Adaptive Vision

Intuitive

Powerful

Adaptable

Future Processing Sp. z o. o.
ul. Bojkowska 37a, 44-100 Gliwice, Polska
Regon: 278263445 VatID: PL6342532128
Tel. +48 32 461 23 30
E-mail: info@adaptive-vision.com

Adaptive Vision Studio 4.10

Introduction

Created: 24.08.2018

Product version: 4.10.2.62669

Table of content:

Installation

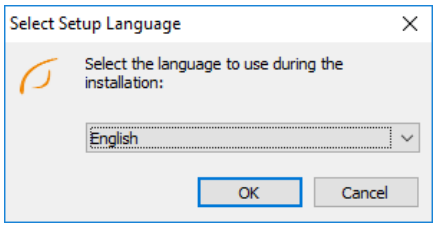
Installation Procedure

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

The installation procedure consists of several straightforward steps:

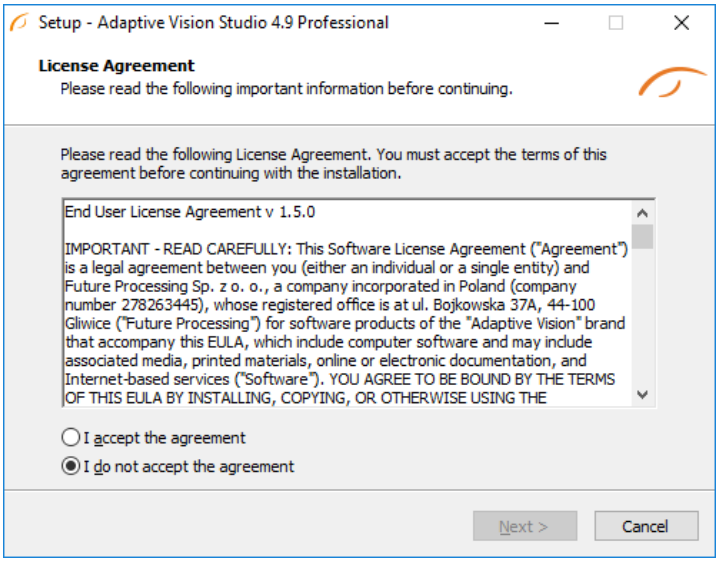
Setup language

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



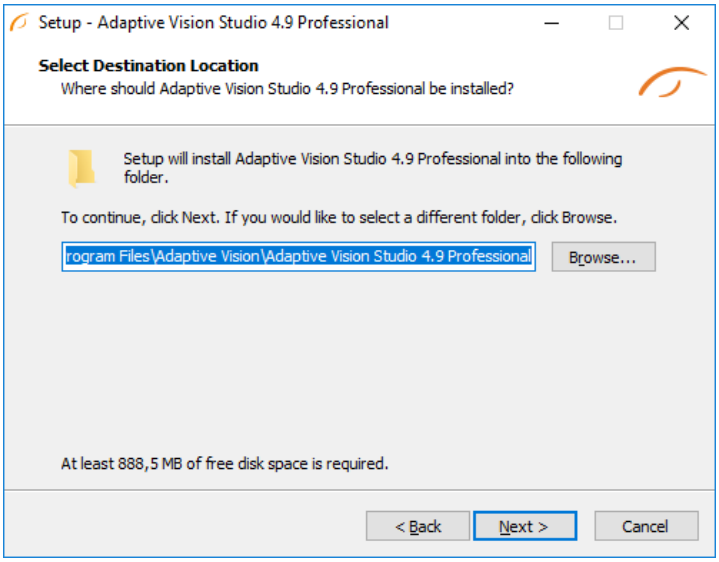
License agreement

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



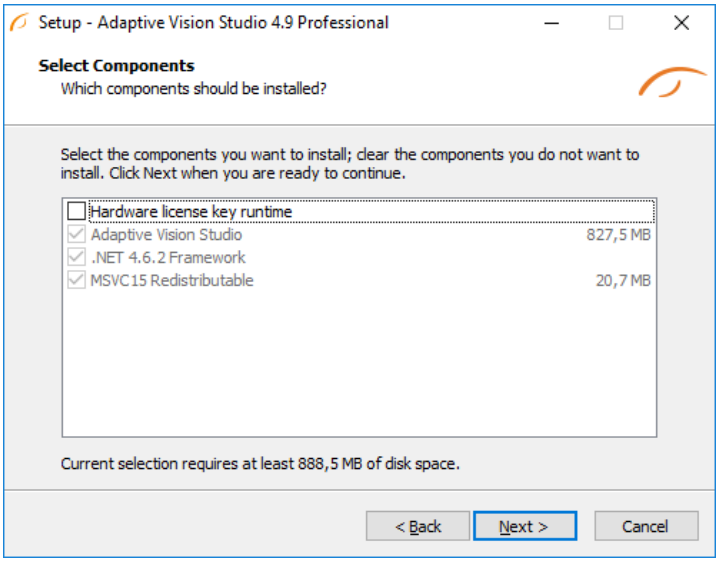
Localization on disk

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



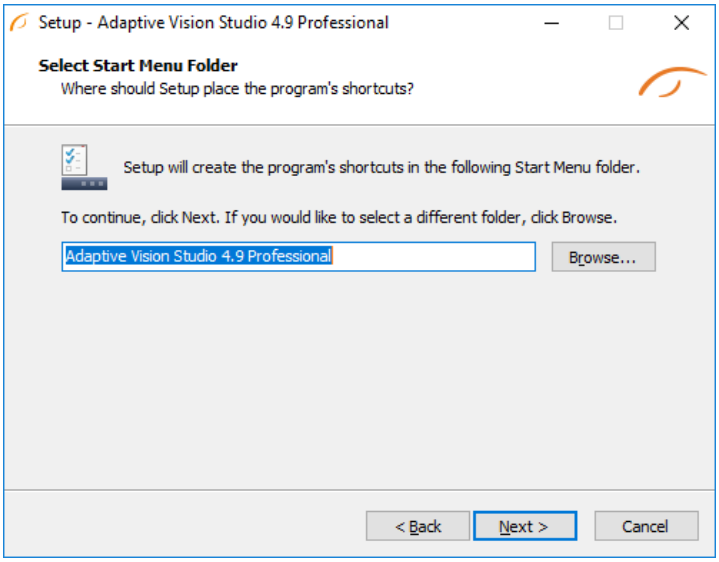
Components

Choose which additional components you wish to install.



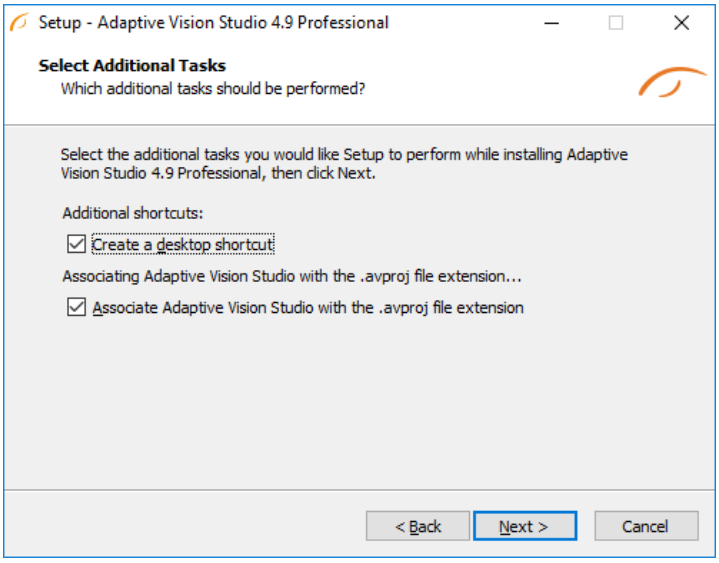
Start Menu shortcut

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



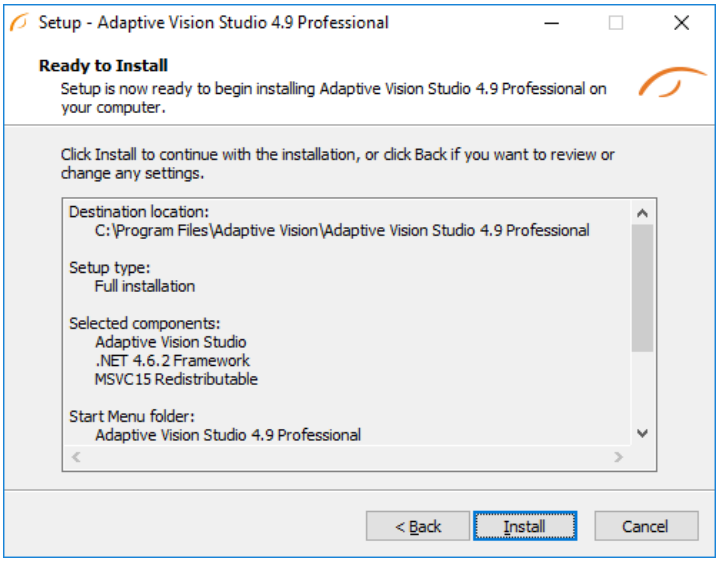
Additional Options

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

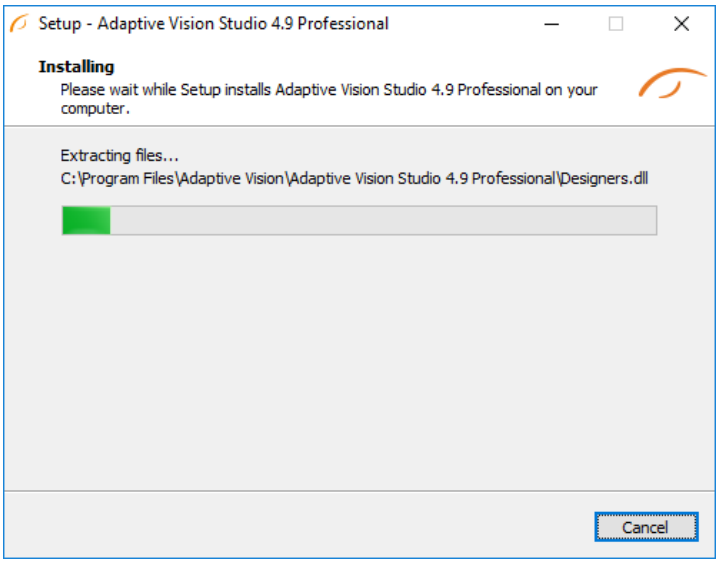


Installing

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

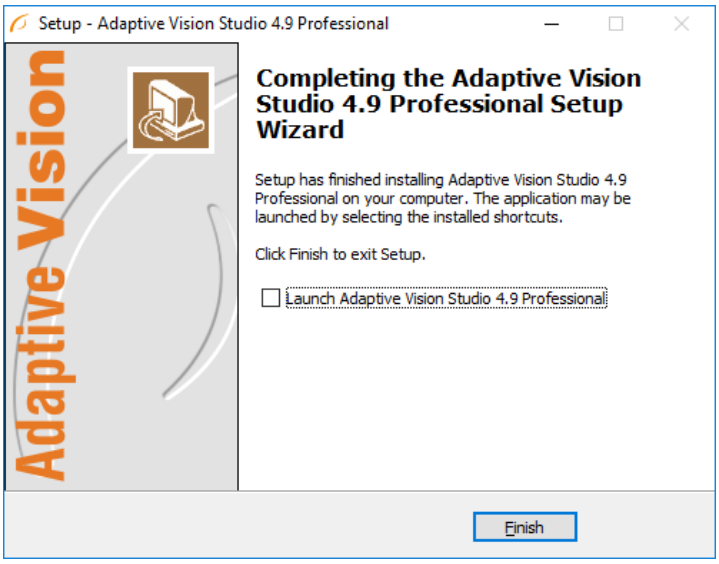


Please wait until all files are copied.



Final Options

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



Additional dependencies

Users, who have custom security policies applied to their accounts, may encounter problems after installation of Adaptive Vision Studio. In that case some additional dependencies must be installed manually:

- **Microsoft Visual C++ 2015 Redistributable Package:**
  - **Combined Installer for 32bit Systems (x86) and 64bit Systems (x64). Allows to choose only one version to install, if needed.**
- **Microsoft .NET Framework 4.6.2** (for legacy systems like Windows Vista)
- **CodeMeter User Runtime for Windows** (only when hardware license key will be used).

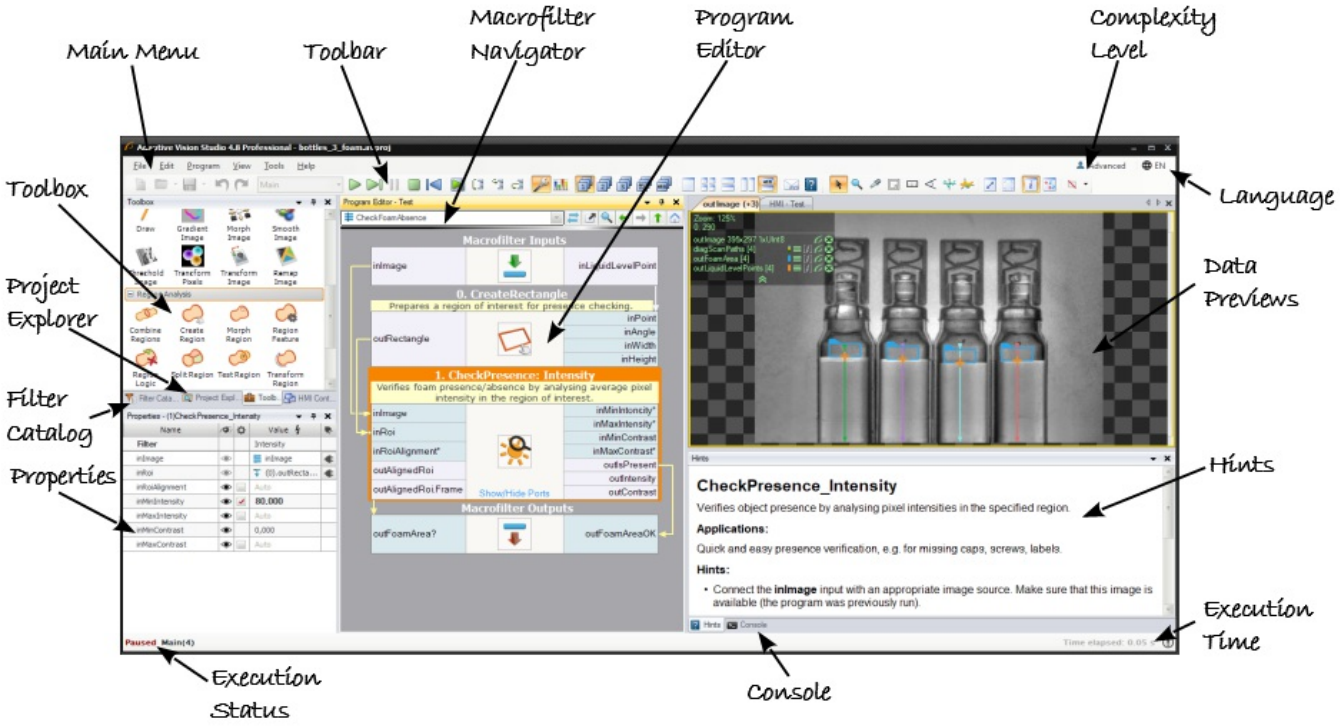
Deinstallation procedure

To remove Adaptive Vision Studio from your computer please launch the program at *Start » All Programs » Adaptive Vision » Adaptive Vision Studio Professional » Uninstall* and follow the on-screen instructions.

Main Window Overview

Elements of the User Interface

The user interface of Adaptive Vision Studio has been carefully designed for good user experience. All the main elements of the application are available on a single screen which is depicted below. The standard layout of windows can be changed in an arbitrary way – use this feature to adapt the application to your specific preferences. In particular, it is advisable to work with two HD monitors and use the second monitor for undocked data previews or for the HMI window.



Elements of the User Interface.

The purpose of the individual controls is as follows:

- **Main Menu**
  This is a standard application menu that contains all major actions and options.
- **Toolbar**
  Toolbar provides quick access to the most common actions.
- **Macrofilter Navigator**
  Displays the tree of macrofilter instances contained in the currently selected program.
- **Program Editor**
  Program Editor is a place, where filters are put and connected to create programs. To insert a filter there just drag and drop one from the Toolbox or double-click it.
- **Data Previews**
  This is a place where data computed by filters are displayed. Just drag and drop filter outputs or inputs here. Additional options can be found on toolbar.
- **Hints**
  Shows additional help for selected filter with additional hints for better understanding of how the filter works and possible solutions to common problems.
- **Console**
  Console informs the user about events related to project edition and execution. It is particularly important for reporting and fixing errors.
- **Execution Time**
  Displays the total execution time of the current execution process.
- **Toolbox**
  This is a simplified, task-oriented catalog of filters used in typical machine vision applications.
- **Filter Catalog** (Advanced)
  This is the complete, library-oriented catalog of all available filters, also featuring a text-based search engine.
- **Project Explorer**
  Displays a list of modules, macrofilters, global parameters and attachments contained in the currently edited project. Please note, that you see a list of definitions of macrofilters here, not their instances. One macrofilter (design) can have multiple instances (uses) and the individual instances can be browsed in the Macrofilter Navigator window. See also: Browsing Macrofilters
- **Properties**
  The Properties window makes it possible to set parameters of filters and HMI controls.
- **Execution Status**
  Execution Status informs the user whether the program is running, paused or stopped. During execution it also displays the current program location (the call-stack).

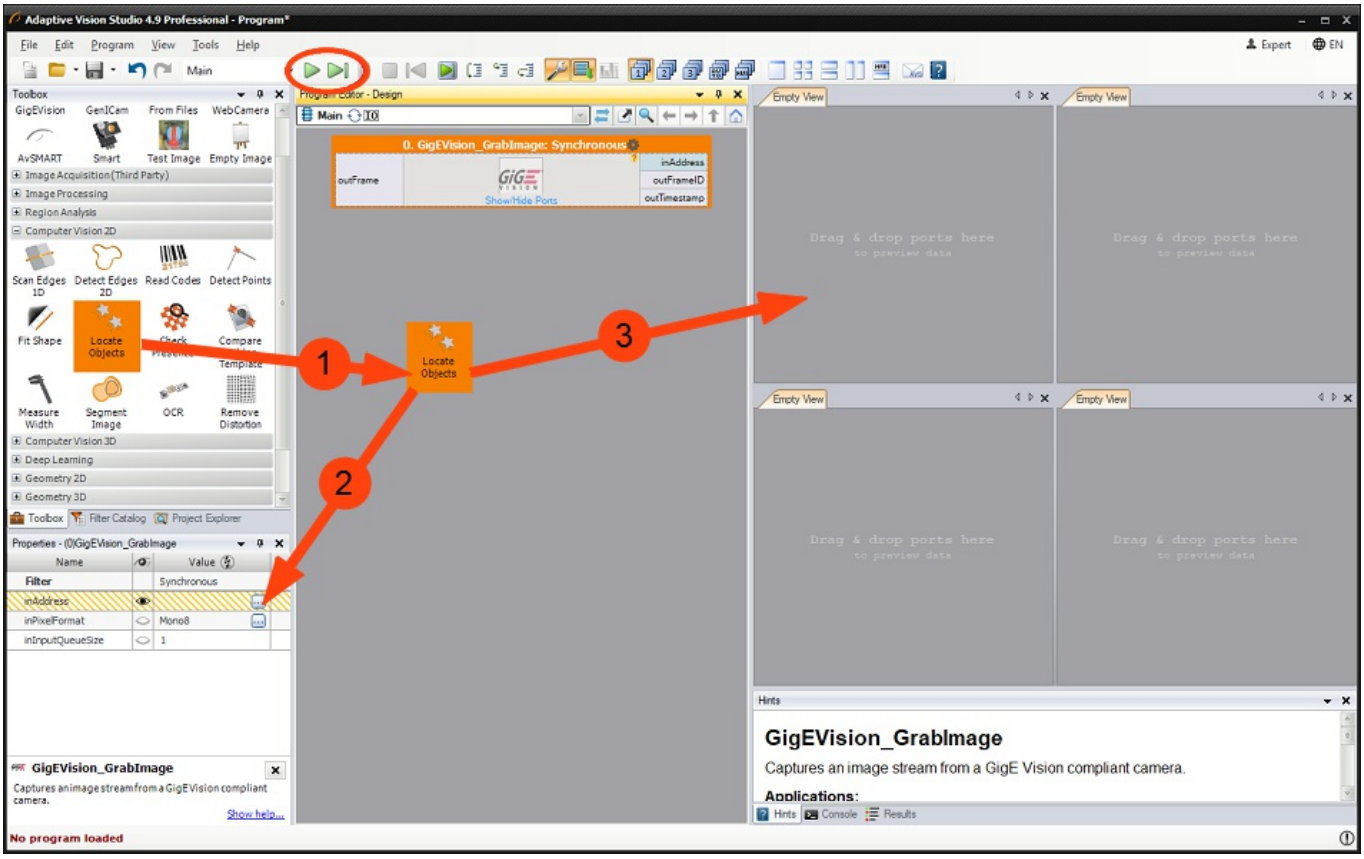
- **Complexity Level**
  This option controls the level of features complexity. See also: Complexity Levels

The Basic Workflow

Creating vision algorithms in Adaptive Vision Studio consists in repeating three intuitive steps:

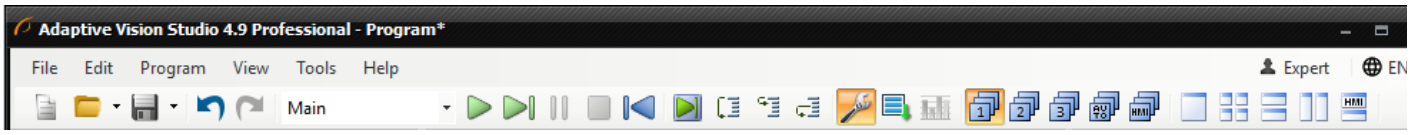
1. Drag & drop (or double-click) filters from the Toolbox (or Filter Catalog) to the Program Editor.
2. Drag & drop connections between the filters or set constant input values in the Properties window.
3. Drag & drop filter outputs to the Data Previews panels.

Whenever a part of a program is ready, click *Run* or *Iterate* buttons to test it. The Console window at the bottom will also have some important information about the execution. Check it especially when something goes wrong.



The workflow of Adaptive 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 *(Ctrl+N)*
Opens a new project.

Open *(Ctrl+O)*
Opens an existing project.

Open Example
Opens an existing example project.

Open Tutorial
Opens an existing tutorial project.

Save *(Ctrl+S)*
Saves the current project.

Save As
Saves the current project in a specified location.

Connect to Remote Executor
Opens window that provides connection with remote Adaptive Vision Executor, allowing application deployment of the current project i.e. on the smart camera and remote control of Adaptive Vision Executor. 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 Adaptive Vision Executor.

Generate C++ Code
Creates a C++ program for the currently opened Adaptive 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 such managed languages as Visual C#, Visual C++/CLI, etc. See .NET Macrofilter Interface Generator for more information.

Edit

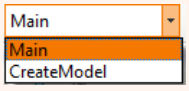
Undo *(Ctrl+Z)*
Reverts the last performed operation.

Redo *(Ctrl+Y)*
Re-performs the operation reverted with Undo command.

Rename Current Macrofilter *(F2)*
Renames current macrofilter.

Remove HMI
Unbinds HMI from the current project and clears all the HMI controls.

Program



Startup Program (combo box)
Selects a macrofilter from which the execution will start.

Run *(F5)*
Executes the program until all iterations are finished or until the user presses *Pause* or *Stop* buttons.

Iterate *(F6)*
Executes the program to the end of a single iteration. See also Execution Process for more information about iterations.

Pause *(Ctrl+Alt+Pause)*
Suspends the current program execution immediately after the last invoked filter is finished.

Stop *(Shift+F5)*
Stops the program immediately after the last invoked filter is finished.

Iterate Back *(Shift+F6)*
Executes the program to the end of a single iteration, reversing direction of enumeration.

Iterate Current Macrofilter *(Ctrl+F10)*
Executes the current program to the end of the currently selected macrofilter (which acts like a breakpoint location).

Step Over *(F10)*
Executes next single instance of a filter or a macrofilter, without entering into the macrofilter.

Step Into *(F11)*
Executes next single filter instance. If it is an instance of a macrofilter, it enters into the macrofilter.

Step Out *(Shift+F11)*
Executes all filters till the end of the current macrofilter, and exits to the parent macrofilter.

Run with Adaptive Vision Executor*(Ctrl+F5)*
Executes the program using Adaptive Vision Executor application installed on the local machine.

Program Statistics
Shows information about execution time of each filter in the selected macrofilter.

Execution Settings
Opens settings for program execution.

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.

Hints
Switches the Hints window visibility.

Dock Open Windows
Docks all undocked data preview windows.

Program Display: Compact
Enables compact display mode with some inputs hidden.

Program Display: Full
Enables full display mode with input values preview on the Program 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.

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.

Update External Data File
Allows to manually update *.avdata files referenced by the current project and generated by Adaptive Vision Studio 2.5 and earlier.

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 Adaptive Vision Studio.

Help

View Help *(F1)*
Opens up the documentation of Adaptive Vision Studio.

Message to Support
Sends an e-mail to support with attached Adaptive 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 Adaptive Vision products.

Check for Updates
Checks if there is newer version of Adaptive Vision Studio.

About Adaptive Vision Studio
Displays information about your copy of Adaptive Vision Studio, e.g. the application version, loaded assemblies and plugins.

Application Settings

Adaptive 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
    - Startup
    - Console
    - Previews
2. Program Execution
    - General
    - HMI
3. Filters
    - Library
    - Filter Catalog
    - Toolbox
    - Filter Properties
    - Project Explorer
4. File Associations
    - Default Program
5. Program Edition
    - Macrofilter Navigator
    - Editor
    - Program Analysis
6. Project files
    - Project Explorer
7. Messages
    - Show these messages:

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 when an exception occurred
Automatically breaks program when exception occurred.

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

Break on assertion failed
Automatically breaks program when assertion failed.

Diagnostic mode
Enables or disables computation of diagnostic output values.

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.

Use global rerun mode
If checked, all filters following the selected one will be executed after rerun.

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 Adaptive 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.

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.

Editor

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

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.

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.

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.

6. Project files

Project Explorer

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

7. 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.

Introduction to Data Flow Programming

Important: Adaptive 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, Connections and Macrofilters.

Data

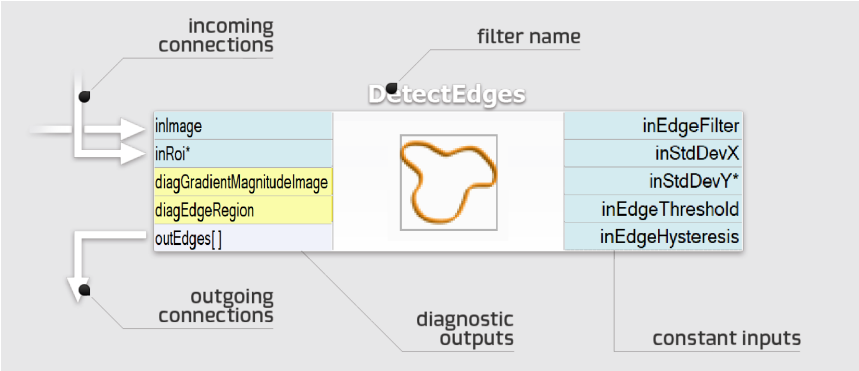
Adaptive Vision Studio is a data processing environment so data is one of its central concepts. The most important fact about data that has to be understood is the distinction between *types* (e.g. **Point2D**) and *values* (e.g. the coordinates *(15.7, 4.1)*). Types define the protocol and guide the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

Adaptive Vision Studio also supports *arrays*, i.e. variable-sized collection 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

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.



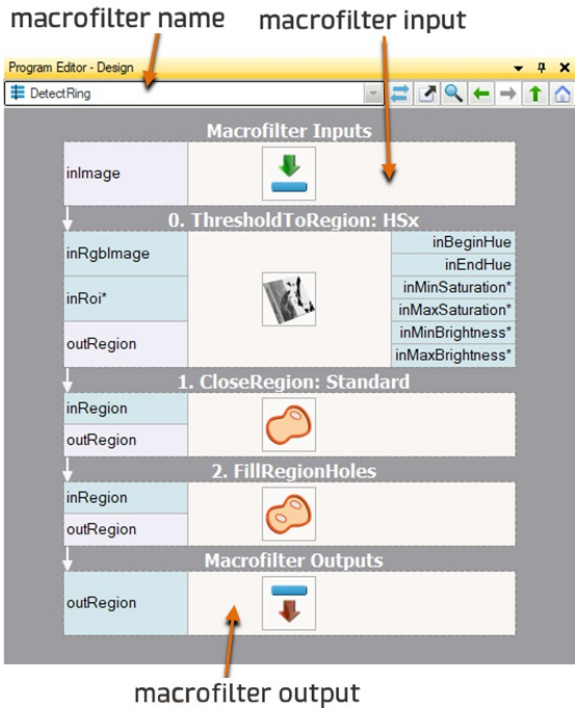
Connections

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 Adaptive 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.

Running and Analysing Programs

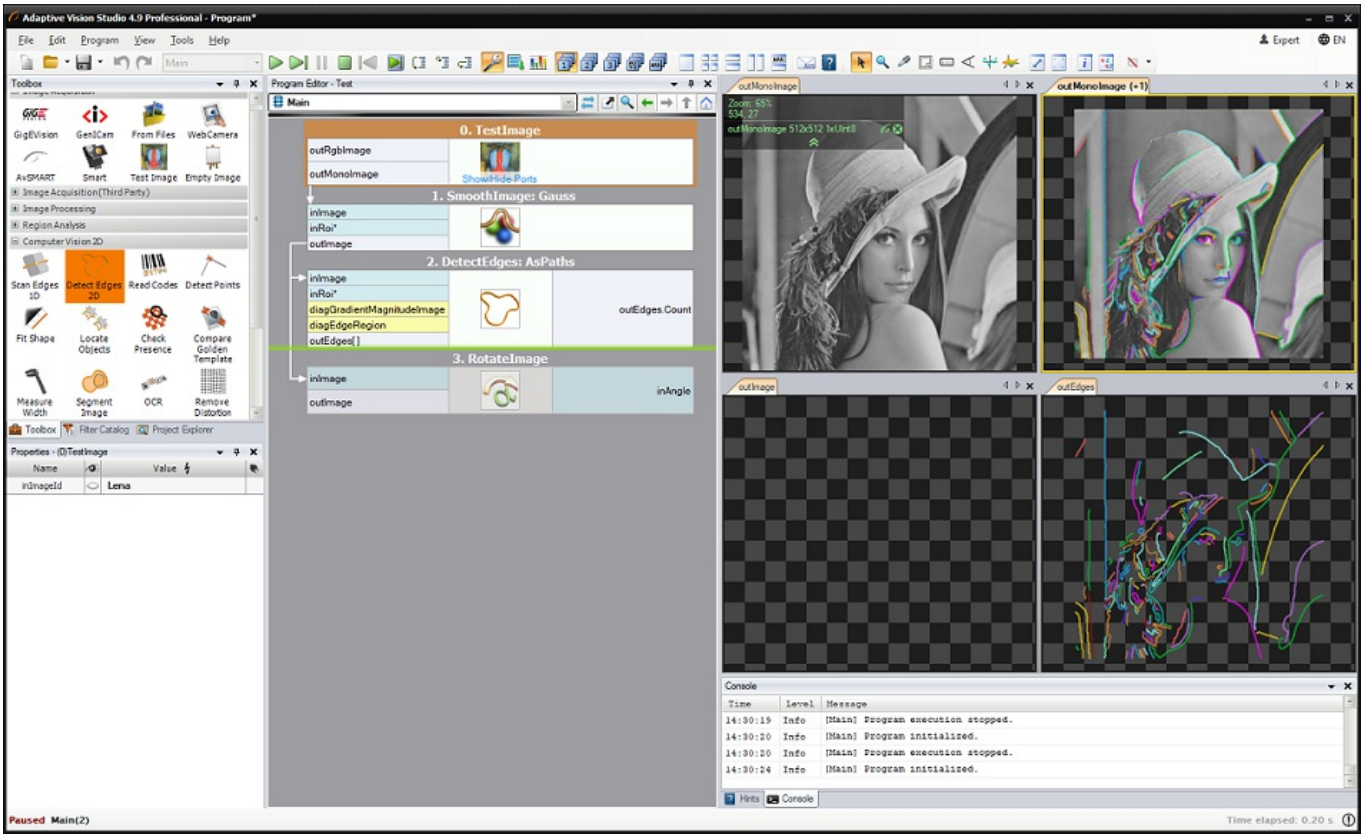
Example Projects

One of the best ways to learn Adaptive 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*.

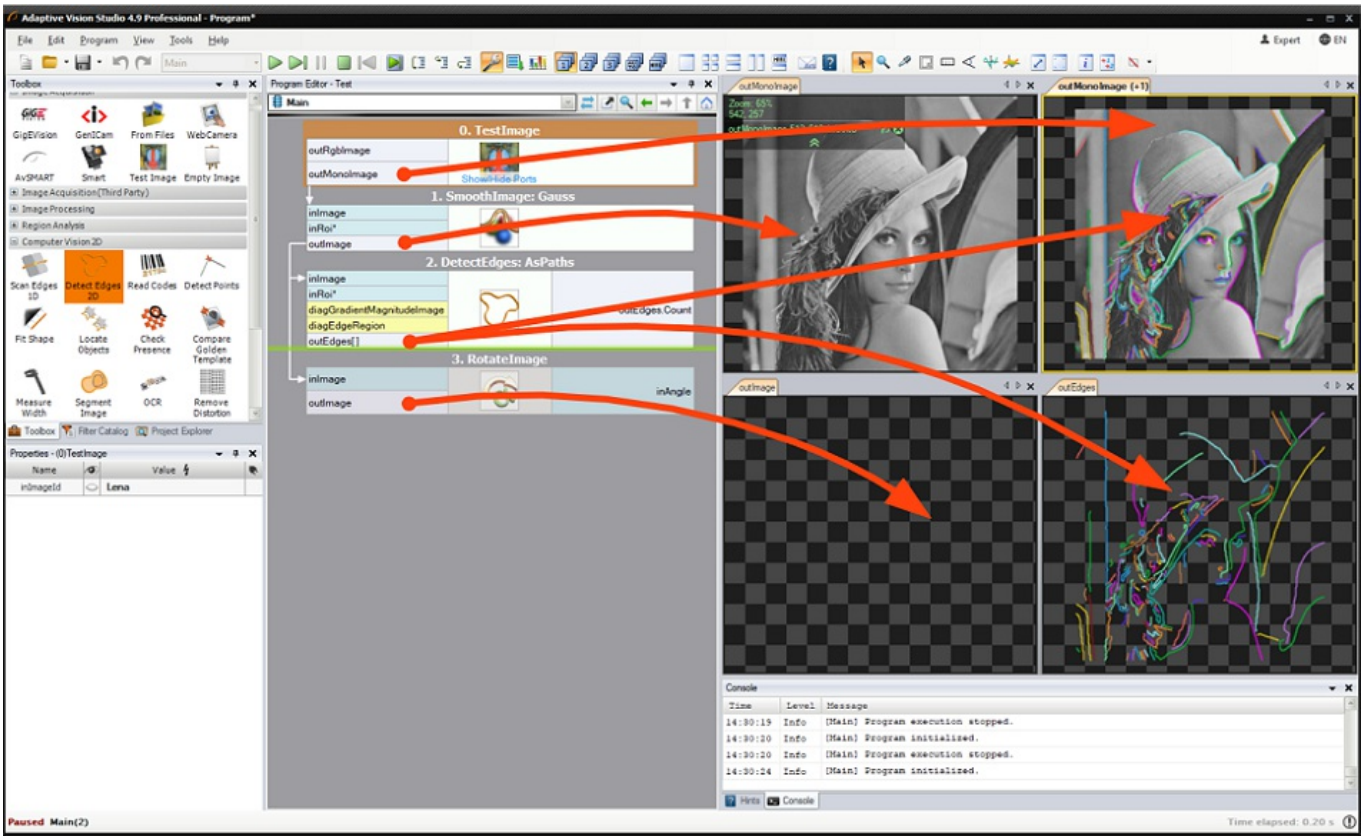
When a project is loaded from a file or when new filters are added, the filter instances are displayed subdued. They become highlighted after they are invoked (executed). It is also possible to invoke individual filters one by one by clicking *Step Over* or *Step Into* , or by pressing *F10* or *F11* 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.

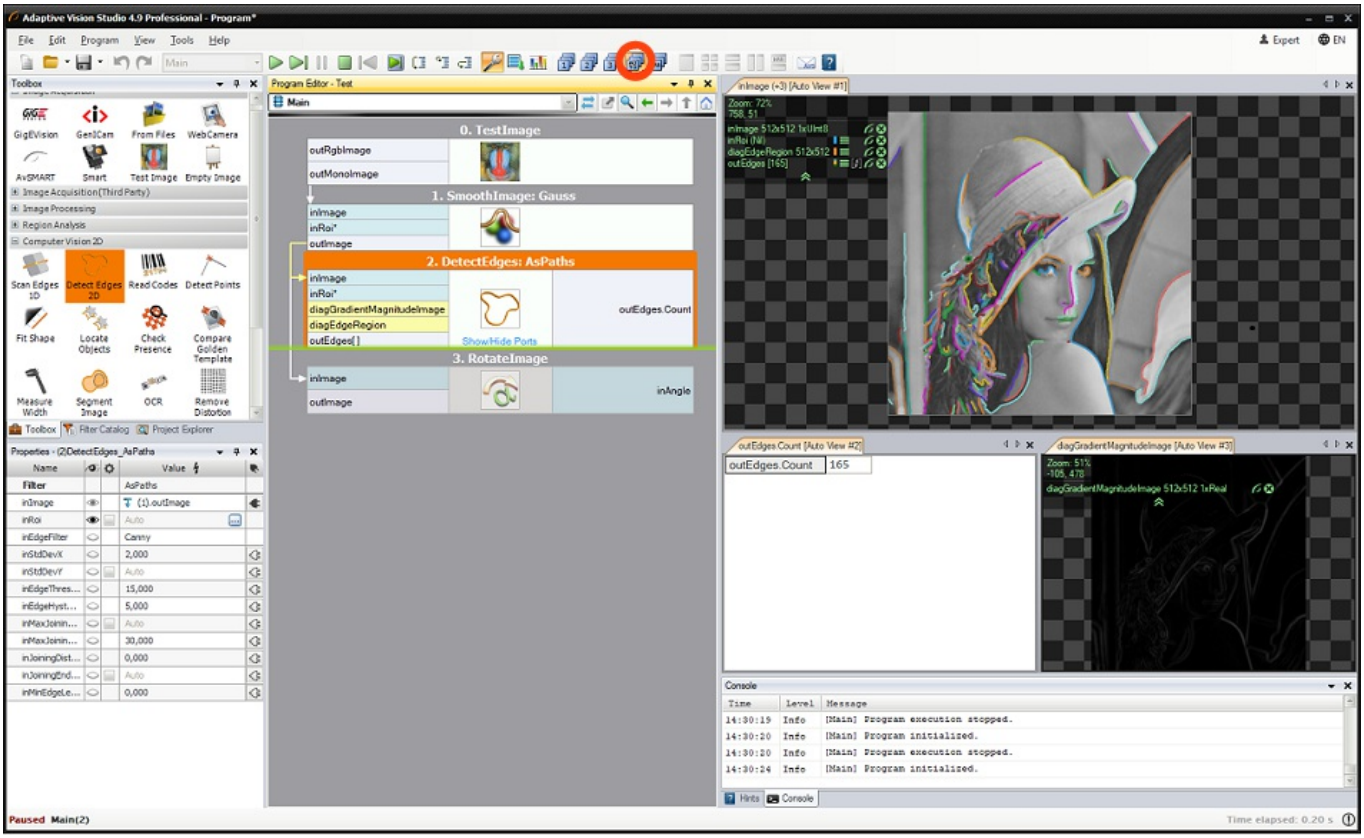
Viewing Results

Once the filters 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. As you can see in the picture below, multiple data can be often displayed in multiple layers of a single preview. This is useful especially for displaying geometrical primitives over images.



User-defined data previews from individual filter outputs.

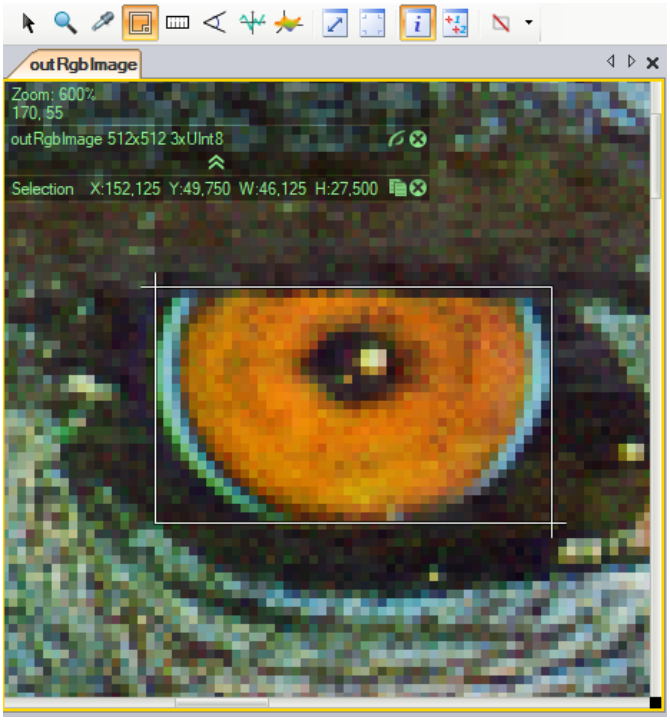
In bigger projects you will also find it useful to switch between three different layouts, which can be created to visualize different stages of the algorithm, as well as to the automatic layout mode, which is very useful for interactive analysis of individual filters:



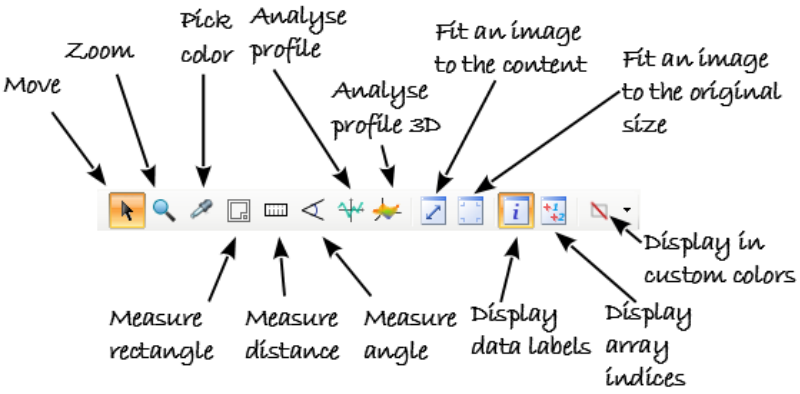
Automatic data previews – the layout adapts to the currently selected filter.

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:



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

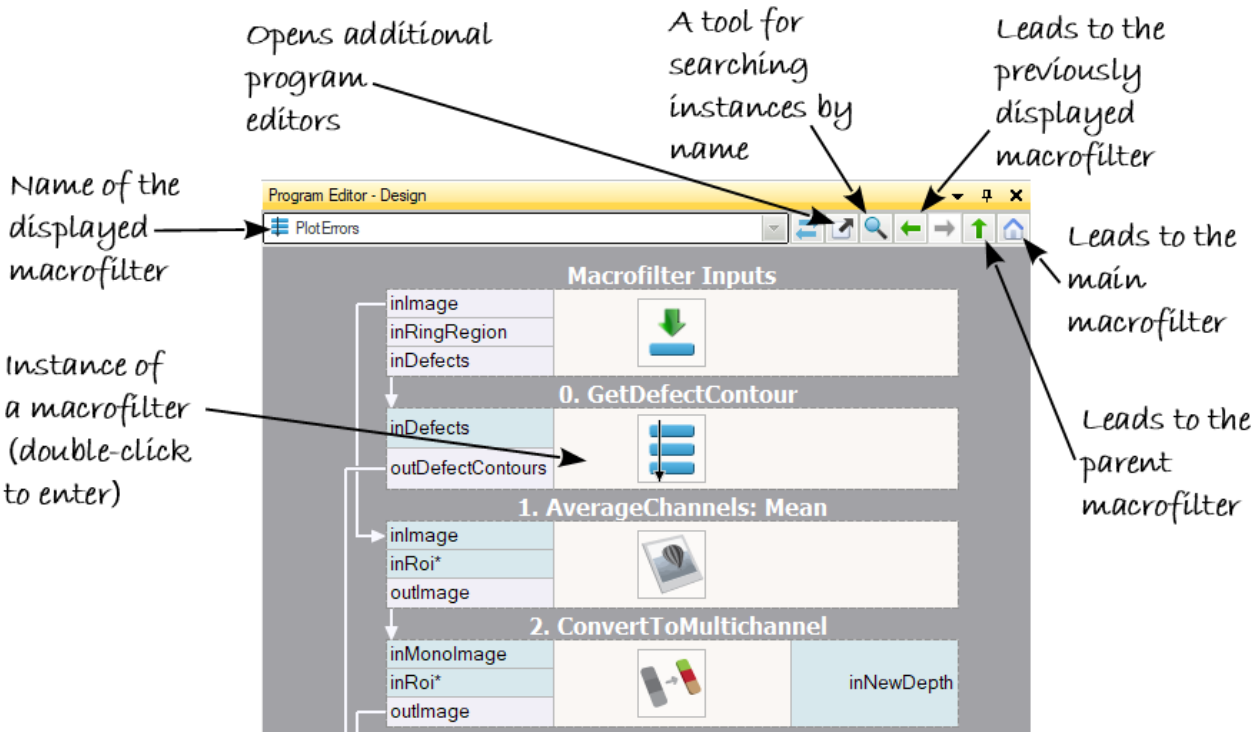


Additionally, there are several usability enhancements:

- **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.

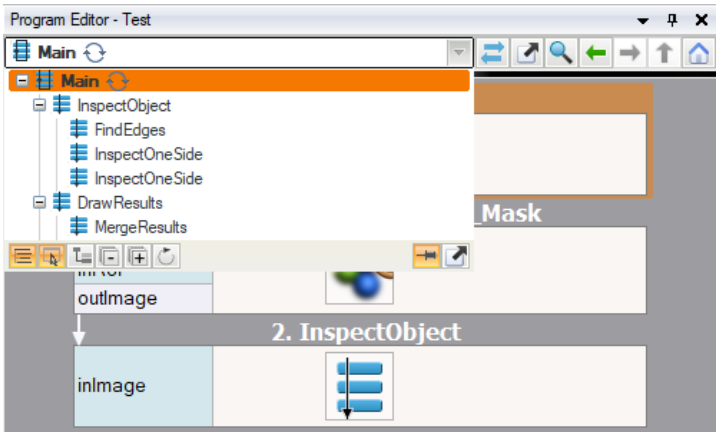
Browsing Macrofilters

Except for the most simple applications, programs in Adaptive 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.

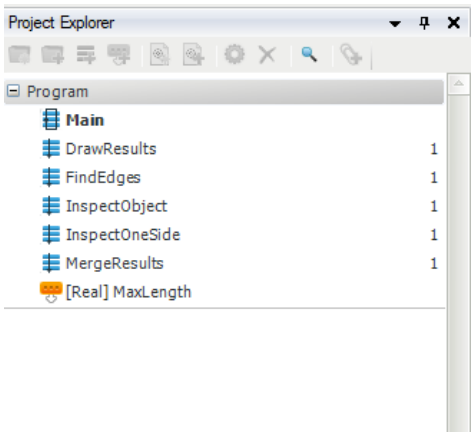


Browsing macrofilters in the Program Editor.

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 *classes* (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.



Classes of macrofilters in the Project Explorer.

c++

Classes of macrofilters correspond to function definitions in C++, whereas instances of macrofilters correspond to function calls on a call-stack. Unlike the C++, there is no recurrence in Adaptive Vision Studio and each macrofilter class 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

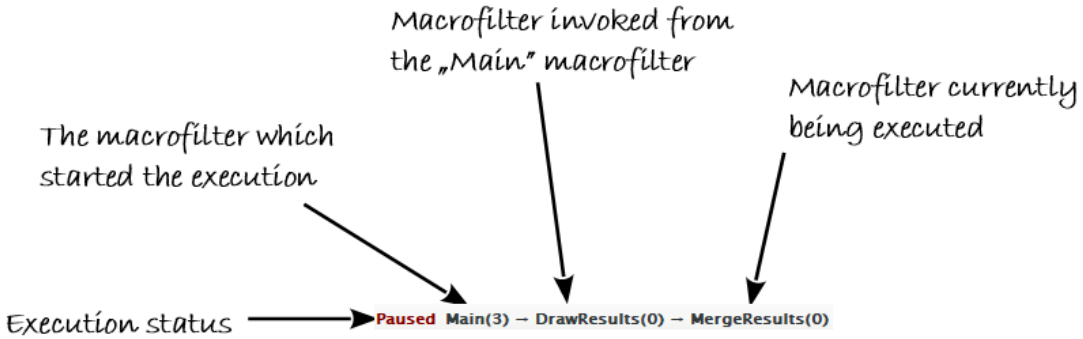
The *Iterate Current Macro* command can be very useful when you want to focus on a single macrofilter in a program with many macrofilters. It executes the whole program and pauses each time when it comes to the end of the currently selected macrofilter instance.

c++

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

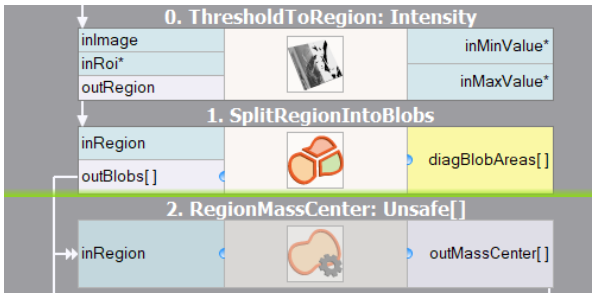
Knowing Where You Are

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.

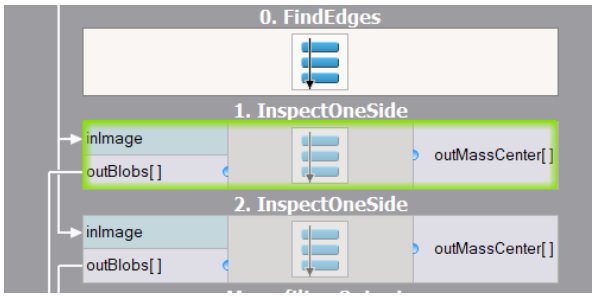


The call-stack of the Application Status Bar

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.

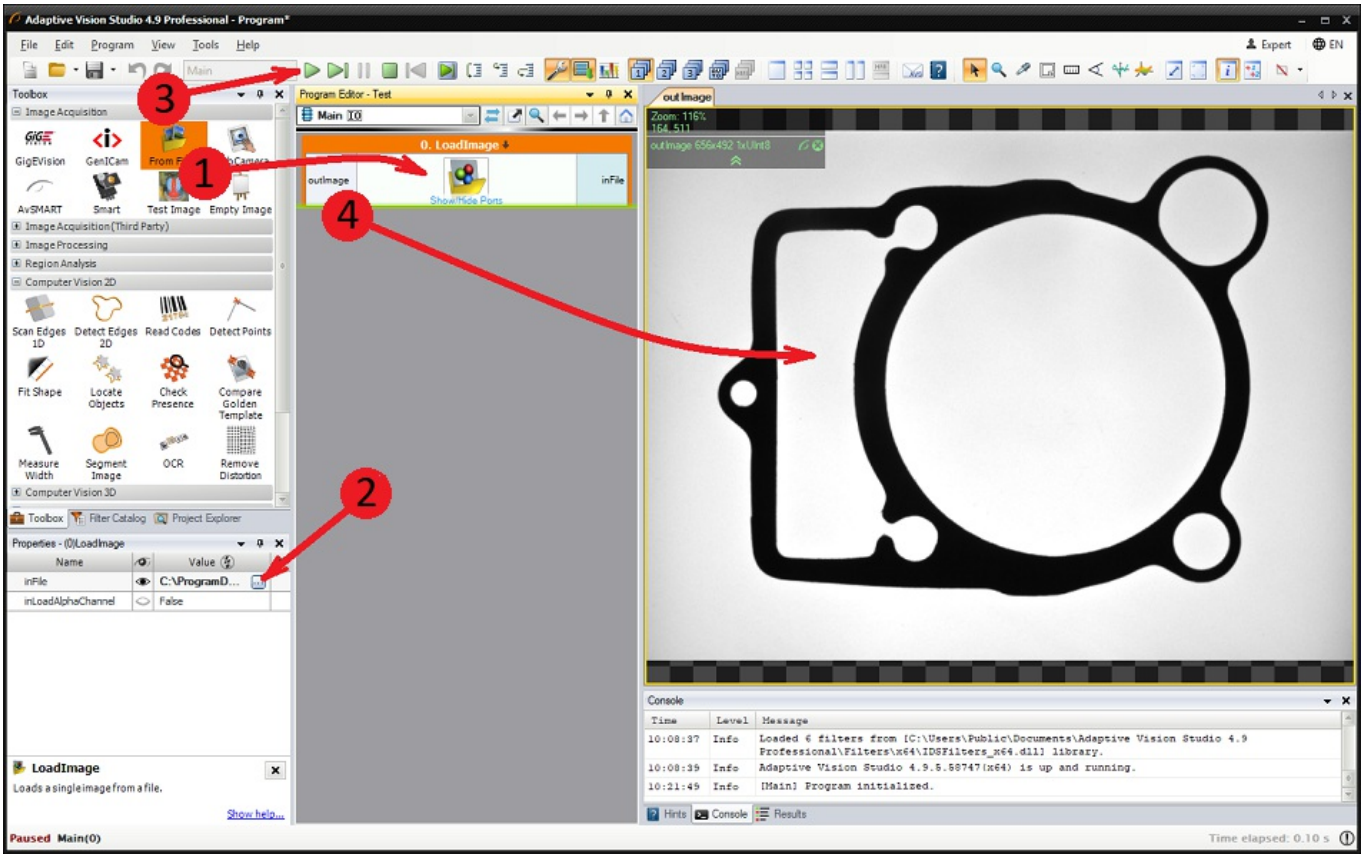
Acquiring Images

Acquiring Images from Files

Adaptive Vision Studio is not limited to any fixed number of image sources. Instead, image acquisition is a regular part of the library of filters. 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. It will make your program work in a loop until all images are read from the directory specified with the **inDirectory** parameter.

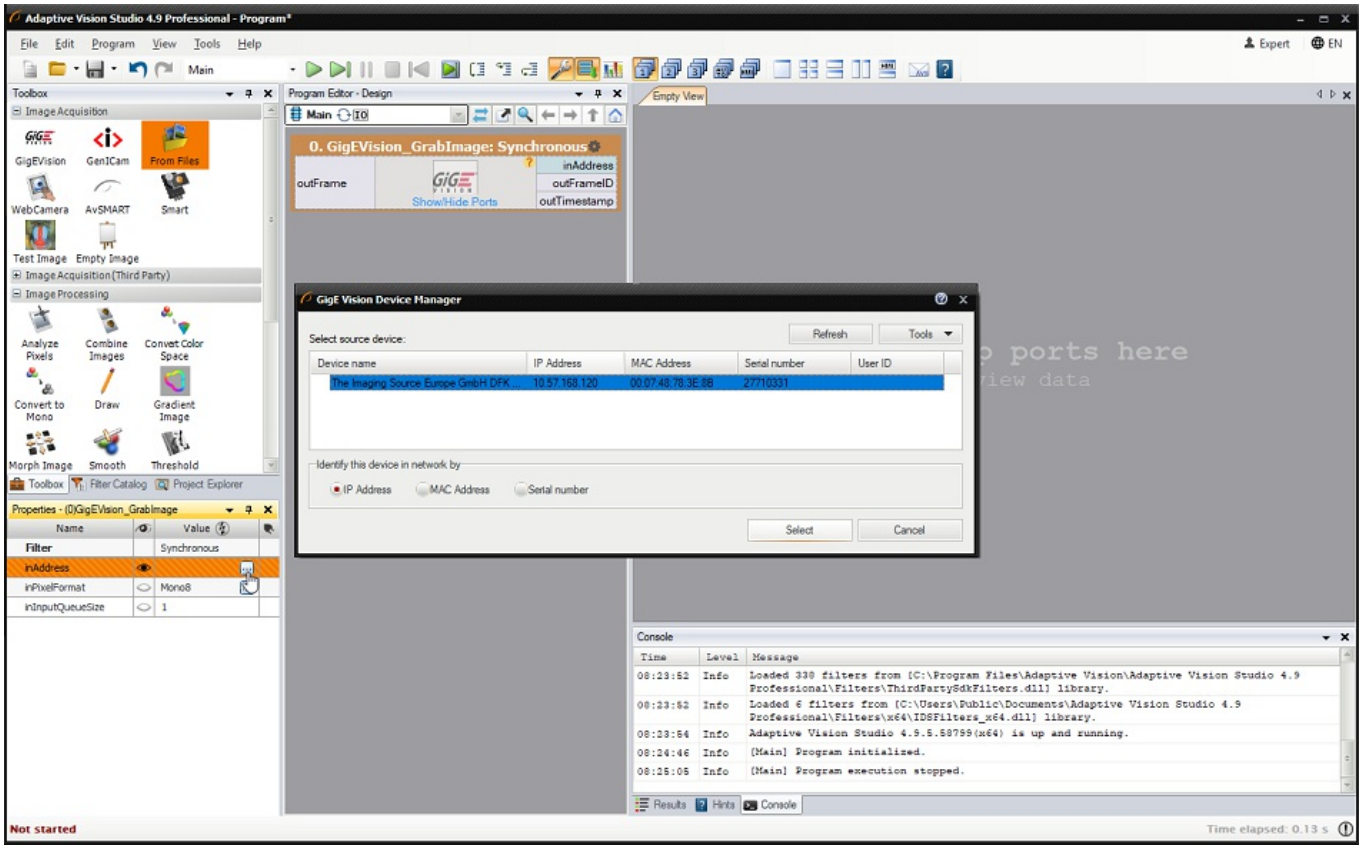
Acquiring Images from Cameras

For acquiring images from cameras and frame grabbers there are filters 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 Adaptive Vision Studio. Vendor specific interfaces are required for older and non-standard

camera models.



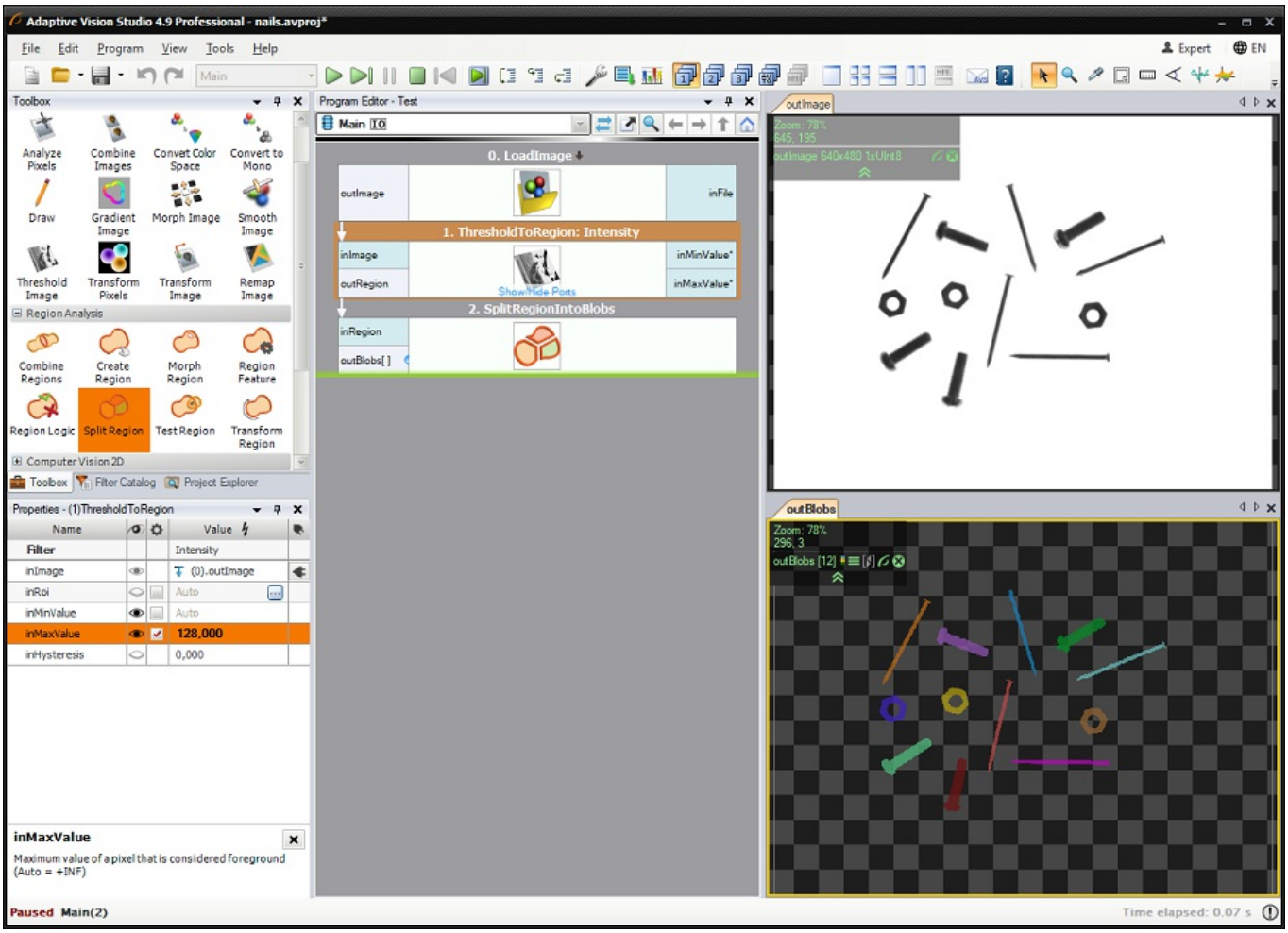
Connecting to a GigE Vision compatible camera.

First Program: Simple Blob Analysis

This article demonstrates the basic workflow in Adaptive 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, 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*):



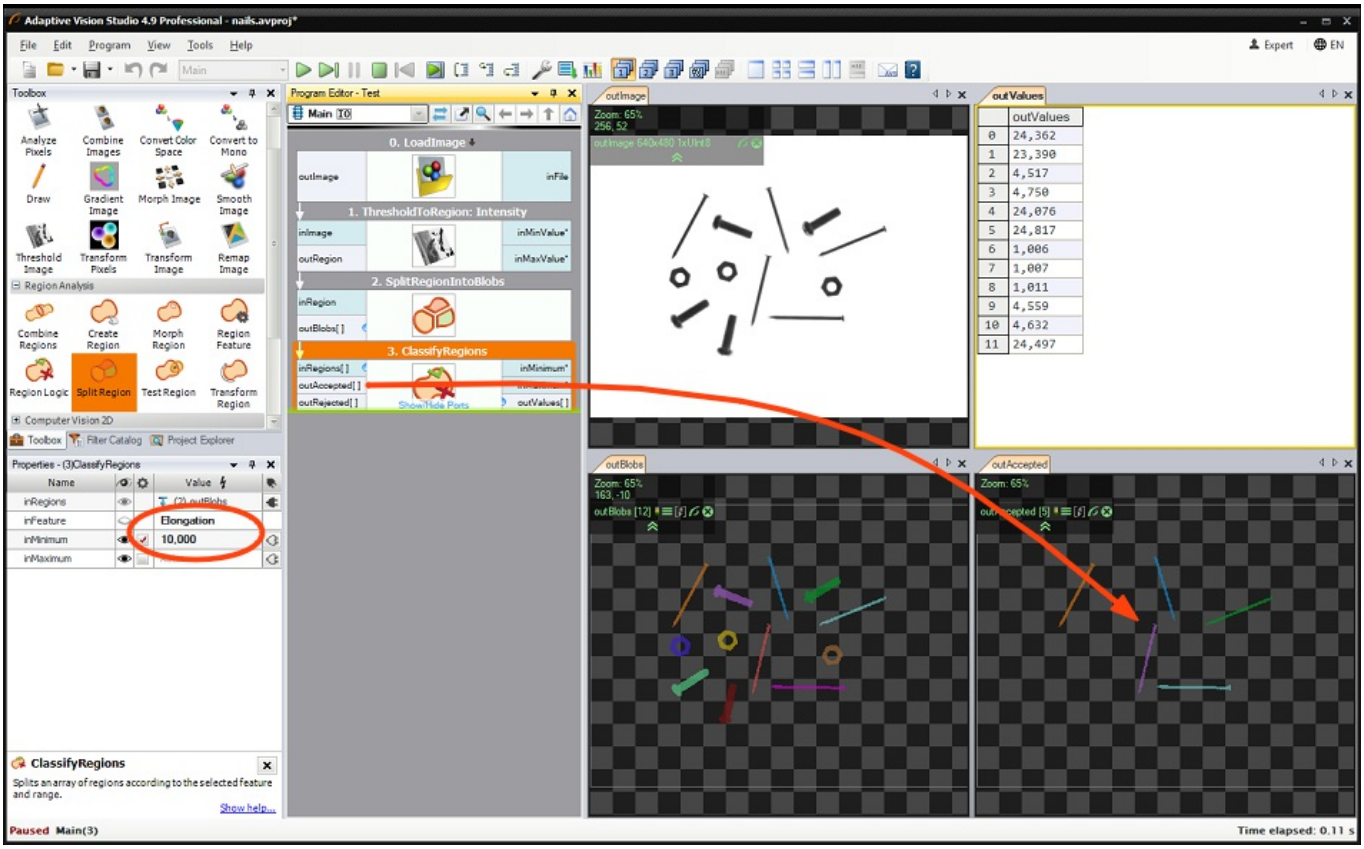
Extracting blobs from an image.

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.

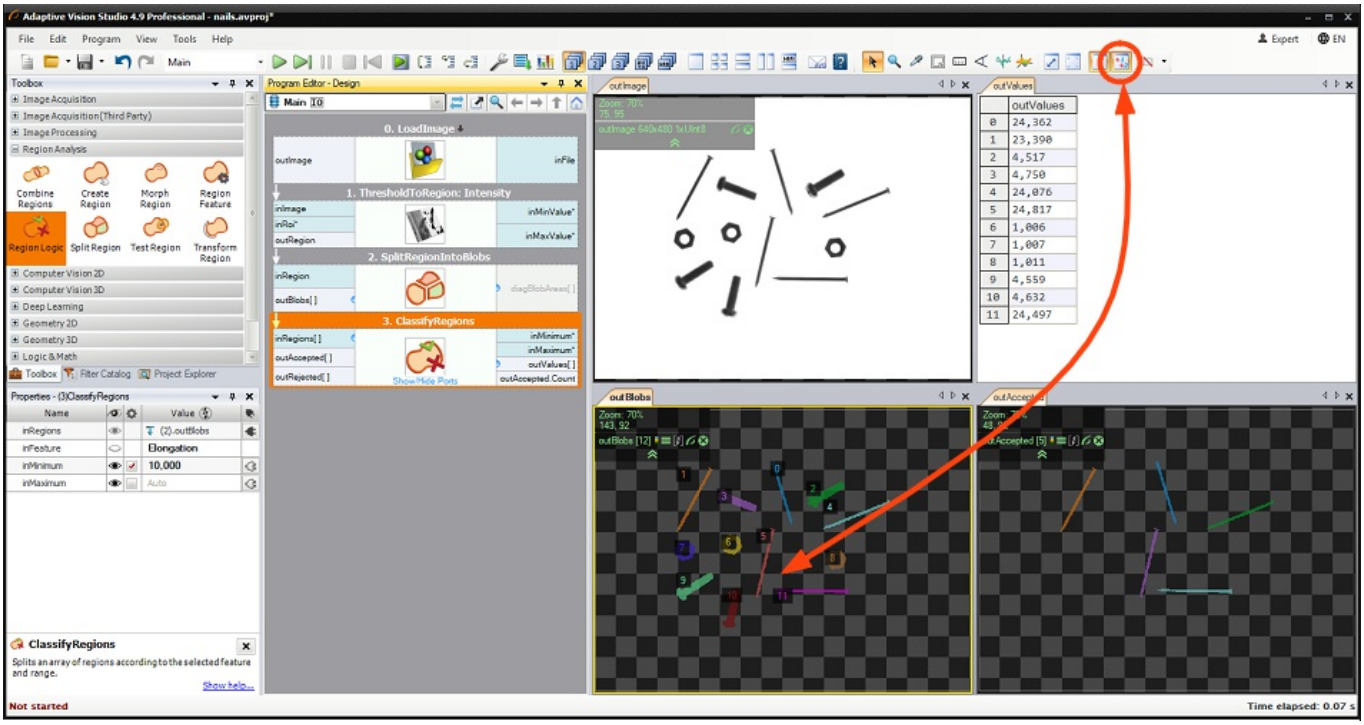
Classifying the Blobs

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.

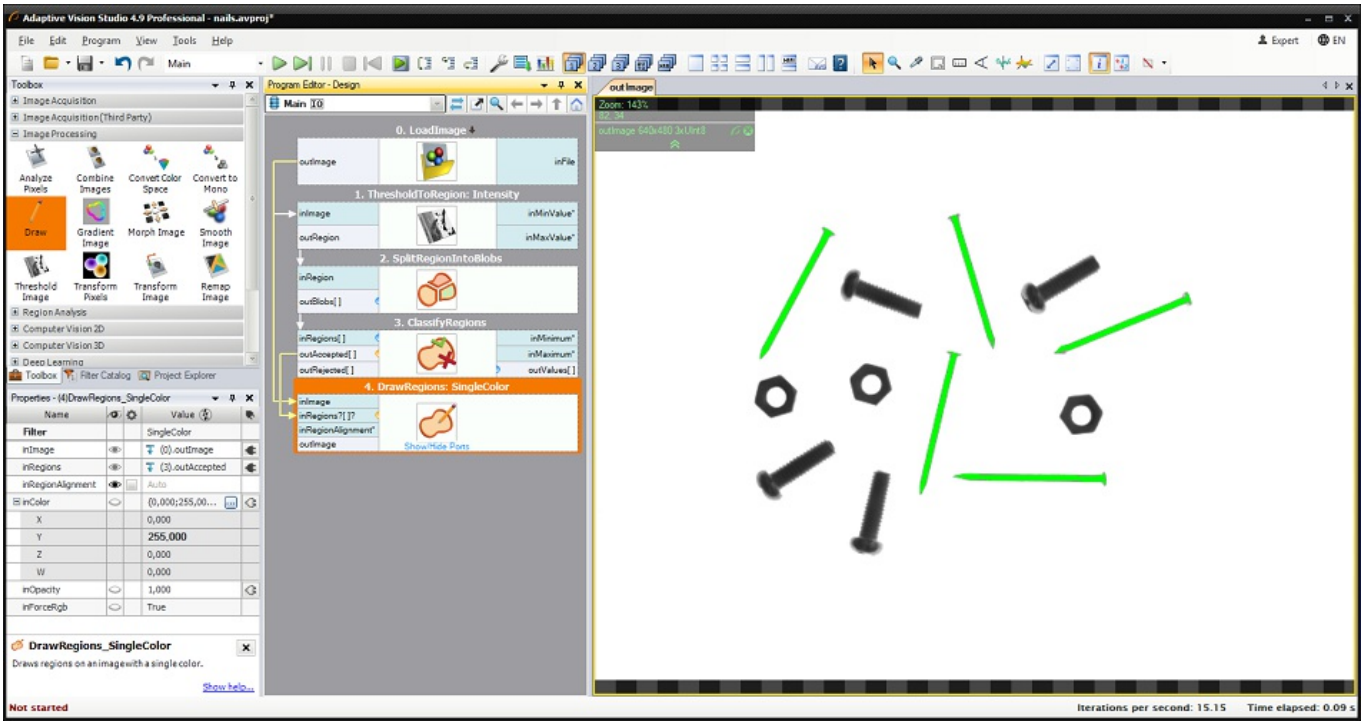
There is also the **outValues** output which contains the feature values of the individual blobs. This can also be displayed in the Data Previews as a table of real numbers. The indexes in this table correspond to the blobs, which can be shown by using the "Show Indexes of Elements" option in the selected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



Showing indexes of individual blobs.

Drawing the Results

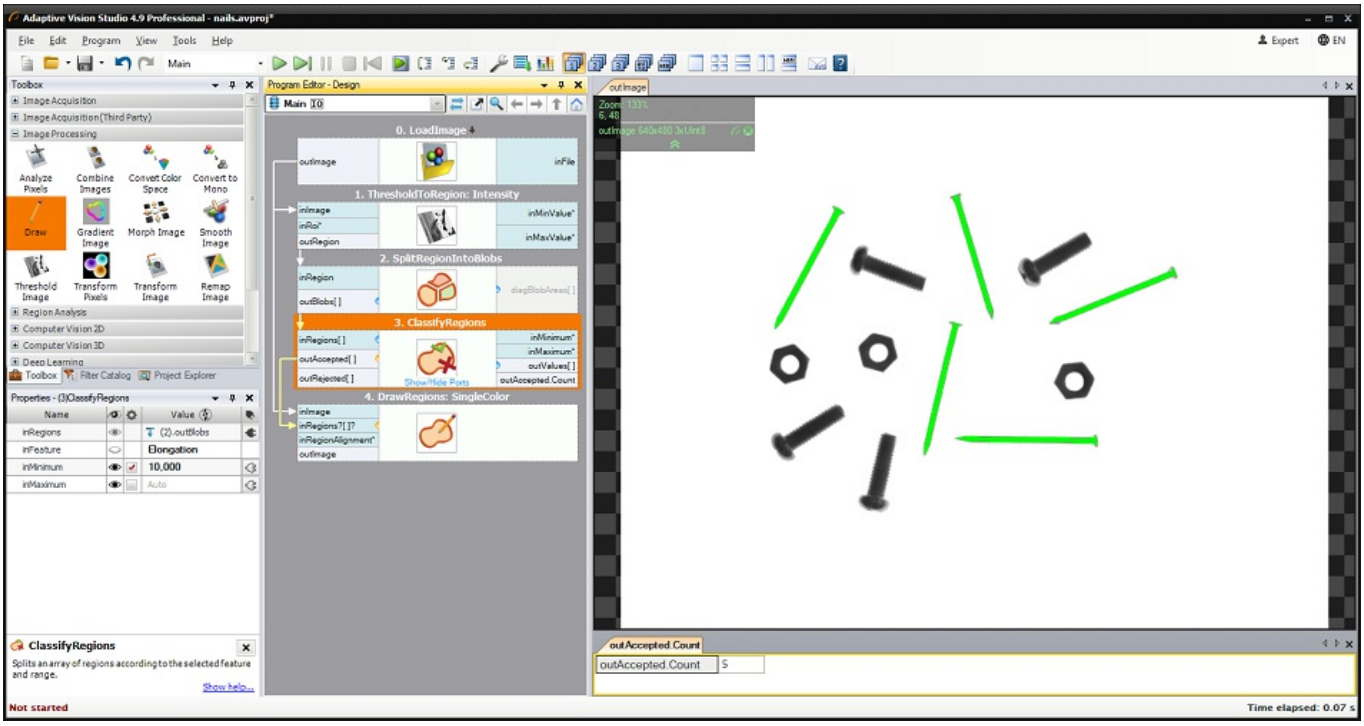
Finally, we can create an output image (e.g. for displaying in the HMI) with the nails marked in green. For this purpose we use the **DrawRegions_SingleColor** filter, which needs to have its **inImage** and **inRegions** inputs appropriately connected. The **inColor** input defines the required color 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 Adaptive Vision Studio consists of selecting filters 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 Adaptive Vision Studio.