https://www.flir.com/support/browse/camera-cores--components/machine-vision-cameras/sdks (registration may be required).

During installation you should select "Application Development". Visual Studio Version should be also changed to "Visual Studio 2015".

Camera identification

When there is only one camera connected to computer, **inDeviceSerialNumber** field can be set to Auto. In this situation first available camera will be found and connected.

inDeviceSerialNumber can be used to pick one of multiple cameras connected to computer. Set this field to device serial number (for example "1234567", should be available on device casing as "s/n: 1234567").

Camera parameters

Most of parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameters for default configuration by camera driver.

To change other and more advanced camera parameters use configuration tool "SpinView" available with Spinnaker SDK. Refer to SDK documentation to find information about parameters and how to save parameters into memory channels.

It is also possible to adjust more advanced parameters using Spinnaker_SetParameter filters. The parameters name should be obtained from SpinView application or from Spinnaker SDK.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Spinnaker_GrabImage_WithTimeout – Captures images from a Flir camera using Spinnaker interface; returns Nil if no frame comes in the specified time.
- Spinnaker_StartAcquisition – Starts image acquisition.
- Spinnaker_StopAcquisition – Stops image acquisition.

Spinnaker_GrabImage_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures images from a Flir camera using Spinnaker interface; returns Nil if no frame comes in the specified time.

Syntax

```
bool avl::Spinnaker_GrabImage_WithTimeout
(
 Spinnaker_State& ioState,
 const atl::Optional<atl::String>& inDeviceSerialNumber,
 int inTimeout,
 atl::Optional<int> inBufferCount,
 atl::Optional<avl::SpinnakerPixelFormat::Type> inPixelFormat,
 avl::SpinnakerBayerConverter::Type inBayerConverter,
 atl::Optional<avl::SpinnakerAutoSettings::Type> inExposureAuto,
 atl::Optional<double> inExposureTime,
 atl::Optional<bool> inFrameRateEnable,
 atl::Optional<double> inFrameRate,
 atl::Optional<avl::SpinnakerAutoSettings::Type> inGainAuto,
 atl::Optional<double> inGain,
 atl::Conditional<avl::Image>& outImage,
 atl::Conditional<atl::int64>& outFrameID,
 atl::Conditional<atl::int64>& outTimestamp
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Spinnaker_State&			Object used to maintain state of the function.
	inDeviceSerialNumber	const Optional<String>&		NIL	Source device serial number
	inTimeout	int	10 - ∞	100	Maximum time to wait for frame in milliseconds
	inBufferCount	Optional<int>	1 - ∞	10	Number of image buffers
	inPixelFormat	Optional<SpinnakerPixelFormat::Type>		NIL	Image pixel format
	inBayerConverter	SpinnakerBayerConverter::Type			Image bayer converter. Used only with Bayer pixel format
	inExposureAuto	Optional<SpinnakerAutoSettings::Type>		NIL	Sets the automatic exposure mode
	inExposureTime	Optional<double>	6 - ∞	NIL	Exposure time in microseconds
	inFrameRateEnable	Optional<bool>		NIL	If enabled, inFrameRate can be used to manually control the frame rate
	inFrameRate	Optional<double>	1 - ∞	NIL	Frame rate in Hertz
	inGainAuto	Optional<SpinnakerAutoSettings::Type>		NIL	Sets the automatic gain mode
	inGain	Optional<double>	0 - ∞	NIL	Controls the amplification of the video signal in dB
	outImage	Conditional<Image>&			Captured frame
	outFrameID	Conditional<int64>&			Captured frame ID
	outTimestamp	Conditional<int64>&			Captured frame timestamp

Description

This filter is intended for establishing connection with general camera device using Spinnaker SDK, for streaming images out of it in continuous mode.

This filter will wait for next valid frame not more that selected timeout value (on **inTimeout** port) in milliseconds. After timeout occurs a Nil value is returned on **outImage** port instead of image.

Similarly to **outImage**, outputs **outFrameID** and **outTimestamp** will return Nil on timeout condition.

Remarks

Camera driver software

This filter is intended to cooperate with camera using its vendor Spinnaker SDK. To be able to connect to camera it is required to install Spinnaker SDK software with camera dedicated drivers. Currently Aurora Vision Studio uses **Spinnaker version v2.7.0.128**.

Spinnaker SDK can be downloaded from the following website: https://www.flir.com/support/browse/camera-cores--components/machine-vision-cameras/sdks (registration may be required).

During installation you should select "Application Development". Visual Studio Version should be also changed to "Visual Studio 2015".

Camera identification

When there is only one camera connected to computer, **inDeviceSerialNumber** field can be set to Auto. In this situation first available camera will be found and connected.

inDeviceSerialNumber can be used to pick one of multiple cameras connected to computer. Set this field to device serial number (for example "1234567", should be available on device casing as "s/n: 1234567").

Camera parameters

Most of parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameters for default configuration by camera driver.

To change other and more advanced camera parameters use configuration tool "SpinView" available with Spinnaker SDK. Refer to SDK documentation to find information about parameters and how to save parameters into memory channels.

It is also possible to adjust more advanced parameters using Spinnaker_SetParameter filters. The parameters name should be obtained from SpinView application or from Spinnaker SDK.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Spinnaker_GrabImage – Captures images from a Flir camera using Spinnaker interface.
- Spinnaker_StartAcquisition – Starts image acquisition.
- Spinnaker_StopAcquisition – Stops image acquisition.

Spinnaker_SetBoolParameter

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type Bool.

Syntax

```
void avl::Spinnaker_SetBoolParameter
(
 Spinnaker_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 bool inValue,
 bool inVerify
)
```

Parameters

	Name	Type	Default	Description
	ioState	Spinnaker_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inParameterName	const String&		Name of feature parameter
	inValue	bool		New value to be set into device parameter
	inVerify	bool	True	Enables AccessMode and Range verification (default = true)

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Spinnaker_SetEnumParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type Enum.

Syntax

```
void avl::Spinnaker_SetEnumParameter
(
 Spinnaker_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 const atl::String& inValue,
 bool inVerify
)
```

Parameters

	Name	Type	Default	Description
	ioState	Spinnaker_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inParameterName	const String&		Name of feature parameter
	inValue	const String&		New value to be set into device parameter
	inVerify	bool	True	Enables AccessMode and Range verification (default = true)

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Spinnaker_SetIntegerParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type Integer.

Syntax

```
void avl::Spinnaker_SetIntegerParameter
(
 Spinnaker_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 int inValue,
 bool inVerify
)
```

Parameters

	Name	Type	Default	Description
	ioState	Spinnaker_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inParameterName	const String&		Name of feature parameter
	inValue	int		New value to be set into device parameter
	inVerify	bool	True	Enables AccessMode and Range verification (default = true)

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Spinnaker_SetRealParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type Real.

Syntax

```
void avl::Spinnaker_SetRealParameter
(
 Spinnaker_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 float inValue,
 bool inVerify
)
```

Parameters

	Name	Type	Default	Description
	ioState	Spinnaker_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inParameterName	const String&		Name of feature parameter
	inValue	float		New value to be set into device parameter
	inVerify	bool	True	Enables AccessMode and Range verification (default = true)

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Spinnaker_SetStringParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type String.

Syntax

```
void avl::Spinnaker_SetStringParameter
(
 Spinnaker_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 const atl::String& inValue,
 bool inVerify
)
```

Parameters

	Name	Type	Default	Description
	ioState	Spinnaker_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inParameterName	const String&		Name of feature parameter
	inValue	const String&		New value to be set into device parameter
	inVerify	bool	True	Enables AccessMode and Range verification (default = true)

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Spinnaker_SetUserOutput

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets user output value.

Syntax

```
void avl::Spinnaker_SetUserOutput
(
 Spinnaker_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inUserOutputSelector,
 bool inValue,
 bool inVerify
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Spinnaker_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Device identifying address
	inUserOutputSelector	int	0 - 3		Selects which bit of the User Output register is set by UserOutputValue.
	inValue	bool			Value of the selected user output, either logic high (enabled) or logic low (disabled).
	inVerify	bool		True	Enables AccessMode and Range verification (default = true)

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Spinnaker_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Starts image acquisition.

Syntax

```
void avl::Spinnaker_StartAcquisition
(
 Spinnaker_State& ioState,
 const atl::Optional<atl::String>& inDeviceSerialNumber,
 atl::Optional<int> inBufferCount,
 atl::Optional<avl::SpinnakerPixelFormat::Type> inPixelFormat,
 atl::Optional<avl::SpinnakerAutoSettings::Type> inExposureAuto,
 atl::Optional<double> inExposureTime,
 atl::Optional<bool> inFrameRateEnable,
 atl::Optional<double> inFrameRate,
 atl::Optional<avl::SpinnakerAutoSettings::Type> inGainAuto,
 atl::Optional<double> inGain
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Spinnaker_State&			Object used to maintain state of the function.
	inDeviceSerialNumber	const Optional<String>&		NIL	Source device serial number
	inBufferCount	Optional<int>	1 - ∞	10	Number of image buffers
	inPixelFormat	Optional<SpinnakerPixelFormat::Type>		NIL	Image pixel format
	inExposureAuto	Optional<SpinnakerAutoSettings::Type>		NIL	Sets the automatic exposure mode
	inExposureTime	Optional<double>	6 - ∞	NIL	Exposure time in microseconds
	inFrameRateEnable	Optional<bool>		NIL	If enabled, inFrameRate can be used to manually control the frame rate
	inFrameRate	Optional<double>	1 - ∞	NIL	Frame rate in Hertz
	inGainAuto	Optional<SpinnakerAutoSettings::Type>		NIL	Sets the automatic gain mode
	inGain	Optional<double>	0 - ∞	NIL	Controls the amplification of the video signal in dB

Description

This filter is intended for establishing connection with a device using Spinnaker SDK, to initialize image streaming. It is only needed when explicit image acquisition start is required in the initial phase of a program. For example, it can be used to prepare a camera, running in triggered mode, to be able to capture trigger signals before the first invoke of Spinnaker_GrabImage or to start multiple cameras in sync before the acquisition phase.

The use of this filter is not obligatory. Spinnaker_GrabImage or Spinnaker_GrabImage_WithTimeout] filters will initialize and start image acquisition upon their first invoke. When this filter is used, the **inPixelFormat** parameter of subsequent Spinnaker_GrabImage and Spinnaker_GrabImage_WithTimeout filters has no effect.

Remarks

Camera driver software

This filter is intended to cooperate with camera using its vendor Spinnaker SDK. To be able to connect to camera it is required to install Spinnaker SDK software with camera dedicated drivers. Currently Aurora Vision Studio uses **Spinnaker version v2.7.0.128**.

Spinnaker SDK can be downloaded from the following website: https://www.flir.com/support/browse/camera-cores--components/machine-vision-cameras/sdks (registration may be required).

During installation you should select "Application Development". Visual Studio Version should be also changed to "Visual Studio 2015".

Camera identification

When there is only one camera connected to computer, **inDeviceSerialNumber** field can be set to Auto. In this situation first available camera will be found and connected.

inDeviceSerialNumber can be used to pick one of multiple cameras connected to computer. Set this field to device serial number (for example "1234567", should be available on device casing as "s/n: 1234567").

Camera parameters

Most of parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameters for default configuration by camera driver.

To change other and more advanced camera parameters use configuration tool "SpinView" available with Spinnaker SDK. Refer to SDK documentation to find information about parameters and how to save parameters into memory channels.

It is also possible to adjust more advanced parameters using Spinnaker_SetParameter filters. The parameters name should be obtained from SpinView application or from Spinnaker SDK.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Spinnaker_GrabImage – Captures images from a Flir camera using Spinnaker interface.
- Spinnaker_GrabImage_WithTimeout – Captures images from a Flir camera using Spinnaker interface; returns Nil if no frame comes in the specified time.
- Spinnaker_StopAcquisition – Stops image acquisition.

Spinnaker_StopAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Stops image acquisition.

Syntax

```
void avl::Spinnaker_StopAcquisition
(
 Spinnaker_State& ioState,
 const atl::Optional<atl::String>& inDeviceSerialNumber
)
```

Parameters

	Name	Type	Default	Description
	ioState	Spinnaker_State&		Object used to maintain state of the function.
	inDeviceSerialNumber	const Optional<String>&	NIL	Source device serial number

Description

This filter is intended for stopping acquisition in a Flir camera device using Spinnaker SDK. It is only needed when implementing explicit stopping and starting of the image acquisition during a program lifetime.

This filter will usually be executed when image acquisition in a camera is already active (after filters like Spinnaker_GrabImage or Spinnaker_StartAcquisition has been executed). After its execution the image acquisition in a camera and on the application level will be stopped and all image data remaining in the input queue will be discarded. Image acquisition can be reestablished later by executing the Spinnaker_GrabImage, Spinnaker_GrabImage_WithTimeout or Spinnaker_StartAcquisition filters (pixel format and image input queue size parameters will be configured again by those filters).

This filter has no effect when invoked while the camera is not in the image acquisition state.

Remarks

Camera driver software

This filter is intended to cooperate with camera using its vendor Spinnaker SDK. To be able to connect to camera it is required to install Spinnaker SDK software with camera dedicated drivers. Currently Aurora Vision Studio uses **Spinnaker version v2.7.0.128**.

Spinnaker SDK can be downloaded from the following website: https://www.flir.com/support/browse/camera-cores--components/machine-vision-cameras/sdks (registration may be required).

During installation you should select "Application Development". Visual Studio Version should be also changed to "Visual Studio 2015".

Camera identification

When there is only one camera connected to computer, **inDeviceSerialNumber** field can be set to Auto. In this situation first available camera will be found and connected.

inDeviceSerialNumber can be used to pick one of multiple cameras connected to computer. Set this field to device serial number (for example "1234567", should be available on device casing as "s/n: 1234567").

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Spinnaker_GrabImage – Captures images from a Flir camera using Spinnaker interface.
- Spinnaker_GrabImage_WithTimeout – Captures images from a Flir camera using Spinnaker interface; returns Nil if no frame comes in the specified time.
- Spinnaker_StartAcquisition – Starts image acquisition.

186. NET SynView

Table of content:

SynView_ConfigureDigitalIO

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Configures SynView digital IO.

Syntax

```
void avl::SynView_ConfigureDigitalIO
(
 SynView_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inLineNumber,
 atl::Optional<bool> inLineInverter,
 atl::Optional<float> inLineDebounceDuration,
 atl::Optional<avl::SynViewLineSource::Type> inLineSource
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SynView_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Tries to find the camera in all available IDs (UserID, VendorName, ModelName...)
	inLineNumber	int	1 - 32		I/O line for configuration
	inLineInverter	Optional<bool>		NIL	Invert the signal on the selected line
	inLineDebounceDuration	Optional<float>		NIL	Input line debounce duration in us (0 for disable debouncer)
	inLineSource	Optional<SynViewLineSource::Type>		NIL	Selects a device internal signal that should drive the output signal of the selected line.

Description

The output can be driven by toggling the value of the corresponding user output. To configure UserOutputValue use SynView_SetUserOutput filter.

To connect user output with digital output select **userOutputX** in parameter **inLineSource**.

Advanced output configuration

The output can be driven by a timer, started by an event (e.g. activation UserOutput, Frame Trigger). To active DigitalOutput by timer you need to set **inLineSource** to **TimerXActive**. See also: SynView_ConfigureTimer, SynView Manual: Section 6.7.6 "Advanced output configuration".

Parameters **inLineInverter** and **inLineSource** can be set only for digital outputs.

Parameter **inLineDebounceDuration** can be set only for digital inputs.

The line mapping is fixed for all camera families - all PicSight and CheckSight models. It's briefly summarized in the following table:

I/O lines	Mapped signals
Line1 - Line4	Optocoupler inputs
Line5 - Line8	Reserved
Line9 - Line12	Optocoupler outputs
Line13 - Line16	Reserved
Line17 - Line18	TTL inputs
Line19 - Line24	Reserved
Line25 - Line26	TTL outputs
Line27 - Line32	Reserved

Remarks

For convenience, this filter may be invoked in the main program loop, without significant impact on the performance.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install SynView SDK software with camera dedicated drivers.

SynView SDK can be downloaded from the following website: https://net-gmbh.com/en/machine-vision/products/software

To verify the driver installation, you can run SynView Explorer (in the directory where camera drivers where installed). If the camera was detected and you can see the view from the camera, you can use SynView camera SDK in Aurora Vision Studio.

Recommended SynView SDK version for Aurora Vision Studio usage is 1.03.010.

Camera identification

When there is only one camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. DeviceID can be set to:

- **UserID** - User ID (also known as nickname) of the selected device,
- **VendorName** - Name of the manufacturer of the device,
- **ModelName** - Model name of the device,
- **DisplayName** - Device display name,
- **IPAddress** - Current IP address of the selected device (GigE Vision devices only),
- **MACAddress** - MAC address of the selected device (GigE Vision devices only),
- **SerialNumber** - String representation of camera's unique serial number.

See Also

- SynView_ConfigureDigitalIO – Configures SynView digital IO.
- SynView_ConfigureTimer – Configures one of the internal hardware timers available in the camera.
- SynView_GetDigitalIOStates – Gets SynView digital IO states.
- SynView_GrabImage – Captures a frame using SynView.

SynView_ConfigureStrobe

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Controls the built-in camera strobe.

Syntax

```
void avl::SynView_ConfigureStrobe
(
 SynView_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::SynViewStrobeEnable::Type inStrobeEnabled,
 float inStrobeDuration,
 float inStrobeDelay
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SynView_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Tries to find the camera in all available IDs (UserID, VendorName, ModelName...)
	inStrobeEnabled	SynViewStrobeEnable::Type			Selects the LED clusters of the strobe light that should be enabled
	inStrobeDuration	float	0.0 - ∞		Duration of the strobe pulse in usec.
	inStrobeDelay	float	0.0 - ∞		A delay before the strobe pulse starts after frame trigger is applied in usec

Remarks

For convenience, this filter may be invoked in the main program loop, without significant impact on the performance.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install SynView SDK software with camera dedicated drivers.

SynView SDK can be downloaded from the following website: https://net-gmbh.com/en/machine-vision/products/software

To verify the driver installation, you can run SynView Explorer (in the directory where camera drivers where installed). If the camera was detected and you can see the view from the camera, you can use SynView camera SDK in Aurora Vision Studio.

Recommended SynView SDK version for Aurora Vision Studio usage is 1.03.010.

Camera identification

When there is only one camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. DeviceID can be set to:

- **UserID** - User ID (also known as nickname) of the selected device,
- **VendorName** - Name of the manufacturer of the device,
- **ModelName** - Model name of the device,
- **DisplayName** - Device display name,
- **IPAddress** - Current IP address of the selected device (GigE Vision devices only),
- **MACAddress** - MAC address of the selected device (GigE Vision devices only),
- **SerialNumber** - String representation of camera's unique serial number.

See Also

- SynView_ConfigureDigitalIO – Configures SynView digital IO.
- SynView_ConfigureTimer – Configures one of the internal hardware timers available in the camera.
- SynView_GetDigitalIOStates – Gets SynView digital IO states.
- SynView_GrabImage – Captures a frame using SynView.

SynView_ConfigureTimer

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Configures one of the internal hardware timers available in the camera.

Syntax

```
void avl::SynView_ConfigureTimer
(
 SynView_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::SynViewTimerSelector::Type inTimerSelector,
 int inTimerDuration,
 int inTimerDelay,
 avl::SynViewTimerTriggerSource::Type inTimerTriggerSource
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SynView_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Tries to find the camera in all available IDs (UserID, VendorName, ModelName...)
	inTimerSelector	SynViewTimerSelector::Type			Timer selector
	inTimerDuration	int	0 - ∞		Sets the duration (in microseconds) of the timer active pulse.
	inTimerDelay	int	0 - ∞		Sets the delay (in microseconds) applied between activating the timer and issuing the timer active signal
	inTimerTriggerSource	SynViewTimerTriggerSource::Type			Internal device signal activating the selected timer.

Description

Advanced digital outputs configuration

The timer has a configurable *Timer Delay* and *Timer Duration* parameters (both in us).

Timers can be used to control digital outputs. In such case, the Line Source in SynView_ConfigureDigitalIO has to be set to e.g. Timer1Active (when **inTimerSelector** is Timer1).

The delay defines, how long after the timer's start the camera asserts the output.

The duration defines, how long the output stays active.

Remarks

The number of timers available depends on the camera model. Timer1 can be assumed to be present.

For convenience, this filter may be invoked in the main program loop, without significant impact on the performance.

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install SynView SDK software with camera dedicated drivers.

SynView SDK can be downloaded from the following website: https://net-gmbh.com/en/machine-vision/products/software

To verify the driver installation, you can run SynView Explorer (in the directory where camera drivers where installed). If the camera was detected and you can see the view from the camera, you can use SynView camera SDK in Aurora Vision Studio.

Recommended SynView SDK version for Aurora Vision Studio usage is 1.03.010.

Camera identification

When there is only one camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. DeviceID can be set to:

- **UserID** - User ID (also known as nickname) of the selected device,
- **VendorName** - Name of the manufacturer of the device,
- **ModelName** - Model name of the device,
- **DisplayName** - Device display name,
- **IPAddress** - Current IP address of the selected device (GigE Vision devices only),
- **MACAddress** - MAC address of the selected device (GigE Vision devices only),
- **SerialNumber** - String representation of camera's unique serial number.

See Also

- SynView_ConfigureDigitalIO – Configures SynView digital IO.
- SynView_ConfigureTimer – Configures one of the internal hardware timers available in the camera.
- SynView_GetDigitalIOStates – Gets SynView digital IO states.
- SynView_GrabImage – Captures a frame using SynView.

SynView_ExecuteCommand

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Executes a command.

Syntax

```
void avl::SynView_ExecuteCommand
(
 SynView_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::SynViewFeatureGroup::Type inFeatureGroup,
 const atl::String& inParameterName
)
```

Parameters

	Name	Type	Default	Description
	ioState	SynView_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Tries to find the camera in all available IDs (UserID, VendorName, ModelName...)
	inFeatureGroup	SynViewFeatureGroup::Type		Parameters feature group
	inParameterName	const String&		Name of feature parameter

SynView_GenerateSoftwareTrigger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Generates a software trigger, trigger source should be set to 'software'.

Syntax

```
void avl::SynView_GenerateSoftwareTrigger
(
 SynView_State& ioState,
 atl::Optional<const atl::String&> inDeviceID
)
```

Parameters

	Name	Type	Default	Description
	ioState	SynView_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Tries to find the camera in all available IDs (UserID, VendorName, ModelName...)

SynView_GetBoolParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type Bool.

Syntax

```
void avl::SynView_GetBoolParameter
(
 SynView_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::SynViewFeatureGroup::Type inFeatureGroup,
 const atl::String& inParameterName,
 bool& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	SynView_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Tries to find the camera in all available IDs (UserID, VendorName, ModelName...)
	inFeatureGroup	SynViewFeatureGroup::Type		Parameters feature group
	inParameterName	const String&		Name of feature parameter
	outValue	bool&		Value retrieved from device parameter

SynView_GetDigitalIOStates

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets SynView digital IO states.

Syntax

```
void avl::SynView_GetDigitalIOStates
(
 SynView_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inLineNumber,
 bool& outState
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SynView_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Tries to find the camera in all available IDs (UserID, VendorName, ModelName...)
	inLineNumber	int	1 - 32		I/O line for querying
	outState	bool&			Current status of the selected line

Description

This filter gets the logical state of the selected Digital IO line.

The line mapping is fixed for all camera families - all PicSight and CheckSight models. It's briefly summarized in the following table, details are listed in SynView Manual Sections: "Input and output signals" and "Connector and cable description".

I/O lines	Mapped signals
Line1 - Line4	Optocoupler inputs
Line5 - Line8	Reserved
Line9 - Line12	Optocoupler outputs
Line13 - Line16	Reserved
Line17 - Line18	TTL inputs
Line19 - Line24	Reserved
Line25 - Line26	TTL outputs
Line27 - Line32	Reserved

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install SynView SDK software with camera dedicated drivers.

SynView SDK can be downloaded from the following website: https://net-gmbh.com/en/machine-vision/products/software

To verify the driver installation, you can run SynView Explorer (in the directory where camera drivers where installed). If the camera was detected and you can see the view from the camera, you can use SynView camera SDK in Aurora Vision Studio.

Recommended SynView SDK version for Aurora Vision Studio usage is 1.03.010.

Camera identification

When there is only one camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. DeviceID can be set to:

- **UserID** - User ID (also known as nickname) of the selected device,
- **VendorName** - Name of the manufacturer of the device,
- **ModelName** - Model name of the device,
- **DisplayName** - Device display name,
- **IPAddress** - Current IP address of the selected device (GigE Vision devices only),
- **MACAddress** - MAC address of the selected device (GigE Vision devices only),
- **SerialNumber** - String representation of camera's unique serial number.

See Also

- SynView_ConfigureDigitalIO – Configures SynView digital IO.
- SynView_ConfigureTimer – Configures one of the internal hardware timers available in the camera.
- SynView_GrabImage – Captures a frame using SynView.
- SynView_SetUserOutput – Sets SynView user outputs.

SynView_GetEnumParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type Enumeration.

Syntax

```
void avl::SynView_GetEnumParameter
(
 SynView_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::SynViewFeatureGroup::Type inFeatureGroup,
 const atl::String& inParameterName,
 atl::String& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	SynView_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Tries to find the camera in all available IDs (UserID, VendorName, ModelName...)
	inFeatureGroup	SynViewFeatureGroup::Type		Parameters feature group
	inParameterName	const String&		Name of feature parameter
	outValue	String&		Value retrieved from device parameter

SynView_GetIntegerParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type Integer.

Syntax

```
void avl::SynView_GetIntegerParameter
(
 SynView_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::SynViewFeatureGroup::Type inFeatureGroup,
 const atl::String& inParameterName,
 int& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	SynView_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Tries to find the camera in all available IDs (UserID, VendorName, ModelName...)
	inFeatureGroup	SynViewFeatureGroup::Type		Parameters feature group
	inParameterName	const String&		Name of feature parameter
	outValue	int&		Value retrieved from device parameter

SynView_GetLongParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type Long.

Syntax

```
void avl::SynView_GetLongParameter
(
 SynView_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::SynViewFeatureGroup::Type inFeatureGroup,
 const atl::String& inParameterName,
 atl::int64& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	SynView_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Tries to find the camera in all available IDs (UserID, VendorName, ModelName...)
	inFeatureGroup	SynViewFeatureGroup::Type		Parameters feature group
	inParameterName	const String&		Name of feature parameter
	outValue	int64&		Value retrieved from device parameter

SynView_GetRealParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type Real.

Syntax

```
void avl::SynView_GetRealParameter
(
 SynView_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::SynViewFeatureGroup::Type inFeatureGroup,
 const atl::String& inParameterName,
 float& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	SynView_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Tries to find the camera in all available IDs (UserID, VendorName, ModelName...)
	inFeatureGroup	SynViewFeatureGroup::Type		Parameters feature group
	inParameterName	const String&		Name of feature parameter
	outValue	float&		Value retrieved from device parameter

SynView_GetStringParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type String.

Syntax

```
void avl::SynView_GetStringParameter
(
 SynView_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::SynViewFeatureGroup::Type inFeatureGroup,
 const atl::String& inParameterName,
 atl::String& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	SynView_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Tries to find the camera in all available IDs (UserID, VendorName, ModelName...)
	inFeatureGroup	SynViewFeatureGroup::Type		Parameters feature group
	inParameterName	const String&		Name of feature parameter
	outValue	String&		Value retrieved from device parameter

SynView_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a frame using SynView.

Syntax

```
bool avl::SynView_GrabImage
(
 SynView_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inInputQueueSize,
 avl::SynViewColorFormat::Type inColorFormat,
 const avl::SynViewAcquisitionParams& inAcquisitionParams,
 const avl::SynViewImageFormatParams& inImageFormatParams,
 const avl::SynViewAnalogParams& inAnalogParams,
 avl::Image& outImage,
 atl::int64& outFrameID,
 atl::int64& outTimestamp
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SynView_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Tries to find the camera in all available IDs (UserID, VendorName, ModelName...)
	inInputQueueSize	int	1 - ∞	4	Capacity of input frames queue
	inColorFormat	SynViewColorFormat::Type			Image color format
	inAcquisitionParams	const SynViewAcquisitionParams&			
	inImageFormatParams	const SynViewImageFormatParams&			
	inAnalogParams	const SynViewAnalogParams&			
	outImage	Image&			Captured frame
	outFrameID	int64&			Output frame number; 0 if not supported
	outTimestamp	int64&			Output image timestamp; 0 if not supported

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install SynView SDK software with camera dedicated drivers.

SynView SDK can be downloaded from the following website: https://net-gmbh.com/en/machine-vision/products/software

To verify the driver installation, you can run SynView Explorer (in the directory where camera drivers where installed). If the camera was detected and you can see the view from the camera, you can use SynView camera SDK in Aurora Vision Studio.

Recommended SynView SDK version for Aurora Vision Studio usage is 1.03.010.

Camera identification

When there is only one camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. DeviceID can be set to:

- **UserID** - User ID (also known as nickname) of the selected device,
- **VendorName** - Name of the manufacturer of the device,
- **ModelName** - Model name of the device,
- **DisplayName** - Device display name,
- **IPAddress** - Current IP address of the selected device (GigE Vision devices only),
- **MACAddress** - MAC address of the selected device (GigE Vision devices only),
- **SerialNumber** - String representation of camera's unique serial number.

Camera parameters

All parameters passed to camera filters are optional, setting them to 'Auto' leaves related parameters unchanged in the camera (device default or user set configuration).

To change other and more advanced camera parameters use the configuration tool "SynView Explorer" available with SynView SDK. Refer to the SDK documentation to find information about parameters and how to create default startup configuration.

See Also

- SynView_ConfigureDigitalIO – Configures SynView digital IO.
- SynView_ConfigureTimer – Configures one of the internal hardware timers available in the camera.
- SynView_GetDigitalIOStates – Gets SynView digital IO states.
- SynView_SetUserOutput – Sets SynView user outputs.

SynView_GrabImage_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a frame using SynView; returns Nil if no frame comes in the specified time.

Applications: Use this filter if the trigger may be not coming for some time, while the application should be performing other operations continuously (e.g. processing HMI events).

Syntax

```
bool avl::SynView_GrabImage_WithTimeout
(
 SynView_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inInputQueueSize,
 avl::SynViewColorFormat::Type inColorFormat,
 int inTimeout,
 const avl::SynViewAcquisitionParams& inAcquisitionParams,
 const avl::SynViewImageFormatParams& inImageFormatParams,
 const avl::SynViewAnalogParams& inAnalogParams,
 atl::Conditional<avl::Image>& outImage,
 atl::Conditional<atl::int64>& outFrameID,
 atl::Conditional<atl::int64>& outTimestamp
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SynView_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Tries to find the camera in all available IDs (UserID, VendorName, ModelName...)
	inInputQueueSize	int	1 - ∞	4	Capacity of input frames queue
	inColorFormat	SynViewColorFormat::Type			Image color format
	inTimeout	int	100 - 3600000	5000	Maximum time to wait for frame in milliseconds
	inAcquisitionParams	const SynViewAcquisitionParams&			
	inImageFormatParams	const SynViewImageFormatParams&			
	inAnalogParams	const SynViewAnalogParams&			
	outImage	Conditional<Image>&			Captured frame
	outFrameID	Conditional<int64>&			Output frame number; 0 if not supported
	outTimestamp	Conditional<int64>&			Output image timestamp; 0 if not supported

SynView_SetBoolParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type Bool.

Syntax

```
void avl::SynView_SetBoolParameter
(
 SynView_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::SynViewFeatureGroup::Type inFeatureGroup,
 const atl::String& inParameterName,
 bool inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	SynView_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Tries to find the camera in all available IDs (UserID, VendorName, ModelName...)
	inFeatureGroup	SynViewFeatureGroup::Type		Parameters feature group
	inParameterName	const String&		Name of feature parameter
	inValue	bool		New value to be set into device parameter

SynView_SetEnumParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type Enumeration.

Syntax

```
void avl::SynView_SetEnumParameter
(
 SynView_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::SynViewFeatureGroup::Type inFeatureGroup,
 const atl::String& inParameterName,
 const atl::String& inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	SynView_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Tries to find the camera in all available IDs (UserID, VendorName, ModelName...)
	inFeatureGroup	SynViewFeatureGroup::Type		Parameters feature group
	inParameterName	const String&		Name of feature parameter
	inValue	const String&		New value to be set into device parameter

SynView_SetIntegerParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type Integer.

Syntax

```
void avl::SynView_SetIntegerParameter
(
 SynView_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::SynViewFeatureGroup::Type inFeatureGroup,
 const atl::String& inParameterName,
 int inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	SynView_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Tries to find the camera in all available IDs (UserID, VendorName, ModelName...)
	inFeatureGroup	SynViewFeatureGroup::Type		Parameters feature group
	inParameterName	const String&		Name of feature parameter
	inValue	int		New value to be set into device parameter

SynView_SetRealParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type Real.

Syntax

```
void avl::SynView_SetRealParameter
(
 SynView_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::SynViewFeatureGroup::Type inFeatureGroup,
 const atl::String& inParameterName,
 float inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	SynView_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Tries to find the camera in all available IDs (UserID, VendorName, ModelName...)
	inFeatureGroup	SynViewFeatureGroup::Type		Parameters feature group
	inParameterName	const String&		Name of feature parameter
	inValue	float		New value to be set into device parameter

SynView_SetStringParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type String.

Syntax

```
void avl::SynView_SetStringParameter
(
 SynView_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::SynViewFeatureGroup::Type inFeatureGroup,
 const atl::String& inParameterName,
 const atl::String& inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	SynView_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Tries to find the camera in all available IDs (UserID, VendorName, ModelName...)
	inFeatureGroup	SynViewFeatureGroup::Type		Parameters feature group
	inParameterName	const String&		Name of feature parameter
	inValue	const String&		New value to be set into device parameter

SynView_SetUserOutput

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets SynView user outputs.

Syntax

```
void avl::SynView_SetUserOutput
(
 SynView_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 avl::SynViewUserOutputs::Type inUserOutput,
 bool inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	SynView_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Tries to find the camera in all available IDs (UserID, VendorName, ModelName...)
	inUserOutput	SynViewUserOutputs::Type		User output for configuration
	inValue	bool		Status of the selected user output

Description

Digital IO output lines can be driven by toggling values of the corresponding user outputs - select the User Output Number, then set its bool value.

To connect a user output with a digital output use the SynView_ConfigureDigitalIO filter.

User outputs can be used to enable synView timers for advanced digital outputs control. To configure it use the SynView_ConfigureTimer filter, setting Timer Trigger Source accordingly.

More info about user outputs can be found in the SynView Manual: Section 6.7.5 "User output configuration."

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install SynView SDK software with camera dedicated drivers.

SynView SDK can be downloaded from the following website: https://net-gmbh.com/en/machine-vision/products/software

To verify the driver installation, you can run SynView Explorer (in the directory where camera drivers where installed). If the camera was detected and you can see the view from the camera, you can use SynView camera SDK in Aurora Vision Studio.

Recommended SynView SDK version for Aurora Vision Studio usage is 1.03.010.

Camera identification

When there is only one camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. DeviceID can be set to:

- **UserID** - User ID (also known as nickname) of the selected device,
- **VendorName** - Name of the manufacturer of the device,
- **ModelName** - Model name of the device,
- **DisplayName** - Device display name,
- **IPAddress** - Current IP address of the selected device (GigE Vision devices only),
- **MACAddress** - MAC address of the selected device (GigE Vision devices only),
- **SerialNumber** - String representation of camera's unique serial number.

See Also

- SynView_ConfigureDigitalIO – Configures SynView digital IO.
- SynView_ConfigureTimer – Configures one of the internal hardware timers available in the camera.
- SynView_GetDigitalIOStates – Gets SynView digital IO states.
- SynView_GrabImage – Captures a frame using SynView.

SynView_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Initializes and starts image acquisition in a SynView camera.

Applications: Typically used for establishing camera connectivity before the first trigger event. Especially important for multiple-camera systems.

Syntax

```
void avl::SynView_StartAcquisition
(
 SynView_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inInputQueueSize,
 avl::SynViewColorFormat::Type inColorFormat,
 const avl::SynViewAcquisitionParams& inAcquisitionParams,
 const avl::SynViewImageFormatParams& inImageFormatParams,
 const avl::SynViewAnalogParams& inAnalogParams
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	SynView_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Tries to find the camera in all available IDs (UserID, VendorName, ModelName...)
	inInputQueueSize	int	1 - ∞	4	Capacity of input frames queue
	inColorFormat	SynViewColorFormat::Type			Image color format
	inAcquisitionParams	const SynViewAcquisitionParams&			
	inImageFormatParams	const SynViewImageFormatParams&			
	inAnalogParams	const SynViewAnalogParams&			

SynView_StopAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Stops image acquisition.

Syntax

```
void avl::SynView_StopAcquisition
(
 SynView_State& ioState,
 atl::Optional<const atl::String&> inDeviceID
)
```

Parameters

	Name	Type	Default	Description
	ioState	SynView_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Tries to find the camera in all available IDs (UserID, VendorName, ModelName...)

187. Thorlabs

Table of content:

Thorlabs_ConfigureCamera

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets various camera's parameters before starting acquisition.

Syntax

```
void avl::Thorlabs_ConfigureCamera
(
 Thorlabs_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 atl::Optional<const avl::Box&> inRoi,
 int inXBin,
 int inYBin,
 avl::ThorlabsReadoutSpeed::Type inReadoutSpeed,
 avl::ThorlabsTaps::Type inTaps,
 bool inTapBalance
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Thorlabs_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Camera serial number or name, Auto = first camera
	inRoi	Optional<const Box&>		NIL	Region of interest, Auto = full frame
	inXBin	int	1 - ∞	1	Horizontal binning factor
	inYBin	int	1 - ∞	1	Vertical binning factor
	inReadoutSpeed	ThorlabsReadoutSpeed::Type			Speed of camera's sensor readout clock
	inTaps	ThorlabsTaps::Type			Number of taps (simultaneous camera sensor readout operations per clock cycle)
	inTapBalance	bool			Whether internal tap balancing is enabled

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Thorlabs_ConfigureTrigger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets triggering parameters.

Syntax

```
void avl::Thorlabs_ConfigureTrigger
(
 Thorlabs_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 bool inTriggerEnabled,
 avl::ThorlabsTriggerPolarity::Type inTriggerPolarity,
 atl::Optional<int> inFrameCount
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	Thorlabs_State&			Object used to maintain state of the function.
→	inDeviceID	Optional<const String&>		NIL	Camera serial number or name, Auto = first camera
→	inTriggerEnabled	bool		False	True if using trigger
→	inTriggerPolarity	ThorlabsTriggerPolarity::Type			Polarity of trigger input
→	inFrameCount	Optional<int>	1 - ∞	NIL	Number of frames to capture after trigger, Auto = continuous acquisition

Remarks

Camera driver software

This filter is intended to cooperate with camera using its vendor Thorlabs SDK. To be able to connect to camera it is required to install Thorlabs SDK software with camera dedicated drivers. Currently Aurora Vision Studio uses **Thorlabs SDK version 2.9.1**.

Thorlabs SDK can be downloaded from the following website: https://www.thorlabs.com/software_pages/ViewSoftwarePage.cfm?Code=ThorCam

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Thorlabs_GrabImage – Captures an image from a Thorlabs device.
- Thorlabs_GrabImage_WithTimeout – Captures an image from a Thorlabs device.

Thorlabs_GetBoolParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type Bool.

Syntax

```
void avl::Thorlabs_GetBoolParameter
(
 Thorlabs_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 bool& outParameterValue
)
```

Parameters

	Name	Type	Default	Description
↔	ioState	Thorlabs_State&		Object used to maintain state of the function.
→	inDeviceID	Optional<const String&>	NIL	Camera serial number or name, Auto = first camera
→	inParameterName	const String&		Name of parameter
←	outParameterValue	bool&		Value of parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Thorlabs_GetEnumParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type Enum.

Syntax

```
void avl::Thorlabs_GetEnumParameter
(
 Thorlabs_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 atl::String& outParameterValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Thorlabs_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Camera serial number or name, Auto = first camera
	inParameterName	const String&		Name of parameter
	outParameterValue	String&		Value of parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Thorlabs_GetIntegerParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type Integer.

Syntax

```
void avl::Thorlabs_GetIntegerParameter
(
 Thorlabs_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 int& outParameterValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Thorlabs_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Camera serial number or name, Auto = first camera
	inParameterName	const String&		Name of parameter
	outParameterValue	int&		Value of parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Thorlabs_GetRealParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type Real.

Syntax

```
void avl::Thorlabs_GetRealParameter
(
 Thorlabs_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 float& outParameterValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Thorlabs_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Camera serial number or name, Auto = first camera
	inParameterName	const String&		Name of parameter
	outParameterValue	float&		Value of parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Thorlabs_GetStringParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type String.

Syntax

```
void avl::Thorlabs_GetStringParameter
(
 Thorlabs_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 atl::String& outParameterValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Thorlabs_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Camera serial number or name, Auto = first camera
	inParameterName	const String&		Name of parameter
	outParameterValue	String&		Value of parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Thorlabs_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an image from a Thorlabs device.

Syntax

```
bool avl::Thorlabs_GrabImage
(
 Thorlabs_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 atl::Optional<int> inInputQueueSize,
 atl::Optional<int> inExposureTime,
 atl::Optional<int> inGain,
 avl::Image& outImage,
 int& outFrameID
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Thorlabs_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Camera serial number or name, Auto = first camera
	inInputQueueSize	Optional<int>	0 - ∞	NIL	Sets size of image queue (on acquisition start, default = 8)
	inExposureTime	Optional<int>	1 - ∞	50000	Sets the target exposure time in microseconds
	inGain	Optional<int>	0 - 1023	120	Sets gain
	outImage	Image&			Captured frame
	outFrameID	int&			Captured frame ID

Remarks

Camera driver software

This filter is intended to cooperate with camera using its vendor Thorlabs SDK. To be able to connect to camera it is required to install Thorlabs SDK software with camera dedicated drivers. Currently Aurora Vision Studio uses **Thorlabs SDK version 2.9.1**.

Thorlabs SDK can be downloaded from the following website: https://www.thorlabs.com/software_pages/ViewSoftwarePage.cfm?Code=ThorCam

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Thorlabs_GrabImage_WithTimeout – Captures an image from a Thorlabs device.
- Thorlabs_StartAcquisition – Initializes and starts image acquisition in a device.
- Thorlabs_ConfigureTrigger – Sets triggering parameters.

Thorlabs_GrabImage_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an image from a Thorlabs device.

Syntax

```
bool avl::Thorlabs_GrabImage_WithTimeout
(
 Thorlabs_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inTimeout,
 atl::Optional<int> inInputQueueSize,
 atl::Optional<int> inExposureTime,
 atl::Optional<int> inGain,
 atl::Conditional<avl::Image>& outImage,
 int& outFrameID
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Thorlabs_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Camera serial number or name, Auto = first camera
	inTimeout	int			Maximum time to wait for frame in milliseconds
	inInputQueueSize	Optional<int>	0 - ∞	NIL	Sets size of image queue (on acquisition start, default = 8)
	inExposureTime	Optional<int>	1 - ∞	50000	Sets the target exposure time in microseconds
	inGain	Optional<int>	0 - 1023	120	Sets gain
	outImage	Conditional<Image>&			Captured image
	outFrameID	int&			Captured frame ID

Remarks

Camera driver software

This filter is intended to cooperate with camera using its vendor Thorlabs SDK. To be able to connect to camera it is required to install Thorlabs SDK software with camera dedicated drivers. Currently Aurora Vision Studio uses **Thorlabs SDK version 2.9.1**.

Thorlabs SDK can be downloaded from the following website: https://www.thorlabs.com/software_pages/ViewSoftwarePage.cfm?Code=ThorCam

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Thorlabs_GrabImage – Captures an image from a Thorlabs device.
- Thorlabs_StartAcquisition – Initializes and starts image acquisition in a device.
- Thorlabs_ConfigureTrigger – Sets triggering parameters.

Thorlabs_SetBoolParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type Bool.

Syntax

```
void avl::Thorlabs_SetBoolParameter
(
 Thorlabs_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 bool inParameterValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Thorlabs_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Camera serial number or name, Auto = first camera
	inParameterName	const String&		Name of parameter
	inParameterValue	bool		Value to set to parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Thorlabs_SetEnumParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type Enum.

Syntax

```
void avl::Thorlabs_SetEnumParameter
(
 Thorlabs_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 const atl::String& inParameterValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Thorlabs_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Camera serial number or name, Auto = first camera
	inParameterName	const String&		Name of parameter
	inParameterValue	const String&		Value to set to parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Thorlabs_SetIntegerParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type Integer.

Syntax

```
void avl::Thorlabs_SetIntegerParameter
(
 Thorlabs_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 int inParameterValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Thorlabs_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Camera serial number or name, Auto = first camera
	inParameterName	const String&		Name of parameter
	inParameterValue	int		Value to set to parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Thorlabs_SetRealParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type Real.

Syntax

```
void avl::Thorlabs_SetRealParameter
(
 Thorlabs_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 float inParameterValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Thorlabs_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Camera serial number or name, Auto = first camera
	inParameterName	const String&		Name of parameter
	inParameterValue	float		Value to set to parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Thorlabs_SetStringParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type String.

Syntax

```
void avl::Thorlabs_SetStringParameter
(
 Thorlabs_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 const atl::String& inParameterValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Thorlabs_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Camera serial number or name, Auto = first camera
	inParameterName	const String&		Name of parameter
	inParameterValue	const String&		Value to set to parameter

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Thorlabs_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Initializes and starts image acquisition in a device.

Syntax

```
void avl::Thorlabs_StartAcquisition
(
 Thorlabs_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 atl::Optional<int> inInputQueueSize,
 atl::Optional<int> inExposureTime,
 atl::Optional<int> inGain
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	Thorlabs_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Camera serial number or name, Auto = first camera
	inInputQueueSize	Optional<int>	0 - ∞	NIL	Sets size of image queue (on acquisition start, default = 8)
	inExposureTime	Optional<int>	1 - ∞	50000	Sets the target exposure time in microseconds
	inGain	Optional<int>	0 - 1023	120	Sets gain

Remarks

Camera driver software

This filter is intended to cooperate with camera using its vendor Thorlabs SDK. To be able to connect to camera it is required to install Thorlabs SDK software with camera dedicated drivers. Currently Aurora Vision Studio uses **Thorlabs SDK version 2.9.1**.

Thorlabs SDK can be downloaded from the following website: https://www.thorlabs.com/software_pages/ViewSoftwarePage.cfm?Code=ThorCam

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Thorlabs_GrabImage – Captures an image from a Thorlabs device.
- Thorlabs_GrabImage_WithTimeout – Captures an image from a Thorlabs device.
- Thorlabs_StopAcquisition – Stops image acquisition in a device.

Thorlabs_StopAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Stops image acquisition in a device.

Syntax

```
void avl::Thorlabs_StopAcquisition
(
 Thorlabs_State& ioState,
 atl::Optional<const atl::String&> inDeviceID
)
```

Parameters

	Name	Type	Default	Description
	ioState	Thorlabs_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Camera serial number or name, Auto = first camera

Remarks

Camera driver software

This filter is intended to cooperate with camera using its vendor Thorlabs SDK. To be able to connect to camera it is required to install Thorlabs SDK software with camera dedicated drivers. Currently Aurora Vision Studio uses **Thorlabs SDK version 2.9.1**.

Thorlabs SDK can be downloaded from the following website: https://www.thorlabs.com/software_pages/ViewSoftwarePage.cfm?Code=ThorCam

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- Thorlabs_GrabImage – Captures an image from a Thorlabs device.
- Thorlabs_GrabImage_WithTimeout – Captures an image from a Thorlabs device.
- Thorlabs_StartAcquisition – Initializes and starts image acquisition in a device.

188. Allied Vision Technologies

Table of content:

Vimba_GetBoolParameter

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets a parameter of type Bool from an Allied Vision camera.

Syntax

```
void avl::Vimba_GetBoolParameter
(
 Vimba_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 bool& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Vimba_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inParameterName	const String&		Name of feature parameter
	outValue	bool&		Value retrieved from device parameter

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Vimba SDK software.

Vimba SDK can be downloaded from the following website: https://www.alliedvision.com/en/products/software.html

To verify the driver installation, you can run Vimba Viewer. If the camera was detected and you can see the view from the camera, you can use Vimba SDK in Aurora Vision Studio.

Recommended Vimba SDK version for Aurora Vision Studio usage is 6.0.

Camera identification

When there is only one Allied Vision camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Vimba_GetDigitalInput

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets a digital input state from an Allied Vision camera.

Syntax

```
void avl::Vimba_GetDigitalInput
(
 Vimba_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inLineNumber,
 bool& outState
)
```

Parameters

	Name	Type	Default	Description
	ioState	Vimba_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inLineNumber	int		Line number to get state
	outState	bool&		State of selected line

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Vimba SDK software.

Vimba SDK can be downloaded from the following website: https://www.alliedvision.com/en/products/software.html

To verify the driver installation, you can run Vimba Viewer. If the camera was detected and you can see the view from the camera, you can use Vimba SDK in Aurora Vision Studio.

Recommended Vimba SDK version for Aurora Vision Studio usage is 6.0.

Camera identification

When there is only one Allied Vision camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Vimba_GetEnumParameter

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets a parameter of type Enumeration from an Allied Vision camera.

Syntax

```
void avl::Vimba_GetEnumParameter
(
 Vimba_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 atl::String& outValue
)
```

Parameters

	Name	Type	Default	Description
↔	ioState	Vimba_State&		Object used to maintain state of the function.
→	inDeviceID	Optional<const String&>	NIL	Device identifying address
→	inParameterName	const String&		Name of feature parameter
←	outValue	String&		Value retrieved from device parameter

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Vimba SDK software.

Vimba SDK can be downloaded from the following website: https://www.alliedvision.com/en/products/software.html

To verify the driver installation, you can run Vimba Viewer. If the camera was detected and you can see the view from the camera, you can use Vimba SDK in Aurora Vision Studio.

Recommended Vimba SDK version for Aurora Vision Studio usage is 6.0.

Camera identification

When there is only one Allied Vision camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Vimba_GetIntegerParameter

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets a parameter of type Integer from an Allied Vision camera.

Syntax

```
void avl::Vimba_GetIntegerParameter
(
 Vimba_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 int& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Vimba_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inParameterName	const String&		Name of feature parameter
	outValue	int&		Value retrieved from device parameter

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Vimba SDK software.

Vimba SDK can be downloaded from the following website: https://www.alliedvision.com/en/products/software.html

To verify the driver installation, you can run Vimba Viewer. If the camera was detected and you can see the view from the camera, you can use Vimba SDK in Aurora Vision Studio.

Recommended Vimba SDK version for Aurora Vision Studio usage is 6.0.

Camera identification

When there is only one Allied Vision camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Vimba_GetRealParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets a parameter of type Real from an Allied Vision camera.

Syntax

```
void avl::Vimba_GetRealParameter
(
 Vimba_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 float& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Vimba_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inParameterName	const String&		Name of feature parameter
	outValue	float&		Value retrieved from device parameter

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Vimba SDK software.

Vimba SDK can be downloaded from the following website: https://www.alliedvision.com/en/products/software.html

To verify the driver installation, you can run Vimba Viewer. If the camera was detected and you can see the view from the camera, you can use Vimba SDK in Aurora Vision Studio.

Recommended Vimba SDK version for Aurora Vision Studio usage is 6.0.

Camera identification

When there is only one Allied Vision camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Vimba_GetStringParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets a parameter of type String from an Allied Vision camera.

Syntax

```
void avl::Vimba_GetStringParameter
(
 Vimba_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 atl::String& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Vimba_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inParameterName	const String&		Name of feature parameter
	outValue	String&		Value retrieved from device parameter

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Vimba SDK software.

Vimba SDK can be downloaded from the following website: https://www.alliedvision.com/en/products/software.html

To verify the driver installation, you can run Vimba Viewer. If the camera was detected and you can see the view from the camera, you can use Vimba SDK in Aurora Vision Studio.

Recommended Vimba SDK version for Aurora Vision Studio usage is 6.0.

Camera identification

When there is only one Allied Vision camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Vimba_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an image from an Allied Vision camera.

Syntax

```
bool avl::Vimba_GrabImage
(
 Vimba_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const avl::VimbaImageFormatParams& inImageFormat,
 const avl::VimbaAcquisitionControlParams& inAcquisitionControl,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Default	Description
	ioState	Vimba_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inImageFormat	const VimbaImageFormatParams&		Image format parameters
	inAcquisitionControl	const VimbaAcquisitionControlParams&		Acquisition control parameters
	outImage	Image&		Captured frame

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Vimba SDK software.

Vimba SDK can be downloaded from the following website: https://www.alliedvision.com/en/products/software.html

To verify the driver installation, you can run Vimba Viewer. If the camera was detected and you can see the view from the camera, you can use Vimba SDK in Aurora Vision Studio.

Recommended Vimba SDK version for Aurora Vision Studio usage is 6.0.

Camera identification

When there is only one Allied Vision camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Vimba_GrabImage_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an image from an Allied Vision camera; returns Nil if no frame comes in the specified time.

Applications: Use this filter if the trigger may be not coming for some time, while the application should be performing other operations continuously (e.g. processing HMI events).

Syntax

```
bool avl::Vimba_GrabImage_WithTimeout
(
 Vimba_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inTimeout,
 const avl::VimbaImageFormatParams& inImageFormat,
 const avl::VimbaAcquisitionControlParams& inAcquisitionControl,
 atl::Conditional<avl::Image>& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	Vimba_State&			Object used to maintain state of the function.
→	inDeviceID	Optional<const String&>		NIL	Device identifying address
→	inTimeout	int	100 - 3600000	5000	Maximum time to wait for frame in milliseconds
→	inImageFormat	const VimbaImageFormatParams&			Image format parameters
→	inAcquisitionControl	const VimbaAcquisitionControlParams&			Acquisition control parameters
←	outImage	Conditional<Image>&			Captured frame

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Vimba SDK software.

Vimba SDK can be downloaded from the following website: https://www.alliedvision.com/en/products/software.html

To verify the driver installation, you can run Vimba Viewer. If the camera was detected and you can see the view from the camera, you can use Vimba SDK in Aurora Vision Studio.

Recommended Vimba SDK version for Aurora Vision Studio usage is 6.0.

Camera identification

When there is only one Allied Vision camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Vimba_SetBoolParameter

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a parameter of type Bool into an Allied Vision camera.

Syntax

```
void avl::Vimba_SetBoolParameter
(
 Vimba_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 bool inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Vimba_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inParameterName	const String&		Name of feature parameter
	inValue	bool		New value to be set into device parameter

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Vimba SDK software.

Vimba SDK can be downloaded from the following website: https://www.alliedvision.com/en/products/software.html

To verify the driver installation, you can run Vimba Viewer. If the camera was detected and you can see the view from the camera, you can use Vimba SDK in Aurora Vision Studio.

Recommended Vimba SDK version for Aurora Vision Studio usage is 6.0.

Camera identification

When there is only one Allied Vision camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Vimba_SetDigitalOutput

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a digital output state into an Allied Vision camera.

Syntax

```
void avl::Vimba_SetDigitalOutput
(
 Vimba_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inLineNumber,
 bool inState
)
```

Parameters

	Name	Type	Default	Description
	ioState	Vimba_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inLineNumber	int		Line number to change state
	inState	bool		State to set

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Vimba SDK software.

Vimba SDK can be downloaded from the following website: https://www.alliedvision.com/en/products/software.html

To verify the driver installation, you can run Vimba Viewer. If the camera was detected and you can see the view from the camera, you can use Vimba SDK in Aurora Vision Studio.

Recommended Vimba SDK version for Aurora Vision Studio usage is 6.0.

Camera identification

When there is only one Allied Vision camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Vimba_SetEnumParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a parameter of type Enumeration into an Allied Vision camera.

Syntax

```
void avl::Vimba_SetEnumParameter
(
 Vimba_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 const atl::String& inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Vimba_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inParameterName	const String&		Name of feature parameter
	inValue	const String&		New value to be set into device parameter

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Vimba SDK software.

Vimba SDK can be downloaded from the following website: https://www.alliedvision.com/en/products/software.html

To verify the driver installation, you can run Vimba Viewer. If the camera was detected and you can see the view from the camera, you can use Vimba SDK in Aurora Vision Studio.

Recommended Vimba SDK version for Aurora Vision Studio usage is 6.0.

Camera identification

When there is only one Allied Vision camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Vimba_SetIntegerParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a parameter of type Integer into an Allied Vision camera.

Syntax

```
void avl::Vimba_SetIntegerParameter
(
 Vimba_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 int inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Vimba_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inParameterName	const String&		Name of feature parameter
	inValue	int		New value to be set into device parameter

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Vimba SDK software.

Vimba SDK can be downloaded from the following website: https://www.alliedvision.com/en/products/software.html

To verify the driver installation, you can run Vimba Viewer. If the camera was detected and you can see the view from the camera, you can use Vimba SDK in Aurora Vision Studio.

Recommended Vimba SDK version for Aurora Vision Studio usage is 6.0.

Camera identification

When there is only one Allied Vision camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Vimba_SetRealParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a parameter of type Real into an Allied Vision camera.

Syntax

```
void avl::Vimba_SetRealParameter
(
 Vimba_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 float inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Vimba_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inParameterName	const String&		Name of feature parameter
	inValue	float		New value to be set into device parameter

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Vimba SDK software.

Vimba SDK can be downloaded from the following website: https://www.alliedvision.com/en/products/software.html

To verify the driver installation, you can run Vimba Viewer. If the camera was detected and you can see the view from the camera, you can use Vimba SDK in Aurora Vision Studio.

Recommended Vimba SDK version for Aurora Vision Studio usage is 6.0.

Camera identification

When there is only one Allied Vision camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Vimba_SetStringParameter

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a parameter of type String into an Allied Vision camera.

Syntax

```
void avl::Vimba_SetStringParameter
(
 Vimba_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inParameterName,
 const atl::String& inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	Vimba_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inParameterName	const String&		Name of feature parameter
	inValue	const String&		New value to be set into device parameter

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Vimba SDK software.

Vimba SDK can be downloaded from the following website: https://www.alliedvision.com/en/products/software.html

To verify the driver installation, you can run Vimba Viewer. If the camera was detected and you can see the view from the camera, you can use Vimba SDK in Aurora Vision Studio.

Recommended Vimba SDK version for Aurora Vision Studio usage is 6.0.

Camera identification

When there is only one Allied Vision camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

Vimba_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Starts capturing images from an Allied Vision camera.

Applications: Typically used for establishing camera connectivity before the first trigger event. Especially important for multiple-camera systems.

Syntax

```
void avl::Vimba_StartAcquisition
(
 Vimba_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const avl::VimbaImageFormatParams& inImageFormat,
 const avl::VimbaAcquisitionControlParams& inAcquisitionControl
)
```

Parameters

	Name	Type	Default	Description
	ioState	Vimba_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying address
	inImageFormat	const VimbaImageFormatParams&		Image format parameters
	inAcquisitionControl	const VimbaAcquisitionControlParams&		Acquisition control parameters

Remarks

Camera driver software

This filter is intended to cooperate with cameras using their vendor's SDK. In order to connect with the camera, it is required to install Vimba SDK software.

Vimba SDK can be downloaded from the following website: https://www.alliedvision.com/en/products/software.html

To verify the driver installation, you can run Vimba Viewer. If the camera was detected and you can see the view from the camera, you can use Vimba SDK in Aurora Vision Studio.

Recommended Vimba SDK version for Aurora Vision Studio usage is 6.0.

Camera identification

When there is only one Allied Vision camera connected, the field **inDeviceID** can be set to Auto. In this situation, the first available camera will be used.

inDeviceID can be used to pick one of multiple cameras connected to the computer. **inDeviceID** should be set to camera ID.

189. Imago Technologies

Table of content:

VisionBox_ConfigureLedGenerator

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Configures led signal generator.

Syntax

```
void avl::VisionBox_ConfigureLedGenerator
(
 VisionBox_State& ioState,
 const int inGeneratorIndex,
 const int inBlinkCount,
 const int inMs_ton,
 const int inMs_toff,
 const int inMs_tpause
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	VisionBox_State&			Object used to maintain state of the function.
	inGeneratorIndex	const int	0 - 1		Number of generator to configure
	inBlinkCount	const int	1 - 16		Count of LED blinks
	inMs_ton	const int	100 - 6400		Duration of "ON" signal
	inMs_toff	const int	100 - 6400		Duration of "OFF" signal
	inMs_tpause	const int	0 - 6400		Duration of pause between sequences

Description

This filter is simple wrapper over Led::ConfigureGenerator SDK function. Refer to AGE-X SDK to see how to use this function properly.

Remarks

Precondition for use of VisionBox filters is to have Imago Technologies AGE-X SKD installed on target computer. AGE-X SDK provides filters with necessary libraries (DLL files) and drivers. After installation of AGE-X SDK, system restart may be required.

Recommended AGE-X SDK version for Aurora Vision Studio usage is **1.6.8.0**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_ConnectMultiplexerOutput

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Connects internal multiplexer signals.

Syntax

```
void avl::VisionBox_ConnectMultiplexerOutput
(
 VisionBox_State& ioState,
 const int inOutputIndex,
 const avl::VisionBoxMuxSource::Type inTriggerSource
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	inOutputIndex	const int		
	inTriggerSource	const VisionBoxMuxSource::Type		

Description

This filter is simple wrapper over Multiplexer::ConnectOutput SDK function. Refer to AGE-X SDK to see how to use this function properly.

Remarks

Precondition for use of VisionBox filters is to have Imago Technologies AGE-X SKD installed on target computer. AGE-X SDK provides filters with necessary libraries (DLL files) and drivers. After installation of AGE-X SDK, system restart may be required.

Recommended AGE-X SDK version for Aurora Vision Studio usage is **1.6.8.0**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_GetCameraTriggerNumberOfOutputs

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets number of outputs of camera trigger device.

Syntax

```
void avl::VisionBox_GetCameraTriggerNumberOfOutputs
(
 VisionBox_State& ioState,
 const int inDeviceNumber,
 int& outNumberOfOutputs
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	inDeviceNumber	const int		
	outNumberOfOutputs	int&		

Description

This filter is simple wrapper over CameraTrigger::GetNumberOfOutputs SDK function. Refer to AGE-X SDK to see how to use this function properly.

Remarks

Precondition for use of VisionBox filters is to have Imago Technologies AGE-X SKD installed on target computer. AGE-X SDK provides filters with necessary libraries (DLL files) and drivers. After installation of AGE-X SDK, system restart may be required.

Recommended AGE-X SDK version for Aurora Vision Studio usage is **1.6.8.0**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_GetDigitalInput

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads a digital input state of Imago VisionBox computers.

Syntax

```
void avl::VisionBox_GetDigitalInput
(
 VisionBox_State& ioState,
 const int inPort,
 const int inInput,
 bool& outState
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	inPort	const int		Number of DigitalInput device to use
	inInput	const int		Number of input to read
	outState	bool&		State of selected input pin

Description

This filter can read state of digital inputs of Imago VisionBox computers. Number of inputs vary between hardware editions.

One should specify **inPort** to access, and **inInput** (singular pin) to read its **outState**. Refer to Hardware Manual to see physical connection of input pins of ones' VisionBox.

Remarks

Precondition for use of VisionBox filters is to have Imago Technologies AGE-X SKD installed on target computer. AGE-X SDK provides filters with necessary libraries (DLL files) and drivers. After installation of AGE-X SDK, system restart may be required.

Recommended AGE-X SDK version for Aurora Vision Studio usage is **1.6.8.0**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- VisionBox_SetDigitalOutput – Sets a digital output state of Imago VisionBox computers.

VisionBox_GetMultiplexerNumberOfOutputs

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets number of multiplexer outputs.

Syntax

```
void avl::VisionBox_GetMultiplexerNumberOfOutputs
(
 VisionBox_State& ioState,
 int& outNumberOfOutputs
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	outNumberOfOutputs	int&		

Description

This filter is simple wrapper over Multiplexer::GetNumberOfOutputs SDK function. Refer to AGE-X SDK to see how to use this function properly.

Remarks

Precondition for use of VisionBox filters is to have Imago Technologies AGE-X SKD installed on target computer. AGE-X SDK provides filters with necessary libraries (DLL files) and drivers. After installation of AGE-X SDK, system restart may be required.

Recommended AGE-X SDK version for Aurora Vision Studio usage is **1.6.8.0**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_GetNumberOfDigitalInputs

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets number of available digital inputs.

Syntax

```
void avl::VisionBox_GetNumberOfDigitalInputs
(
 VisionBox_State& ioState,
 const int inPort,
 int& outNumberOfInputs
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	inPort	const int		Number of DigitalInput device to use
	outNumberOfInputs	int&		Number of Digital Inputs

Description

This filter is simple wrapper over DigitalInput::GetNumberOfInputs SDK function. Refer to AGE-X SDK to see how to use this function properly.

Remarks

Precondition for use of VisionBox filters is to have Imago Technologies AGE-X SKD installed on target computer. AGE-X SDK provides filters with necessary libraries (DLL files) and drivers. After installation of AGE-X SDK, system restart may be required.

Recommended AGE-X SDK version for Aurora Vision Studio usage is **1.6.8.0**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_GetNumberOfDigitalOutputs

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets number of digital outputs.

Syntax

```
void avl::VisionBox_GetNumberOfDigitalOutputs
(
 VisionBox_State& ioState,
 const int inPort,
 int& outNumberOfOutputs
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	inPort	const int		
	outNumberOfOutputs	int&		

Description

This filter is simple wrapper over DigitalOutput::GetNumberOfOutputs SDK function. Refer to AGE-X SDK to see how to use this function properly.

Remarks

Precondition for use of VisionBox filters is to have Imago Technologies AGE-X SKD installed on target computer. AGE-X SDK provides filters with necessary libraries (DLL files) and drivers. After installation of AGE-X SDK, system restart may be required.

Recommended AGE-X SDK version for Aurora Vision Studio usage is **1.6.8.0**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_GetNumberOfLeds

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets number of available LEDs.

Syntax

```
void avl::VisionBox_GetNumberOfLeds
(
 VisionBox_State& ioState,
 int& outCount
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	outCount	int&		Number of LEDs in system

Description

This filter is simple wrapper over Led::GetNumberOfLeds SDK function. Refer to AGE-X SDK to see how to use this function properly.

Remarks

Precondition for use of VisionBox filters is to have Imago Technologies AGE-X SKD installed on target computer. AGE-X SDK provides filters with necessary libraries (DLL files) and drivers. After installation of AGE-X SDK, system restart may be required.

Recommended AGE-X SDK version for Aurora Vision Studio usage is **1.6.8.0**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_GetTriggerGeneratorConfiguration

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets trigger generator configuration.

Syntax

```
void avl::VisionBox_GetTriggerGeneratorConfiguration
(
 VisionBox_State& ioState,
 const atl::String& inConfigCommand,
 int& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	inConfigCommand	const String&		String with value to read
	outValue	int&		Requested value

Description

This filter is simple wrapper over TriggerGenerator::ConfigureGet SDK function. Refer to AGE-X SDK to see how to use this function properly.

Remarks

Precondition for use of VisionBox filters is to have Imago Technologies AGE-X SKD installed on target computer. AGE-X SDK provides filters with necessary libraries (DLL files) and drivers. After installation of AGE-X SDK, system restart may be required.

Recommended AGE-X SDK version for Aurora Vision Studio usage is **1.6.8.0**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_IOSchedulerGetBufferFillLevel

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Returns number of occupied entries in IO Scheduler buffer.

Syntax

```
void avl::VisionBox_IOSchedulerGetBufferFillLevel
(
 VisionBox_State& ioState,
 const int inSchedulerUnitNumber,
 int& outBufferFillLevel
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	VisionBox_State&			Object used to maintain state of the function.
	inSchedulerUnitNumber	const int	0 - ∞		
	outBufferFillLevel	int&			

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_IOSchedulerGetBufferMaxElementCount

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Returns maximum number of entries in IO Scheduler buffer.

Syntax

```
void avl::VisionBox_IOSchedulerGetBufferMaxElementCount
(
 VisionBox_State& ioState,
 const int inSchedulerUnitNumber,
 int& outBufferSize
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	VisionBox_State&			Object used to maintain state of the function.
	inSchedulerUnitNumber	const int	0 - ∞		
	outBufferSize	int&			

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_IOSchedulerGetCounter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets values of selected counter in IO Scheduler unit.

Syntax

```
void avl::VisionBox_IOSchedulerGetCounter
(
 VisionBox_State& ioState,
 const int inSchedulerUnitNumber,
 avl::VisionBoxIOSchedulerCounter::Type inCounterType,
 int& outCounterValue
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	VisionBox_State&			Object used to maintain state of the function.
	inSchedulerUnitNumber	const int	0 - ∞		
	inCounterType	VisionBoxIOSchedulerCounter::Type			
	outCounterValue	int&			

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_IOSchedulerGetNumberOfOutputs

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Returns number of available outputs of IO Scheduler unit of Vision Box computer.

Syntax

```
void avl::VisionBox_IOSchedulerGetNumberOfOutputs
(
 VisionBox_State& ioState,
 const int inSchedulerUnitNumber,
 int& outNumberOfOutputs
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	VisionBox_State&			Object used to maintain state of the function.
	inSchedulerUnitNumber	const int	0 - ∞		
	outNumberOfOutputs	int&			

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_IOSchedulerPushValue

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Puts value in IO Scheduler buffer.

Syntax

```
void avl::VisionBox_IOSchedulerPushValue
(
 VisionBox_State& ioState,
 const int inSchedulerUnitNumber,
 const int inCompareValue,
 const int inOutputValue
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	VisionBox_State&			Object used to maintain state of the function.
	inSchedulerUnitNumber	const int	0 - ∞		
	inCompareValue	const int			
	inOutputValue	const int			

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

List of possible exceptions:

Error type	Description
DomainError	Cannot accept negative value on inCompareValue
DomainError	Cannot accept negative value on inOutputValue

VisionBox_IOSchedulerReset

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Resets and disable IO Scheduler logic.

Syntax

```
void avl::VisionBox_IOSchedulerReset
(
)
```

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_IOSchedulerSetCompareSource

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Configures comparison element of IO Scheduler unit.

Syntax

```
void avl::VisionBox_IOSchedulerSetCompareSource
(
 VisionBox_State& ioState,
 const int inSchedulerUnitNumber,
 avl::VisionBoxIOSchedulerCmpSrc::Type inCompareSource
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	VisionBox_State&			Object used to maintain state of the function.
	inSchedulerUnitNumber	const int	0 - ∞		
	inCompareSource	VisionBoxIOSchedulerCmpSrc::Type			

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_IOSchedulerSetEncoderDirection

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Configures direction of encoder input of IO Scheduler unit.

Syntax

```
void avl::VisionBox_IOSchedulerSetEncoderDirection
(
 VisionBox_State& ioState,
 const int inSchedulerUnitNumber,
 avl::VisionBoxIOSchedulerEncoderDir::Type inEncoderDirection
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	VisionBox_State&			Object used to maintain state of the function.
	inSchedulerUnitNumber	const int	0 - ∞		
	inEncoderDirection	VisionBoxIOSchedulerEncoderDir::Type			

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_IOSchedulerSetOutputPulsTiming

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Configure pulse parameters of IO Scheduler output.

Syntax

```
void avl::VisionBox_IOSchedulerSetOutputPulsTiming
(
 VisionBox_State& ioState,
 const int inSchedulerUnitNumber,
 const int inDelayInUs,
 const int inTimeInUs
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	VisionBox_State&			Object used to maintain state of the function.
	inSchedulerUnitNumber	const int	0 - ∞		
	inDelayInUs	const int			
	inTimeInUs	const int			

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

List of possible exceptions:

Error type	Description
DomainError	Cannot accept negative time values!

VisionBox_IOSchedulerSetTriggerSource

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Connects signal to IO Scheduler input.

Syntax

```
void avl::VisionBox_IOSchedulerSetTriggerSource
(
 VisionBox_State& ioState,
 const int inSchedulerUnitNumber,
 const int inInputIndex,
 const bool inInvertInput
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	VisionBox_State&			Object used to maintain state of the function.
	inSchedulerUnitNumber	const int	0 - ∞		
	inInputIndex	const int			
	inInvertInput	const bool			

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

List of possible exceptions:

Error type	Description
DomainError	Cannot accept negative value of inInputIndex

VisionBox_IOSchedulerStart

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Enables IO Scheduler logic in FPGA unit of Vision Box.

Syntax

```
void avl::VisionBox_IOSchedulerStart
(
)
```

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_RS232ConfigureBaudRate

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Configures RS232 port. Has to be called before any other Vision Box' RS232 operation.

Syntax

```
void avl::VisionBox_RS232ConfigureBaudRate
(
 VisionBox_State& ioState,
 const int inBaudRate
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	VisionBox_State&			Object used to maintain state of the function.
	inBaudRate	const int	9600 - 921600		

Description

This filter is simple wrapper over Rs232::ConfigureBaudRate SDK function. Refer to AGE-X SDK to see how to use this function properly.

Remarks

Precondition for use of VisionBox filters is to have Imago Technologies AGE-X SKD installed on target computer. AGE-X SDK provides filters with necessary libraries (DLL files) and drivers. After installation of AGE-X SDK, system restart may be required.

Recommended AGE-X SDK version for Aurora Vision Studio usage is **1.6.8.0**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_RS232GetAvailableByteCount

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets number of bytes waiting in RS232 buffer.

Syntax

```
void avl::VisionBox_RS232GetAvailableByteCount
(
 VisionBox_State& ioState,
 int& outBytesAvailable
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	outBytesAvailable	int&		

Description

This filter is simple wrapper over Rs232::GetNumberOfBytesAvailable SDK function. Refer to AGE-X SDK to see how to use this function properly.

Remarks

Precondition for use of VisionBox filters is to have Imago Technologies AGE-X SKD installed on target computer. AGE-X SDK provides filters with necessary libraries (DLL files) and drivers. After installation of AGE-X SDK, system restart may be required.

Recommended AGE-X SDK version for Aurora Vision Studio usage is **1.6.8.0**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_RS232ReadNBytes

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads byte array from RS232 port.

Syntax

```
void avl::VisionBox_RS232ReadNBytes
(
 VisionBox_State& ioState,
 const int inBytesToRead,
 const int inTimeoutInMs,
 atl::Array<int>& outData,
 int& outBytesReceived
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	VisionBox_State&			Object used to maintain state of the function.
	inBytesToRead	const int			
	inTimeoutInMs	const int	0 - ∞		
	outData	Array<int>&			
	outBytesReceived	int&			

Description

This filter is simple wrapper over Rs232::ReadNByte SDK function. Refer to AGE-X SDK to see how to use this function properly.

Remarks

Precondition for use of VisionBox filters is to have Imago Technologies AGE-X SKD installed on target computer. AGE-X SDK provides filters with necessary libraries (DLL files) and drivers. After installation of AGE-X SDK, system restart may be required.

Recommended AGE-X SDK version for Aurora Vision Studio usage is **1.6.8.0**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_RS232WriteNBytes

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sends byte array through RS232 port.

Syntax

```
void avl::VisionBox_RS232WriteNBytes
(
 VisionBox_State& ioState,
 const atl::Array<int>& inBytesToWrite
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	inBytesToWrite	const Array<int>&		

Description

This filter is simple wrapper over Rs232::WriteNByte SDK function. Refer to AGE-X SDK to see how to use this function properly.

Remarks

Precondition for use of VisionBox filters is to have Imago Technologies AGE-X SKD installed on target computer. AGE-X SDK provides filters with necessary libraries (DLL files) and drivers. After installation of AGE-X SDK, system restart may be required.

Recommended AGE-X SDK version for Aurora Vision Studio usage is **1.6.8.0**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

List of possible exceptions:

Error type	Description
DomainError	Values of inBytesToWrite array has to be between 0 a 255

VisionBox_RS422Get

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads state of inputs as integer number.

Syntax

```
void avl::VisionBox_RS422Get
(
 VisionBox_State& ioState,
 int& outInputState
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	outInputState	int&		

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_RS422GetBit

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads state of particular input of RS 422 port.

Syntax

```
void avl::VisionBox_RS422GetBit
(
 VisionBox_State& ioState,
 const int inBitIndex,
 bool& outBitState
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	inBitIndex	const int		
	outBitState	bool&		

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

List of possible exceptions:

Error type	Description
DomainError	Cannot accept negative bit index

VisionBox_RS422GetNumberOfInputs

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Returns number of available inputs of RS 422 port.

Syntax

```
void avl::VisionBox_RS422GetNumberOfInputs
(
 VisionBox_State& ioState,
 int& outNumberOfInputs
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	outNumberOfInputs	int&		

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_RS422GetNumberOfOutputs

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Returns number of available outputs of RS 422 port.

Syntax

```
void avl::VisionBox_RS422GetNumberOfOutputs
(
 VisionBox_State& ioState,
 int& outNumberOfOutputs
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	outNumberOfOutputs	int&		

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_RS422Set

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets all outputs of RS422 port.

Syntax

```
void avl::VisionBox_RS422Set
(
 VisionBox_State& ioState,
 const int inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	inValue	const int		

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_RS422SetBit

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets state of particular output of RS 422 port.

Syntax

```
void avl::VisionBox_RS422SetBit
(
 VisionBox_State& ioState,
 const int inBitIndex,
 const bool inBitState
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	inBitIndex	const int		
	inBitState	const bool		

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

List of possible exceptions:

Error type	Description
RuntimeError	Cannot accept negative bit index.

VisionBox_RS422SetSource

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets source of RS 422 port outputs.

Syntax

```
void avl::VisionBox_RS422SetSource
(
 VisionBox_State& ioState,
 const int inBitIndex,
 const avl::VisionBoxRS422OutSource::Type inSourceType,
 const bool inInvertOutput
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	inBitIndex	const int		
	inSourceType	const VisionBoxRS422OutSource::Type		
	inInvertOutput	const bool		

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

List of possible exceptions:

Error type	Description
RuntimeError	Cannot accept negative bit index

VisionBox_SetCameraTrigger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets camera trigger configuration.

Syntax

```
void avl::VisionBox_SetCameraTrigger
(
 VisionBox_State& ioState,
 const int inDeviceNumber,
 const atl::Optional<int> inChannelNumber,
 const avl::VisionBoxTriggerSource::Type inTriggerSource,
 const bool inTrgPOff,
 const bool inTrgPOn,
 const bool inTrgNOff,
 const bool inTrgNOn
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	inDeviceNumber	const int		Device number
	inChannelNumber	const Optional<int>	NIL	Channel in device - when set to auto all channels will be configured at the same time
	inTriggerSource	const VisionBoxTriggerSource::Type		Signal used to issue the trigger
	inTrgPOff	const bool		P Gate OFF signal
	inTrgPOn	const bool		P Gate ON signal
	inTrgNOff	const bool		N Gate OFF signal
	inTrgNOn	const bool		N Gate ON signal

Description

This filter is simple wrapper over CameraTrigger::Set SDK function. Refer to AGE-X SDK to see how to use this function properly.

When **inChannelNumber** is set to auto, CameraTrigger::SetAll is used.

Remarks

Precondition for use of VisionBox filters is to have Imago Technologies AGE-X SKD installed on target computer. AGE-X SDK provides filters with necessary libraries (DLL files) and drivers. After installation of AGE-X SDK, system restart may be required.

Recommended AGE-X SDK version for Aurora Vision Studio usage is **1.6.8.0**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_SetDigitalOutput

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a digital output state of Imago VisionBox computers.

Syntax

```
void avl::VisionBox_SetDigitalOutput
(
 VisionBox_State& ioState,
 const int inPort,
 const int inOutput,
 const bool inState
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	inPort	const int		Number of DigitalOutput device to use
	inOutput	const int		Number of output to change
	inState	const bool		State of pin output to set

Description

This filter can set state of digital outputs of Imago VisionBox computers. Number of outputs vary between hardware editions.

One should specify **inPort** to access, and **inOutput** (singular pin) to set given **inState**. Refer to Hardware Manual to see physical connection of output pins of ones' VisionBox.

Remarks

Precondition for use of VisionBox filters is to have Imago Technologies AGE-X SKD installed on target computer. AGE-X SDK provides filters with necessary libraries (DLL files) and drivers. After installation of AGE-X SDK, system restart may be required.

Recommended AGE-X SDK version for Aurora Vision Studio usage is **1.6.8.0**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- VisionBox_GetDigitalInput – Reads a digital input state of Imago VisionBox computers.

VisionBox_SetDigitalOutputSource

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets source for specified digital output pin.

Syntax

```
void avl::VisionBox_SetDigitalOutputSource
(
 VisionBox_State& ioState,
 const int inPort,
 const int inPin,
 avl::VisionBoxDigitalOutSource::Type inSource,
 const bool inInvertOutput
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	inPort	const int		
	inPin	const int		
	inSource	VisionBoxDigitalOutSource::Type		
	inInvertOutput	const bool		

Description

This filter is simple wrapper over DigitalOutput::SetSource SDK function. Refer to AGE-X SDK to see how to use this function properly.

Remarks

Precondition for use of VisionBox filters is to have Imago Technologies AGE-X SKD installed on target computer. AGE-X SDK provides filters with necessary libraries (DLL files) and drivers. After installation of AGE-X SDK, system restart may be required.

Recommended AGE-X SDK version for Aurora Vision Studio usage is **1.6.8.0**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_SetLed

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets led state.

Syntax

```
void avl::VisionBox_SetLed
(
 VisionBox_State& ioState,
 const int inLedIndex,
 const bool inLedState
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	inLedIndex	const int		Selects led
	inLedState	const bool		Whether to turn led on or off

Description

This filter is simple wrapper over Led::SetLED SDK function. Refer to AGE-X SDK to see how to use this function properly.

Remarks

Precondition for use of VisionBox filters is to have Imago Technologies AGE-X SKD installed on target computer. AGE-X SDK provides filters with necessary libraries (DLL files) and drivers. After installation of AGE-X SDK, system restart may be required.

Recommended AGE-X SDK version for Aurora Vision Studio usage is **1.6.8.0**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_SetLedMode

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets led mode.

Syntax

```
void avl::VisionBox_SetLedMode
(
 VisionBox_State& ioState,
 const int inLedIndex,
 const avl::VisionBoxLedMode::Type inMode
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	inLedIndex	const int		Selects led
	inMode	const VisionBoxLedMode::Type		Selects led mode

Description

This filter is simple wrapper over Led::SetMode SDK function. Refer to AGE-X SDK to see how to use this function properly.

Remarks

Precondition for use of VisionBox filters is to have Imago Technologies AGE-X SKD installed on target computer. AGE-X SDK provides filters with necessary libraries (DLL files) and drivers. After installation of AGE-X SDK, system restart may be required.

Recommended AGE-X SDK version for Aurora Vision Studio usage is **1.6.8.0**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_SetTriggerGeneratorConfiguration

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets trigger generator configuration.

Syntax

```
void avl::VisionBox_SetTriggerGeneratorConfiguration
(
 VisionBox_State& ioState,
 const atl::String& inConfigCommand
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	inConfigCommand	const String&		String with settings command

Description

This filter is simple wrapper over TriggerGenerator::ConfigureSet SDK function. Refer to AGE-X SDK to see how to use this function properly.

Remarks

Precondition for use of VisionBox filters is to have Imago Technologies AGE-X SKD installed on target computer. AGE-X SDK provides filters with necessary libraries (DLL files) and drivers. After installation of AGE-X SDK, system restart may be required.

Recommended AGE-X SDK version for Aurora Vision Studio usage is **1.6.8.0**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_StrobeInit

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Initializes Strobe. Has to be called before any other Strobe related function.

Syntax

```
void avl::VisionBox_StrobeInit
(
 VisionBox_State& ioState,
 const int inStrobeUnit,
 const bool inReset
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	inStrobeUnit	const int		
	inReset	const bool	True	

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_StrobeSetCurrent

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets strobes current.

Syntax

```
void avl::VisionBox_StrobeSetCurrent
(
 VisionBox_State& ioState,
 const int inStrobeUnit,
 const int inCurrent,
 const int inValidate
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	inStrobeUnit	const int		Number of strobe device to use
	inCurrent	const int		Desired current in mA
	inValidate	const int		Whether to validate current

Description

This filter is simple wrapper over Strobe::SetCurrent SDK function. Refer to AGE-X SDK to see how to use this function properly.

Remarks

Precondition for use of VisionBox filters is to have Imago Technologies AGE-X SKD installed on target computer. AGE-X SDK provides filters with necessary libraries (DLL files) and drivers. After installation of AGE-X SDK, system restart may be required.

Recommended AGE-X SDK version for Aurora Vision Studio usage is **1.6.8.0**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_StrobeSetFixedCurrent

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets strobes fixed current.

Syntax

```
void avl::VisionBox_StrobeSetFixedCurrent
(
 VisionBox_State& ioState,
 const int inStrobeUnit,
 const int inSupplyVoltage,
 const int inLoadVoltage,
 const int inMaxOnTime,
 const int inMinOffTime,
 const int inCurrent,
 int& outResult
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	inStrobeUnit	const int		Number of strobe device to use
	inSupplyVoltage	const int		Supply voltage used for the strobe output
	inLoadVoltage	const int		The voltage across the load when the output current is equal to inCurrent
	inMaxOnTime	const int		Maximum allowed on-time in us
	inMinOffTime	const int		Minimum allowed off-time in us
	inCurrent	const int		Output current in mA
	outResult	int&		Minimum allowed off-time

Description

This filter is simple wrapper over Strobe::SetFixedCurrent SDK function. Refer to AGE-X SDK to see how to use this function properly.

Remarks

Precondition for use of VisionBox filters is to have Imago Technologies AGE-X SKD installed on target computer. AGE-X SDK provides filters with necessary libraries (DLL files) and drivers. After installation of AGE-X SDK, system restart may be required.

Recommended AGE-X SDK version for Aurora Vision Studio usage is **1.6.8.0**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_StrobeSetLimits

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets strobes limits.

Syntax

```
void avl::VisionBox_StrobeSetLimits
(
 VisionBox_State& ioState,
 const int inStrobeUnit,
 const int inLimitCurrent,
 const int inMaxSupplyVoltage,
 const int inMaxLoadVoltage,
 const int inMaxOnTime,
 const int inMinOffTime,
 const int inMaxCurrent,
 int& outResult
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	inStrobeUnit	const int		Number of strobe device to use
	inLimitCurrent	const int		Calculate allowed off-time or maximum current
	inMaxSupplyVoltage	const int		Supply voltage
	inMaxLoadVoltage	const int		Voltage across the load with MaxCurrent current
	inMaxOnTime	const int		Maximum allowed on-time
	inMinOffTime	const int		Minimum allowed off-time
	inMaxCurrent	const int		Maximum allowed current in mA
	outResult	int&		Result calculated regarding inLimitCurrent value

Description

This filter is simple wrapper over Strobe::SetLimits SDK function. Refer to AGE-X SDK to see how to use this function properly.

Remarks

Precondition for use of VisionBox filters is to have Imago Technologies AGE-X SKD installed on target computer. AGE-X SDK provides filters with necessary libraries (DLL files) and drivers. After installation of AGE-X SDK, system restart may be required.

Recommended AGE-X SDK version for Aurora Vision Studio usage is **1.6.8.0**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_StrobeSetOnTime

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets strobes on-time.

Syntax

```
void avl::VisionBox_StrobeSetOnTime
(
 VisionBox_State& ioState,
 const int inStrobeUnit,
 const int inOnTime
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	inStrobeUnit	const int		Number of strobe device to use
	inOnTime	const int		On-time in us

Description

This filter is simple wrapper over Strobe::SetOnTime SDK function. Refer to AGE-X SDK to see how to use this function properly.

Remarks

Precondition for use of VisionBox filters is to have Imago Technologies AGE-X SKD installed on target computer. AGE-X SDK provides filters with necessary libraries (DLL files) and drivers. After installation of AGE-X SDK, system restart may be required.

Recommended AGE-X SDK version for Aurora Vision Studio usage is **1.6.8.0**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_StrobeSetTriggerDelay

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets strobes trigger delay.

Syntax

```
void avl::VisionBox_StrobeSetTriggerDelay
(
 VisionBox_State& ioState,
 const int inStrobeUnit,
 const int inTriggerDelay
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	inStrobeUnit	const int		Number of strobe device to use
	inTriggerDelay	const int		Delay in us

Description

This filter is simple wrapper over Strobe::SetTriggerDelay SDK function. Refer to AGE-X SDK to see how to use this function properly.

Remarks

Precondition for use of VisionBox filters is to have Imago Technologies AGE-X SKD installed on target computer. AGE-X SDK provides filters with necessary libraries (DLL files) and drivers. After installation of AGE-X SDK, system restart may be required.

Recommended AGE-X SDK version for Aurora Vision Studio usage is **1.6.8.0**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_StrobeSetTriggerMode

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets strobes trigger mode.

Syntax

```
void avl::VisionBox_StrobeSetTriggerMode
(
 VisionBox_State& ioState,
 const int inStrobeUnit,
 const avl::VisionBoxStrobeMode::Type inMode,
 int& outResult
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	inStrobeUnit	const int		Number of strobe device to use
	inMode	const VisionBoxStrobeMode::Type		Selects strobe mode
	outResult	int&		Whether new mode has been selected

Description

This filter is simple wrapper over Strobe::SetTriggerMode SDK function. Refer to AGE-X SDK to see how to use this function properly.

Remarks

Precondition for use of VisionBox filters is to have Imago Technologies AGE-X SKD installed on target computer. AGE-X SDK provides filters with necessary libraries (DLL files) and drivers. After installation of AGE-X SDK, system restart may be required.

Recommended AGE-X SDK version for Aurora Vision Studio usage is **1.6.8.0**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_StrobeSetTriggerSource

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets strobes trigger source.

Syntax

```
void avl::VisionBox_StrobeSetTriggerSource
(
 VisionBox_State& ioState,
 const int inStrobeUnit,
 const avl::VisionBoxStrobeSource::Type inSource,
 const int inInvertTrigger
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	inStrobeUnit	const int		Number of strobe device to use
	inSource	const VisionBoxStrobeSource::Type		Selects the signal used as trigger source
	inInvertTrigger	const int		Whether to invert trigger signal

Description

This filter is simple wrapper over Strobe::SetTriggerSource SDK function. Refer to AGE-X SDK to see how to use this function properly.

Remarks

Precondition for use of VisionBox filters is to have Imago Technologies AGE-X SKD installed on target computer. AGE-X SDK provides filters with necessary libraries (DLL files) and drivers. After installation of AGE-X SDK, system restart may be required.

Recommended AGE-X SDK version for Aurora Vision Studio usage is **1.6.8.0**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

VisionBox_StrobeSoftwareTrigger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets strobes software trigger.

Syntax

```
void avl::VisionBox_StrobeSoftwareTrigger
(
 VisionBox_State& ioState,
 const int inStrobeUnit,
 const bool inState
)
```

Parameters

	Name	Type	Default	Description
	ioState	VisionBox_State&		Object used to maintain state of the function.
	inStrobeUnit	const int		Number of strobe device to use
	inState	const bool		Set software trigger on or off

Description

This filter can be used to control strobes' trigger by software.

This filter is simple wrapper over Strobe::SwTrigger SDK function. Refer to AGE-X SDK to see how to use this function properly.

Remarks

Precondition for use of VisionBox filters is to have Imago Technologies AGE-X SKD installed on target computer. AGE-X SDK provides filters with necessary libraries (DLL files) and drivers. After installation of AGE-X SDK, system restart may be required.

Recommended AGE-X SDK version for Aurora Vision Studio usage is **1.6.8.0**.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

190. WebCamera

Table of content:

WebCamera_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a frame from a camera using Direct Show interface.

Applications: Not recommended for industrial operation.

Syntax

```
bool avl::WebCamera_GrabImage
(
 WebCamera_GrabImageState& ioState,
 atl::Optional<int> inCameraIndex,
 atl::Optional<avl::Size> inFrameSize,
 avl::WebCameraInputFormat::Type inPixelType,
 avl::Image& outFrame
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	WebCamera_GrabImageState&			Object used to maintain state of the function.
	inCameraIndex	Optional<int>	0 - +∞	NIL	Index number of camera device to connect with
	inFrameSize	Optional<Size>		NIL	Requested size of source frame in pixels selected from camera output formats
	inPixelType	WebCameraInputFormat::Type		Any	Requested specific pixel type from camera (may change image quality)
	outFrame	Image&			Captured frame

Description

Filter uses DirectShow API to acquire images from devices registered in system. Images can be acquired from many sources like: web cameras with support for DirectShow , video grabbers and system video streams.

Input **inCameraIndex** selects device in order provided by the operational system.

Input **inFrameSize** selects output image size in pixels. If an image in requested size is not available an error occurs. If value of **inFrameSize** is not provided filter returns image with first available size found.

Remarks

Best quality of images can be achieved using **inPixelType** with values **RGB24** or **RGB32**. Format **YUY2** stores two RGB pixels using only 4 bytes.

To check which pixel types are supported you can use application GraphStudioNext

.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

Errors

"Unable to connect with chosen Web Camera." - error will occur when provided **inCameraIndex** is not valid device number.

"Unable to connect with media type of Web Camera. Required image format may not be supported. " - selected **inFrameSize** is not supported by selected device.

191. XIMEA

Table of content:

XiApi_GenerateSoftwareTrigger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Generates software trigger for XIMEA device.

Syntax

```
void avl::XiApi_GenerateSoftwareTrigger
(
 XIAPIState& ioState,
 const atl::Optional<atl::String>& inDeviceID
)
```

Parameters

	Name	Type	Default	Description
	ioState	XIAPIState&		Object used to maintain state of the function.
	inDeviceID	const Optional<String>&	NIL	Camera chip ID or camera index

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

XiApi_GetGPILevel

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Returns the value from selected GPI (digital input).

Syntax

```
void avl::XiApi_GetGPILevel
(
 XIAPIState& ioState,
 const atl::Optional<atl::String>& inDeviceID,
 int inGPISelector,
 int& outGPILevel
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XIAPIState&			Object used to maintain state of the function.
	inDeviceID	const Optional<String>&		NIL	Camera chip ID or camera index
	inGPISelector	int	0 - +∞	1	GPI number
	outGPILevel	int&			

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

XiApi_GetParamFloat

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type Float from XIMEA device.

Syntax

```
void avl::XiApi_GetParamFloat
(
 XIAPIState& ioState,
 const atl::Optional<atl::String>& inDeviceID,
 const atl::String& inParameter,
 float& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	XIAPIState&		Object used to maintain state of the function.
	inDeviceID	const Optional<String>&	NIL	Camera chip ID or camera index
	inParameter	const String&		Parameter name
	outValue	float&		Received value

Remarks

The value of the parameter is retrieved in every filter iteration.

The possible values (strings) of **inParameter** input can be found in the XIMEA documentation at the following website .

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- XiApi_GrabImage – Captures an image from a XIMEA camera.
- XiApi_SetGPIMode – Defines selected GPI (digital input) functionality.
- XiApi_GetGPILevel – Returns the value from selected GPI (digital input).
- XiApi_SetGPOMode – Defines GPO (digital output) functionality.
- XiApi_SetParamInt – Sets parameter of type Integer in XIMEA device.
- XiApi_SetParamFloat – Sets parameter of type Float in XIMEA device.
- XiApi_GetParamInt – Gets parameter of type Integer from XIMEA device.
- XiApi_GenerateSoftwareTrigger – Generates software trigger for XIMEA device.

XiApi_GetParamInt

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets parameter of type Integer from XIMEA device.

Syntax

```
void avl::XiApi_GetParamInt
(
 XIAPIState& ioState,
 const atl::Optional<atl::String>& inDeviceID,
 const atl::String& inParameter,
 int& outValue
)
```

Parameters

	Name	Type	Default	Description
↔	ioState	XIAPIState&		Object used to maintain state of the function.
→	inDeviceID	const Optional<String>&	NIL	Camera chip ID or camera index
→	inParameter	const String&		Parameter name
←	outValue	int&		Received value

Remarks

The value of the parameter is retrieved in every filter iteration.

The possible values (strings) of **inParameter** input can be found in the XIMEA documentation at the following website .

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- XiApi_GrabImage – Captures an image from a XIMEA camera.
- XiApi_SetGPIMode – Defines selected GPI (digital input) functionality.
- XiApi_GetGPILevel – Returns the value from selected GPI (digital input).
- XiApi_SetGPOMode – Defines GPO (digital output) functionality.
- XiApi_SetParamInt – Sets parameter of type Integer in XIMEA device.
- XiApi_SetParamFloat – Sets parameter of type Float in XIMEA device.
- XiApi_GetParamFloat – Gets parameter of type Float from XIMEA device.
- XiApi_GenerateSoftwareTrigger – Generates software trigger for XIMEA device.

XiApi_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an image from a XIMEA camera.

Syntax

```
bool avl::XiApi_GrabImage
(
 XIAPIState& ioState,
 const atl::Optional<atl::String>& inDeviceID,
 avl::XiApiImageDataFormat::Type inImageDataFormat,
 atl::Optional<const avl::XiApiTriggerSource::Type&> inTriggerSource,
 atl::Optional<const avl::XiApiDownsampling::Type&> inDownsampling,
 atl::Optional<int> inExposureTime,
 atl::Optional<float> inGain,
 atl::Optional<const avl::Box&> inAoi,
 atl::Optional<bool> inBadPixelCorrection,
 atl::Optional<bool> inAutomaticExposureGain,
 avl::Image& outImage,
 atl::int64& outFrameID,
 atl::int64& outTimestamp
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XIAPIState&			Object used to maintain state of the function.
	inDeviceID	const Optional<String>&		NIL	Camera chip ID or camera index
	inImageDataFormat	XiApiImageDataFormat::Type			Output data format
	inTriggerSource	Optional<const XiApiTriggerSource::Type&>		NIL	Defines source of trigger
	inDownsampling	Optional<const XiApiDownsampling::Type&>		NIL	Changes image resolution by binning or skipping
	inExposureTime	Optional<int>	0 - +∞	NIL	Exposure time in microseconds
	inGain	Optional<float>		NIL	Gain in dB
	inAoi	Optional<const Box&>		NIL	Required fragment of image to stream
	inBadPixelCorrection	Optional<bool>		NIL	Correction of bad pixels
	inAutomaticExposureGain	Optional<bool>		NIL	Automatic exposure gain
	outImage	Image&			Captured frame
	outFrameID	int64&			Captured frame ID
	outTimestamp	int64&			Captured frame timestamp

Description

To be able to use a XIMEA camera, you need to install camera driver. You can find it at the following address (select binaries):

https://www.ximea.com/support/wiki/apis/XIMEA_API_Software_Package

Pleas make sure that xiApi SDK is properly installed on your computer. To verify the driver installation, you can run xiApiViewer.exe. If the camera was detected and you can see the image from it, you can use your XIMEA camera in Aurora Vision Studio.

Recommended xiApi SDK version for Aurora Vision Studio usage is **4.26**.

Setting parameters

All the auto parameters (those with checkboxes in the Property window) are initially set to the default values of the camera (they are not changed in Aurora Vision Studio). You can modify them by checking a checkbox and entering your own value.

The **inDeviceID** parameter is set only once when the filter is executed for the first time (if it set to Auto, the first available camera in the system will be used). All the other parameters are set during the first filter execution or when the parameter is changed (they are not set in every iteration when the value is not changed).

In **inAoi** parameter, the values of X, Y and Height should be even and the value of Width should be divisible by 4 or 16 (the exact value depends on the camera model).

To set more complex parameters, please use one of the following filters: XiApi_SetParamInt, XiApi_SetParamFloat. Please note, that if you set some parameter value in the XiApi_GrabImage filter you should not modify it by xiApi_SetParamXxx filter - it may cause problems.

Remarks

The full description of camera parameters can be found at the XIMEA website .

Alternatively you can also use GenICam filters to work with XIMEA cameras. Please refer to the article Working with Gen TL devices.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- XiApi_GrabImage_WithTimeout – Captures an image from a XIMEA camera.
- XiApi_StartAcquisition – Initializes and starts image acquisition in a camera.
- XiApi_StopAcquisition – Stops image acquisition in a camera.
- XiApi_SetGPIMode – Defines selected GPI (digital input) functionality.
- XiApi_GetGPILevel – Returns the value from selected GPI (digital input).
- XiApi_SetGPOMode – Defines GPO (digital output) functionality.
- XiApi_SetParamInt – Sets parameter of type Integer in XIMEA device.
- XiApi_SetParamFloat – Sets parameter of type Float in XIMEA device.
- XiApi_GetParamInt – Gets parameter of type Integer from XIMEA device.
- XiApi_GetParamFloat – Gets parameter of type Float from XIMEA device.
- XiApi_GenerateSoftwareTrigger – Generates software trigger for XIMEA device.

XiApi_GrabImage_WithTimeout

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an image from a XIMEA camera.

Syntax

```
bool avl::XiApi_GrabImage_WithTimeout
(
 XIAPIState& ioState,
 const atl::Optional<atl::String>& inDeviceID,
 int inTimeout,
 avl::XiApiImageDataFormat::Type inImageDataFormat,
 atl::Optional<const avl::XiApiTriggerSource::Type&> inTriggerSource,
 atl::Optional<const avl::XiApiDownsampling::Type&> inDownsampling,
 atl::Optional<int> inExposureTime,
 atl::Optional<float> inGain,
 atl::Optional<const avl::Box&> inAoi,
 atl::Optional<bool> inBadPixelCorrection,
 atl::Optional<bool> inAutomaticExposureGain,
 atl::Conditional<avl::Image>& outImage,
 atl::Conditional<atl::int64>& outFrameID,
 atl::Conditional<atl::int64>& outTimestamp
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XIAPIState&			Object used to maintain state of the function.
	inDeviceID	const Optional<String>&		NIL	Camera chip ID or camera index
	inTimeout	int	0 - ∞	100	Maximum time to wait for frame in milliseconds
	inImageDataFormat	XiApiImageDataFormat::Type			Output data format
	inTriggerSource	Optional<const XiApiTriggerSource::Type&>		NIL	Defines source of trigger
	inDownsampling	Optional<const XiApiDownsampling::Type&>		NIL	Changes image resolution by binning or skipping
	inExposureTime	Optional<int>	0 - + ∞	NIL	Exposure time in microseconds
	inGain	Optional<float>		NIL	Gain in dB
	inAoi	Optional<const Box&>		NIL	Required fragment of image to stream
	inBadPixelCorrection	Optional<bool>		NIL	Correction of bad pixels
	inAutomaticExposureGain	Optional<bool>		NIL	Automatic exposure gain
	outImage	Conditional<Image>&			Captured frame
	outFrameID	Conditional<int64>&			Captured frame ID
	outTimestamp	Conditional<int64>&			Captured frame timestamp

Description

To be able to use a XIMEA camera, you need to install camera driver. You can find it at the following address (select binaries):

https://www.ximea.com/support/wiki/apis/XIMEA_API_Software_Package

Pleas make sure that xiApi SDK is properly installed on your computer. To verify the driver installation, you can run xiApiViewer.exe. If the camera was detected and you can see the image from it, you can use your XIMEA camera in Aurora Vision Studio.

Recommended xiApi SDK version for Aurora Vision Studio usage is **4.26**.

Setting parameters

All the auto parameters (those with checkboxes in the Property window) are initially set to the default values of the camera (they are not changed in Aurora Vision Studio). You can modify them by checking a checkbox and entering your own value.

The **inDeviceID** parameter is set only once when the filter is executed for the first time (if it set to Auto, the first available camera in the system will be used). All the other parameters are set during the first filter execution or when the parameter is changed (they are not set in every iteration when the value is not changed).

In **inAoi** parameter, the values of X, Y and Height should be even and the value of Width should be divisible by 4 or 16 (the exact value depends on the camera model).

To set more complex parameters, please use one of the following filters: XiApi_SetParamInt, XiApi_SetParamFloat. Please note, that if you set some parameter value in the XiApi_GrabImage filter you should not modify it by xiApi_SetParamXxx filter - it may cause problems.

Remarks

The full description of camera parameters can be found at the XIMEA website .

Alternatively you can also use GenICam filters to work with XIMEA cameras. Please refer to the article Working with Gen TL devices.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- XiApi_GrabImage – Captures an image from a XIMEA camera.
- XiApi_StartAcquisition – Initializes and starts image acquisition in a camera.
- XiApi_StopAcquisition – Stops image acquisition in a camera.
- XiApi_SetGPIMode – Defines selected GPI (digital input) functionality.
- XiApi_GetGPILevel – Returns the value from selected GPI (digital input).
- XiApi_SetGPOMode – Defines GPO (digital output) functionality.
- XiApi_SetParamInt – Sets parameter of type Integer in XIMEA device.
- XiApi_SetParamFloat – Sets parameter of type Float in XIMEA device.
- XiApi_GetParamInt – Gets parameter of type Integer from XIMEA device.
- XiApi_GetParamFloat – Gets parameter of type Float from XIMEA device.
- XiApi_GenerateSoftwareTrigger – Generates software trigger for XIMEA device.

XiApi_SetGPIMode

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Defines selected GPI (digital input) functionality.

Syntax

```
void avl::XiApi_SetGPIMode
(
 XIAPIState& ioState,
 const atl::Optional<atl::String>& inDeviceID,
 int inGPISelector,
 avl::XiApiGPIMode::Type inGPIMode
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XIAPIState&			Object used to maintain state of the function.
	inDeviceID	const Optional<String>&		NIL	Camera chip ID or camera index
	inGPISelector	int	0 - +∞	1	GPI number
	inGPIMode	XiApiGPIMode::Type			

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

XiApi_SetGPOMode

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Defines GPO (digital output) functionality.

Syntax

```
void avl::XiApi_SetGPOMode
(
 XIAPIState& ioState,
 const atl::Optional<atl::String>& inDeviceID,
 int inGPOSelector,
 avl::XiApiGPOMode::Type inGPOMode
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XIAPIState&			Object used to maintain state of the function.
	inDeviceID	const Optional<String>&		NIL	Camera chip ID or camera index
	inGPOSelector	int	0 - + ∞	1	GPO number
	inGPOMode	XiApiGPOMode::Type			

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

XiApi_SetParamFloat

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type Float in XIMEA device.

Syntax

```
void avl::XiApi_SetParamFloat
(
 XIAPIState& ioState,
 const atl::Optional<atl::String>& inDeviceID,
 const atl::String& inParameter,
 float inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	XIAPIState&		Object used to maintain state of the function.
	inDeviceID	const Optional<String>&	NIL	Camera chip ID or camera index
	inParameter	const String&		Parameter name
	inValue	float		Value to set

Remarks

The value of the parameter is set in every filter iteration. Please note, that if you set some parameter value in this filter you should not modify it by XiApi_GrabImage filter - it may cause problems.

The possible values (strings) of **inParameter** input can be found in the XIMEA documentation at the following website .

The full description of camera parameters can be found at the XIMEA website .

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- XiApi_GrabImage – Captures an image from a XIMEA camera.
- XiApi_SetGPIMode – Defines selected GPI (digital input) functionality.
- XiApi_GetGPILevel – Returns the value from selected GPI (digital input).
- XiApi_SetGPOMode – Defines GPO (digital output) functionality.
- XiApi_SetParamInt – Sets parameter of type Integer in XIMEA device.
- XiApi_GetParamInt – Gets parameter of type Integer from XIMEA device.
- XiApi_GetParamFloat – Gets parameter of type Float from XIMEA device.
- XiApi_GenerateSoftwareTrigger – Generates software trigger for XIMEA device.

XiApi_SetParamInt

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets parameter of type Integer in XIMEA device.

Syntax

```
void avl::XiApi_SetParamInt
(
 XIAPIState& ioState,
 const atl::Optional<atl::String>& inDeviceID,
 const atl::String& inParameter,
 int inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	XIAPIState&		Object used to maintain state of the function.
	inDeviceID	const Optional<String>&	NIL	Camera chip ID or camera index
	inParameter	const String&		Parameter name
	inValue	int		Value to set

Remarks

The value of the parameter is set in every filter iteration. Please note, that if you set some parameter value in this filter you should not modify it by XiApi_GrabImage filter - it may cause problems.

The possible values (strings) of **inParameter** input can be found in the XIMEA documentation at the following website .

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- XiApi_GrabImage – Captures an image from a XIMEA camera.
- XiApi_SetGPIMode – Defines selected GPI (digital input) functionality.
- XiApi_GetGPILevel – Returns the value from selected GPI (digital input).
- XiApi_SetGPOMode – Defines GPO (digital output) functionality.
- XiApi_SetParamFloat – Sets parameter of type Float in XIMEA device.
- XiApi_GetParamInt – Gets parameter of type Integer from XIMEA device.
- XiApi_GetParamFloat – Gets parameter of type Float from XIMEA device.
- XiApi_GenerateSoftwareTrigger – Generates software trigger for XIMEA device.

XiApi_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Initializes and starts image acquisition in a camera.

Syntax

```
void avl::XiApi_StartAcquisition
(
 XIAPIState& ioState,
 const atl::Optional<atl::String>& inDeviceID,
 avl::XiApiImageDataFormat::Type inImageDataFormat,
 atl::Optional<const avl::XiApiTriggerSource::Type&> inTriggerSource,
 atl::Optional<const avl::XiApiDownsampling::Type&> inDownsampling,
 atl::Optional<int> inExposureTime,
 atl::Optional<float> inGain,
 atl::Optional<const avl::Box&> inAoi,
 atl::Optional<bool> inBadPixelCorrection,
 atl::Optional<bool> inAutomaticExposureGain
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XIAPIState&			Object used to maintain state of the function.
	inDeviceID	const Optional<String>&		NIL	Camera chip ID or camera index
	inImageDataFormat	XiApiImageDataFormat::Type			Output data format
	inTriggerSource	Optional<const XiApiTriggerSource::Type&>		NIL	Defines source of trigger
	inDownsampling	Optional<const XiApiDownsampling::Type&>		NIL	Changes image resolution by binning or skipping
	inExposureTime	Optional<int>	0 - +∞	NIL	Exposure time in microseconds
	inGain	Optional<float>		NIL	Gain in dB
	inAoi	Optional<const Box&>		NIL	Required fragment of image to stream
	inBadPixelCorrection	Optional<bool>		NIL	Correction of bad pixels
	inAutomaticExposureGain	Optional<bool>		NIL	Automatic exposure gain

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

XiApi_StopAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Stops image acquisition in a camera.

Syntax

```
void avl::XiApi_StopAcquisition
(
 XIAPIState& ioState,
 const atl::Optional<atl::String>& inDeviceID
)
```

Parameters

	Name	Type	Default	Description
	ioState	XIAPIState&		Object used to maintain state of the function.
	inDeviceID	const Optional<String>&	NIL	Camera chip ID or camera index

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

192. XSightSmartCamera

Table of content:

XSightSmartCamera_GenerateSoftwareTrigger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Generates software trigger.

Syntax

```
void avl::XSightSmartCamera_GenerateSoftwareTrigger
(
 XSightSmartCamera_State& ioState,
 atl::Optional<int> inDeviceIndex,
 bool inGenerate
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSightSmartCamera_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inGenerate	bool		True	

Remarks

Camera driver software

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight smart camera identifying internal index.

Camera device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight smart camera SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0**.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

See Also

- XSightSmartCamera_GrabImage – Captures an image using X-Sight smart camera.
- XSightSmartCamera_StartAcquisition – Starts image acquisition.

XSightSmartCamera_GetDigitalInput

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets digital input.

Syntax

```
void avl::XSightSmartCamera_GetDigitalInput
(
 XSightSmartCamera_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort,
 bool& outValue
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSightSmartCamera_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inPort	int	1 - 4		
	outValue	bool&			

Remarks

Camera driver software

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight smart camera identifying internal index.

Camera device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight smart camera SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0**.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

See Also

- XSightSmartCamera_GrabImage – Captures an image using X-Sight smart camera.
- XSightSmartCamera_StartAcquisition – Starts image acquisition.

XSightSmartCamera_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an image using X-Sight smart camera.

Syntax

```
bool avl::XSightSmartCamera_GrabImage
(
 XSightSmartCamera_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inInputQueueSize,
 atl::Optional<const avl::XSightSmartCameraImageFormat&> inImageFormat,
 atl::Optional<int> inExposureTime,
 atl::Optional<int> inFrameRate,
 atl::Optional<bool> inGainAuto,
 atl::Optional<const atl::Array<int>&> inGain,
 atl::Optional<int> inBlackLevel,
 atl::Optional<bool> inWhiteBalanceAuto,
 atl::Optional<int> inContrast,
 atl::Optional<int> inGamma,
 atl::Optional<int> inLumination,
 atl::Optional<int> inSaturation,
 atl::Optional<int> inHUE,
 atl::Optional<int> inColorCorrection,
 atl::Optional<int> inSharpness,
 atl::Optional<bool> inTriggerMode,
 atl::Optional<const avl::XSightSmartCameraTriggerConfiguration&> inTriggerConfiguration,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSightSmartCamera_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>		NIL	Device identifying number
	inInputQueueSize	int	1 - ∞	5	Capacity of output frames queue
	inImageFormat	Optional<const XSightSmartCameraImageFormat&>		NIL	
	inExposureTime	Optional<int>	1 - ∞	NIL	Exposure time in milliseconds
	inFrameRate	Optional<int>	1 - ∞	NIL	
	inGainAuto	Optional<bool>		NIL	
	inGain	Optional<const Array<int>&>		NIL	
	inBlackLevel	Optional<int>	1 - ∞	NIL	
	inWhiteBalanceAuto	Optional<bool>		NIL	
	inContrast	Optional<int>	1 - 128	NIL	
	inGamma	Optional<int>	0 - ∞	NIL	
	inLumination	Optional<int>	1 - 128	NIL	
	inSaturation	Optional<int>	1 - ∞	NIL	
	inHUE	Optional<int>	1 - ∞	NIL	
	inColorCorrection	Optional<int>	1 - ∞	NIL	
	inSharpness	Optional<int>	1 - ∞	NIL	
	inTriggerMode	Optional<bool>		NIL	
	inTriggerConfiguration	Optional<const XSightSmartCameraTriggerConfiguration&>		NIL	
	outImage	Image&			Captured image

Remarks

Camera driver software

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight smart camera identifying internal index.

Camera device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight smart camera SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0**.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

See Also

- XSightSmartCamera_GrabImage_WithTimeout – Captures with timeout an image using X-Sight smart camera.
- XSightSmartCamera_StartAcquisition – Starts image acquisition.

XSightSmartCamera_GrabImage_WithTimeout

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures with timeout an image using X-Sight smart camera.

Syntax

```
bool avl::XSightSmartCamera_GrabImage_WithTimeout
(
 XSightSmartCamera_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inInputQueueSize,
 const atl::Optional<int> inTimeout,
 atl::Optional<const avl::XSightSmartCameraImageFormat&> inImageFormat,
 atl::Optional<int> inExposureTime,
 atl::Optional<int> inFrameRate,
 atl::Optional<bool> inGainAuto,
 atl::Optional<const atl::Array<int>&> inGain,
 atl::Optional<int> inBlackLevel,
 atl::Optional<bool> inWhiteBalanceAuto,
 atl::Optional<int> inContrast,
 atl::Optional<int> inGamma,
 atl::Optional<int> inLumination,
 atl::Optional<int> inSaturation,
 atl::Optional<int> inHUE,
 atl::Optional<int> inColorCorrection,
 atl::Optional<int> inSharpness,
 atl::Optional<bool> inTriggerMode,
 atl::Optional<const avl::XSightSmartCameraTriggerConfiguration&> inTriggerConfiguration,
 atl::Conditional<avl::Image>& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSightSmartCamera_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>		NIL	Device identifying number
	inInputQueueSize	int	1 - ∞	5	Capacity of output frames queue
	inTimeout	const Optional<int>	1 - ∞	100	Frame timeout in milliseconds
	inImageFormat	Optional<const XSightSmartCameraImageFormat&>		NIL	
	inExposureTime	Optional<int>	1 - ∞	NIL	Exposure time in milliseconds
	inFrameRate	Optional<int>	1 - ∞	NIL	
	inGainAuto	Optional<bool>		NIL	
	inGain	Optional<const Array<int>&>		NIL	
	inBlackLevel	Optional<int>	1 - ∞	NIL	
	inWhiteBalanceAuto	Optional<bool>		NIL	
	inContrast	Optional<int>	1 - 128	NIL	
	inGamma	Optional<int>	0 - ∞	NIL	
	inLumination	Optional<int>	1 - 128	NIL	
	inSaturation	Optional<int>	1 - ∞	NIL	
	inHUE	Optional<int>	1 - ∞	NIL	
	inColorCorrection	Optional<int>	1 - ∞	NIL	
	inSharpness	Optional<int>	1 - ∞	NIL	
	inTriggerMode	Optional<bool>		NIL	
	inTriggerConfiguration	Optional<const XSightSmartCameraTriggerConfiguration&>		NIL	
	outImage	Conditional<Image>&			Captured image

Remarks

Camera driver software

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight smart camera identifying internal index.

Camera device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight smart camera SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0**.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

See Also

- XSightSmartCamera_GrabImage_WithTimeout – Captures with timeout an image using X-Sight smart camera.
- XSightSmartCamera_StartAcquisition – Starts image acquisition.

XSightSmartCamera_SetDigitalOutput

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets digital output.

Syntax

```
void avl::XSightSmartCamera_SetDigitalOutput
(
 XSightSmartCamera_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort,
 bool inValue
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSightSmartCamera_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inPort	int	1 - 4		
	inValue	bool			

Remarks

Camera driver software

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight smart camera identifying internal index.

Camera device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight smart camera SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0**.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

See Also

- XSightSmartCamera_GrabImage – Captures an image using X-Sight smart camera.
- XSightSmartCamera_StartAcquisition – Starts image acquisition.

XSightSmartCamera_SetPWMControl

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets PWM control properties.

Syntax

```
void avl::XSightSmartCamera_SetPWMControl
(
 XSightSmartCamera_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inValue
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSightSmartCamera_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inValue	int			

Remarks

Camera driver software

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight smart camera identifying internal index.

Camera device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight smart camera SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0**.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

See Also

- XSightSmartCamera_GrabImage – Captures an image using X-Sight smart camera.
- XSightSmartCamera_StartAcquisition – Starts image acquisition.

XSightSmartCamera_SetPWMLumination

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets PWM light.

Syntax

```
void avl::XSightSmartCamera_SetPWMLumination
(
 XSightSmartCamera_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inValue
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSightSmartCamera_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inValue	int	0 - 127		Current value adjustment range, unit: 1A/127

Remarks

Camera driver software

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight smart camera identifying internal index.

Camera device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight smart camera SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0**.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

See Also

- XSightSmartCamera_GrabImage – Captures an image using X-Sight smart camera.
- XSightSmartCamera_StartAcquisition – Starts image acquisition.

XSightSmartCamera_SetStatusOutput

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets status light.

Syntax

```
void avl::XSightSmartCamera_SetStatusOutput
(
 XSightSmartCamera_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort,
 bool inValue
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSightSmartCamera_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inPort	int	1 - 2		
	inValue	bool			

Remarks

Camera driver software

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight smart camera identifying internal index.

Camera device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight smart camera SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0**.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

See Also

- XSightSmartCamera_GrabImage – Captures an image using X-Sight smart camera.
- XSightSmartCamera_StartAcquisition – Starts image acquisition.

XSightSmartCamera_SetStrobeActive

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets strobe active.

Syntax

```
void avl::XSightSmartCamera_SetStrobeActive
(
 XSightSmartCamera_State& ioState,
 atl::Optional<int> inDeviceIndex,
 bool inActive
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSightSmartCamera_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inActive	bool			

Remarks

Camera driver software

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight smart camera identifying internal index.

Camera device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight smart camera SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0**.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

See Also

- XSightSmartCamera_GrabImage – Captures an image using X-Sight smart camera.
- XSightSmartCamera_StartAcquisition – Starts image acquisition.

XSightSmartCamera_SetStrobeAhead

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets strobe ahead.

Syntax

```
void avl::XSightSmartCamera_SetStrobeAhead
(
 XSightSmartCamera_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inCount
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSightSmartCamera_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inCount	int			

Remarks

Camera driver software

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight smart camera identifying internal index.

Camera device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight smart camera SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0**.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

See Also

- XSightSmartCamera_GrabImage – Captures an image using X-Sight smart camera.
- XSightSmartCamera_StartAcquisition – Starts image acquisition.

XSightSmartCamera_SetStrobeInvert

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets strobe invert.

Syntax

```
void avl::XSightSmartCamera_SetStrobeInvert
(
 XSightSmartCamera_State& ioState,
 atl::Optional<int> inDeviceIndex,
 bool inInvert
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSightSmartCamera_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inInvert	bool			

Remarks

Camera driver software

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight smart camera identifying internal index.

Camera device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight smart camera SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0**.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

See Also

- XSightSmartCamera_GrabImage – Captures an image using X-Sight smart camera.
- XSightSmartCamera_StartAcquisition – Starts image acquisition.

XSightSmartCamera_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Starts image acquisition.

Syntax

```
void avl::XSightSmartCamera_StartAcquisition
(
 XSightSmartCamera_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inInputQueueSize,
 atl::Optional<const avl::XSightSmartCameraImageFormat&> inImageFormat,
 atl::Optional<int> inExposureTime,
 atl::Optional<int> inFrameRate,
 atl::Optional<bool> inGainAuto,
 atl::Optional<const atl::Array<int>&> inGain,
 atl::Optional<int> inBlackLevel,
 atl::Optional<bool> inWhiteBalanceAuto,
 atl::Optional<int> inContrast,
 atl::Optional<int> inGamma,
 atl::Optional<int> inLumination,
 atl::Optional<int> inSaturation,
 atl::Optional<int> inHUE,
 atl::Optional<int> inColorCorrection,
 atl::Optional<int> inSharpness,
 atl::Optional<bool> inTriggerMode,
 atl::Optional<const avl::XSightSmartCameraTriggerConfiguration&> inTriggerConfiguration
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSightSmartCamera_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inInputQueueSize	int	1 - ∞	5	Capacity of output frames queue
	inImageFormat	Optional<const XSightSmartCameraImageFormat&>		NIL	
	inExposureTime	Optional<int>	1 - ∞	NIL	Exposure time in milliseconds
	inFrameRate	Optional<int>	1 - ∞	NIL	
	inGainAuto	Optional<bool>		NIL	
	inGain	Optional<const Array<int>&>		NIL	
	inBlackLevel	Optional<int>	1 - ∞	NIL	
	inWhiteBalanceAuto	Optional<bool>		NIL	
	inContrast	Optional<int>	1 - 128	NIL	
	inGamma	Optional<int>	0 - ∞	NIL	
	inLumination	Optional<int>	1 - 128	NIL	
	inSaturation	Optional<int>	1 - ∞	NIL	
	inHUE	Optional<int>	1 - ∞	NIL	
	inColorCorrection	Optional<int>	1 - ∞	NIL	
	inSharpness	Optional<int>	1 - ∞	NIL	
	inTriggerMode	Optional<bool>		NIL	
	inTriggerConfiguration	Optional<const XSightSmartCameraTriggerConfiguration&>		NIL	

Remarks

Camera driver software

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight smart camera identifying internal index.

Camera device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight smart camera SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0**.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

See Also

- XSightSmartCamera_GrabImage_WithTimeout – Captures with timeout an image using X-Sight smart camera.
- XSightSmartCamera_GrabImage – Captures an image using X-Sight smart camera.

XSightSmartCamera_StopAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Stops image acquisition.

Syntax

```
void avl::XSightSmartCamera_StopAcquisition
(
 XSightSmartCamera_State& ioState,
 atl::Optional<int> inDeviceIndex
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSightSmartCamera_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number

Remarks

Camera driver software

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight smart camera identifying internal index.

Camera device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight smart camera SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0**.

All the other parameters exposed by camera filters are optional, setting them to 'Auto' leaves related parameter unchanged in camera (device default or user set configuration).

See Also

- XSightSmartCamera_GrabImage_WithTimeout – Captures with timeout an image using X-Sight smart camera.
- XSightSmartCamera_StartAcquisition – Starts image acquisition.

193. XSight

Table of content:

- XSight_SetOutputLatchEnable
- XSight_SetOutputLatchSource
- XSight_SetOutputMode
- XSight_SetOutputPulse
- XSight_SetOutputPulseConfiguration
- XSight_SetOutputPulseSoft
- XSight_SetOutputPulseTriggered
- XSight_SetPWMFlashTime
- XSight_SetPWMParameters
- XSight_SetPWMStart
- XSight_SetPWMStop
- XSight_SetPWMTriggerMode
- XSight_SetPWMTriggerSource
- XSight_WriteOutputLevel_Multiple
- XSight_WriteOutputLevel_Single

XSight_GetCompareStatus

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets encoder stop.

Syntax

```
void avl::XSight_GetCompareStatus
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inChannel,
 bool& outFinished,
 int& outCount
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inChannel	int	1 - 16	1	
	outFinished	bool&			
	outCount	int&			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_GetEncoderCaptureConfiguration

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets encoder capture configuration.

Syntax

```
void avl::XSight_GetEncoderCaptureConfiguration
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inChannel,
 int& outPort,
 int& outEncoder
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inChannel	int	0 - 16	1	Encoder channel
	outPort	int&			:Trigger input signal
	outEncoder	int&			:Captured encoder channel

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_GetEncoderCaptureCount

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets encoder capture count.

Syntax

```
void avl::XSight_GetEncoderCaptureCount
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inChannel,
 atl::int64& outCount
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inChannel	int	0 - 16	1	Encoder channel
	outCount	int64&			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_GetEncoderCaptureStatus

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets compare status.

Syntax

```
void avl::XSight_GetEncoderCaptureStatus
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inChannel,
 bool& outActive
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inChannel	int	0 - 16	1	Encoder channel
	outActive	bool&			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_GetEncoderCount

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets encoder counter.

Syntax

```
void avl::XSight_GetEncoderCount
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inChannel,
 atl::int64& outCount
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inChannel	int	0 - 16	1	Encoder channel
	outCount	int64&			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_GetEncoderDirection

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets encoder direction.

Syntax

```
void avl::XSight_GetEncoderDirection
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inChannel,
 avl::XSightEncoderDirection::Type& outDirection
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inChannel	int	0 - 16	0	Encoder channel
	outDirection	XSightEncoderDirection::Type&			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_GetEncoderEnable

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets encoder enable status.

Syntax

```
void avl::XSight_GetEncoderEnable
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inChannel,
 bool& outEnable
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inChannel	int	0 - 16	0	Encoder channel
	outEnable	bool&			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_GetEncoderSide

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets encoder side.

Syntax

```
void avl::XSight_GetEncoderSide
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inChannel,
 avl::XSightEncoderSide::Type& outSide
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inChannel	int	0 - 16	1	Encoder channel
	outSide	XSightEncoderSide::Type&			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_GetInputCounter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets input port counter value.

Syntax

```
void avl::XSight_GetInputCounter
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort,
 int& outCount
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number.
	inPort	int	1 - 16	1	IO port description number.
	outCount	int&			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_GetInputFilter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets input filter property.

Syntax

```
void avl::XSight_GetInputFilter
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort,
 int& outFilterTime
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number.
	inPort	int	1 - 16	1	IO port description number.
	outFilterTime	int&			Filter time in us unit.

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_GetInputInvertMode

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets input port inverse property.

Syntax

```
void avl::XSight_GetInputInvertMode
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort,
 bool& outInvert
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inPort	int	1 - 16	1	IO port description number.
	outInvert	bool&			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_GetInputMode

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads input port working mode.

Syntax

```
void avl::XSight_GetInputMode
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort,
 avl::XSightInputMode::Type& outMode
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inPort	int	1 - 16	1	IO port description number.
	outMode	XSightInputMode::Type&			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_GetOutputInvertMode

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets output port inverse property.

Syntax

```
void avl::XSight_GetOutputInvertMode
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort,
 bool& outInvert
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inPort	int	1 - 16	1	IO port description number.
	outInvert	bool&			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_GetOutputLatchCount

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets output latch count.

Syntax

```
void avl::XSight_GetOutputLatchCount
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort,
 atl::int64& outCount
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number.
	inPort	int	1 - 16	1	IO port description number.
	outCount	int64&			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_GetOutputLatchStatus

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets output latch status.

Syntax

```
void avl::XSight_GetOutputLatchStatus
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort,
 bool& outFinished
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number.
	inPort	int	1 - 16	1	IO port description number.
	outFinished	bool&			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_GetOutputMode

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets output port working mode.

Syntax

```
void avl::XSight_GetOutputMode
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort,
 avl::XSightOutputMode::Type& outMode
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inPort	int	1 - 16	1	IO port description number.
	outMode	XSightOutputMode::Type&			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_GetOutputPulseBufferStatus

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets output pulse buffer status.

Syntax

```
void avl::XSight_GetOutputPulseBufferStatus
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort,
 bool& outFull
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number.
	inPort	int	1 - 16	1	IO port description number.
	outFull	bool&			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_GetOutputPulseConfiguration

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets output pulse configuration.

Syntax

```
void avl::XSight_GetOutputPulseConfiguration
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort,
 atl::int64& outDelay,
 atl::int64& outWidth,
 atl::int64& outCycle,
 atl::int64& outCount,
 int& outSource
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number.
	inPort	int	1 - 16	1	Output port description number.
	outDelay	int64&			Output delay in us unit.
	outWidth	int64&			Pulse width in us unit.
	outCycle	int64&			Pulse cycle in us unit.
	outCount	int64&			
	outSource	int&			Source index depends on port operation mode

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_GetOutputPulseCount

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets output pulse count.

Syntax

```
void avl::XSight_GetOutputPulseCount
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort,
 atl::int64& outCount
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number.
	inPort	int	1 - 16	1	IO port description number.
	outCount	int64&			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_GetOutputPulseRunStatus

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets output pulse run status.

Syntax

```
void avl::XSight_GetOutputPulseRunStatus
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort,
 bool& outReady
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number.
	inPort	int	1 - 16	1	IO port description number.
	outReady	bool&			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_GetOutputPulseStatus

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets output pulse status.

Syntax

```
void avl::XSight_GetOutputPulseStatus
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort,
 bool& outOverflow
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number.
	inPort	int	1 - 16	1	IO port description number.
	outOverflow	bool&			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_GetPWMFlashTime

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets PWM flash time.

Syntax

```
void avl::XSight_GetPWMFlashTime
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inChannel,
 atl::int64& outTime
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inChannel	int	1 - 6	1	PWM channel description number.
	outTime	int64&			Time in us unit.

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_GetPWMParameters

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets PWM channel parameters.

Syntax

```
void avl::XSight_GetPWMParameters
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inChannel,
 int& outFrequency,
 int& outDutyCycle
)
```

Parameters

	Name	Type	Range	Default	Description
↻	ioState	XSight_State&			Object used to maintain state of the function.
→	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
→	inChannel	int	1 - 6	1	PWM channel description number.
←	outFrequency	int&			Output frequency as Hz.
←	outDutyCycle	int&			Duty cycle.

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_GetPWMState

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets PWM state.

Syntax

```
void avl::XSight_GetPWMState
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inChannel,
 bool& outEnable
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inChannel	int	1 - 6	1	PWM channel description number.
	outEnable	bool&			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_GetPWMTriggerMode

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets PWM channel working mode.

Syntax

```
void avl::XSight_GetPWMTriggerMode
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inChannel,
 avl::XSightPWMTrigger::Type& outMode
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inChannel	int	1 - 6	1	PWM channel description number.
	outMode	XSightPWMTrigger::Type&			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_GetPWMTriggerSource

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets PWM trigger source.

Syntax

```
void avl::XSight_GetPWMTriggerSource
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inChannel,
 int& outPort
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inChannel	int	1 - 6	1	PWM channel description number.
	outPort	int&			Input port description number.

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_OpenDevice

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Opens IO device.

Syntax

```
void avl::XSight_OpenDevice
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_ReadInputLevel_Multiple

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads input port levels from device in one routine.

Syntax

```
void avl::XSight_ReadInputLevel_Multiple
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 atl::Array<avl::XSightIOLevel::Type>& outLevels
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	outLevels	Array<XSightIOLevel::Type>&			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_ReadInputLevel_Single

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads input port level from device.

Syntax

```
void avl::XSight_ReadInputLevel_Single
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort,
 avl::XSightIOLevel::Type& outLevel
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inPort	int	1 - 16	1	IO port description number.
	outLevel	XSightIOLevel::Type&			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_ReadOutputLevel_Multiple

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads output port levels from device in one routine.

Syntax

```
void avl::XSight_ReadOutputLevel_Multiple
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 atl::Array<avl::XSightIOLevel::Type>& outLevels
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	outLevels	Array<XSightIOLevel::Type>&			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_ReadOutputLevel_Single

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Reads output port level from device.

Syntax

```
void avl::XSight_ReadOutputLevel_Single
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort,
 avl::XSightIOLevel::Type& outLevel
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inPort	int	1 - 16	1	IO port description number.
	outLevel	XSightIOLevel::Type&			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_ResetEncoderCount

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Resets encoder counter.

Syntax

```
void avl::XSight_ResetEncoderCount
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inChannel
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inChannel	int	0 - 16	1	Encoder channel

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_ResetInputCounter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Resets input port counter value.

Syntax

```
void avl::XSight_ResetInputCounter
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inPort	int	1 - 16	1	IO port description number.

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_ResetOutputPulseCount

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Resets output pulse count.

Syntax

```
void avl::XSight_ResetOutputPulseCount
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number.
	inPort	int	1 - 16	1	IO port description number.

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_SetCompareDataStart

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets compare data.

Syntax

```
void avl::XSight_SetCompareDataStart
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inChannel,
 int inCount
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inChannel	int	1 - ∞	1	
	inCount	int	1 - 15		

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_SetCompareGTStart

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets compare greater than.

Syntax

```
void avl::XSight_SetCompareGTStart
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inChannel,
 int inPosition
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inChannel	int	1 - 16	1	
	inPosition	int			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_SetCompareLinearStart

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets compare linear stop.

Syntax

```
void avl::XSight_SetCompareLinearStart
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inChannel,
 int inStartPosition,
 int inRepeatTimes,
 int inInterval
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inChannel	int	1 - ∞	1	Encoder channel
	inStartPosition	int			The first comparison position
	inRepeatTimes	int	0 - 15		Number of comparison, 0 means unlimited comparison times
	inInterval	int	0 - ∞	1	

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_SetCompareStop

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets encoder stop.

Syntax

```
void avl::XSight_SetCompareStop
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_SetEncoderCaptureConfiguration

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets encoder capture configuration.

Syntax

```
void avl::XSight_SetEncoderCaptureConfiguration
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inChannel,
 int inPort,
 int inEncoder
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inChannel	int	0 - 16	1	Encoder channel
	inPort	int	1 - 16	1	:Trigger input signal
	inEncoder	int	0 - 16	0	:Captured encoder channel

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_SetEncoderCaptureStart

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets encoder start.

Syntax

```
void avl::XSight_SetEncoderCaptureStart
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inChannel
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inChannel	int	0 - 16	1	Encoder channel

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_SetEncoderCaptureStop

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets encoder stop.

Syntax

```
void avl::XSight_SetEncoderCaptureStop
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inChannel
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inChannel	int	0 - 16	1	Encoder channel

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_SetEncoderDirection

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets encoder direction.

Syntax

```
void avl::XSight_SetEncoderDirection
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inChannel,
 avl::XSightEncoderDirection::Type inDirection
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inChannel	int	0 - 16	0	Encoder channel
	inDirection	XSightEncoderDirection::Type			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_SetEncoderEnable

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets encoder enable status.

Syntax

```
void avl::XSight_SetEncoderEnable
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inChannel,
 bool inEnable
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inChannel	int	0 - 16	0	Encoder channel
	inEnable	bool		True	

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_SetEncoderSide

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets encoder side.

Syntax

```
void avl::XSight_SetEncoderSide
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inChannel,
 avl::XSightEncoderSide::Type inSide
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inChannel	int	0 - 16	1	Encoder channel
	inSide	XSightEncoderSide::Type			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_SetInputFilter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Setups input filter.

Syntax

```
void avl::XSight_SetInputFilter
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort,
 int inFilterTime
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number.
	inPort	int	1 - 16	1	IO port description number.
	inFilterTime	int	0 - ∞	100	Filter time in us unit.

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_SetInputInvertMode

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets input port inverse property.

Syntax

```
void avl::XSight_SetInputInvertMode
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort,
 bool inInvert
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inPort	int	1 - 16	1	IO port description number.
	inInvert	bool		True	

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_SetInputMode

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets input port working mode.

Syntax

```
void avl::XSight_SetInputMode
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort,
 avl::XSightInputMode::Type inMode
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inPort	int	1 - 16	1	IO port description number.
	inMode	XSightInputMode::Type			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_SetOutputInvertMode

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets output port inverse property.

Syntax

```
void avl::XSight_SetOutputInvertMode
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort,
 bool inInvert
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inPort	int	1 - 16	1	IO port description number.
	inInvert	bool		True	

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_SetOutputLatchEnable

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets output latch enable.

Syntax

```
void avl::XSight_SetOutputLatchEnable
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort,
 bool inEnable
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number.
	inPort	int	1 - 16	1	Output port description number.
	inEnable	bool			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_SetOutputLatchSource

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets output latch source.

Syntax

```
void avl::XSight_SetOutputLatchSource
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort,
 int inTriggerSource
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number.
	inPort	int	1 - 16	1	Output port description number.
	inTriggerSource	int	1 - 16	1	Input port description number.

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_SetOutputMode

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets output port working mode.

Syntax

```
void avl::XSight_SetOutputMode
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort,
 avl::XSightOutputMode::Type inMode
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inPort	int	1 - 16	1	IO port description number.
	inMode	XSightOutputMode::Type			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_SetOutputPulse

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets output pulse.

Syntax

```
void avl::XSight_SetOutputPulse
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort,
 int inDelay,
 int inPulseWidth
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number.
	inPort	int	1 - 16	1	IO port description number.
	inDelay	int	0 - ∞	0	Output delay in us unit.
	inPulseWidth	int	1 - ∞	100	Pulse width in us unit.

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

This function is old way to start pulse. Use XSight_SetOutputPulseSoft.

XSight_SetOutputPulseConfiguration

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets output pulse configuration.

Syntax

```
void avl::XSight_SetOutputPulseConfiguration
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort,
 atl::int64 inDelay,
 atl::int64 inWidth,
 atl::int64 inCycle,
 atl::int64 inCount,
 int inSource
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number.
	inPort	int	1 - 16	1	Output port description number.
	inDelay	int64	0 - ∞	0L	Output delay in us unit.
	inWidth	int64	1 - ∞	100L	Pulse width in us unit.
	inCycle	int64	0 - ∞	0L	Pulse cycle in us unit.
	inCount	int64	0 - ∞	0L	
	inSource	int	0 - ∞	1	Source index depends on port operation mode

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_SetOutputPulseSoft

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets output pulse soft.

Syntax

```
void avl::XSight_SetOutputPulseSoft
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number.
	inPort	int	1 - 16	1	Output port description number.

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

Use this function with XSight_SetOutputPulseConfiguration.

XSight_SetOutputPulseTriggered

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets output pulse triggered by input port.

Syntax

```
void avl::XSight_SetOutputPulseTriggered
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort,
 int inDelay,
 int inPulseWidth,
 int inTriggerPort,
 avl::XSightTriggerEdge::Type inTriggerEdge
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number.
	inPort	int	1 - 16	1	IO port description number.
	inDelay	int	0 - ∞	0	Output delay in us unit.
	inPulseWidth	int	1 - ∞	100	Pulse width in us unit.
	inTriggerPort	int	1 - 16	1	Input port description number.
	inTriggerEdge	XSightTriggerEdge::Type			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_SetPWMFlashTime

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets PWM flash time.

Syntax

```
void avl::XSight_SetPWMFlashTime
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inChannel,
 atl::int64 inTime
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inChannel	int	1 - 6	1	PWM channel description number.
	inTime	int64	0 - ∞	10000L	Time in us unit.

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_SetPWMParameters

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets PWM channel parameters.

Syntax

```
void avl::XSight_SetPWMParameters
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inChannel,
 int inFrequency,
 int inDutyCycle
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inChannel	int	1 - 6	1	PWM channel description number.
	inFrequency	int	1 - 500000	10000	Output frequency as Hz.
	inDutyCycle	int	0 - 127	1	Duty cycle.

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_SetPWMStart

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets PWM channel on.

Syntax

```
void avl::XSight_SetPWMStart
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inChannel
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inChannel	int	1 - 6	1	PWM channel description number.

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_SetPWMStop

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets PWM channel off.

Syntax

```
void avl::XSight_SetPWMStop
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inChannel
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inChannel	int	1 - 6	1	PWM channel description number.

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_SetPWMTriggerMode

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets PWM channel working mode.

Syntax

```
void avl::XSight_SetPWMTriggerMode
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inChannel,
 avl::XSightPWMTrigger::Type inMode
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inChannel	int	1 - 6	1	PWM channel description number.
	inMode	XSightPWMTrigger::Type			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_SetPWMTriggerSource

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets PWM trigger source.

Syntax

```
void avl::XSight_SetPWMTriggerSource
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inChannel,
 int inPort
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inChannel	int	1 - 6	1	PWM channel description number.
	inPort	int			Input port description number.

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_WriteOutputLevel_Multiple

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Writes output port levels to device in one routine.

Syntax

```
void avl::XSight_WriteOutputLevel_Multiple
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 const atl::Array<avl::XSightIOLevel::Type>& inLevels
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inLevels	const Array<XSightIOLevel::Type>&			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

XSight_WriteOutputLevel_Single

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Writes output port levels to device.

Syntax

```
void avl::XSight_WriteOutputLevel_Single
(
 XSight_State& ioState,
 atl::Optional<int> inDeviceIndex,
 int inPort,
 avl::XSightIOLevel::Type inLevel
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	XSight_State&			Object used to maintain state of the function.
	inDeviceIndex	Optional<int>	0 - ∞	NIL	Device identifying number
	inPort	int	1 - 16	1	IO port description number.
	inLevel	XSightIOLevel::Type			

Remarks

Device identification

inDeviceIndex can be used to pick one of multiple devices connected to the computer. **inDeviceIndex** can be set to:

- **Device Index** - X-Sight I/O device identifying internal index.

I/O device driver software

This filter is intended to cooperate with a device using its vendor SDK. To be able to connect to a device, it is required to install X-Sight SDK.

Add DLL path to system environment variable is required.

Recommended Runtime version for Aurora Vision Studio usage is **1.0.0**.

Invoking I/O port filters who force specifics behaviour (for example, set output level) may also change I/O mode to fitted.

194. ZebraCameras

Table of content:

ZebraCameras_GenerateSoftwareTrigger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Generates a software trigger.

Syntax

```
void avl::ZebraCameras_GenerateSoftwareTrigger
(
 ZebraCameras_State& ioState,
 atl::Optional<const atl::String&> inDeviceID
)
```

Parameters

	Name	Type	Default	Description
↔	ioState	ZebraCameras_State&		Object used to maintain state of the function.
→	inDeviceID	Optional<const String&>	NIL	Device identifying number

ZebraCameras_GetBooleanParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets a boolean parameter of the ZebraCameras device.

Syntax

```
void avl::ZebraCameras_GetBooleanParameter
(
 ZebraCameras_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 bool& outValue
)
```

Parameters

	Name	Type	Default	Description
↔	ioState	ZebraCameras_State&		Object used to maintain state of the function.
→	inDeviceID	Optional<const String&>	NIL	Device identifying number
→	inName	const String&		Parameter name
←	outValue	bool&		Parameter value

ZebraCameras_GetEnumParameterAsInteger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets an enum parameter of the ZebraCameras device.

Syntax

```
void avl::ZebraCameras_GetEnumParameterAsInteger
(
 ZebraCameras_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 atl::int64& outValue
)
```

Parameters

	Name	Type	Default	Description
↔	ioState	ZebraCameras_State&		Object used to maintain state of the function.
→	inDeviceID	Optional<const String&>	NIL	Device identifying number
→	inName	const String&		Parameter name
←	outValue	int64&		Parameter value

ZebraCameras_GetEnumParameterAsString

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets an enum parameter of the ZebraCameras device.

Syntax

```
void avl::ZebraCameras_GetEnumParameterAsString
(
 ZebraCameras_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 atl::String& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	ZebraCameras_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inName	const String&		Parameter name
	outValue	String&		Parameter value

ZebraCameras_GetFloatParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets a float parameter of the ZebraCameras device.

Syntax

```
void avl::ZebraCameras_GetFloatParameter
(
 ZebraCameras_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 double& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	ZebraCameras_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inName	const String&		Parameter name
	outValue	double&		Parameter value

ZebraCameras_GetIntegerParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets an integer parameter of the ZebraCameras device.

Syntax

```
void avl::ZebraCameras_GetIntegerParameter
(
 ZebraCameras_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 atl::int64& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	ZebraCameras_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inName	const String&		Parameter name
	outValue	int64&		Parameter value

ZebraCameras_GetStringParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Gets a string parameter of the ZebraCameras device.

Syntax

```
void avl::ZebraCameras_GetStringParameter
(
 ZebraCameras_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 atl::String& outValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	ZebraCameras_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inName	const String&		Parameter name
	outValue	String&		Parameter value

ZebraCameras_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an image using a ZebraCameras device.

Syntax

```
bool avl::ZebraCameras_GrabImage
(
 ZebraCameras_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inInputQueueSize,
 atl::Optional<avl::EBUSPixelFormat::Type> inPixelFormat,
 atl::Optional<const avl::Box&> inAoi,
 atl::Optional<avl::EBUSAcquisitionMode::Type> inAcquisitionMode,
 atl::Optional<int> inAcquisitionFrameCount,
 atl::Optional<double> inFrameRate,
 atl::Optional<avl::EBUSAutoExposureMode::Type> inExposureAuto,
 atl::Optional<double> inExposureTime,
 atl::Optional<avl::EBUSAutoGainMode::Type> inGainAuto,
 atl::Optional<double> inGain,
 atl::Optional<avl::EBUSTriggerSource::Type> inTriggerSource,
 atl::Optional<avl::EBUSTriggerActivation::Type> inTriggerActivation,
 bool inSkipInvalidFrames,
 avl::Image& outImage,
 avl::EBUSFrameData& outFrameData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	ZebraCameras_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Device identifying number
	inInputQueueSize	int	1 - ∞	12	Capacity of output frames queue
	inPixelFormat	Optional<EBUSPixelFormat::Type>		NIL	Image pixel format
	inAoi	Optional<const Box&>		NIL	Area of interest
	inAcquisitionMode	Optional<EBUSAcquisitionMode::Type>		NIL	Acquisition mode
	inAcquisitionFrameCount	Optional<int>	1 - 65535	NIL	Number of frames to acquire in MultiFrame acquisition mode
	inFrameRate	Optional<double>	0.125 - ∞	NIL	Acquisition frame rate
	inExposureAuto	Optional<EBUSAutoExposureMode::Type>		NIL	Automatic exposure mode
	inExposureTime	Optional<double>	1 - ∞	NIL	Exposure time in us
	inGainAuto	Optional<EBUSAutoGainMode::Type>		NIL	Automatic gain mode
	inGain	Optional<double>	1 - ∞	NIL	Gain as an absolute physical value
	inTriggerSource	Optional<EBUSTriggerSource::Type>		NIL	Trigger source
	inTriggerActivation	Optional<EBUSTriggerActivation::Type>		NIL	Trigger activation mode
	inSkipInvalidFrames	bool		True	Skipping invalid images
	outImage	Image&			Captured frame
	outFrameData	EBUSFrameData&			Captured frame data

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install proper SDK with camera dedicated drivers.

Please contact our technical support to obtain the SDK.

Add DLL path to system environment variable may be required.

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, the first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to the computer. DeviceID can be set to:

- **Serial Number** - should be printed on the device housing.
- **Device ID** - unique device identifier.
- **IP Address** - network IP address of device.
- **MAC Address** - MAC address of device.

See Also

- ZebraCameras_GrabImage_WithTimeout – Captures an image using a ZebraCameras device with timeout.
- ZebraCameras_StartAcquisition – Initializes and starts image acquisition in a camera.

ZebraCameras_GrabImage_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures an image using a ZebraCameras device with timeout.

Syntax

```
bool avl::ZebraCameras_GrabImage_WithTimeout
(
 ZebraCameras_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inTimeout,
 int inInputQueueSize,
 atl::Optional<avl::EBUSPixelFormat::Type> inPixelFormat,
 atl::Optional<const avl::Box&> inAoi,
 atl::Optional<avl::EBUSAcquisitionMode::Type> inAcquisitionMode,
 atl::Optional<int> inAcquisitionFrameCount,
 atl::Optional<double> inFrameRate,
 atl::Optional<avl::EBUSAutoExposureMode::Type> inExposureAuto,
 atl::Optional<double> inExposureTime,
 atl::Optional<avl::EBUSAutoGainMode::Type> inGainAuto,
 atl::Optional<double> inGain,
 atl::Optional<avl::EBUSTriggerSource::Type> inTriggerSource,
 atl::Optional<avl::EBUSTriggerActivation::Type> inTriggerActivation,
 bool inSkipInvalidFrames,
 atl::Conditional<avl::Image>& outImage,
 atl::Conditional<avl::EBUSFrameData>& outFrameData
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	ZebraCameras_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Device identifying number
	inTimeout	int	100 - ∞	100	Maximum time to wait for frame in milliseconds
	inInputQueueSize	int	1 - ∞	12	Capacity of output frames queue
	inPixelFormat	Optional<EBUSPixelFormat::Type>		NIL	Image pixel format
	inAoi	Optional<const Box&>		NIL	Area of interest
	inAcquisitionMode	Optional<EBUSAcquisitionMode::Type>		NIL	Acquisition mode
	inAcquisitionFrameCount	Optional<int>	1 - 65535	NIL	Number of frames to acquire in MultiFrame acquisition mode
	inFrameRate	Optional<double>	0.125 - ∞	NIL	Acquisition frame rate
	inExposureAuto	Optional<EBUSAutoExposureMode::Type>		NIL	Automatic exposure mode
	inExposureTime	Optional<double>	1 - ∞	NIL	Exposure time in us
	inGainAuto	Optional<EBUSAutoGainMode::Type>		NIL	Automatic gain mode
	inGain	Optional<double>	1 - ∞	NIL	Gain as an absolute physical value
	inTriggerSource	Optional<EBUSTriggerSource::Type>		NIL	Trigger source
	inTriggerActivation	Optional<EBUSTriggerActivation::Type>		NIL	Trigger activation mode
	inSkipInvalidFrames	bool		True	Skipping invalid images
	outImage	Conditional<Image>&			Captured frame
	outFrameData	Conditional<EBUSFrameData>&			Captured frame data

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install proper SDK with camera dedicated drivers.

Please contact our technical support to obtain the SDK.

Add DLL path to system environment variable may be required.

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, the first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to the computer. DeviceID can be set to:

- **Serial Number** - should be printed on the device housing.
- **Device ID** - unique device identifier.
- **IP Address** - network IP address of device.
- **MAC Address** - MAC address of device.

See Also

- ZebraCameras_GrabImage – Captures an image using a ZebraCameras device.
- ZebraCameras_StartAcquisition – Initializes and starts image acquisition in a camera.

ZebraCameras_SetBooleanParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a boolean parameter of the ZebraCameras device.

Syntax

```
void avl::ZebraCameras_SetBooleanParameter
(
 ZebraCameras_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 bool inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	ZebraCameras_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inName	const String&		Parameter name
	inValue	bool		Parameter value

ZebraCameras_SetEnumParameterAsInteger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets an enum parameter of the ZebraCameras device.

Syntax

```
void avl::ZebraCameras_SetEnumParameterAsInteger
(
 ZebraCameras_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 atl::int64 inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	ZebraCameras_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inName	const String&		Parameter name
	inValue	int64		Parameter value

ZebraCameras_SetEnumParameterAsString

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets an enum parameter of the ZebraCameras device.

Syntax

```
void avl::ZebraCameras_SetEnumParameterAsString
(
 ZebraCameras_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 const atl::String& inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	ZebraCameras_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inName	const String&		Parameter name
	inValue	const String&		Parameter value

ZebraCameras_SetFloatParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a float parameter of the ZebraCameras device.

Syntax

```
void avl::ZebraCameras_SetFloatParameter
(
 ZebraCameras_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 double inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	ZebraCameras_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inName	const String&		Parameter name
	inValue	double		Parameter value

ZebraCameras_SetIntegerParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets an integer parameter of the ZebraCameras device.

Syntax

```
void avl::ZebraCameras_SetIntegerParameter
(
 ZebraCameras_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 atl::int64 inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	ZebraCameras_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inName	const String&		Parameter name
	inValue	int64		Parameter value

ZebraCameras_SetStringParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Sets a string parameter of the ZebraCameras device.

Syntax

```
void avl::ZebraCameras_SetStringParameter
(
 ZebraCameras_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 const atl::String& inName,
 const atl::String& inValue
)
```

Parameters

	Name	Type	Default	Description
	ioState	ZebraCameras_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number
	inName	const String&		Parameter name
	inValue	const String&		Parameter value

ZebraCameras_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Initializes and starts image acquisition in a camera.

Syntax

```
void avl::ZebraCameras_StartAcquisition
(
 ZebraCameras_State& ioState,
 atl::Optional<const atl::String&> inDeviceID,
 int inInputQueueSize,
 atl::Optional<avl::EBUSPixelFormat::Type> inPixelFormat,
 atl::Optional<const avl::Box&> inAoi,
 atl::Optional<avl::EBUSAcquisitionMode::Type> inAcquisitionMode,
 atl::Optional<int> inAcquisitionFrameCount,
 atl::Optional<double> inFrameRate,
 atl::Optional<avl::EBUSAutoExposureMode::Type> inExposureAuto,
 atl::Optional<double> inExposureTime,
 atl::Optional<avl::EBUSAutoGainMode::Type> inGainAuto,
 atl::Optional<double> inGain,
 atl::Optional<avl::EBUSTriggerSource::Type> inTriggerSource,
 atl::Optional<avl::EBUSTriggerActivation::Type> inTriggerActivation
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	ZebraCameras_State&			Object used to maintain state of the function.
	inDeviceID	Optional<const String&>		NIL	Device identifying number
	inInputQueueSize	int	1 - ∞	12	Capacity of output frames queue
	inPixelFormat	Optional<EBUSPixelFormat::Type>		NIL	Image pixel format
	inAoi	Optional<const Box&>		NIL	Area of interest
	inAcquisitionMode	Optional<EBUSAcquisitionMode::Type>		NIL	Acquisition mode
	inAcquisitionFrameCount	Optional<int>	1 - 65535	NIL	Number of frames to acquire in MultiFrame acquisition mode
	inFrameRate	Optional<double>	0.125 - ∞	NIL	Acquisition frame rate
	inExposureAuto	Optional<EBUSAutoExposureMode::Type>		NIL	Automatic exposure mode
	inExposureTime	Optional<double>	1 - ∞	NIL	Exposure time in us
	inGainAuto	Optional<EBUSAutoGainMode::Type>		NIL	Automatic gain mode
	inGain	Optional<double>	1 - ∞	NIL	Gain as an absolute physical value
	inTriggerSource	Optional<EBUSTriggerSource::Type>		NIL	Trigger source
	inTriggerActivation	Optional<EBUSTriggerActivation::Type>		NIL	Trigger activation mode

Remarks

Camera driver software

This filter is intended to cooperate with a camera using its vendor SDK. To be able to connect to a camera, it is required to install proper SDK with camera dedicated drivers.

Please contact our technical support to obtain the SDK.

Add DLL path to system environment variable may be required.

Camera identification

When there is only one camera connected to a computer, field **inDeviceID** can be set to Auto. In this case, the first available camera will be found and connected.

inDeviceID field can be used to pick one of multiple cameras connected to the computer. DeviceID can be set to:

- **Serial Number** - should be printed on the device housing.
- **Device ID** - unique device identifier.
- **IP Address** - network IP address of device.
- **MAC Address** - MAC address of device.

See Also

- ZebraCameras_GrabImage – Captures an image using a ZebraCameras device.
- ZebraCameras_GrabImage_WithTimeout – Captures an image using a ZebraCameras device with timeout.

ZebraCameras_StopAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Stops image acquisition in a camera.

Syntax

```
void avl::ZebraCameras_StopAcquisition
(
 ZebraCameras_State& ioState,
 atl::Optional<const atl::String&> inDeviceID
)
```

Parameters

	Name	Type	Default	Description
	ioState	ZebraCameras_State&		Object used to maintain state of the function.
	inDeviceID	Optional<const String&>	NIL	Device identifying number

195. ZebraScanEngines

Table of content:

ZebraScanEngines_BeepBuzzer

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Beeps the in-built buzzer according to the beep code.

Syntax

```
void avl::ZebraScanEngines_BeepBuzzer
(
 ZebraScanEngines_State& ioState,
 atl::Optional<int> inDeviceID,
 avl::ZebraScanEnginesBeepCodes::Type inBeepCode
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	ZebraScanEngines_State&			Object used to maintain state of the function.
	inDeviceID	Optional<int>	1 - ∞	NIL	Device identifying number
	inBeepCode	ZebraScanEnginesBeepCodes::Type			Buzzer beep pattern

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

ZebraScanEngines_ControlAim

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Turns device aim on or off.

Syntax

```
void avl::ZebraScanEngines_ControlAim
(
 ZebraScanEngines_State& ioState,
 atl::Optional<int> inDeviceID,
 bool inTurnOn
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	ZebraScanEngines_State&			Object used to maintain state of the function.
	inDeviceID	Optional<int>	1 - ∞	NIL	Device identifying number
	inTurnOn	bool			Turn aim on or off.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

ZebraScanEngines_ControlLeds

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Turns device LEDs on and off.

Syntax

```
void avl::ZebraScanEngines_ControlLeds
(
 ZebraScanEngines_State& ioState,
 atl::Optional<int> inDeviceID,
 avl::ZebraScanEnginesLeds::Type inLedAction
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	ZebraScanEngines_State&			Object used to maintain state of the function.
	inDeviceID	Optional<int>	1 - ∞	NIL	Device identifying number
	inLedAction	ZebraScanEnginesLeds::Type			LED action to perform

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

ZebraScanEngines_GenerateSoftwareTrigger

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Generates a software trigger.

Syntax

```
void avl::ZebraScanEngines_GenerateSoftwareTrigger
(
 ZebraScanEngines_State& ioState,
 atl::Optional<int> inDeviceID,
 bool inRelease
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	ZebraScanEngines_State&			Object used to maintain state of the function.
	inDeviceID	Optional<int>	1 - ∞	NIL	Device identifying number
	inRelease	bool			Whether to pull or to release the trigger

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

ZebraScanEngines_GetParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Get parameter of a specified scanner.

Syntax

```
void avl::ZebraScanEngines_GetParameter
(
 ZebraScanEngines_State& ioState,
 atl::Optional<int> inDeviceID,
 int inParameterID,
 atl::String& outParameterType,
 atl::String& outParameterValue
)
```

Parameters

	Name	Type	Range	Default	Description
↻	ioState	ZebraScanEngines_State&			Object used to maintain state of the function.
→	inDeviceID	Optional<int>	1 - ∞	NIL	Device identifying number
→	inParameterID	int	0 - ∞		Attribute number
←	outParameterType	String&			Attribute type
←	outParameterValue	String&			Attribute value

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

ZebraScanEngines_GrabImage

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a frame using a Zebra scanner.

Syntax

```
bool avl::ZebraScanEngines_GrabImage
(
 ZebraScanEngines_State& ioState,
 atl::Optional<int> inDeviceID,
 int inInputQueueSize,
 bool inAutoTrigger,
 atl::Optional<int> inAutoTriggerPeriod,
 avl::ZebraScanEnginesOperationModes::Type inOperationMode,
 avl::ZebraScanEnginesImageTypes::Type inImageType,
 avl::Image& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	ZebraScanEngines_State&			Object used to maintain state of the function.
	inDeviceID	Optional<int>	1 - ∞	NIL	Device identifying number
	inInputQueueSize	int	1 - 200	4	Capacity of output frames queue
	inAutoTrigger	bool		True	Enables or disables auto triggering
	inAutoTriggerPeriod	Optional<int>	10 - ∞	NIL	Period of automatic triggers (in ms)
	inOperationMode	ZebraScanEnginesOperationModes::Type		VideoMode	Scanner operation mode (video/snapshot)
	inImageType	ZebraScanEnginesImageTypes::Type		BmpFile	Image format for the snapshot mode
	outImage	Image&			Captured frame

Remarks

This filter is intended to cooperate with a scanner using its vendor SDK. To be able to connect to a scanner, it is required to install Zebra Scanner SDK software with Zebra CoreScanner Driver.

Zebra Scanner SDK can be downloaded from the following website: https://www.zebra.com/us/en/support-downloads/software/developer-tools/scanner-sdk-for-windows.html (registration may be required).

Add DLL path to system environment variable may be required.

Recommended Zebra Scanner SDK version for Aurora Vision Studio usage is **3.06.0024**.

Scanner identification

When there is only one scanner connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available scanner will be found and connected.

inDeviceID field can be used to pick one of multiple scanners connected to the computer. DeviceID can be set to:

- **ID number** - can be obtained from Scanner SDK Sample Application which is installed during the Zebra Scanner SDK installation. The ID is the number in the column **"#"** under "Connected Scanners".

Automatic trigger

inAutoTriggerPeriod field can be used to specify the period of the automatic trigger when **inAutoTrigger** is set to true. By default, when this field is set to Auto, the triggers are generated with a 1-second period. Be careful when choosing values for the period, as values that are too small can destabilize the device and not perform as expected.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- ZebraScanEngines_GrabImage_WithTimeout – Captures a frame using a Zebra scanner.
- ZebraScanEngines_StartAcquisition – Initializes and starts image acquisition in a scanner.

ZebraScanEngines_GrabImage_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Captures a frame using a Zebra scanner.

Syntax

```
bool avl::ZebraScanEngines_GrabImage_WithTimeout
(
 ZebraScanEngines_State& ioState,
 int inTimeout,
 atl::Optional<int> inDeviceID,
 int inInputQueueSize,
 bool inAutoTrigger,
 atl::Optional<int> inAutoTriggerPeriod,
 avl::ZebraScanEnginesOperationModes::Type inOperationMode,
 avl::ZebraScanEnginesImageTypes::Type inImageType,
 atl::Conditional<avl::Image>& outImage
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	ZebraScanEngines_State&			Object used to maintain state of the function.
	inTimeout	int	1 - ∞	100	Maximum time to wait for a frame in milliseconds
	inDeviceID	Optional<int>	1 - ∞	NIL	Device identifying number
	inInputQueueSize	int	1 - 200	4	Capacity of output frames queue
	inAutoTrigger	bool		True	Enables or disables auto triggering
	inAutoTriggerPeriod	Optional<int>	10 - ∞	NIL	Period of automatic triggers (in ms)
	inOperationMode	ZebraScanEnginesOperationModes::Type		VideoMode	Scanner operation mode (video/snapshot)
	inImageType	ZebraScanEnginesImageTypes::Type		BmpFile	Image format for the snapshot mode
	outImage	Conditional<Image>&			Captured frame

Remarks

This filter is intended to cooperate with a scanner using its vendor SDK. To be able to connect to a scanner, it is required to install Zebra Scanner SDK software with Zebra CoreScanner Driver.

Zebra Scanner SDK can be downloaded from the following website: https://www.zebra.com/us/en/support-downloads/software/developer-tools/scanner-sdk-for-windows.html (registration may be required).

Add DLL path to system environment variable may be required.

Recommended Zebra Scanner SDK version for Aurora Vision Studio usage is **3.06.0024**.

Scanner identification

When there is only one scanner connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available scanner will be found and connected.

inDeviceID field can be used to pick one of multiple scanners connected to the computer. DeviceID can be set to:

- **ID number** - can be obtained from Scanner SDK Sample Application which is installed during the Zebra Scanner SDK installation. The ID is the number in the column **"#"** under "Connected Scanners".

Automatic trigger

inAutoTriggerPeriod field can be used to specify the period of the automatic trigger when **inAutoTrigger** is set to true. By default, when this field is set to Auto, the triggers are generated with a 1-second period. Be careful when choosing values for the period, as values that are too small can destabilize the device and not perform as expected.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- ZebraScanEngines_GrabImage – Captures a frame using a Zebra scanner.
- ZebraScanEngines_StartAcquisition – Initializes and starts image acquisition in a scanner.

ZebraScanEngines_ListScanners

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Lists available Zebra scanners.

Syntax

```
void avl::ZebraScanEngines_ListScanners
(
 ZebraScanEngines_State& ioState,
 atl::Array<avl::ZebraScanEnginesDeviceInfo>& outScannerList
)
```

Parameters

	Name	Type	Default	Description
↔	ioState	ZebraScanEngines_State&		Object used to maintain state of the function.
←	outScannerList	Array<ZebraScanEnginesDeviceInfo>&		List of connected scanners

Remarks

During frequent requests for a list of devices, it is possible to break the connection with the polling device. It is not recommended to use this filter in a production environment. It is for debugging only.

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- ZebraScanEngines_GrabImage – Captures a frame using a Zebra scanner.
- ZebraScanEngines_GrabImage_WithTimeout – Captures a frame using a Zebra scanner.
- ZebraScanEngines_StopAcquisition – Stops image acquisition in a scanner.

ZebraScanEngines_ScanBarcode

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Scans a barcode using a Zebra scanner.

Syntax

```
bool avl::ZebraScanEngines_ScanBarcode
(
 ZebraScanEngines_State& ioState,
 atl::Optional<int> inDeviceID,
 avl::ZebraScanEnginesBarcodeDataTypes::Type& outBarcodeType,
 atl::String& outBarcodeDecoded,
 avl::ByteBuffer& outRawData
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	ZebraScanEngines_State&			Object used to maintain state of the function.
→	inDeviceID	Optional<int>	1 - ∞	NIL	Device identifying number
←	outBarcodeType	ZebraScanEnginesBarcodeDataTypes::Type&			Barcode type
←	outBarcodeDecoded	String&			Barcode data
←	outRawData	ByteBuffer&			Raw barcode data

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

ZebraScanEngines_ScanBarcode_WithTimeout

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Scans a barcode using a Zebra scanner.

Syntax

```
bool avl::ZebraScanEngines_ScanBarcode_WithTimeout
(
 ZebraScanEngines_State& ioState,
 int inTimeout,
 atl::Optional<int> inDeviceID,
 atl::Conditional<avl::ZebraScanEnginesBarcodeDataTypes::Type>& outBarcodeType,
 atl::Conditional<atl::String>& outBarcodeDecoded,
 atl::Conditional<avl::ByteBuffer>& outRawData
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	ZebraScanEngines_State&			Object used to maintain state of the function.
→	inTimeout	int	1 - ∞	100	Maximum time to wait for a barcode in milliseconds
→	inDeviceID	Optional<int>	1 - ∞	NIL	Device identifying number
←	outBarcodeType	Conditional<ZebraScanEnginesBarcodeDataTypes::Type>&			Barcode type
←	outBarcodeDecoded	Conditional<String>&			Barcode data
←	outRawData	Conditional<ByteBuffer>&			Raw barcode data

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

ZebraScanEngines_SetParameter

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Set parameter of a specified scanner temporarily or persistently.

Syntax

```
void avl::ZebraScanEngines_SetParameter
(
 ZebraScanEngines_State& ioState,
 atl::Optional<int> inDeviceID,
 int inParameterID,
 const atl::String& inParameterType,
 atl::Optional<atl::String>& inParameterValue,
 bool inIsPersistent
)
```

Parameters

	Name	Type	Range	Default	Description
↔	ioState	ZebraScanEngines_State&			Object used to maintain state of the function.
→	inDeviceID	Optional<int>	1 - ∞	NIL	Device identifying number
→	inParameterID	int	0 - ∞		Attribute number
→	inParameterType	const String&			Attribute type
→	inParameterValue	Optional<String>&		NIL	Attribute value
→	inIsPersistent	bool		False	Whether to make the change persistent

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

ZebraScanEngines_StartAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Initializes and starts image acquisition in a scanner.

Syntax

```
void avl::ZebraScanEngines_StartAcquisition
(
 ZebraScanEngines_State& ioState,
 atl::Optional<int> inDeviceID,
 int inInputQueueSize,
 bool inAutoTrigger,
 atl::Optional<int> inAutoTriggerPeriod,
 avl::ZebraScanEnginesOperationModes::Type inOperationMode,
 avl::ZebraScanEnginesImageTypes::Type inImageType
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	ZebraScanEngines_State&			Object used to maintain state of the function.
	inDeviceID	Optional<int>		NIL	Device identifying number
	inInputQueueSize	int	1 - 200	4	Capacity of output frames queue
	inAutoTrigger	bool		True	Enables or disables auto triggering
	inAutoTriggerPeriod	Optional<int>	10 - ∞	NIL	Period of automatic triggers (in ms)
	inOperationMode	ZebraScanEnginesOperationModes::Type		VideoMode	Scanner operation mode (video/snapshot)
	inImageType	ZebraScanEnginesImageTypes::Type		BmpFile	Image format for the snapshot mode

Remarks

This filter is intended for establishing connection with a Zebra scanner device using Zebra Scanner SDK interface, to initialize image streaming. It is only needed when explicit image acquisition start is required in the initial phase of a program.

The use of this filter is not obligatory. ZebraScanEngines_GrabImage or ZebraScanEngines_GrabImage_WithTimeout] filters will initialize and start image acquisition upon their first invoke.

This filter has no effect when invoked for the second time and when invoked after image grabbing filters.

This filter is intended to cooperate with a scanner using its vendor SDK. To be able to connect to a scanner, it is required to install Zebra Scanner SDK software with Zebra CoreScanner Driver.

Zebra Scanner SDK can be downloaded from the following website: https://www.zebra.com/us/en/support-downloads/software/developer-tools/scanner-sdk-for-windows.html (registration may be required).

Add DLL path to system environment variable may be required.

Recommended Zebra Scanner SDK version for Aurora Vision Studio usage is **3.06.0024**.

Scanner identification

When there is only one scanner connected to a computer, field **inDeviceID** can be set to Auto. In this case, first available scanner will be found and connected.

inDeviceID field can be used to pick one of multiple scanners connected to the computer. DeviceID can be set to:

- **ID number** - can be obtained from Scanner SDK Sample Application which is installed during the Zebra Scanner SDK installation. The ID is the number in the column **"#"** under "Connected Scanners".

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

See Also

- ZebraScanEngines_GrabImage – Captures a frame using a Zebra scanner.
- ZebraScanEngines_GrabImage_WithTimeout – Captures a frame using a Zebra scanner.

ZebraScanEngines_StopAcquisition

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Stops image acquisition in a scanner.

Syntax

```
void avl::ZebraScanEngines_StopAcquisition
(
 ZebraScanEngines_State& ioState,
 atl::Optional<int> inDeviceID
)
```

Parameters

	Name	Type	Default	Description
	ioState	ZebraScanEngines_State&		Object used to maintain state of the function.
	inDeviceID	Optional<int>	NIL	Device identifying number

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

ZebraScanEngines_SwitchHostMode

Also in AVL Lite

Header: ThirdPartySdk.h
Namespace: avl
Module: ThirdParty

Switch host mode of a specified scanner.

Syntax

```
void avl::ZebraScanEngines_SwitchHostMode
(
 ZebraScanEngines_State& ioState,
 atl::Optional<int> inDeviceID,
 avl::ZebraScanEnginesHostModes::Type inHostMode,
 bool inSilentSwitch,
 bool inIsPermanentChange
)
```

Parameters

	Name	Type	Range	Default	Description
	ioState	ZebraScanEngines_State&			Object used to maintain state of the function.
	inDeviceID	Optional<int>	1 - ∞	NIL	Device identifying number
	inHostMode	ZebraScanEnginesHostModes::Type		USB_SNAPI_with_Imaging	Scanner USB host mode
	inSilentSwitch	bool			Whether to suppress the typical device reboot beeps
	inIsPermanentChange	bool			Whether to keep the targeted host mode as the permanent host mode of the device

Multithreaded environment

This function is not guaranteed to be thread-safe. When used in multithreaded environment, it has to be manually synchronized.

196. GigE Vision

Table of content:

GigEVision_CloseHandle

Header: Genicam.h
Namespace: avl
Module: Genicam

Closes a handle opened in GigE Vision subsystem.

Syntax

```
void avl::GigEVision_CloseHandle
(
 GigEHandle inHandle
)
```

Parameters

	Name	Type	Default	Description
➡	inHandle	GigEHandle		

GigEVision_FindDevices

Header: Genicam.h
Namespace: avl
Module: Genicam

This function performs enumeration of network devices and returns a list of present device descriptors.

Syntax

```
void avl::GigEVision_FindDevices
(
 int inTime,
 int inMaxDevices,
 atl::Array< GigEVision_DeviceDescriptor >& outDevices
)
```

Parameters

	Name	Type	Default	Description
➡	inTime	int		Time, in milliseconds, that the function will wait for device response. This is the time that the function will block execution before return.
➡	inMaxDevices	int	0	Maximum number of devices to enumerate or 0 to return all devices.
⬅	outDevices	Array< GigEVision_DeviceDescriptor >&		Returns a list with device descriptors. Array passed as buffer is cleared before search and appended with descriptors.

Description

This function broadcasts a single discovery request in the local network and waits for a specified amount of time for devices' responses. All responses are converted into device descriptors and appended into output array.

Devices that cause errors related to their single response are ignored in the output list, only network related errors corresponding to the discovery broadcasts cause the function to fail with exception.

A single device is identified by GigEVision_DeviceDescriptor structure.

Exceptions

This function will throw an exception in the following situation:

- Unexpected network related error.

GigEVision_FlushInputQueue

Header: Genicam.h
Namespace: avl
Module: Genicam

Discards remaining queued image frames and video stream data from network buffers.

Syntax

```
void avl::GigEVision_FlushInputQueue
(
 GigEHandle inDeviceHandle
)
```

Parameters

	Name	Type	Default	Description
➡	inDeviceHandle	GigEHandle		Handle of an opened device which streaming queue should be cleared.

Description

Flushing input queue means, that all frames that have been received from device and that have not been retrieved yet by the application, will be discarded and lost (including a partially filled frame and buffered network packets of next frames).

Exceptions

This function will throw an exception in the following situations:

- Device handle is invalid.

GigEVision_GetAcquisitionActive

Header: Genicam.h
Namespace: avl
Module: Genicam

Checks if specified device is currently streaming video.

Syntax

```
bool avl::GigEVision_GetAcquisitionActive
(
 GigEHandle inDeviceHandle
)
```

Parameters

	Name	Type	Default	Description
➡	inDeviceHandle	GigEHandle		Handle of an opened device that will be checked for streaming.

Description

Return Value

`true`, when device connection is active and device is streaming video, `false` otherwise.

Exceptions

This function will throw an exception in the following situation:

- Device handle is invalid.

GigEVision_GetDeviceConnectionOpened

Header: Genicam.h
Namespace: avl
Module: Genicam

Checks if connection with specified device is still active.

Syntax

```
bool avl::GigEVision_GetDeviceConnectionOpened
(
 GigEHandle inDeviceHandle
)
```

Parameters

	Name	Type	Default	Description
→	inDeviceHandle	GigEHandle		Handle of an opened device that will be checked.

Description

This function will return false after connection with device has been lost.

This function is indicative only. There is no guarantee that subsequent operations will not fail even if this function returns true. Broken connection can be detected after significant delay or during execution of other operations.

Exceptions

This function will throw an exception in the following situation:

- Device handle is invalid.

GigEVision_GetPixelFormats

Header: Genicam.h
Namespace: avl
Module: Genicam

Retrieves a list of available pixel formats supported by opened device.

Syntax

```
void avl::GigEVision_GetPixelFormats
(
 GigEHandle inDeviceHandle,
 atl::Array< atl::String >& outFormats
)
```

Parameters

	Name	Type	Default	Description
→	inDeviceHandle	GigEHandle		Handle of an opened device.
←	outFormats	Array< String >&		Array that will return a list of format strings.

Description

This function will access GenICam "PixelFormat" standard enumeration parameter and read its entries. Only entries marked as implemented and available will be present in the returned list. The list will contain names of PixelFormat enumeration entries (not display names).

One of the format names returned by this function can be passed to GigEVision_StartAcquisition function.

Exceptions

This function will throw an exception in the following situations:

- Device handle is invalid.
- Connection with device is lost.
- Device is not a video transmitter.
- Other unexpected GenICam or connection error occurred.

GigEVision_GetStreamingStatistics

Header: Genicam.h
Namespace: avl
Module: Genicam

Retrieves basic statistics of video streaming out of a GigEVision device.

Syntax

```
void avl::GigEVision_GetStreamingStatistics
(
 GigEHandle inDeviceHandle,
 bool inReset,
 GigEVision_StreamingStatistics& outStatistics
)
```

Parameters

	Name	Type	Default	Description
→	inDeviceHandle	GigEHandle		Handle of an opened device that is streaming video.
→	inReset	bool		True to reset internal device statistics after read.
←	outStatistics	GigEVision_StreamingStatistics&		

Description

This function is intended to help with diagnosing video streaming and network transfer issues, and it allows to read basic streaming statistic counters from the acquisition engine.

Exceptions

This function will throw an exception in the following situations:

- Device handle is invalid.

GigEVision_OpenDevice

Header: Genicam.h
Namespace: avl
Module: Genicam

Opens a handle to a GigE Vision device.

Syntax

```
GigEHandle avl::GigEVision_OpenDevice
(
 const atl::String& inDeviceAddress
)
```

Parameters

	Name	Type	Default	Description
➡	inDeviceAddress	const String&		Textual address that identifies a device in the network.

Description

When specified device is already opened upon function call, a new handle to the same device connection is returned.

When the device is opened for the first time, the library will attempt to establish a network connection, set up its state, load and interpret GenICam description.

For every call to GigEVision_OpenDevice a GigEVision_CloseHandle function must be called on returned handle.

When the device is opened, a new thread is started in the process. It controls and sustains the connection with the device and will be responsible for receiving stream packets.

Device address can be specified in one of following forms:

- **IP Address** - standard IPv4 address, for example "192.168.0.100". In this option the library will immediately attempt to establish a connection with specified address.
- **MAC Address** - 6 hexadecimals numbers representing the device network interface hardware address, for example "11:22:33:aa:bb:cc". In this option the library will first attempt to find the specified device in the network to retrieve its current IP address. This option can be used when the device is configured for automatic IP setup.
- **Serial number** - identified by "SN:" prefix, for example "SN:123456789". This option is similar to a MAC Address but uses a serial number to find the device in the network. Please note that not all GigE Vision devices support serial number identification.

Return Value

Handle to an open device. This handle can be used with other GigE Vision functions to control device state and image streaming, as well as with GenApi functions to access device GenApi parameters set.

Exceptions

This function will throw an exception in the following situations:

- Device address string has an invalid format.
- Device with specified address cannot be found in the network.
- Device connection operation failed.
- Device initial setup failed.
- GenICam device description xml file could not be loaded.
- Other GenICam or network related error occurred.

GigEVision_OpenSystemConfiguration

Header: Genicam.h
Namespace: avl
Module: Genicam

Opens configuration node set of GigE Vision application transport layer.

Syntax

```
GigEHandle avl::GigEVision_OpenSystemConfiguration
(
)
```

Description

This function allows to access library global configuration parameters (connection configuration on application side) for GigE Vision subsystem. The function returns a handle to parameters' set, that can be accessed with GenApi functions. Every handle returned by this function must be closed with GigEVision_CloseHandle function.

Configuration changes take effects only for devices opened afterwards (so configuration of currently opened and active devices is not changed).

Configuration set contains the following parameters:

Name (id)	Type	Description

GevAppTLPacketResendsCount	Integer	Maximum number of command resends to device on control channel, when command acknowledgment is not received. After specified number of attempts current operation is terminated with "no response" error. Range: 1...10 Default: 3
GevAppTLAckTimeout	Integer	Time in milliseconds that the application will wait for command acknowledgment from the device when communicating on control channel. After timeout elapses with no acknowledgment the application will assume that command message was lost. Range: 100...2000 Default: 300
GevAppTLHertbeatTimeout	Integer	Value in milliseconds of maximum allowed time without communication on control channel. After this time the application will send a so called heartbeat signal to the device. When the device is left without a heartbeat signal or the application did not receive a heartbeat acknowledgment, connection is assumed to be lost. Range: 500...20000 Default: 1000
GevAppTLDefaultFrameWaitTimeout	Integer	Value in milliseconds of wait timeout for GigEVision_ReceiveImage function. After this time when no new frame is available function will control connection state and raise an appropriate exception. Range: 500...30000 Default: 4000
GevAppTLConnectionOpeningAttemptsCount	Integer	Number of connection attempts for opening device. When connection related errors occur during device finding or device connection opening, library will not raise an exception but will attempt to open the connection again. This is the maximum allowed number of such attempts. Range: 1...30 Default: 3
GevAppTLDisablePacketSizeNegotiation	Boolean	When set to false, allows the library to automatically negotiate packet size with a device. Video streaming performance is higher when packet size is bigger, that is why the library will try to find the maximum allowed size for the current network configuration, but this negotiation take some time upon connection opening. When network configuration is complicated, maximum allowed packet size can change dynamically which cannot be noticed by the application. In such situation packet negotiation should be disabled. Default: False
GevAppTLEnableConstantPacketSize	Boolean	When packet size negotiation is disabled, the packet size may be left unchanged in device settings or set by library to a constant value on every connection. Set this parameter to False to left packet size unchanged, or to True to use packet size from GevAppTLConstantPacketSize parameter. Requires: GevAppTLDisablePacketSizeNegotiation=True Default: False
GevAppTLConstantPacketSize	Integer	When packet size negotiation is disabled and constant packet size is enabled this parameter specifies network packet size in bytes to use on video stream connection. Larger packet sizes are better for performance but are not always supported by network infrastructure. Requires: GevAppTLEnableConstantPacketSize=True Range: 100...100000
GevAppTLSocketBufferSize	Integer	Specifies the stream channel network socket input buffer size in kilobytes. Range: 2048kB...262144kB (2MB...256MB)
GevAppTLConnectInExclusiveMode	Boolean	Determines privilege mode of connection with device. When set to True enables Exclusive mode connection. In Exclusive mode only one application can be connected with device and only one application can read device state. When set to False disables exclusive mode and connects in Control mode. In Control mode other applications are permitted to monitor state of device. Default: True
GevAppTLUseMessageChannel	Boolean	When set to True, enables driver to open message channel in device. When device supports asynchronous events a message channel is used to deliver notifications with their data. Disabling message channel will disable event notifications. When device is not supporting message channel its opening will fail and driver will continue without message channel. Default: True
GevAppTLIgnoreGevImageTrailer	Boolean	When set to True, causes that image format parameters sent in image trailer packet are ignored and only image data leader participates in image format. Default: False

GevAppTLEnablePacketsRetransmission	Boolean	Enables stream missing packets detection and retransmission request (when supported by the device). Default: True
GevAppTLResetBlockIdOnAcquisitionStart	Boolean	When enabled resets stream channels BlockId counter on every acquisition start (used to resolve some compatibility issues with cameras). Default: False
GevAppTLEnableGenApiCache	Boolean	When set to True, enables GenApi cache. GenApi cache will store values read from device in the local computer memory to limit network traffic. Not all parameters can be cached. Cache performance depends on individual device and accessed parameters. Disabling cache will cause driver to read all parameters from device every time, including meta data like value ranges and access mode. Disabling cache will decrease speed of GenApi parameter operations. Default: True
GevAppTLRequest64BitBlockIdMode	Boolean	For GigEVision 2.0 compatible devices enables requesting the device to use 64bit Block ID mode. Default: True
GevAppTLActivateDevicePTP	Boolean	When set activates Precision Time Protocol in the device (when supported by the device). Default: False
GevAppTLAllowLocalXmlOverride	Boolean	When set to True allows to load GenICam xml file from local file system (when found) instead of location specified by device. Default: True

Examples

This example shows how to set up the library for forcing video stream packet size in a device equal to 8000 bytes.

```
avl::GigEHandle hSet = avl::GigEVision_OpenSystemConfiguration();

avl::GenApi_SetBooleanParam(hSet, "GevAppTLDisablePacketSizeNegotiation", true);
avl::GenApi_SetBooleanParam(hSet, "GevAppTLEnableConstantPacketSize", true);
avl::GenApi_SetIntegerParam(hSet, "GevAppTLConstantPacketSize", 8000);

avl::GigEVision_CloseHandle(hSet);
```

GigEVision_ReceiveImage

Header: Genicam.h
Namespace: avl
Module: Genicam

Receives next frame from video stream.

Syntax

```
void avl::GigEVision_ReceiveImage
(
 GigEHandle inDeviceHandle,
 avl::Image& outImage,
 atl::Optional<atl::uint64&> outFrameId = atl::NIL,
 atl::Optional<atl::uint64&> outTimestamp = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inDeviceHandle	GigEHandle		Handle of an opened device that is streaming video.
←	outImage	Image&		Image buffer that will receive a new frame.
←	outFrameId	Optional<uint64&>	NIL	Frame block Id set by device
←	outTimestamp	Optional<uint64&>	NIL	Frame capture timestamp set by device (when supported)

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outFrameId**, **outTimestamp**.

Read more about Optional Outputs.

Description

This function will return the next available frame from incoming buffer. When input buffer is empty the function will block execution and wait for the next frame to be completed.

The function returns only complete and valid frames, when a frame is corrupted and it cannot be fixed, it is automatically discarded by the library.

A frame is converted from the device stream pixel format to the best suitable format of avl::Image. Automatic conversion includes:

- Conversion from Bayer color mosaic to RGB.
- Conversion of YUV to RGB.
- Ordering of RGB/aRGB channels.
- Unpacking 10, 12 or 14 bpp into 16 bpp images.

Returned image can contain 1, 3 or 4 channels with sint8, uint8 or uint16 pixel type.

outFrameId parameter is optional (can be set to Nil, when not used or given a reference to an output variable). This parameter returns a number identifying current frame in video sequence. This number is generated by device and incremented by one at the beginning of transmission of every frame block. The valid range of this id, for GigEVision compliant devices, is 1...65535 (starting at 1 and incrementing sequentially up to 65535, then wrapping back to 1). This value can be used to control correctness of received image sequence and detection of lost frames. Please note that this method can be used only to detect frames that was transmitted by device but lost before application could process them. This method cannot be used when lost image was not transmitted by device (e.g. when device missed trigger signal).

outTimestamp parameter is optional (can be set to Nil, when not used or given a reference to an output variable). This parameter returns a timestamp of current frame that was marked by the device at point of grabbing the image. Support of this parameter by the device is optional and for some devices this output can be not effective. Frequency of the timer used for this timestamp is device dependent. Refer to device documentation for details about timestamp timer. Usually a frequency of this timer can be retrieved using a GevTimestampTickFrequency parameter available in device parameters tree. Value of this output will wrap around in range from 0 to 2^{31}-1.

Exceptions

This function will throw an exception in the following situations:

- Device handle is invalid.
- Device is not streaming video.
- Timeout occurred when waiting for next complete frame.
- Device sends image in pixel format that cannot be converted.
- Device streams in a mode that does not transport images or the library is unable to read the image from stream.
- Device connection is lost and cannot be restored.

See Also

- GigEVision_OpenDevice – Opens a handle to a GigE Vision device.
- GigEVision_StartAcquisition – Initializes and starts image acquisition in a device.

GigEVision_StartAcquisition

Header: Genicam.h
Namespace: avl
Module: Genicam

Initializes and starts image acquisition in a device.

Applications: Typically used for initialization of multi-camera systems.

Syntax

```
void avl::GigEVision_StartAcquisition
(
 GigEHandle inDeviceHandle,
 const atl::String& inPixelFormat,
 int inInputQueueSize = 1
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inDeviceHandle	GigEHandle			Handle of an opened device that should start acquisition.
➡	inPixelFormat	const String&			Pixel format name in GenICam naming convention that will be configured as device stream PixelFormat parameter.
➡	inInputQueueSize	int	1 - 400	1	Capacity of queue of input frames.

Description

This function sets up the device to stream with specified pixel format, prepares internal buffers for stream and starts acquisition in device by executing StartAcquisition command.

Set **inPixelFormat** parameter to empty string to leave current device pixel format unchanged.

Use GigEVision_GetPixelFormats function to determine device supported pixel formats and their names.

Do not access GenICam StartAcquisition and StopAcquisition commands directly, always use GigEVision_StartAcquisition and GigEVision_StopAcquisition functions so that the library state can stay in sync with device state.

Exceptions

This function will throw an exception in the following situations:

- Device handle is invalid.
- Device is not a video transmitter.
- Specified pixel format name is not supported by device.
- Device connection is lost.
- Other unexpected GenICam error or device setup error occurred.

See Also

- GigEVision_StopAcquisition – Stops image acquisition in a device.

GigEVision_StopAcquisition

Header: Genicam.h
Namespace: avl
Module: Genicam

Stops image acquisition in a device.

Syntax

```
void avl::GigEVision_StopAcquisition
(
 GigEHandle inDeviceHandle
)
```

Parameters

	Name	Type	Default	Description
➡	inDeviceHandle	GigEHandle		Handle of an opened device that should stop acquisition

Description

This function stops acquisition started by GigEVision_StartAcquisition by executing GenICam StopAcquisition command. The function will not fail when a device is not streaming video.

Do not access GenICam StartAcquisition and StopAcquisition commands directly, always use GigEVision_StartAcquisition and GigEVision_StopAcquisition function so that the library state can stay in sync with device state.

Exceptions

This function will throw an exception in the following situations:

- Device handle is invalid.
- Connection is lost.
- Other unexpected GenICam or network error occurred.

GigEVision_TryReceiveImage

Header: Genicam.h
Namespace: avl
Module: Genicam

Receives next frame from video stream.

Syntax

```
bool avl::GigEVision_TryReceiveImage
(
 GigEHandle inDeviceHandle,
 int inTimeout,
 avl::Image& outImage,
 atl::Optional<atl::uint64&> outFrameId = atl::NIL,
 atl::Optional<atl::uint64&> outTimestamp = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inDeviceHandle	GigEHandle		Handle of an opened device that is streaming video.
→	inTimeout	int		Maximum time in milliseconds that the function is allowed to wait for the next complete frame. This parameter overrides the global frame timeout configuration.
←	outImage	Image&		Image buffer that will receive a new frame.
←	outFrameId	Optional<uint64&>	NIL	Frame block Id set by device
←	outTimestamp	Optional<uint64&>	NIL	Frame capture timestamp set by device (when supported)

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outFrameId**, **outTimestamp**.

Read more about Optional Outputs.

Description

This function is a close equivalent of GigEVision_ReceiveImage, that gives additional control over frame timeout situation.

See GigEVision_ReceiveImage definition for more information.

Return Value

`true` when new frame is successfully received and stored into **outImage**, `false` when timeout occurs or device is not streaming.

Exceptions

This function will throw an exception in the following situations:

- Device handle is invalid.
- Connection with device is lost.
- Device sends image in pixel format that cannot be converted.
- Device streams in a mode that does not transport images or the library is unable to read the image from stream.

197. GenApi

Table of content:

GenApi_ExecuteCommand

Header: Genicam.h
Namespace: avl
Module: Genicam

Executes named command parameter using GenICam ICommand interface.

Syntax

```
void avl::GenApi_ExecuteCommand
(
 GenApiHandle inHandle,
 const char* inCommandName,
 bool inVerifyAccess = true
)
```

Parameters

	Name	Type	Default	Description
➡	inHandle	GenApiHandle		Handle to object providing GenApi interface.
➡	inCommandName	const char*		Textual name of the command parameter that should be executed in GenICam naming convention or device specific name. Command name is case sensitive.
➡	inVerifyAccess	bool	true	True to verify parameter access.

Description

This function accesses a device or software module using the GenICam GenApi interface. The actual GenApi interface is accessed through the handle of opened device or software module provided by other subsystem (like opened GigEVision device handle). A call to this function may result in time expensive (blocking) data exchange with device.

This function does not control parameter state or wait for command completion. Accessing parameter in an improper device state may silently fail, results on the device side are undefined.

Refer to device documentation or use the Device Manager to find the proper command parameter name. Do not use commands to start or stop image acquisition, use dedicated library's APIs instead.

Exceptions

This function will throw an exception in the following situations:

- Object handle is invalid or does not provide the GenApi interface.
- GenApi description does not provide a parameter with the specified name.
- Named parameter does not provide the ICommand interface.
- Connection with device is lost.
- Other unexpected GenICam or connection error occurred.

GenApi_GetBooleanParam

Header: Genicam.h
Namespace: avl
Module: Genicam

Reads named parameter value using GenICam IBoolean interface.

Syntax

```
bool avl::GenApi_GetBooleanParam
(
 GenApiHandle inHandle,
 const char* inParameterName,
 bool inVerifyAccess = true
)
```

Parameters

	Name	Type	Default	Description
➡	inHandle	GenApiHandle		Handle to object providing GenApi interface.
➡	inParameterName	const char*		Textual name of the parameter that should be accessed in GenICam naming convention or device specific name. Parameter name is case sensitive.
➡	inVerifyAccess	bool	true	True to verify parameter access.

Description

This function accesses the configuration of a device or software module using the GenICam GenApi interface. The actual GenApi interface is accessed through the handle of opened device or software module provided by other subsystem (like opened GigEVision device handle). A call to this function may result in time expensive (blocking) data exchange with device.

Set **inVerifyAccess** argument to true to check parameter state. When this argument is false accessing parameter in an improper device state may silently fail, the result is undefined.

Refer to device documentation or use the Device Manager to find proper parameter name.

Exceptions

This function will throw an exception in the following situations:

- Object handle is invalid or does not provide the GenApi interface.
- GenApi description does not provide a parameter with specified name.
- Named parameter does not provide the IBoolean interface.
- Connection with device is lost.
- Other unexpected GenICam or connection error occurred.

GenApi_GetCategoryDescriptor

Header: Genicam.h
Namespace: avl
Module: Genicam

Reads named category description from GenICam interface.

Syntax

```
void avl::GenApi_GetCategoryDescriptor
(
 GenApiHandle inHandle,
 const char* inCategoryName,
 GenApi_CategoryDescriptor& outDescriptor
)
```

Parameters

	Name	Type	Default	Description
→	inHandle	GenApiHandle		Handle to object providing GenApi interface.
→	inCategoryName	const char*		Textual name of the category that should be accessed in GenICam naming convention or device specific name. Category name is case sensitive.
←	outDescriptor	GenApi_CategoryDescriptor&		

Description

This function accesses the configuration of a device or a software module using the GenICam GenApi interface. The actual GenApi interface is accessed through the handle of opened device or software module provided by other subsystem (like opened GigEVision device handle).

Categories are special types of GenApi nodes not associated with a parameter value. Instead, a category contains a list of other parameters or categories, forming a tree structure of top level parameters. Categories' trees are usable for creating user interface. The main category of every GenApi set is named "Root".

This function can be used to obtain description of named Category. Information is returned using GenApi_CategoryDescriptor structure.

Exceptions

This function will throw an exception in the following situations:

- Object handle is invalid or does not provide the GenApi interface.
- GenApi description does not provide a category with specified name.
- Named parameter does not provide the ICategory interface.

GenApi_GetEnumDescriptor

Header: Genicam.h
Namespace: avl
Module: Genicam

Reads named enum parameter description from GenICam IEnumeration interface.

Syntax

```
void avl::GenApi_GetEnumDescriptor
(
 GenApiHandle inHandle,
 const char* inParameterName,
 bool inAvailableEntriesOnly,
 GenApi_EnumDescriptor& outDescriptor
)
```

Parameters

	Name	Type	Default	Description
→	inHandle	GenApiHandle		Handle to object providing GenApi interface.
→	inParameterName	const char*		Textual name of the parameter that should be accessed in GenICam naming convention or device specific name. Parameter name is case sensitive.
→	inAvailableEntriesOnly	bool		True to include in returned description only implemented and available enumeration entries.
←	outDescriptor	GenApi_EnumDescriptor&		

Description

This function accesses the configuration of a device or software module using the GenICam GenApi interface. The actual GenApi interface is accessed through the handle of opened device or software module provided by other subsystem (like opened GigEVision device handle). A call to this function may result in time expensive (blocking) data exchange with device.

This function can be used to obtain the description of named Enumeration parameter with the list of enumeration entries. The information is returned using GenApi_EnumDescriptor structure.

Exceptions

This function will throw an exception in the following situations:

- Object handle is invalid or does not provide the GenApi interface.
- GenApi description does not provide a parameter with specified name.
- Named parameter does not provide the IEnumeration interface.
- Connection with device is lost.
- Other unexpected GenICam or connection error occurred.

GenApi_GetEnumParam

Header: Genicam.h
Namespace: avl
Module: Genicam

Reads named parameter value using GenICam IEnumeration interface.

Syntax

```
void avl::GenApi_GetEnumParam
(
 GenApiHandle inHandle,
 const char* inParameterName,
 bool inVerifyAccess,
 atl::String& outValue
)
```

Parameters

	Name	Type	Default	Description
→	inHandle	GenApiHandle		Handle to object providing GenApi interface.
→	inParameterName	const char*		Textual name of the parameter that should be accessed in GenICam naming convention or device specific name. Parameter name is case sensitive.
→	inVerifyAccess	bool		True to verify parameter access.
←	outValue	String&		

Description

This function accesses the configuration of a device or software module using the GenICam GenApi interface. The actual GenApi interface is accessed through the handle of opened device or software module provided by other subsystem (like opened GigEVision device handle). A call to this function may result in time expensive (blocking) data exchange with device.

The returned value is a textual id of the currently selected enumeration entry.

Set **inVerifyAccess** argument to true to check parameter state. When this argument is false accessing parameter in an improper device state may silently fail, the result is undefined.

Refer to device documentation or use the Device Manager to find proper parameter name.

Exceptions

This function will throw an exception in the following situations:

- Object handle is invalid or does not provide the GenApi interface.
- GenApi description does not provide a parameter with specified name.
- Named parameter does not provide the IEnumeration interface.
- Connection with device is lost.
- Other unexpected GenICam or connection error occurred.

GenApi_GetFloatDescriptor

Header: Genicam.h
Namespace: avl
Module: Genicam

Reads named float parameter attributes from GenICam IFloat interface.

Syntax

```
void avl::GenApi_GetFloatDescriptor
(
 GenApiHandle inHandle,
 const char* inParameterName,
 GenApi_FloatDescriptor& outDescriptor
)
```

Parameters

	Name	Type	Default	Description
→	inHandle	GenApiHandle		Handle to object providing GenApi interface.
→	inParameterName	const char*		Textual name of the parameter that should be accessed in GenICam naming convention or device specific name. Parameter name is case sensitive.
←	outDescriptor	GenApi_FloatDescriptor&		

Description

This function accesses the configuration of a device or software module using the GenICam GenApi interface. The actual GenApi interface is accessed through the handle of opened device or software module provided by other subsystem (like opened GigEVision device handle). A call to this function may result in time expensive (blocking) data exchange with device.

This function can be used to obtain additional information of named Float parameter, like its range. The information is returned using GenApi_FloatDescriptor structure.

Exceptions

This function will throw an exception in the following situations:

- Object handle is invalid or does not provide the GenApi interface.
- GenApi description does not provide a parameter with specified name.
- Named parameter does not provide the IFloat interface.
- Connection with device is lost.
- Other unexpected GenICam or connection error occurred.

GenApi_GetFloatParam

Header: Genicam.h
Namespace: avl
Module: Genicam

Reads named parameter value using GenICam IFloat interface.

Syntax

```
double avl::GenApi_GetFloatParam
(
 GenApiHandle inHandle,
 const char* inParameterName,
 bool inVerifyAccess = true
)
```

Parameters

	Name	Type	Default	Description
➡	inHandle	GenApiHandle		Handle to object providing GenApi interface.
➡	inParameterName	const char*		Textual name of the parameter that should be accessed in GenICam naming convention or device specific name. Parameter name is case sensitive.
➡	inVerifyAccess	bool	true	True to verify parameter access.

Description

This function accesses the configuration of a device or software module using the GenICam GenApi interface. The actual GenApi interface is accessed through the handle of opened device or software module provided by other subsystem (like opened GigEVision device handle). A call to this function may result in time expensive (blocking) data exchange with device.

Set **inVerifyAccess** argument to true to check parameter state. When this argument is set to false accessing parameter in an improper device state may silently fail, the result is undefined.

Refer to device documentation or use the Device Manager to find proper parameter name.

Exceptions

This function will throw an exception in the following situations:

- Object handle is invalid or does not provide the GenApi interface.
- GenApi description does not provide a parameter with specified name.
- Named parameter does not provide the IFloat interface.
- Connection with device is lost.
- Other unexpected GenICam or connection error occurred.

GenApi_GetIntegerDescriptor

Header: Genicam.h
Namespace: avl
Module: Genicam

Reads named integer parameter attributes from GenICam IInteger interface.

Syntax

```
void avl::GenApi_GetIntegerDescriptor
(
 GenApiHandle inHandle,
 const char* inParameterName,
 GenApi_IntegerDescriptor& outDescriptor
)
```

Parameters

	Name	Type	Default	Description
→	inHandle	GenApiHandle		Handle to object providing GenApi interface.
→	inParameterName	const char*		Textual name of the parameter that should be accessed in GenICam naming convention or device specific name. Parameter name is case sensitive.
←	outDescriptor	GenApi_IntegerDescriptor&		

Description

This function accesses the configuration of a device or software module using the GenICam GenApi interface. The actual GenApi interface is accessed through the handle of opened device or software module provided by other subsystem (like opened GigEVision device handle). A call to this function may result in time expensive (blocking) data exchange with device.

This function can be used to obtain additional information of named Integer parameter, like its range and allowed increment. The information is returned using GenApi_IntegerDescriptor structure.

Exceptions

This function will throw an exception in the following situations:

- Object handle is invalid or does not provide the GenApi interface.
- GenApi description does not provide a parameter with specified name.
- Named parameter does not provide the IInteger interface.
- Connection with device is lost.
- Other unexpected GenICam or connection error occurred.

GenApi_GetIntegerParam

Header: Genicam.h
Namespace: avl
Module: Genicam

Reads named parameter value using GenICam IInteger interface.

Syntax

```
atl::sint64 avl::GenApi_GetIntegerParam
(
 GenApiHandle inHandle,
 const char* inParameterName,
 bool inVerifyAccess = true
)
```

Parameters

	Name	Type	Default	Description
➡	inHandle	GenApiHandle		Handle to object providing GenApi interface.
➡	inParameterName	const char*		Textual name of the parameter that should be accessed in GenICam naming convention or device specific name. Parameter name is case sensitive.
➡	inVerifyAccess	bool	true	True to verify parameter access.

Description

This function accesses the configuration of a device or software module using GenICam GenApi interface. The actual GenApi interface is accessed through the handle of opened device or software module provided by other subsystem (like opened GigEVision device handle). A call to this function may result in time expensive (blocking) data exchange with device.

Set **inVerifyAccess** argument to true to check parameter state. When this argument is set to false accessing parameter in an improper device state may silently fail, the result is undefined.

Refer to device documentation or use device manager to find proper parameter name.

Exceptions

This function will throw an exception in the following situations:

- Object handle is invalid or does not provide the GenApi interface.
- GenApi description does not provide a parameter with specified name.
- Named parameter does not provide the IInteger interface.
- Connection with device is lost.
- Other unexpected GenICam or connection error occurred.

GenApi_GetParamDescriptor

Header: Genicam.h
Namespace: avl
Module: Genicam

Reads named parameter description and attributes from GenICam interface.

Syntax

```
void avl::GenApi_GetParamDescriptor
(
 GenApiHandle inHandle,
 const char* inParameterName,
 GenApi_ParameterDescriptor& outDescriptor
)
```

Parameters

	Name	Type	Default	Description
→	inHandle	GenApiHandle		Handle to object providing GenApi interface.
→	inParameterName	const char*		Textual name of the parameter that should be accessed in GenICam naming convention or device specific name. Parameter name is case sensitive.
←	outDescriptor	GenApi_ParameterDescriptor&		

Description

This function accesses the configuration of a device or software module using the GenICam GenApi interface. The actual GenApi interface is accessed through the handle of opened device or software module provided by other subsystem (like opened GigEVision device handle). A call to this function may result in time expensive (blocking) data exchange with device.

This function can be used to obtain additional information, meta data and the state of a GenICam parameter. The information is returned using GenApi_ParameterDescriptor structure.

Exceptions

This function will throw an exception in the following situations:

- Object handle is invalid or does not provide the GenApi interface.
- GenApi description does not provide a parameter with specified name.
- Connection with device is lost.
- Other unexpected GenICam or connection error occurred.

GenApi_GetParamExists

Header: Genicam.h
Namespace: avl
Module: Genicam

Checks if parameter, category or command with specified name exists in GenApi set.

Syntax

```
bool avl::GenApi_GetParamExists
(
 GenApiHandle inHandle,
 const char* inParameterName
)
```

Parameters

	Name	Type	Default	Description
→	inHandle	GenApiHandle		Handle to object providing GenApi interface.
→	inParameterName	const char*		Textual name of the parameter, category or command that should be checked in GenICam naming convention or device specific name. Name is case sensitive.

Description

This function accesses the configuration of a device or software module using the GenICam GenApi interface. The actual GenApi interface is accessed through the handle of opened device or software module provided by other subsystem (like opened GigEVision device handle). A call to this function may result in time expensive (blocking) data exchange with device.

Return value

False, when GenApi description does not provide a parameter or command with specified name. True, when any kind of parameter or command exists in the referenced GenApi description, regardless of its interface, implementation state or availability (use GenApi_GetParamDescriptor function to retrieve more information about existing parameter).

Exceptions

This function will throw an exception in the following situations:

- Object handle is invalid or does not provide the GenApi interface.

GenApi_GetRegisterDescriptor

Header: Genicam.h
Namespace: avl
Module: Genicam

Reads named register parameter description from GenICam IRegister interface.

Syntax

```
void avl::GenApi_GetRegisterDescriptor
(
 GenApiHandle inHandle,
 const char* inParameterName,
 GenApi_RegisterDescriptor& outDescriptor
)
```

Parameters

	Name	Type	Default	Description
→	inHandle	GenApiHandle		Handle to object providing GenApi interface.
→	inParameterName	const char*		Textual name of the parameter that should be accessed in GenICam naming convention or device specific name. Parameter name is case sensitive.
←	outDescriptor	GenApi_RegisterDescriptor&		

GenApi_GetStringParam

Header: Genicam.h
Namespace: avl
Module: Genicam

Reads named parameter value using GenICam IInteger interface.

Syntax

```
void avl::GenApi_GetStringParam
(
 GenApiHandle inHandle,
 const char* inParameterName,
 bool inVerifyAccess,
 atl::String& outValue
)
```

Parameters

	Name	Type	Default	Description
→	inHandle	GenApiHandle		Handle to object providing GenApi interface.
→	inParameterName	const char*		Textual name of the parameter that should be accessed in GenICam naming convention or device specific name. Parameter name is case sensitive.
→	inVerifyAccess	bool		True to verify parameter access.
←	outValue	String&		

Description

This function accesses the configuration of a device or software module using the GenICam GenApi interface. The actual GenApi interface is accessed through the handle of opened device or software module provided by other subsystem (like opened GigEVision device handle). A call to this function may result in time expensive (blocking) data exchange with device.

Set **inVerifyAccess** argument to true to check parameter state. When this argument is set to false accessing parameter in an improper device state may silently fail, the result is undefined.

Refer to device documentation or use the Device Manager to find proper parameter name.

Exceptions

This function will throw an exception in the following situations:

- Object handle is invalid or does not provide the GenApi interface.
- GenApi description does not provide a parameter with specified name.
- Named parameter does not provide the IString interface.
- Connection with device is lost.
- Other unexpected GenICam or connection error occurred.

GenApi_ReadRegisterData

Header: Genicam.h
Namespace: avl
Module: Genicam

Reads raw binary data from the objects memory using the IRegister interface.

Syntax

```
void avl::GenApi_ReadRegisterData
(
 GenApiHandle inHandle,
 const char* inParameterName,
 bool inVerifyAccess,
 atl::sint64 inLength,
 void* ioBuffer
)
```

Parameters

	Name	Type	Default	Description
→	inHandle	GenApiHandle		Handle to object providing GenApi interface.
→	inParameterName	const char*		Textual name of the parameter that should be accessed in GenICam naming convention or device specific name. Parameter name is case sensitive.
→	inVerifyAccess	bool		True to verify parameter access.
→	inLength	sint64		Size of the data to read.
↔	ioBuffer	void*		Buffer for the read data to be placed in (allocated with at least inLength bytes).

Description

This function accesses the configuration of a device or software module using the GenICam GenApi interface. The actual GenApi interface is accessed through the handle of opened device or software module provided by other subsystem (like opened GigEVision device handle). A call to this function may result in time expensive (blocking) data exchange with device.

Application must allocate a buffer for the data and provide its pointer on the **ioBuffer** parameter. **inLength** specifies the amount of data (in bytes) to read from the register (starting at its beginning). GenApi_GetRegisterDescriptor can be used to determine the register size.

Set **inVerifyAccess** argument to true to check parameter state. When this argument is set to false accessing parameter in an improper device state may silently fail, the result is undefined.

Refer to the device documentation or use Device Manager to find a proper parameter name.

Exceptions

This function will throw an exception in the following situations:

- Object handle is invalid or does not provide the GenApi interface.
- GenApi description does not provide a parameter with specified name.
- Named parameter does not provide the IRegister interface.
- Null pointer is provided on the **ioBuffer** parameter.
- Requested data read size is larger than the range of the register.
- Connection with device is lost.
- Other unexpected GenICam or connection error occurred.

Errors

List of possible exceptions:

Error type	Description
DomainError	Null pointer on ioBuffer parameter.

GenApi_SetBooleanParam

Header: Genicam.h
Namespace: avl
Module: Genicam

Writes named parameter value using GenICam IBoolean interface.

Syntax

```
void avl::GenApi_SetBooleanParam
(
 GenApiHandle inHandle,
 const char* inParameterName,
 bool inValue,
 bool inVerify = true
)
```

Parameters

	Name	Type	Default	Description
➡	inHandle	GenApiHandle		Handle to object providing GenApi interface.
➡	inParameterName	const char*		Textual name of the parameter that should be accessed in GenICam naming convention or device specific name. Parameter name is case sensitive.
➡	inValue	bool		New value that should be written to parameter.
➡	inVerify	bool	true	True to verify parameter access.

Description

This function accesses the configuration of a device or software module using the GenICam GenApi interface. The actual GenApi interface is accessed through the handle of opened device or software module provided by other subsystem (like opened GigEVision device handle). A call to this function may result in time expensive (blocking) data exchange with device.

Set **inVerify** argument to true to check parameter state. When this argument is set to false accessing parameter in an improper device state may silently fail, the results on device side are undefined.

Refer to device documentation or use the Device Manager to find proper parameter name.

Exceptions

This function will throw an exception in the following situations:

- Object handle is invalid or does not provide the GenApi interface.
- GenApi description does not provide parameter with specified name.
- Named parameter does not provide the IBoolean interface.
- Connection with device is lost.
- Other unexpected GenICam or connection error occurred.

GenApi_SetEnumParam

Header: Genicam.h
Namespace: avl
Module: Genicam

Writes named parameter value using GenICam IEnumeration interface.

Syntax

```
void avl::GenApi_SetEnumParam
(
 GenApiHandle inHandle,
 const char* inParameterName,
 const char* inValue,
 bool inVerify = true
)
```

Parameters

	Name	Type	Default	Description
➡	inHandle	GenApiHandle		Handle to object providing GenApi interface.
➡	inParameterName	const char*		Textual name of the parameter that should be accessed in GenICam naming convention or device specific name. Parameter name is case sensitive.
➡	inValue	const char*		New enumeration entry id that should be written to parameter.
➡	inVerify	bool	true	True to verify parameter access and value range and alignment.

Description

This function accesses the configuration of a device or software module using the GenICam GenApi interface. The actual GenApi interface is accessed through the handle of opened device or software module provided by other subsystem (like opened GigEVision device handle). A call to this function may result in time expensive (blocking) data exchange with device.

The new value must be specified as a textual id of one of enumeration entries.

Set **inVerify** argument to true to check new value range, increment and parameter state. When this argument is set to false writing invalid value or accessing parameter in an improper device state will silently fail, the results on device side are undefined.

Refer to device documentation or use the Device Manager to find proper parameter name.

Exceptions

This function will throw an exception in the following situations:

- Object handle is invalid or does not provide the GenApi interface.
- GenApi description does not provide a parameter with specified name.
- Named parameter does not provide the IEnumeration interface.
- Enumeration value does not exist in parameter.
- Connection with device is lost.
- Other unexpected GenICam or connection error occurred.

GenApi_SetFloatParam

Header: Genicam.h
Namespace: avl
Module: Genicam

Writes named parameter value using GenICam IFloat interface.

Syntax

```
void avl::GenApi_SetFloatParam
(
 GenApiHandle inHandle,
 const char* inParameterName,
 double inValue,
 bool inVerify = true
)
```

Parameters

	Name	Type	Default	Description
➡	inHandle	GenApiHandle		Handle to object providing GenApi interface.
➡	inParameterName	const char*		Textual name of the parameter that should be accessed in GenICam naming convention or device specific name. Parameter name is case sensitive.
➡	inValue	double		New value that should be written to parameter.
➡	inVerify	bool	true	True to verify parameter access and value range.

Description

This function accesses the configuration of a device or software module using the GenICam GenApi interface. The actual GenApi interface is accessed through the handle of opened device or software module provided by other subsystem (like opened GigEVision device handle). A call to this function may result in time expensive (blocking) data exchange with device.

Set **inVerify** argument to true to check new value range and parameter state. When this argument is set to false writing invalid value or accessing parameter in an improper device state will silently fail, the results on device side are undefined.

Refer to device documentation or use the Device Manager to find proper parameter name.

Exceptions

This function will throw an exception in the following situations:

- Object handle is invalid or does not provide the GenApi interface.
- GenApi description does not provide a parameter with specified name.
- Named parameter does not provide the IFloat interface.
- Connection with device is lost.
- Other unexpected GenICam or connection error occurred.

GenApi_SetIntegerParam

Header: Genicam.h
Namespace: avl
Module: Genicam

Writes named parameter value using GenICam IInteger interface.

Syntax

```
void avl::GenApi_SetIntegerParam
(
 GenApiHandle inHandle,
 const char* inParameterName,
 atl::sint64 inValue,
 bool inVerify = true
)
```

Parameters

	Name	Type	Default	Description
➡	inHandle	GenApiHandle		Handle to object providing GenApi interface.
➡	inParameterName	const char*		Textual name of the parameter that should be accessed in GenICam naming convention or device specific name. Parameter name is case sensitive.
➡	inValue	sint64		New value that should be written to parameter.
➡	inVerify	bool	true	True to verify parameter access and value range and alignment.

Description

This function accesses the configuration of a device or software module using the GenICam GenApi interface. The actual GenApi interface is accessed through the handle of opened device or software module provided by other subsystem (like opened GigEVision device handle). A call to this function may result in time expensive (blocking) data exchange with device.

Set **inVerify** argument to true to check new value range, increment and parameter state. When this argument is set to false writing invalid value or accessing parameter in an improper device state will silently fail, the results on device side are undefined.

Refer to device documentation or use the Device Manager to find proper parameter name.

Exceptions

This function will throw an exception in the following situations:

- Object handle is invalid or does not provide the GenApi interface.
- GenApi description does not provide a parameter with specified name.
- Named parameter does not provide the IInteger interface.
- Connection with device is lost.
- Other unexpected GenICam or connection error occurred.

GenApi_SetStringParam

Header: Genicam.h
Namespace: avl
Module: Genicam

Writes named parameter value using GenICam IString interface.

Syntax

```
void avl::GenApi_SetStringParam
(
 GenApiHandle inHandle,
 const char* inParameterName,
 const char* inValue,
 bool inVerify = true
)
```

Parameters

	Name	Type	Default	Description
➡	inHandle	GenApiHandle		Handle to object providing GenApi interface.
➡	inParameterName	const char*		Textual name of the parameter that should be accessed in GenICam naming convention or device specific name. Parameter name is case sensitive.
➡	inValue	const char*		New value that should be written to parameter.
➡	inVerify	bool	true	True to verify parameter access and value range and alignment.

Description

This function accesses the configuration of a device or software module using the GenICam GenApi interface. The actual GenApi interface is accessed through the handle of opened device or software module provided by other subsystem (like opened GigEVision device handle). A call to this function may result in time expensive (blocking) data exchange with device.

Set **inVerify** argument to true to check parameter state. When this argument is set to false accessing parameter in an improper device state may silently fail, the results on device side are undefined.

Refer to device documentation or use the Device Manager to find proper parameter name.

Exceptions

This function will throw an exception in the following situations:

- Object handle is invalid does not provide the GenApi interface.
- GenApi description does not provide a parameter with specified name.
- Named parameter does not provide the IString interface.
- Connection with device is lost.
- Other unexpected GenICam or connection error occurred.

GenApi_WriteRegisterData

Header: Genicam.h
Namespace: avl
Module: Genicam

Writes raw binary data into the objects memory using the IRegister interface.

Syntax

```
void avl::GenApi_WriteRegisterData
(
 GenApiHandle inHandle,
 const char* inParameterName,
 const void* inData,
 atl::sint64 inLength,
 bool inVerifyAccess = true
)
```

Parameters

	Name	Type	Default	Description
➡	inHandle	GenApiHandle		Handle to object providing GenApi interface.
➡	inParameterName	const char*		Textual name of the parameter that should be accessed in GenICam naming convention or device specific name. Parameter name is case sensitive.
➡	inData	const void*		Data that should be written to the parameter.
➡	inLength	sint64		Size of the data to be written.
➡	inVerifyAccess	bool	true	True to verify the parameter access.

Description

This function accesses the configuration of a device or software module using the GenICam GenApi interface. The actual GenApi interface is accessed through the handle of opened device or software module provided by other subsystem (like opened GigEVision device handle). A call to this function may result in time expensive (blocking) data exchange with the device.

Application provides a pointer on the data to write in the **inData** parameter, **inLength** specifies the amount (in bytes) of data to write (starting on the beginning of the register). GenApi_GetRegisterDescriptor can be used to determine the register size.

Set **inVerifyAccess** argument to true to check parameter state. When this argument is set to false accessing parameter in an improper device state may silently fail, the result is undefined.

Refer to the device documentation or use Device Manager to find a proper parameter name.

Exceptions

This function will throw an exception in the following situations:

- Object handle is invalid does not provide the GenApi interface.
- GenApi description does not provide a parameter with specified name.
- Named parameter does not provide the IRegister interface.
- Null pointer is provided on the **inData** parameter.
- Requested data write size is larger than the range of the register.
- Connection with device is lost.
- Other unexpected GenICam or connection error occurred.

Errors

List of possible exceptions:

Error type	Description
DomainError	Null pointer on inData parameter.

198. GenTL

Table of content:

GenTL_Cleanup

Header: Genicam.h
Namespace: avl
Module: Genicam

Closes all internally opened (cached) GenTL software modules not held any more by any opened handle.

Syntax

```
void avl::GenTL_Cleanup
(
)
```

Description

When accessing structures of the GenTL system, software modules are opened to manage its entities (like library system module or interface module). GenTL subsystem can left some modules opened after operations on such structures (like enumeration or information access) to optimize subsequent function calls. Every open module uses some resources, both on GenTL provider side and on AVL side. You can use this function to force to close unused modules and to release unused resources.

Only unreferenced modules will be closed. A module is unreferenced, when the application does not hold any handle to it or its child modules.

GenTL subsystem never leaves a device module opened after its last handle has been closed. Also, the cleanup operation is performed after closing of any device.

GenTL_CloseHandle

Header: Genicam.h
Namespace: avl
Module: Genicam

Closes a handle opened in GenTL subsystem.

Syntax

```
void avl::GenTL_CloseHandle
(
 GenTLHandle inHandle
)
```

Parameters

	Name	Type	Default	Description
→	inHandle	GenTLHandle		

GenTL_EnumLibraries

Header: Genicam.h
Namespace: avl
Module: Genicam

Returns a list of loaded GenTL provider libraries available in the local system.

Syntax

```
void avl::GenTL_EnumLibraries
(
 atl::Array<atl::String>& outLibraryPaths
)
```

Parameters

	Name	Type	Default	Description
←	outLibraryPaths	Array<String>&		List of paths identifying libraries.

Description

At first use of any function, the GenTL subsystem will enumerate and attempt to load GenTL provider libraries registered in system. This function returns a list of successfully loaded GenTL provider libraries that can be used to communicate with devices. Single GenTL provider is identified by a path to its main module (a file with .cti extension).

To obtain additional information about provider library use GenTL_GetLibraryDescriptor function. To enumerate communication interfaces provided by the library use GenTL_EnumLibraryInterfaces function.

GenTL_EnumLibraryInterfaces

Header: Genicam.h
Namespace: avl
Module: Genicam

Returns a list of communication interfaces available in specified provider module library.

Syntax

```
void avl::GenTL_EnumLibraryInterfaces
(
 const atl::String& inLibraryPath,
 atl::Array<atl::String>& outInterfaceIds
)
```

Parameters

	Name	Type	Default	Description
→	inLibraryPath	const String&		Path identifying provider library.
←	outInterfaceIds	Array<String>&		List of strings identifying interfaces.

Description

This function requests the GenTL provider to refresh a list of available communication interfaces. Parameter **outInterfaceIds** is than filled with unique textual ids of such interfaces.

To obtain additional information about an interface use GenTL_GetInterfaceDescriptor function.

See Also

- GenTL_EnumLibraries – Returns a list of loaded GenTL provider libraries available in the local system.

GenTL_FindDevices

Header: Genicam.h
Namespace: avl
Module: Genicam

Performs enumeration of GenTL devices and returns a list of present device descriptors.

Syntax

```
void avl::GenTL_FindDevices
(
 atl::Optional<const atl::String&> inLibraryPath,
 atl::Optional<const atl::String&> inInterfaceId,
 atl::Optional<const atl::String&> inTLType,
 int inInterfaceScanTime,
 atl::Array<GenTL_DeviceDescriptor>& outDevices
)
```

Parameters

	Name	Type	Default	Description
→	inLibraryPath	Optional<const String&>	NIL	Path identifying provider library to limit enumeration to.
→	inInterfaceId	Optional<const String&>	NIL	String identifying communication interface to limit enumeration to.
→	inTLType	Optional<const String&>	NIL	Name of interface transport technology to limit enumeration to.
→	inInterfaceScanTime	int		Time limit, in milliseconds, that the function will wait for device response on each enumerated interface.
←	outDevices	Array<GenTL_DeviceDescriptor>&		Returns a list with device list descriptors.

Description

This function requests available GenTL provider libraries to perform enumeration of their interfaces and devices, and returns a unified list of all available devices. The function can perform a global enumeration (all available libraries and interfaces) as well as limited enumeration, where search can be narrowed to specific library, interface or transport technology.

inLibraryPath parameter can be set to a path identifying one of available GenTL libraries to limit enumeration to (only interfaces provided by this library will be checked). Setting this parameter will not force the GenTL subsystem to load the specified file. Requested library must be properly registered in local system and loaded upon GenTL subsystem initialization. Setting this parameter to `atl::NIL` causes that all available providers will be included in enumeration.

inInterfaceId parameter can be set to an internal identifier of provider interfaces to limit enumeration to (only these interfaces will be enumerated). When this argument is used, parameter **inLibraryPath** must be also specified. Setting this parameter to `atl::NIL` causes that all interfaces of applicable providers are enumerated.

inTLType parameter can be set to a name of Transport Technology Type. When this argument is used, only interfaces with equal transport technology type name are enumerated. When this parameter is set to `atl::NIL` enumeration is not restricted by interface type. This parameter cannot be used simultaneously with **inInterfaceId** (at least one of those arguments must be equal `atl::NIL`).

Any textual name of transport technology can be used for **inTLType** argument, including following standard type names:

- **"Custom"** - Custom technologies.
- **"GEV"** - GigE Vision technology.
- **"CL"** - Camera Link technology.
- **"IIDC"** - IIDC 1394 technology.
- **"UVC"** - USB video class devices.
- **"CXP"** - CoaXPress.
- **"CLHS"** - Camera Link HS.
- **"USB3"** - USB3 Vision Standard.
- **"Ethernet"** - Ethernet devices.
- **"PCI"** - PCI/PCIe devices.

(according to GenTL standard version 1.3)

inInterfaceScanTime argument specifies the time limit in milliseconds, that a single provider interface will be given to enumerate its devices. Every applicable interface will be given the same amount of time. Exact time that this function will block execution depends on number of applicable providers, number of applicable interfaces and on the way that provider performs enumeration.

All found devices (regardless of parent library or interface) are stored in a unified list returned by **outDevices** parameter. Single device is described by GenTL_DeviceDescriptor structure. Please note, that it is not impossible that single device is reachable by multiple interfaces or provider libraries. In such case, single device will be present in the list multiple times.

Exceptions

This function will throw an exception in the following situation:

- Optional arguments are used in improper way.
- Library specified by **inLibraryPath** or interface specified by **inInterfaceId** is not found.
- Other general GenTL error occurred.

Errors

List of possible exceptions:

Error type	Description
DomainError	Parameter inLibraryPath is required when inInterfaceId is specified.
DomainError	Parameters inInterfaceId and inTLType cannot be used simultaneously.

GenTL_FlushInputQueue

Header: Genicam.h
Namespace: avl
Module: Genicam

Discards remaining received and queued image frames in device video stream.

Syntax

```
void avl::GenTL_FlushInputQueue
(
 GenTLHandle inDeviceHandle
)
```

Parameters

	Name	Type	Default	Description
➡	inDeviceHandle	GenTLHandle		Handle of an opened device which streaming queue should be cleared.

Description

Flushing input queue means that all frames that have been received from device and that have not been yet retrieved by the application, will be discarded and lost (including a partially filled frame).

Exceptions

This function will throw an exception in the following situations:

- Device handle is invalid.

GenTL_GetAcquisitionActive

Header: Genicam.h
Namespace: avl
Module: Genicam

Checks if specified device is currently streaming video.

Syntax

```
bool avl::GenTL_GetAcquisitionActive
(
 GenTLHandle inDeviceHandle
)
```

Parameters

	Name	Type	Default	Description
➡	inDeviceHandle	GenTLHandle		Handle of an opened device that will be checked for streaming.

Description

Return Value

`true`, when device video streaming is currently active on application side (GenTL_StartAcquisition function has been successfully executed for specified handle), `false` otherwise.

Exceptions

This function will throw an exception in the following situation:

- Device handle is invalid.

GenTL_GetInterfaceDescriptor

Header: Genicam.h
Namespace: avl
Module: Genicam

Reads description of GenTL provider communication interface.

Syntax

```
void avl::GenTL_GetInterfaceDescriptor
(
 const atl::String& inLibraryPath,
 const atl::String& inInterfaceId,
 GenTL_InterfaceDescriptor& outDescriptor
)
```

Parameters

	Name	Type	Default	Description
→	inLibraryPath	const String&		Path identifying provider library.
→	inInterfaceId	const String&		String identifying communication interface.
←	outDescriptor	GenTL_InterfaceDescriptor&		Description retrieved from provider interface.

Description

This function accesses provider interface module and retrieves additional information from it. Data is returned using GenTL_InterfaceDescriptor structure.

See Also

- GenTL_EnumLibraries – Returns a list of loaded GenTL provider libraries available in the local system.

GenTL_GetLibraryDescriptor

Header: Genicam.h
Namespace: avl
Module: Genicam

Reads description of GenTL provider library.

Syntax

```
void avl::GenTL_GetLibraryDescriptor
(
 const atl::String& inLibraryPath,
 GenTL_LibraryDescriptor& outDescriptor
)
```

Parameters

	Name	Type	Default	Description
→	inLibraryPath	const String&		Path identifying provider library.
←	outDescriptor	GenTL_LibraryDescriptor&		Description retrieved from provider library.

Description

This function accesses GenTL provider library system module and retrieves additional information from it. Data is returned using GenTL_LibraryDescriptor structure.

See Also

- GenTL_EnumLibraries – Returns a list of loaded GenTL provider libraries available in the local system.

GenTL_GetPixelFormats

Header: Genicam.h
Namespace: avl
Module: Genicam

Retrieves a list of available pixel formats supported by opened device.

Syntax

```
void avl::GenTL_GetPixelFormats
(
 GenTLHandle inDeviceHandle,
 atl::Array< atl::String >& outFormats
)
```

Parameters

	Name	Type	Default	Description
→	inDeviceHandle	GenTLHandle		Handle of an opened device.
←	outFormats	Array< String >&		Array that will return a list of format strings.

Description

This function accesses GenICam "PixelFormat" standard enumeration parameter and reads its entries. Only entries marked as implemented and available will be present in the returned list. The list will contain names of PixelFormat enumeration entries (not display names).

One of the format names returned by this function can be passed to GenTL_StartAcquisition function.

Exceptions

This function will throw an exception in the following situations:

- Device handle is invalid.
- Connection with device is lost.
- Device is not a video transmitter.
- Other general GenTL or GenICam error occurred.

GenTL_GetStreamingStatistics

Header: Genicam.h
Namespace: avl
Module: Genicam

Retrieves basic statistics of video streaming out of GenICam device.

Syntax

```
void avl::GenTL_GetStreamingStatistics
(
 GenTLHandle inDeviceHandle,
 GenTL_StreamingStatistics& outStatistics
)
```

Parameters

	Name	Type	Default	Description
→	inDeviceHandle	GenTLHandle		Handle of an open device that is streaming video.
←	outStatistics	GenTL_StreamingStatistics&		

Description

This function is intended to help with diagnosing video streaming issues and it allows to read basic streaming statistic counters from the acquisition engine.

Exceptions

This function will throw an exception in the following situations:

- Device handle is invalid.

GenTL_OpenDevice

Header: Genicam.h
Namespace: avl
Module: Genicam

Opens a handle to a device provided by GenTL library.

Syntax

```
GenTLHandle avl::GenTL_OpenDevice
(
 const atl::String& inLibraryPath,
 const atl::String& inInterfaceId,
 const atl::String& inDeviceId
)
```

Parameters

	Name	Type	Default	Description
➡	inLibraryPath	const String&		Path identifying provider library.
➡	inInterfaceId	const String&		String identifying communication interface through which device is accessed.
➡	inDeviceId	const String&		Unique identifier of device in provider communication interface.

Description

This function accesses a GenTL provider library modules' chain and device described by passed textual identifiers. When specified device is already opened upon function call, and a primary unique identifier of device is used, a new handle to the same device instance is returned.

inLibraryPath parameter identifies a provider library to use as a device driver. Although library is identified by its file path, this function will not load the specified file. Library must be properly registered in system and enumerated upon GenTL subsystem first use. **inInterfaceId** parameter specifies an internal provider library textual id of communication interface to use for connecting with device. Communication interface can correspond to a hardware interface or to a virtual category for the device class. Interfaces' organization depends on used GenTL provider library. **inDeviceId** parameter identifies the requested device in its communication interface. **inDeviceId** can be a unique device id defined by interface module or other alternative device textual address (when used GenTL provider library accepts one). When using alternative device address a single device cannot be opened multiple times using GenTL_OpenDevice function.

You can use GenTL_FindDevices function to quickly obtain library, interface and device identifying parameters. Alternatively you can use GenTL_EnumLibraries and GenTL_EnumLibraryInterfaces functions to discover available GenTL modules structure and pick required library and interface identifiers. When textual ids are not dynamically changed by provider at runtime it is also acceptable to statically define some identifiers of known library-interface-device chain.

For every call to GenTL_OpenDevice a GenTL_CloseHandle function must be called on returned handle.

Return Value

Handle to an opened device. This handle can be used with other GenTL functions to control device state and image streaming, as well as with GenApi functions to access device GenApi parameters set.

Exceptions

This function will throw an exception in the following situations:

- One of device addressing parameters is missing or has invalid format.
- Requested device cannot be found.
- Device connection operation failed.
- Device initial setup failed.
- GenICam device description xml file could not be loaded.
- Other general GenTL or GenICam error occurred.

GenTL_OpenDeviceModuleSettings

Header: Genicam.h
Namespace: avl
Module: Genicam

Opens configuration node set of GenTL device (a device module configuration) of application side transport layer.

Syntax

```
GenTLHandle avl::GenTL_OpenDeviceModuleSettings
(
 GenTLHandle inDeviceHandle
)
```

Parameters

	Name	Type	Default	Description
➡	inDeviceHandle	GenTLHandle		Handle of an opened device which application side module should be accessed.

Description

This function allows to access the connected device's software-side transport layer extended settings through GenAPI interface. The function takes a handle of the opened device (returned by GenTL_OpenDevice) and returns a new handle to its software module settings. You can use GenApi functions to access module settings.

For every call to GenTL_OpenDeviceModuleSettings GenTL_CloseHandle function must be called on returned handle. Device will be closed after all device and device module settings' handles have been closed.

Exceptions

This function will throw an exception in the following situations:

- Device handle is invalid.
- GenICam device description xml file of software module could not be loaded.
- Other general GenTL or GenICam error occurred.

GenTL_OpenDeviceStreamModuleSettings

Header: Genicam.h
Namespace: avl
Module: Genicam

Opens configuration node set of application side transport layer of GenTL device first data stream (a stream module configuration).

Syntax

```
GenTLHandle avl::GenTL_OpenDeviceStreamModuleSettings
(
 GenTLHandle inDeviceHandle
)
```

Parameters

	Name	Type	Default	Description
➡	inDeviceHandle	GenTLHandle		Handle of an opened device which application side stream module should be accessed.

Description

When working with a video transmitter device a separate software module is responsible for managing transmission on its video stream. This function allows to access the connected device's software-side video stream transport layer extended settings through GenAPI interface. The function takes a handle of opened video transmitter device (returned by GenTL_OpenDevice) and returns a new handle to its software stream module settings. You can use GenApi functions to access module settings.

For every call to GenTL_OpenDeviceStreamModuleSettings a GenTL_CloseHandle function must be called on returned handle. Device will be closed after all device and device module settings' handles have been closed.

Exceptions

This function will throw an exception in the following situations:

- Device handle is invalid.
- Device has no stream module or failed to open stream module.
- GenICam device description xml file of software module could not be loaded.
- Other general GenTL or GenICam error occurred.

GenTL_OpenInterfaceModuleSettings

Header: Genicam.h
Namespace: avl
Module: Genicam

Opens configuration node set (an interface module configuration) of GenTL provider communication interface.

Syntax

```
GenTLHandle avl::GenTL_OpenInterfaceModuleSettings
(
 const atl::String& inLibraryPath,
 const atl::String& inInterfaceId
)
```

Parameters

	Name	Type	Default	Description
➡	inLibraryPath	const String&		Path identifying provider library.
➡	inInterfaceId	const String&		String identifying communication interface which should be accessed.

Description

This function allows to access communication interface extended settings through GenAPI system. The function will open specified provider library interface module and return a handle to its instance. You can use GenApi functions to access module settings. These settings affect software-side transport layer of requested interface.

For every call to GenTL_OpenInterfaceModuleSettings a GenTL_CloseHandle function must be called on returned handle.

Exceptions

This function will throw an exception in the following situations:

- Invalid provider library path or interface id is specified.
- GenICam device description xml file of software module could not be loaded.
- Other general GenTL or GenICam error occurred.

GenTL_OpenLibrarySystemModuleSettings

Header: Genicam.h
Namespace: avl
Module: Genicam

Opens configuration node set (a system module configuration) of GenTL provider library.

Syntax

```
GenTLHandle avl::GenTL_OpenLibrarySystemModuleSettings
(
 const atl::String& inLibraryPath
)
```

Parameters

	Name	Type	Default	Description
➡	inLibraryPath	const String&		Path identifying library which system module should be accessed.

Description

This function allows to access library extended settings through GenAPI interface. The function will open specified library system module and return a handle to its instance. You can use GenApi functions to access module settings. These settings affect software-side transport layer across whole provider library.

For every call to GenTL_OpenLibrarySystemModuleSettings a GenTL_CloseHandle function must be called on returned handle.

Exceptions

This function will throw an exception in the following situations:

- Invalid provider library path is specified.
- GenICam device description xml file of software module could not be loaded.
- Other general GenTL or GenICam error occurred.

GenTL_ReceiveImage

Header: Genicam.h
Namespace: avl
Module: Genicam

Receives next frame from video stream.

Syntax

```
void avl::GenTL_ReceiveImage
(
 GenTLHandle inDeviceHandle,
 avl::Image& outImage,
 atl::Optional<atl::uint64&> outFrameId = atl::NIL,
 atl::Optional<atl::uint64&> outTimestamp = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inDeviceHandle	GenTLHandle		Handle of an opened device that is streaming video.
←	outImage	Image&		Image buffer that will receive a new frame.
←	outFrameId	Optional<uint64&>	NIL	Frame block Id set by device (when supported)
←	outTimestamp	Optional<uint64&>	NIL	Frame capture timestamp set by device (when supported)

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outFrameId**, **outTimestamp**.

Read more about Optional Outputs.

Description

This function will return the next available frame from incoming buffer. When input buffer is empty the function will block execution and wait for the next frame to be completed.

The function returns only complete and valid frames, when a frame is corrupted and it cannot be fixed, it is automatically discarded by the library.

A frame is converted from the device stream pixel format to the best suitable format of avl::Image. Automatic conversion includes:

- Conversion from Bayer color mosaic to RGB.
- Conversion of YUV to RGB.
- Ordering of RGB/aRGB channels.
- Unpacking 10, 12 or 14 bpp into 16 bpp images.

Returned image can contain 1, 3 or 4 channels with sint8, uint8 or uint16 pixel type.

outFrameId parameter is optional (can be set to Nil, when not used or given a reference to an output variable). This parameter returns a sequentially incremented number identifying current frame in video sequence. Support of this parameter is optional and for some devices this parameter can be not effective. The value of this output is supplied by the GenTL provider library. Its support, range and wrap around depends on GenTL provider library and transport technology of the device. Refer to device vendor documentation for more information about Buffer FrameId. This value can be used to control correctness of received image sequence and detection of lost frames. Please note that this method can be used only to detect frames that was lost on application side. Detection of frames lost in device and transport layer depends on GenTL provider library implementation and device transport technology.

outTimestamp parameter is optional (can be set to Nil, when not used or given a reference to an output variable). This parameter returns a timestamp of current frame. Support of this parameter is optional and for some devices this output can be not effective. Frequency of the timer used for this timestamp is device and GenTL provider library dependent. Refer to device documentation for details about timestamp timer. Value of this output will wrap around in range from 0 to 2^{31}-1.

Exceptions

This function will throw an exception in the following situations:

- Device handle is invalid.
- Device video stream is not initialized.
- Timeout occurred when waiting for next complete frame.
- Device sends image in pixel format that cannot be converted.
- Device streams in a mode that does not transport images or the library is unable to read the image from stream.

See Also

- GenTL_OpenDevice – Opens a handle to a device provided by GenTL library.
- GenTL_StartAcquisition – Initializes and starts image acquisition in a device.

GenTL_StartAcquisition

Header: Genicam.h
Namespace: avl
Module: Genicam

Initializes and starts image acquisition in a device.

Syntax

```
void avl::GenTL_StartAcquisition
(
 GenTLHandle inDeviceHandle,
 const atl::String& inPixelFormat,
 int inInputQueueSize = 1
)
```

Parameters

	Name	Type	Range	Default	Description
➡	inDeviceHandle	GenTLHandle			Handle of an opened device that should start acquisition.
➡	inPixelFormat	const String&			Pixel format name in GenICam naming convention that will be configured as device stream PixelFormat parameter.
➡	inInputQueueSize	int	1 - 200	1	Number of incoming frames that can be buffered before the application is able to process them.

Description

This function opens device stream, sets up the device to stream with specified pixel format, prepares internal buffers for stream and starts acquisition in device by executing StartAcquisition command.

Set **inPixelFormat** parameter to empty string to leave current device pixel format unchanged.

Use GenTL_GetPixelFormats function to determine device supported pixel formats and their names.

Do not access GenICam StartAcquisition and StopAcquisition commands directly, always use GenTL_StartAcquisition and GenTL_StopAcquisition functions so that the library state can stay in sync with device state.

Exceptions

This function will throw an exception in the following situations:

- Device handle is invalid.
- Device is not a video transmitter.
- Specified pixel format name is not supported by device.
- Device connection is lost.
- Other general GenTL or GenICam error occurred.

See Also

- GenTL_StopAcquisition – Stops image acquisition in a device.

GenTL_StopAcquisition

Header: Genicam.h
Namespace: avl
Module: Genicam

Stops image acquisition in a device.

Syntax

```
void avl::GenTL_StopAcquisition
(
 GenTLHandle inDeviceHandle
)
```

Parameters

	Name	Type	Default	Description
➡	inDeviceHandle	GenTLHandle		Handle of an opened device that should stop acquisition

Description

This function stops acquisition started by GenTL_StartAcquisition. The function will not fail when a device is not streaming video.

Do not access GenICam StartAcquisition and StopAcquisition commands directly, always use GenTL_StartAcquisition and GenTL_StopAcquisition function so that the library state can stay in sync with device state.

Exceptions

This function will throw an exception in the following situations:

- Device handle is invalid.

GenTL_TryReceiveImage

Header: Genicam.h
Namespace: avl
Module: Genicam

Receives next frame from video stream.

Syntax

```
bool avl::GenTL_TryReceiveImage
(
 GenTLHandle inDeviceHandle,
 int inTimeout,
 avl::Image& outImage,
 atl::Optional<atl::uint64&> outFrameId = atl::NIL,
 atl::Optional<atl::uint64&> outTimestamp = atl::NIL
)
```

Parameters

	Name	Type	Default	Description
→	inDeviceHandle	GenTLHandle		Handle of an opened device that is streaming video.
→	inTimeout	int		Maximum time, in milliseconds, that the function is allowed to wait for the next complete frame. This parameter overrides global frame timeout configuration.
←	outImage	Image&		Image buffer that will receive a new frame.
←	outFrameId	Optional<uint64&>	NIL	Frame block Id set by device (when supported)
←	outTimestamp	Optional<uint64&>	NIL	Frame capture timestamp set by device (when supported)

Optional Outputs

The computation of following outputs can be switched off by passing value `atl::NIL` to these parameters: **outFrameId**, **outTimestamp**.

Read more about Optional Outputs.

Description

This function is a close equivalent of GenTL_ReceiveImage, that gives additional control over frame timeout situation.

See GenTL_ReceiveImage definition for more information.

Return Value

`true` when new frame is successfully received and stored into **outImage**, `false` when timeout occurs or device is not streaming.

Exceptions

This function will throw an exception in the following situations:

- Device handle is invalid.
- Connection with device is lost.
- Device sends image in pixel format that cannot be converted.
- Device streams in a mode that does not transport images or the library is unable to read the image from stream.