Zebra
Aurora™ Vision

Aurora Vision Studio 5.6

Getting Started

Created: 9/25/2025

Product version: 5.6.1.79554

Table of content:

Installation

Aurora Vision Studio system requirements

Aurora Vision Studio is supported on Windows 10 & 11 64-bit operating system, including embedded editions. It requires .NET 4.8 environment to be installed on your computer (if you happen to encounter any problems with running Aurora Vision Studio on the mentioned systems, it is highly possible that .NET is not installed properly, so please reinstall or repair it then).

The minimal display resolution required to present all the Aurora Vision Studio features (e.g. the Calibration Editor window) is 1280x1024.

To be able to fully take advantage of the integration possibilities provided by Aurora Vision Studio (e.g. creating user filters(tools)) you will also need Microsoft Visual Studio development environment, version 2015 or higher (Express Editions are also supported).

Will Aurora Vision Studio work on my PC?

We intentionally do not indicate the minimum system requirements. Rather than "will it work?", we should ask about "how will it work?" instead. The performance of your application might depend on a lot of factors. The most important among them are:

- Hardware. CPU and GPU benchmarks can be found in under this link. We hope this will help you to evaluate the hardware impact on the performance of your algorithm.
- Complexity of your algorithm (Number of tools)
- Your programming skills (Appropriate choice of the tools and how they are parametrized - especially important when more advanced tools i.e. Template Matching are used)
- Speed requirements of your vision system
- Size of the input data
- Capacity and configuration of the network

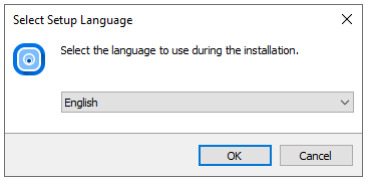
Installation Procedure

Important: Aurora Vision Studio setup program requires the user to have administrative privileges.

The installation procedure consists of several straightforward steps:

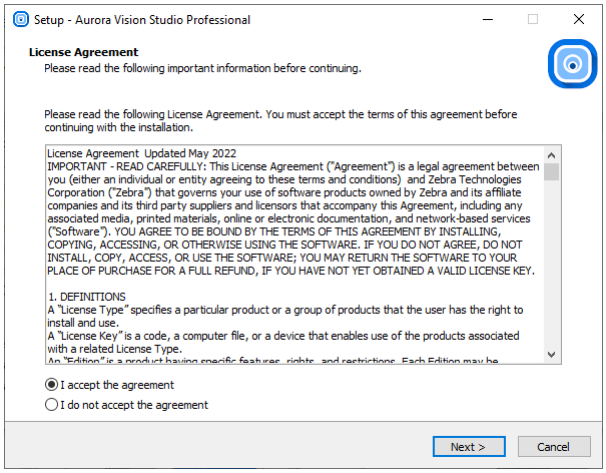
Setup language

Please choose the language for the installation process and click *OK* to proceed.



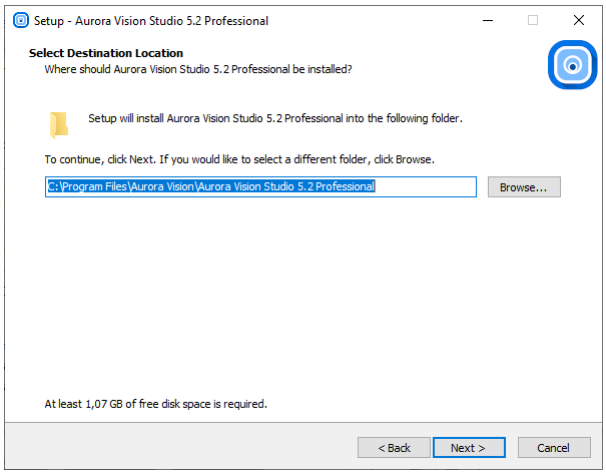
License agreement

Click *I accept the agreement* if you agree with the license agreement and then click *Next* to continue.



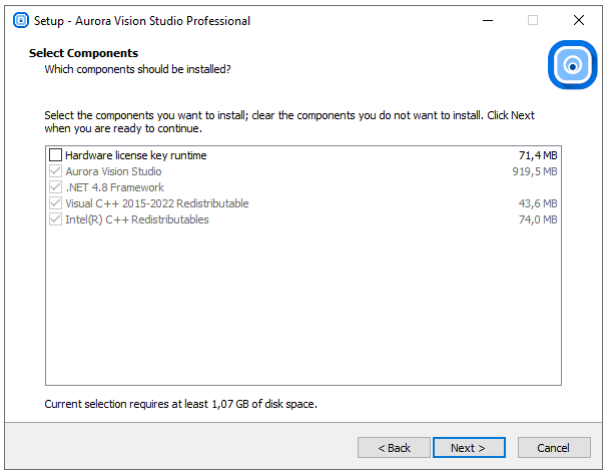
Localization on disk

Choose where Aurora Vision Studio should be installed on your disk.



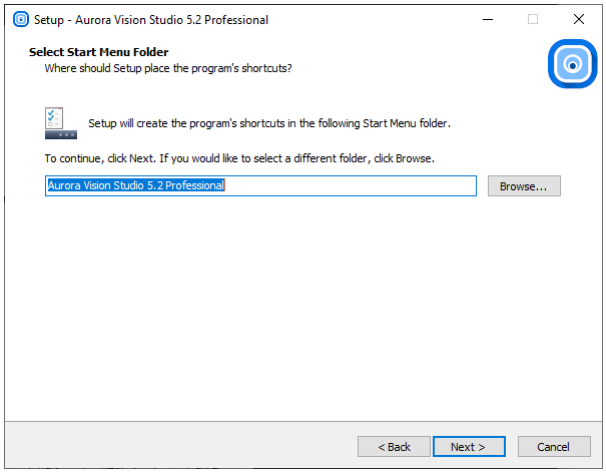
Components

Choose which additional components you wish to install.



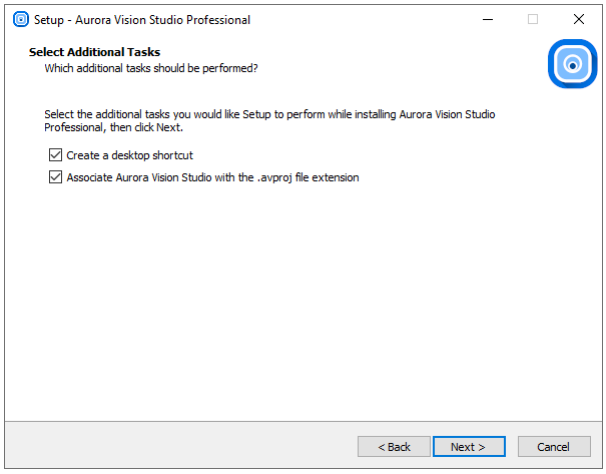
Start Menu shortcut

Choose if Aurora Vision Studio should create a Start Menu shortcut.



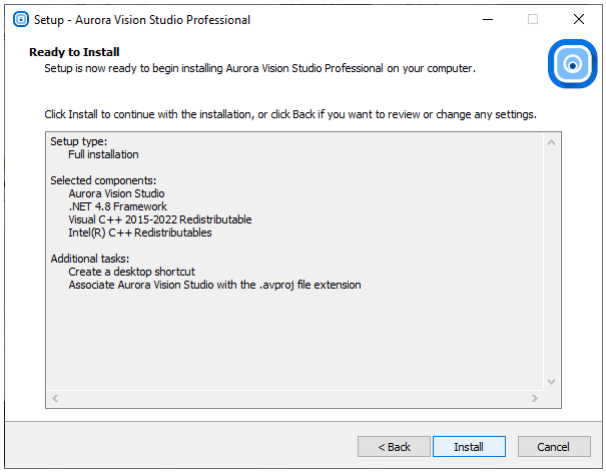
Additional Options

On this screen you can decide if the application should create a Desktop shortcut. You can also decide to associate Aurora Vision Studio with the .avproj files.

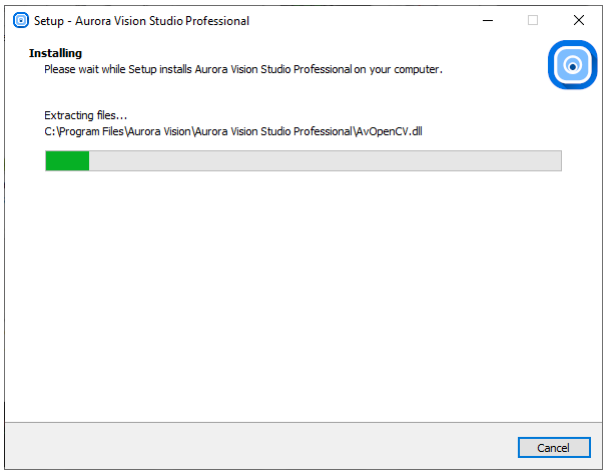


Installing

Click *Install* to continue with the installation or *Back* to review or change any settings.

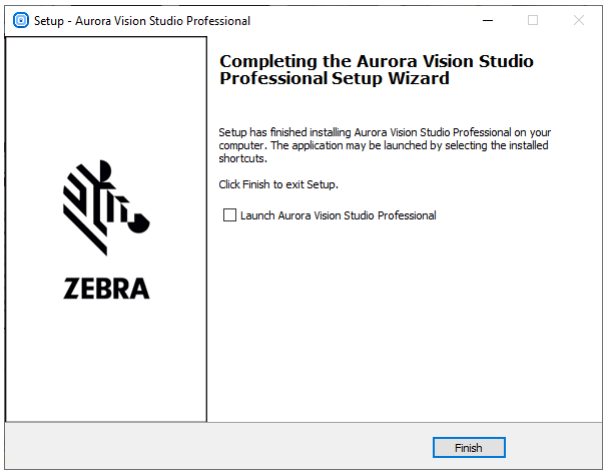


Please wait until all files are copied.



Final Options

At the end of the installation you are able to run the application.



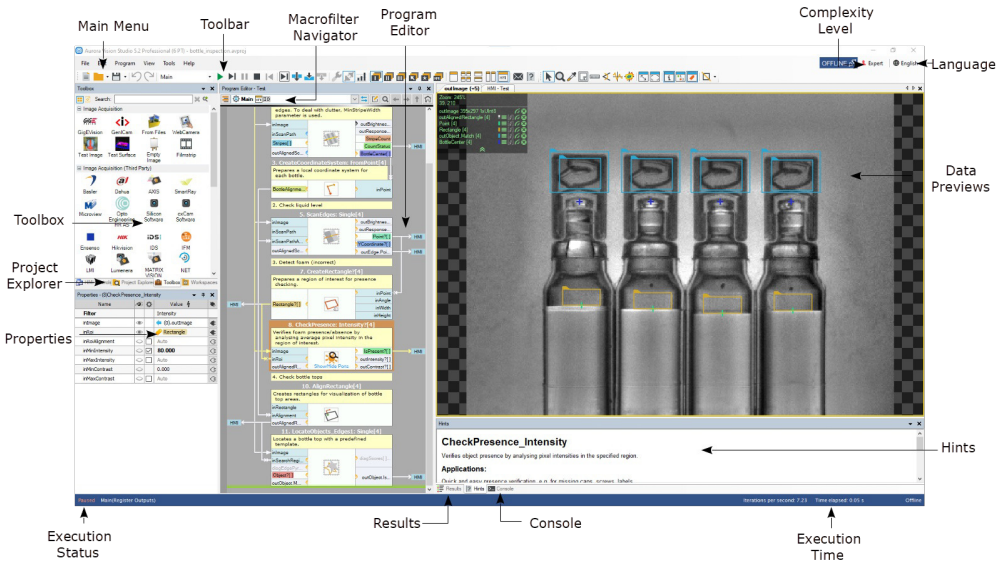
Deinstallation procedure

To remove Aurora Vision Studio from your computer please launch *Add or remove programs*, select *Aurora Vision Studio Professional » Uninstall* and follow the on-screen instructions.

Main Window Overview

Elements of the User Interface

The user interface of Aurora Vision Studio has been carefully designed for optimal user experience. All the main elements of the application are available on a single screen, as depicted below. The standard layout of windows can be changed in an arbitrary way – use this feature to adapt the application to your preferences. In particular, it is advisable to work with two HD monitors (or one curved ultra-wide, 34'') and use the second monitor to display undocked data previews or the HMI window.



Elements of the User Interface.

The purpose of the individual controls, going clockwise starting from the top left corner, is as follows:

- **Main Menu** and **Toolbar**
  This is a standard application menu that contains all major actions and options, as well as a control bar which provides convenient access to the most common actions.
  See also: Main Menu and Application Toolbar
- **Macrofilter Navigator**
  This control displays the tree of the macrofilter instances contained within the selected program. By default it is attached to the top of Program Editor, but can also be undocked and placed anywhere in the main window.
- **Program Editor**
  The Program Editor is the area within Aurora Vision Studio in which filters (tools) are placed and connected to create programs. To insert a filter there, just drag and drop one from the Toolbox or double-click on it.
  See also: Program Editor
- **Complexity Level**
  This option controls the amount of features that are available according to the user's personal experience and needs.
  See also: Complexity Levels
- **Language**
  This option controls the language of the user interface (English, German, Japanese, Polish, Simplified Chinese and Traditional Chinese).
- **Data Previews**
  These panels display the data computed by filters. If you drag and drop inputs and outputs of various filters, this is where you will see the corresponding data. Extra options can be found in the Toolbar.
  See also: Preview and Data Presenting
- **Hints**
  This window provides hints as for the use and function of various filters as well as possible solutions to the most common problems.
- **Execution Time**
  It displays the total execution time of the ongoing execution process.
  See also: Optimizing Image Analysis for Speed
- **Results**
  The Results window has two functions. First, it displays numerical and textual outputs of the selected filter as well as the relevant statistics. Second, it allows for setting limits upon output values, thus providing a quick and easy way for telling OK objects from NOK ones during inspection.
- **Console**
  The Console window informs the user about events related to project editing and execution. It is particularly important for finding errors.
- **Execution Status**
  It informs the user whether the program is running, paused or stopped. During execution it also displays the current program location (the call-stack).
- **Properties**
  The Properties window is where the parameters of filters and HMI controls are set.
- **Project Explorer**
  It displays a list of modules, macrofilters, global parameters and attachments contained within the currently edited project. Please note that you only see a list of definitions of macrofilters here, not their instances. One macrofilter (design) can have multiple instances (uses) and individual instances can be browsed in the Macrofilter Navigator control.
  See also: Browsing Macrofilters
- **Toolbox**
  It is a task-oriented catalog of filters commonly used in machine vision applications.
  See also: Toolbox

Please have a look at the the video tutorial about the interface on our YouTube channel:

Program Display

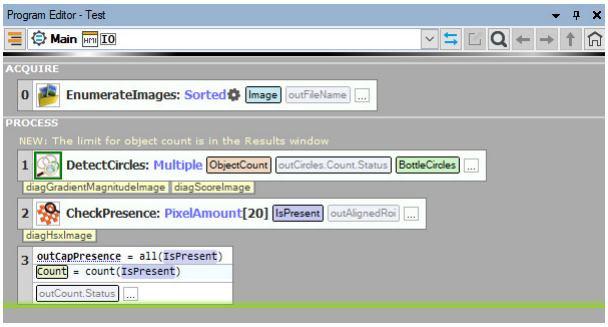
The Program Editor in Aurora Vision Studio is available with three display modes featuring different level of details. To change the display mode, expand the *View* bar and select **Minimal**, **Compact** or **Full**; each of them will suit various users to different degrees.

- **Minimal**

  Minimal program view is recommended for small-scale applications, typically ones that fit into one screen. In this mode, connections between filters are hidden and named output labels are used instead. The main benefit here is that basic applications can be easily presented in a single view. On the other hand when the program is getting bigger, dependencies between different program elements may become less clear and the program becomes more difficult to analyze.

  In Minimal program view you create connections by naming outputs and selecting them from a drop-down menu in the Properties window. The drag & drop action between filters is also possible, but the connection still remains hidden.

  This view is the only one available in the SMART edition.

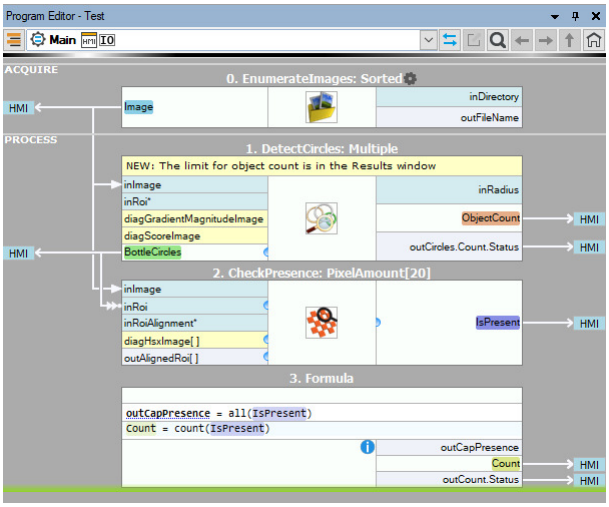


The Minimal view in the Bottle Crate example.

- **Compact**

  Compact program view is the default one in the Professional edition. It aims to provide the optimal level of detail by displaying explicit connections between filters while still hiding some inputs and outputs that are usually not being connected (you can use the Show/Hide action to change that for any port).

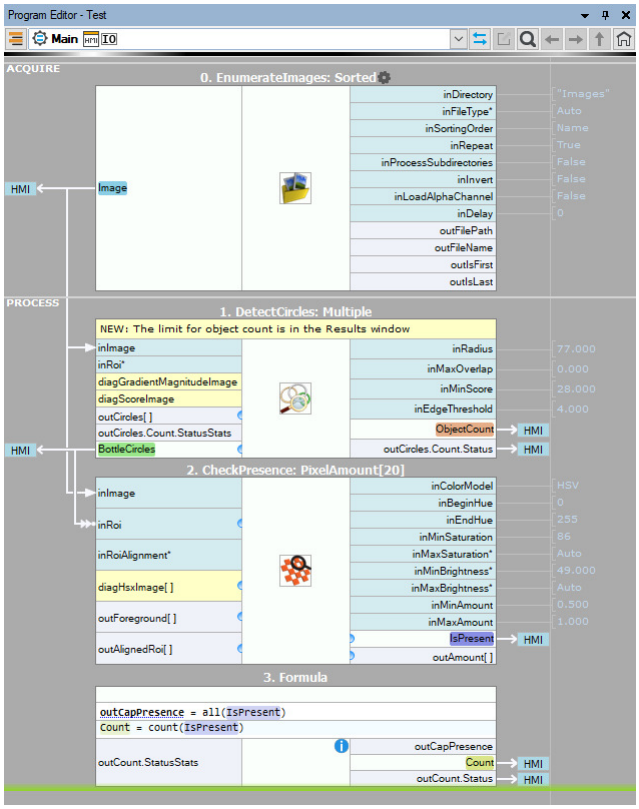
  This mode suits well both simple and complex applications alike.



The Compact view in the Bottle Crate example.

- **Full**

  Unlike in the Compact mode, here all the inputs and outputs of the given filter are always visible. Most importantly, this mode provides a handy preview of the filter's properties directly in the Program Editor – whenever possible they are displayed on the right margin of the editor. This, however, comes at the cost of higher verbosity which may impact readability.



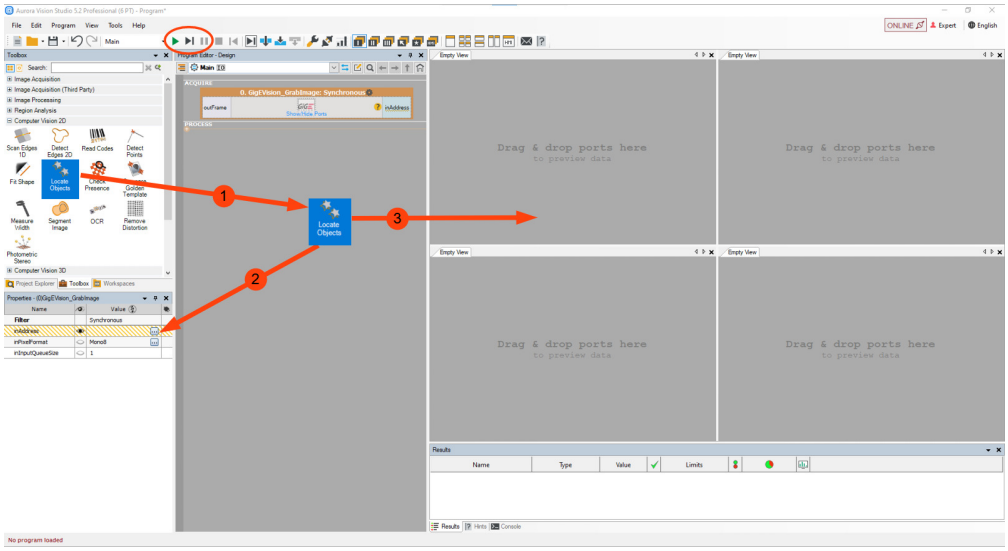
The Full view in the Bottle Crate example.

The Basic Workflow

Creating vision algorithms in Aurora Vision Studio comes down to three intuitive steps:

1. Drag & drop (or double-click) filters from the Toolbox to the Program Editor.
2. Drag & drop connections between filters or set constant input values in the Properties window.
3. Drag & drop filter outputs to the Data Previews panels.

Whenever a part of a program is ready, click *Run* or *Iterate* buttons in order to test it. The Console window at the bottom will also display some important information concerning the execution. Keep an eye on it – especially when something goes wrong.

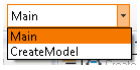


The workflow of Aurora Vision Studio.

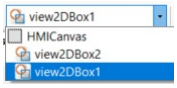
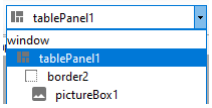
Main Menu and Application Toolbar

The Application Toolbar contains buttons for most commonly used actions and settings.

Here is the list of the most important menu commands:

File			
	New	Opens a new project.	Ctrl+N
	Open	Opens an existing project.	Ctrl+O
	Open Example	Opens an existing example project.	
	Open Tutorial	Opens an existing tutorial project.	
	Save	Saves the current project.	Ctrl+S
	Save As	Saves the current project in a specified location.	
	Connect to Remote Executor	Opens a window that provides connection with a remote Aurora Vision Executor, allowing application deployment of the current project. See also Remote Executor for details.	
	Export to Runtime Executable	Exports the current project to an executable file which can be run (but not edited) with Aurora Vision Executor.	
	Generate C++ Code	Creates a C++ program for the currently opened Aurora Vision program. See also C++ Code Generator for details.	
	Generate .NET Macrofilter Interface	Generates a .NET assembly which provides the chosen macrofilters as class methods. Generated assembly may be referenced in managed languages: C# or Visual Basic .NET. See .NET Macrofilter Interface Generator for more information.	
Edit			
	Undo	Reverts the last performed operation.	Ctrl+Z
	Redo	Re-performs the operation reverted with Undo command.	Ctrl+Y
	Rename Current Macrofilter	Renames current macrofilter.	F2
	Remove HMI	Unbinds HMI from the current project and clears all the HMI controls.	
	Remove All Breakpoints	Removes all execution breakpoints from the current project.	
Program			
	Startup Program (combo box)	Selects a worker task from which the execution process will start. When the execution is paused, shows active worker tasks an allows the user to switch to them. See Testing and Debugging for more details.	
	Run	Executes the program until all iterations are finished, the user presses *Pause* or *Stop* buttons or something else pauses program. See Testing and Debugging for more details.	F5
	Run Single Worker	Executes the program running only the primary worker task until all iterations are finished, the user presses *Pause* or *Stop* buttons or something else pauses program.	F5
	Iterate	Executes the program to the end of a single iteration of the outermost Task. See also Execution Process and Testing and Debugging for more information about iterations.	F6
	Pause	Suspends the current program execution immediately after the last invoked filter (tool) is finished.	Ctrl+Alt+Pause
	Stop	Stops the program immediately after the last invoked filter is finished.	Shift+F5
	Iterate Back	Executes the program to the end of a single iteration, reversing direction of enumerating filters. See Testing and Debugging for more details.	Shift+F6
	Iterate Current Macrofilter	Executes the current program to the end of the currently selected macrofilter. See Testing and Debugging for more details.	Ctrl+F10
	Step Over	Executes the next single instance of a filter or a macrofilter, without entering into the macrofilter.	F10
	Step Into	Executes the next single filter instance. If it is an instance of a macrofilter it enters inside of it.	F11
	Step Out	Executes all filters till the end of the current macrofilter, and exits to the parent macrofilter.	Shift+F11

	Run with Aurora Vision Executor	Executes the program using Aurora Vision Executor application installed on the local machine.	Ctrl+F5
	Diagnostic Mode	Turns on or off Diagnostic Mode, which controls whether filters compute additional information helpful for program debugging, but resulting in a slower program execution.	
	Program Statistics	Shows information about execution time of each filter in the selected macrofilter.	F8
View			
	Program Editor	Switches the Program Editor window visibility.	
	Filter Catalog	Switches the Filter Catalog window visibility.	
	Console	Switches the Console window visibility.	
	Filter Properties	Switches the Filter Properties window visibility.	
	Project Explorer	Switches the Project Explorer window visibility.	
	HMI Controls	Switches the HMI Controls window visibility.	
	Toolbox	Switches the Toolbox window visibility.	
	HMI Designer	Switches the HMI Designer window visibility. See Designing HMI for details.	
	WebHMI Designer	Switches the WebHMI Designer window visibility. See Designing WebHMI for details.	
	Hints	Switches the Hints window visibility.	
	Dock Open Windows	Docks all undocked data preview windows.	
	Program Display: Minimal	Switches Program Editor into a mode where no inputs or outputs are displayed, and connections are created by selecting data sources in the Properties window.	
	Program Display: Compact	Switches Program Editor into a mode where primary inputs and outputs are visible and can be connected in a visual way.	
	Program Display: Full	Switches Program Editor into a mode where all inputs and outputs are visible together with input values preview on the editor's margin.	
	1x1 Preview	Arranges data previews in a single tabbed window.	
	2x2 Preview	Arranges data previews in the 2 x 2 layout.	
	Arrange Horizontally	Arranges data previews horizontally.	
	Arrange Vertically	Arranges data previews vertically.	
	User-defined Preview Layout 1	Activates the first preview layout defined by the user.	
	User-defined Preview Layout 2	Activates the second preview layout defined by the user.	
	User-defined Preview Layout 3	Activates the third preview layout defined by the user.	
	Auto Preview Layout	Activates a preview layout which will be automatically filled in accordance with the currently selected filter.	
	HMI Design Layout	Activates a preview layout containing only the HMI window.	
	WebHMI Design Layout	Activates a preview layout containing only the WebHMI window.	
Tools			
	Check Project for Issues	Checks if current project has any issues.	
	Manage GenICam Devices	Opens a manager for enumerating and configuring GenICam/GenTL compatible cameras.	
	Manage GigE Vision Devices	Opens a manager for enumerating and configuring GigE Vision compatible cameras.	
	Macrofilters Preview Generator	Saves a graphical overview of selected macrofilters.	
	Edit HMI User Credentials File	Opens a window which allows to configure the credentials of password-protected HMI. See Protecting HMI with a Password for details.	
	Settings	Opens a window which allows to customize the settings of Aurora Vision Studio.	

Help			
	View Help	Opens up the documentation of Aurora Vision Studio.	F1
	Message to Support	Sends an e-mail to support with attached Aurora Vision Studio screenshot, log and optional user supplied message.	
	Download Remote Support Client	Downloads TeamViewer application that allows for remote support.	
	License Manager	Allows to view and manage available licenses for Aurora Vision products.	
	Check for Updates	Checks if there is newer version of Aurora Vision Studio.	
	About Aurora Vision Studio	Displays information about your copy of Aurora Vision Studio, e.g. the application version, loaded assemblies and plugins.	
HMI Designer			
	HMI elements list	Selects a HMI element which will be edited.	
	Send To Back	Sends the HMI control to back.	
	Bring To Front	Brings the HMI control to front.	
	Align Lefts	Aligns all marked HMI controls' lefts.	
	Align Centers	Aligns all marked HMI controls' centers.	
	Align Centers	Aligns all marked HMI controls' rights.	
	Align Tops	Aligns all marked HMI controls' tops.	
	Align Middles	Aligns all marked HMI controls' middles.	
	Align Bottoms	Aligns all marked HMI controls' bottoms.	
	Make Same Width	Makes all marked HMI controls' same width.	
	Make Same Height	Makes all marked HMI controls' same height.	
	Make Horizontal Spacing Equal	Makes all marked HMI controls' horizontal spacing equal.	
	Make Vertical Spacing Equal	Makes all marked HMI controls' vertical spacing equal.	
	Center horizontally	Centers the HMI control horizontally.	
	Center vertically	Centers the HMI control vertically.	
WebHMI Designer			
	WebHMI elements list	Selects a WebHMI element which will be edited.	
</> XAML	XAML	Shows the WebHMI code.	

Application Settings

Aurora Vision Studio is a customizable environment and all its settings can be adjusted in the Settings window, located in the *Tools » Settings* menu. Falling back to defaults is possible with the *Reset Environment* button located at the bottom of the window. Here is the list of the Application Settings:

1. Environment

General

Complexity level:
Selecting tools level complexity.

Language:
User interface language.

Theme:
User interface color theme.

Display short project name in the Title Bar
If checked, only name of the current project is displayed in the Main Window title. Otherwise, full path to the project file is displayed.

Floating point significant digits:
Sets the number of digits after floating point separator.

Startup

Load last project on application startup
Loads previously opened program at AVStudio startup.

Console

Show date column
Adds or removes 'Date' column in the Console.

Show time column
Adds or removes 'Time' column in the Console.

Show message level column
Adds or removes 'Level' column in the Console.

Previews

Display region border in image previews
Displays region bounding rectangle.

2. Program Execution

General

Break before undefined mandatory inputs
If checked, automatically breaks the program when the filter with undefined mandatory inputs is about to be executed.

Edit undefined mandatory inputs on break
If checked, tries to edit missing input data when program execution breaks before undefined mandatory inputs.

Break when an exception occurred
Automatically breaks the program when exception occurred.

Break when a warning occurred
Automatically breaks the program when warning occurred.

Break on assertion failed
Automatically breaks the program when assertion failed.

Diagnostic mode
Enables or disables computation of diagnostic output values.

Offline mode
Enables or disables the offline mode, in which data for filter outputs within the ACQUIRE section is acquired from local storage.

Display performance statistics automatically when execution is finished
Automatically opens the Statistics window after program execution.

Save changes before execution
Automatically saves program before execution start.

Previews update mode:
Sets how program will be visualized in context of previews refreshing.

HMI

Display HMI in Standalone Window
Displays HMI in a separate window as if it was run in Aurora Vision Executor.

3. Filters

Library

Auto reload filters
Tracks loaded filters directories for changes and reloads filters if any has been detected.

Filter Catalog

Close open groups
Allows only one opened group at a time.

Merge filters by group
Allows AVStudio to combine several filters items into single convenient tool.

Search bar enabled
Enables searching in the Filter Catalog.

Group categories by library
Filters in Filters Catalog will be split by library origin.

Toolbox

Show small icons in toolbox
Showing small icons in the Toolbox allows to unlock Toolbox window as a small convenient set of tools.

Use library view as staring page
Showing small icons in the Toolbox allows to unlock Toolbox window as a small convenient set of tools.

Filter Properties

Show context help
Shows help for current selected property at the bottom of the Filter Properties window.

Project Explorer

Show macrofilter usage
Shows usage count of each macrofilter.

4. File Associations

Default Program

Check .avproj file association on startup
Check if current program is the default for opening .avproj Files on startup.

5. Program Edition

Macrofilter Navigator

Expanded tree view
Shows parent-child relations between macrofilters. If not checked, macrofilters will be listed without any marked relations.

Show macrofilters preview in tooltips
Shows the graphical previews of macrofilters as tooltips.

Auto hide macrofilter list
Hides the expanded list of macrofilters on selection.

Program Editor

Show warning when connecting diagnostic outputs
Enables warning on creating diagnostic connections.

Smooth variant switch
Enables animated variants switch.

Show comments
Enables filter instances comments.

Wrap comments
Wraps long comments or, when unchecked, clips comments to single line.

Wrap formulas
Expands formula blocks to display entire formulas.

Tab size
Number of spaces the tab is equal to in the inline editors, e.g. formula editor.

Block quick commands visibility:
Defines the visibility of block quick commands (for adding new inputs and outputs).

Editor Zoom [%]:
Defines a program editor font and image size.

Filter icon size:
Defines the size of the filter icon in Program Editor.

Use larger snap size for editor points
Allows easier dragging of points in editors, useful on touchscreens.

Editor view type:
Enables choosing the information amount displayed in the program editor.

Highlight compatible ports while creating connections
Highlights compatible ports when either dragging global parameter or filter port.

Tooltip delay [ms]:
Defines a delay in milliseconds of the tooltip appearance for hovered program elements.

Show indices of filter instances
If checked, indices of filter instances will be visible.

Show array and conditional markers on filter ports
If checked, ports that accepts or introduce array ([]), conditional (?) or optional (*) data will include appropriate information in their names.

Add generic filters as uninstantiated
If checked, generic filters will be added to program as uninstantiated, without prompting for choosing data type.

Autoconnect filters on insertion (experimental)
If checked, automatically connects filter image and coordinate system inputs with deduced outputs.

Open the default editor when adding the filters to the program
If checked, automatically opens the default data editor for inserting filter.

Close filter variant selection dialog after filter insertion
If checked, filter variant selection dialog will close after inserting one filter. Uncheck to insert multiple filters.

Preserve port previews when extracting macrofilter
If checked, port previews of filters extracted to new macrofilter will be preserved.

Program Analysis

Check array synchronization problems during program edition
If checked, potential array synchronization problems caused by user action are detected and require confirmation.

Show array synchronization in Program Editor
If checked, array synchronization is presented in Program Edition.

Check array synchronization during program loading
If checked, array synchronization is verified during program loading.

Check array synchronization before program start
If checked, array synchronization is verified before program start.

Ignore warnings in disabled macrofilters during program execution
If checked, warnings in disabled macrofilters will be ignored during program execution.

6. Results

View

Flat view
If checked, outputs are arranged in a flat list in the results. Otherwise, outputs are only displayed in a tree layout according to their logical relationship to their parent outputs.

Show statistics columns
If checked, statistics-related columns are displayed.

Use output labels
If checked, labeled outputs are presented as their Data Source Labels. Otherwise, original output name is used.

Contents
Defines the source for the results control contents. Either the selected filters or all filters in the current macrofilter (variant).

Show only visible outputs
If checked, hidden outputs are also hidden in the results.

Show only textual outputs
If checked, only outputs of textual data type (e.g. String, Integer, Real, etc.) are shown in the results.

Show only checked outputs
If checked, only outputs which are checked.

Show only labeled outputs
If checked, only outputs which have the label defined.

7. Project Files

Project Explorer

Watch for project file changes
Notifies, when project file has been changed i.e. by external program.

Project Binary Files

Binary Files Compression Level:
Configures project avdata files compression level.

8. Messages

Show these messages:

Warn when trying to modify running program
If checked, warning will be displayed before stopping program to make modification in it.

Ask about opening an example documentation on startup
If checked, application will notify about an example's description in documentation.

Warn when trying to replace existing preview
When checked, a warning is shown to prevent from accidentally removing carefully prepared preview.

Show warning when reconnecting inputs
Enables warning on creating a connection which would replace existing connection.

Show message after extracting macrofilter
If checked, additional help message is shown each time a macrofilter is extracted from selection.

Warn about using themes at high resolution and DPI
If checked, application will notify about the recommendation not using themes at high resolutions and DPI.

Ask for refactoring older project to use sections on loading project
Enables warning on creating a connection which would replace existing connection.

Show message after Workspace change when executing the program
If checked, the application will notify about program stop when the Workspace has changed.

Notify when path was automatically changed to relative
When checked, a message is shown after selected file path was automatically changed to relative.

Introduction to Data Flow Programming

Important: Aurora Vision Studio does not require the user to have any experience in low-level programming. Nevertheless, it is a highly specialized tool for professional engineers and a fully-fledged visual programming language. You will need to understand its four core concepts: Data, Filters (Tools), Connections, Macrofilters and the general program structure defined by four Sections.

Data

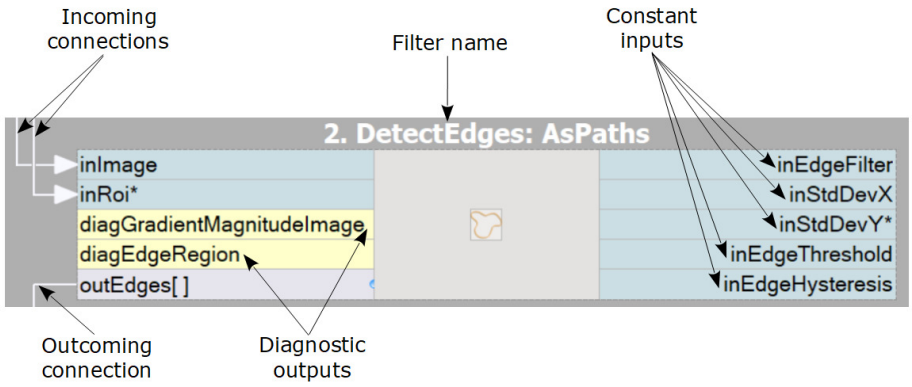
Aurora Vision Studio is a data processing environment so data is one of its central concepts. The most important fact about data that has to be understood is the distinction between *types* (e.g. Point2D) and *values* (e.g. the coordinates *(15.7, 4.1)*). Types define the protocol and guide the program construction, whereas values appear during program execution and represent information that is processed. Examples of common types of data are: Integer, Rectangle2D, Image.

Aurora Vision Studio also supports *arrays*, i.e. variable-sized collections of data items that can be processed together. For each data type there is a corresponding array type. For example, just as *4* is a value of the Integer type, the collection *{1, 5, 4}* is a value of the IntegerArray type. Nested arrays (arrays of arrays) are also possible.

Filters (Tools)

Filters are the basic data processing elements in data flow programming. In a typical machine vision application there is an image acquisition filter at the beginning followed by a sequence of filters that extract information about regions, contours, geometrical primitives and then produce a final result such as a pass/fail indication.

A filter usually has several inputs and one or more outputs. Each of the ports has a specific type (e.g. Image, Point2D etc.) and only connections between ports with compatible types can be created. Values of unconnected inputs can be set in the Properties window, which also provides graphical editors for convenient defining of geometrical data. When a filter is invoked (executed), its output data can be displayed and analyzed in the Data Preview panels.



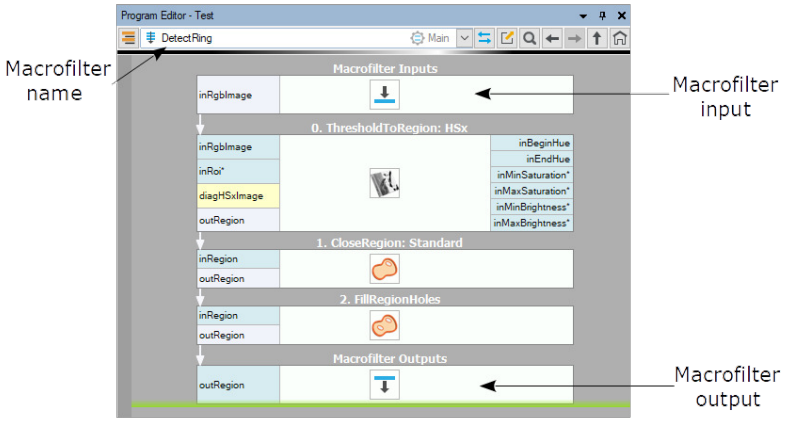
Connections

Connections transmit data between filters, but they also play an important role in encapsulating much of the complexity typical for low-level programming constructs like loops and conditions. Different types of connections support: basic flow of data , automatic conversions , array (*for-each*) processing and conditional processing . You do not define the connection types explicitly – they are inferred automatically on the *do what I mean* basis. For example, if an array of regions is connected to an input accepting only a single region, then an array connection is created and the individual regions are processed in a loop.

Macrofilters

Macrofilters provide a means for building bigger real-life projects. They are reusable subprograms with their own inputs and outputs. Once a macrofilter is created, it appears in the Project Explorer window and since then can be used in exactly the same drag and drop way as any regular filter.

Most macrofilters (we call them Steps) are just substitutions of several filters that help to keep the program clean and organized. Some other, however, can create nested data processing loops (Tasks) or direct the program execution into one of several clearly defined conditional paths (Variant Steps). These constructs provide an elegant way to create data flow programs of any complexity.



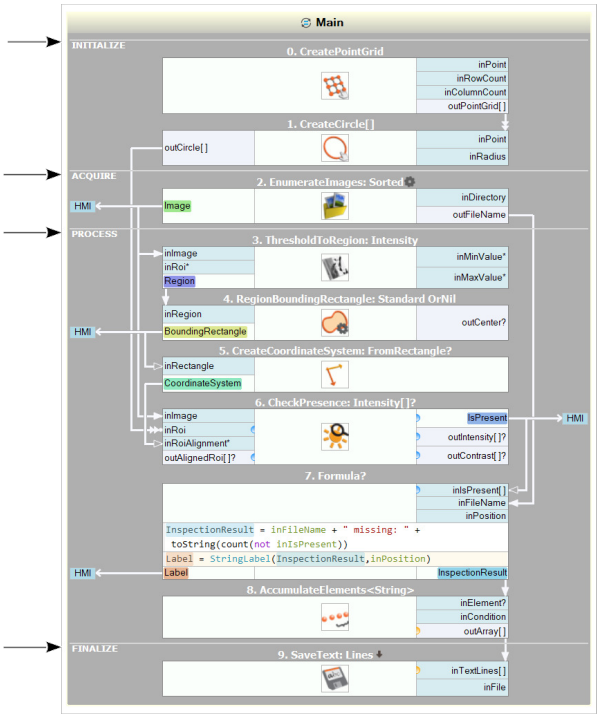
c++

Data and types are very similar to what you know from C++. We also have a generic collection type – *array* – which is very similar to *std::vector*. Filters and macrofilters are just equivalents of functions. But, instead of a single returned value they often have several output parameters. Connections correspond to local variables, which do not have to be named. On the other hand loops and conditions in Aurora Vision Studio are a bit different to C++ – the former are done with array connections or with Task macrofilters, for the latter there are conditional connections and Variant Step macrofilters. See also: Quick Start Guide for the C/C++ Programmers.

Sections

In order to improve clarity of applications and make it easier for the user to manage the data flow, starting from Aurora Vision Studio 5.0 we have introduced a new feature called "Sections". These special areas visible in the Program Editor are responsible for dividing the application code into four consecutive stages. Placing filters in the right section makes the application more readable and reduces necessity of using macrofilters in simple applications.

Currently, there are four sections available: **INITIALIZE**, **ACQUIRE**, **PROCESS** and **FINALIZE**.



INITIALIZE – this section should consist of filters that have to be executed only once, before the loop is started. The filters in this section will not be repeated during the loop. Such filters are usually responsible for initiation of a connection (e.g. GigEVision_StartAcquisition, TcpIp_Accept) or setting values of constant parameters for application execution.

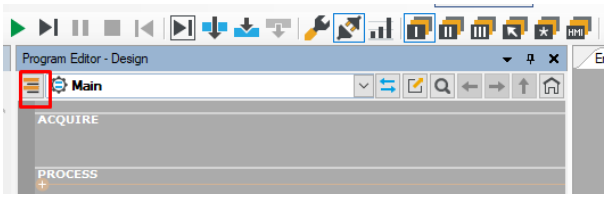
ACQUIRE – this section is intended to include filters from loop generation category like EnumerateImages or GigEVision_GrabImage that generate stream of data that will be processed or analyzed in the next section. Filters in this section will be executed in every iteration.

PROCESS – this is the main section that contains filters responsible for analyzing, processing and calculation of data. In most applications the **PROCESS** section will be largest one because its main purpose is to perform all the key tasks of the application. Filters in this section will be executed in every iteration.

FINALIZE – filters placed in this section are executed only once, after the loop. In most cases this section is used to close all the connections and save or show the inspection results (e.g. GigEVision_StopAcquisition, TcpIp_Close, SaveText). This section is executed only if the task macrofilter finishes without exceptions - it is not executed if an error occurs, even if it is handled by the error handler.

By default, sections are not visible inside Step and Variant Step macrofilters. However, the view in these macrofilters can be extended with **INITIALIZE** and **PROCESS** sections. In Worker Task and Task macrofilters the default view consist of **ACQUIRE** and **PROCESS** sections and can be extended to all four sections.

To change the default view of the sections in the Program Editor click on the button located to the left of the macrofilter name in the top bar of the Program Editor



Running and Analysing Programs

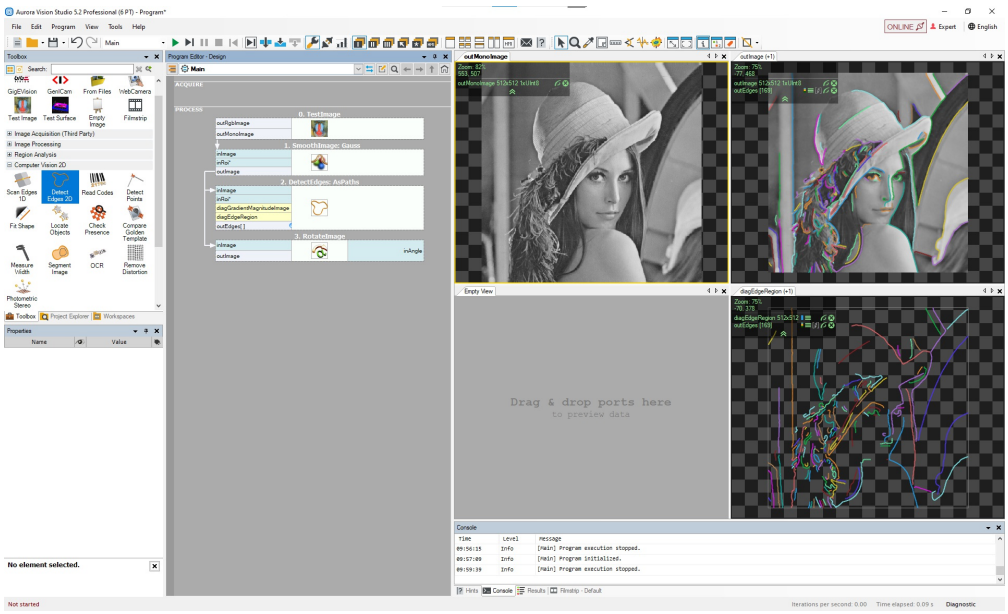
Example Projects

One of the best ways to learn Aurora Vision Studio quickly is to study the example projects that come with the application. The *File » Open Example...* command is a shortcut to the location they are stored in. The list of available projects is also available in the Program Examples section.

Executing Programs

When a project is loaded you can run it simply by clicking the *Run* button on the Application Toolbar or by pressing F5. This will start continuous program execution; results will be visible in the Data Preview panels. You can also run the program iteration by iteration by clicking Iterate or pressing F6.

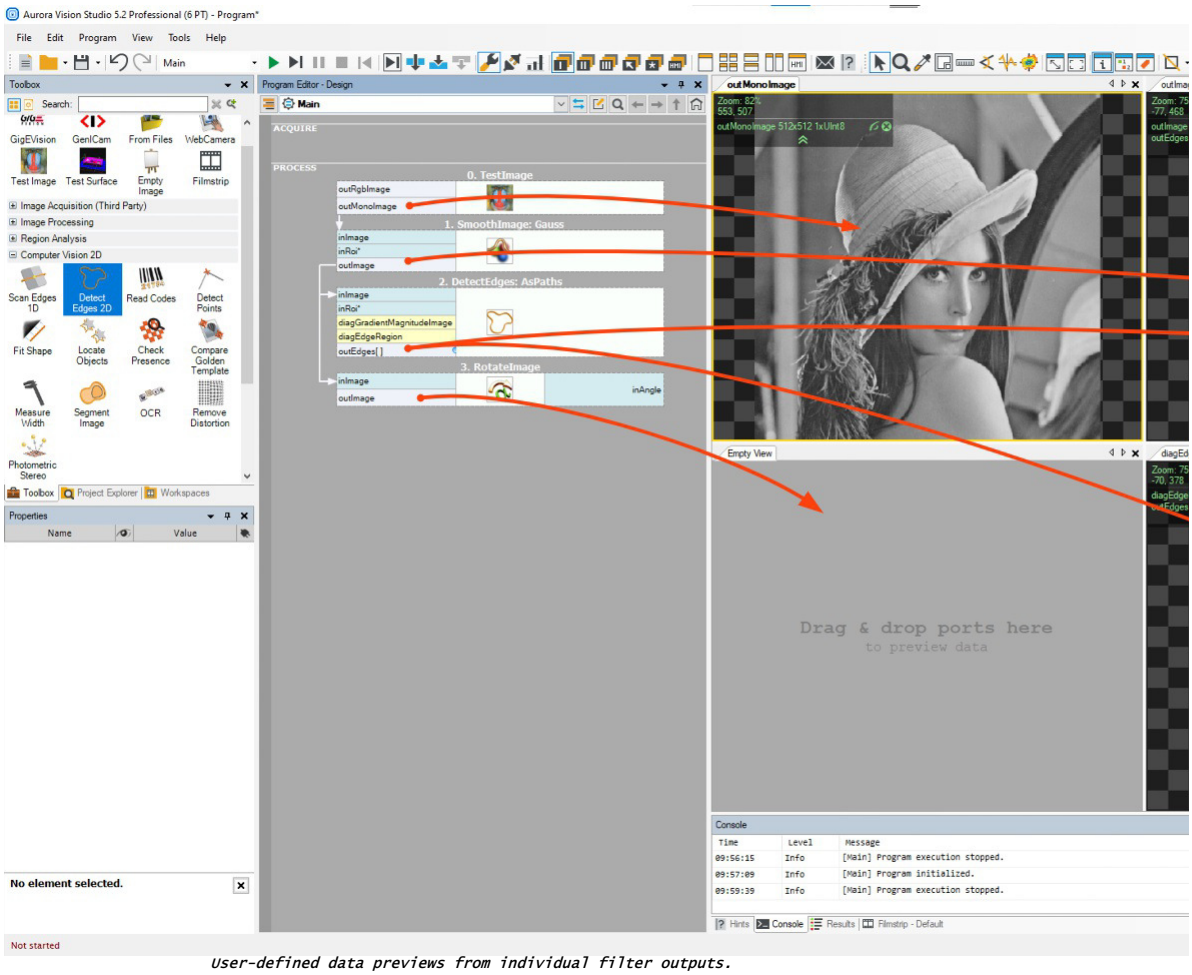
When a project is loaded from a file or when new filters (tools) are added, the filter instances are displayed subdued. They become highlighted after they are invoked (executed). It is also possible to invoke individual filters one by one by clicking *Step Over* or *Step Into*, or by pressing F10 or F11 respectively. The difference between *Step Over* and *Step Into* is related to macrofilters – the former invokes entire macrofilters, whereas the latter invokes individual filters inside.



A program with four filter instances; three of them have been already invoked.

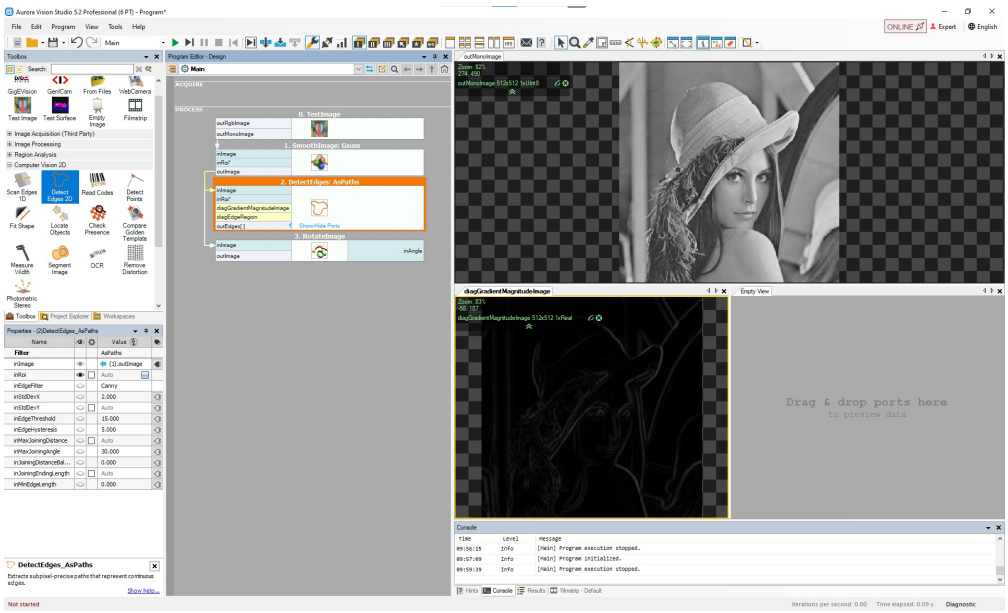
Viewing Results

Once the filters (tools) are executed, their output data can be displayed in the Data Previews panels. To display a value of a particular output, just drag and drop from the port of the filter to a Data Preview panel. Remember to press and hold left mouse button during entire movement from the port to the preview. Multiple data can be often displayed in multiple layers of a single preview. This is useful especially for displaying geometrical primitives, paths or regions over images.



User-defined data previews from individual filter outputs.

In bigger projects you will also find it useful to switch between three different layouts, which can be created to visualize different stages of the algorithm, as well as to the automatic layout mode, which is very useful for interactive analysis of individual filters:

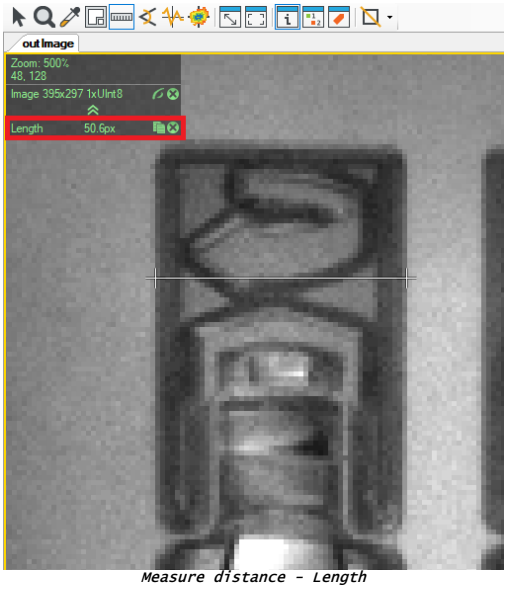


Automatic data previews – the layout adapts to the currently selected filter.

Analysing Data

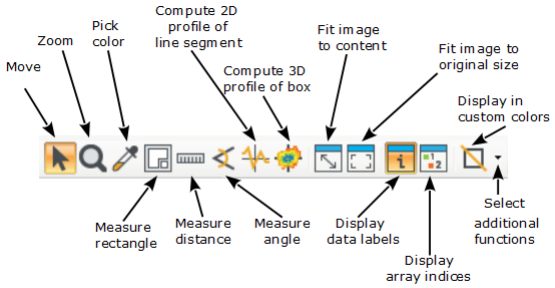
Data displayed in the Data Preview panels can be analyzed interactively. There are different tools for different types of data available in the main window toolbar. These tools depend on currently selected preview which is marked with a yellow border when window is docked.

Measure rectangle - Selection



Measure distance - Length

For the most common Image type the Data Preview window has the following appearance:

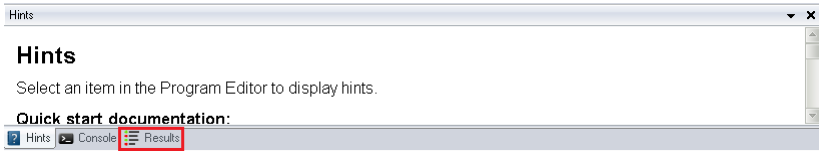


Additionally, there are several usability enhancements:

- **Mouse Wheel** – zooms in or out the image.
- **3rd Mouse Button + Drag** – moves the view.
- **Right Click** – opens a context menu and allows to save the view to an image file.

Results Control

While analyzing your application, it is useful to switch to the Results control available near the bottom of the screen. If you cannot see it, you need to enable it through *View » Results*.



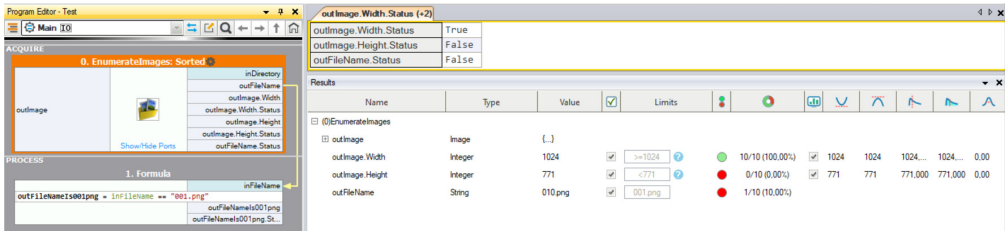
The location of the Results control (when enabled).

Upon clicking on a filter or checking a group of filters, a table with appropriate data will be displayed.

- **Name**
  Displays the name of the selected filter or filters as well as some of their outputs, including diagnostic ones (if the Diagnostic Mode is enabled).
- **Type**
  Displays the type of data returned by the given output.
- **Value**
  Displays the type of data returned by the given output in the last iteration.
- **Checked**
  If the given output can be subject to a Pass/Fail inspection (so any output returning a numerical value but also such types as String or Bool), you may check the box visible in this column in the corresponding row, thus enabling the following few columns. The application must be stopped for it to be allowed.
  - **Limits**
    In this parser you can insert a simple formula describing the criterion for a Pass (hover the cursor over the question mark to see examples). The application must be stopped for it to be allowed.
  - **Pass/Fail status**
    A green or red dot will appear signifying whether the criterion described under Limits has been met or not.
  - **Pass/Fail status statistics**
    Displays the number of iterations in which this particular output has so far met the criterion described under Limits.
- **Monitored**
  If the given output can be subject to a mathematical analysis, you may check the box visible in this column in the corresponding row in order to enable the following few statistics. The application must be stopped for that to be allowed.
  - **Minimum**
    Displays the lowest value so far returned.
  - **Maximum**
    Displays the highest value so far returned.
  - **Mean**
    Displays the mean value of those so far returned.
  - **Median**
    Displays the median value of those so far returned.

- **Standard Deviation**
  Displays the standard deviation value of those so far returned.

Below you can see a modified Fiducial Markers example. Two filters display the application of the Results control.



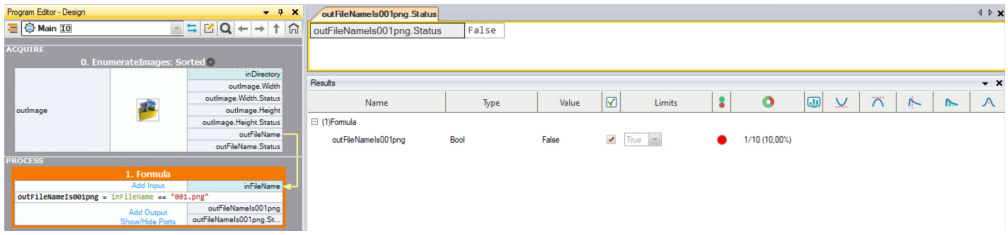
The Results control in action: types Integer and String.

After performing ten iterations on equally-sized images, we can see that:

- The **outImage.Width** Integer output returned True on all ten iterations (because the value was the same and the parser allowed it to be either the same or higher).
- The **outImage.Height** Integer output returned False on all ten iterations (because the value was never lower and the parser only allowed it to be lower).
- The **outFileName** String output returned True on one out of ten iterations (because the number within the name kept increasing so it did not stay the same).

Note how next to the parsers applied to numerical values there is a blue question mark . If you hover your cursor over it, you will see suggestions of formulas that you can use within the parser. These suggestions are not available with non-numerical types of data. Also note how after executing the program, new out...Status outputs became available in the filter (they are shown in the preview window above the Results control).

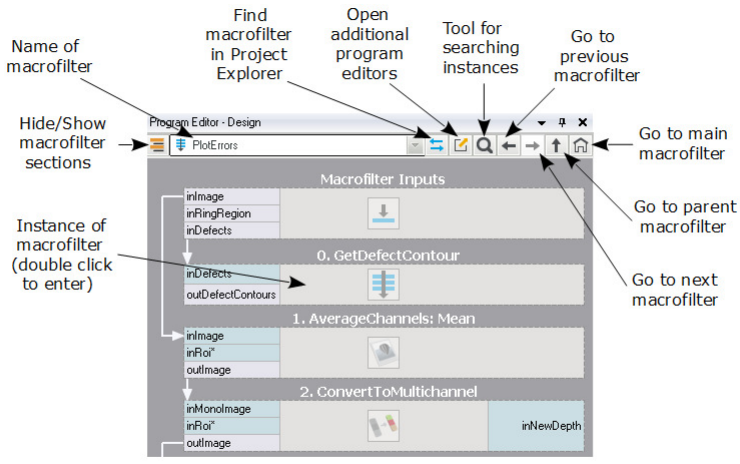
Below you can see the file name being verified again, this time as the Bool type output **outFileNameIsOO1png**. The Results control shows the identical effect to that of the outFileName String output above—except that this time the Limits parser can only be set either to True or to False.



The Results control in action: type Bool.

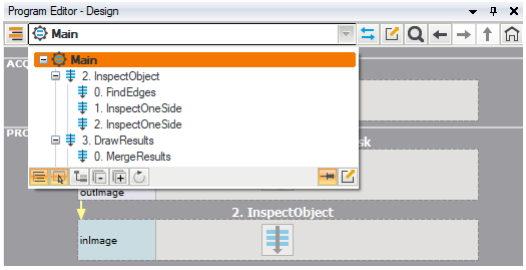
Browsing Macrofilters

Except for the most simple applications, programs in Aurora Vision Studio are composed of many so called macrofilters (subprograms). Instances of macrofilters in the Program Editor can be recognized through the icon which depicts several blue bars. Double clicking on an instance opens the macrofilter in the Program Editor.

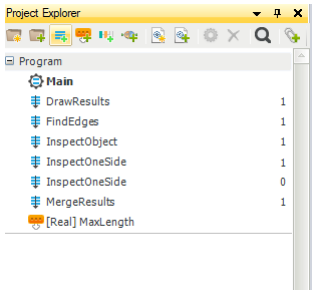


Browsing macrofilters in the Program Editor.

There are two other ways of browsing macrofilters. One is through the Macrofilter Navigator at the top of the Program Editor, which displays *instances* of macrofilters (how they are actually used and nested in the program). Another is through the Project Explorer, which displays *definitions* of macrofilters (a plain list of all macrofilters from which the user can create instances by dragging and dropping them to the Program Editor; double clicking on an item here opens the macrofilter in the Program Editor).



Instances of macrofilters in the Macrofilter Navigator.



Definitions of macrofilters in the Project Explorer.

C++

Definitions of macrofilters correspond to definitions of functions in C++, whereas instances of macrofilters correspond to function calls on a call-stack. Unlike the C++, there is no recurrence in Aurora Vision Studio and each macrofilter definition has a constant and finite number of instances. Thus, we actually have a static call-tree instead of a dynamic call-stack. This makes program understanding and debugging much easier.

Analysing Operation of a Single Macrofilter

The *Iterate Current Macro* command can be very useful when you want to focus on a single macrofilter in a program with many macrofilters. It executes the whole program and pauses each time when it comes to the end of the currently selected macrofilter instance.

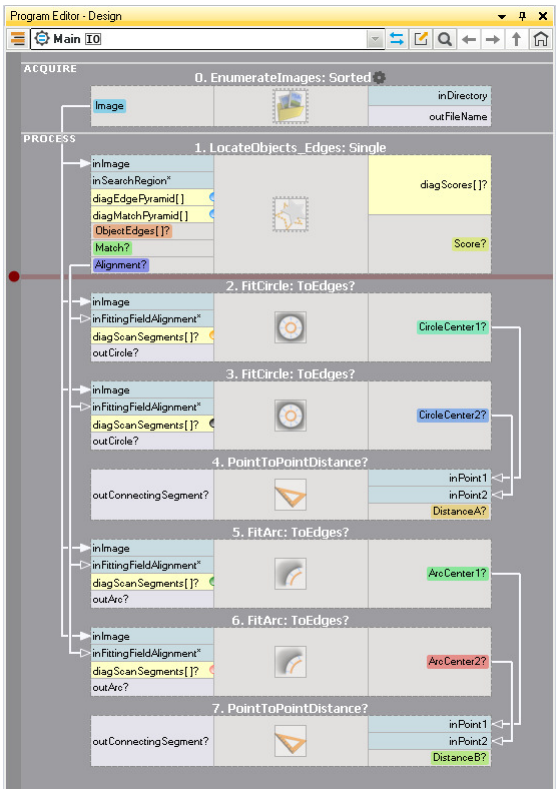
C++

The *Iterate Current Macro* command is very similar to setting a breakpoint at the end of some function in a C++ debugger.

More information about program iteration is available in Testing and Debugging.

Execution Breakpoints

Pausing the program execution in any algorithm location is a fundamental technique while debugging program in any programming language. In Aurora Vision Studio such a pausing can be done with breakpoints in the program editor. Breakpoints are visualized with red circle in the left margin of the program editor and a line next to it:



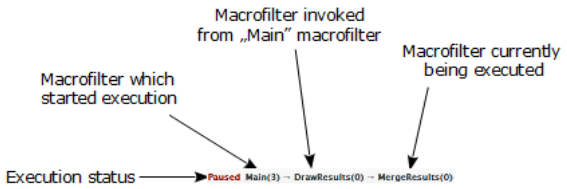
Breakpoints can be activated (toggled) in several ways:

- by clicking appropriate locations within the program editor margin,
- through the context menu of the filter blocks or macrofilter outputs block,
- with F9 shortcut on selected filter block or macrofilter outputs block.

You can read more about Breakpoints in Testing and Debugging.

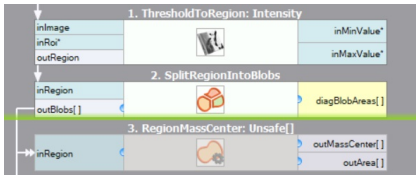
Knowing Where You Are

At each moment of a program execution you can see on the Application Status Bar which macrofilter instance is currently being executed. This is called a *call-stack*, because not only the name of the macrofilter is displayed, but also all the names of the parent macrofilters.

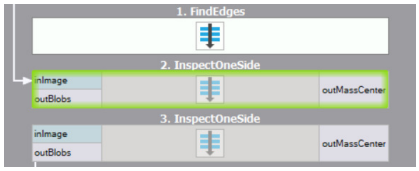


The call-stack of the Application Status Bar

The current position of the execution process is also marked with a green line or frame in the Program Editor:



Execution marker showing the exact position of the execution process.



Execution marker showing that the execution process is currently inside this macrofilter instance.

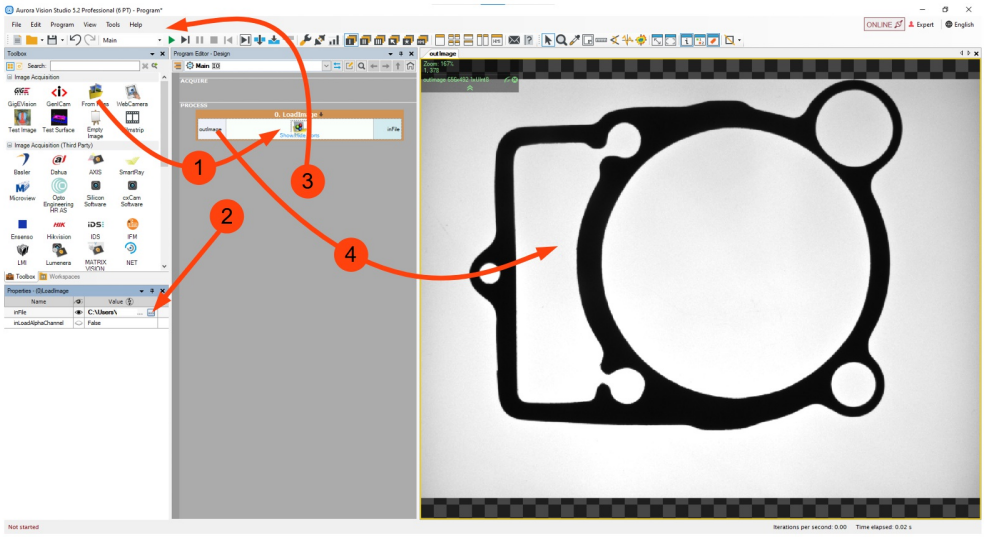
Acquiring Images

Acquiring Images from Files

Aurora Vision Studio is not limited to any fixed number of image sources. Instead, image acquisition is a regular part of the library of filters (tools). In particular, to load an image from a file, the user needs to create a program with an instance of the LoadImage filter.

Four steps are required to get the result ready:

1. Add a LoadImage filter to the Program Editor:
   a. Either by choosing it from the *Image Acquisition* section of the Toolbox (recommended).
   b. Or by dragging and dropping it from the Image :: Image IO category of the Filter Catalog.
2. Select the new filter instance and click on "..." by the **inFile** port in the Properties window. Then select a PNG, JPG, BMP or TIFF file.
3. Run the program.
4. Drag and drop the **outImage** port to the Data Previews panel.



Creating a program that loads an image from a file.

Enumerating Images from a Directory

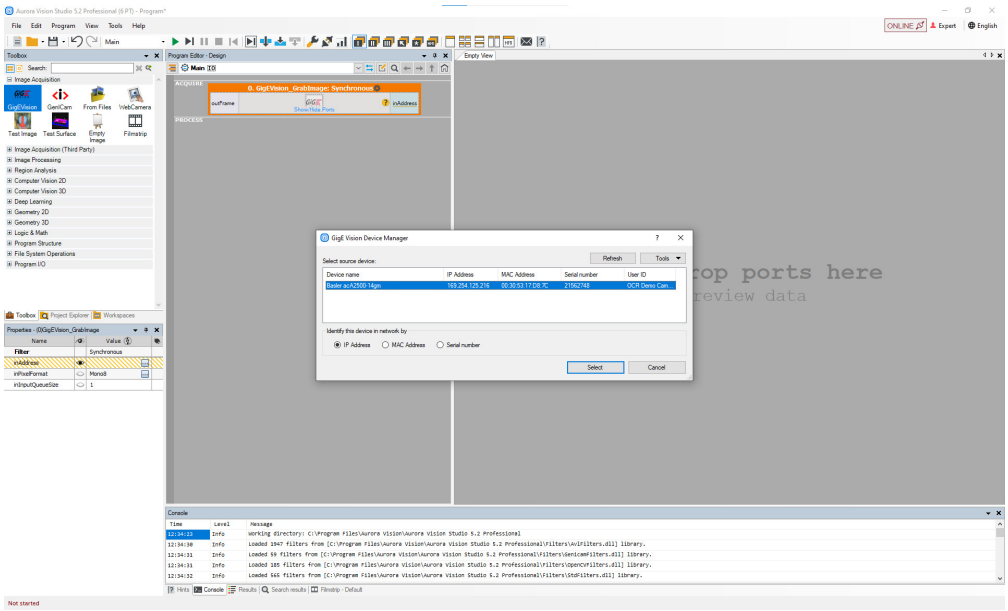
If you have multiple images, e.g. recorded from a camera, and you want to simulate that camera with the files, you can use the EnumerateImages filter (tool). It will make your program work in a loop until all images are read from the directory specified with the **inDirectory** parameter.

Acquiring Images from Cameras

For acquiring images from cameras and frame grabbers there are filters (tools) in several different categories:

- **GigE Vision**
  Provides an interface to all GigE Vision compatible devices. See also Working with GigE Vision Devices.
- **GenICam**
  Provides an interface to all GenICam / GenTL compatible devices. See also Working with GenTL Devices.
- **Camera Support**
  Provides multiple interfaces to drivers provided by individual camera vendors: NET (SynView, ICube), Allied Vision Technologies (Vimba), Basler (Pylon), Kinect, Matrix Vision (mvGenTLAcquire), LMI (Gocator), IFM, PointGrey (FlyCapture, Spinnaker), The Imaging Source (Imaging Control), XIMEA (m3api).
- **Camera Support :: Web Camera**
  Provides an interface to DirectShow based cameras, e.g. to standard web cameras.
- ***User Filters***
  Non-standard cameras can be connected by writing User Filters in C++. A sample implementation for cameras from IDS is available.

It is advisable to use the general GigE Vision or GenICam interfaces in the first place as they provide the most complete feature set and the most comprehensive support in the graphical interface of Aurora Vision Studio. Vendor specific interfaces are required for older and non-standard camera models.



Connecting to a GigE Vision compatible camera.

For more details on the topic, please refer to the video tutorial on our YouTube channel:

Preview and Data Presenting

Creating and Closing Data Previews

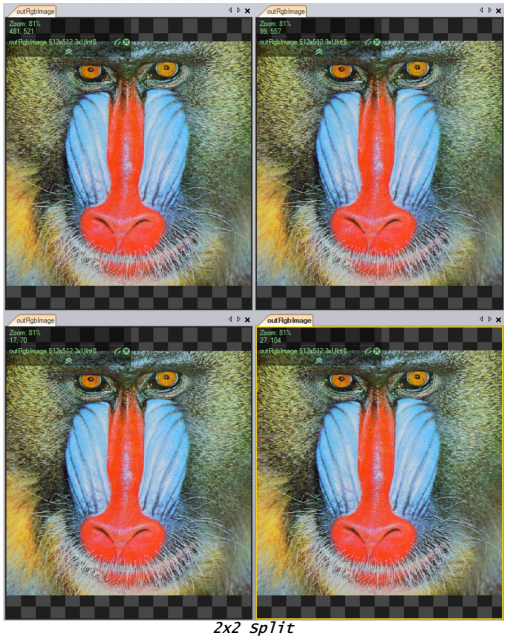
Once you execute filters (tools) in the program it is possible to display their inputs and outputs on the *Data Preview* panel. To do so, you can drag the input or the output of the filter and drop it on the panel to the right of the *Program Editor*. Similar types of data such as **Image** and **Region** or **StringArray** and **IntegerArray** can be displayed on the same preview.

There are several ways of removing elements from the data preview. You can:

- Click the *X* at the top right corner. In this way, you will remove the last data that has been added to the preview. If the closed one is also the last, the preview window is closed.
- Use the mouse wheel button and press it when mouse cursor is placed on the name of the preview (top left corner).
- Right-click on the name of the preview and select one of the options, which are *Remove Preview*, *Close*, *Close All But This* and *Close All*.

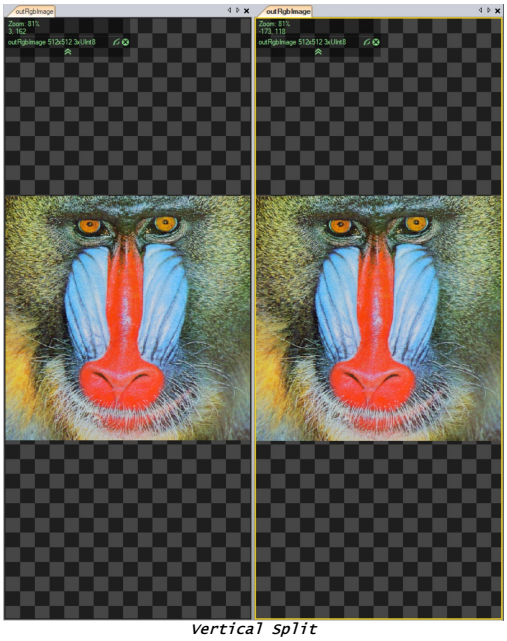
Arranging Data Previews

There are five default options that you can use to arrange your preview:  . Despite displaying your preview in a single window, you can use four other options:

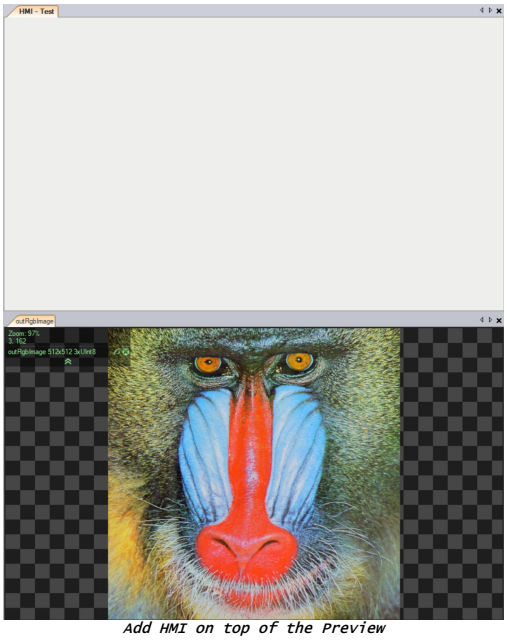


2x2 Split

Horizontal Split



Vertical Split



Add HMI on top of the Preview

Additionally, you can further divide the preview windows by right-clicking on their name and selecting "Split View" option.

Another option that is at your disposal is docking. You can either double-click on the name of the preview or right-click and select *Undock* option. To restore the window to its previous preview simply right-click on the bar and select *Dock* option. Also, you can click and hold a Tab with the name of data preview and

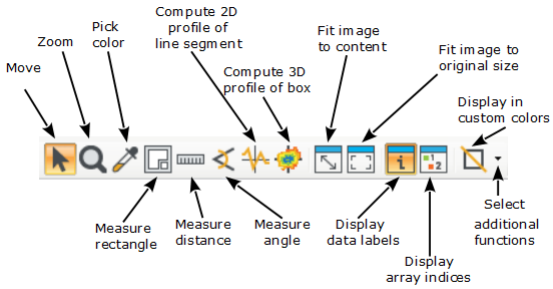
move it to another position. During that process the navigation pane appears and you can specify new appearance of the preview.

Layouts

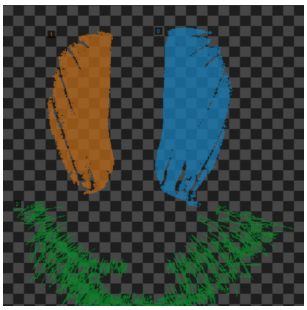
It is possible to have up to three separate preview layouts arranged in different ways and switch between them when working on an algorithm. Additionally, *Auto mode*, which shows the most useful inputs and outputs of the currently active filter (tool), is available. All the above-mentioned features are available on the Toolbar:  . There is also the HMI button that opens the standalone tab where you can design the user interface.

Image Tools

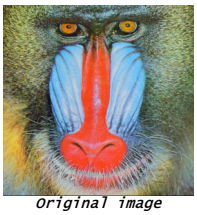
You gain access to the Image Tools after clicking on the image. It contains handy tools that will help you during the analysis of images. Note that they do not modify the image in the program, but rather help you set appropriate parameters and collect data necessary for the inspection, by influencing the preview.



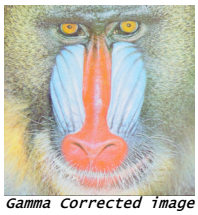
- *Move* and *Zoom* are purely for the navigation on the preview. It is possible to use the same functions with the scroll wheel of your mouse.
- The *Pick Color* tool is handy if you want to check the intensity in the image or RGB/HSV values before performing further operations on the image.
- The next three icons are all about the measurement. *Measure rectangle* allows to select a rectangle and measure its width and height. *Measure distance* returns the distance in pixels between two points on the image. *Measure angle* simply checks the angle defined by three points selected by user.
- Next, there are tools for 1D and 2D image profile analysis. After marking the segment or box on the image and you will receive a graph ready for a quick analysis.
- Following buttons are for fitting the image to the content and returning it to the original size.
- *Display data labels* option allows you to see the labels on the objects and *Display array indices* shows the numbers on the image that correspond to the position of the features in the input array, as seen on the image below:



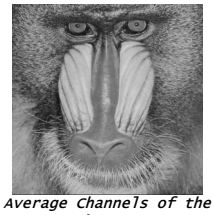
- Possible effects of the *Display in custom colors* tool are displayed below:

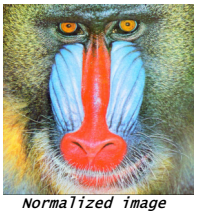

Original image

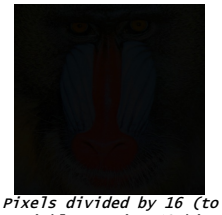

Colorized image


Gamma Corrected image


Negated image


Average Channels of the image


Normalized image


Pixels divided by 16 (to quickly preview 12-bit images)

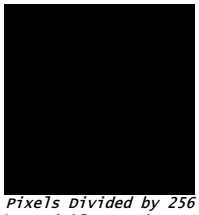
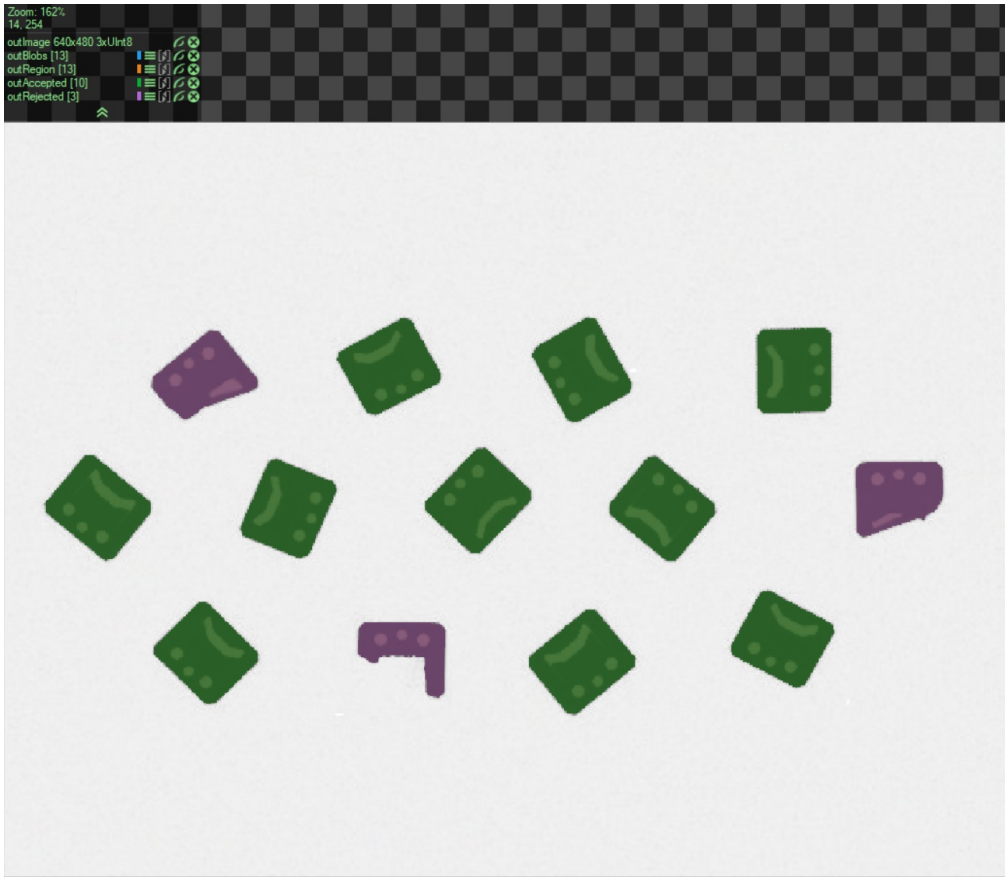

Pixels Divided by 256 (to quickly preview 16-bit images)

Image options on the preview

If you drop an image on the preview you will get access to additional options after right-clicking on it:

- *Save Image As* - saves the image with its original size and other parameters in the specified folder.
- *Save View As* - saves the current view of the image, with other elements (e.g. geometric primitives) located on it, in the specified folder.
- *Copy View As Image* - copies the current view of the image.
- *Zoom to Fit* - scales the image to the size of the preview window so it can fit in it.
- *Zoom to Original size* - returns the image to the original size.
- *Show Information* - turns on/off the information about the current preview.
- *Show Array Indices* - shows sorted numbers on the preview. You can only see the results of that button if it was dropped to the preview as an array.

Preview information

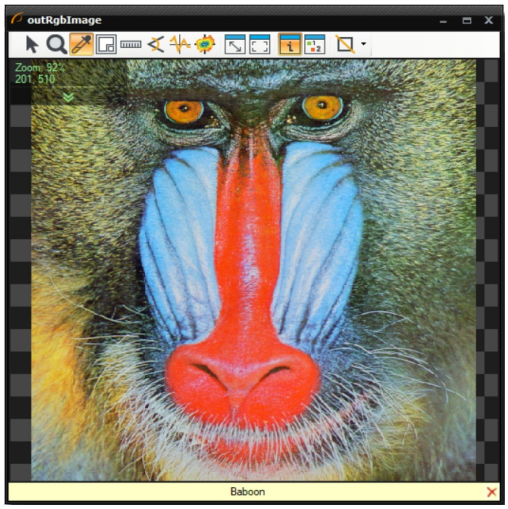


The green text at the top left corner of the preview contains the names of all the outputs and inputs that were dropped on it. They are always followed by additional information. In case of Image or Region it is usually its size. It is also possible to see a number in the brackets - [4]. It indicates the number of the elements of the array of data. To the right of the names you may notice up to five buttons:

-  - it tells you about the color the type of data is being displayed in. If you press it, you will be able to change its shade.
-  - by dragging and moving it up and down you can organize the data on the preview. Note that the color changes with the order of data, so it does not stay the same after being moved.
-  - it allows you to turn on and off the data from the preview without completely removing it.
-  - that button removes the data from the preview.
-  - it helps you navigate through elements of the arrays. If you turn it off, you will be able to see all the elements of the array on the preview at once. Turning it on switches the preview to show only one element and allows to iterate them one by one, however, it makes it impossible to see them all.

Adding comments

Adding a comment is a handful option that you can use on all your preview windows. To do so right-click on the name of the preview and choose *Add Comment* option. After that a yellow box at the bottom of the preview will appear. You will be able to write anything of your liking there, as in the image below:



Useful Settings

To change the quality of the previews you can enter *Tools>>Settings>>Previews>>Preview Quality* or *Program>>Preview Quality*. You will be able to choose from three different settings:

- Fast
- Balanced
- High Quality

Note that in case of larger images *Image is too big!* warning pops out. It automatically lowers down the quality of the image even though the best one was selected beforehand.

Users may extract macrofilters in their program. By default that operation removes the previews of the filters that were enclosed. To prevent that you can access *Tools>>Settings>>Editor>>Preserve port previews when extracting macrofilter*.

For more details on the topic, please refer to a video tutorial on our YouTube channel:

Please note that previews of the 3D data are explained in a separate article: Working With 3D Data.

Known Issues

The SciTextRenderer exception

It might happen that during opening a project Aurora Vision Studio returns the error message that SciTextRenderer threw an exception.

If the issue happens, try the following steps:

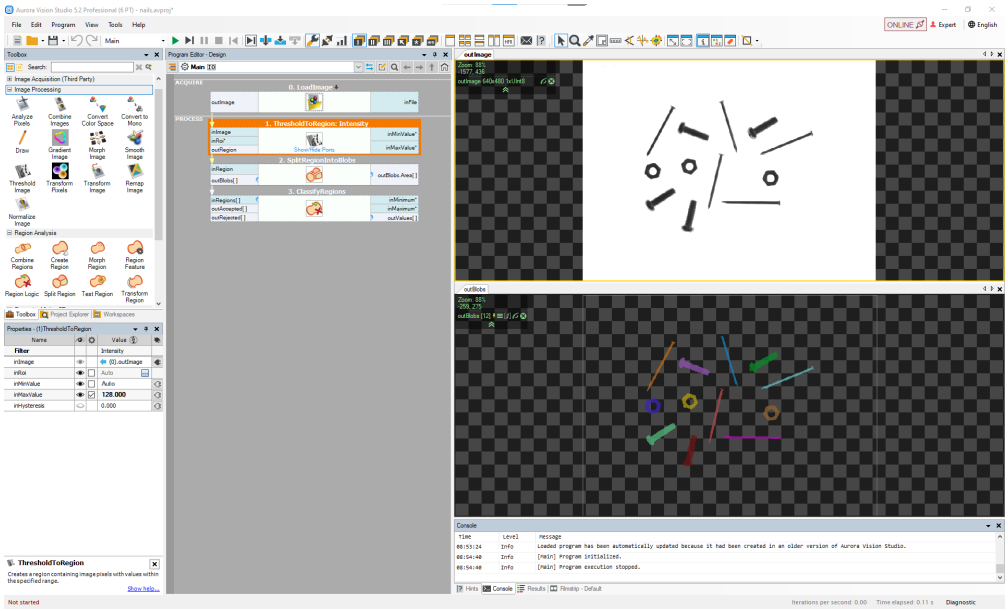
1. Open the project with Administrator privileges.
2. Remove the directory at `%localappdata%\Temp\Scintilla.NET`.
3. Reinstall the **Microsoft Visual C++ Redistributable Package**.

First Program: Simple Blob Analysis

This article demonstrates the basic workflow in Aurora Vision Studio with an example of simple blob analysis. The task here is to separate nails from other objects which are present in the input image.

Extracting Blobs

To start this simple demonstration we load an image from a file – with the LoadImage filter (tool), which is available in the *Image Acquisition* section of the Toolbox, the *From File* group. The image used in the example has been acquired with a backlight, so it is easy to separate its foreground from the background simply with the ThresholdToRegion filter (*Image Processing → Threshold Image*). The result of this filter is a Region, i.e. a compressed binary image or a set of pixel locations that correspond to the foreground objects. The next step is to transform this single region into an array (a list) of regions depicting individual objects. This is done with the SplitRegionIntoBlobs filter (*Region Analysis → Split Region*):



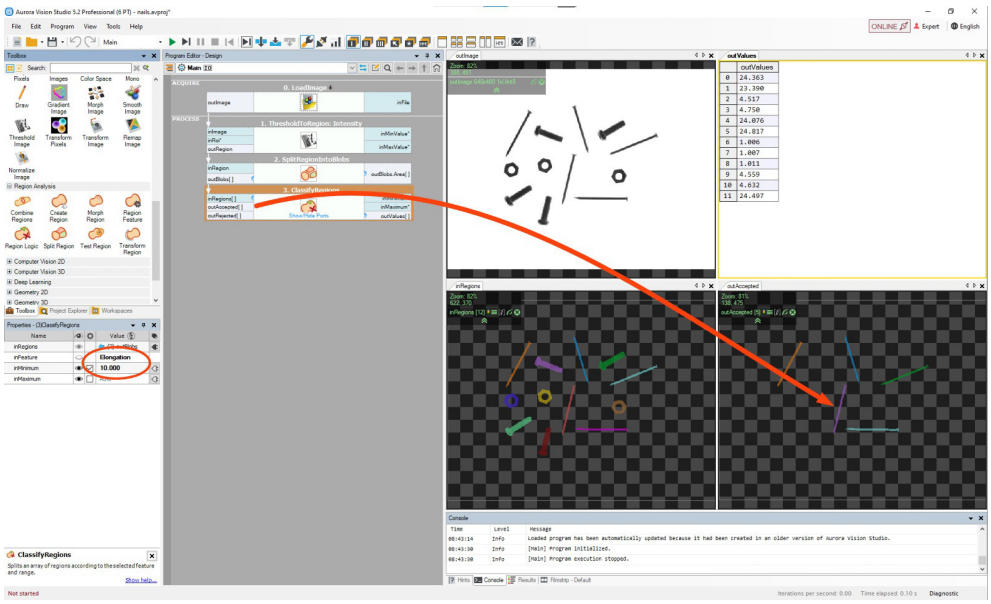
Extracting blobs from an image.

Notes:

- Connections between filters are created by dragging with a mouse from a filter output to an input of another filter.
- The data previews on the right are created by dragging and dropping filter outputs.
- The input file is available here: parts.png.

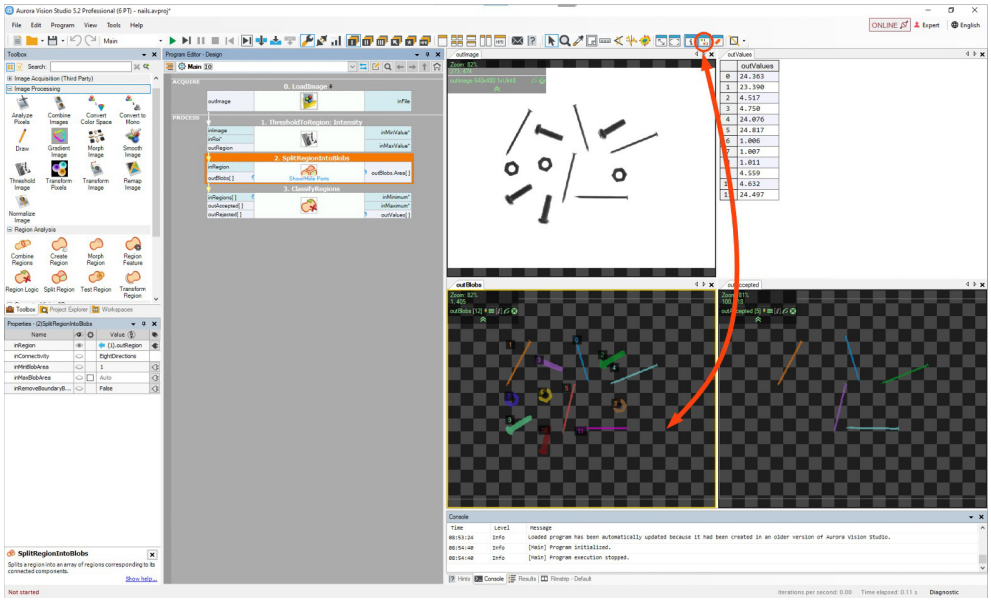
Classifying the Blobs

At this stage what we have is an array of regions. This array has 12 elements, of which 4 are nails. To separate the nails from other objects, we can use the fact that they are longer and thinner. The ClassifyRegions filter (*Region Analysis → Region Logic*) with **inFeature** input set to *Elongation* and **inMinimum** set to *10* will return only the nails on its **outAccepted** output:



Classifying blobs by the "elongation" feature.

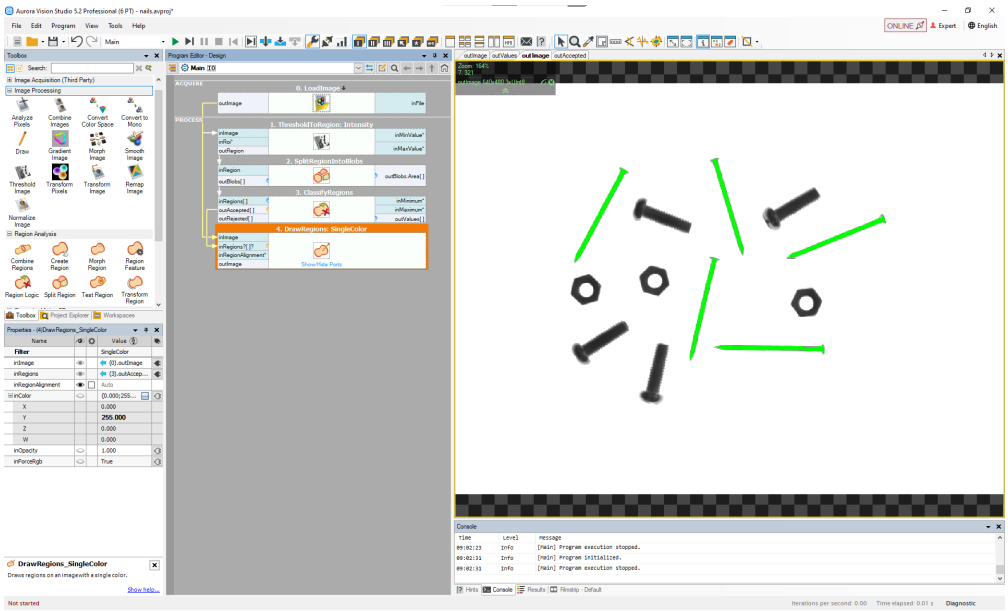
There is also the **outValues** output which contains the feature values of the individual blobs. This can also be displayed in the Data Previews as a table of real numbers. The indexes in this table correspond to the blobs, which can be shown by using the "Show Indexes of Elements" option in the selected data preview toolbar:



Showing indexes of individual blobs.

Drawing the Results

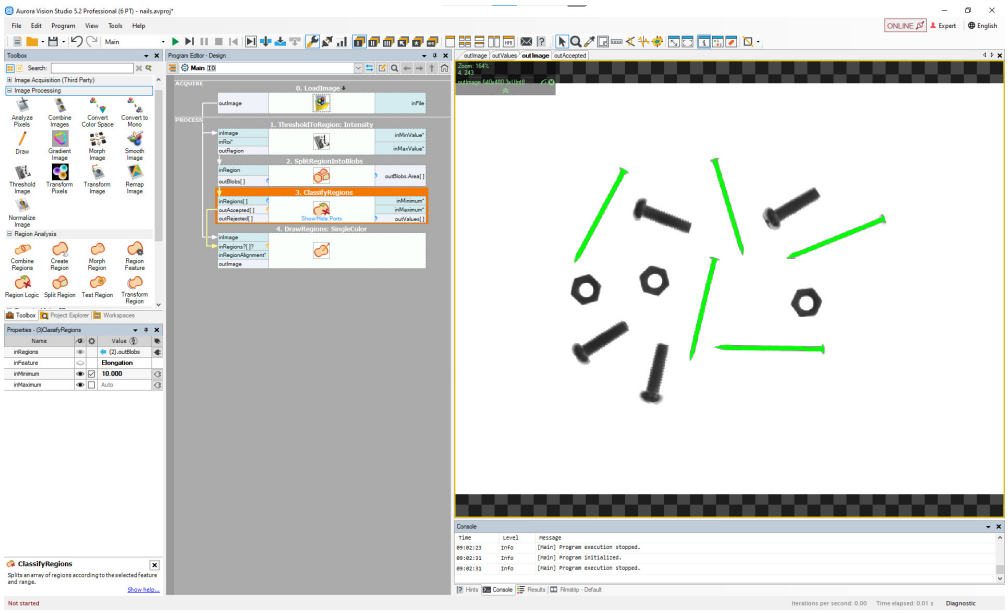
Finally, we can create an output image (e.g. for displaying in the HMI) with the nails marked in green. For this purpose we use the DrawRegions_SingleColor filter, which needs to have its **inImage** and **inRegions** inputs appropriately connected. The **inColor** input defines the required color and can be edited through the Properties window.



Drawing the nails in green.

Getting the Number of Elements

If we want to obtain also the number of elements found, we can right-click on the **outAccepted** output and select *Property Outputs → Count*, which creates an additional output named **outAccepted.Count**. In this case we are getting the number *5*:



Counting the found objects.

Conclusion

As this example demonstrates, creating programs in Aurora Vision Studio consists of selecting filters (tools) from the Toolbox (or from the Filter Catalog), connecting them and configuring. Data previews are created by dragging and dropping filter ports to the data preview area. This simple workflow is common for the most basic programs, as the one just shown, as well as for highly advanced industrial applications which can contain multiple image sources, hundreds of filters and arbitrarily complex data-flow logic.

Note: This program is available as "Nails Screws and Nuts" in the official examples of Aurora Vision Studio.

Zebra
Aurora™ Vision

This article is valid for version 5.6.1