Zebra
Aurora™ Vision

Aurora Vision Studio 5.3

User Interface

Created: 6/8/2023

Product version: 5.3.4.94078

Table of content:

Complexity Levels

Introduction

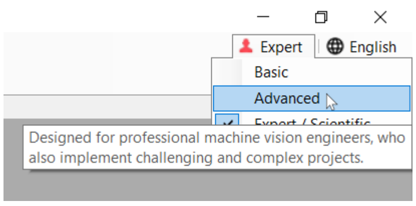
Aurora Vision Studio has three levels of feature complexity. At lower levels we hide advanced features and ones that may be confusing.

Available Levels

- **Basic** – Designed for production engineers who want to quickly build simple machine vision projects without devoting much time to learning the full capabilities of the software.
- **Advanced** – Designed for professional machine vision engineers who also implement challenging and complex projects.
- **Expert / Scientific** – Gives access to experimental and scientific filters (tools), which are not recommended for typical machine vision applications, but might be useful for research purposes.

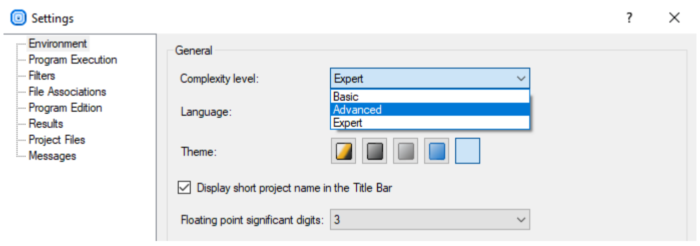
Changing Complexity Level

Complexity Level can be changed at any time. It can be done by clicking on the level name in the upper-right corner of Aurora Vision Studio:



Designed for professional machine vision engineers, who also implement challenging and complex projects.

Complexity Level can also be changed in the application settings:



Complexity level: Expert

Language:

[✓] Display short project name in the Title Bar

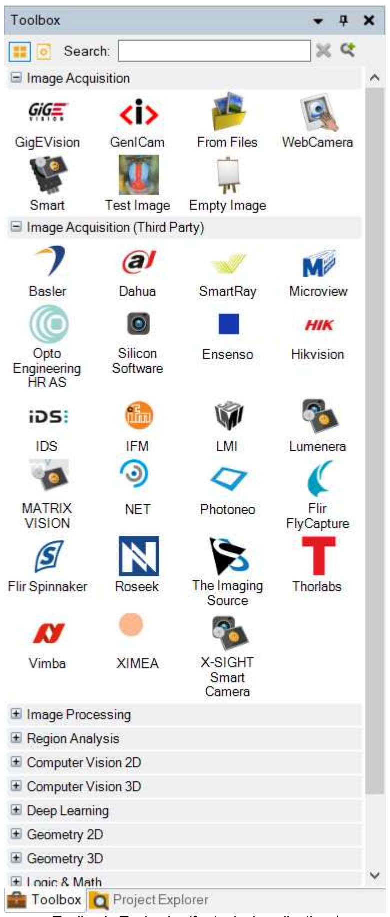
Floating point significant digits: 3

Finding Filters

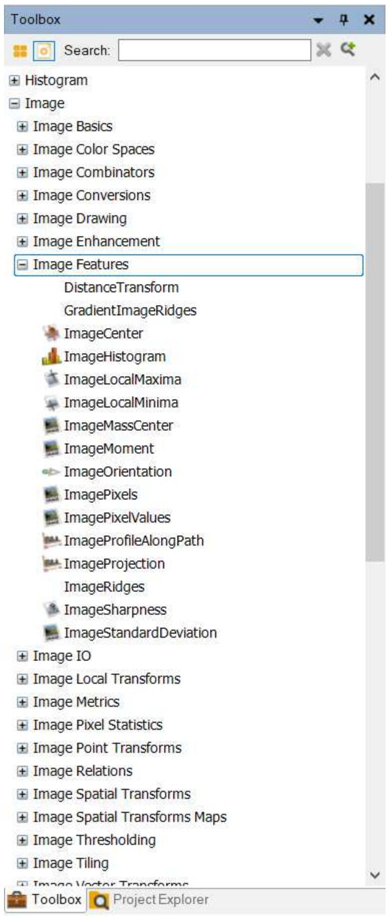
Introduction

There are many hundreds of ready-for-use filters (tools) in Aurora Vision Studio implementing common image processing algorithms, planar geometry, specialized machine vision tools as well as things like basic arithmetics or operating system functions. On the one hand, it means that you get instant access to results of tens of thousands of programmers' work hours, but it also means that there are quite a lot of various libraries and filter categories that you need to know. This article advises you how to cope with that multitude and how to find filters that you need.

The most important thing to know is that there are two different catalogs of filters, designed for different types of users:



Toolbox's Tools view(for typical applications)



Toolbox's Libraries view (for advanced users)

The Toolbox's Tools view is designed for use in typical machine vision applications. It is task-oriented, most filters are grouped into tools and they are supported with intuitive illustrations. This makes it easy to find the filters you need the most. It does not, however, contain all the advanced filters, which might be required in more challenging applications. To access the complete list of filters, you should use the Toolbox's Libraries view. This catalog is organized in libraries, categories and subcategories. Due to its comprehensiveness, it usually takes more time to find what you need, but there is also an advanced text-based search engine, which is very useful if you can guess a part of the filter name.

Toolbox

Sections

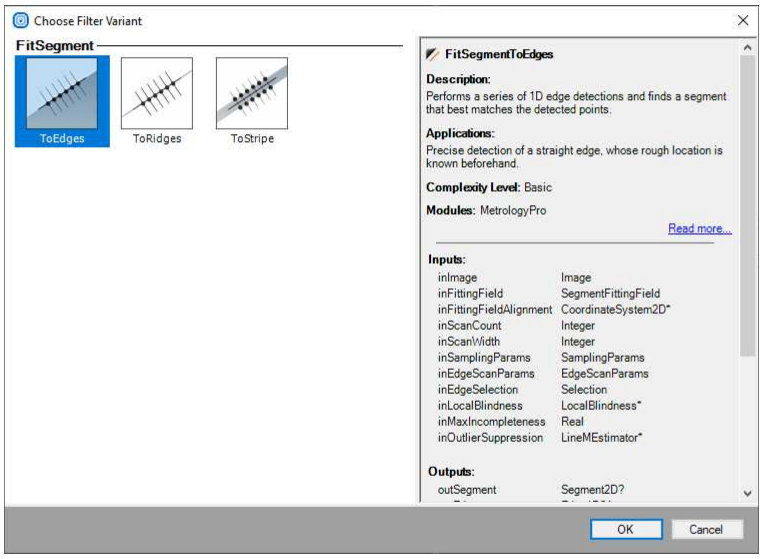
When you use the Toolbox's Tools view, the general idea is that you will most probably start with a filter (tool) from the first section, Image Acquisition, and then follow with filters from consecutive sections:

1. **Image Acquisition**
   – Acquiring images from cameras, frame grabbers or files.
2. **Image Acquisition (Third Party)**
   – Acquiring images from third-party cameras.
3. **Image Processing**
   – Image conversions, enhancements, transformations etc.
4. **Region Analysis**
   – Operations on pixel sets representing foreground objects (blobs).
5. **Computer Vision 2D**
   – Specialized tools for high-level image analysis and measurements.
6. **Computer Vision 3D**
   – Specialized tools for analysis of 3D point clouds.
7. **Deep Learning**
   – Self-learning tools based on deep neural networks.
8. **Geometry 2D**
   – Filters for constructing, transforming and analysing primitives of 2D geometry.
9. **Geometry 3D**
   – Filters for constructing, transforming and analysing primitives of 3D geometry.
10. **Logic & Math**
    – Numerical calculations, arrays and conditional data processing.
11. **Program Structure**
    – Category contains the basic program structure elements.
12. **File System**
    – Filters for working with files on disk.
13. **Program I/O**
    – Filters for communication with external devices.

Choosing Filter from Tools

A tool is a collection of related filters. The process of selecting a filter from the Toolbox thus consists of two steps:

1. First you select a tool in the Toolbox, e.g. "Fit Shape".
2. Then you select a filter from the tool in the "Choose Filter of Group" window.


Choose Filter Variant

FitSegment

ToEdges ToRidges ToStripe

FitSegmentToEdges

Description:
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.

Complexity Level: Basic

Modules: MetrologyPro

Read more...

Inputs:

inImage	Image
inFittingField	SegmentFittingField
inFittingFieldAlignment	CoordinateSystem2D*
inScanCount	Integer
inScanWidth	Integer
inSamplingParams	SamplingParams
inEdgeScanParams	EdgeScanParams
inEdgeSelection	Selection
inLocalBlindness	LocalBlindness*
inMaxIncompleteness	Real
inOutlierSuppression	LineMEstimator*

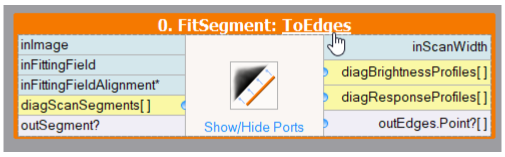
Outputs:

outSegment	Segment2D?

OK Cancel

Choosing a filter from a tool (Toolbox).

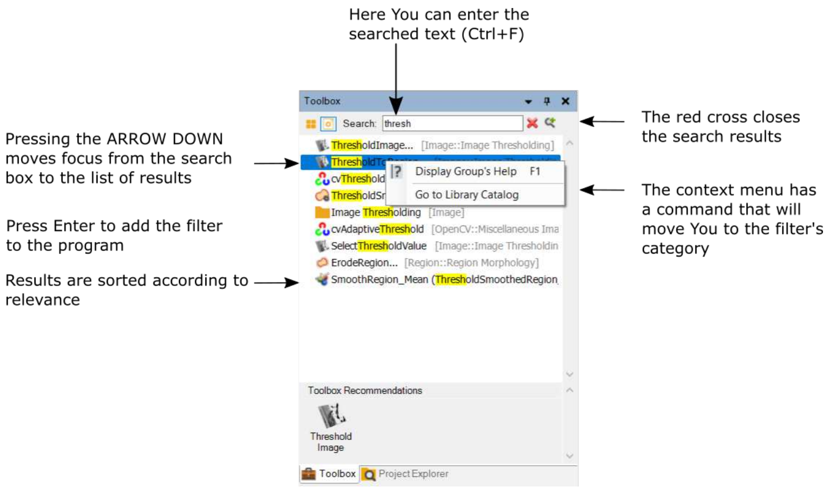
This dialog can have several sections (e.g. "Fit Segment", "Fit Circle" etc.)



Changing filter variant.

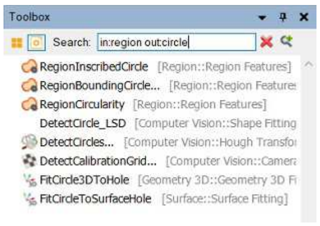
The Search Box

If you know a part of the name of the filter (tool) that you need, or if you can guess it, the Filter Search Box will make your work much more efficient. Just enter the searched text there and you will get a list of filters with most relevant matches at the top:


Here You can enter the searched text (Ctrl+F)
Pressing the ARROW DOWN moves focus from the search box to the list of results
Press Enter to add the filter to the program
Results are sorted according to relevance
The red cross closes the search results
The context menu has a command that will move You to the filter's category
Toolbox
Search: thresh
ThresholdImage... [Image::Image Thresholding]
cvThreshold
ThresholdSr
Display Group's Help F1
Go to Library Catalog
Image Thresholding [Image]
cvAdaptiveThreshold [OpenCV::Miscellaneous Ima
SelectThresholdValue [Image::Image Thresholdin
ErodeRegion... [Region::Region Morphology]
SmoothRegion_Mean (ThresholdSmoothedRegion
Toolbox Recommendations
Threshold Image
Toolbox
Project Explorer

The Search Box and search results in the Toolbox's Libraries view.

If you can not guess the filter name, but you know what you expect on the inputs and outputs, you can use special queries with "in:" and "out:" operators as the image below depicts:


Toolbox
Search: in:region out:circle
RegionInscribedCircle [Region::Region Features]
RegionBoundingCircle... [Region::Region Feature
RegionCircularity [Region::Region Features]
DetectCircle_LSD [Computer Vision::Shape Fitting
DetectCircles... [Computer Vision::Hough Transfo
DetectCalibrationGrid... [Computer Vision::Camer
FitCircle3DToHole [Geometry 3D::Geometry 3D Fi
FitCircleToSurfaceHole [Surface::Surface Fitting]

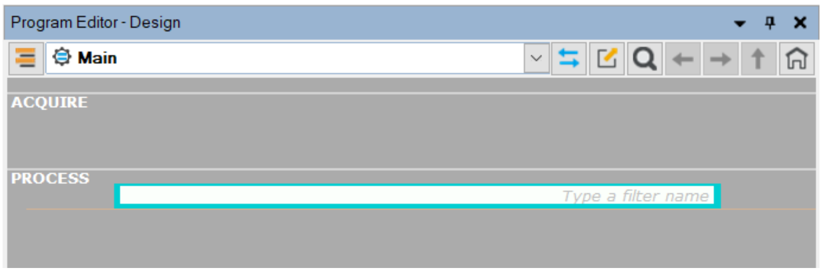
Advanced search with expected inputs and outputs.

As a matter of fact, some advanced users of Aurora Vision Studio stop browsing the categories and just type the filter names in the Search Box to have them quickly added to the program.

Program Editor

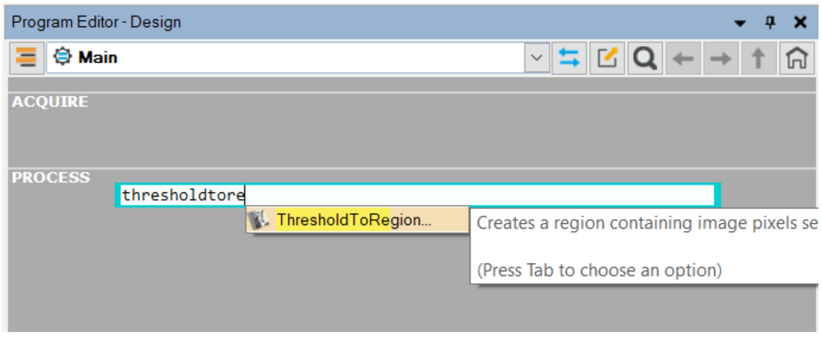
Ctrl+Space / Ctrl+T

When you know the name of a filter, which you would like to add into your program, you can use a keyboard shortcut **Ctrl+Space** or **Ctrl+T** to find it straightaway in the Program Editor, without having to open the Toolbox's Libraries view. After applying this shortcut, you are prompted to type a filter name:



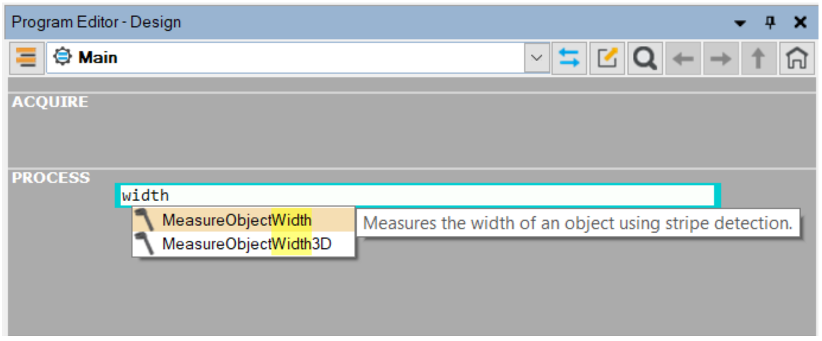
Advanced search using CTRL+SPACE shortcut.

Let us assume that you need to perform simple thresholding on your input image. You already know that there is a ThresholdToRegion filter in our library, which you have utilized many times so far. Instead of time-consuming searching in either Toolbox's Tools view or Libraries view, you can quickly access the desired function by typing its name:


Program Editor - Design
Main
ACQUIRE
PROCESS
thresholdtore
ThresholdToRegion...
Creates a region containing image pixels se
(Press Tab to choose an option)

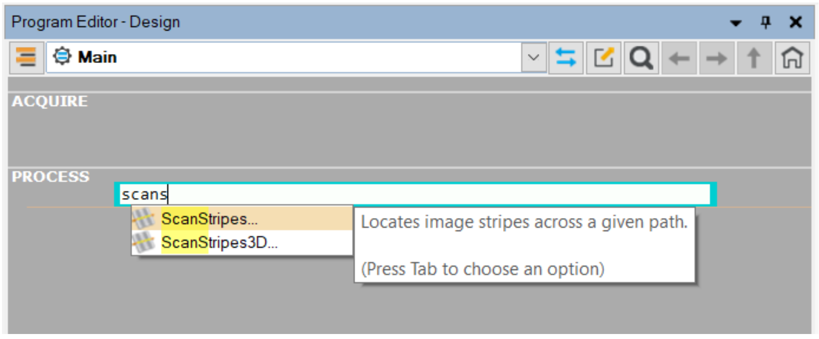
Browsing for a particular filter.

If you do not remember well the name of a filter, you can track it as well by typing only a part of it:



Searching a filter without its full name.

The search engine in the Program Editor also allows you to display the description of a filter in case you want to make sure or remind yourself what the filter is for:

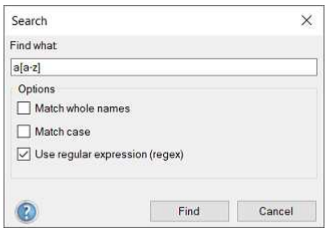


The description of a filter.

Search Window

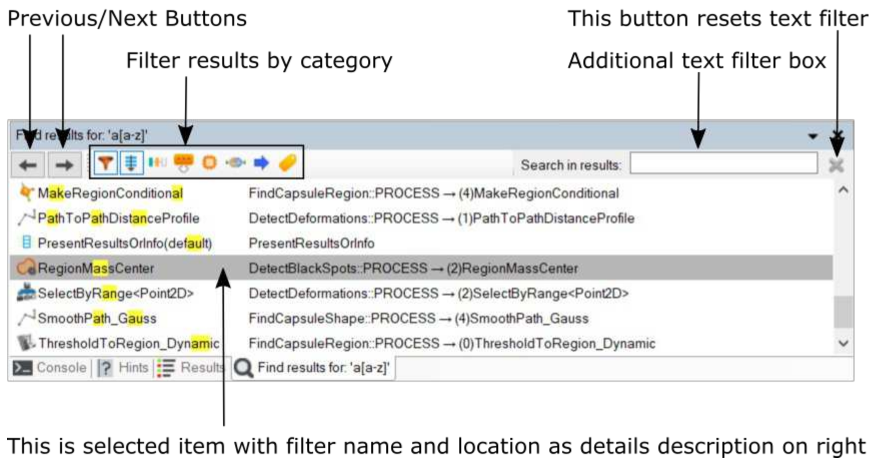
Ctrl+F

Creating a large program in Aurora Vision Studio may require finding elements in it's structure. Using Search Window is the best way to find Filters (tools), Macrofilters, Variants and Global parameters which already exist in the project. To open the search window find a Magnifier button in Program Editor or press Ctrl+F. In the search window insert the name of an element you want to find or a part you remember. Press 'Find' button and results of search will appear in the new window. You may also pick some search options like using case sensitivity, matching whole names or regular expression which can narrow your search.



Browsing for elements in project.

In Search Results window you may select element on the list to indicate an object in project. It is possible to filter results with buttons on panel on the top of the window. Additionally if there are too many results they can be filtered with keyword given in the box on top-right.


Previous/Next Buttons
Filter results by category
This button resets text filter
Additional text filter box
Search in results:
Console
Hints
Results
This is selected item with filter name and location as details description on right

Find results for: 'a[a-z]'

MakeRegionConditional	FindCapsuleRegion::PROCESS → (4)MakeRegionConditional
PathToPathDistanceProfile	DetectDeformations::PROCESS → (1)PathToPathDistanceProfile
PresentResultsOrInfo(default)	PresentResultsOrInfo
RegionMassCenter	DetectBlackSpots::PROCESS → (2)RegionMassCenter
SelectByRange<Point2D>	DetectDeformations::PROCESS → (2)SelectByRange<Point2D>
SmoothPath_Gauss	FindCapsuleShape::PROCESS → (4)SmoothPath_Gauss
ThresholdToRegion_Dynamic	FindCapsuleRegion::PROCESS → (0)ThresholdToRegion_Dynamic

Find results for: 'a[a-z]'

Managing search results.

Rules

Entering a search phrase also allows to pre-filter data with some keywords. When your project is large it might be useful to use Rules in the search phrase. Here are some examples of using Rules in Search Window.

Name	Syntax	Example	Constraint
Parent	`parent:<MacrofilterName>`	`rectangle parent:Initialize`	Must have an instance in Macrofilter <MacrofilterName>.
Input	`in:<Name>`	`load in:File`	Must contain an input which either name or type contains phrase <Name>.
Output	`out:<Name>`	`load out:Integer`	Must contain an output which either name, type or Data Source Label contains phrase <Name>.

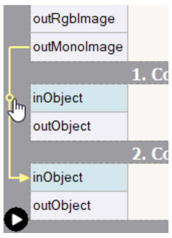
Connecting and Configuring Filters

After a filter is added to the program it has to be configured. This consists in setting its inputs to appropriate constant values in the Properties window or connecting them with outputs of other filters. It is also possible to make an input linked with an external AVDATA file, connected with HMI elements or with Global Parameters.

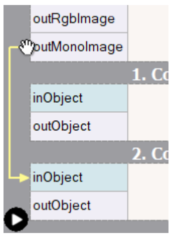
Connecting with Other Filters

Filters receive data from other filters through connections. To create a connection, drag an output of a filter to an input of a filter located below in the Program Editor (upward connections are not allowed). When you drag an output - possible destinations will be highlighted. It is basic way to create program's work-flow.

For convenience, new connections can also be created by forking existing connections (drag from the vertical part of the existing connection):



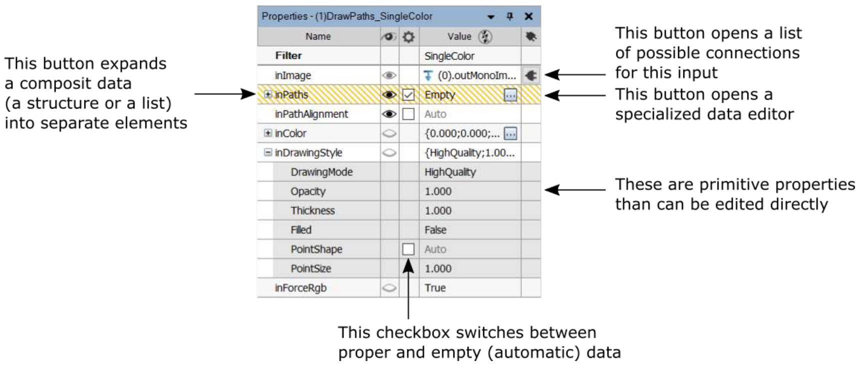
Moreover, it is possible to reconnect an existing connection to another output or input port by dragging the connection's head (near the destination port) or the connection's first horizontal segment (near the source port).



Another way to create connections between filters is by using only the Properties window. When you click the plug ( ) icon in the right-most column, you get a list with possible connections for this input. When the input is already connected the plug icon is solid and you can change the connection to another one.

Setting Basic Properties

Most of the filters have several parameters which you can set in the Properties window as shown on the picture below. It is very important to go through these parameters is order to get desired results for your specific application. To start, first select a filter in the Program Editor window.



This button expands a composit data (a structure or a list) into separate elements
This button opens a list of possible connections for this input
This button opens a specialized data editor
These are primitive properties than can be edited directly
This checkbox switches between proper and empty (automatic) data

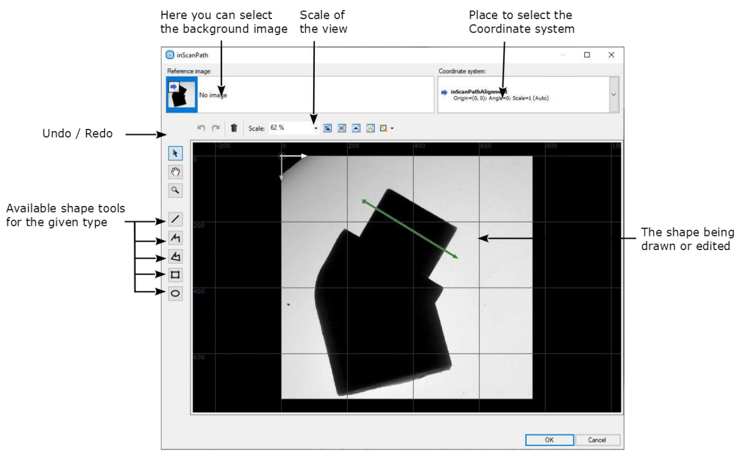
Properties - (1)DrawPaths_SingleColor

Name			Value	
Filter			SingleColor	
inImage			(0).outMonoIm...	
⊞ inPaths		✓	Empty	…
inPathAlignment		☐	Auto	
⊞ inColor			{0.000;0.000;...	…
⊟ inDrawingStyle			{HighQuality;1.00...	
DrawingMode			HighQuality	
Opacity			1.000	
Thickness			1.000	
Filled			False	
PointShape		☐	Auto	
PointSize			1.000	
inForceRgb			True	

Note: After clicking on the header of the properties table it is possible to choose additional columns.

Editing Geometrical Primitives

To edit geometrical data, such as line segments, circle, paths or regions, click the three dots button (…) in the Properties window at the input port you want to set or modify. A window similar to the one below will appear. The first thing you will usually need to do, when you open this window for the first time, is to select the background image from the list at the top of the window. This will set a context for your geometric data.



Here you can select the background image
Scale of the view
Place to select the Coordinate system
Undo / Redo
Available shape tools for the given type
The shape being drawn or edited
inScanPath
Reference image:
No image
Coordinate system:
inScanPathAlignment
Origin=(0, 0); Angle=0; Scale=1 (Auto)
Scale: 62 %
OK
Cancel

Editing of a path in a context of an image.

Tips:

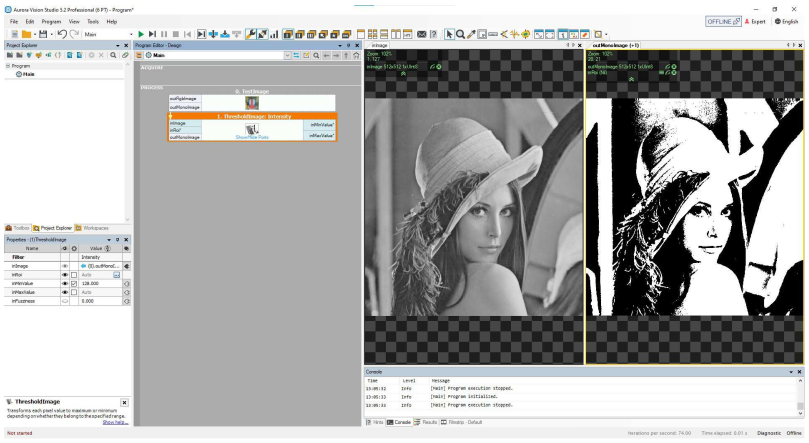
- Select a point and use its context menu to inspect or set the numeric coordinates.
- Use the mouse wheel to zoom the view without changing the tool.
- Hold 3rd mouse button and drag to move the view without changing the tool.
- Hold *Ctrl* to limit the segment angles to the multitudes of 15 degrees.

Testing Parameters in Real Time

One of the greatest features of Aurora Vision Studio is its orientation on rapid development of algorithms. This encompasses the ability to instantly see how different values of parameters affect the results. Due to the dynamic nature of this feature it cannot be presented in a static picture, so please follow the instructions:

1. Create a simple program with a TestImage filter connected to a ThresholdImage filter.
2. Put the output of the ThresholdImage to a data preview.
3. Click *Iterate Current Macro* ▶| to execute the program to the last filter, but without ending it as the whole (the execution state will be: *Paused*).
4. Select the ThresholdImage filter in the Program Editor and change its **inMinValue** input in the Properties window.

Now, you will be able to see in real time how changing the value affects the result.



```
Time        Level   Message
13:05:32    Info    [Main] Program execution stopped.
13:05:33    Info    [Main] Program initialized.
13:05:33    Info    [Main] Program execution stopped.
```

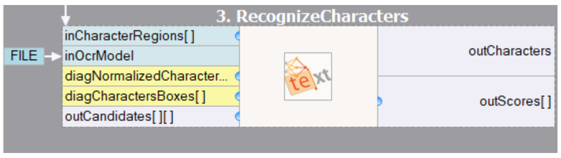
Re-executing a filter after a change of a parameter.

Remarks:

- Please note, that although being extremely useful, this is a "dirty" feature and may sometimes lead to inconsistencies in the program state. In case of problems, stop the program and run it again.
- This feature works only when the filter has already been executed and the program is *Paused*, but NOT *Stopped*.
- By default re-executed is the entire macrofilter. By unchecking the "Global Rerun Mode" setting the re-execution can be limited to a single filter. This can be useful when there are long-lasting computations in the current macrofilter.
- It is not possible to re-execute i/o filters, loop accumulators or loop generators (because this would lead to undesirable side effects). These filters are skipped when the entire macrofilter is getting re-executed.
- When re-executing a nested instance of another macrofilter, the previews of its internal data are NOT updated.
- Re-executing some filters, especially macrofilters, can take much time. Use the Stop command (Shift+F5) to break it, when necessary.
- If you set an improper value and cause a *Domain Error*, the program will stop and it will have to be started again to use the re-execution feature.
- The filter parameters can also be modified during continuous program execution (*F5*).

Linking or Loading Data From a File

Sometimes data values that have to be used on an input of a filter are stored in an .avdata file, that has been created as a result of some Aurora Vision Studio program (for example creating a custom OCR model). It is possible to load such data to the program with the LoadObject filter, but most often it is more convenient and more effective to link the input port directly to the file. To create such a link choose *Link from AVDATA File...* from the context menu of the input (click with the right mouse button on an input port of a filter).

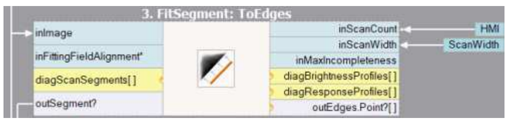


Input data linked from an AVDATA file.

It is also possible to load, not link, data from an .avdata file. This is done with the *Import from AVDATA File...* command in the context menu, which copies the data and makes them part of the current project.

Connecting HMI and Global Parameters

It is also possible to connect filter inputs from outputs of HMI elements and from Global Parameters. Both get displayed as rectangular labels at the sides of the Program Editor as can be seen on the image below:



A filter with connections from HMI and from a Global Parameter.

For further details please refer to the documentation on the specific topics:

- HMI Designer
- Global Parameters

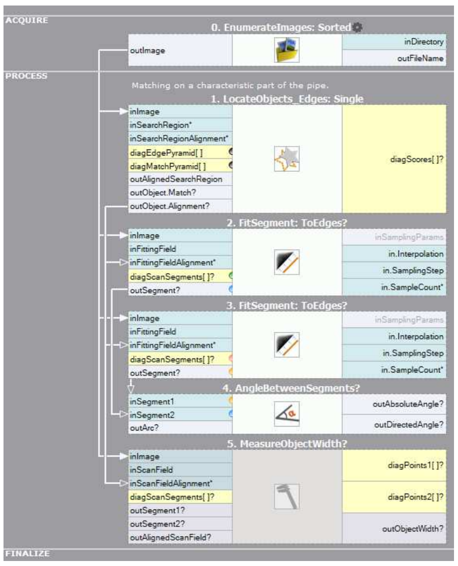
Writable and Readable Global Parameters

It is possible to read and write the value of global parameter by using **ReadParameter** and **WriteParameter** filters. This approach allows users to change global parameter values accordingly to their needs at any point of an algorithm. A few practical applications of this feature are listed below:

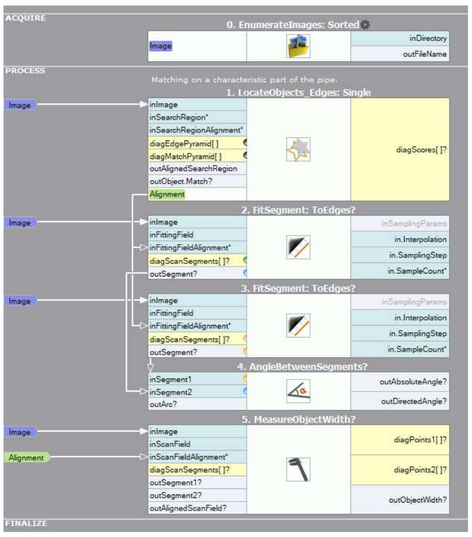
- Managing recipes depending on a signal from PLC,
- Storing data transferred between nested macrofilters,
- Setting global flags.

Labeling Connections

Connections are easier to follow than variables as long as there are not too many of them. When your program becomes complicated, with many intersecting connections, its readability may become reduced. To solve this problem Aurora Vision Studio provides a way to replace some connections with labels. You can right-click on a connection, select "Label All Connections..." - when there is more than one connection or "Label Connection..." when only one connection is present. Then set the name of a label that will be displayed instead. The labels are similar to variables known from textual programming languages – they are named and can be more easily followed if the connected filters are far away from each other. Here is an example:



A program with some long connections becomes not easy to analyze.



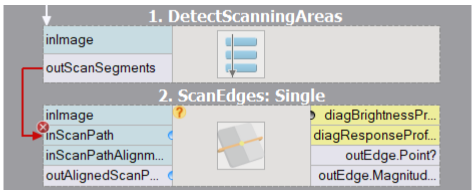
Long connections are replaced with labels for better readability.

Remarks:

- Labeled connections can be brought back (unlabeled) by using the "Un-label This Connection" or "Un-label All Connections" commands available in the context menu of a label.
- Please note, that when your program becomes complicated, the first thing you should consider is reducing its complexity by refactoring the macrofilter hierarchy or rethinking the overall structure. Labeling connections is only a way to visualize the program in a more convenient way and does not make its structure any simpler. It is the user's responsibility to keep it well organized anyway.
- Aurora Vision Studio enforces that all connections between filters are clearly visualized, even if making them implicit would make programming easier in typical machine vision applications. This stems from our design philosophy that assumes that: (1) it is wrong to hide something that the user has to think about anyway, (2) the user should be able to understand all the details of a macrofilter looking at a static screen image.
- When the amount of connections becomes large despite good program structure you should also consider creating User Structures that may be used for merging multiple connections into one. (Do NOT use global parameters for that purpose).

Invalid Connections

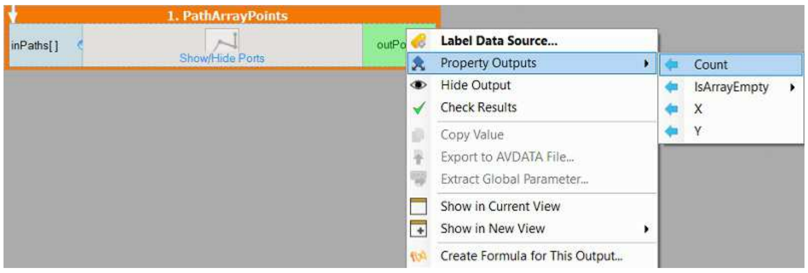
As types of ports in macrofilters and formulas may be changed after connections with other filters have been created, sometimes an existing connection may become invalid. Such an invalid connection will be marked with red line and white cross in the Program Editor:



Invalid connections do not allow to run the program. There are two ways to fix it: replace the invalid connections or revise the types of the connected ports.

Property Outputs

In addition to available filter's outputs, it is possible to get much more information out of a filter. Many data types are represented as structures containing fields, e.g. Point2D consists of "X" and "Y" fields of Real data type. Although these fields are not available as standard outputs, a user can easily add them as additional filter outputs - we call them "Property Outputs". That way, they are directly available for creating connections with inputs of other filters.



Accessing fields of Point2DArray type.

Additional Property Outputs

Some of the properties are calculated based on the current state of the output (for example, checking whether the array is empty or counting its elements), whereas other are derived directly from the internal structure of the type (for example, "X" and "Y" fields composing Point2D structure). The types providing such properties are listed in the table below.

Structure name	Property outputs
Bool	Not
ByteBuffer	Size IsByteBufferEmpty
Histogram	Size
Matrix	IsMatrixEmpty
Path	Points Size Length IsPathEmpty
Profile	Size IsProfileEmpty
Region	Area IsRegionEmpty
Segment2D	Length Direction Center
String	Length IsStringEmpty
Vector2D	Length Direction
Vector3D	Length

There are also special types, which cannot exist independently. They are used for wrapping outputs, which may not be produced under some conditions (Conditional) or optional inputs (Optional), or else for keeping a set of data of specified type (Array). Property outputs of such types are listed in the table below.

Type name	Property outputs
Array	Count IsArrayEmpty
ArrayArray (array of arrays)	Count IsArrayEmpty IsNestedArrayEmpty
Conditional	IsNil
Optional	IsNil

All of the above-mentioned property outputs are specific for the type. However, ports of Array, Optional or Conditional type may have more property outputs, depending on the wrapped type. For example, if the output of a filter is an array of regions, this port will have Count, IsArrayEmpty (both resulting from the array form of the port), IsRegionEmpty and Area (both resulting from the type of the objects held in the array - in this case, regions) as property outputs. However, if the output of a filter is an array of objects without any property outputs (e.g. Integer), only outputs resulting from the array form of the port (Count and IsArrayEmpty) will be available.

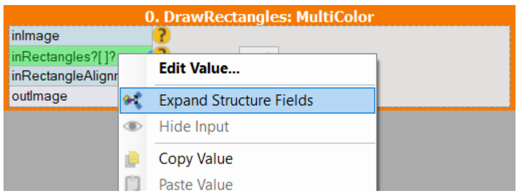
Tip: Avoid using basic filters like Not, ArraySize which will have bigger performance impact than additional property outputs.

What Do *IsNil* and *IsEmpty* Mean:

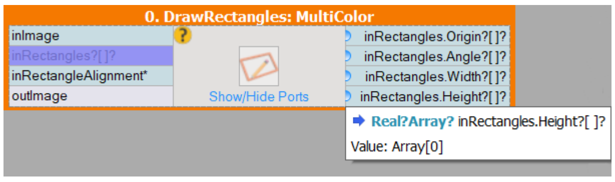
These property outputs return a bool value depending on the content of the data. If the data is *Nil*, what usually means that no object was detected, the property output *IsNil* is set to *True* otherwise it remains *False*. The *IsEmpty* property has a similar use. The only difference is that it reacts if the inputted object is *Empty*. In most common cases one use it for conditional execution of macrofilters or as an input date to formulas. You can read how to deal with *Nil* and *Empty* objects in an article about conditional execution

Expanded Input Structures

Analogously to expanding output properties, it is also possible to expand input structures. This works only for basic structures which allow access to all their fields (e.g. Point2D, but not Image). Please note, that the expanded input structure is replaced by the fields it consists of (unlike by adding property outputs, where the structure itself remains available).



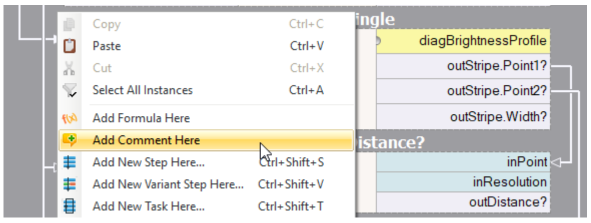
Expanding filter input of Rectangle2D?Array? type.



Result of expanding input of Rectangle2D?Array? type.

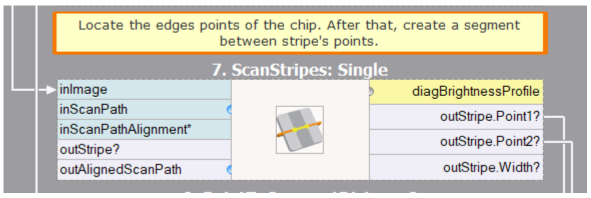
Comment Blocks

Comments are very important part of every computer program. They keep it clear and easy to understand during development and further maintenance. There are two ways to make a comment in Aurora Vision Studio. One way is to add a special comment block. Another option is adding a comment directly in the filter. To add a new comment block to program click the right mouse button on the Program's Editor background and select the "Add Comment Here" option like on the image below.



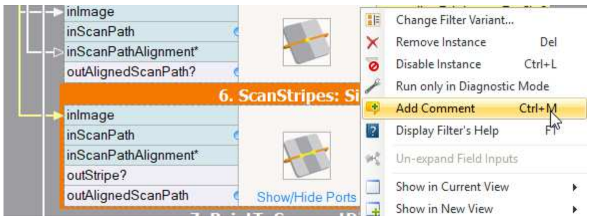
Adding a new comment.

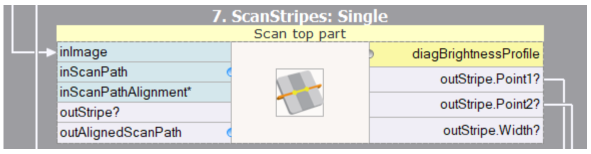
Comment block can be very useful for describing the details of the algorithm.



Comments can be used for describing program steps.

But when you need just a simple tip or short remark, use a "Add Comment" after mouse right click on the filter:





New comment directly in the filter.

Creating Macrofilters

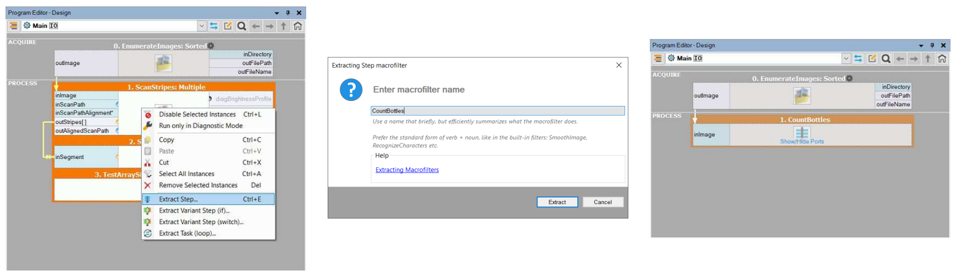
Introduction

Macrofilters play an important role in developing bigger programs. They are subprograms providing a means to isolate sequences of filters and re-use them in other places of the program. After creating a macrofilter you can create its instances. From outside an instance looks like a regular filter, but by double-clicking on it you navigate inside and you can see its internal structure.

When you create a new project, it contains exactly one macrofilter, named "Main". This is the top level macrofilter from which the program execution will start. Further macrofilters can be added by extracting them from existing filters or by creating them in the Project Explorer window.

Extracting Macrofilters (The Quick Way)

The most straightforward way of creating a macrofilter is by extracting it from several filters contained in an existing macrofilter. This is done by selecting several filters in the Program Editor and choosing the *Extract Step...* command from the context menu, as depicted in the picture below:


Program Editor - Design
Main
0. EnumerateImages: Sorted
1. ScanStripes: Multiple
3. TestArraySize
Disable Selected Instances Ctrl+L
Run only in Diagnostic Mode
Copy Ctrl+C
Paste Ctrl+V
Cut Ctrl+X
Select All Instances Ctrl+A
Remove Selected Instances Del
Extract Step... Ctrl+E
Extract Variant Step (if)...
Extract Variant Step (switch)...
Extract Task (loop)...
1. CountBottles
Show/Hide Ports

Extracting Step macrofilter

Enter macrofilter name

CountBottles

Use a name that briefly, but efficiently summarizes what the macrofilter does.

Prefer the standard form of verb + noun, like in the built-in filters: SmoothImage, RecognizeCharacters etc.

Help

Extracting Macrofilters

Extract Cancel

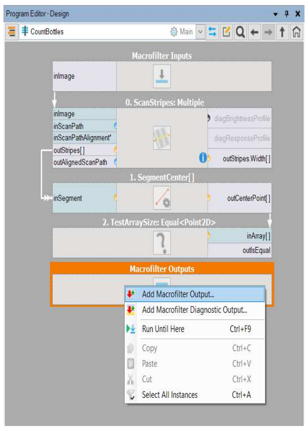
Extracting a macrofilter from three existing filters.

After completing this operation, a new macrofilter definition is created and the previously selected filters are replaced with an instance of this macrofilter. Now, additional inputs and outputs of the new macrofilter can be created by dragging and dropping connections over the new macrofilter instance.

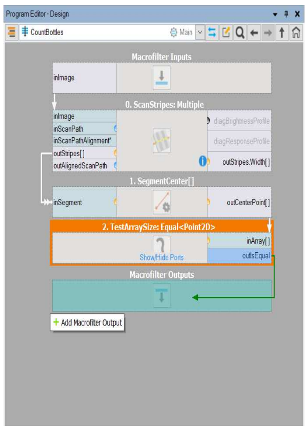
Remark: The "Extract Task (loop)..." command creates a new Task macrofilter which should be used only in special cases. Execution of the Task macrofilter is more complex than execution of the Step macrofilter. Usage of the Task macrofilter for reducing macrofilter complexity may be inappropriate. For more details please read section about Task macrofilters.

Defining the Interface

Being inside of a macrofilter other than "Main" you can see two special blocks: the *Macrofilter Inputs* and the *Macrofilter Outputs*. The context menus of these blocks allow to add new inputs or outputs. Another method of adding them is by dragging connections and dropping them over these blocks.



Adding a new output using the context menu of the Macrofilter Outputs block.



Adding a new output by dragging and dropping a connection.

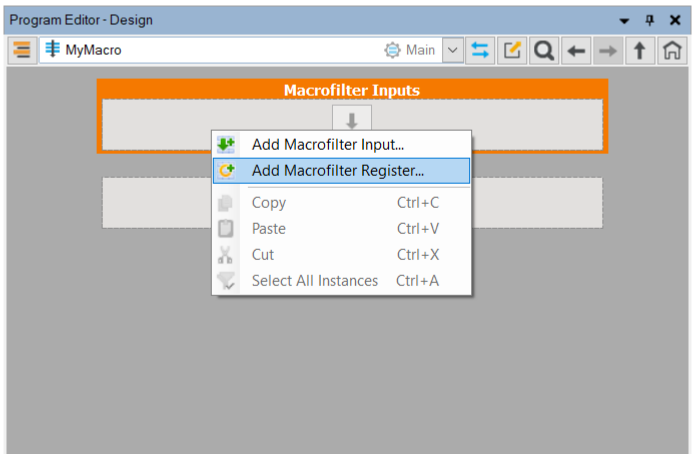
Before the new port is created you need to provide at least its name and type:

Defining a port of a macrofilter.

NOTE: Names of macrofilter inputs and outputs should be clear and meaningful. Names of inputs always start with "in", while names of outputs always start with "out".

Adding Registers

The context menus of macrofilter input and output blocks also contain a command for adding macrofilter registers, *Add Macrofilter Register...*. This is an advanced feature.

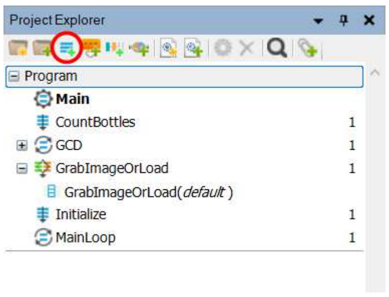


Creating Macrofilters in the Project Explorer

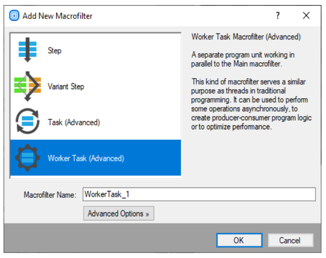
All macrofilter definitions that are contained in the current project can be displayed in the Project Explorer window (by default it is behind the Toolbox on a tab page). Using this control, the user can create or edit macrofilters, but it also acts as a filter catalog from which instances can be created by dragging and dropping the items into the Program Editor.

To create a new macrofilter in the Project Explorer click the *Create New Macrofilter...* button. A new window will appear allowing you to select the name and structure of the new macrofilter.

Please note, that due to their special use *Worker Tasks* can only be created in the Project Explorer. For more information on *Worker Tasks* please refer to Macrofilters article.



Adding Macrofilters in the Project Explorer.



Worker Task Macrofilter (Advanced)

A separate program unit working in parallel to the Main macrofilter.

This kind of macrofilter serves a similar purpose as threads in traditional programming. It can be used to perform some operations asynchronously, to create producer-consumer program logic or to optimize performance.

Macrofilter Name: WorkerTask_1

Selecting the name and the structure of a new macrofilter.

Copying Macrofilters

When copying macrofilters in the Program Editor window, you can only create their new instances: modifying one instance will inevitably affect all the others. In order to duplicate the definitions (create brand-new copies) of old macrofilters as independent entities that you will be able to modify separately, you need to do it inside the Project Explorer window.

Copying macrofilters in the Project Explorer - creating new definitions.

Copying macrofilters in the Program Editor - creating new instances.

Please keep in mind, that regardless of whether you copy macrofilters in the Program Editor or the Project Explorer, with other macrofilters nested inside, no new definitions of nested macrofilters will be created.

1. When copying the parent macrofilter (first nest level) in the Program Editor, the number of instances of nested (child) macrofilters will not change.

2. When copying the parent macrofilter (first nest level) in the Project Explorer, new instances of nested (child) macrofilters will be created.

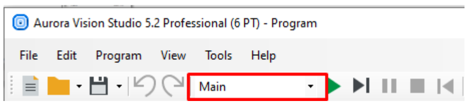
If you want to create a new copy of the whole macrofilter tree (copy of all definitions) you will have to copy each macrofilter separately, starting with the most nested one.

The StartUp Program

It is sometimes useful to have several programs in a single project. The most common scenarios are:

- When some more complex data, e.g. custom OCR models or calibrated reference images, need to be prepared in a separate program.
- When it is convenient to have multiple versions of a single program customized for different installations.
- When the highest reliability is important and sets of automated tests need to be performed on recorded images.

In Aurora Vision Studio a program can be any Worker Task macrofilter. The list of all macrofilters fulfilling this criterion can be seen in the StartUp Macrofilter combo box on the Application Toolbar. This control makes it possible to choose the program that will be run. The related macrofilter will be displayed in the Project Explorer with the bold font.



The StartUp Program Combo Box.

Macrofilter Guidelines

Macrofilters organize big projects, but it is responsibility of the user to make this organization clean and effective. When a project grows, especially under the pressure of time, it is easy to forget this and create programs that are difficult to understand and maintain. To avoid this, please follow the following guidelines:

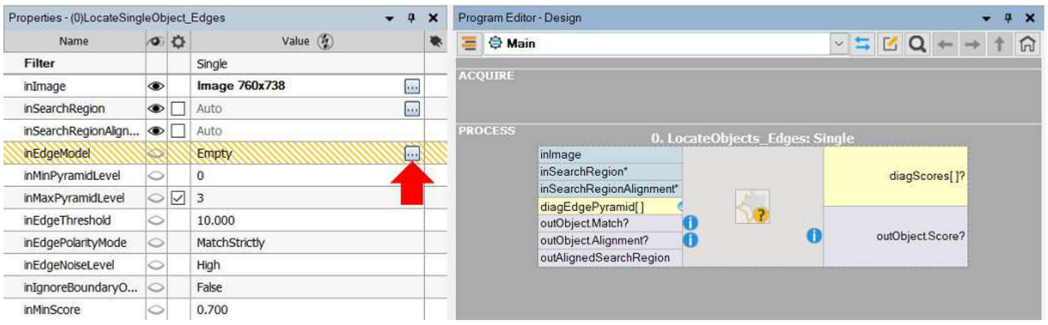
1. Macrofilters should not be too big. For program clarity they should typically consist of 3-15 filters. If you macrofilters have more than 20, it is certainly going to be cause trouble.
2. Each macrofilter should have a clear and single function well reflected in its name. It is recommended to create macrofilters as tools that could possibly be used in many different projects.
3. Macrofilter names do not affect the program execution, but they are very important for effective maintenance. Choose precise and unambiguous English names. Follow the *verb + noun* naming convention whenever possible and avoid abbreviations (this applies also to names of macrofilter ports).
4. Do not mix data analysis with things like communication or visualization in a single macrofilter. These should be separated. If data visualization is needed for HMI, compute results in one macrofilter and then prepare the visualization in another. This will make the data flow more clear.
5. During project development it is very common that the initial program structure becomes inappropriate for what has been added during development, or that it becomes too complicated and unclear. This is when you should consider REFACTORING, i.e. redesigning the program structure, boundaries of macrofilters and revising any complicated sequences of filters. We strongly recommend considering refactoring as a routine part of the job.

Creating Models for Template Matching

Introduction

Template Matching tools are very often used as one of the first steps in industrial inspection applications. The goal is to detect the location of an object. Before this can be done the user has to create a model representing the expected object's shape or structure. To make this step straightforward, Aurora Vision Studio provides an easy user interface. We call it a "GUI for Template Matching".

The GUI for Template Matching is an editor for values of two types: EdgeModel and GrayModel. This means that to open it the user has to select a template matching filter in the program and then click on the button at the **inGrayModel** or **inEdgeModel** input in the Properties window:



Name	Value
Filter	Single
inImage	Image 760x738
inSearchRegion	Auto
inSearchRegionAlign...	Auto
inEdgeModel	Empty
inMinPyramidLevel	0
inMaxPyramidLevel	3
inEdgeThreshold	10.000
inEdgePolarityMode	MatchStrictly
inEdgeNoiseLevel	High
inIgnoreBoundaryO...	False
inMinScore	0.700

Opening GUI for Template Matching.

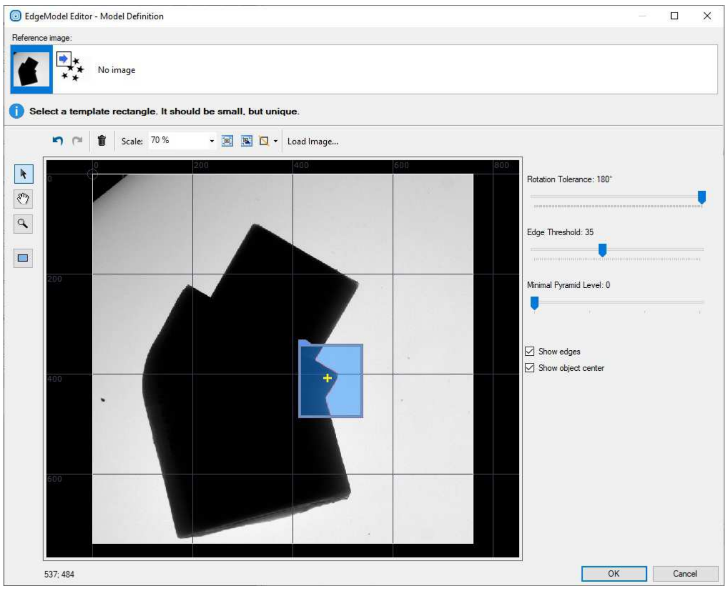
In majority of industrial applications Template Matching algorithms implemented in Aurora Vision Studio easily detect the location of an object using default parameters. However, there are cases where it is needed to tune some of them, mainly in order to achieve higher reliability. It turned out that there is a group of parameters, which is used more often than others. Therefore GUI for Template Matching is available in two variants: Basic and Expert. This is related to Complexity Levels available in Aurora Vision Studio.

Creating a Model

Basic

The Basic window contains the following elements:

1. At the top: a list of possible template images
2. Below: a simple toolbar that also contains a button for loading a template image from a file
3. On the left: a tool for selecting a rectangular template region
4. On the right: track-bars for setting parameters and some options related to the view
5. In the center: an area for selecting the template region in the context of the selected template image



Select a template rectangle. It should be small, but unique.

Rotation Tolerance: 180°

Edge Threshold: 35

Minimal Pyramid Level: 0

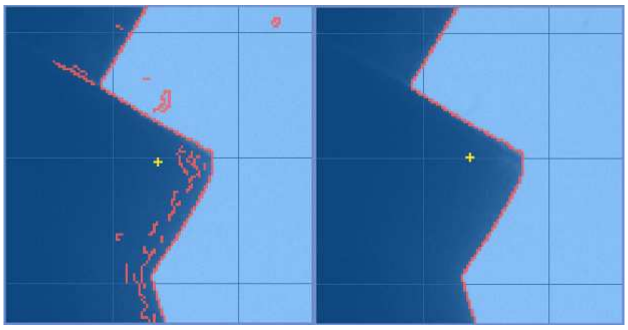
☑ Show edges

☑ Show object center

Basic GUI for Template Matching window

To create a template matching model you need to:

1. Choose a template image from the "Reference image" list.
2. Select a rectangular template region using drawing tool. For maximum performance this rectangle should be as small as possible.
3. Edge-based matching only: Set the **Edge Threshold** parameter, which should be set to value that results in the best quality of the edges. **Edge Threshold** determines the minimum strength (gradient's magnitude) of pixels that can be considered as an edge.



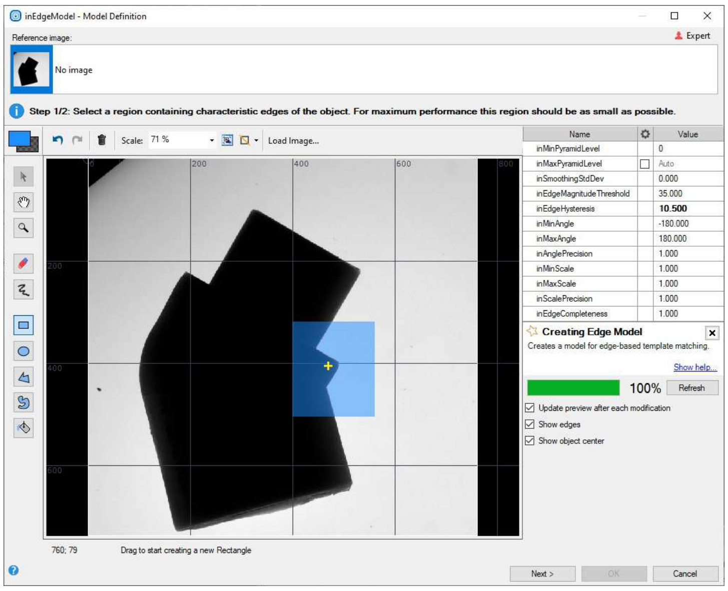
Low quality edges (Edge Threshold = 8) and high quality edges (Edge Threshold = 30)

4. Set the **Rotation Tolerance** in range from 0° to 360°. This parameter determines the maximum expected object rotations. Please note that the smaller the **Rotation Tolerance**, the faster the matching.
5. Set the **Minimal Pyramid Level** parameter which determines the lowest pyramid level used to validate candidates who were found on the higher levels. What is worth mentioning is that setting this parameter to a value greater than 0 may speed up the computation significantly, however, the accuracy of the matching can be reduced. More detailed information about Image Pyramid is provided in Template Matching document in our Machine Vision Guide.

Expert

The Expert window contains the following elements:

1. At the top: a list of possible template images
2. Below: a simple toolbar that also contains a button for loading a template image from a file
3. On the left: a tool for selecting a template region of any shape
4. On the right: parameters of the model, their description and some options related to the view
5. In the center: an area for selecting the template region in the context of the selected template image



Step 1/2: Select a region containing characteristic edges of the object. For maximum performance this region should be as small as possible.

Name		Value
inMinPyramidLevel		0
inMaxPyramidLevel	☐	Auto
inSmoothingStdDev		0.000
inEdgeMagnitudeThreshold		35.000
inEdgeHysteresis		**10.500**
inMinAngle		-180.000
inMaxAngle		180.000
inAnglePrecision		1.000
inMinScale		1.000
inMaxScale		1.000
inScalePrecision		1.000
inEdgeCompleteness		1.000

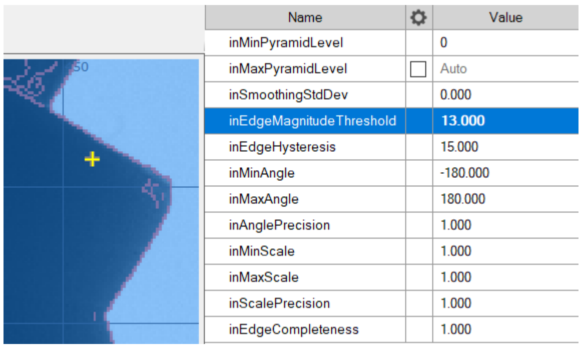
Creating Edge Model
Creates a model for edge-based template matching.
Show help...
100% Refresh
☑ Update preview after each modification
☑ Show edges
☑ Show object center

GUI for Template Matching window

To create a template matching model you need to:

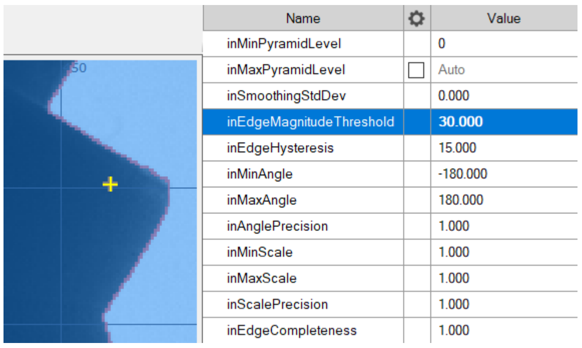
1. Choose a template image from the "Reference image" list.
2. Mark a template region using drawing tools (please note, that the button above the tools switches the current color, i.e. you can both draw or erase shapes). For maximum performance this region should be as small as possible.

3. Edge-base matching only: Set the **inSmoothingStdDev**, **inEdgeMagnitudeThreshold** and **inEdgeHysteresis** parameters to values that result in the best quality of the found edges. It is advisable to first set **inEdgeHysteresis** to zero, then choose a value for **inEdgeMagnitudeThreshold** that assures that all edges have some parts detected and then increase **inEdgeHysteresis**. Small noise might by removed by change the value of parameter **inSmoothingStdDev**. For example:



Name		Value
inMinPyramidLevel		0
inMaxPyramidLevel	☐	Auto
inSmoothingStdDev		0.000
inEdgeMagnitudeThreshold		13.000
inEdgeHysteresis		15.000
inMinAngle		-180.000
inMaxAngle		180.000
inAnglePrecision		1.000
inMinScale		1.000
inMaxScale		1.000
inScalePrecision		1.000
inEdgeCompleteness		1.000

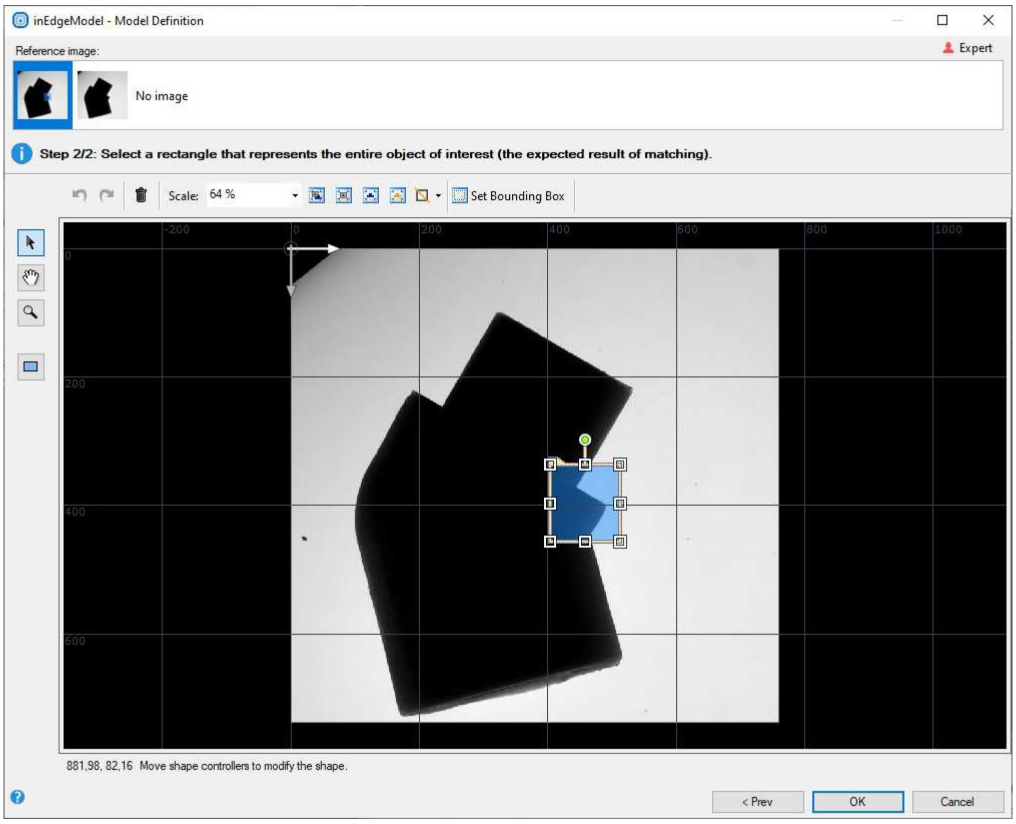
Low quality edges



Name		Value
inMinPyramidLevel		0
inMaxPyramidLevel	☐	Auto
inSmoothingStdDev		0.000
inEdgeMagnitudeThreshold		30.000
inEdgeHysteresis		15.000
inMinAngle		-180.000
inMaxAngle		180.000
inAnglePrecision		1.000
inMinScale		1.000
inMaxScale		1.000
inScalePrecision		1.000
inEdgeCompleteness		1.000

High quality edges

4. Set the **inMinAngle** and **inMaxAngle** parameters accordingly to the expected range of object rotations (the smaller the range, the faster the matching).
5. Set the **inAnglePrecision** to a value lower that 1.0 if you prefer to lower the angular precision for the benefit of speed.
6. Set the **inMinScale**, **inMaxScale** and **inScalePrecision** to appropriate value if you need to detect objects in different scale. You should be aware of longer detection time when you detect objects in scale.
7. Click the "Refresh" button or check "Update preview after each modification" to review the results.
8. Click the "Next >" button to select template rectangle that represents the entire object of interest (the expected result of matching)



inEdgeModel - Model Definition

Reference image:

No image

Step 2/2: Select a rectangle that represents the entire object of interest (the expected result of matching).

Scale: 64 %

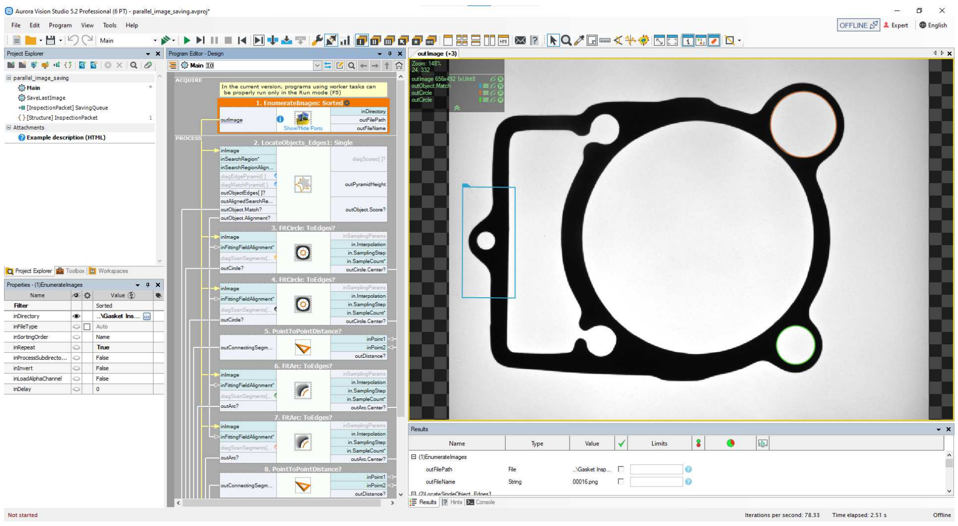
881,98, 82,16 Move shape controllers to modify the shape.

Template rectangle selection

9. Click "OK" to close the window and generate the model.

Performing Template Matching

When the model is ready, performing template matching in an application is straightforward – after connecting the filters and setting the matching parameters, the results are on the outputs of the LocateSingleObject_Edges1 filter (or similar):



Properties - (1)EnumerateImages

Name	Value
Filter	Sorted
inDirectory	..\Gasket Ins...
inFileType	Auto
inSortingOrder	Name
inRepeat	True
inProcessSubdirecto...	False
inInvert	False
inLoadAlphaChannel	False
inDelay	0

Results

Name	Type	Value
(1)EnumerateImages		
outFilePath	File	..\Gasket Insp...
outFileName	String	00016.png

Iterations per second: 78.33 Time elapsed: 2.51 s

Example result of template matching

See also:

Template Matching guide, Template Matching filters

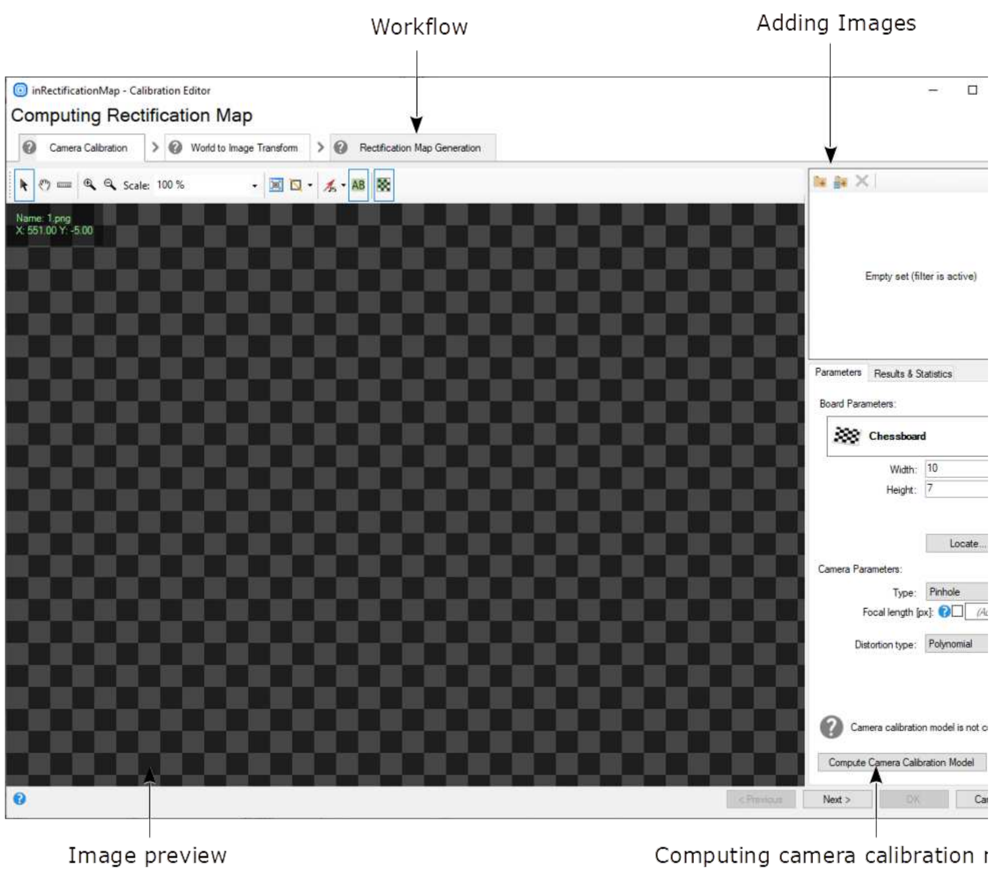
Preparing Rectification Transform Map

Overview

Before you read this article, make sure you are acquainted with Camera Calibration and World Coordinates.

If you want to perform camera calibration in Aurora Vision Studio, there two basic ways to do that. You can either use a calibration editor (plugin) or use filters (manual configuring parameters and connecting outputs with corresponding inputs). Both approaches provide you with the very same results, but the latter way allows you to control intermediate steps and outputs if they are relevant to you for some reason.

This guide will focus mainly on the first approach and show how to perform calibration with the editor step by step. To perform image rectification, the RectifyImage filter should be applied. When you click on the **inRectificationMap** input, the Calibration Editor will be displayed:



The overview of the Calibration Editor.

As you can see the editor consists of three pages:

1. **Camera Calibration** - in which camera lens parameters are computed.
2. **World to Image Transform** - in which perspective and transform converting real-world points to image points are computed.
3. **Rectification Map Generator** - in which fast pixel transform is computed to get a rectified image.

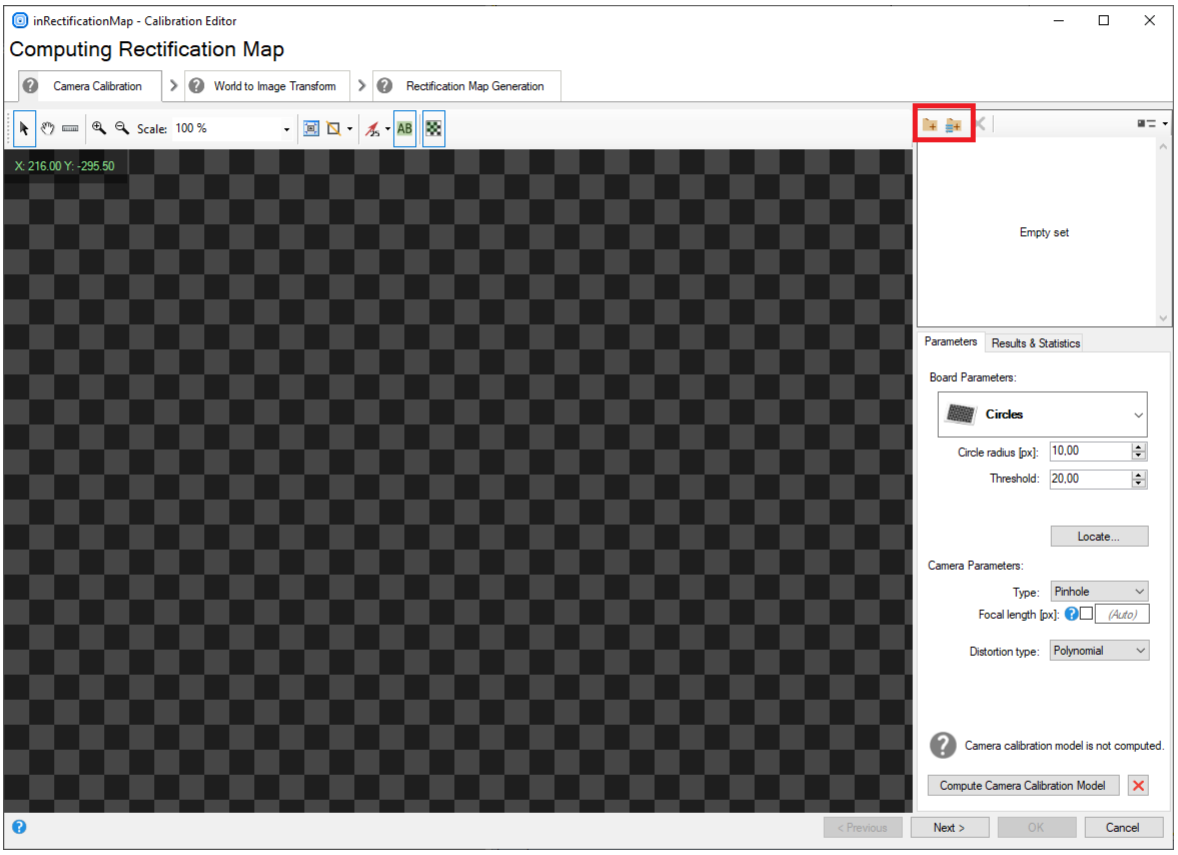
Each of the pages will be individually described, but as it has been already noted, each of the pages in the Calibration Editor invokes corresponding calibration filters. The general overview is in the table below:

Page name	Corresponding filters
Camera Calibration	CalibrateCamera_Pinhole CalibrateCamera_Telecentric
World to Image Transform	CalibrateWorldPlane_Default CalibrateWorldPlane_Labeled CalibrateWorldPlane_Manual CalibrateWorldPlane_Multigrid
Rectification Map Generation	CreateRectificationMap_Advanced CreateRectificationMap_PixelUnits CreateRectificationMap_WorldUnits

Camera Calibration Page

On the first page of the Calibration Editor you have to provide images of calibration boards, using the buttons marked in red.

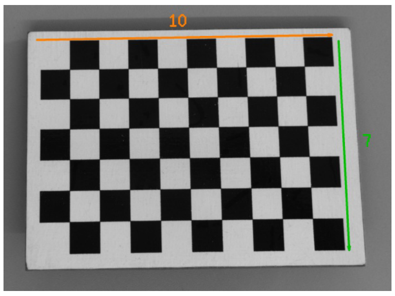
The purpose of this stage is to estimate intrinsic camera parameters. They do not depend on the transformation between the camera and its external world, therefore this step can be done before mounting the camera at the target place.


inRectificationMap - Calibration Editor
Computing Rectification Map
Camera Calibration
World to Image Transform
Rectification Map Generation
Scale: 100 %
X: 216.00 Y: -295.50
Empty set
Parameters
Results & Statistics
Board Parameters:
Circles
Circle radius [px]: 10.00
Threshold: 20.00
Locate...
Camera Parameters:
Type: Pinhole
Focal length [px]: (Auto)
Distortion type: Polynomial
Camera calibration model is not computed.
Compute Camera Calibration Model
< Previous
Next >
OK
Cancel

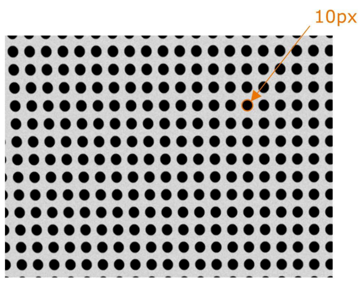
Camera Calibration Page - adding images.

Next, you need to specify board and camera parameters. There are two possible options to choose from, depending on what kind of pattern is used:

- **Chessboard** - where you have to define two parameters of the board: width and height, which correspond to the number of squares in horizontal and vertical directions, respectively.
- **Circles** - where you have to define a single circle's radius and threshold value.



If you are using chessboard pattern, you have to count squares in both dimensions.
In this example, the width is 10 and the height is 7.

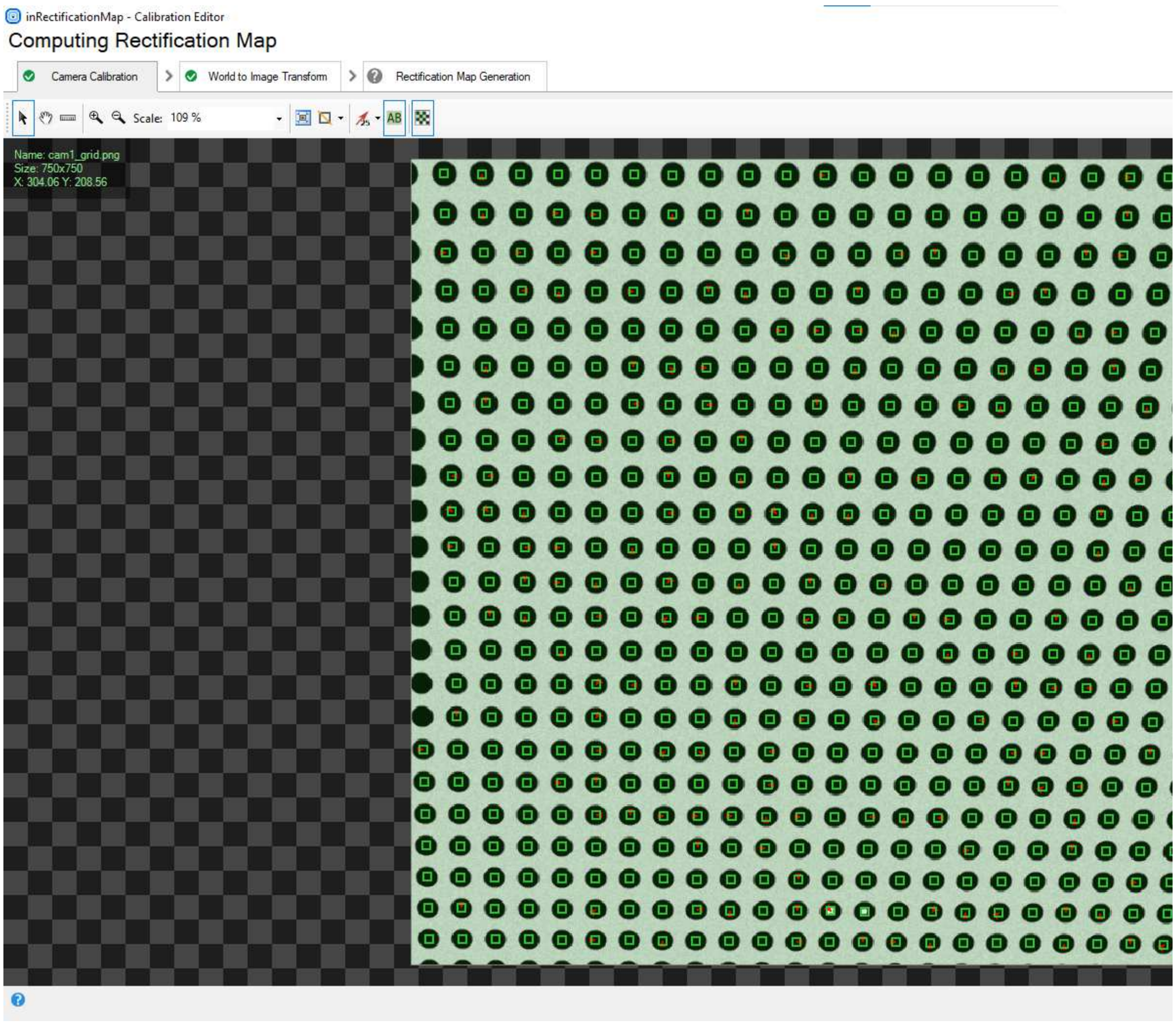


If you are using circle pattern, you have to measure radius of any circle.
In this example, it is about 10px. Note: it is important to use a symmetric board as shown in the image. Asymmetric boards are currently not supported.

Once they are set, you should adjust camera type according to an applied camera, which can be either **pinhole** (which uses a standard perspective projection) or **telecentric** (uses an orthographic projection).

A few distortion model types are supported. The simplest - divisional - supports most use cases and has predictable behavior 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 px. Of course this is only a rule of thumb, as each lens is different and there are exceptions.

Results & Statistics tab informs you about results of the calibration, especially about reprojection error, which corresponds with reprojection vectors shown in red on the preview in the picture below:

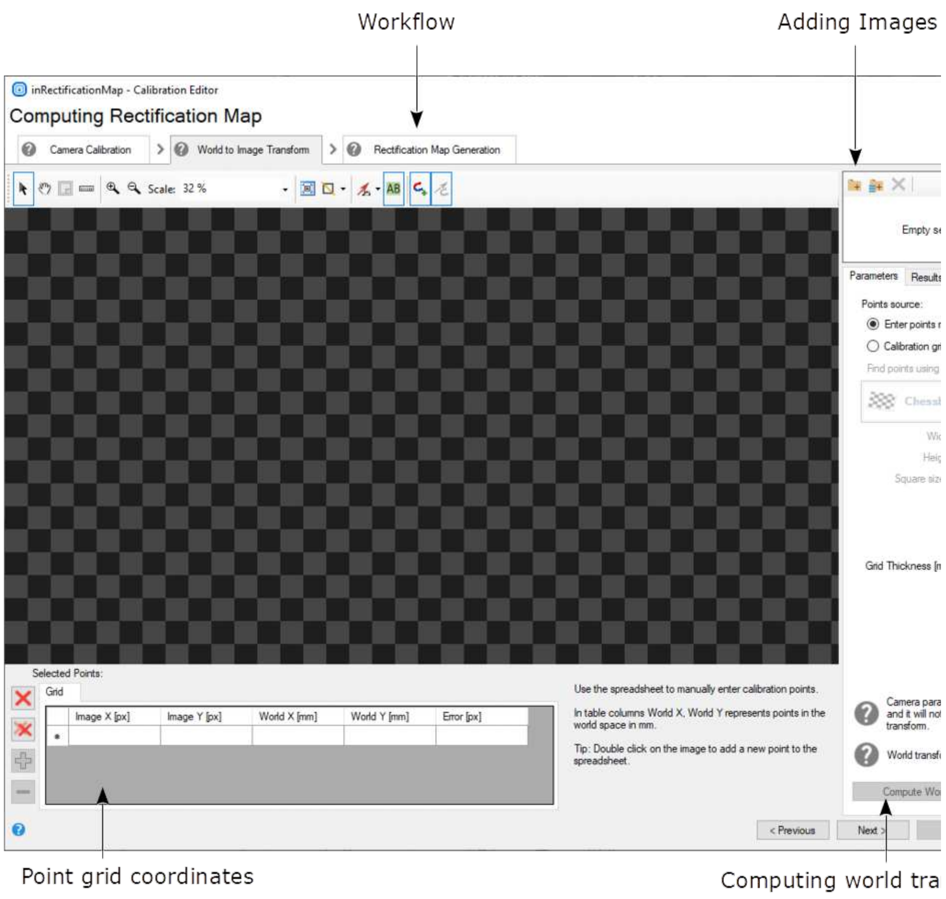


They indicate the direction towards each located point should be moved in order to deliver an immaculate grid. Additionally, there are values of Root Mean Square and Maximal error. The tab also displays value and standard deviation for each computed camera and lens distortion parameter.

For more details, please refer to the corresponding group of filters for this page in the table above.

World to Image Transform Page

Points added to the second page are used to find transformation between real-world and image points.



Computing Rectification Map

	Image X [px]	Image Y [px]	World X [mm]	World Y [mm]	Error [px]
*					

Use the spreadsheet to manually enter calibration points.

In table columns World X, World Y represents points in the world space in mm.

Tip: Double click on the image to add a new point to the spreadsheet.

World to Image Transform - overview.

First of all, in this step, you need to add images of the calibration board as well (using the very same buttons as in the previous step). Please note that in this step they have to be taken from a fixed angle in contrast to the previous step, where all of images could be taken from different angles.

In the next step you should indicate points source in either of two ways:

- By **entering points manually**
- By using **calibration grid**

Points can be found automatically using the calibration board as in the previous step.

After successful points location on the calibration board, you should use the spreadsheet to manually enter coordinates in world plane:

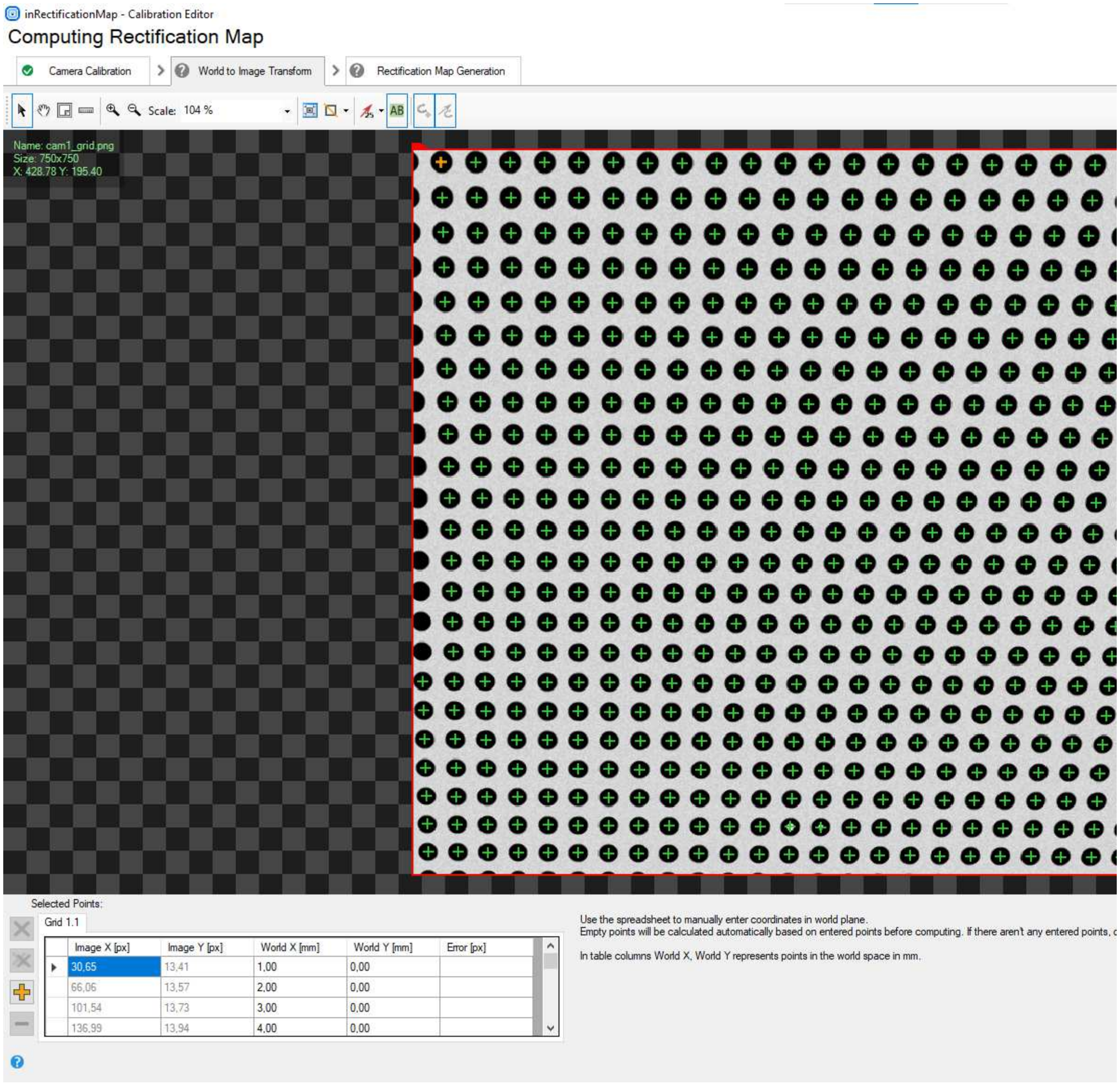


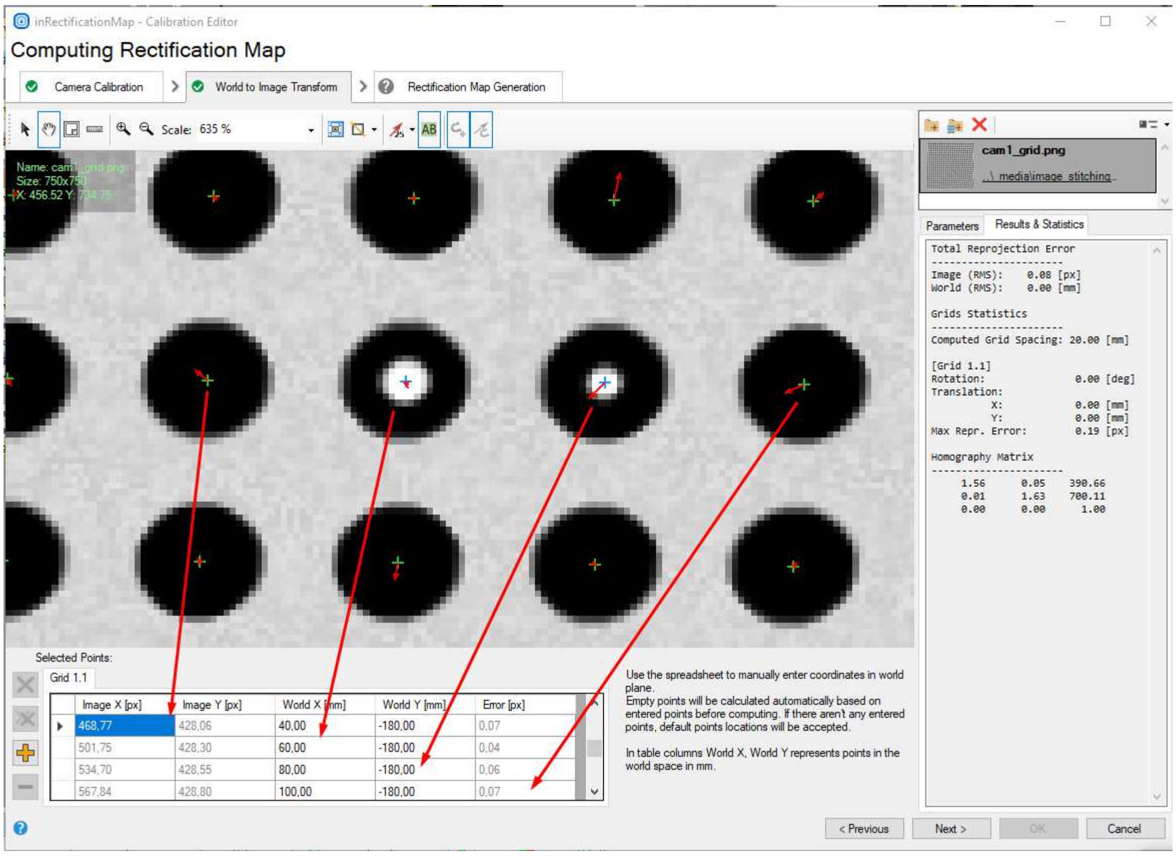
Image X [px]	Image Y [px]	World X [mm]	World Y [mm]	Error [px]
30,65	13,41	1,00	0,00	
66,06	13,57	2,00	0,00	
101,54	13,73	3,00	0,00	
136,99	13,94	4,00	0,00	

Use the spreadsheet to manually enter coordinates in world plane.
Empty points will be calculated automatically based on entered points before computing. If there aren't any entered points, c

In table columns World X, World Y represents points in the world space in mm.

World to Image Transform - modifying points in the spreadsheet.

Points with unspecified world coordinates will have them calculated automatically based on points with specified coordinates. If there are no world coordinates entered, then algorithm will assume some default world point locations, what might produce false results. At least two grid point world coordinates are needed to uniquely determine the world plane position, rotation and spacing.



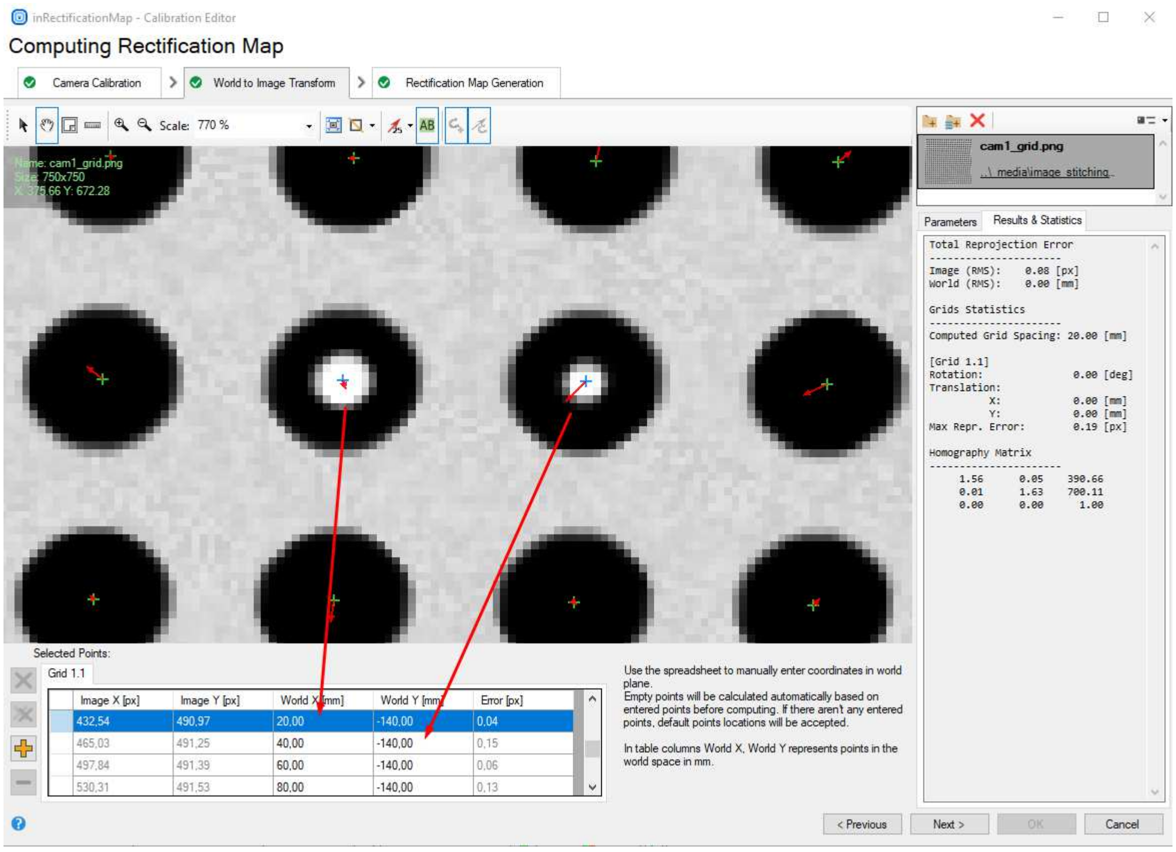
```
Total Reprojection Error
------------------------
Image (RMS):    0.08 [px]
World (RMS):    0.00 [mm]

Grids Statistics
------------------------
Computed Grid Spacing: 20.00 [mm]

[Grid 1.1]
Rotation:                 0.00 [deg]
Translation:
          X:              0.00 [mm]
          Y:              0.00 [mm]
Max Repr. Error:          0.19 [px]

Homography Matrix
------------------------
     1.56      0.05    390.66
     0.01      1.63    700.11
     0.00      0.00      1.00
```

Image X [px]	Image Y [px]	World X [mm]	World Y [mm]	Error [px]
468,77	428,06	40,00	-180,00	0,07
501,75	428,30	60,00	-180,00	0,04
534,70	428,55	80,00	-180,00	0,06
567,84	428,80	100,00	-180,00	0,07

Use the spreadsheet to manually enter coordinates in world plane.
Empty points will be calculated automatically based on entered points before computing. If there aren't any entered points, default points locations will be accepted.

In table columns World X, World Y represents points in the world space in mm.

World to Image Transform - automatic grid spacing assumed.



```
Total Reprojection Error
------------------------
Image (RMS):    0.08 [px]
World (RMS):    0.00 [mm]

Grids Statistics
------------------------
Computed Grid Spacing: 20.00 [mm]

[Grid 1.1]
Rotation:                 0.00 [deg]
Translation:
         X:               0.00 [mm]
         Y:               0.00 [mm]
Max Repr. Error:          0.19 [px]

Homography Matrix
------------------------
     1.56      0.05    390.66
     0.01      1.63    700.11
     0.00      0.00      1.00
```

Image X [px]	Image Y [px]	World X [mm]	World Y [mm]	Error [px]
432,54	490,97	20,00	-140,00	0,04
465,03	491,25	40,00	-140,00	0,15
497,84	491,39	60,00	-140,00	0,06
530,31	491,53	80,00	-140,00	0,13

Use the spreadsheet to manually enter coordinates in world plane.
Empty points will be calculated automatically based on entered points before computing. If there aren't any entered points, default points locations will be accepted.

In table columns World X, World Y represents points in the world space in mm.

World to Image Transform - grid spacing calculated from world point coordinates given by the user.

Arrows indicate which points from the calibration grid relate to corresponding rows in the spreadsheet. As you can see each row consists of coordinates in the image plane (given in **pixels**), coordinates in the world plane (given in **mm**), and error (in **pixels**), which means how much a point is deviated from its model location. In this case reprojection vectors, which are marked as small, red arrows, also indicate the deviation from the model.

Colors of points have their own meaning:

- Green Point - the point has been computed automatically.
- Orange Point - the point which has been selected.
- Blue Point - the point has been adjusted manually.

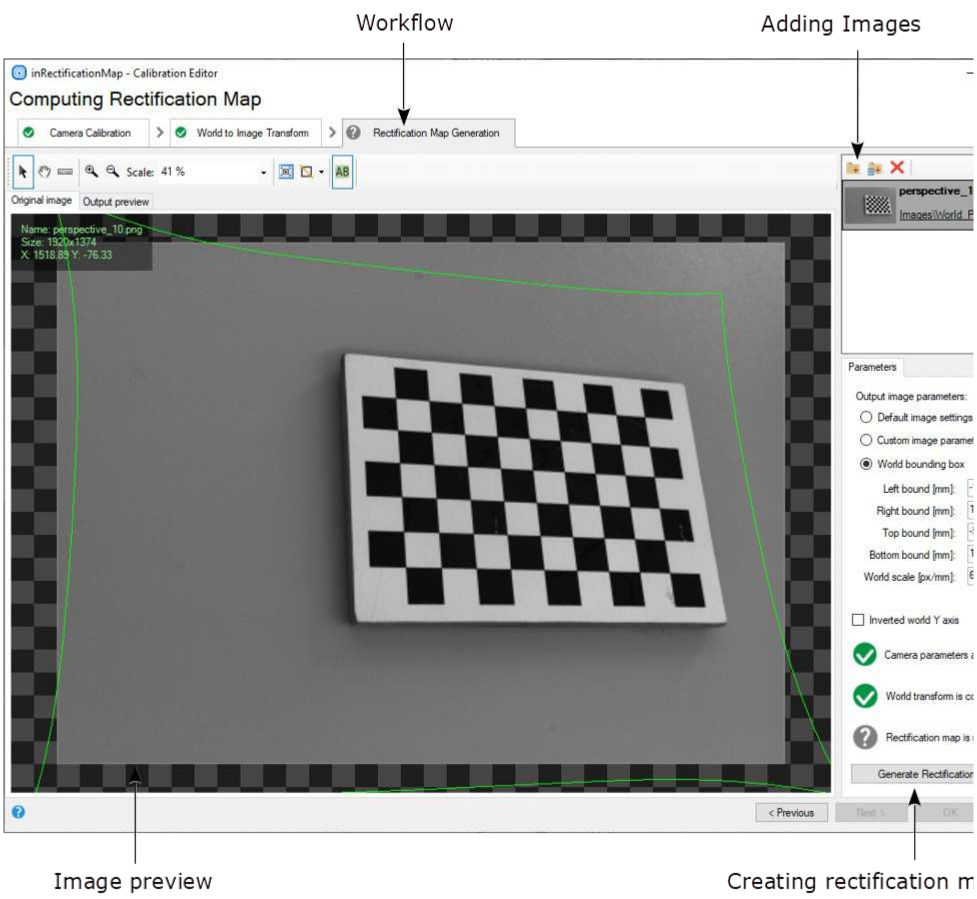
The *Results & Statistics* tab shows information about computed errors for both image and world coordinates. The output reprojection errors are useful tool for assessing the feasibility of computed solution. There are two errors in the plugin: Image (RMS) and World (RMS). The first one denotes how inaccurate the evaluation of perspective is. The latter reflects inaccuracy of labeling of grid 2D world coordinate system. They are independent, however they both influence quality of the solution, so their values should remain low.

For the details, please refer to the corresponding group of filters for this page in the table above.

Rectification Map Generator Page

Last page is used to set parameters of an image after the rectification.

First of all, you need to load images of the calibration board in the same way as in previous steps.



Rectification Map Generation - rectifying the image.

Next, you have to choose one of three basic options of the output image:

- **Default image settings**
- **Custom image parameters**
- **World bounding box**

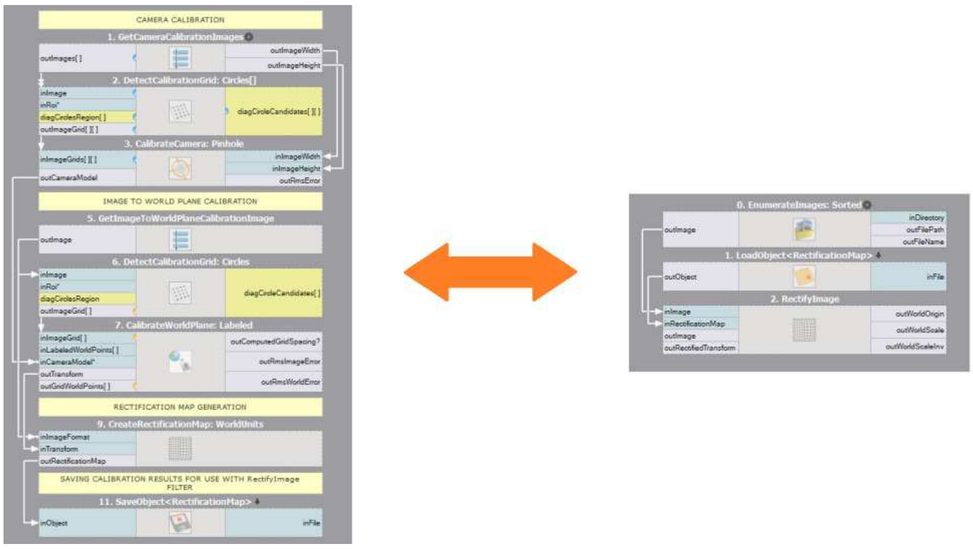
The green frame on the preview informs you about the size and borders of the output image, and depending on what option you have chosen, you can set different parameters. Using *custom parameters* or *world bounding box* will provide you with the same rectification map, but what makes the difference is that you are working in different domains - in the first case you are operating in the image coordinates (given in pixels), whereas in the other one you are operating in the world coordinates (given in millimeters).

When you are done, you can click on the **Generate Rectification Map** button and assess the rectified image displayed on the preview.

For the details, please refer to the corresponding group of filters for this page in the table above.

Relation between the Calibration Editor and filters

The relation between both approaches could be presented in a form of the below graphics:


CAMERA CALIBRATION
1. GetCameraCalibrationImages
2. DetectCalibrationGrid: Circles[]
3. CalibrateCamera: Pinhole
IMAGE TO WORLD PLANE CALIBRATION
5. GetImageToWorldPlaneCalibrationImage
6. DetectCalibrationGrid: Circles
7. CalibrateWorldPlane: Labeled
RECTIFICATION MAP GENERATION
9. CreateRectificationMap: WorldUnits
SAVING CALIBRATION RESULTS FOR USE WITH RectifyImage FILTER
11. SaveObject<RectificationMap>
0. EnumerateImages: Sorted
1. LoadObject<RectificationMap>
2. RectifyImage

The left side presents which filters are necessary to generate a rectification map and how to save it using SaveObject. The right side presents how to load the rectification map using LoadObject and passing it to the RectifyImage filter.

This is just an exemplary set of filters which might be applied, but using specific filters depend on the calibration board and other parameters relevant to a case.

Further readings

- Calibration-related list of filters in Aurora Vision Studio

Creating Text Segmentation Models

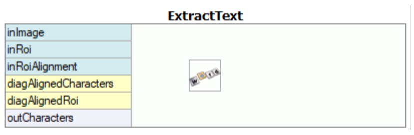
The graphical editor for text segmentation performs two operations:

1. **Thresholding** an image with one of several different methods to get a single foreground region corresponding to all characters.
2. **Splitting** the foreground region into an array of regions corresponding to individual characters.

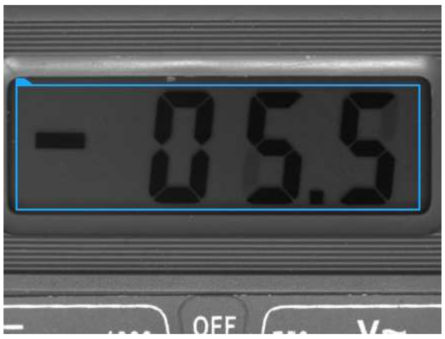
Details about using OCR filters can be found in Machine Vision Guide: Optical Character Recognition.

To configure text extraction please perform the following steps:

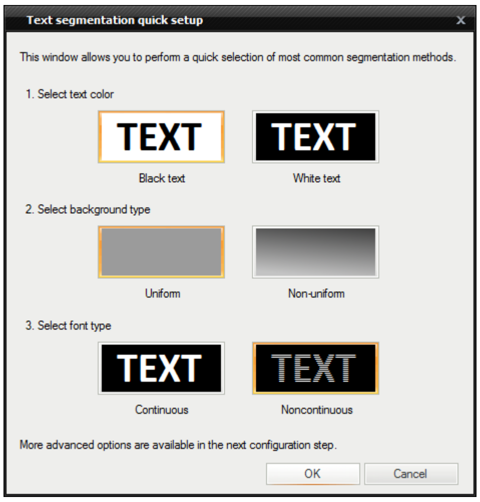
1. Add an ExtractText filter to the program.



2. Set the region of interest on the inRoi input. This step is necessary before performing next steps. The image below shows how the ROI was selected in an example application:



3. Click on the "..." button at the **inSegmentationModel** input to enter the graphical editor.
4. When entering first time, complete the quick setup by selecting most common settings. In this example a black non-continuous text should be extracted from a uniform background. Configuration was set to meet these requirements.



Text segmentation quick setup

This window allows you to perform a quick selection of most common segmentation methods.

1. Select text color

[✓ Black text] White text

2. Select background type

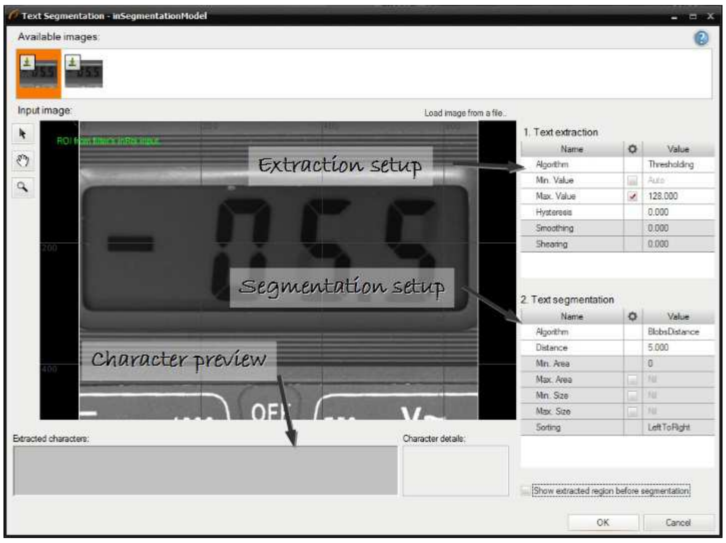
[✓ Uniform] Non-uniform

3. Select font type

Continuous [✓ Noncontinuous]

More advanced options are available in the next configuration step.

5. After the quick setup the graphical editor starts with some parameter set. Adjust the pre-configured parameters to get best results.



Text Segmentation - inSegmentationModel

Available images:

Input image:

Load image from a file..

ROI from filter's inRoi input

1. Text extraction

Name	⚙	Value
Algorithm		Thresholding
Min. Value	☐	Auto
Max. Value	✓	128.000
Hysteresis		0.000
Smoothing		0.000
Shearing		0.000

2. Text segmentation

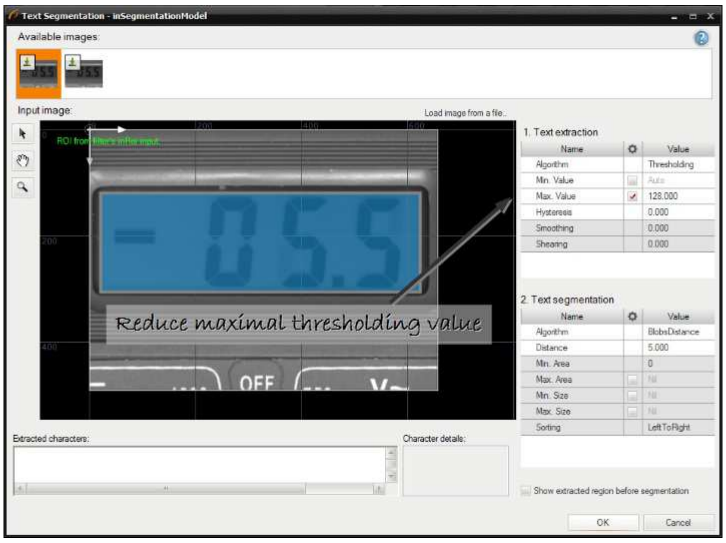
Name	⚙	Value
Algorithm		BlobsDistance
Distance		5.000
Min. Area		0
Max. Area	☐	Nil
Min. Size	☐	Nil
Max. Size	☐	Nil
Sorting		LeftToRight

☐ Show extracted region before segmentation

Extracted characters:

Character details:

6. Configure a character extraction algorithm. In this case thresholding value is too high and must be reduced.


Text Segmentation - inSegmentationModel
Available images:
Input image:
Load image from a file..
ROI from filter's inRoi input
Reduce maximal thresholding value
Extracted characters:
Character details:
OK
Cancel

1. Text extraction

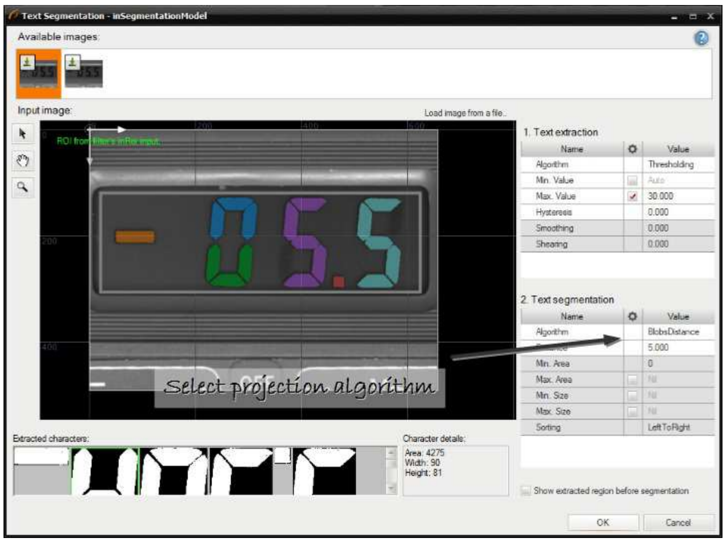
Name	⚙	Value
Algorithm		Thresholding
Min. Value	☐	Auto
Max. Value	☑	128.000
Hysteresis		0.000
Smoothing		0.000
Shearing		0.000

2. Text segmentation

Name	⚙	Value
Algorithm		BlobsDistance
Distance		5.000
Min. Area		0
Max. Area	☐	Nil
Min. Size	☐	Nil
Max. Size	☐	Nil
Sorting		LeftToRight

☐ Show extracted region before segmentation

7. Select a character segmentation algorithm.



1. Text extraction

Name	⚙	Value
Algorithm		Thresholding
Min. Value	☐	Auto
Max. Value	☑	30.000
Hysteresis		0.000
Smoothing		0.000
Shearing		0.000

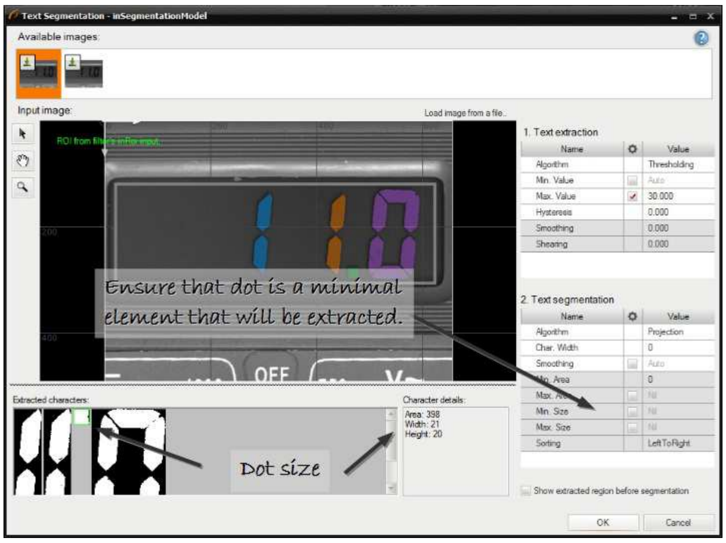
2. Text segmentation

Name	⚙	Value
Algorithm		BlobsDistance
[illegible]		5.000
Min. Area		0
Max. Area	☐	Nil
Min. Size	☐	Nil
Max. Size	☐	Nil
Sorting		LeftToRight

Character details:
Area: 4275
Width: 90
Height: 81

☐ Show extracted region before segmentation

8. Set the minimal and the maximal size of a character. The editor shows the character dimensions when the character is selected in the list below.


Text Segmentation - inSegmentationModel
Available images:
Input image:
Load image from a file..
ROI from filter's inRoi input
Ensure that dot is a minimal element that will be extracted.
Dot size
Extracted characters:
Character details:
OK
Cancel

1. Text extraction

Name	⚙	Value
Algorithm		Thresholding
Min. Value	☐	Auto
Max. Value	☑	30.000
Hysteresis		0.000
Smoothing		0.000
Shearing		0.000

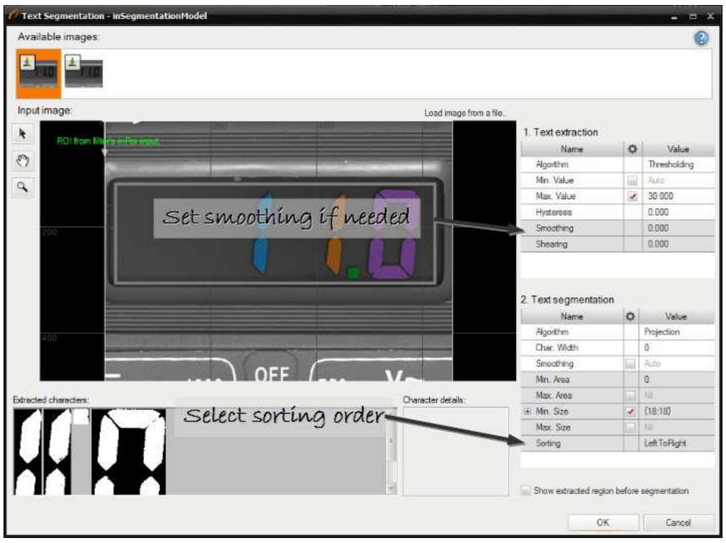
2. Text segmentation

Name	⚙	Value
Algorithm		Projection
Char. Width		0
Smoothing	☐	Auto
Min. Area		0
Max. Area	☐	Nil
Min. Size	☐	Nil
Max. Size	☐	Nil
Sorting		LeftToRight

Character details:
Area: 398
Width: 21
Height: 20

☐ Show extracted region before segmentation

9. Select a character sorting order, character deskewing (shearing) and image smoothing. Smoothing is important when images have low quality.



1. Text extraction

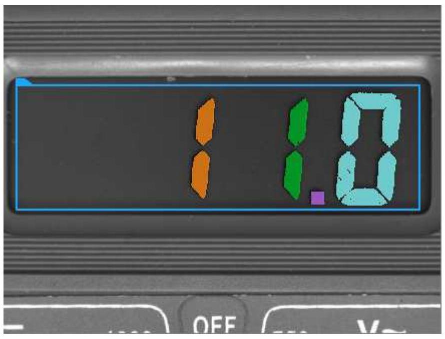
Name	⚙	Value
Algorithm		Thresholding
Min. Value	☐	Auto
Max. Value	✓	30.000
Hysteresis		0.000
Smoothing		0.000
Shearing		0.000

2. Text segmentation

Name	⚙	Value
Algorithm		Projection
Char. Width		0
Smoothing	☐	Auto
Min. Area		0
Max. Area	☐	Nil
Min. Size	✓	{18:18}
Max. Size	☐	Nil
Sorting		LeftToRight

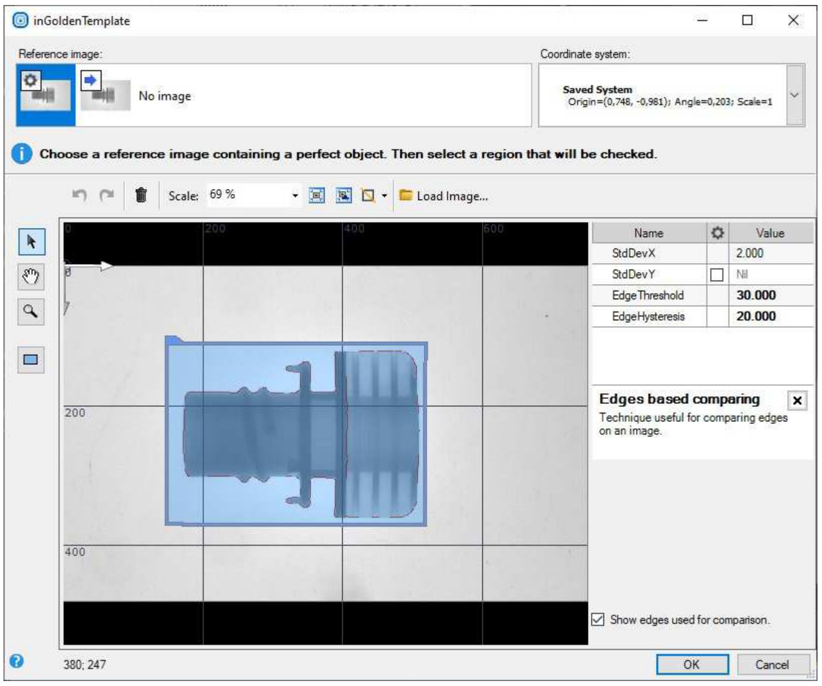
☐ Show extracted region before segmentation

10. Check results using available images.



Creating Golden Template Models

Golden template technique is the most powerful method for finding objects' defects. Editor presented below is available in filters CompareGoldenTemplate_Intensity and in CompareGoldenTemplate_Edges.



inGoldenTemplate
Reference image:
No image
Coordinate system:
Saved System
Origin=(0,748, -0,981); Angle=0,203; Scale=1
Choose a reference image containing a perfect object. Then select a region that will be checked.
Scale: 69 %
Load Image...
Edges based comparing
Technique useful for comparing edges on an image.
Show edges used for comparison.
380; 247
OK
Cancel

Name		Value
StdDevX		2.000
StdDevY	☐	Nil
EdgeThreshold		30.000
EdgeHysteresis		20.000

To create golden template, select template region and configure its parameters.

Remarks:

- To reduce computation time try to select only necessary part of an object,
- For comparing both edges and surface use two CompareGoldenTemplate filters,
- To create mode programmatically use filter CreateGoldenTemplate.

Creating Models for Golden Template

Introduction

Golden Template is an image comparison technique. It is based on the pixel-to-pixel comparison but uses multiple images and advanced algorithms to create a multi-image model. It is useful for finding general defects of objects that have fixed shape. In order to simplify the process of model creation a "GUI for Golden Template 2" was created. It is possible to access it by clicking on the button at the inModel input in the Properties window:

Opening GUI for Golden Template 2

Image Preparation

To create a golden template model you need at least three same-sized images representing the same object. The object should be placed in the same way in all the images. Otherwise, the final model may not be accurate enough.

The first step is to prepare the images. To be sure that the object is always precisely positioned and have the same size - in both the model and the program - you can use the sequence of filters presented below:

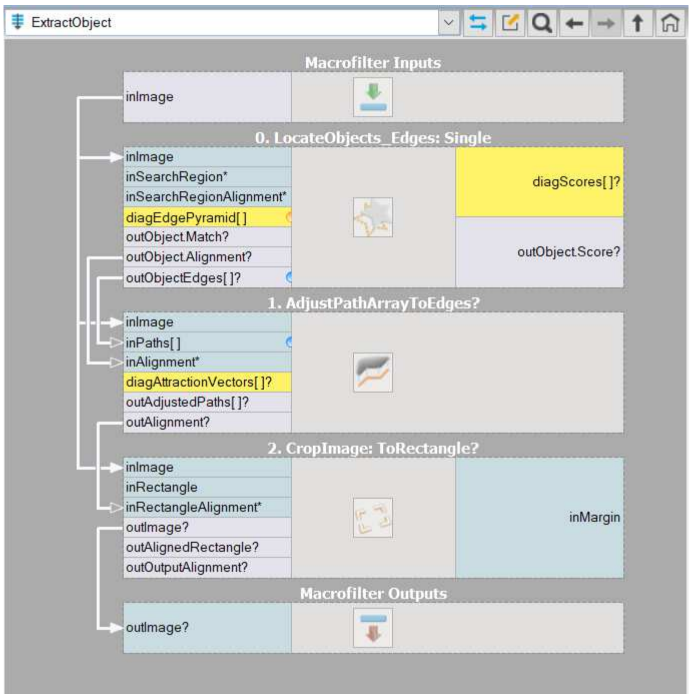


Image preparation process

Following steps were performed there:

- LocateSingleObject_Edges1 filter was used to create a robust model able to locate the logo and remember its alignment
- AdjustPathArrayToEdges allows you to improve the template matching results
- CropImageToRectangle gives evenly-cut images of the object. You should specify the **inRectangle** input manually.

If you have at least three images prepared, you can proceed to the next stage.

Model Creation

Add CreateGoldenTemplate2 filter to the program and click **inModel** in the properties window, as shown in the image at the beginning of the article. After that the following window will appear:

Golden Template Editor

Firstly, add images to the editor. It is possible to either use the drag and drop function or load images from the directory by clicking on the icon. The images will be used to create the Golden Template model, so they should present samples without defects.

After loading all the images, draw a mask that represents the object of interest on one image. It applies to all the other images that you loaded. That is why it was necessary to properly position the objects in the images. If no mask is detected the warning will appear. If the object covers most of the field of view, please feel free to mark the whole image.

Now you can check the preview with the "Show average of all images" and adjust the training parameters:

- **Downscale** - resizing the image dividing by the value. It greatly speeds up the computing in exchange for the ability to spot pixel-size defects
- **MaximalDisplacement** - possible error in object positioning, high values may impair detection of small defects, especially near edges
- **LargeDefectSize** - expected diameter of largest, extensive defects
- **BrightnessAugmentation** - allows for additional brightness deviation
- **NoiseAugmentation** - allows for additional noise presence in the images
- **SmoothingAugmentation** - allows for additional smoothing in the images, uses gaussian smoothing with specified standard deviation

Clicking *OK* button will start the training. The editor will close itself afterwards.

The model is created and loaded in the CreateGoldenTemplate2 filter. If you prepare the inspection images in the same way as during model creation (by positioning and cropping them) everything should work properly.

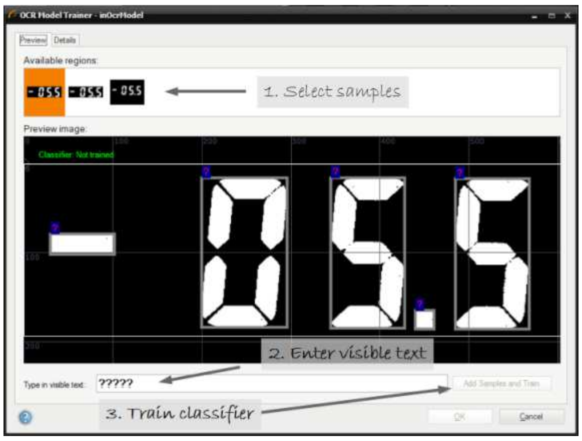
Creating Text Recognition Models

Text recognition editor creates an OCR model for getting text from regions. More details about the OCR technique can be found in Machine Vision Guide: Optical Character Recognition.

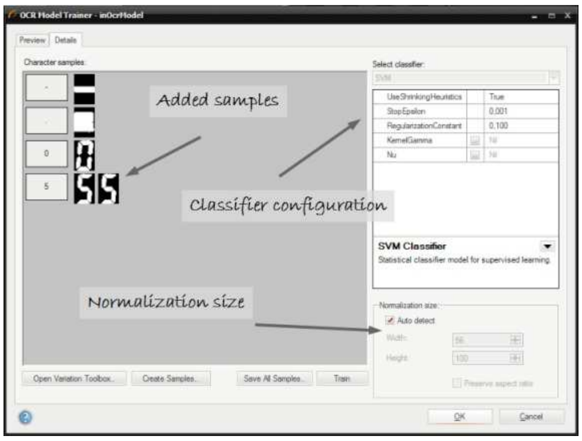
To create an OCR model a set of characters should be collected. If recognition score is low after training based on real samples then artificial character variations can created.

Creation of a model consists of following steps:

1. **Collecting real samples** - after opening the editor characters are visible and can be added to a training set.



After the training character samples can be viewed in the details tab:



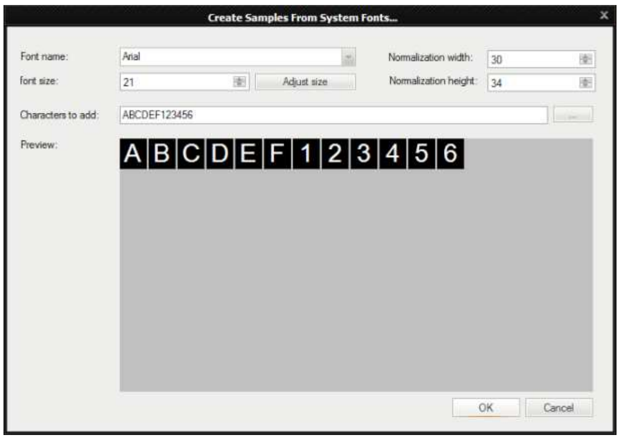
Select classifier: SVM

UseShrinkingHeuristics		True
StopEpsilon		0.001
RegularizationConstant		0.100
KernelGamma		Nil
Nu		Nil

SVM Classifier
Statistical classifier model for supervised learning.

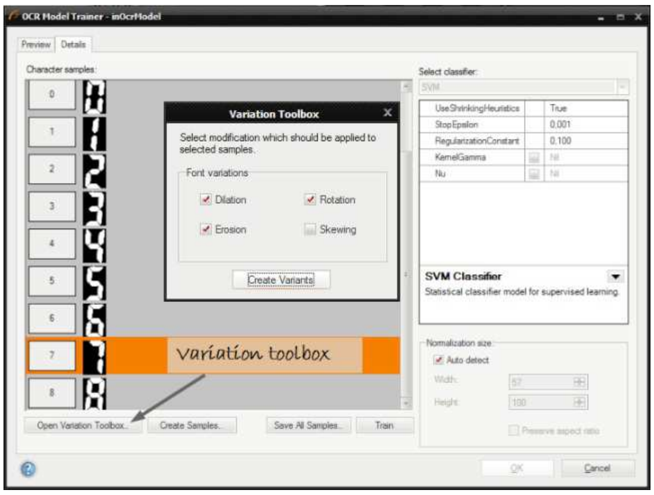
Normalization size:
[✓] Auto detect
Width: 56
Height: 100
[ ] Preserve aspect ratio

2. **Creating artificial samples** - when no samples are available user can create a training set using systems fonts.



Font name: Arial
font size: 21
Normalization width: 30
Normalization height: 34
Characters to add: ABCDEF123456

3. **Creating character variations** in case when no more samples are available and the training result is not fine the editor can modify existing samples to create a new set.



Select modification which should be applied to selected samples.

Font variations

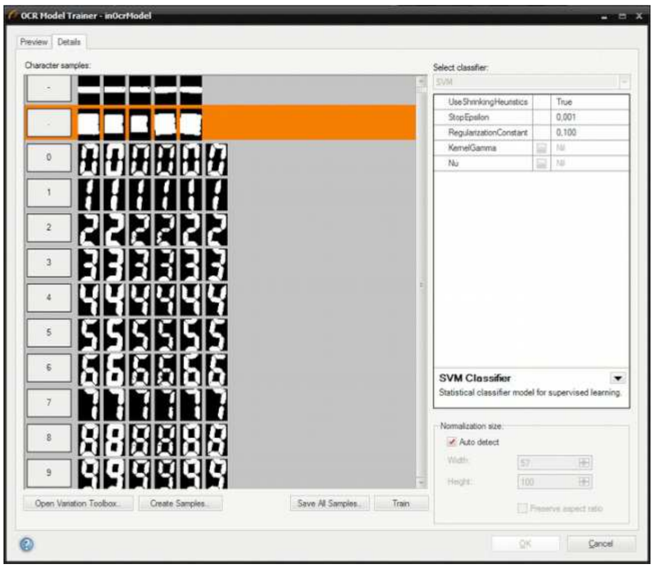
- [✓] Dilation
- [✓] Rotation
- [✓] Erosion
- [ ] Skewing

UseShrinkingHeuristics	True
StopEpsilon	0.001
RegularizationConstant	0.100
KernelGamma	Nil
Nu	Nil

SVM Classifier
Statistical classifier model for supervised learning.

Normalization size:
[✓] Auto detect
Width: 57
Height: 100
[ ] Preserve aspect ratio

The training set after adding new samples variations:

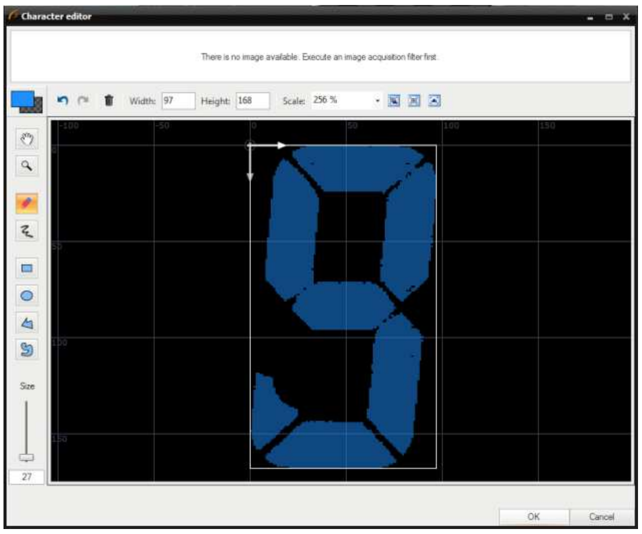


UseShrinkingHeuristics	True
StopEpsilon	0.001
RegularizationConstant	0.100
KernelGamma	Nil
Nu	Nil

SVM Classifier
Statistical classifier model for supervised learning.

Normalization size:
[✓] Auto detect
Width: 57
Height: 100
[ ] Preserve aspect ratio

4. **Editing samples** - in case when gathered samples contain noises, or its quality is low, user can edit them manually. The image below show how to edit a character '8' to get character '9'.



There is no image available. Execute an image acquisition filter first.

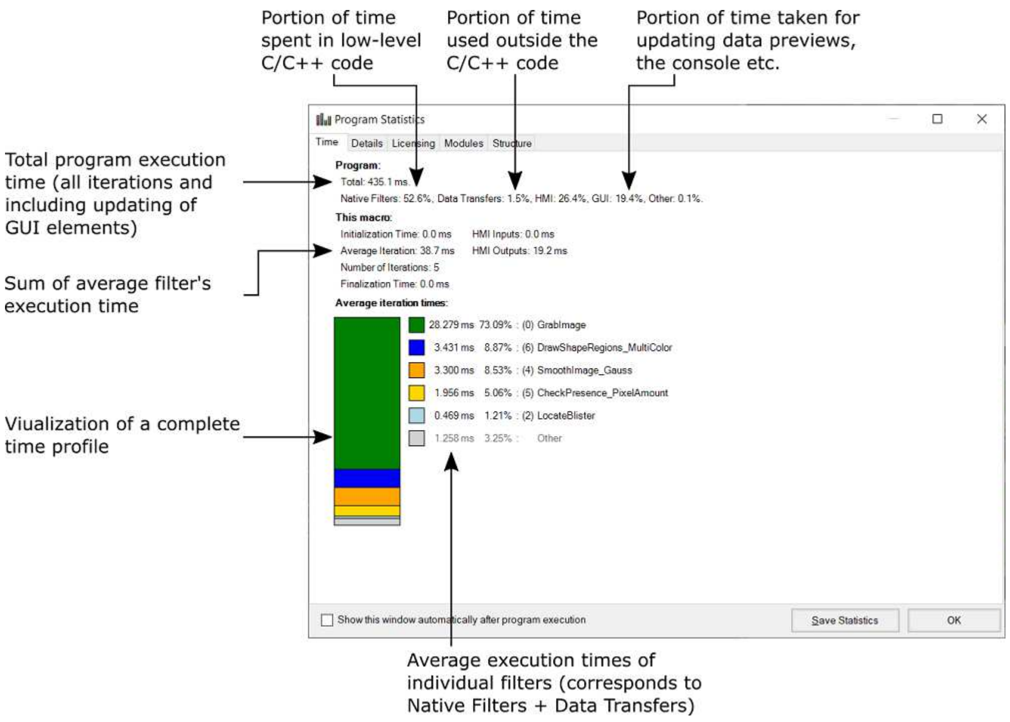
Width: 97 Height: 168 Scale: 256 %

Note:

- Each training character should have this same number of samples.
- In cases when some characters are very similar number of samples can be increased to improve classification.
- Character samples can be stored in an external directory to perform experiments on them.

Analysing Filter Performance

The Program Statistics window contains information about the time profile of the selected macrofilter. This is very important as real vision algorithms need to be very fast. However, before starting program optimization we must know what needs to be optimized, so that later we optimize the right thing. Here is how the required information is presented:


Portion of time spent in low-level C/C++ code
Portion of time used outside the C/C++ code
Portion of time taken for updating data previews, the console etc.
Total program execution time (all iterations and including updating of GUI elements)
Sum of average filter's execution time
Viualization of a complete time profile
Average execution times of individual filters (corresponds to Native Filters + Data Transfers)
Program Statistics
Time
Details
Licensing
Modules
Structure
Show this window automatically after program execution
Save Statistics
OK

Program:

Total: 435.1 ms.

Native Filters: 52.6%, Data Transfers: 1.5%, HMI: 26.4%, GUI: 19.4%, Other: 0.1%.

This macro:

Initialization Time: 0.0 ms	HMI Inputs: 0.0 ms
Average Iteration: 38.7 ms	HMI Outputs: 19.2 ms
Number of Iterations: 5	
Finalization Time: 0.0 ms	

Average iteration times:

🟩	28.279 ms	73.09% :	(0) GrabImage
🟦	3.431 ms	8.87% :	(6) DrawShapeRegions_MultiColor
🟧	3.300 ms	8.53% :	(4) SmoothImage_Gauss
🟨	1.956 ms	5.06% :	(5) CheckPresence_PixelAmount
⬜	0.469 ms	1.21% :	(2) LocateBlister
⬜	1.258 ms	3.25% :	Other

As can be seen on the above illustration, program execution time is affected by several different factors. The most important is the "Native Filters" time, which corresponds to the core data processing tools. Additional time is consumed by "Data Transfers", which is related to everything that happens on connections between filters – this encompasses automatic conversions as well as packing and unpacking arrays on array and singleton connections. Another statistic called "Other" is related to the virtual machine that executes the program. If it value is significant, then C++ code generation might be worth considering. The last element, "GUI", corresponds to visualization of data and program execution progress. You can expect that this part is related only to the development environment and can possibly be reduced down to zero in the runtime environment.

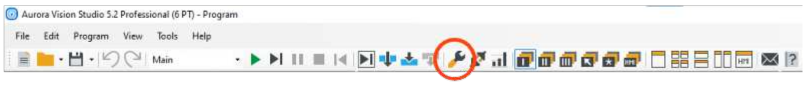
Remarks:

- In practice, performance statistics may vary significantly in consecutive program executions. It is advisable to run the program several times and check if the statistics are coherent. It might also be useful to add the EnumerateIntegers filter to your program to force a loop and collect performance statistics not from one, but from many program iterations.
- Turn off the diagnostic mode when testing performance.
- Data Preview Panels, animations in the Program Editor and even the Console Window can affect performance in Aurora Vision Studio. Choose *Program » Previews Update Mode » Disable Visualization* to test performance with minimal influence of the graphical environment. (Do not be surprised however that nothing is visible then).
- Even with all windows closed there are some background threads that affect performance. Performance may still be higher when you run the program with the Executor (runtime) application.
- Please note that the first program iteration might be slower. This is due to the fact that in the first iteration memory buffers are allocated, filters are initialized, communication with external devices is established etc.

See also: Optimizing Image Analysis for Speed.

Seeing More in the Diagnostic Mode

Programs can be executed in two different modes: *Diagnostic* and *Non-Diagnostic*. The difference between them is in the computation of values on the diagnostic outputs. Values of this kind of outputs are computed only in the *Diagnostic* mode. They can be helpful in debugging programs but are not necessary in its final version. In the *Non-Diagnostic* mode, execution is faster because no diagnostic values are computed.



The Diagnostic Mode switch.

The execution mode can be easily changed in Aurora Vision Studio using a button on the Application Toolbar. Outside of Aurora Vision Studio, programs are always executed in the *Non-Diagnostic* mode to provide the highest performance.

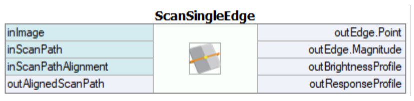
Diagnostic Filter Instances

Filters that have inputs connected to diagnostic outputs of some filters above them, are said to be diagnostic too. They are executed only when the program runs in the *Diagnostic* mode.

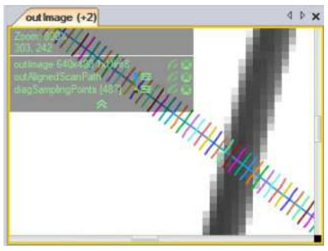
Example

The ScanSingleEdge filter has three diagnostic outputs:

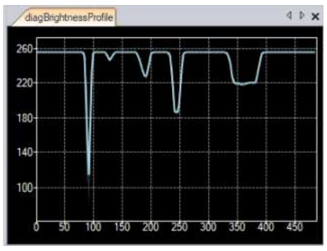
- **diagBrightnessProfile** is the profile of image brightness sampled along the scan path.
- **diagResponseProfile** is the profile derivative after preprocessing.
- **diagSamplingPoints** visualizes the points on the input image from where the brightness samples were taken.



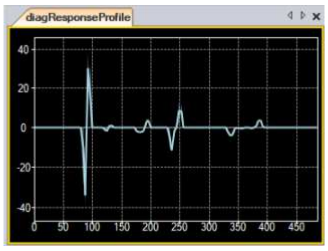
The ScanSingleEdge filter.



Input image with the scan path and the diagnostic sampling points.



The brightness profile.



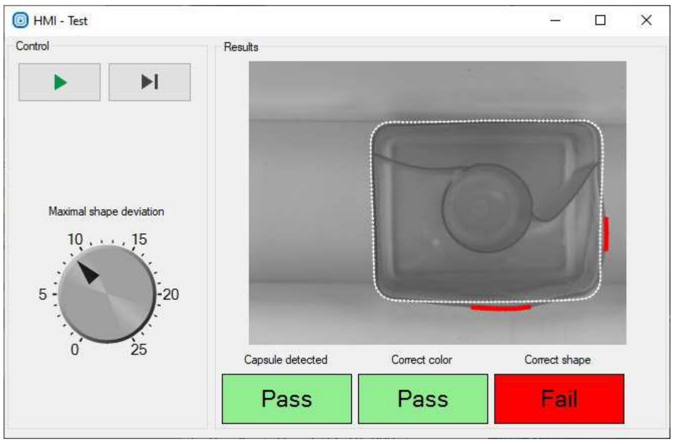
The response profile.

Deploying Programs with the Runtime Application

Introduction

Aurora Vision Executor is a lightweight application that can run programs created with Aurora Vision Studio. The GUI controls that appear in this application are the ones that have been created with the HMI Designer. The end user can manipulate the controls to adjust parameters and can see the results, but he is not able change the project.

Aurora Vision Executor application is installed with the Aurora Vision Studio Runtime package. It can be used on computers without the full development license. Only a runtime license is required. What is more, programs executed in Aurora Vision Executor usually run significantly faster, because there is no overhead of the advanced program control and visualization features of the graphical environment of Aurora Vision Studio.



The screen of Aurora Vision Executor.

Usage

Open a project from a file and use standard buttons to control the program execution. A file can also be started using the Windows Explorer context menu command *Run*, which is default for computers with Aurora Vision Studio Runtime and no Aurora Vision Studio installed.

Please note, that Aurora Vision Executor can only run projects created in exactly the same version of Aurora Vision Studio. This limitation is introduced on purpose – little changes between versions of Studio may affect program compatibility. After any upgrade your application should first be loaded and re-saved with Aurora Vision Studio as it then runs some backward compatibility checks and adjustments that are not available in Executor.

Console mode

It is possible to run Aurora Vision Executor in the console mode. To do so, the `--console` argument is needed to be passed. Note, that this mode makes the `--program` argument required so the application will know which program to run at startup.

Aurora Vision Executor is able to open a named pipe where it's log will be write into. This is possible with `--log-pipe` argument which accepts a pipe name to be opened. One may then connect to the pipe and process Aurora Vision Executor log live. This can be easily done e.g. in C#:

```
var logPipe = new NamedPipeClientStream(".", "myProjectPipe", PipeDirection.In);
logPipe.Connect();

byte[] buffer = new byte[1024];
int count = 0;
while (logPipe.IsConnected && (count = logPipe.Read(buffer, 0, 1024)) > 0)
{
 Console.WriteLine(Encoding.UTF8.GetString(buffer, 0, count));
}
```

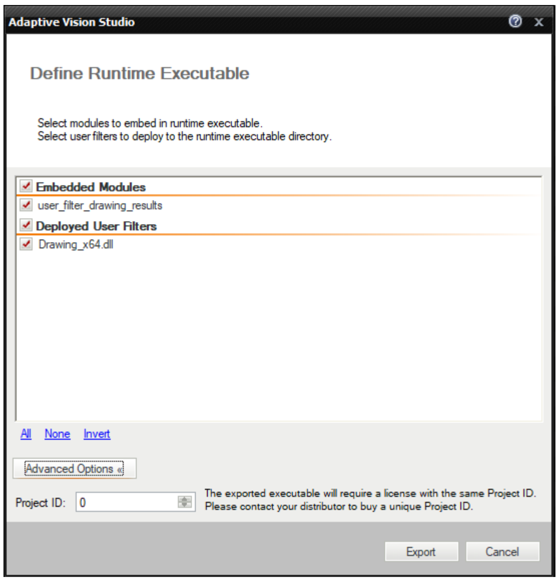
Available Aurora Vision Executor arguments are as follows:

--program	Path to the program to be loaded
--log-level	Sets the logged information level
--console	Runs the application in the console mode
--auto-close	Automatically closes the application when program is finished. Meaningful only in console mode.
--language	Specifies the language code to use as the user interface language.
--attach	Attaches application process to the calling process console.
--log-pipe	Creates a named pipe which will be populated with log entries during application lifetime. Meaningful only in console mode.
--help	Displays help

Runtime Executables

Aurora Vision Executor can open .avproj files, the same as Aurora Vision Studio, however it is better to use .avexe files here. Firstly one can have a single binary executable file for the runtime environment. Secondly this file is encrypted so that nobody is able to look at project details. To create one, open a project in Aurora Vision Studio and use *File » Export to Runtime Executable...*. This will produce an .avexe file that can be executed directly from the Windows Explorer.

If Aurora Vision project contains any User Filter libraries, it is crucial to put their *.dll files into the appropriate directory when running in Aurora Vision Executor. This is when exporting to .avexe file might also be a handy option. While defining the .avexe contents, it is possible to select all the User Filters libraries that the exported project depends on. Selected libraries are deployed then to the same directory as generated .avexe file and the .avexe itself is set to use all User Filter libraries from its directory.


Adaptive Vision Studio

Define Runtime Executable

Select modules to embed in runtime executable.
Select user filters to deploy to the runtime executable directory.

- [✓] Embedded Modules
  - [✓] user_filter_drawing_results
- [✓] Deployed User Filters
  - [✓] Drawing_x64.dll

All None Invert

Advanced Options «

Project ID: 0

The exported executable will require a license with the same Project ID.
Please contact your distributor to buy a unique Project ID.

Export Cancel

Defining the Runtime Executable.

In case there are any other dependencies, e.g. exposed by used User Filter libraries, one can add them into the Aurora Vision project as an attachment in Project Explorer and also deploy with .avexe file during export.

Project ID available in *Advanced Options* is an additional parameter that makes only a specific license able to run the application. At the customer request, Aurora Vision Team can generate the key (Project ID) and add it to the Runtime license. The same Project ID must be set for Runtime Executable export to connect the application with the specific license. If you leave this field default, any license will be able to run the .avexe file.

Trick: Configuration File as a Module Not Exported to AVEXE

It is often convenient to have an configuration file separated from the executable so that various parameters can be adjusted for a particular installation (but not made available to the end user). This can be easily implemented with a simple programming idiom:

1. Use global parameters in your project for values that might require adjusting.
2. Place the global parameters in a separate module (through the Project Explorer window).
3. Exclude the module when exporting the .avexe file.
4. In the runtime environment copy the .avexe file and the module (as a separate file) with global parameters.
5. Open the config module in the Notepad to edit it when needed.

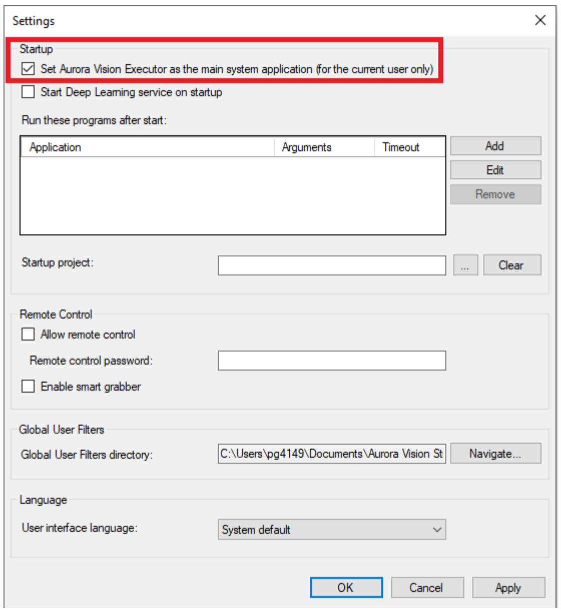
Other Runtime Options

Please note that Aurora Vision Executor is only one of several options for creating end-user's applications. Other available options are:

- .NET Macrofilter Interface Generator – generates a native .NET assembly (a DLL file) and makes it possible to invoke macrofilters created in Aurora Vision Studio as simple class methods from a .NET project. Internally the execution engine of Aurora Vision Studio is used, so modifying the related macrofilters does not require to re-compile the .NET solution. The HMI can be implemented with WinForms, WPF or similar technologies.
- C++ Code Generator – generates native C++ code (a CPP file) that is based on Aurora Vision Library C++. This code can be integrated with bigger C++ projects and the HMI can be implemented with Qt, MFC or similar libraries. Each time you modify the program in Studio, the C++ code has to be re-generated and re-compiled.

Set Aurora Vision Executor as the "system shell"

On Windows systems it is possible to set Aurora Vision Executor as the "system shell", thus removing Desktop, Menu Start etc. completely. Go to Settings in Aurora Vision Executor and the Startup section. Mark the *Set Aurora Vision Executor as the main system application (for the current user only)*. Please be informed that this option requires administrator privileges.


Settings

Startup
- [✓] Set Aurora Vision Executor as the main system application (for the current user only)
- [ ] Start Deep Learning service on startup

Run these programs after start:

Application	Arguments	Timeout

Add Edit Remove

Startup project: ... Clear

Remote Control
- [ ] Allow remote control

Remote control password:
- [ ] Enable smart grabber

Global User Filters

Global User Filters directory: C:\Users\pg4149\Documents\Aurora Vision St Navigate...

Language

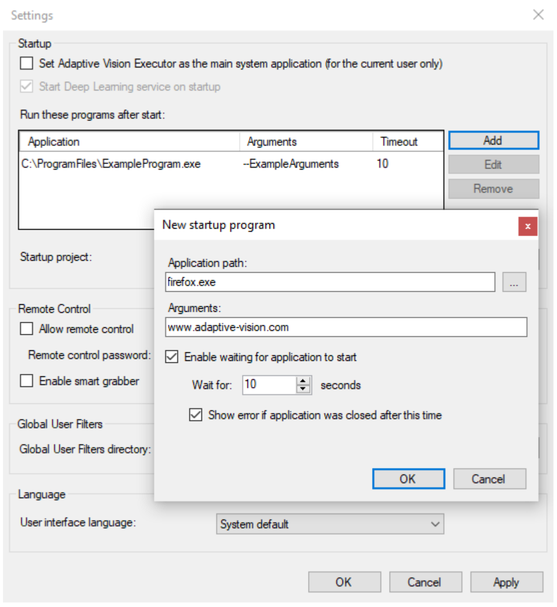
User interface language: System default

OK Cancel Apply

Set Aurora Vision Executor as the "system shell"

Startup Applications

It is possible to run any process before starting a program in Aurora Vision Executor. Go to Settings in Aurora Vision Executor and the Startup section. To define a new startup program select the *Add* button on the right. In a *New startup program* dialog box you need to specify the application path (obligatory) and arguments (optional). It is similar to typing the application name and command-line arguments in the *Run* dialog box of the Windows Start menu. The added program will appear in the list. All added programs will start each time you run Aurora Vision Executor.


Settings

Startup
- [ ] Set Adaptive Vision Executor as the main system application (for the current user only)
- [✓] Start Deep Learning service on startup

Run these programs after start:

Application	Arguments	Timeout
C:\ProgramFiles\ExampleProgram.exe	--ExampleArguments	10

Startup project:

Remote Control
- [ ] Allow remote control

Remote control password:
- [ ] Enable smart grabber

Global User Filters

Global User Filters directory:

Language

User interface language: System default

New startup program

Application path: firefox.exe

Arguments: www.adaptive-vision.com

- [✓] Enable waiting for application to start

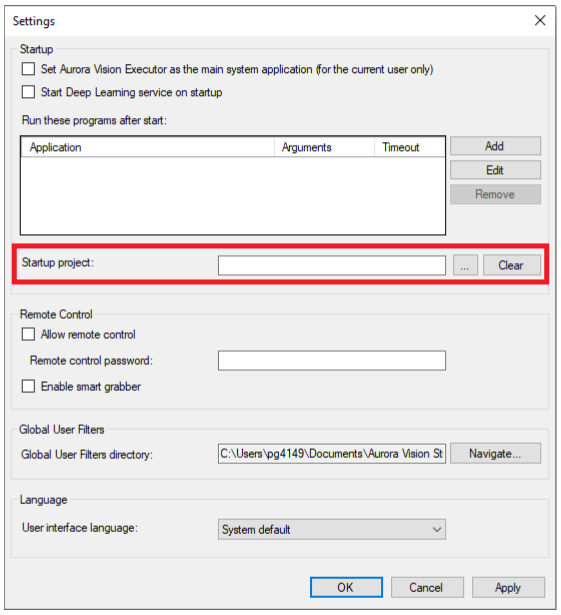
Wait for: 10 seconds

- [✓] Show error if application was closed after this time

Defining Startup Applications

Startup Project

It is possible to choose the project the Aurora Vision Executor should run after the startup . Go to Settings in Aurora Vision Executor and the Startup section. To define the startup project select the ... button on the right. In an *Open* dialog box you need to specify the startup project path (obligatory). The added project's path will appear in the box. It will start each time you run Aurora Vision Executor.


Settings

Startup
- [ ] Set Aurora Vision Executor as the main system application (for the current user only)
- [ ] Start Deep Learning service on startup

Run these programs after start:

Application	Arguments	Timeout

Startup project:

Remote Control
- [ ] Allow remote control

Remote control password:
- [ ] Enable smart grabber

Global User Filters

Global User Filters directory: C:\Users\pg4149\Documents\Aurora Vision St

Language

User interface language: System default

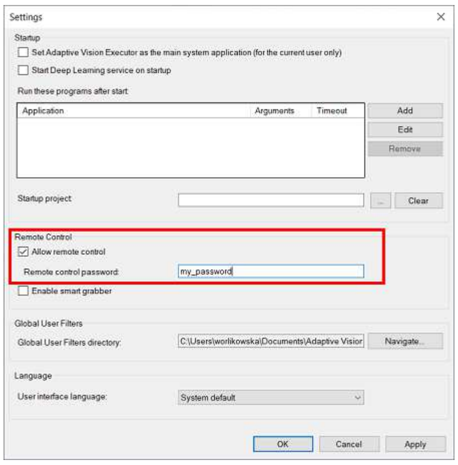
Defining Startup Project

If Deep Learning Service is installed you can choose to run it on start by selecting 'Start Deep Learning service on startup'.

Remote Access to the Runtime Application

Introduction

It is possible to use Aurora Vision Studio to control an Aurora Vision Executor running on a remote computer, if only the two computers are in the same Ethernet subnet. To enable remote access in Aurora Vision Executor, the option "Allow remote control" should be enabled. For security reasons, it is also possible to protect connection with a password:


Settings

Startup

☐ Set Adaptive Vision Executor as the main system application (for the current user only)

☐ Start Deep Learning service on startup

Run these programs after start

Application	Arguments	Timeout

Startup project:

Remote Control

☑ Allow remote control

Remote control password: my_password

☐ Enable smart grabber

Global User Filters

Global User Filters directory: C:\Users\worlikowska\Documents\Adaptive Visior

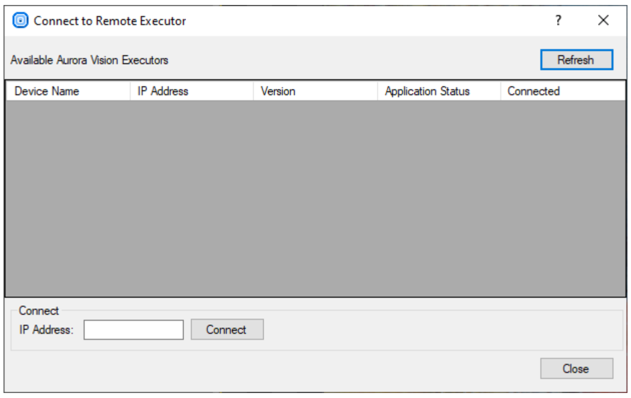
Language

User interface language: System default

Enable remote executor.

Usage

In Aurora Vision Studio the list of available remote systems is accessible in the Connect to Remote Executor window which opens up after choosing the File › Connect to Remote Executor menu command:


Connect to Remote Executor

Available Aurora Vision Executors

Device Name	IP Address	Version	Application Status	Connected

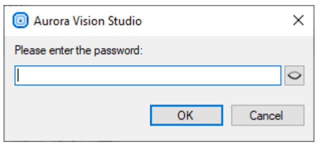
Connect

IP Address:

Browse remote systems.

If an expected remote system is not visible on the list, please verify that: (1) it is configured for the same local area network, and (2) it can access Ethernet through its firewall (in the first place verify that in the Windows' Control Panel, Windows Firewall section, the option "Notify me when Firewall blocks a new program" is enabled). In some local networks information about communication type and ports may be useful to assign application access through firewall. In this case please use UDP ports 6342 and 6343 or TCP port 6342.

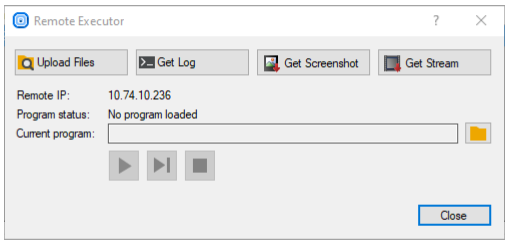
When connecting to a selected device, the program will ask for the password (if it has been set):


Aurora Vision Studio

Please enter the password:

Password protection window.

After successful connection, management of the selected executor becomes possible. Now you are able to do several actions: Upload Files, Control Program and Get Diagnostics.


Remote Executor

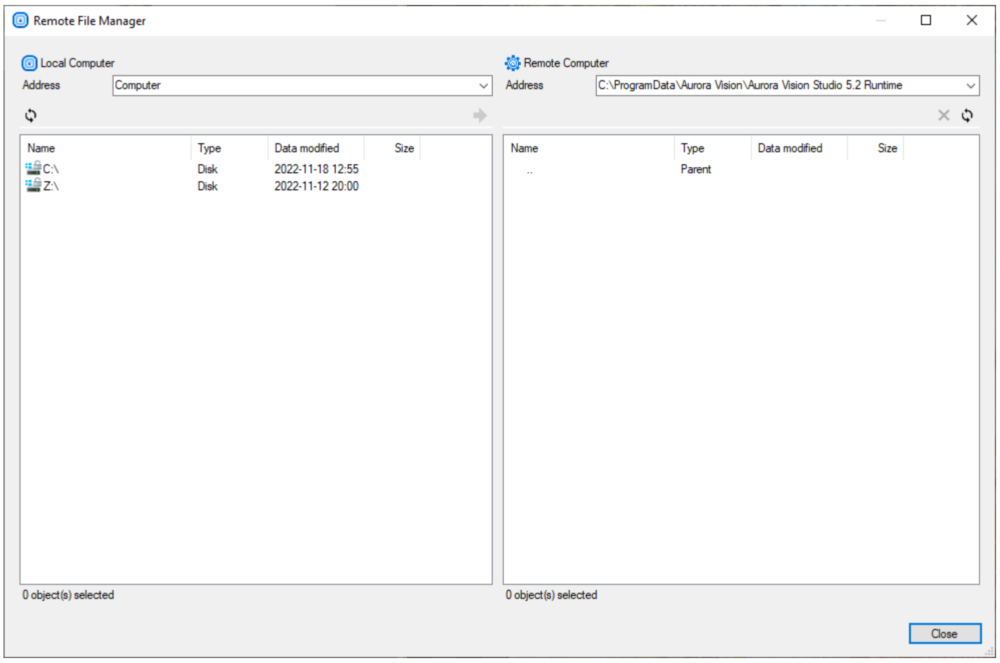
Remote IP: 10.74.10.236

Program status: No program loaded

Current program:

Remote executor window.

Upload Files:


Remote File Manager

Local Computer

Address: Computer

Name	Type	Data modified	Size
C:\	Disk	2022-11-18 12:55	
Z:\	Disk	2022-11-12 20:00	

0 object(s) selected

Remote Computer

Address: C:\ProgramData\Aurora Vision\Aurora Vision Studio 5.2 Runtime

Name	Type	Data modified	Size
..	Parent		

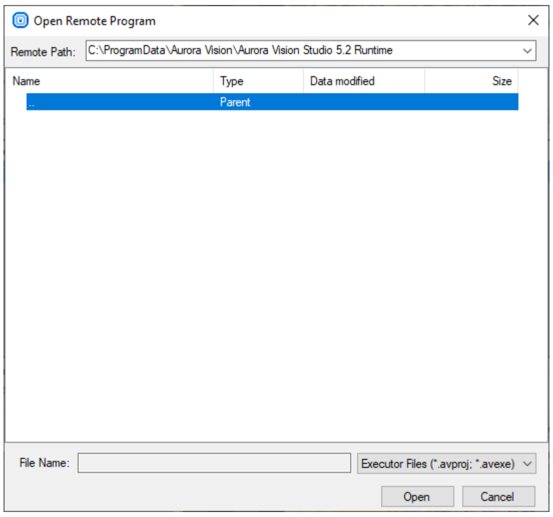
0 object(s) selected

Upload files window.

To Manage Files on remote executor, click Upload Files button. In this window it is possible to send program files to the remote executor. By default all files are stored in the directory <CommonApplicationData>\Aurora Vision\Aurora Vision Executor.

Program Control:

In the Remote executor window you are able to control currently executed program. In the middle of the window we can see the current program status and the path to the currently loaded program. Next there are four buttons controlling program execution. Finally there is an option to open another program.


Open Remote Program

Remote Path: C:\ProgramData\Aurora Vision\Aurora Vision Studio 5.2 Runtime

Name	Type	Data modified	Size
..	Parent		

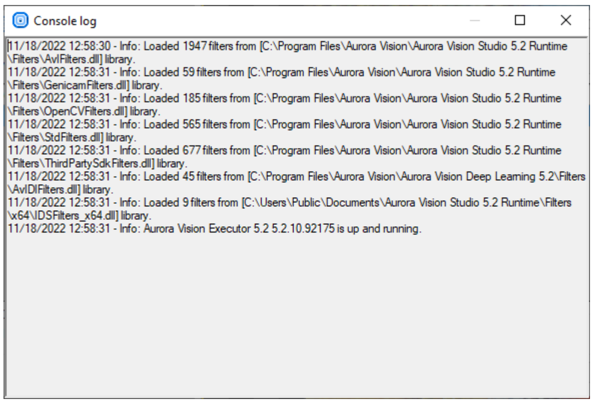
File Name:

Executor Files (*.avproj; *.avexe)

Remote file dialog window.

Diagnostics:

In the Remote Executor window there is available a few diagnostic tools. Here, it is possible to preview the execution log and a screen preview from the currently connected executor. All diagnostic options are located at the top of the Remote executor window.


Console log

```
11/18/2022 12:58:30 - Info: Loaded 1947 filters from [C:\Program Files\Aurora Vision\Aurora Vision Studio 5.2 Runtime
\Filters\AvlFilters.dll] library.
11/18/2022 12:58:31 - Info: Loaded 59 filters from [C:\Program Files\Aurora Vision\Aurora Vision Studio 5.2 Runtime
\Filters\GenicamFilters.dll] library.
11/18/2022 12:58:31 - Info: Loaded 185 filters from [C:\Program Files\Aurora Vision\Aurora Vision Studio 5.2 Runtime
\Filters\OpenCVFilters.dll] library.
11/18/2022 12:58:31 - Info: Loaded 565 filters from [C:\Program Files\Aurora Vision\Aurora Vision Studio 5.2 Runtime
\Filters\StdFilters.dll] library.
11/18/2022 12:58:31 - Info: Loaded 677 filters from [C:\Program Files\Aurora Vision\Aurora Vision Studio 5.2 Runtime
\Filters\ThirdPartySdkFilters.dll] library.
11/18/2022 12:58:31 - Info: Loaded 45 filters from [C:\Program Files\Aurora Vision\Aurora Vision Deep Learning 5.2\Filters
\AvlDlFilters.dll] library.
11/18/2022 12:58:31 - Info: Loaded 9 filters from [C:\Users\Public\Documents\Aurora Vision Studio 5.2 Runtime\Filters
\x64\IDSFilters_x64.dll] library.
11/18/2022 12:58:31 - Info: Aurora Vision Executor 5.2 5.2.10.92175 is up and running.
```

Log preview window.

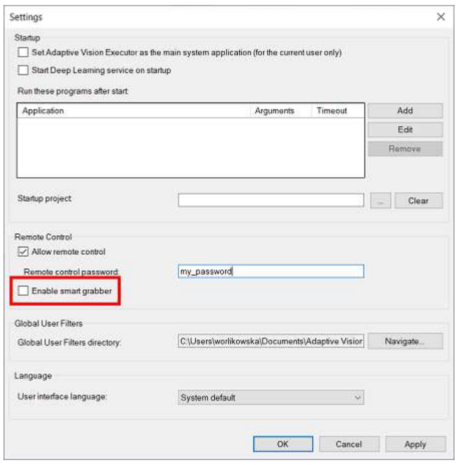
Remote Image Acquisition

Introduction

It is also possible to get images from a connected device using the remote executor. This method offers an easy way to acquire images using a different image protocols. Remote image acquisition filters enable to run this same code on the smart cameras (client side) as good as on developer computers without any additional modifications.

Usage

To enable remote image acquisition it is necessary to stop the executor program and enable the "Enable smart grabber" option:



Settings

Startup

- [ ] Set Adaptive Vision Executor as the main system application (for the current user only)
- [ ] Start Deep Learning service on startup

Run these programs after start

Application	Arguments	Timeout

Startup project:

Remote Control

- [✓] Allow remote control

Remote control password: my_password

- [ ] Enable smart grabber

Global User Filters

Global User Filters directory: C:\Users\worlikowska\Documents\Adaptive Visior

Language

User interface language: System default

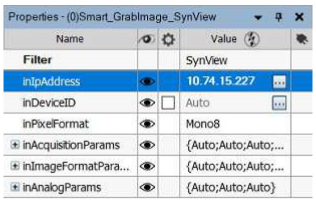
Enable Remote Image Acquisition.

In Aurora Vision Studio it is possible to use the filters from the category Camera Support\Smart to get images from a device connected to the remote Executor system.

Currently the available image acquisition protocols are:

- **AvSMART** – access to AvSMART,
- **GenICam** – access to GenICam complaint devices,
- **Roseek** – access to Roseek cameras,
- **SynView**– access to NET GmbH cameras,
- **WebCamera**– access to DirectX complaint image sources like web cameras or frame grabbers.

To grab images using one of these protocols it is necessary to set **inIpAddress** in the filter input. The IP Address can be checked in the Connect to Remote Executor window when "Allow remote control" option is enabled in the Executor.



Properties - (0)Smart_GrabImage_SynView

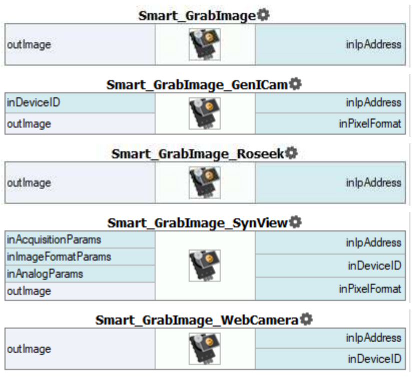
Name	Value
Filter	SynView
inIpAddress	10.74.15.227
inDeviceID	Auto
inPixelFormat	Mono8
⊞ inAcquisitionParams	{Auto;Auto;Auto;...
⊞ inImageFormatPara...	{Auto;Auto;Auto;...
⊞ inAnalogParams	{Auto;Auto;Auto}

Smart_GrabImage_SynView filter properties.

If system detect that the provided IP address describes local machine the Smart_GrabImage filters will perform all operation on local computer. So it is no need to perform any changes during transferring project from developers machine to the client side device.

If the program is deployed to a device with another IP address then the input **inIpAddress** should be changed to the IP of the Executor or left empty. If the input **inIpAddress** is empty then the image is always grabbed using an appropriate local image acquisition protocol.

Available filters for the remote image acquisition:



If **inDeviceID** is set to*NIL*the first found device will be used.

Performing General Calculations

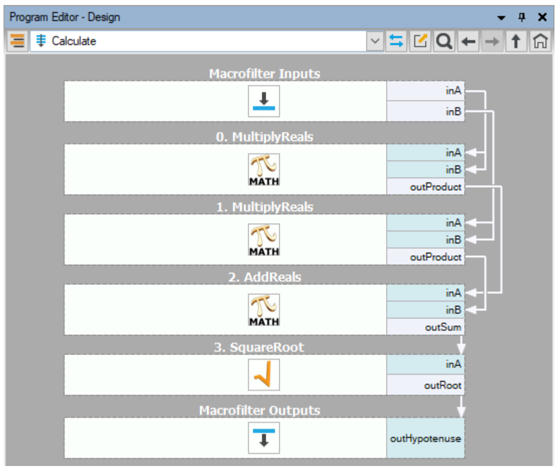
Introduction

Apart from using image processing or computer vision tools, most often it is also necessary to perform some general calculations on numeric values or geometrical coordinates. There are two ways to do this in Aurora Vision Studio:

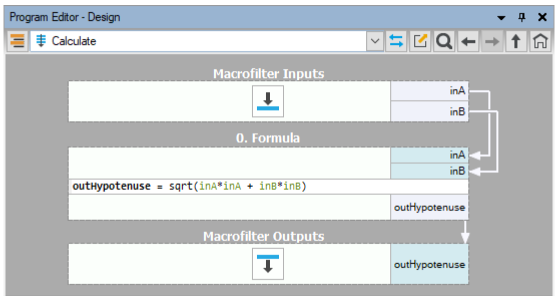
1. Using filters from the Standard Library.
2. Using special formula blocks.

Example

Let us assume that we need to compute the hypotenuse $\sqrt{a^2+b^2}$. Here are the two possible solutions:



Calculations using standard filters.



```
outHypotenuse = sqrt(inA*inA + inB*inB)
```

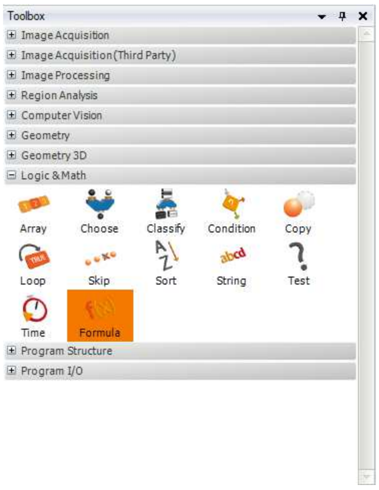
Calculations using formula blocks.

The second approach, using formula blocks, is the most recommended. Data flow programming is just not well suited for numerical calculations and standard formulas, which can be defined directly in formula blocks, are much easier to read and understand. You can also think of this feature as an ability to embed little spreadsheets into your machine vision algorithms.

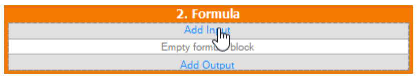
Creating Formula Blocks

To create a formula block:

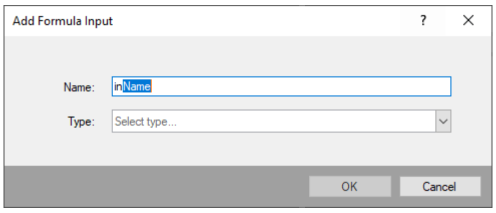
1. Drag and drop the Formula tool from the Toolbox (*Logic & Math* section) to the Program Editor:



2. Add inputs and outputs using the context menu or by clicking *Add Input* and *Add Output* links:



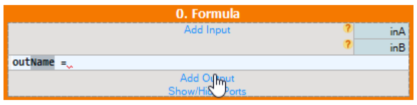
3. For each input define the name and the type:



Name: inName

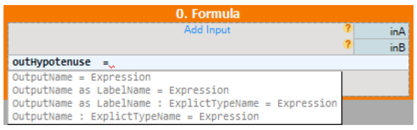
Type: Select type...

4. In order to create an output, first press *Add Output* button. It will create a new output with template name: *outName*:



```
outName =
```

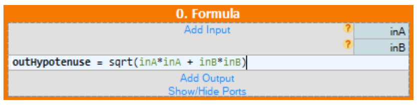
Replace the highlighted *Name* with meaningful name. While typing a Tooltip with available options should appear:



```
outHypotenuse  =
OutputName = Expression
OutputName as LabelName = Expression
OutputName as LabelName : ExplictTypeName = Expression
OutputName : ExplictTypeName = Expression
```

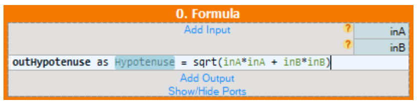
Example usage of each option:

- After the name type equal sign = and the expression. There is no need to define a type of output. The type will be assigned automatically.



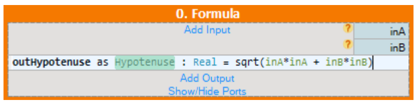
```
outHypotenuse = sqrt(inA*inA + inB*inB)
```

- After the name add *as* followed by *LabelName*. This label will later appear as a formula output.



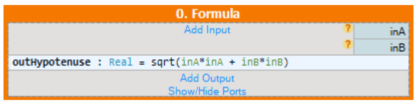
```
outHypotenuse as Hypotenuse = sqrt(inA*inA + inB*inB)
```

- After the name add *label* as described in the previous point. Then add colon *:* and directly specify the output type:



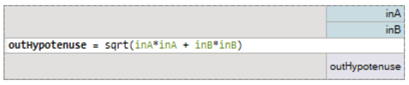
```
outHypotenuse as Hypotenuse : Real = sqrt(inA*inA + inB*inB)
```

- In the last option output type is defined directly. After the name add colon *:* and directly specify the output type:



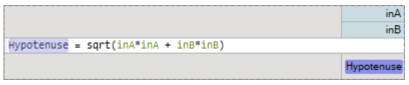
```
outHypotenuse : Real = sqrt(inA*inA + inB*inB)
```

Final formula should look like this:



```
outHypotenuse = sqrt(inA*inA + inB*inB)
```

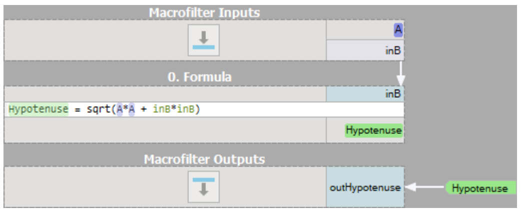
Without output label.



```
Hypotenuse = sqrt(inA*inA + inB*inB)
```

With output label.

Not only the output data of the formulas can be labeled but also inputs and outputs of other filters outside of the formula. This way you can perform a direct call to another labeled data inside a formula without explicitly defining new formula inputs. In the image below the input *A* of the macrofilter is labeled (highlighted in violet color) and a direct call of this label (A) is performed inside the formula block. Later the output *Hypotenuse* is connected via a label with the macrofilter output, instead of linking it directly using an outgoing arrow connection.



```
Hypotenuse = sqrt(A*A + inB*inB)
```

Remarks

- Existing formulas can also be edited directly in a formula block in the Program Editor.
- It is also possible to create new inputs and outputs by dragging and dropping connections onto a formula block.
- Formula blocks containing incorrect formulas are marked with red background. Programs containing such filters cannot be run.
- When defining a formula for an output it is possible to use other outputs, provided that they are defined earlier. The order can be changed through the outputs context menu.
- For efficiency reasons it is advisable not to use "heavy" objects in formulas, such as images or regions.

Syntax and Semantics

For complete information about the syntax and semantics please refer to the Formulas article in the Programming Reference.

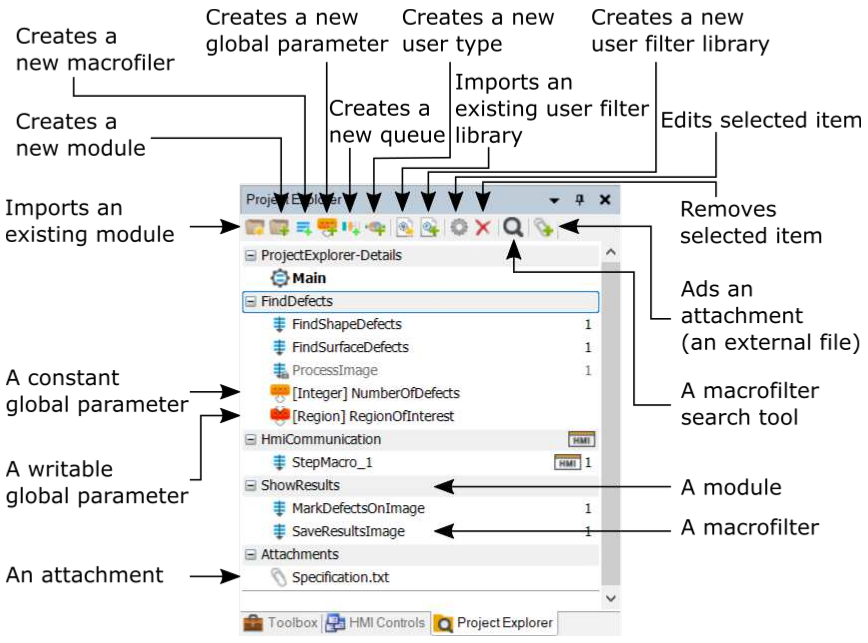
Managing Projects with Project Explorer

Introduction

Project Explorer is a window displaying elements which are contained in the currently opened project:

- Modules
    - Macrofilters (definitions)
    - Global Parameters
    - User Types
- User Filter libraries
- Attachments

Its main goal is to provide a single place to browse, add, remove, rename or open the items, which are grouped into categories in the same way as the filters in the Filter Catalog. A category may correspond to a standard module or to a module of a User Filter library. There is also one special category, Attachments, which appears when the user adds an external file to the project (it might be for example a text document with a piece of documentation).



Modules in the Project Explorer.

Opening Macrofilters

As described in Running and Analysing Programs, there are two ways of navigating through the existing macrofilters. One of them is with the Project Explorer window, which displays *definitions* of macrofilters, not the instances. After double-clicking on a macrofilter in the Project Explorer, however, a macrofilter instance is opened in the Program Editor. As one macrofilter definition can have zero, one or many instances. Some special rules apply to which of the instances it is:

- If possible, the most recently executed instance is opened.
- If no instance has been executed yet, the most recently created one is opened.
- If there are no instances at all the "ghost instance" is presented, which allows editing the macrofilter, but will never have any data on the output ports.

Macrofilter Counter

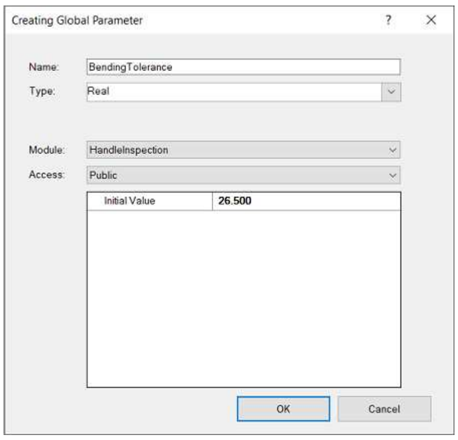
Macrofilter counter shows how many times a given macrofilter is used in the program.

ADVANCED NOTE: If a macrofilter X is used in a macrofilter Y and there are multiple instances of the macrofilter Y, we still consider macrofilter X being used once. Number of uses is not the same as number of instances.

Global Parameters

If some value is used many times in several different places of a program then it should be turned into a global parameter. Otherwise, consecutive changes to the value will require the error-prone manual process of finding and changing all the occurrences. It is also advisable to use global parameters to clearly distinguish the most important values from the project specification – for example the expected dimensions and tolerances. This will make a program much easier to maintain in future.

In Aurora Vision Studio global parameters belong to specific modules and are managed in the Project Explorer. To create one, click the *Create New Global Parameter...* button and then a dialog box will appear where you will provide the name, the type and the value of the new item. After a global parameter is created it can be dragged-and-dropped on filter inputs and appropriate connections will be created with a visual label displaying the name of the parameter.


Creating Global Parameter

Name: BendingTolerance

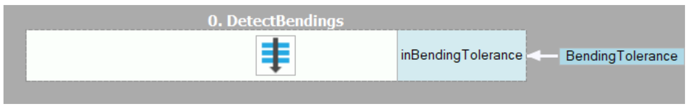
Type: Real

Module: HandleInspection

Access: Public

Initial Value	26.500

Creating a global parameter.

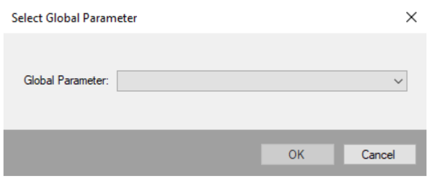


Global parameter used in a program.

Global parameters contained in a project can also be edited in the Properties window after being selected in the Project Explorer or in the Program Editor.

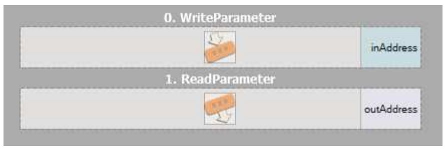
Since version 4.11 it is possible to create and manage global parameters with dedicated filters: WriteParameter and ReadParameter. They are available in the Program Structure category of the Toolbox.

While adding a new input, "Select Global Parameter" window is displayed:


Select Global Parameter

Global Parameter:

You can select an already created global parameter or create a new one.



Global parameter "inAddress" created within the filter WriteParameter can be read with the ReadParameter filter.

Thanks to these filters you can easily read or write the values of global parameters anywhere in your algorithm. In order to facilitate development the icon of the global parameter has different appearance depending on whether it is overwritten somewhere in the program. The color of the icon will be red then so that you will know that this value may change during the execution of your application.

To see how Global Parameters work in practice, check out our official example: HMI Handling Events.

Remarks:

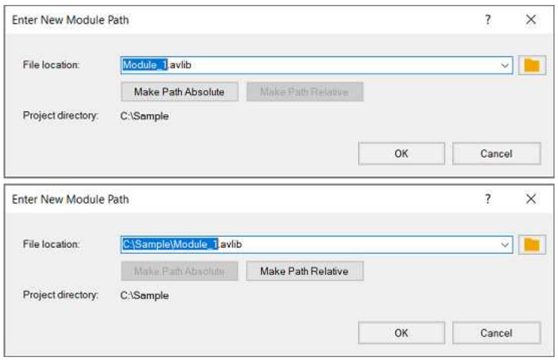
- Connected filters are not re-executed after the global parameter is changed. This is due to the fact, that many filters in different parts of the program can be connected to one global parameters. Re-executing all of them could cause unexpected non-local program state changes and thus is forbidden.
- Do NOT use writable global parameters unless you really must. In most cases data should be passed between filters with explicit connections, even if there are a lot of them. Writable global parameters should be used only for some very specific tasks, most notably for configuration parameters that may be dynamically loaded during program execution and for high level program statistics that may be manipulated through the HMI (like the last defect time).

Modules

When a project grows above 10-20 macrofilters it might be appropriate to divide it into several separate modules, each of which would correspond to some logical part. It is advisable to create separate modules for things like i/o communication, configuration management or for automated unit testing. Macrofilters and global variables will be then grouped in a logical way and it will be easier to browse them.

Modules are also sometimes called "libraries of macrofilters". This is because they provide a means to develop sets of common user's tools that can be used in many different projects. This might be very handy for users who specialize in specific market areas and who find some standard tasks appearing again and again.

To create a module, click the *Create New module...* button and then a dialog box will appear. In there you specify the location and name of the module. The path may be absolute or relative. Modules are saved with extension **.avlib**. Saving and updating the module files happens when the containing program is saved.


Enter New Module Path

File location: Module_1.avlib

Project directory: C:\Sample

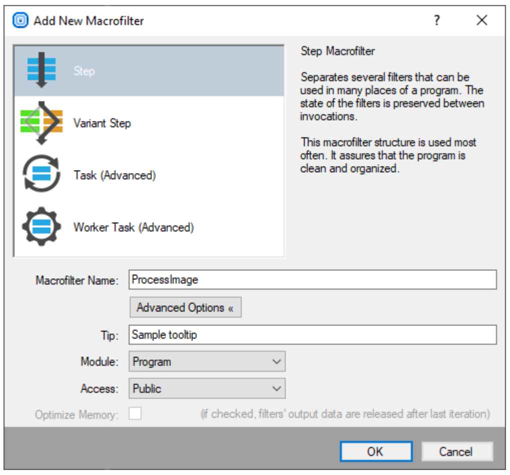
Enter New Module Path

File location: C:\Sample\Module_1.avlib

Project directory: C:\Sample

Module creation windows for both absolute and relative paths.

After creating a module you may move already existing macrofilters into the module by dragging them onto the module in Project Explorer. Clicking the *Create New Macrofilter...* button allows you to create a new macrofilter with the given name. During that you can access *Advanced options* by clicking the appropriate button. There you can specify the parent module and access to the macrofilter. Access can be either *private* or *public*. Private macrofilters cannot be directly used outside its module. You can also provide a tooltip (description) for the macrofilter. This will show up in the properties section of an instance and when hovered over in the Project Explorer. Tooltip should describe the macrofilter's purpose and be concise. They are especially important when creating an


Add New Macrofilter

Step Macrofilter

Separates several filters that can be used in many places of a program. The state of the filters is preserved between invocations.

This macrofilter structure is used most often. It assures that the program is clean and organized.

Macrofilter Name: ProcessImage

Advanced Options «

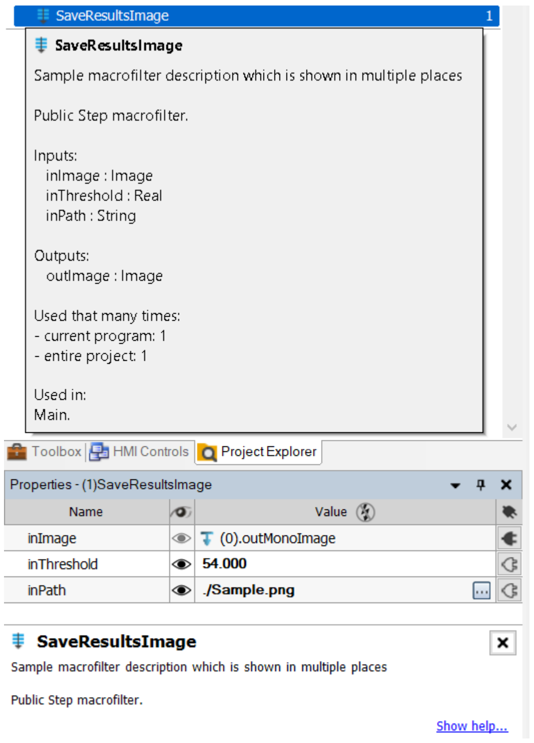
Tip: Sample tooltip

Module: Program

Access: Public

Optimize Memory: (if checked, filters' output data are released after last iteration)

Macrofilter creation with advanced options.


SaveResultsImage 1

SaveResultsImage

Sample macrofilter description which is shown in multiple places

Public Step macrofilter.

Inputs:
  inImage : Image
  inThreshold : Real
  inPath : String

Outputs:
  outImage : Image

Used that many times:
- current program: 1
- entire project: 1

Used in:
Main.

Properties - (1)SaveResultsImage

Name	Value
inImage	(0).outMonoImage
inThreshold	54.000
inPath	./Sample.png

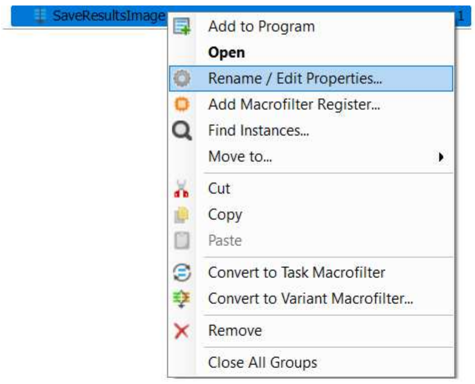
SaveResultsImage

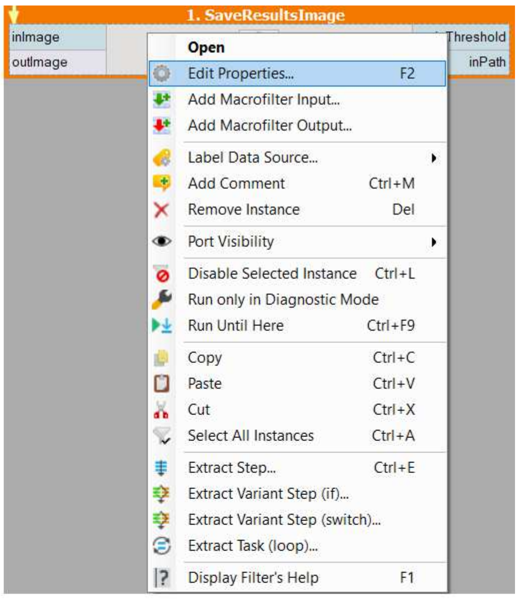
Sample macrofilter description which is shown in multiple places

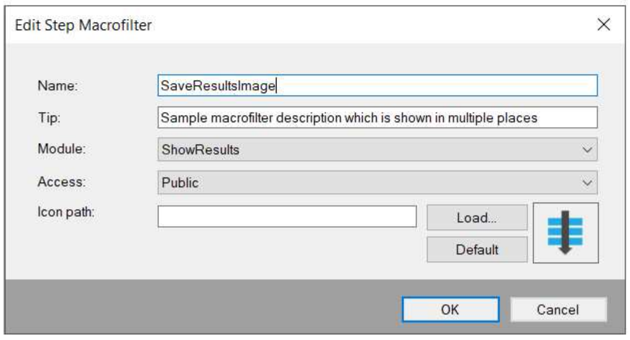
Public Step macrofilter.

View showing a sample tooltip.

Right clicking a macrofilter and selecting *Edit properties...* allows to modify their properties, including access, after their creation.


SaveResultsImage
Add to Program
Open
Rename / Edit Properties...
Add Macrofilter Register...
Find Instances...
Move to...
Cut
Copy
Paste
Convert to Task Macrofilter
Convert to Variant Macrofilter...
Remove
Close All Groups




Edit Step Macrofilter

Name: SaveResultsImage

Tip: Sample macrofilter description which is shown in multiple places

Module: ShowResults

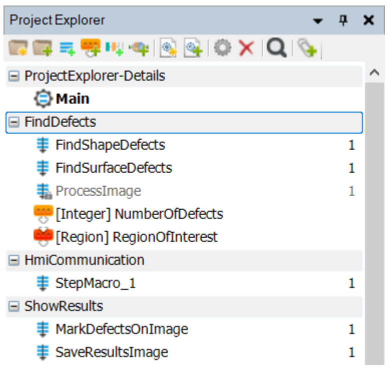
Access: Public

Icon path:

Ways to access the editing window. The option to change the icon is highlighted

The image below shows a structure of an example program. The macrofilters have been grouped into two Modules. Module *FindDefects* has macrofilters related to defect detection and a global parameter used by the macrofilters. Notice how the *ProcessImage* macrofilter is grayed out. It indicates it is a *private* macrofilter (here it is used by *FindShapeDefects*). *ProcessImage* cannot be used outside its module.

The other module has macrofilters related to showing and storing the results of the inspection.



Example program structure with macrofilters grouped into modules.

Here are some guidelines on how to use modules in such situations:

- Create a separate module for each set of related, standard macrofilters.
- Give each module a unique and clear name.
- Use the English language and follow the same naming conventions as in the native filters.
- Create the common macrofilters in such a way, that they do not have to be modified between different projects and only the values of their parameters have to be adapted.
- If some of the macrofilters are intended as implementation only (not to be used from other modules), mark them as *private*.

It is important to note that modules containing filters interfacing with the HMI should not be shared between programs. Every filter port connected to the HMI has a unique identifier. Those identifiers vary between programs - ports of the same filter in different programs will have different identifiers.

Generally it is a good practice to create a separate module for all things related to the HMI. That way every other module can be shared between programs without any problems.

Importing Modules

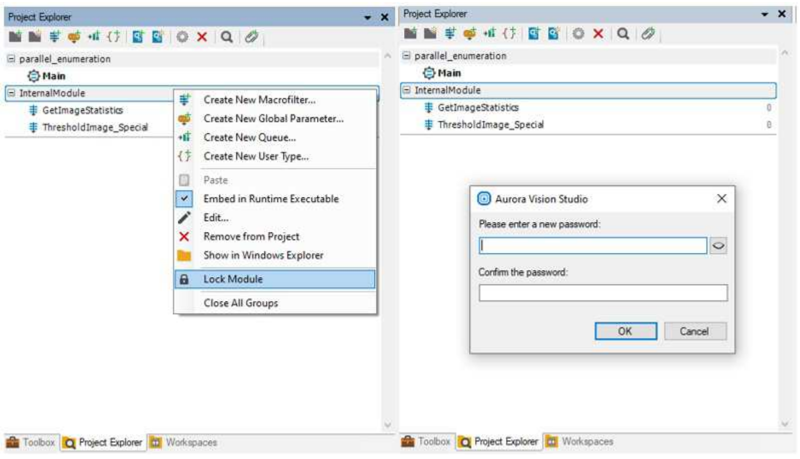
If you want to use a module which had been created before click the *Import Existing Module...* . This will open a window in which you can select modules to add. Now the path to the module will be linked to the project. Similarly to creating a new module you can choose whether the path to it will be relative or absolute.

Remember that modules are separate files and as such can be modified externally. This is especially important with modules which are shared between multiple projects at the same time.

See also: Trick: INI File as a Module Not Exported to AVEXE.

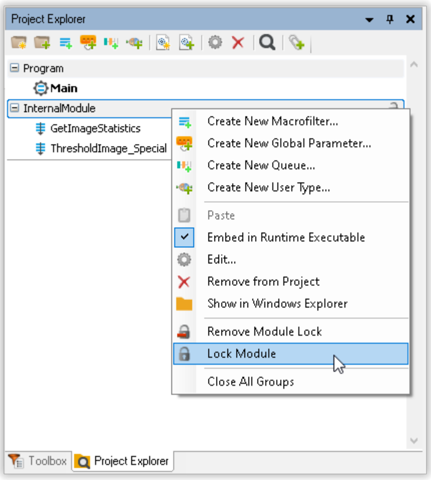
Locking Modules

Sometimes users would like to hide the contents of some of their macrofilter to protect them against unauthorized access. They can do this by placing them inside a module and locking it. To do this it is necessary to right click on the module in the Project Explorer and select Lock Module option. User will be then prompted to provide a password for this specific module.

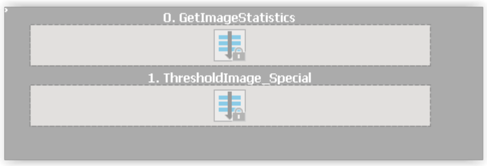


Adding lock to the module.

After creating password user should see an open lock icon next to the module name in the Project Explorer. You can work with this module exactly the same way as with any other module. What distinguishes it from other modules is that it can be locked, which will make checking implementation of macrofilters included inside not possible. Such modules will be similar to filters in this matter. This operation will also encrypt the avlib file. It is important to lock the module every time you finish working on it.



Locking module in project explorer.



Macrofilters from locked modules in the program.

In order to unlock a module you need to select the Unlock Module option in the ProjectExplorer and enter the correct password. To completely remove module locking feature from a selected module, use the Remove Module Lock option.

Keyboard Shortcuts

Introduction

Many of Aurora Vision Studio actions can be invoked with keyboard shortcuts. Most of them have default shortcuts - such as copying with Ctrl+C, pasting with Ctrl+V, finding elements with Ctrl+F, saving with Ctrl+S, navigating with arrow keys etc.
In some controls - e.g. formula editor - there are present commonly used keyboard shortcuts for text editors: navigating through whole words with Ctrl+Left/Right Arrow, selecting consecutive letters/lines (Shift+Arrow keys), selecting whole words (Ctrl+Shift+Left/Right Arrow), increasing indents with Tab, decreasing them with Shift+Tab etc.

In this article you can find listed all of the less-known shortcuts.

Table of Contents

Shortcuts Table

Command	Shortcut
Program Editor Back to Top	
Run program	F5
Run program with Aurora Vision Executor	Ctrl+F5
Run program until selected point	Ctrl+F9
Stop program	Shift+F5
Pause program at current executing step	Ctrl+Alt+Pause
Iterate program	F6
Iterate current macrofilter	Ctrl+F10
Step over	F10
Step into	F11
Step out	Shift+F11
Insert new filter instance	Ctrl+T Ctrl+Space
Load recent project	Alt+Number
Rename currently open macrofilter	F2
View program statistics	F8
Toggles breakpoint for currently selected filter and macrofilter outputs block	F9
Copy element	Ctrl+Insert Ctrl+C
Paste element	Shift+Insert Ctrl+V
Change currently selected macrofilter	Ctrl+Number
Navigate to home macrofilter	Alt+Home
Navigate to parent macrofilter	Shift+Enter
Open next macrofilter in history	Alt+Right Arrow
Open previous macrofilter in history	Alt+Left Arrow
Extract step from selected filter/filters	Ctrl+E
Add new Formula instance to current program	Ctrl+Shift+E
Add new Comment instance to current program	Ctrl+Shift+K
Create step macrofilter and add instance	Ctrl+Shift+S
Create task macrofilter and add instance	Ctrl+Shift+T
Create variant macrofilter and add instance	Ctrl+Shift+V
Remove currently opened macrofilter with all its instances	Ctrl+Shift+R

Undock currently opened macrofilter	Ctrl+U
Enable/Disable selected filter instance/instances	Ctrl+L
Add new/Edit existing filter comment	Ctrl+M
Copy currently moving instance	Hold Ctrl while moving instance
Navigate and select elements instances	Shift+Up/Down/Home/End
Move selected filters	Alt+Up Alt+Down
HMI Designer Back to Top	
Move currently selected control	Arrows
Move currently selected control to the edge of its container	Ctrl+Arrows
Resize currently selected control	Shift+Arrows
Resize currently selected control to the edge of its container	Ctrl+Shift+Arrows
Properties Control Back to Top	
Hide/Show selected property	Ctrl+H
Restore default value of selected property	Ctrl+D
Enable/Disable port in current macrofilter variant	Ctrl+E
Set/Unset optional value	Ctrl+P
Enable/Disable port in current macrofilter variant	Ctrl+E
Change scale on property value slider from 1 to 10	Hold Shift while modifying value with slider
Change scale on property value slider from 1 to 0.1	Hold Ctrl while modifying value with slider
Change scale on property value slider from 1 to 0.01	Hold Ctrl+Shift while modifying value with slider
Insert a new line into the text	Shift+Enter
3D View Back to Top	
Show bounding box	B
Show grid	G
Increase point size	]
Decrease point size	[
Move view up	Q Up Arrow
Move view down	Z Down Arrow
Move view left	A Left Arrow
Move view right	D Up Arrow
Zoom in	W Page Up
Zoom out	S Page Down
Decrease rotation angle	Hold CTRL while rotating view
Deep Learning Editors Back to Top	
Change current class	0 - 9
Move one image up	Page Up
Move one image down	Page Down

Save current state of the model	Ctrl+S
Automatic training	Alt+F2
Generate the report	Alt+F3
Deep Learning Editors - Anomalies Detection only Back to Top	
Change the current image's annotation	Space
Label image as good and go to the next one	G
Label image as bad and go to the next one	B
Deep Learning Editors - Object Classification only Back to Top	
Remove selected ROI	Delete
Deep Learning Editors - Instance Segmentation only Back to Top	
Remove single instance	Delete
Add new instance	Space
Split the instances into separate blobs	S
Deep Learning Editors - Point Location only Back to Top	
Remove selected point	Delete

Working with 3D data

Introduction

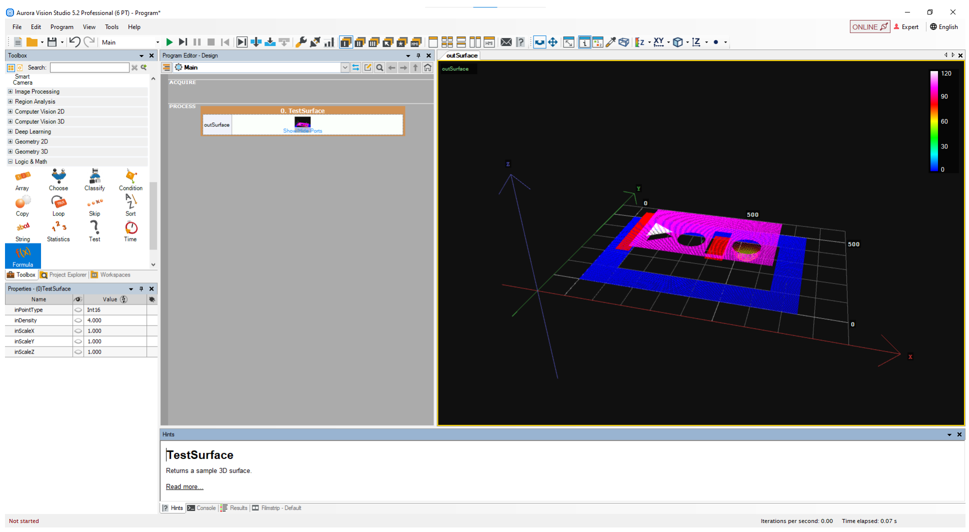
This article summarizes how to work with 3D data previews. Aurora Vision Studio allows you to work not only with images or conventional data types, but also with point clouds of following types:

- Surface
- Point3DGrid
- Point3DArray

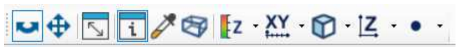
Please note that the term *surface* may be used in this article to denote a point cloud of Surface type.

Toolbar

The toolbar appears the moment a point cloud is previewed:



The location of 3D tools in Aurora Vision Studio.



3D tools available in the toolbar.

Button (Tool)	Description of the tool
Rotate	This button permits to rotate a point cloud around the rotation center point.
Pan	This button permits to move the rotation center point.
Resetting view	This button permits to revert changes made by rotating or panning a point cloud.
Probe Point Coordinates	This button permits to obtain information about coordinates of a point.
Bounding Boxes	This button permits to display the bounding box of a point cloud.
Coloring Mode	This button permits to switch between different coloring modes which are divided into 4 categories: • Solid (whole point cloud is of a single color) • Along X-axis (greater values along the X-axis are colored according to the colors' scale bar in the top right corner of the preview) • Along Y-axis (greater values along the Y-axis are colored according to the colors' scale bar in the top right corner of the preview) • Along Z-axis (greater values along the Z-axis are colored according to the colors' scale bar in the top right corner of the preview)
Grid Mode	This button permits to display a grid plane to your liking. There are also 4 modes available: • None (no grid) • XY (the grid is displayed in XY plane) • XZ (the grid is displayed in XZ plane) • YZ (the grid is displayed in YZ plane)
View Projection Mode	This button permits to display a point cloud in 4 different modes: • Perspective (default mode) • Up (overhead view – showing a point cloud from above – looking down at XY plane) • Back (view from the back – looking at XZ plane) • Left (side view – looking at YZ plane)
World Orientation	This button permits to display a coordinate system in 4 different modes: • Right Handed Z Up • Right Handed Y Up • Left Handed Z Up • Left Handed Y Up
Point size	This button permits to control the size of a single point within a point cloud. There are 5 possible sizes: • Very small • Small • Medium • Large • Very large

Using a computer mouse

Aurora Vision Studio also allows you to control a 3D preview with your computer mouse. You can perform some of the above features to analyze the point cloud without a necessity to find and switch between buttons in the toolbar.

If you click **Left Mouse Button** on a preview, it works as if you used the Rotate button, so you can rotate a point cloud around a fixed center point until the button is released.

If you click **Right Mouse Button** on a point cloud, a dropdown list with all features described in the previous section appears.

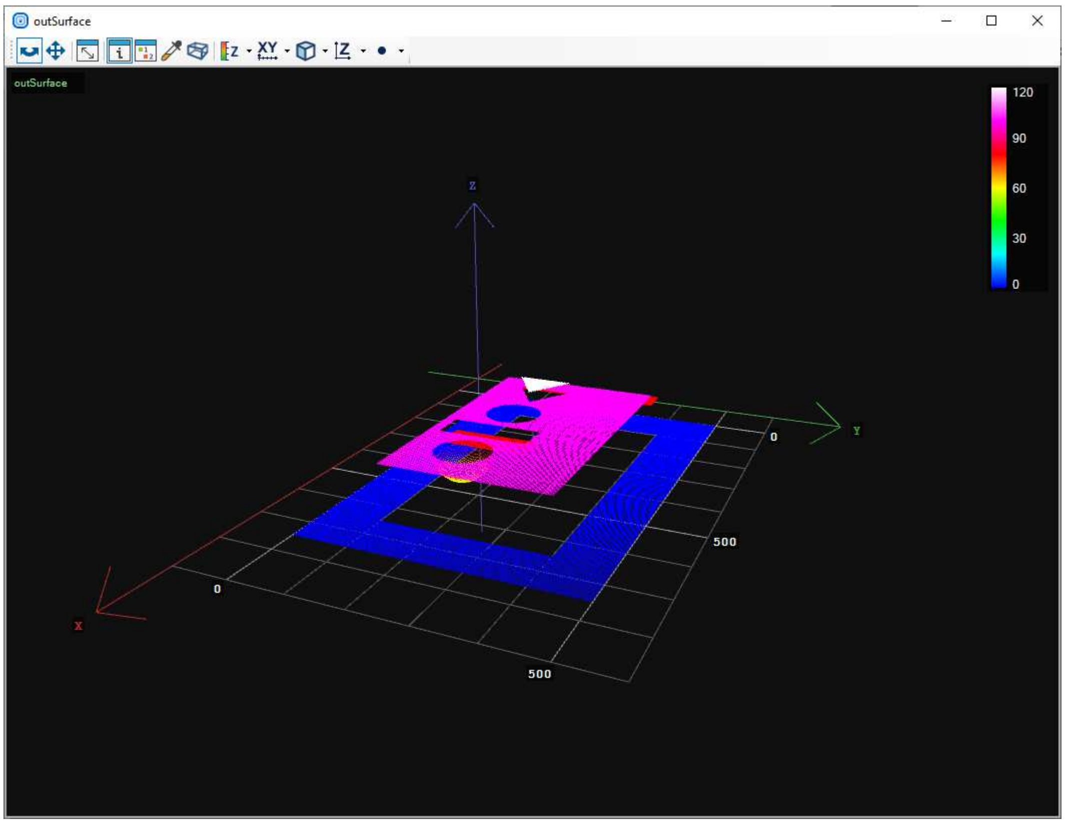
If you use **Mouse Wheel** on a point cloud, you can zoom it in and out.

If you press **Mouse Wheel** on a point cloud, you can move the rotation center point of the coordinate system (works like the "Pan" button).

Preview

When you drag and drop the output, which is of 3D data type, and you have already previewed some images, string or real values, please note that it can be only previewed in a separate view. It is not possible to display an image and a point cloud in the very same view.

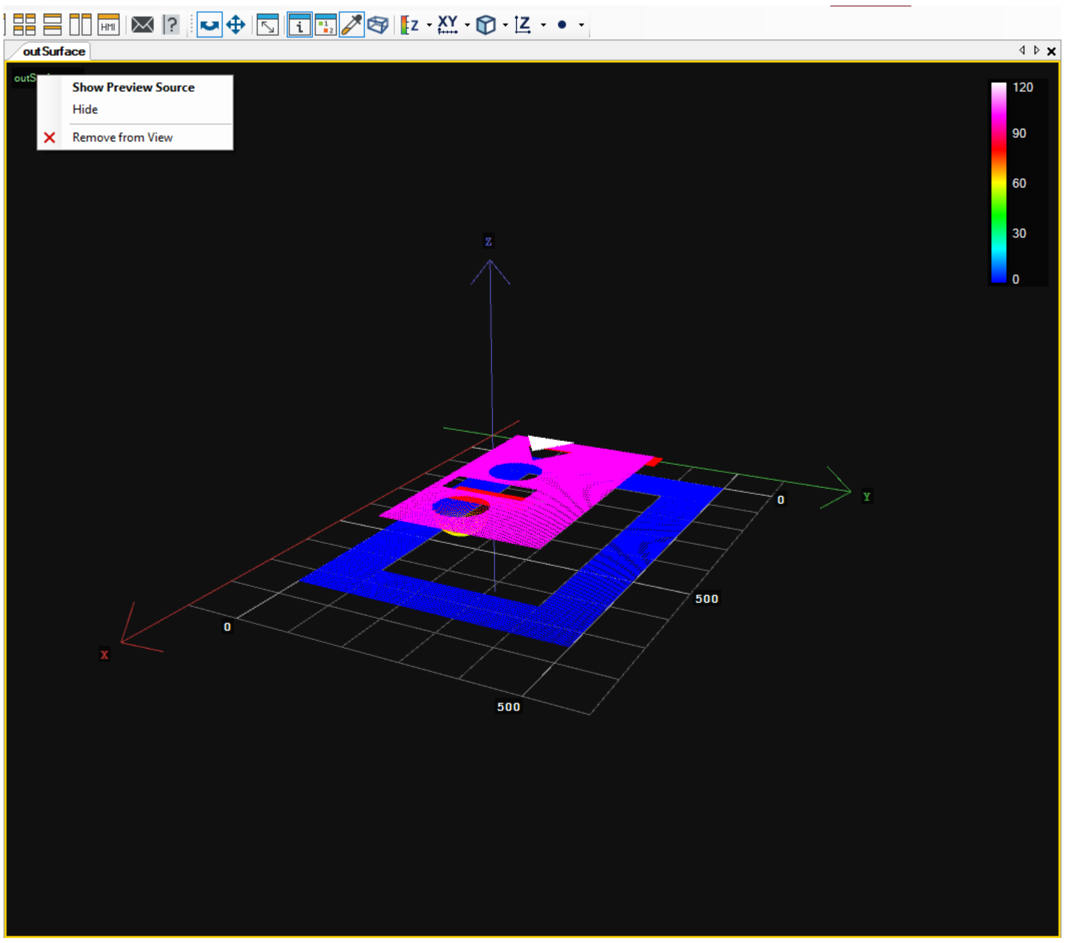
While analyzing 3D data, please pay attention to the colors' scale bar in the top right corner of the preview:



Colors' scale bar.

It might be helpful if you want to estimate the coordinate value based on color.

You can also hide or remove surfaces from a view. To do it, you have to right-click on the information about a preview in the top left corner of the view:



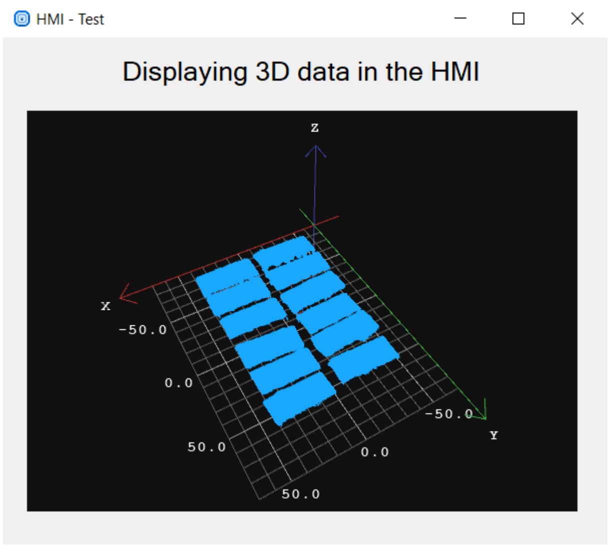
Dropdown list with possible actions on previews.

As it is shown in the above image, you can:

- Show Preview Source - it will indicate the filter in the Program Editor, of which the output has been previewed
- Hide - it will hide only indicated surface without removing the preview
- Remove from View - it will delete the preview, leaving an empty view

HMI

3D data can be displayed in the HMI as well - using the View3DBox control:



View3DBox HMI control - a point cloud is displayed in the same manner as in the preview.

It is worth mentioning that in this HMI control you can also rotate and change the rotation center of the point cloud. For more details about designing and using HMI, please refer to HMI Controls.

Creating Deep Learning Model

Contents:

Introduction

Deep Learning editors are dedicated graphical user interfaces for DeepModel objects (which represent training results). Each time a user opens such an editor, he is able to add or remove images, adjust parameters and perform new training.

Since version 4.10 it is also an option to open a Deep Learning Editor as a stand-alone application, which is especially useful for re-training models with new images in production environment.

Requirements:

- A DeepLearning license is required to use Deep Learning editors and filters.
- Deep Learning Service must be up and running to perform model training.

Currently available deep learning tools are:

1. **Anomaly Detection** – for detecting unexpected object variations; trained with sample images marked simply as good or bad.
2. **Feature Detection** – for detecting regions of defects (such as surface scratches) or features (such as vessels on medical images); trained with sample images accompanied with precisely marked ground-truth regions.
3. **Object Classification** – for identifying the name or the class of the most prominent object on an input image; trained with sample images accompanied with the expected class labels.
4. **Instance Segmentation** – for simultaneous location, segmentation and classification of multiple objects in the scene; trained with sample images accompanied with precisely marked regions of each individual object.
5. **Point Location** – for location and classification of multiple key points; trained with sample images accompanied with marked points of expected classes.
6. **Read Characters** – for location and classification of multiple characters; this tool uses a pretrained model and cannot be trained, so it is not described in this article
7. **Object Location** – for location and classification of multiple objects; trained with sample images accompanied with marked bounding rectangles of expected classes.

Technical details about these tools are available at Machine Vision Guide: Deep Learning while this article focuses on the training graphical user interface.

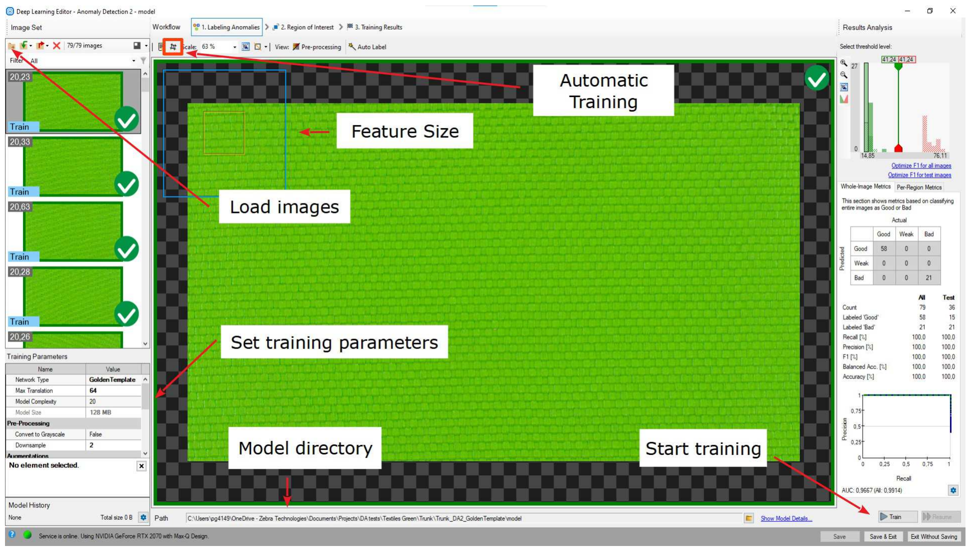
Workflow

You can open a **Deep Learning Editor** via:

- a filter in Aurora Vision Studio:
  1. Place the relevant DL filter (e.g. DL_DetectFeatures or DL_DetectFeatures_Deploy) in the Program Editor.
  2. Go to its Properties.
  3. Click on the button next to the **inModelDirectory** or **inModelId.ModelDirectory** parameter.
- a standalone **Deep Learning Editor** application:
  1. Open a standalone **Deep Learning Editor** application (which can be found in the **Aurora Vision Studio installation folder** as "DeepLearningEditor.exe", in Aurora Vision folder in **Start menu** or in Aurora Vision Studio application in **Tools menu**).
  2. Choose whether you want to create a new model or use an existing one:
     - **Creating a new model**: Select the relevant tool for your model and press OK, then select or create a new folder where files for your model will be contained and press OK.
     - **Choosing existing model**: Navigate to the folder containing your model files – either write a path to it, click on the button next to field to browse to it or select one of the recent paths if there are any; then press OK.

The Deep Learning model preparation process is usually split into the following steps:

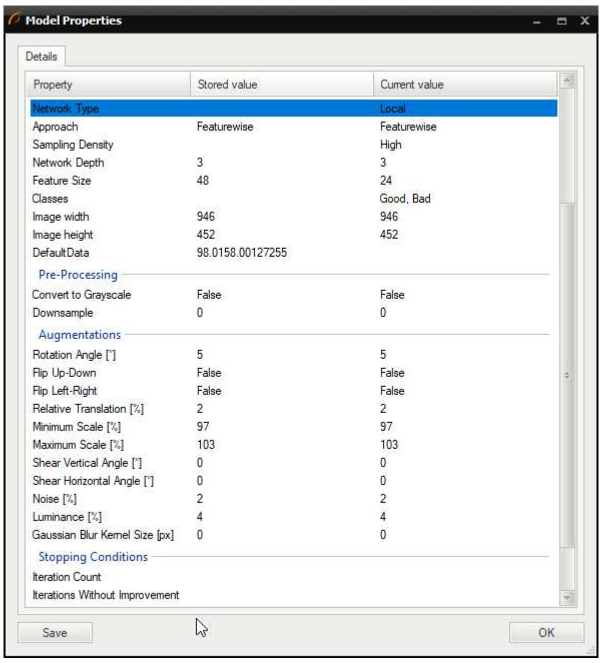
1. **Loading images** – load training images from disk
2. **Labeling images** – mark features or attach labels on each training image
3. **Setting the region of interest (optional)** – select the area of the image to be analyzed
4. **Adjusting training parameters** – select training parameters, preprocessing steps and augmentations specific for the application at hand
5. **Training the model and analyzing results**



Overview of a Deep Learning Editor.

Important features:

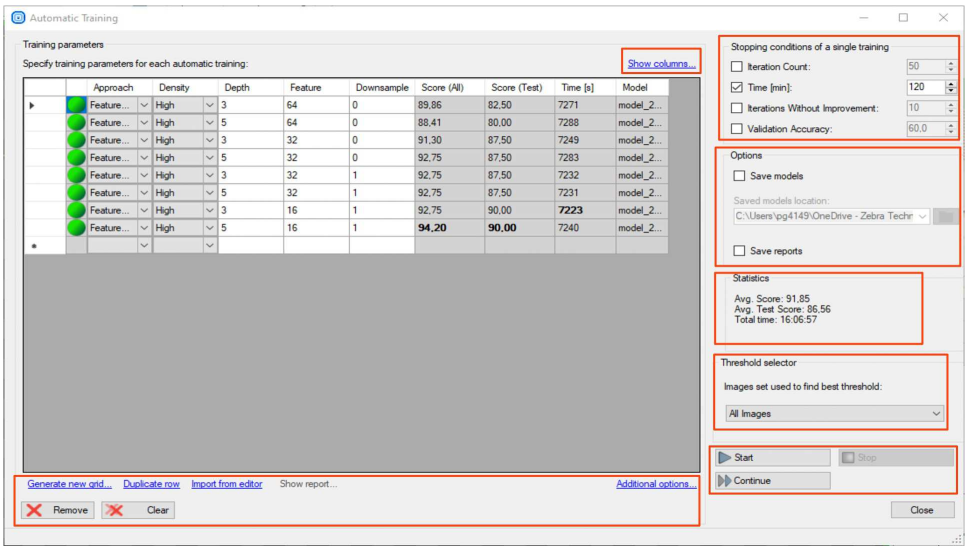
- **Pre-processing button** – located in the top toolbar; allows you to see the changes applied to a training image e.g. grayscale or downsampling.
- **Current Model directory** – located in the bottom toolbar; allows you to switch a model being in another directory or to simply see which model you are actually working on.
- **Show Model Details button** – located next to the previous control; allows you to display information on the current model and save them to a file.



Property	Stored value	Current value
Network Type		Local
Approach	Featurewise	Featurewise
Sampling Density		High
Network Depth	3	3
Feature Size	48	24
Classes		Good, Bad
Image width	946	946
Image height	452	452
DefaultData	98.0158.00127255	
Pre-Processing		
Convert to Grayscale	False	False
Downsample	0	0
Augmentations		
Rotation Angle [°]	5	5
Flip Up-Down	False	False
Flip Left-Right	False	False
Relative Translation [%]	2	2
Minimum Scale [%]	97	97
Maximum Scale [%]	103	103
Shear Vertical Angle [°]	0	0
Shear Horizontal Angle [°]	0	0
Noise [%]	2	2
Luminance [%]	4	4
Gaussian Blur Kernel Size [px]	0	0
Stopping Conditions		
Iteration Count		
Iterations Without Improvement		

- **Train & Resume buttons** – allow you to start training or resume it in case you have changed some of training parameters.
- Saving buttons:
  - **Save** – saves the current model in chosen location.
  - **Save & Exit** – saves the model and then exits the Deep Learning Editor.
  - **Exit Without Saving** – exits the editor, but the model is not being saved.

Open automatic training window button – allows you to prepare a training series for different parameters. If you are not sure, which parameter settings will give you the best result, you can prepare a combination for each value to compare the results. The test parameters can be prepared automatically with **Generate new grid** or manually entered. After setting the parameters you need to start the test. The settings and the results are shown in the grid, one row for one model.


Automatic Training

Training parameters

Specify training parameters for each automatic training:

		Approach	Density	Depth	Feature	Downsample	Score (All)	Score (Test)	Time [s]	Model
▸	●	Feature...	High	3	64	0	89,86	82,50	7271	model_2...
	●	Feature...	High	5	64	0	88,41	80,00	7288	model_2...
	●	Feature...	High	3	32	0	91,30	87,50	7249	model_2...
	●	Feature...	High	5	32	0	92,75	87,50	7283	model_2...
	●	Feature...	High	3	32	1	92,75	87,50	7232	model_2...
	●	Feature...	High	5	32	1	92,75	87,50	7231	model_2...
	●	Feature...	High	3	16	1	92,75	90,00	**7223**	model_2...
	●	Feature...	High	5	16	1	**94,20**	**90,00**	7240	model_2...
*										

Stopping conditions of a single training
- Iteration Count: 50
- [✓] Time [min]: 120
- Iterations Without Improvement: 10
- Validation Accuracy: 60,0

Options
- Save models
- Saved models location: C:\Users\pg4149\OneDrive - Zebra Techn
- Save reports

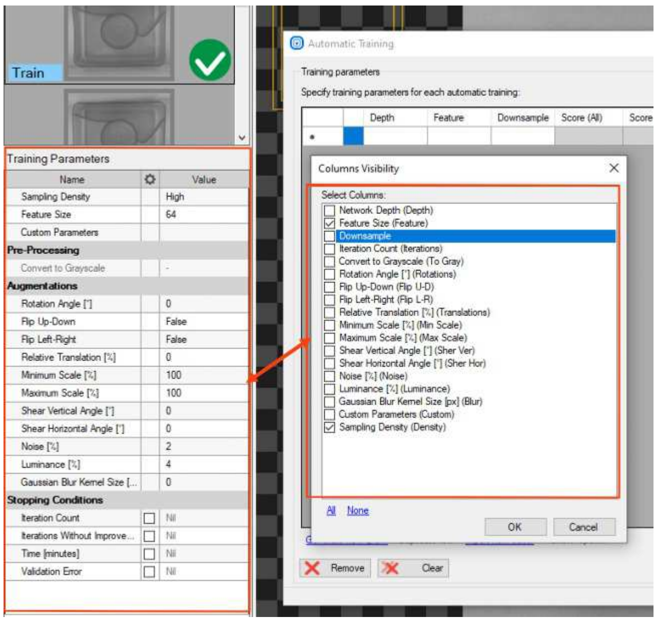
Statistics

Avg. Score: 91,85
Avg. Test Score: 86,56
Total time: 16:06:57

Threshold selector

Images set used to find best threshold: All Images

Show columns – hides / shows model parameters you will use in your test. The view is common for all deep learning tools. To create an appropriate grid search, choose these parameters which are correct for the used tool (the ones you can see in Training Parameters). For DL_DetectAnomalies2 choose the network type first to see the appropriate parameters.

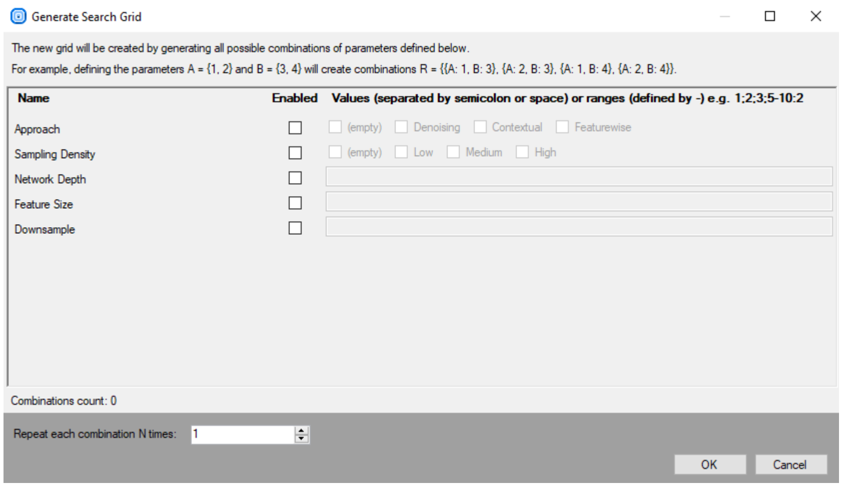

Training Parameters

Name	Value
Sampling Density	High
Feature Size	64
Custom Parameters	
Pre-Processing	
Convert to Grayscale	-
Augmentations	
Rotation Angle [°]	0
Flip Up-Down	False
Flip Left-Right	False
Relative Translation [%]	0
Minimum Scale [%]	100
Maximum Scale [%]	100
Shear Vertical Angle [°]	0
Shear Horizontal Angle [°]	0
Noise [%]	2
Luminance [%]	4
Gaussian Blur Kernel Size [...	0
Stopping Conditions	
Iteration Count	Nil
Iterations Without Improve...	Nil
Time [minutes]	Nil
Validation Error	Nil

Columns Visibility

Select Columns:
- Network Depth (Depth)
- [✓] Feature Size (Feature)
- Downsample
- Iteration Count (Iterations)
- Convert to Grayscale (To Gray)
- Rotation Angle [°] (Rotations)
- Flip Up-Down (Flip U-D)
- Flip Left-Right (Flip L-R)
- Relative Translation [%] (Translations)
- Minimum Scale [%] (Min Scale)
- Maximum Scale [%] (Max Scale)
- Shear Vertical Angle [°] (Sher Ver)
- Shear Horizontal Angle [°] (Sher Hor)
- Noise [%] (Noise)
- Luminance [%] (Luminance)
- Gaussian Blur Kernel Size [px] (Blur)
- Custom Parameters (Custom)
- [✓] Sampling Density (Density)

Generate new grid – prepares the search grid for the given parameters. Only parameters chosen in **Show columns** are available. The values should be separated with *;* sign.


Generate Search Grid

The new grid will be created by generating all possible combinations of parameters defined below.
For example, defining the parameters A = {1, 2} and B = {3, 4} will create combinations R = {{A: 1, B: 3}, {A: 2, B: 3}, {A: 1, B: 4}, {A: 2, B: 4}}.

Name	Enabled	Values (separated by semicolon or space) or ranges (defined by -) e.g. 1;2;3;5-10:2
Approach	☐	☐ (empty) ☐ Denoising ☐ Contextual ☐ Featurewise
Sampling Density	☐	☐ (empty) ☐ Low ☐ Medium ☐ High
Network Depth	☐	
Feature Size	☐	
Downsample	☐	

Combinations count: 0

Repeat each combination N times: 1

Duplicate rows – duplicates a training parameters configuration. If the parameters inside the row won't be modified, this model will be trained twice.

Import from editor – copies the training parameters from **Editor Window** to the last search grid row.

Show report – shows the report for the chosen model (the chosen row). This option is available only if you choose **Save Reports** before starting the training session.

Additional options

- **Export grid to CSV file** – exports the grid of training parameters to a CSV file.
- **Import grid from CSV file** – imports the grid of training parameters from a CSV file.

Remove – removes a chosen training configuration.

Clear – clears the whole search grid.

Stopping conditions of a single training – determines when a single training stops.

- Iteration Count
- Time
- Iterations without improvement
- Validation Accuracy

Options

- **Save models** – saves each trained model in the defined folder.
- **Save reports** – saves a report for each trained model in the folder defined for saving models.

Statistics – shows statistics for all trainings.

- **Avg. Score** – shows average score of all trained models for all images.
- **Avg. Test Score** – shows average score of all trained models for test images.
- **Total time** – sums up time of each training.

Threshold selector – choses for which image group the best threshold is searched.

- **All Images**
- **Test Images**

Start – starts the training series with the first defined configuration.

Stop – stops the training series.

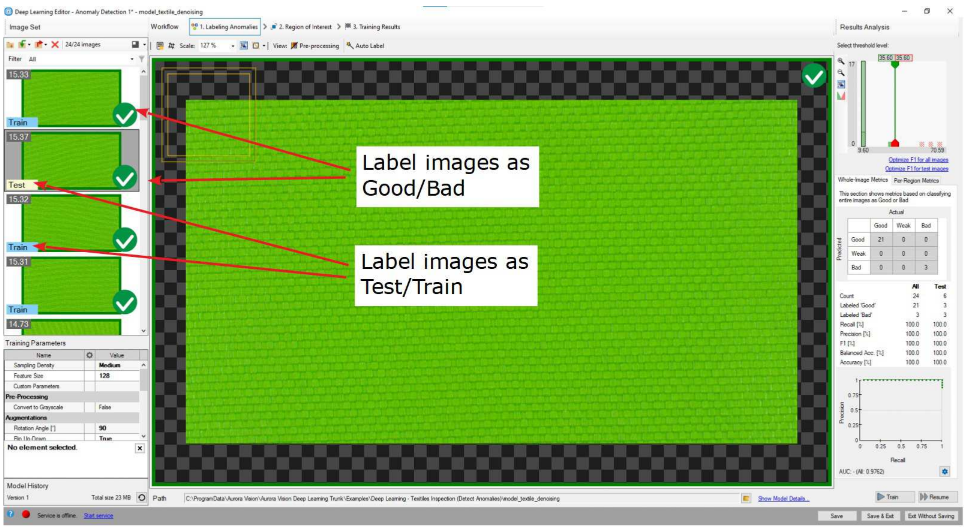
Continue – continues the stopped training series with the next configuration of the parameters.

Detecting anomalies 1

In this tool the user only needs to mark which images contain correct cases (good) or incorrect ones (bad).

1. Marking Good and Bad samples and dividing them into Test and Training data.

Click on a question mark sign to label each image in the training set as **Good** or **Bad**. Green and red icons on the right side of the training images indicate to which set the image belongs to. Alternatively, you can use **Add images and mark...**. Then divide the images into **Train** or **Test** by clicking on the left label on an image. Remember that all bad samples should be marked as **Test**.



Training Parameters

Name	Value
Sampling Density	Medium
Feature Size	128
Custom Parameters	
Pre-Processing	
Convert to Grayscale	False
Augmentations	
Rotation Angle [°]	90
Flip Up-Down	True

No element selected.

Model History

Version 1 Total size 23 MB

Path C:\ProgramData\Aurora Vision\Aurora Vision Deep Learning Trunk\Examples\Deep Learning - Textiles Inspection (Detect Anomalies)\model_textile_denoising

Service is offline. Start service

This section shows metrics based on classifying entire images as Good or Bad

Predicted \ Actual	Good	Weak	Bad
Good	21	0	0
Weak	0	0	0
Bad	0	0	3

	All	Test
Count	24	6
Labeled 'Good'	21	3
Labeled 'Bad'	3	3
Recall [%]	100.0	100.0
Precision [%]	100.0	100.0
F1 [%]	100.0	100.0
Balanced Acc. [%]	100.0	100.0
Accuracy [%]	100.0	100.0

AUC: - (All: 0.9762)

Labeled images in Deep Learning Editor.

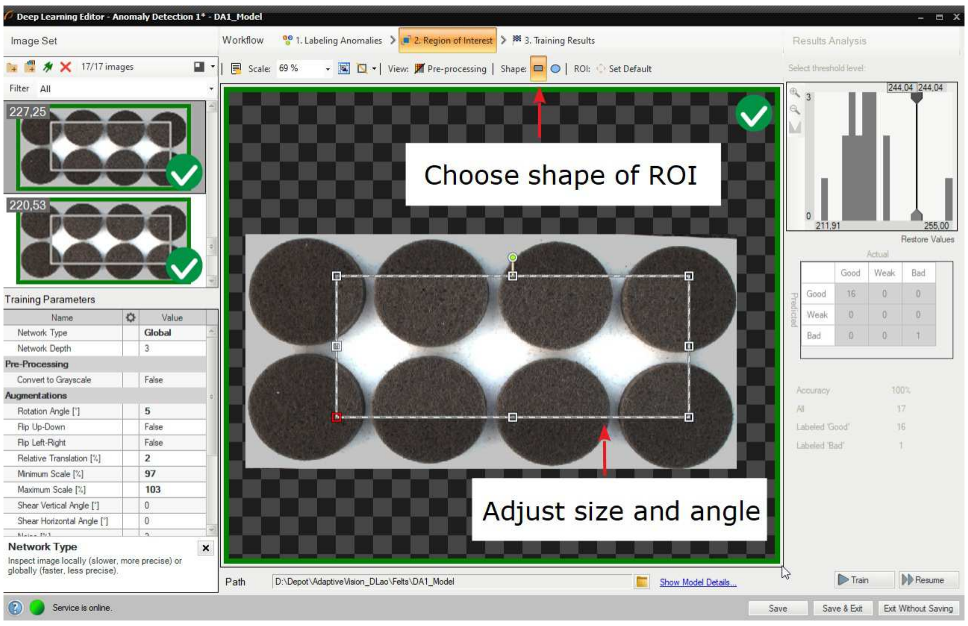
2. Configuring augmentations

It is usually recommended to add some additional sample augmentations, especially when the training set is small. For example, the user can add additional variations in pixels intensity to prepare the model for varying lighting conditions on the production line. Refer to "Augmentation" section for detailed description of parameters: Deep Learning – Augmentation.

3. Reducing region of interest

Reduce region of interest to focus only on the important part of the image. Reducing region of interest will speed up both training and inference.

Please note that the Region of Interest in this tool is the same for each image in a training set and it cannot be adjusted individually. As a result, this Region of Interest is automatically applied during execution of a model, so a user has no impact on its size or shape in the Program Editor.



Training Parameters

Name	Value
Network Type	Global
Network Depth	3
Pre-Processing	
Convert to Grayscale	False
Augmentations	
Rotation Angle [°]	5
Flip Up-Down	False
Flip Left-Right	False
Relative Translation [%]	2
Minimum Scale [%]	97
Maximum Scale [%]	103
Shear Vertical Angle [°]	0
Shear Horizontal Angle [°]	0

Network Type
Inspect image locally (slower, more precise) or globally (faster, less precise).

Path: D:\Depot\AdaptiveVision_DLao\Felts\DA1_Model

Predicted \ Actual	Good	Weak	Bad
Good	16	0	0
Weak	0	0	0
Bad	0	0	1

Accuracy	100%
All	17
Labeled 'Good'	16
Labeled 'Bad'	1

By default region of interest contains the whole image.

4. Setting training parameters

- **Sampling density** (for the Featurewise approach in DL_DetectAnomalies1 and in DL_DetectAnomalies2 only) – selecting from: Low, Medium and High.
- **Feature size** (for the Local type only) – the width of the inspection window. In the top left corner of the editor a small rectangle visualizes the selected feature size.
- **Stopping conditions** – define when the training process should stop.

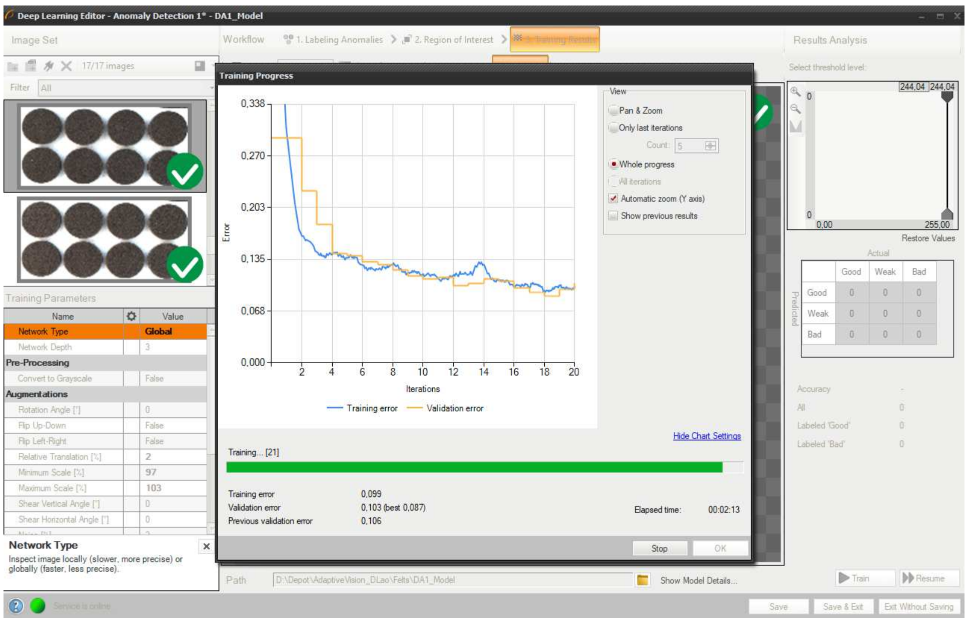
For more details read Deep Learning – Setting parameters.

5. Performing training

During training, two figures are successively displayed: training error and validation error. Both charts should have a similar pattern.

More detailed information is displayed on the chart below:

- current training statistics (training and validation),
- number of processed samples (depends on the number of images and feature size),
- elapsed time.



Training error	0,099
Validation error	0,103 (best 0,087)
Previous validation error	0,106
Elapsed time:	00:02:13

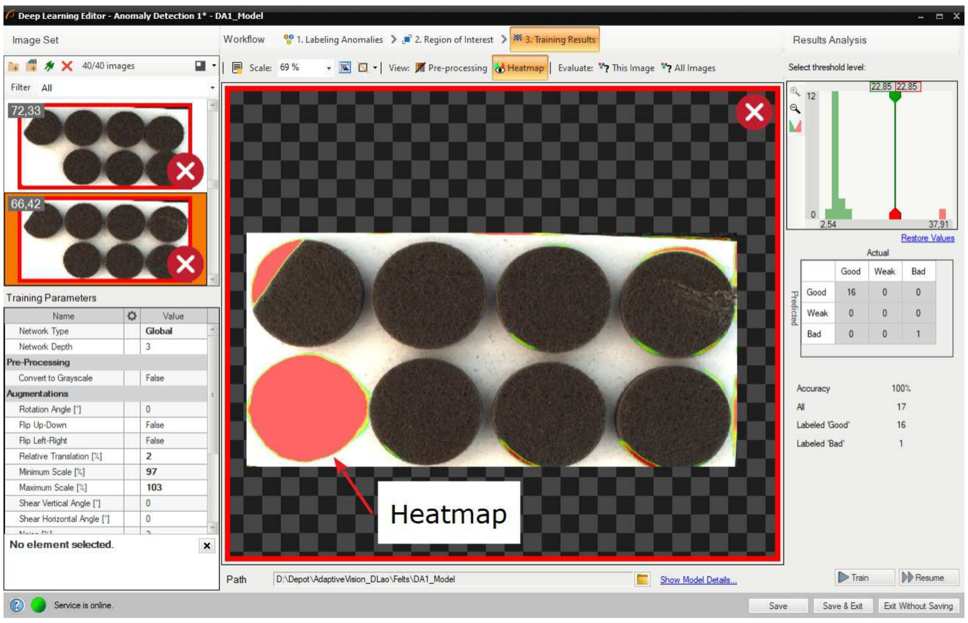
Training process consists of computing training and validation errors.

Training process can take its time depending on the selected stopping criteria and available hardware. During this time training can be finished manually anytime by clicking Stop button. If no model is present (the first training attempt) model with the best validation accuracy will be saved. Consecutive training attempts will prompt user about old model replacement.

6. Analyzing results

The window shows a histogram of sample scores and a heatmap of found defects. The left column contains a histogram of scores computed for each image in the training set. Additional statistics are displayed below the histogram.

To evaluate trained model, **Evaluate: This Image** or **Evaluate: All Images** buttons can be used. It can be useful after adding new images to the data set or after changing the area of interest.



Training Parameters

Name		Value
Network Type		Global
Network Depth		3
Pre-Processing		
Convert to Grayscale		False
Augmentations		
Rotation Angle [°]		0
Flip Up-Down		False
Flip Left-Right		False
Relative Translation [%]		2
Minimum Scale [%]		97
Maximum Scale [%]		103
Shear Vertical Angle [°]		0
Shear Horizontal Angle [°]		0

No element selected.

Actual

Predicted	Good	Weak	Bad
Good	16	0	0
Weak	0	0	0
Bad	0	0	1

Accuracy	100%
All	17
Labeled 'Good'	16
Labeled 'Bad'	1

Path D:\Depot\AdaptiveVision_DLao\Felts\DA1_Model

Service is online.

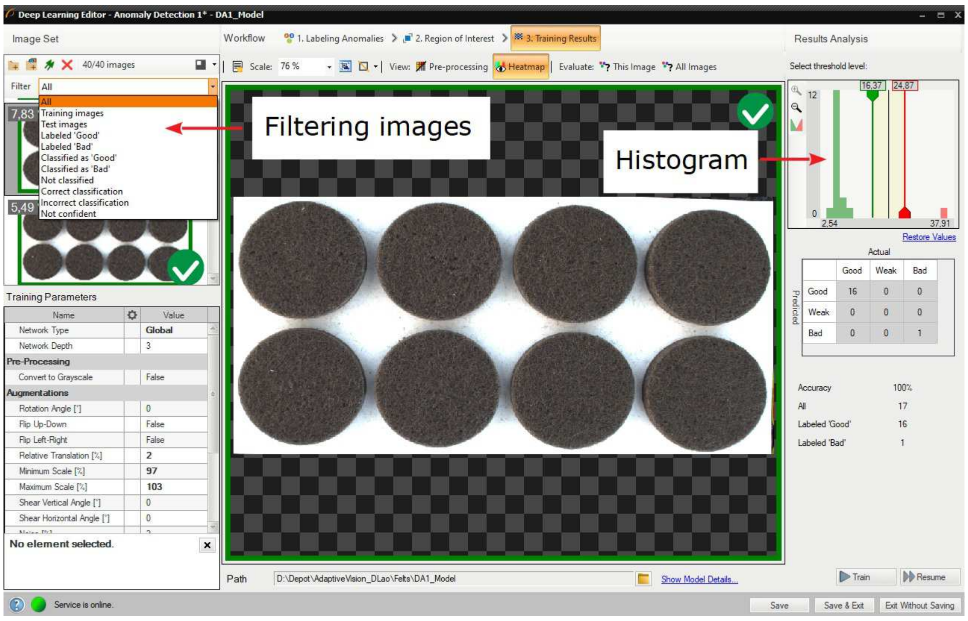
Heatmap presents the area of possible defects.

After training two border values are computed:

1. Maximum good sample score (T1) – all values from 0 to T1 are marked as Good.
2. Minimum bad sample score (T2) – all values greater than T2 are marked as Bad.

All scores between T1 and T2 are marked as "Low quality". Results in this range are uncertain and may be not correct. Filters contain an additional output **outIsConfident** which determines the values which are not in the T1-T2 range.

After evaluation additional filtering options may be used in the the list of training images.



Training Parameters

Name		Value
Network Type		Global
Network Depth		3
Pre-Processing		
Convert to Grayscale		False
Augmentations		
Rotation Angle [°]		0
Flip Up-Down		False
Flip Left-Right		False
Relative Translation [%]		2
Minimum Scale [%]		97
Maximum Scale [%]		103
Shear Vertical Angle [°]		0
Shear Horizontal Angle [°]		0

No element selected.

Actual

Predicted	Good	Weak	Bad
Good	16	0	0
Weak	0	0	0
Bad	0	0	1

Accuracy	100%
All	17
Labeled 'Good'	16
Labeled 'Bad'	1

Path D:\Depot\AdaptiveVision_DLao\Felts\DA1_Model

Service is online.

Filtering the images in the training set.

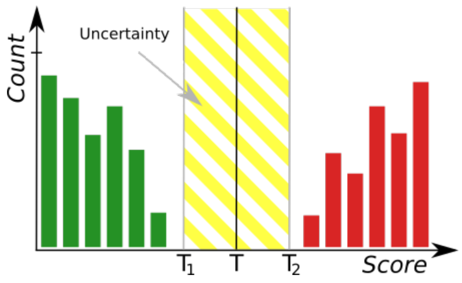
Interactive histogram tool

DetectAnomalies filters measure deviation of samples from the normal image appearances learned during the training phase. If the deviation exceeds a given threshold, the image is marked as anomalous. The suggested threshold is automatically calculated after the training phase but it can also be adjusted by the user in the Deep Learning Editor using an interactive histogram tool described below.

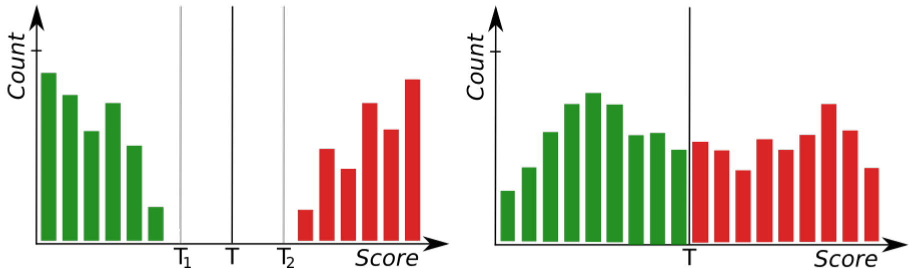
After the training phase, scores are calculated for every training sample and are presented in the form of a histogram; good samples are marked with green, bad samples with red bars. In the perfect case, the scores for good samples should be all lower than for bad samples and the threshold should be automatically calculated to give the optimal accuracy of the model. However, the groups may sometimes overlap because of:

1. incorrectly labeled samples,
2. bad Feature Size,
3. ambiguous definition of the expected defects,
4. high variability of the samples appearance or environmental conditions.

In order to achieve more robust threshold, it is recommended to perform training with a large number of samples from both groups. If the number of samples is limited, our software makes it possible to manually set the uncertainty area with additional thresholds (the information about the confidence of the model can be then obtained from the hidden outIsConfident filter output).



The histogram tool where green bars represent correct samples and red bars represent anomalous samples. T marks the main threshold and T1, T2 define the area of uncertainty.

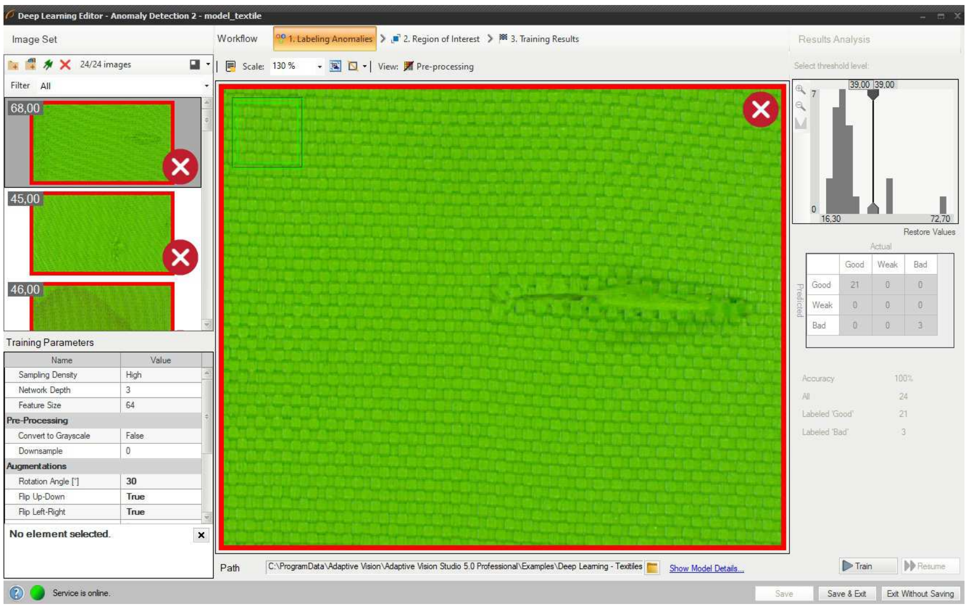


Left: a histogram presenting well-separated groups indicating a good accuracy of the model. Right: a poor accuracy of the model.

Detecting anomalies 2 (classificational approach)

DL_DetectAnomalies2 is another filter which can be used for detecting anomalies. It is designed to solve the same problem, but in a different way. Instead of using image reconstruction techniques Anomaly Detection 2 performs one-class classification of each part of the input image.

As both tools are very similar the steps of creating a model are the same. There are only a few differences between those filters in the model parameters section. In case of DL_DetectAnomalies2 the user does not need to change iteration count and network type. Instead, it is possible to set the Sampling density that defines the step of analysis using the inspection window. The higher Sampling density, the more precise heatmaps, but longer training and inference times.



Name	Value
Sampling Density	High
Network Depth	3
Feature Size	64
Pre-Processing	
Convert to Grayscale	False
Downsample	0
Augmentations	
Rotation Angle [°]	**30**
Flip Up-Down	**True**
Flip Left-Right	**True**

No element selected.

Path: C:\ProgramData\Adaptive Vision\Adaptive Vision Studio 5.0 Professional\Examples\Deep Learning - Textiles

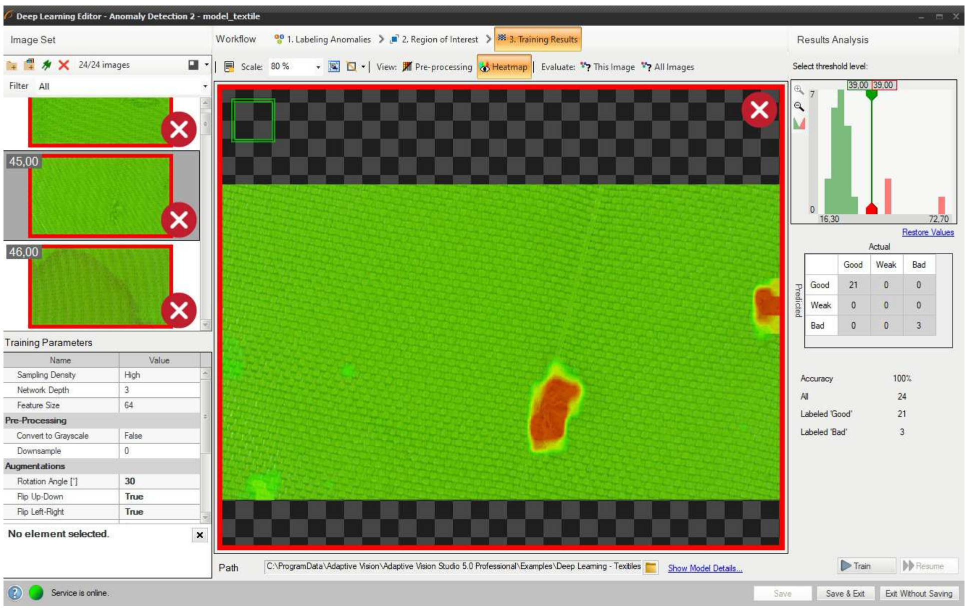
Actual / Predicted

	Good	Weak	Bad
Good	21	0	0
Weak	0	0	0
Bad	0	0	3

Accuracy	100%
All	24
Labeled 'Good'	21
Labeled 'Bad'	3

Results are marked as rectangles of either of two colors: green (classified as good) or red (classified as bad).

The resulting heatmap is usually not as accurate spatially as in case of using reconstructive anomaly detection, but the score accuracy and group separation on the histogram may be much better.



Training Parameters

Name	Value
Sampling Density	High
Network Depth	3
Feature Size	64
Pre-Processing	
Convert to Grayscale	False
Downsample	0
Augmentations	
Rotation Angle [°]	**30**
Flip Up-Down	**True**
Flip Left-Right	**True**

No element selected.

Actual

Predicted	Good	Weak	Bad
Good	21	0	0
Weak	0	0	0
Bad	0	0	3

Accuracy	100%
All	24
Labeled 'Good'	21
Labeled 'Bad'	3

Heatmap indicates the most possible locations of defects.

Detecting features (segmentation)

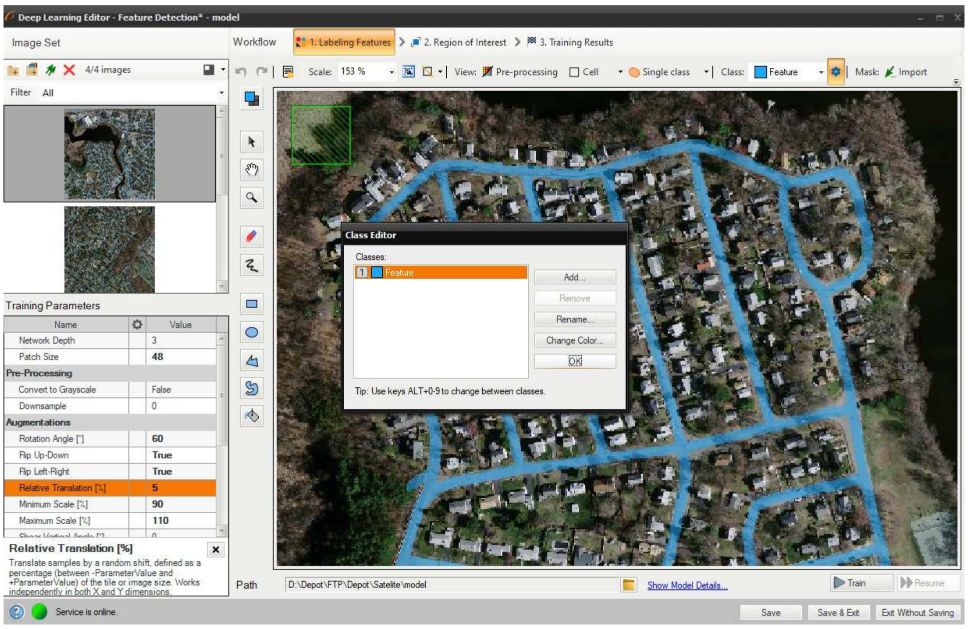
In this tool the user has to define each feature class and then mark features on each image in the training set. This technique is used to find object defects like scratches or color changes, and for detecting image parts trained on a selected pattern.

1. Defining feature classes (Marking class)

First, the user has to define classes of defects. Generally, they should be features which user would like to detect on images. Multiple different classes can be defined but it is not recommended to use more than a few.

Class editor is available under sprocket wheel icon in the top bar.

To manage classes, **Add**, **Remove** or **Rename** buttons can be used. To customize appearance, color of each class can be changed using **Change Color** button.



Training Parameters

Name	Value
Network Depth	3
Patch Size	**48**
Pre-Processing	
Convert to Grayscale	False
Downsample	0
Augmentations	
Rotation Angle [°]	**60**
Flip Up-Down	**True**
Flip Left-Right	**True**
Relative Translation [%]	**5**
Minimum Scale [%]	**90**
Maximum Scale [%]	**110**
Shear Vertical Angle [°]	0

Relative Translation [%]
Translate samples by a random shift, defined as a percentage (between -ParameterValue and +ParameterValue) of the tile or image size. Works independently in both X and Y dimensions.

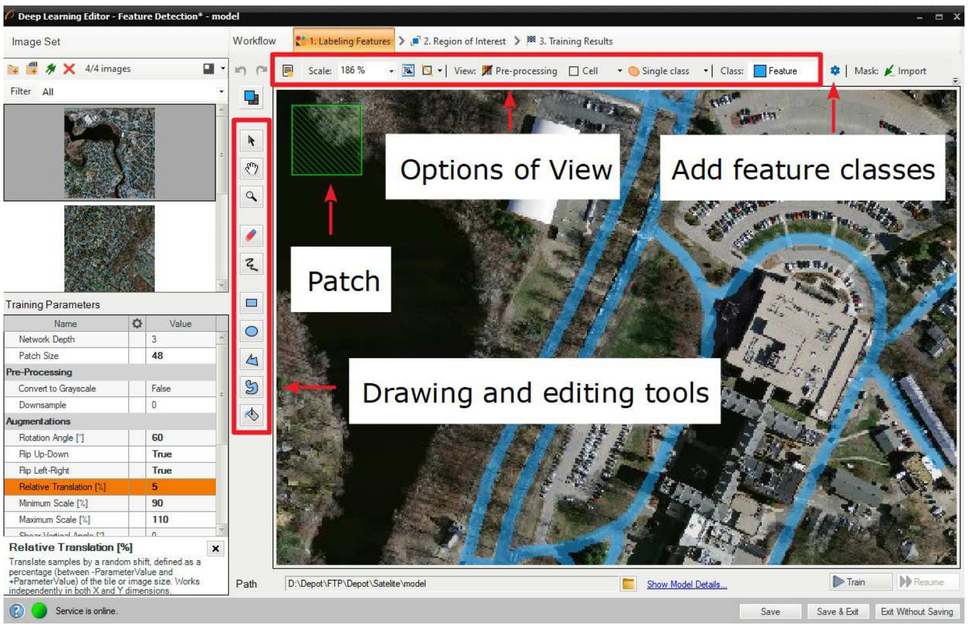
In this tool it is possible to define more classes of defects.

The current class for editing is displayed on the left, the user can select different class after click.

Use drawing tool to mark features on the input images. Tools such as **Brush** or **Rectangle** can be used for selecting features.

In addition, class masks can be imported from external files. There are buttons for **Import** and **Export** created classes so that the user can create an image of a mask automatically prior to a Deep Learning model.

The image mask should have the same size as the selected image in the input set. When importing an image mask, all non-black pixels will be included in the current mask.



Name	Value
Network Depth	3
Patch Size	48
Pre-Processing	
Convert to Grayscale	False
Downsample	0
Augmentations	
Rotation Angle [°]	60
Flip Up-Down	True
Flip Left-Right	True
Relative Translation [%]	5
Minimum Scale [%]	90
Maximum Scale [%]	110

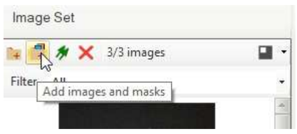
Relative Translation [%]
Translate samples by a random shift, defined as a percentage (between -ParameterValue and +ParameterValue) of the tile or image size. Works independently in both X and Y dimensions.

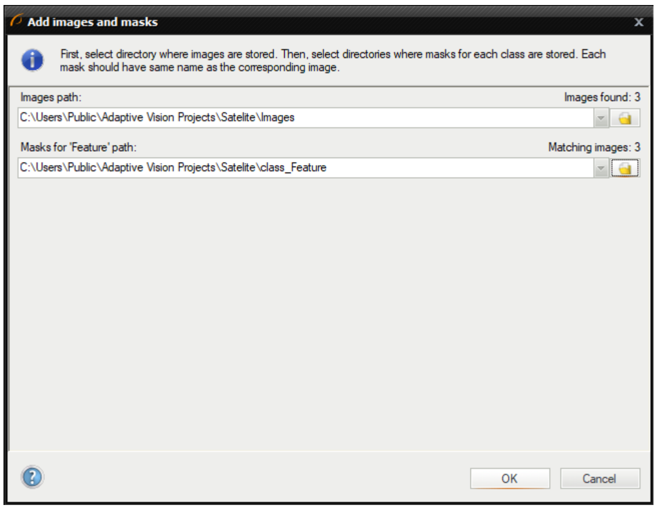
Path: D:\Depot\FTP\Depot\Satelite\model

Service is online.

The most important features of the tool.

The user can also load multiple images and masks at the same time, using **Add images and masks** button.





First, select directory where images are stored. Then, select directories where masks for each class are stored. Each mask should have same name as the corresponding image.

Images path: C:\Users\Public\Adaptive Vision Projects\Satelite\Images — Images found: 3

Masks for 'Feature' path: C:\Users\Public\Adaptive Vision Projects\Satelite\class_Feature — Matching images: 3

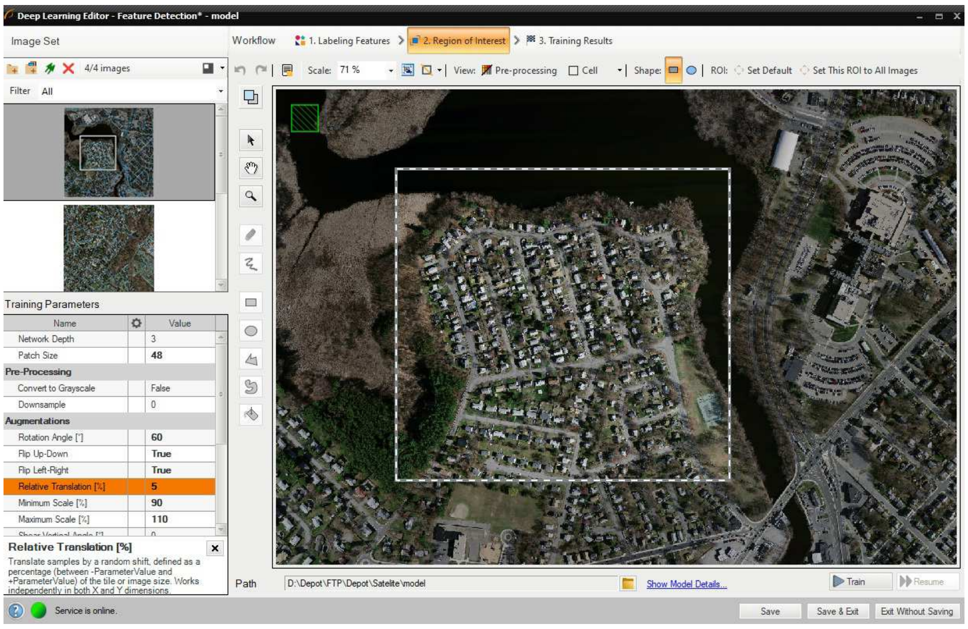
Selecting path to images and masks.

Directory containing input images should be selected first. Then, directories for each feature class can be selected below. Images and masks are matched automatically using their file names. For example, let us assume that "images" directory contains images 001.png, 002.png, 003.png; "mask_class1" directory contains 001.png, 002.png, 003.png; and "mask_class2" directory contains 001.png, 002.png, 003.png. Then "images\001.png" image will be loaded together with "mask_class1\001.png" and "mask_class2\001.png" masks.

2. Reducing region of interest

The user can reduce the input image size to speed up the training process. In many cases, number of features on an image is very large and most of them are the same. In such case the region of interest also can be reduced.

On the top bar there are tools for applying the current ROI to all images as well as for resetting the ROI.



Training Parameters

Name	Value
Network Depth	3
Patch Size	48
Pre-Processing	
Convert to Grayscale	False
Downsample	0
Augmentations	
Rotation Angle [°]	60
Flip Up-Down	True
Flip Left-Right	True
Relative Translation [%]	5
Minimum Scale [%]	90
Maximum Scale [%]	110
Shear Vertical Angle [°]	0

Relative Translation [%]
Translate samples by a random shift, defined as a percentage (between -ParameterValue and +ParameterValue) of the tile or image size. Works independently in both X and Y dimensions.

Path D:\Depot\FTP\Depot\Satelite\model

Service is online.

Setting ROI.

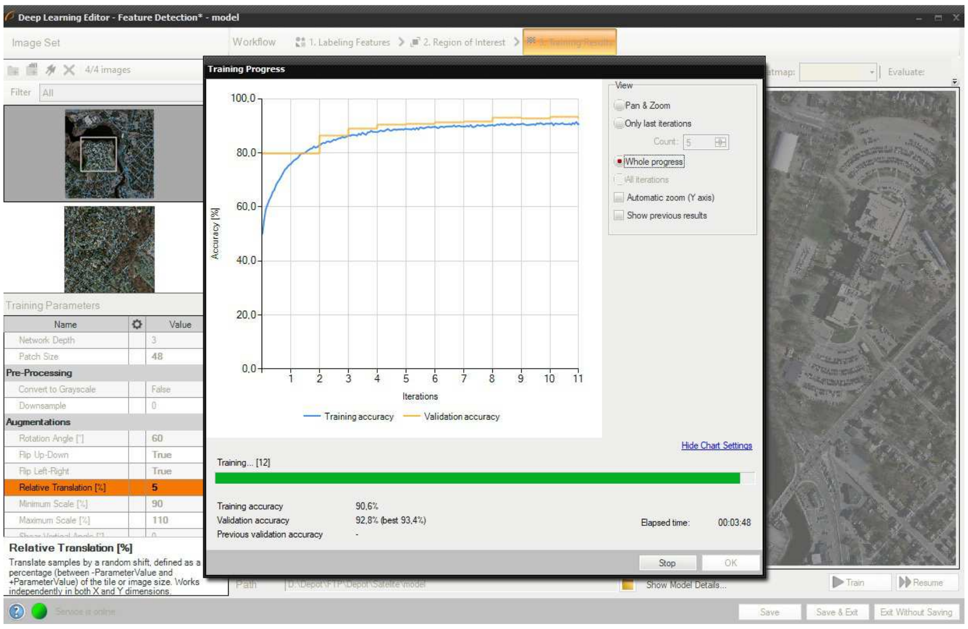
3. Setting training parameters

- **Network depth** – chooses one of several predefined network architectures varying in their complexity. For bigger and more complex image patterns a higher depth might be necessary.
- **Patch size** – the size of an image part that will be analyzed with one pass through the neural network. It should be significantly bigger than any feature of interest, but not too big – as the bigger the patch size, the more difficult and time consuming is the training process.
- **Stopping conditions** – define when the training process should stop.

For more details please refer to Deep Learning – Setting parameters and Deep Learning – Augmentation.

4. Model training

The chart contains two series: training and validation score. Higher score value leads to better results.



Training... [12]

Training accuracy	90,6%
Validation accuracy	92,8% (best 93,4%)
Previous validation accuracy	-

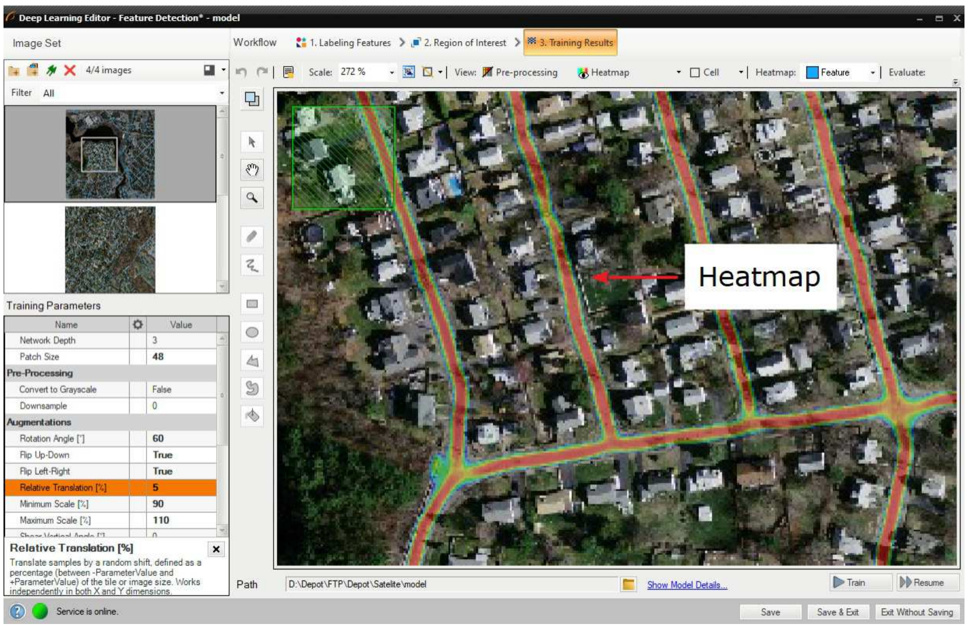
Elapsed time: 00:03:48

In this case training process consists of computing training and validation accuracies.

5. Result analysis

Image scores (heatmaps) are presented in blue-yellow-red colors palette after using the model to evaluation of image. The color represents the probability of the element belonging to the currently selected feature class.

Evaluate: This Image and **Evaluate: All Images** buttons can be used to classify images. It can be useful after adding new images to the data set or after changing the area of interest.



Training Parameters

Name	Value
Network Depth	3
Patch Size	48
Pre-Processing	
Convert to Grayscale	False
Downsample	0
Augmentations	
Rotation Angle [°]	60
Flip Up-Down	True
Flip Left-Right	True
Relative Translation [%]	5
Minimum Scale [%]	90
Maximum Scale [%]	110

Relative Translation [%]

Translate samples by a random shift, defined as a percentage (between -ParameterValue and +ParameterValue) of the tile or image size. Works independently in both X and Y dimensions.

Path D:\Depot\FTP\Depot\Satelite\model Show Model Details...

Service is online.

Image after classification.

In the top left corner of the editor, a green rectangle visualizes the selected patch size.

Classifying objects

In this too, the user only has to label images with respect to a desired number of classes. Theoretically, the number of classes that a user can create is infinite, but please note that you are limited by the amount of data your GPU can process. Labeled images will allow to train model and determine features which will be used to evaluate new samples and assign them to proper classes.

1. Editing number of classes

By default two classes are defined. If the problem is more complex than that, the user can edit classes and define more if needed. Once the user is ready with definition of classes, images can be labeled.

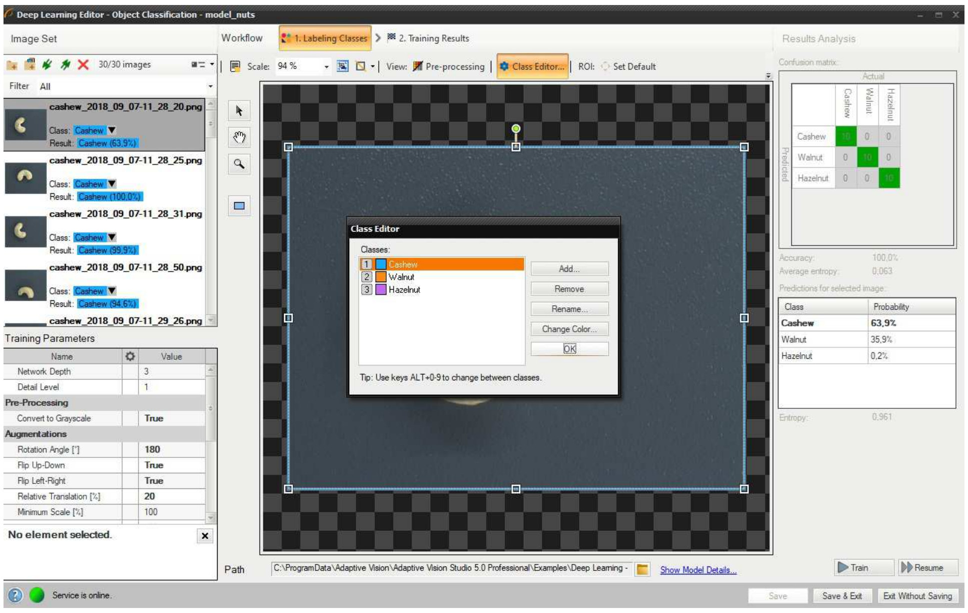


Image	Class	Result
cashew_2018_09_07-11_28_20.png	Cashew	Cashew (63,9%)
cashew_2018_09_07-11_28_25.png	Cashew	Cashew (100,0%)
cashew_2018_09_07-11_28_31.png	Cashew	Cashew (99,9%)
cashew_2018_09_07-11_28_50.png	Cashew	Cashew (94,6%)
cashew_2018_09_07-11_29_26.png		

Training Parameters

Name	Value
Network Depth	3
Detail Level	1
Pre-Processing	
Convert to Grayscale	True
Augmentations	
Rotation Angle [°]	180
Flip Up-Down	True
Flip Left-Right	True
Relative Translation [%]	20
Minimum Scale [%]	100

No element selected.

Class Editor

Classes:
1 Cashew
2 Walnut
3 Hazelnut

Tip: Use keys ALT+0-9 to change between classes.

Confusion matrix:

Predicted \ Actual	Cashew	Walnut	Hazelnut
Cashew	10	0	0
Walnut	0	10	0
Hazelnut	0	0	10

Accuracy: 100,0%
Average entropy: 0,063

Predictions for selected image:

Class	Probability
Cashew	**63,9%**
Walnut	35,9%
Hazelnut	0,2%

Entropy: 0,961

Path C:\ProgramData\Adaptive Vision\Adaptive Vision Studio 5.0 Professional\Examples\Deep Learning - Show Model Details...

Service is online.

Using Class Editor.

2. Labeling samples

Labeling of samples is possible after adding training images. Each image has a corresponding drop-down list which allows for assigning a specific class. It is possible to assign a single class to multiple images by selecting desired images in Deep Learning Editor.

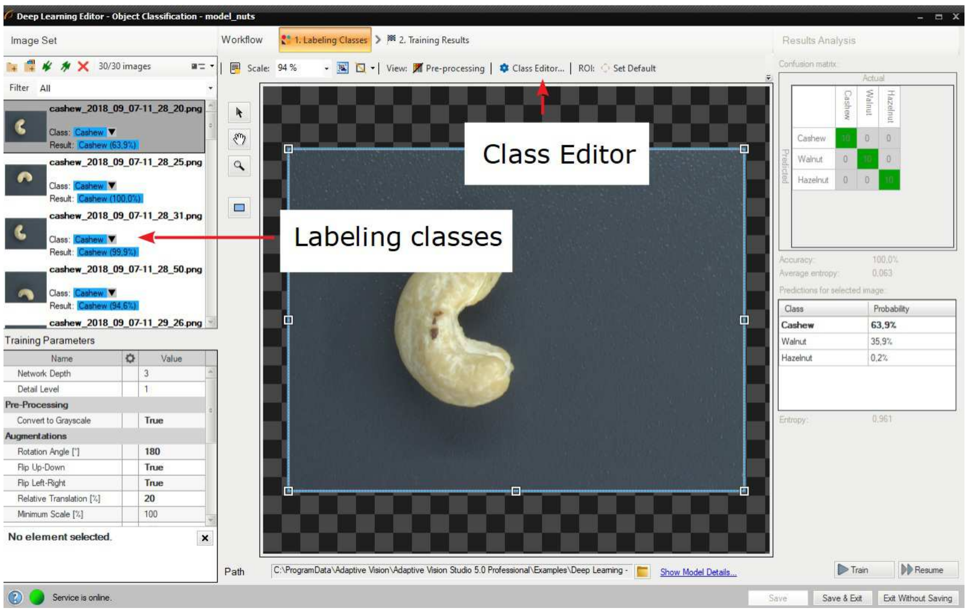

Deep Learning Editor - Object Classification - model_nuts

Image Set | 30/30 images | Filter: All

cashew_2018_09_07-11_28_20.png	Class: Cashew; Result: Cashew (63,9%)
cashew_2018_09_07-11_28_25.png	Class: Cashew; Result: Cashew (100,0%)
cashew_2018_09_07-11_28_31.png	Class: Cashew; Result: Cashew (99,9%)
cashew_2018_09_07-11_28_50.png	Class: Cashew; Result: Cashew (94,6%)
cashew_2018_09_07-11_29_26.png	

Training Parameters

Name	Value
Network Depth	3
Detail Level	1
Pre-Processing	
Convert to Grayscale	True
Augmentations	
Rotation Angle [°]	180
Flip Up-Down	True
Flip Left-Right	True
Relative Translation [%]	20
Minimum Scale [%]	100

No element selected.

Workflow: 1. Labeling Classes > 2. Training Results | Scale: 94 % | View: Pre-processing | Class Editor... | ROI: Set Default

Path: C:\ProgramData\Adaptive Vision\Adaptive Vision Studio 5.0 Professional\Examples\Deep Learning - | Show Model Details...

Results Analysis — Confusion matrix:

Predicted \ Actual	Cashew	Walnut	Hazelnut
Cashew	10	0	0
Walnut	0	10	0
Hazelnut	0	0	10

Accuracy: 100,0%
Average entropy: 0,063

Predictions for selected image:

Class	Probability
Cashew	**63,9%**
Walnut	35,9%
Hazelnut	0,2%

Entropy: 0,961

Labeling images with classes.

3. Reducing region of interest

Reduce the region of interest to focus only on the important part of the image. Reducing region of interest will speed up both training and classification. By default the region of interest contains the whole image.

To get the best classification results, use the same region of interest for training and classification.

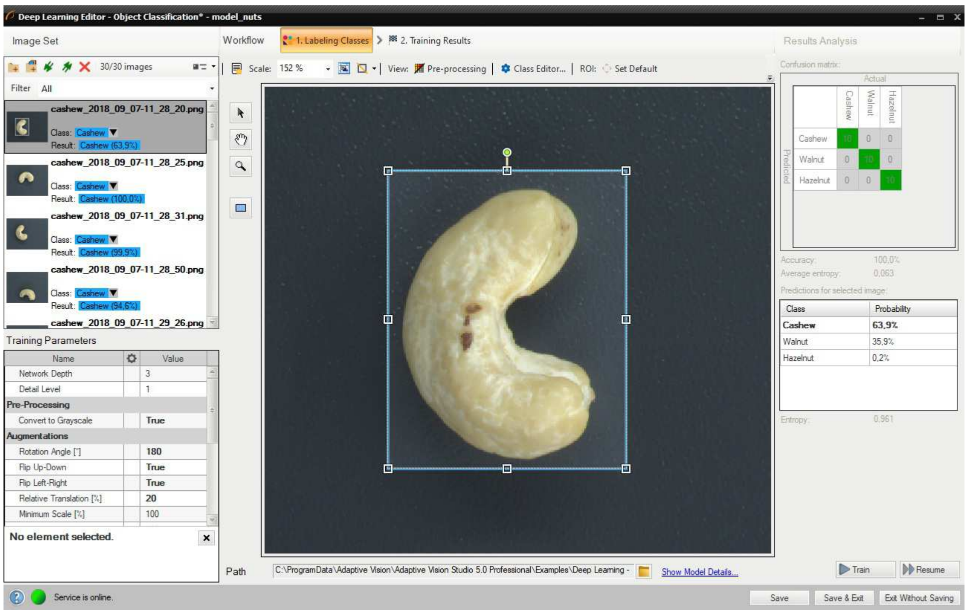

Deep Learning Editor - Object Classification* - model_nuts

Image Set | 30/30 images | Filter: All

cashew_2018_09_07-11_28_20.png	Class: Cashew; Result: Cashew (63,9%)
cashew_2018_09_07-11_28_25.png	Class: Cashew; Result: Cashew (100,0%)
cashew_2018_09_07-11_28_31.png	Class: Cashew; Result: Cashew (99,9%)
cashew_2018_09_07-11_28_50.png	Class: Cashew; Result: Cashew (94,6%)
cashew_2018_09_07-11_29_26.png	

Training Parameters

Name	Value
Network Depth	3
Detail Level	1
Pre-Processing	
Convert to Grayscale	True
Augmentations	
Rotation Angle [°]	180
Flip Up-Down	True
Flip Left-Right	True
Relative Translation [%]	20
Minimum Scale [%]	100

No element selected.

Workflow: 1. Labeling Classes > 2. Training Results | Scale: 152 % | View: Pre-processing | Class Editor... | ROI: Set Default

Path: C:\ProgramData\Adaptive Vision\Adaptive Vision Studio 5.0 Professional\Examples\Deep Learning - | Show Model Details...

Results Analysis — Confusion matrix:

Predicted \ Actual	Cashew	Walnut	Hazelnut
Cashew	10	0	0
Walnut	0	10	0
Hazelnut	0	0	10

Accuracy: 100,0%
Average entropy: 0,063

Predictions for selected image:

Class	Probability
Cashew	**63,9%**
Walnut	35,9%
Hazelnut	0,2%

Entropy: 0,961

Changed region of interest.

4. Setting training parameters

- **Network depth** – predefined network architecture parameter. For more complex problems higher depth might be necessary.
- **Detail level** – level of detail needed for a particular classification task. For majority of cases the default value of 1 is appropriate, but if images of different classes are distinguishable only by small features, increasing value of this parameter may improve classification results.
- **Stopping conditions** – define when the training process should stop.

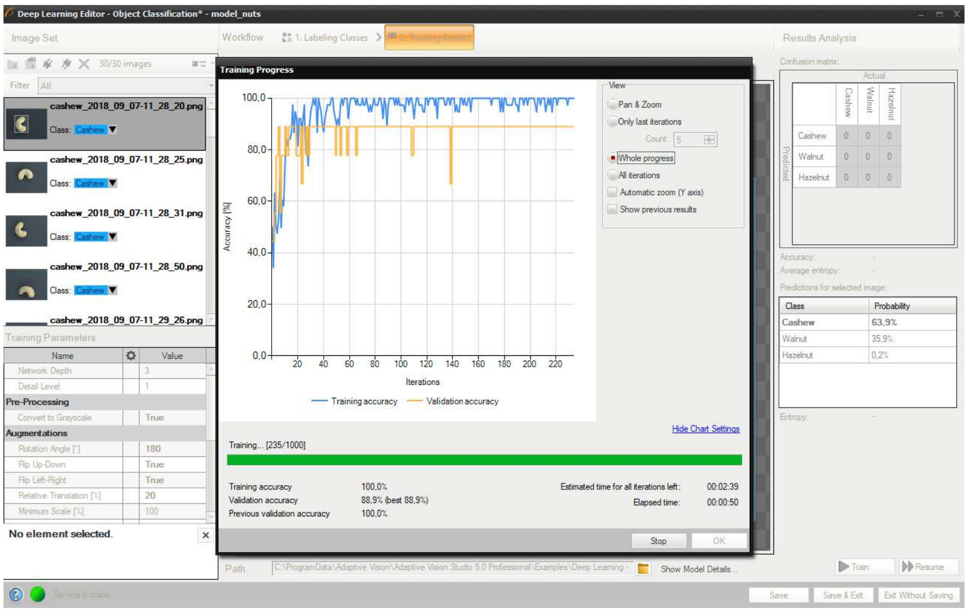
For more details please refer to Deep Learning – Setting parameters and Deep Learning – Augmentation.

5. Performing training

During training two series are visible: training accuracy and validation accuracy. Both charts should have a similar pattern.

More detailed information is displayed below the chart:

- current training statistics (training and validation accuracy),
- number of processed samples (depends on the number of images),
- elapsed time.


Deep Learning Editor - Object Classification* - model_nuts

Training Progress

Training... [235/1000]

Training accuracy	100,0%	Estimated time for all iterations left:	00:02:39
Validation accuracy	88,9% (best 88,9%)	Elapsed time:	00:00:50
Previous validation accuracy	100,0%		

Training Parameters

Name	Value
Network Depth	3
Detail Level	1
Pre-Processing	
Convert to Grayscale	True
Augmentations	
Rotation Angle [°]	180
Flip Up-Down	True
Flip Left-Right	True
Relative Translation [%]	20
Minimum Scale [%]	100

Path C:\ProgramData\Adaptive Vision\Adaptive Vision Studio 5.0 Professional\Examples\Deep Learning -

Training object classification model.

Training process can take a couple of minutes or even longer. It can be manually finished if needed. The final result of one training is one of the partial models that achieved the highest validation accuracy (not necessarily the last one). Consecutive training attempts will prompt the user whether to save a new model or keep the old one.

6. Analyzing results

The window shows a confusion matrix which indicates how well the training samples have been classified.

The image view contains a heatmap which indicates which part of the image contributed the most to the classification result.

Evaluate: This Image and **Evaluate: All Images** buttons can be used to classify training images. It can be useful after adding new images to the data set or after changing the area of interest.

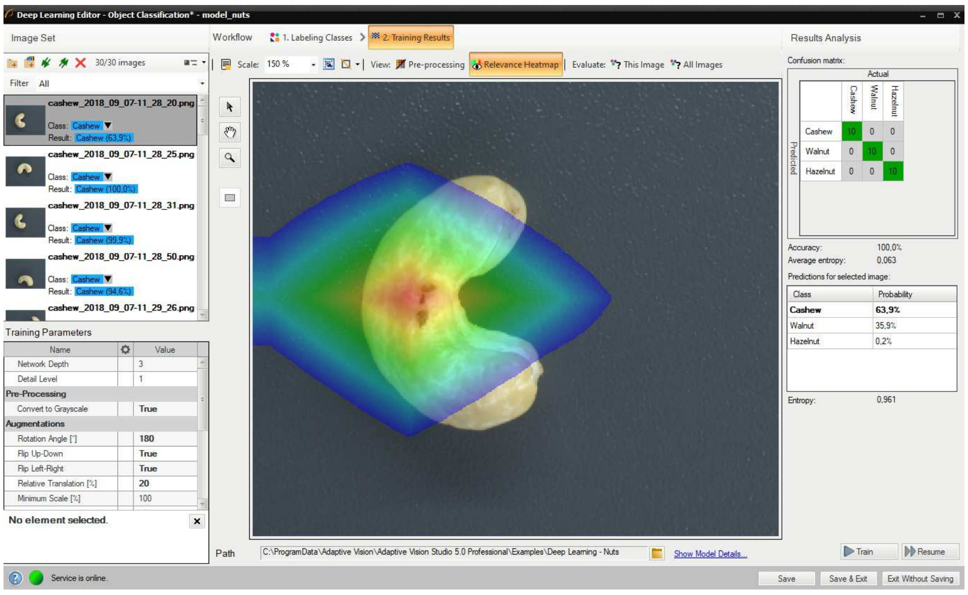

Deep Learning Editor - Object Classification* - model_nuts

Image	Class	Result
cashew_2018_09_07-11_28_20.png	Cashew	Cashew (63,9%)
cashew_2018_09_07-11_28_25.png	Cashew	Cashew (100,0%)
cashew_2018_09_07-11_28_31.png	Cashew	Cashew (99,9%)
cashew_2018_09_07-11_28_50.png	Cashew	Cashew (94,6%)

Confusion matrix:

Predicted \ Actual	Cashew	Walnut	Hazelnut
Cashew	10	0	0
Walnut	0	10	0
Hazelnut	0	0	10

Accuracy: 100,0%
Average entropy: 0,063

Predictions for selected image:

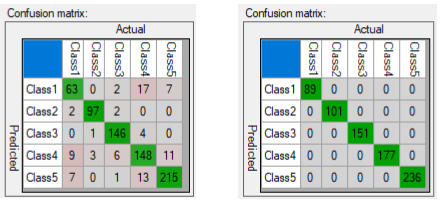
Class	Probability
Cashew	**63,9%**
Walnut	35,9%
Hazelnut	0,2%

Entropy: 0,961

Path C:\ProgramData\Adaptive Vision\Adaptive Vision Studio 5.0 Professional\Examples\Deep Learning - Nuts

Confusion matrix and class assignment after the training.

Sometimes it is hard to guess the right parameters in the first attempt. The picture below shows confusion matrix that indicates inaccurate classification during the training (left).


Confusion matrix:

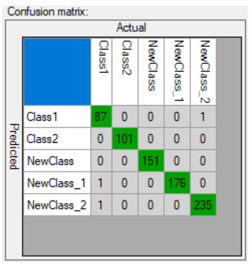
Predicted \ Actual	Class1	Class2	Class3	Class4	Class5
Class1	63	0	2	17	7
Class2	2	97	2	0	0
Class3	0	1	146	4	0
Class4	9	3	6	148	11
Class5	7	0	1	13	215

Confusion matrix:

Predicted \ Actual	Class1	Class2	Class3	Class4	Class5
Class1	89	0	0	0	0
Class2	0	101	0	0	0
Class3	0	0	151	0	0
Class4	0	0	0	177	0
Class5	0	0	0	0	236

Confusion matrices for model that needs more training (left) and for model well trained (right).

It is possible that confusion matrix indicates that trained model is not 100% accurate with respect to training samples (numbers assigned exclusively on main diagonal represent 100% accuracy). User needs to properly analyze this data, and use to his advantage.


Confusion matrix:

Predicted \ Actual	Class1	Class2	NewClass	NewClass_1	NewClass_2
Class1	87	0	0	0	1
Class2	0	101	0	0	0
NewClass	0	0	151	0	0
NewClass_1	1	0	0	176	0
NewClass_2	1	0	0	0	235

Confusion matrix indicating good generalization.

Too many erroneous classifications indicate poor training. Few of them may indicate that model is properly focused on generalization rather than exact matching to training samples (possible overfitting). Good generalization can be achieved if images used for training are varied (even among single class). If provided data is not varied within classes (user expects exact matching), and still some images are classified outside main diagonal after the training, user can:

- increase the network depth,
- prolong training by increasing number of iterations,
- increase amount of data used for training,
- use augmentation,
- increase the detail level parameter.

Segmenting instances

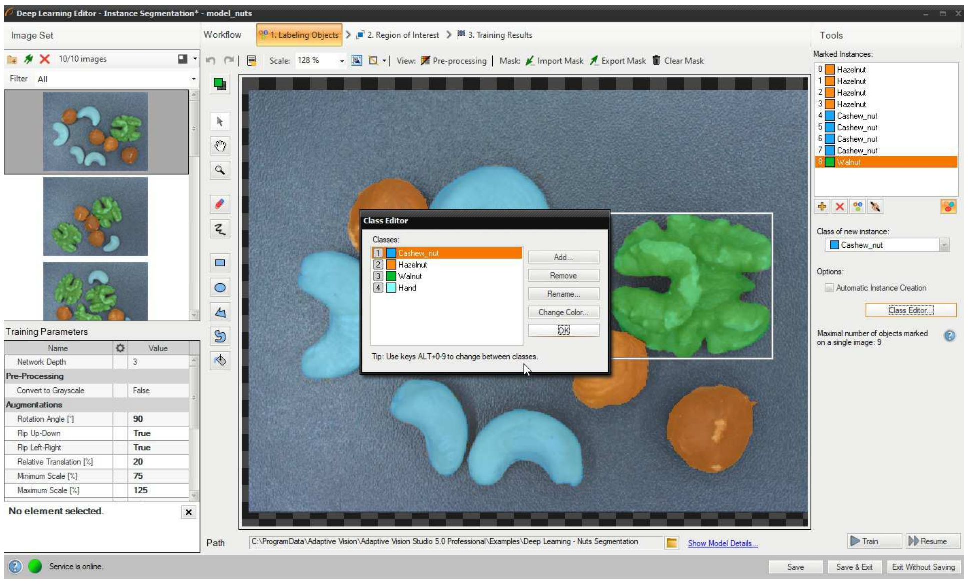
In this tool a user needs to draw regions (masks) corresponding to the objects in the scene and specify their classes. These images and masks are used to train a model which then in turn is used to locate, segment and classify objects in the input images.

1. Defining object classes

First, a user needs to define classes of objects that the model will be trained on and that later it will be used to detect. Instance segmentation model can deal with single as well as multiple classes of objects.

Class editor is available under the Class Editor button.

To manage classes, Add, Remove or Rename buttons can be used. To customize appearance, color of each class can be changed using the Change Color button.



Training Parameters

Name	Value
Network Depth	3
Pre-Processing	
Convert to Grayscale	False
Augmentations	
Rotation Angle [°]	**90**
Flip Up-Down	**True**
Flip Left-Right	**True**
Relative Translation [%]	**20**
Minimum Scale [%]	**75**
Maximum Scale [%]	**125**

No element selected.

Class Editor

Classes:

1 🟦 Cashew_nut
2 🟧 Hazelnut
3 🟩 Walnut
4 🟦 Hand

Tip: Use keys ALT+0-9 to change between classes.

Marked Instances:

0 🟧 Hazelnut
1 🟧 Hazelnut
2 🟧 Hazelnut
3 🟧 Hazelnut
4 🟦 Cashew_nut
5 🟦 Cashew_nut
6 🟦 Cashew_nut
7 🟦 Cashew_nut
8 🟩 Walnut

Class of new instance: Cashew_nut

Options:

Automatic Instance Creation

Maximal number of objects marked on a single image: 9

Path C:\ProgramData\Adaptive Vision\Adaptive Vision Studio 5.0 Professional\Examples\Deep Learning - Nuts Segmentation

Service is online.

Using Class Editor.

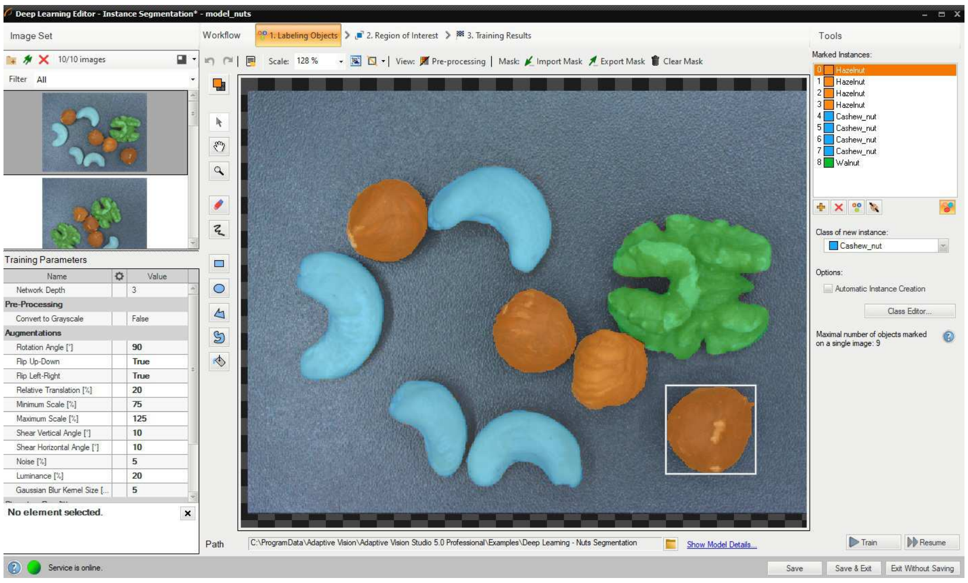
2. Labeling objects

After adding training images and defining classes a user needs to draw regions (masks) to mark objects in images.

To mark an object the user needs to select a proper class in the Current Class drop-down menu and click the Add Instance button (green plus). Alternatively, for convenience of labeling, it is possible to apply **Automatic Instance Creation** which allows a user to draw quickly masks on multiple objects in the image without having to add a new instance every time.

Use drawing tool to mark objects on the input images. Multiple tools such as brush and shapes can be used to draw object masks. Masks are the same color as previously defined for the selected classes.

The Marked Instances list in the top left corner displays a list of defined objects for the current image. If an object does not have a corresponding mask created in the image, it is marked as "(empty)". When an object is selected, a bounding box is displayed around its mask in the drawing area. Selected object can be modified in terms of a class (Change Class button) as well as a mask (by simply drawing new parts or erasing existing ones). Remove Instance button (red cross) allows to completely remove a selected object.



10/10 images

Filter: All

Training Parameters

Name	Value
Network Depth	3
Pre-Processing	
Convert to Grayscale	False
Augmentations	
Rotation Angle [°]	90
Flip Up-Down	True
Flip Left-Right	True
Relative Translation [%]	20
Minimum Scale [%]	75
Maximum Scale [%]	125
Shear Vertical Angle [°]	10
Shear Horizontal Angle [°]	10
Noise [%]	5
Luminance [%]	20
Gaussian Blur Kernel Size [...	5

No element selected.

Scale: 128 %

Marked Instances:

0	🟧	Hazelnut
1	🟧	Hazelnut
2	🟧	Hazelnut
3	🟧	Hazelnut
4	🟦	Cashew_nut
5	🟦	Cashew_nut
6	🟦	Cashew_nut
7	🟦	Cashew_nut
8	🟩	Walnut

Class of new instance: Cashew_nut

Options:

☐ Automatic Instance Creation

Maximal number of objects marked on a single image: 9

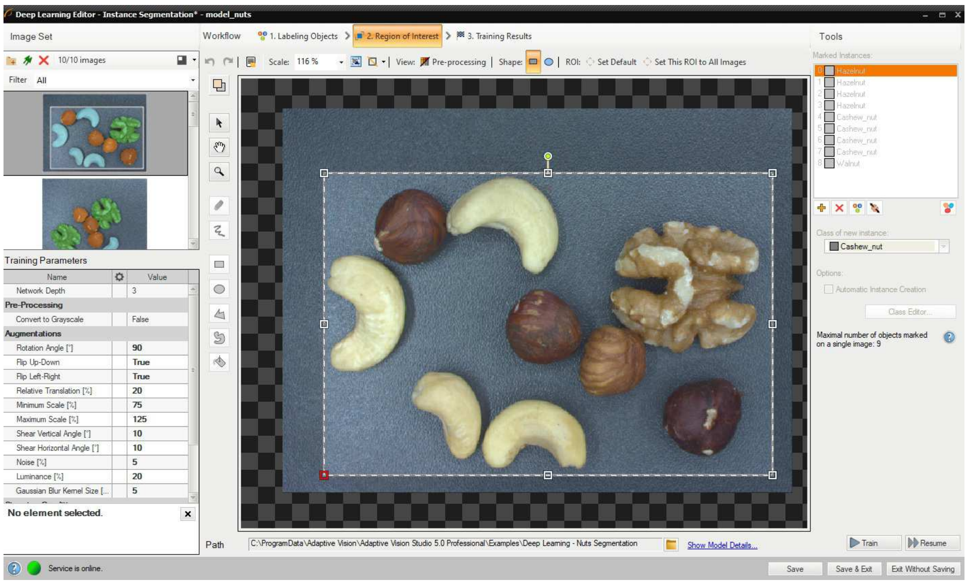
Path C:\ProgramData\Adaptive Vision\Adaptive Vision Studio 5.0 Professional\Examples\Deep Learning - Nuts Segmentation

Service is online.

Labeling objects.

3. Reducing region of interest

Reduce region of interest to focus only on the important part of the image. By default region of interest contains the whole image.



10/10 images

Filter: All

Training Parameters

Name	Value
Network Depth	3
Pre-Processing	
Convert to Grayscale	False
Augmentations	
Rotation Angle [°]	90
Flip Up-Down	True
Flip Left-Right	True
Relative Translation [%]	20
Minimum Scale [%]	75
Maximum Scale [%]	125
Shear Vertical Angle [°]	10
Shear Horizontal Angle [°]	10
Noise [%]	5
Luminance [%]	20
Gaussian Blur Kernel Size [...	5

No element selected.

Scale: 116 %

Marked Instances:

0	⬜	Hazelnut
1	⬜	Hazelnut
2	⬜	Hazelnut
3	⬜	Hazelnut
4	⬜	Cashew_nut
5	⬜	Cashew_nut
6	⬜	Cashew_nut
7	⬜	Cashew_nut
8	⬜	Walnut

Class of new instance: Cashew_nut

Options:

☐ Automatic Instance Creation

Maximal number of objects marked on a single image: 9

Path C:\ProgramData\Adaptive Vision\Adaptive Vision Studio 5.0 Professional\Examples\Deep Learning - Nuts Segmentation

Service is online.

Changing region of interest.

4. Setting training parameters

- **Network depth** – predefined network architecture parameter. For more complex problems higher depth might be necessary,
- **Stopping conditions** – define when the training process should stop.

For more details read Deep Learning – Setting parameters.

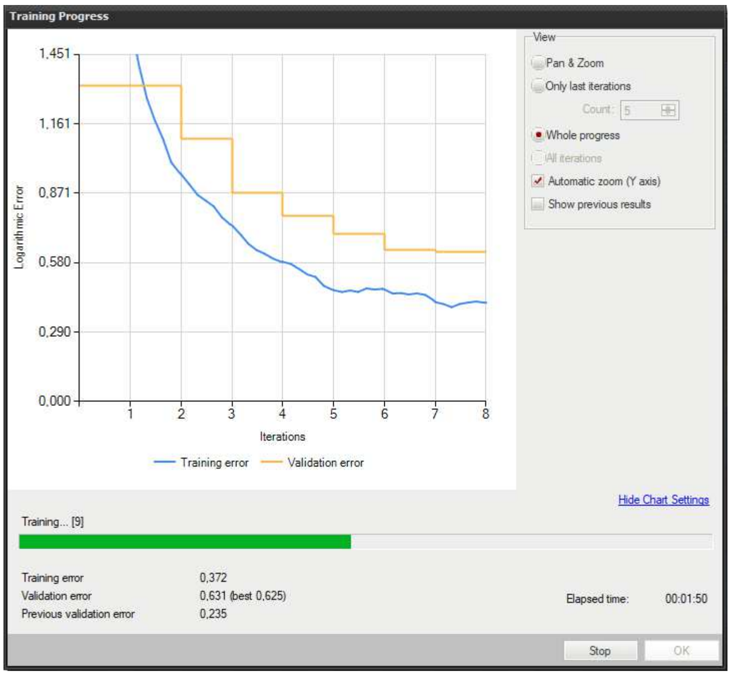
Details regarding augmentation parameters. Deep Learning – Augmentation

5. Performing training

During training two main series are visible: training error and validation error. Both charts should have a similar pattern. If a training was run before the third series with previous validation error is also displayed.

More detailed information is displayed below the chart:

- current iteration number,
- current training statistics (training and validation error),
- number of processed samples,
- elapsed time.



Training... [9]

Training error	0,372
Validation error	0,631 (best 0,625)
Previous validation error	0,235
Elapsed time:	00:01:50

Training instance segmentation model.

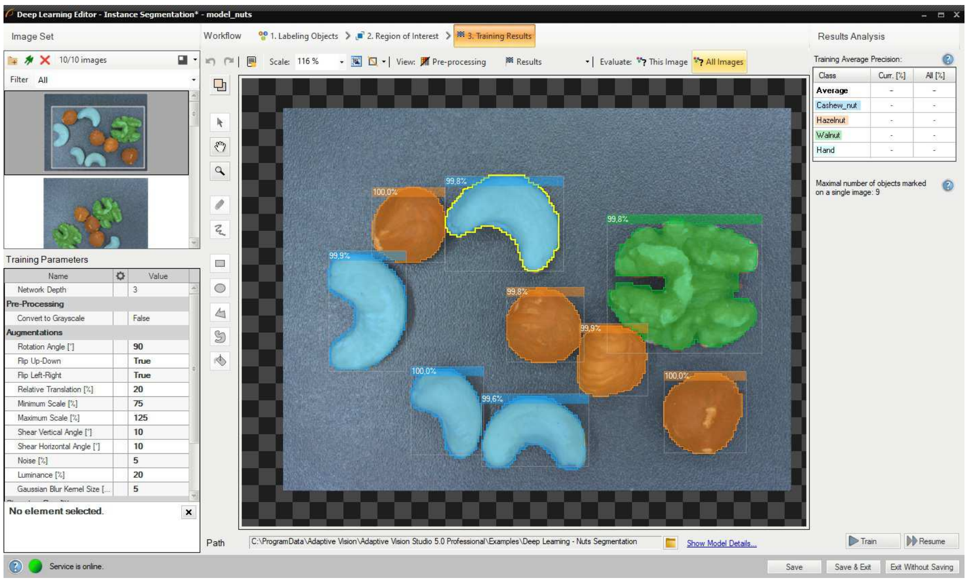
Training may be a long process. During this time, training can be stopped. If no model is present (first training attempt) model with best validation accuracy will be saved. Consecutive training attempts will prompt a user whether to replace the old model.

6. Analyzing results

The window shows the results of instance segmentation. Detected objects are displayed on top of the images. Each detection consists of following data:

- class (identified by a color),
- bounding box,
- model-generated instance mask,
- confidence score.

Evaluate: This Image and **Evaluate: All Images** buttons can be used to perform instance segmentation on the provided images. It can be useful after adding new images to the data set or after changing the area of interest.



Training Parameters

Name	Value
Network Depth	3
Pre-Processing	
Convert to Grayscale	False
Augmentations	
Rotation Angle [°]	90
Flip Up-Down	True
Flip Left-Right	True
Relative Translation [%]	20
Minimum Scale [%]	75
Maximum Scale [%]	125
Shear Vertical Angle [°]	10
Shear Horizontal Angle [°]	10
Noise [%]	5
Luminance [%]	20
Gaussian Blur Kernel Size [...	5

No element selected.

Training Average Precision:

Class	Curr. [%]	All [%]
Average	-	-
Cashew_nut	-	-
Hazelnut	-	-
Walnut	-	-
Hand	-	-

Maximal number of objects marked on a single image: 9

Path C:\ProgramData\Adaptive Vision\Adaptive Vision Studio 5.0 Professional\Examples\Deep Learning - Nuts Segmentation

Show Model Details...

Instance segmentation results visualized after the training.

Instance segmentation is a complex task therefore it is highly recommended to use data augmentations to improve network's ability to generalize learned information. If results are still not satisfactory the following standard methods can be used to improve model performance:

- providing more training data,
- increasing number of training iterations,
- increasing the network depth.

Locating points

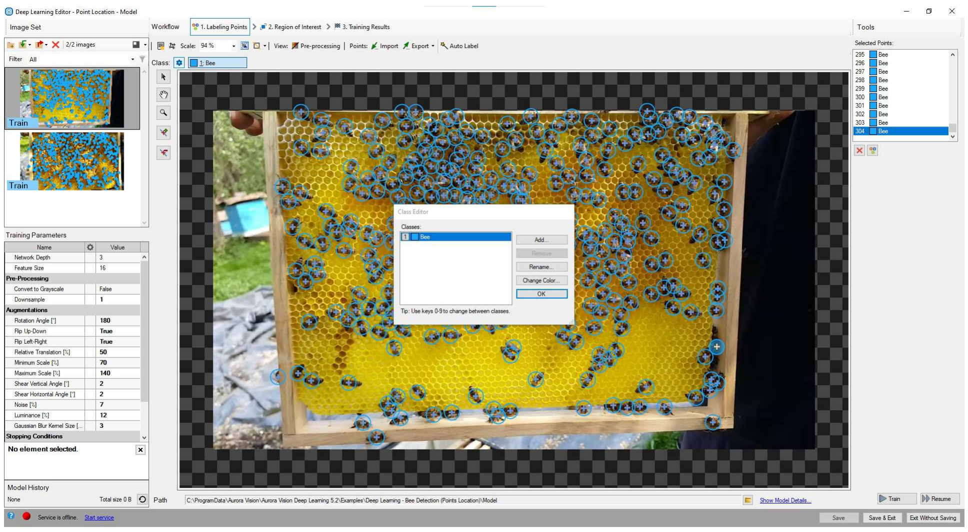
In this tool the user defines classes and marks key points in the image. This data is used to train a model which then is used to locate and classify key points in images.

1. Defining classes

First, a user needs to define classes of key points that the model will be trained on and later used to detect. Point location model can deal with single as well as multiple classes of key points.

Class editor is available under the Class Editor button.

To manage classes, Add, Remove or Rename buttons can be used. Color of each class can be changed using Change Color button.



Training Parameters

Name	Value
Network Depth	3
Feature Size	16
Pre-Processing	
Convert to Grayscale	False
Downsample	1
Augmentations	
Rotation Angle [°]	180
Flip Up-Down	True
Flip Left-Right	True
Relative Translation [%]	50
Minimum Scale [%]	70
Maximum Scale [%]	140
Shear Vertical Angle [°]	2
Shear Horizontal Angle [°]	2
Noise [%]	7
Luminance [%]	12
Gaussian Blur Kernel Size [...	3
Stopping Conditions	

No element selected.

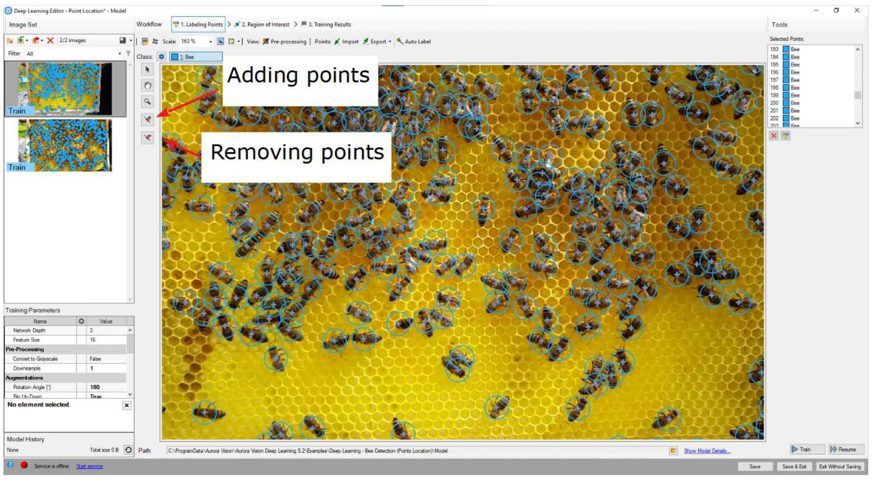
Using Class Editor.

2. Marking key points

After adding training images and defining classes a user needs to mark points in images.

To mark an object a user needs to select a proper class in the Current Class drop-down menu and click the Add Point button. Points have the same color as previously defined for the selected class.

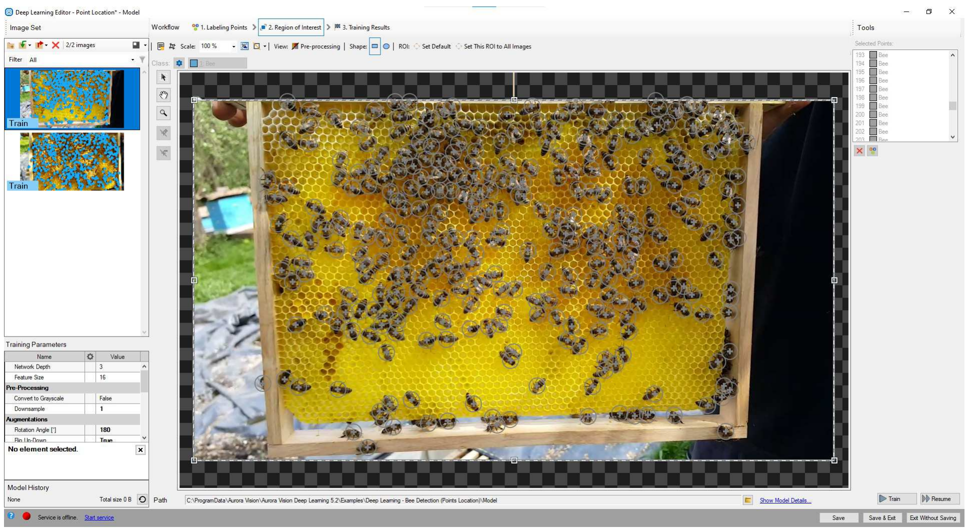
The Selected Points list in the top left corner displays a list of defined points for the current image. A point can be selected either from the list of directly on the image area. A selected point can be moved, removed (Remove Point button) or has its class changed (Change Class button).



Marking points.

3. Reducing region of interest

Reduce region of interest to focus only on the important part of the image and to speed up the training process. By default region of interest contains the whole image.



Changing region of interest.

4. Setting training parameters

- **Network depth** – predefined network architecture parameter. For more complex problems higher depth might be necessary.

- **Feature size** – the size of an small object or of a characteristic part. If images contain objects of different scales, it is recommended to use feature size slightly larger than the average object size, although it may require experimenting with different values to achieve optimal results.
- **Stopping conditions** – define when the training process should stop.

For more details read Deep Learning – Setting parameters.

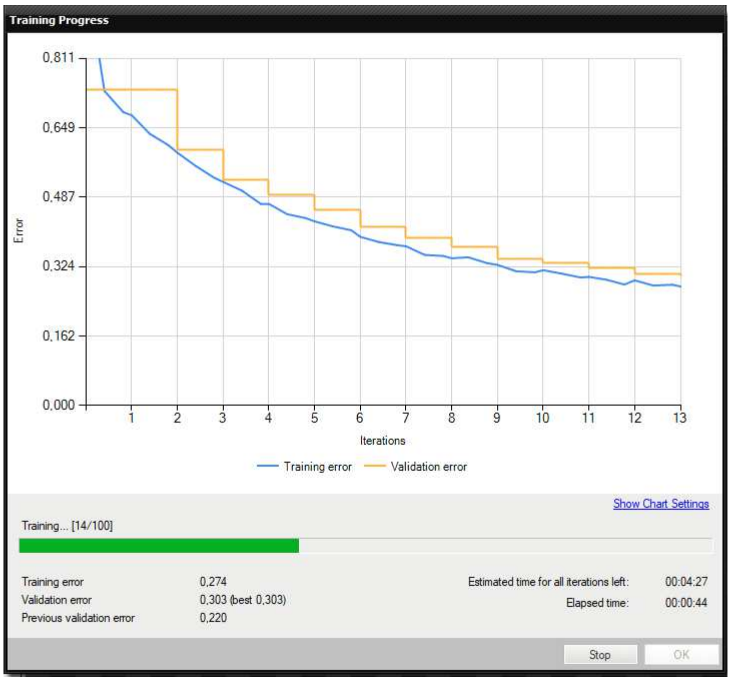
Details regarding augmentation parameters. Deep Learning – Augmentation

5. Performing training

During training, two main series are visible: training error and validation error. Both charts should have a similar pattern. If a training was run before the third series with previous validation error is also displayed.

More detailed information is displayed below the chart:

- current iteration number,
- current training statistics (training and validation error),
- number of processed samples,
- elapsed time.



Training... [14/100]

Training error	0,274	Estimated time for all iterations left:	00:04:27
Validation error	0,303 (best 0,303)	Elapsed time:	00:00:44
Previous validation error	0,220		

Training point location model.

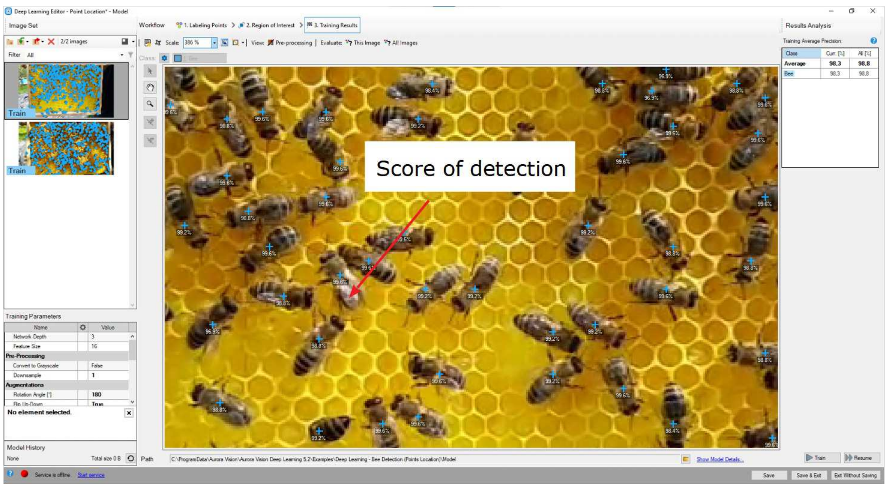
Training may be a long process. During this time, training can be stopped. If no model is present (first training attempt) model with best validation accuracy will be saved. Consecutive training attempts will prompt user whether to replace the old model.

6. Analyzing results

The window shows results of point location. Detected points are displayed on top of the images. Each detection consists of following data:

- visualized point coordinates,
- class (identified by a color),
- confidence score.

Evaluate: This Image and **Evaluate: All Images** buttons can be used to perform point location on the provided images. It may be useful after adding new training or test images to the data set or after changing the area of interest.



Training Parameters

Name		Value
Network Depth		3
Feature Size		16
Pre-Processing		
Convert to Grayscale		False
Downsample		1
Augmentations		
Rotation Angle [°]		180
Flip Up-Down		True

No element selected.

Model History: None — Total size 0 B

Service is offline. Start service

Training Average Precision:

Class	Curr. [%]	All [%]
Average	**98.3**	**98.8**
Bee	98.3	98.8

Point location results visualized after the training.

It is highly recommended to use data augmentations (appropriate to the task) to improve network's ability to generalize learned information. If results are still not satisfactory the following standard methods can be used to improve model performance:

- changing the feature size,
- providing more training data,
- increasing number of training iterations,

- increasing the network depth.

Locating objects

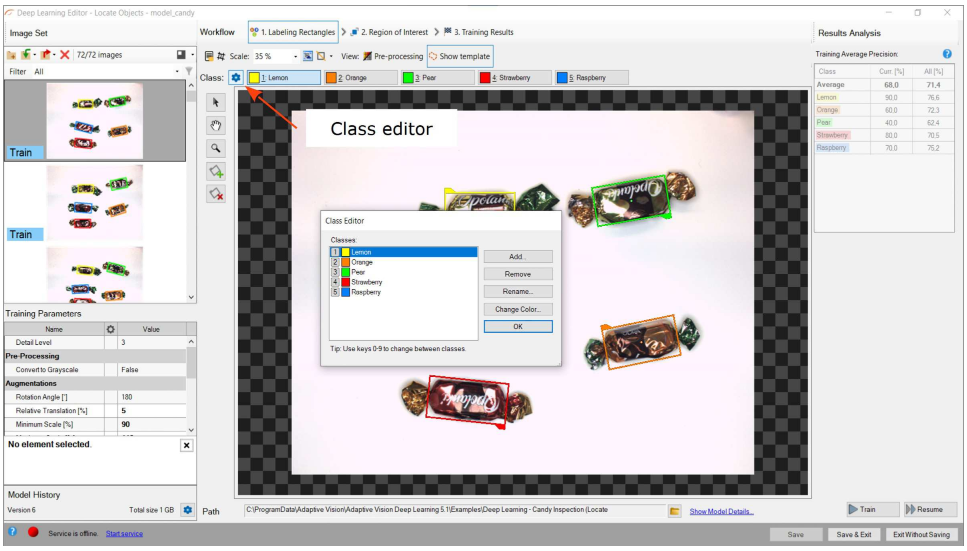
In this tool a user needs to draw rectangles bounding the objects in the scene and specify their classes. These images and rectangles are used to train a model to locate and classify objects in the input images. This tool doesn't require from a user to mark the objects as precisely as it is required for segmenting instances.

1. Defining classes

First, a user needs to define classes of objects that the model will be trained on and later used to detect. Object location model can deal with single as well as multiple classes of objects.

Class editor is available under the Class Editor button.

To manage classes, Add, Remove or Rename buttons can be used. Color of each class can be changed using Change Color button.



Classes:

1	🟨	Lemon
2	🟧	Orange
3	🟩	Pear
4	🟥	Strawberry
5	🟦	Raspberry

Tip: Use keys 0-9 to change between classes.

Training Parameters

Name	Value
Detail Level	3
Pre-Processing	
Convert to Grayscale	False
Augmentations	
Rotation Angle [°]	180
Relative Translation [%]	5
Minimum Scale [%]	90

No element selected.

Model History
Version 6 Total size 1 GB

Service is offline. Start service

Path C:\ProgramData\Adaptive Vision\Adaptive Vision Deep Learning 5.1\Examples\Deep Learning - Candy Inspection (Locate

Training Average Precision:

Class	Curr. [%]	All [%]
Average	**68.0**	**71.4**
Lemon	90.0	76.6
Orange	60.0	72.3
Pear	40.0	62.4
Strawberry	80.0	70.5
Raspberry	70.0	75.2

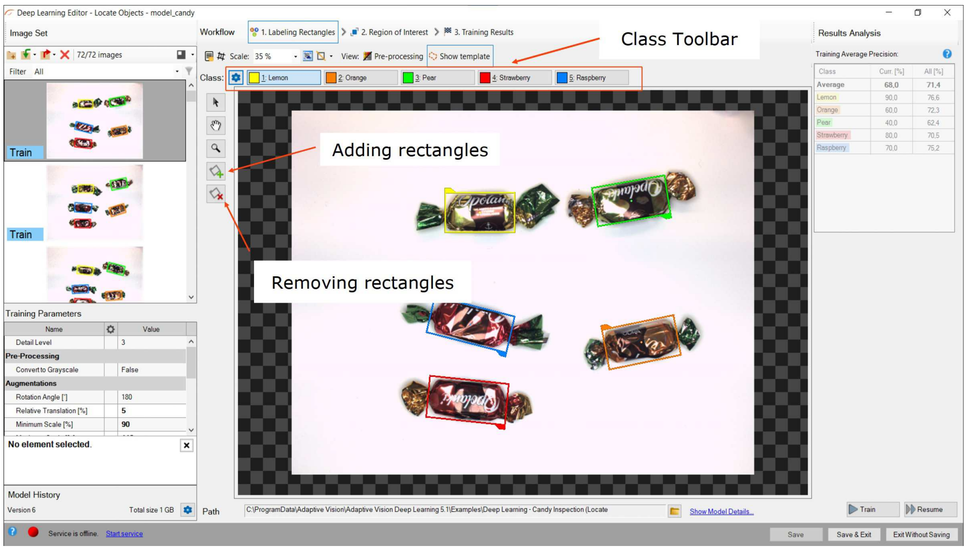
Using Class Editor.

2. Marking bounding rectangles

After adding training images and defining classes a user needs to mark rectangles in images.

To mark an object a user needs to click on a proper class from the Class Toolbar and click the Creating Rectangle button. Rectangles have the same color as previously defined for the selected class.

A rectangle can be selected directly on the image area. A selected rectangle can be moved, rotated and resized to fit it to the object, removed (Remove Region button) or has its class changed (*Right click on the rectangle » Change Class button*).



Training Parameters

Name	Value
Detail Level	3
Pre-Processing	
Convert to Grayscale	False
Augmentations	
Rotation Angle [°]	180
Relative Translation [%]	5
Minimum Scale [%]	90

No element selected.

Model History
Version 6 Total size 1 GB

Service is offline. Start service

Path C:\ProgramData\Adaptive Vision\Adaptive Vision Deep Learning 5.1\Examples\Deep Learning - Candy Inspection (Locate

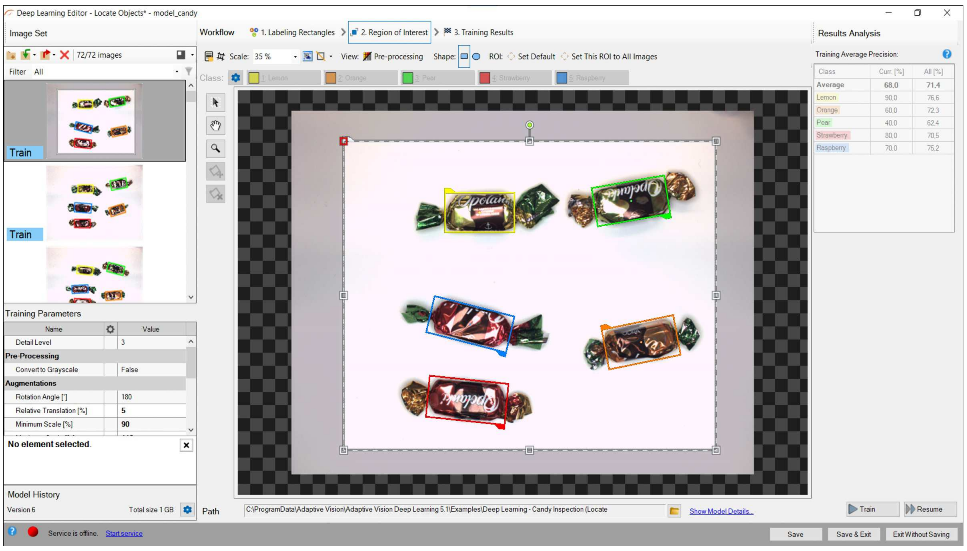
Training Average Precision:

Class	Curr. [%]	All [%]
Average	**68.0**	**71.4**
Lemon	90.0	76.6
Orange	60.0	72.3
Pear	40.0	62.4
Strawberry	80.0	70.5
Raspberry	70.0	75.2

Marking rectangles.

3. Reducing region of interest

Reduce region of interest to focus only on the important part of the image and to speed up the training process. By default region of interest contains the whole image.



Training Parameters

Name	Value
Detail Level	3
Pre-Processing	
Convert to Grayscale	False
Augmentations	
Rotation Angle [°]	180
Relative Translation [%]	5
Minimum Scale [%]	90

No element selected.

Model History

Version 6 Total size 1 GB

Service is offline. Start service

C:\ProgramData\Adaptive Vision\Adaptive Vision Deep Learning 5.1\Examples\Deep Learning - Candy Inspection (Locate

Results Analysis

Training Average Precision:

Class	Curr. [%]	All [%]
Average	**68.0**	**71.4**
Lemon	90.0	76.6
Orange	60.0	72.3
Pear	40.0	62.4
Strawberry	80.0	70.5
Raspberry	70.0	75.2

Changing region of interest.

4. Setting training parameters

- **Detail level** – level of detail needed for a particular classification task. For majority of cases the default value of 3 is appropriate, but if images of different classes are distinguishable only by small features, increasing value of this parameter may improve classification results.
- **Stopping conditions** – define when the training process should stop.

For more details read Deep Learning – Setting parameters.

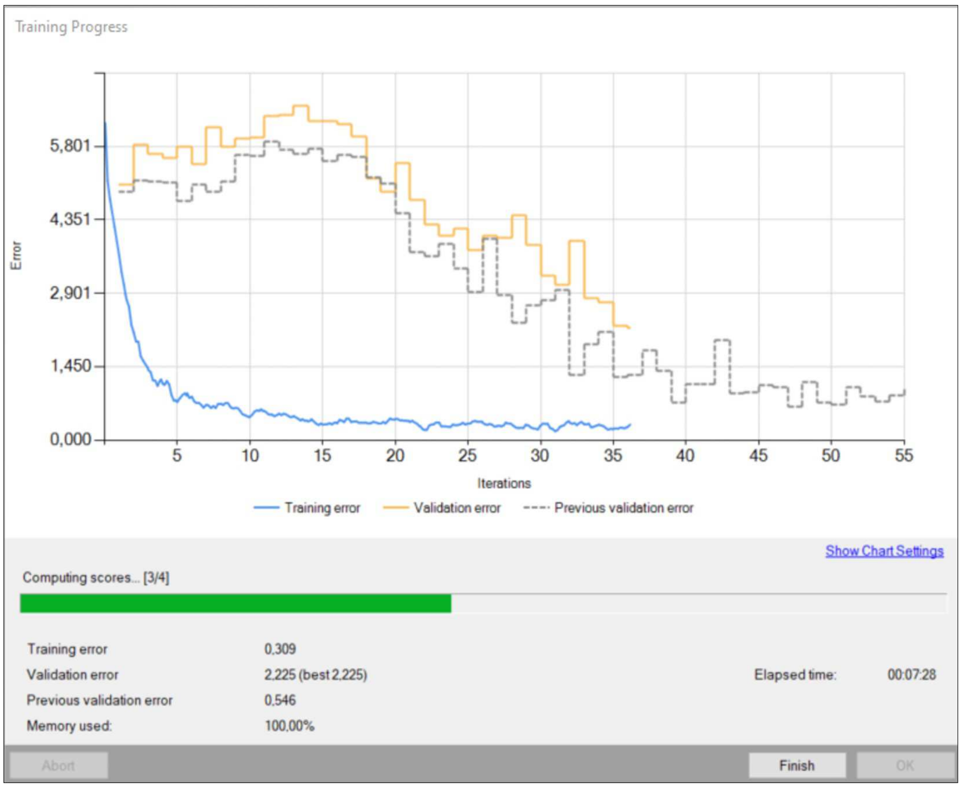
Details regarding augmentation parameters. Deep Learning – Augmentation

5. Performing training

During training, two main series are visible: training error and validation error. Both charts should have a similar pattern. If a training was run before the third series with previous validation error is also displayed.

More detailed information is displayed below the chart:

- current iteration number,
- current training statistics (training and validation error),
- number of processed samples,
- elapsed time.



Computing scores... [3/4]

```
Training error              0,309
Validation error            2,225 (best 2,225)              Elapsed time:    00:07:28
Previous validation error   0,546
Memory used:                100,00%
```

Training object location model.

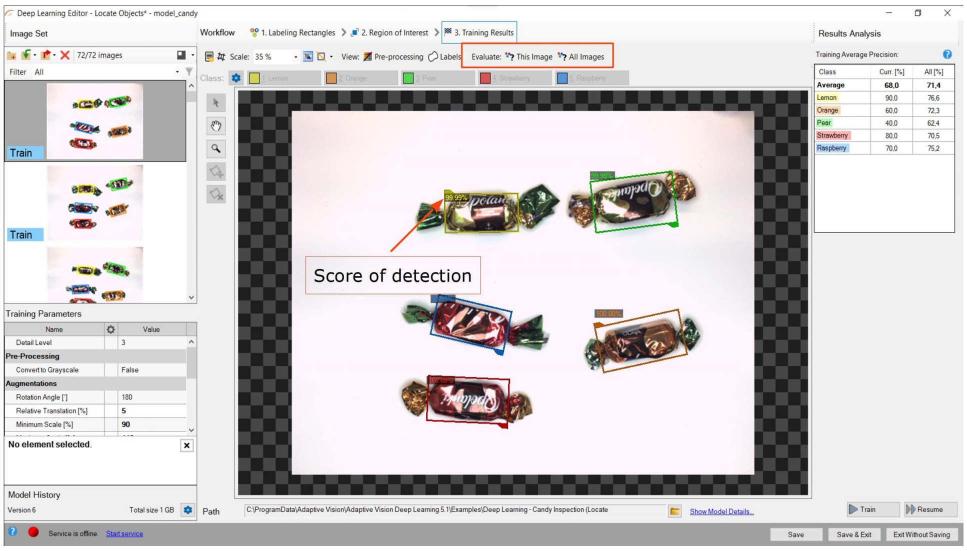
Training may be a long process. During this time, training can be stopped. If no model is present (first training attempt) model with best validation accuracy will be saved. Consecutive training attempts will prompt user whether to replace the old model.

6. Analyzing results

The window shows results of point location. Detected points are displayed on top of the images. Each detection consists of following data:

- visualized rectangle (object) coordinates,
- class (identified by a color),
- confidence score.

Evaluate: This Image and **Evaluate: All Images** buttons can be used to perform object location on the provided images. It may be useful after adding new training or test images to the data set or after changing the area of interest.



Results Analysis

Training Average Precision:

Class	Curr. [%]	All [%]
Average	**68,0**	**71,4**
Lemon	90,0	76,6
Orange	60,0	72,3
Pear	40,0	62,4
Strawberry	80,0	70,5
Raspberry	70,0	75,2

Training Parameters

Name	Value
Detail Level	3
Pre-Processing	
Convert to Grayscale	False
Augmentations	
Rotation Angle [°]	180
Relative Translation [%]	**5**
Minimum Scale [%]	**90**

No element selected.

Model History

Version 6 Total size 1 GB

Object location results visualized after the training.

It is highly recommended to use data augmentations (appropriate to the task) to improve network's ability to generalize learned information. If results are still not satisfactory the following standard methods can be used to improve model performance:

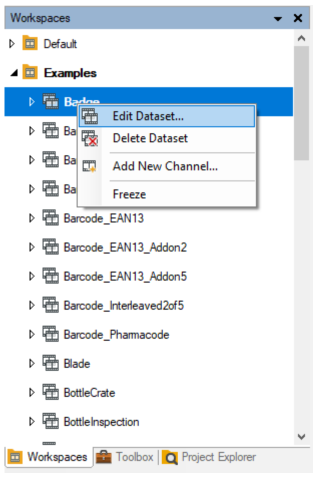
- changing the detail level,
- providing more training data,
- increasing number of training iterations or extending the duration of the training.

See also:

- Machine Vision Guide: Deep Learning – Deep Learning technique overview,
- Deep Learning Installation – installation and configuration of Aurora Vision Deep Learning.

Managing Workspaces

Workspace window enables user a convenient way to store datasets grouped by this same category or purpose. For example single workspace may be created for a single project like "Part Inspection" and each included dataset may represent inspection from different day or images of the different part types.

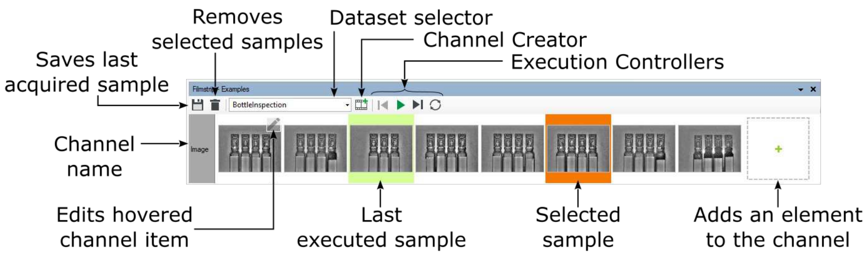


Example Workspace window with examples dataset.

Using Filmstrip Control

Overview

Filmstrip control is a powerful tool for controlling the execution of the program in the Offline Mode where the Filmstrip data is accessed with the ReadFilmstrip filters and through the bound outputs of the Online-Only filters.

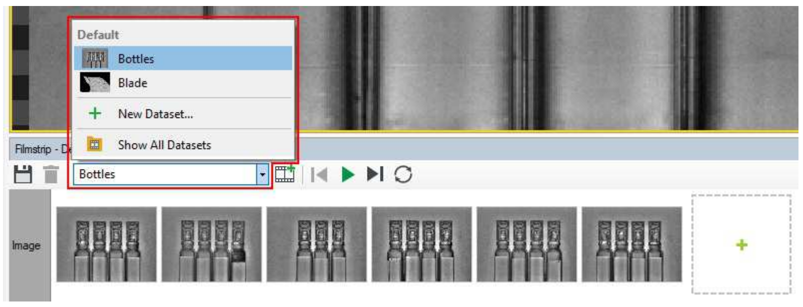


Filmstrip window with a one-channel dataset.

The data presented in the Filmstrip control is arranged in the grid layout, where the rows represents the Channels and the columns represents the Samples.

Additionally, the control enables most common operations over the current workspace, i.e., adding Datasets and Channels.

Changing the current dataset is as easy as selecting one from the datasets combo box:



Dataset selection.

To keep the program consistent, the to be selected dataset must contain channels with the same names as channels bound in the current Worker Task.

Common Tasks

- Dragging a channel from the Filmstrip control onto the Program Editor empty area inserts the ReadFilmstrip filter assigned to that channel,
- Dragging a channel from the Filmstrip control onto the filter instance output binds the output with that channel, as long as:
  - The filter is the Online-Only filter,
  - The filter is in the ACQUIRE section of the Worker Task,
  - The output data type is the same as the channel data type.
- Dragging files onto the Filmstrip control empty area creates a new Channel with the dragged files included.
- Dragging files onto the existing channel within the Filmstrip control appends the dragged data to that channel, if only the dragged data type match the channel type.
- Double-click on the Filmstrip sample executes one program iteration with the clicked sample. Requirements:
  - The Offline mode is active
  - There is at least one channel assigned in the Single-Threaded application's Worker Task or in the Multi-Threaded application's Primary Worker Task

See Also

1. Managing Workspaces - extensive description how to manage dataset workspaces in Aurora Vision Studio.
2. Offline Mode - the mode that enables access to the Channel data items.

Zebra
Aurora™ Vision