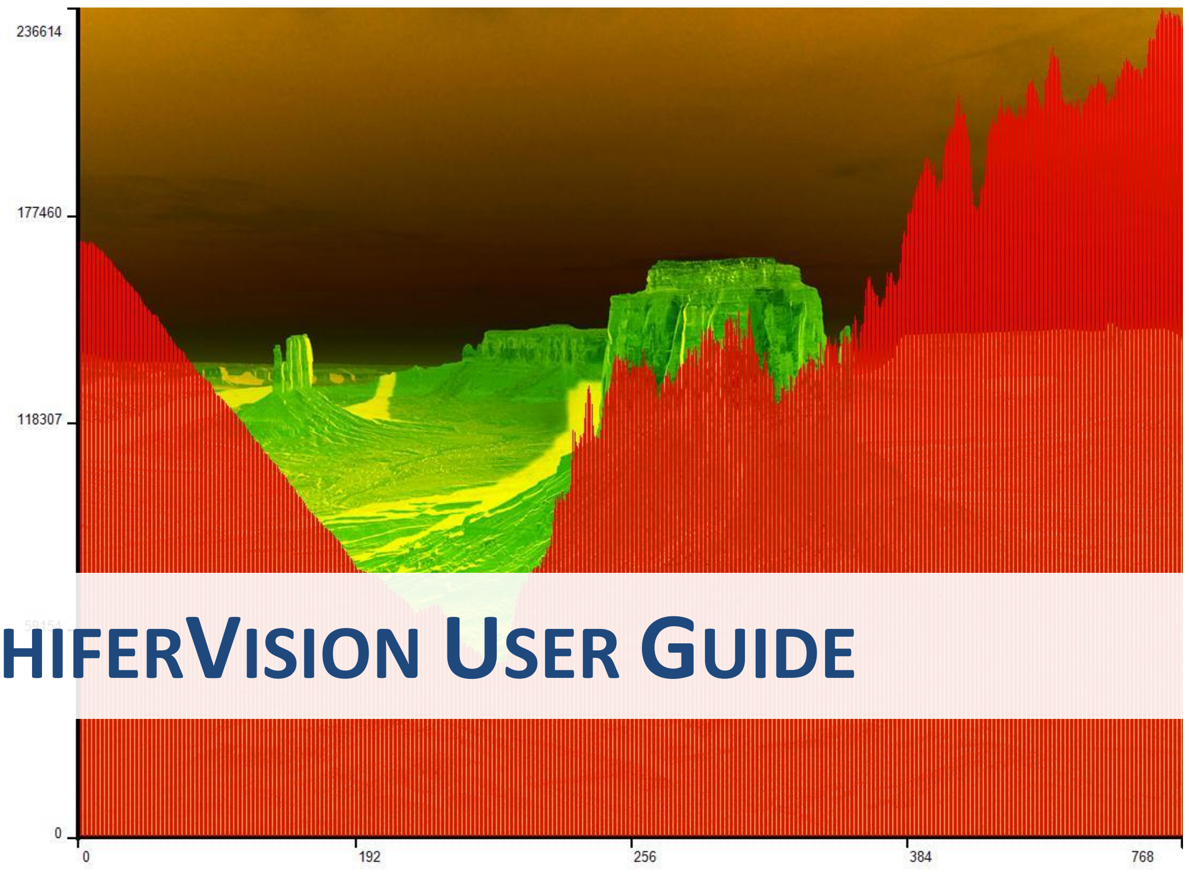
Bernhard Schifer GmbH



SchiferVision User Guide

1 Contents

2 What is SchiferVision

SchiferVision is a simple-to-use image processing tool, which allows building a chain of image processing filters and saving this chain as a project for further use. This further usage is shown in section “Second Step Using SchiferVisionProject” below.

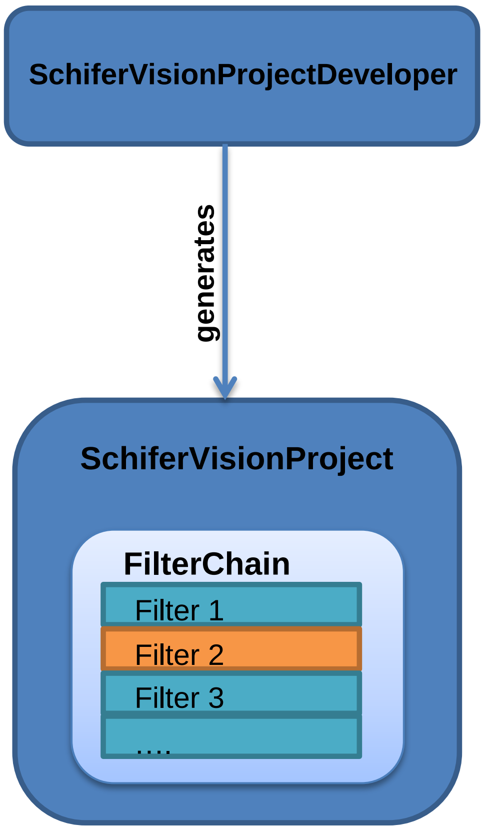
3 The architecture of SchiferVision

SchiferVision works with a two-step concept. The first step is the development of a project, which is a chain of different filters. More about filters later. You can then save your project and use it over and over again.

The second step is executing this project to realize some imaging tasks for a certain image, a folder of images or an image selection.

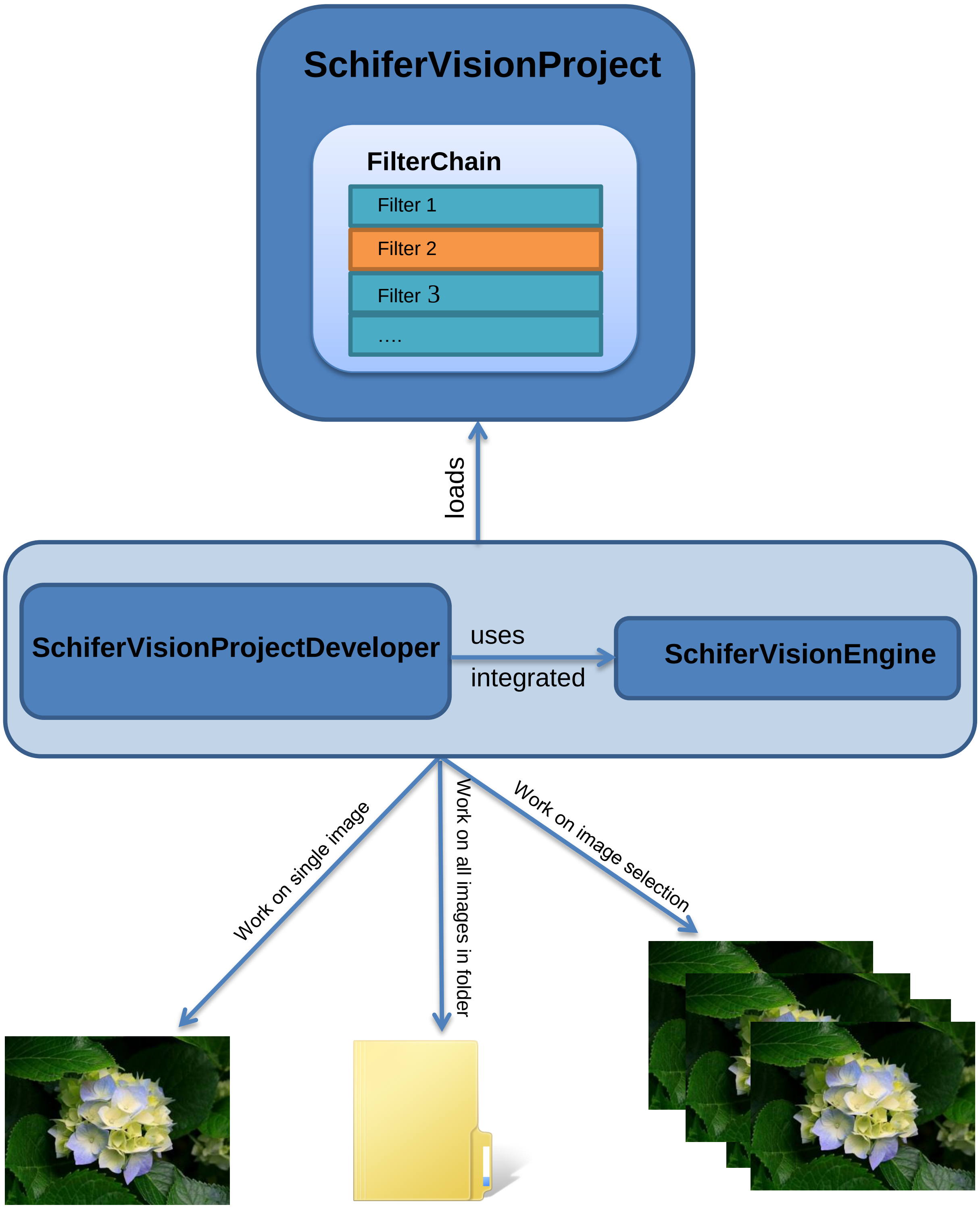
You can use the SchiferVisionProjectDeveloper, the Batch Processor or the CommandLineProcessTool to execute a project, or you can build your own .Net-Application and use the SchiferVisionEngine to execute the project.

3.1 First Step: Developing a project

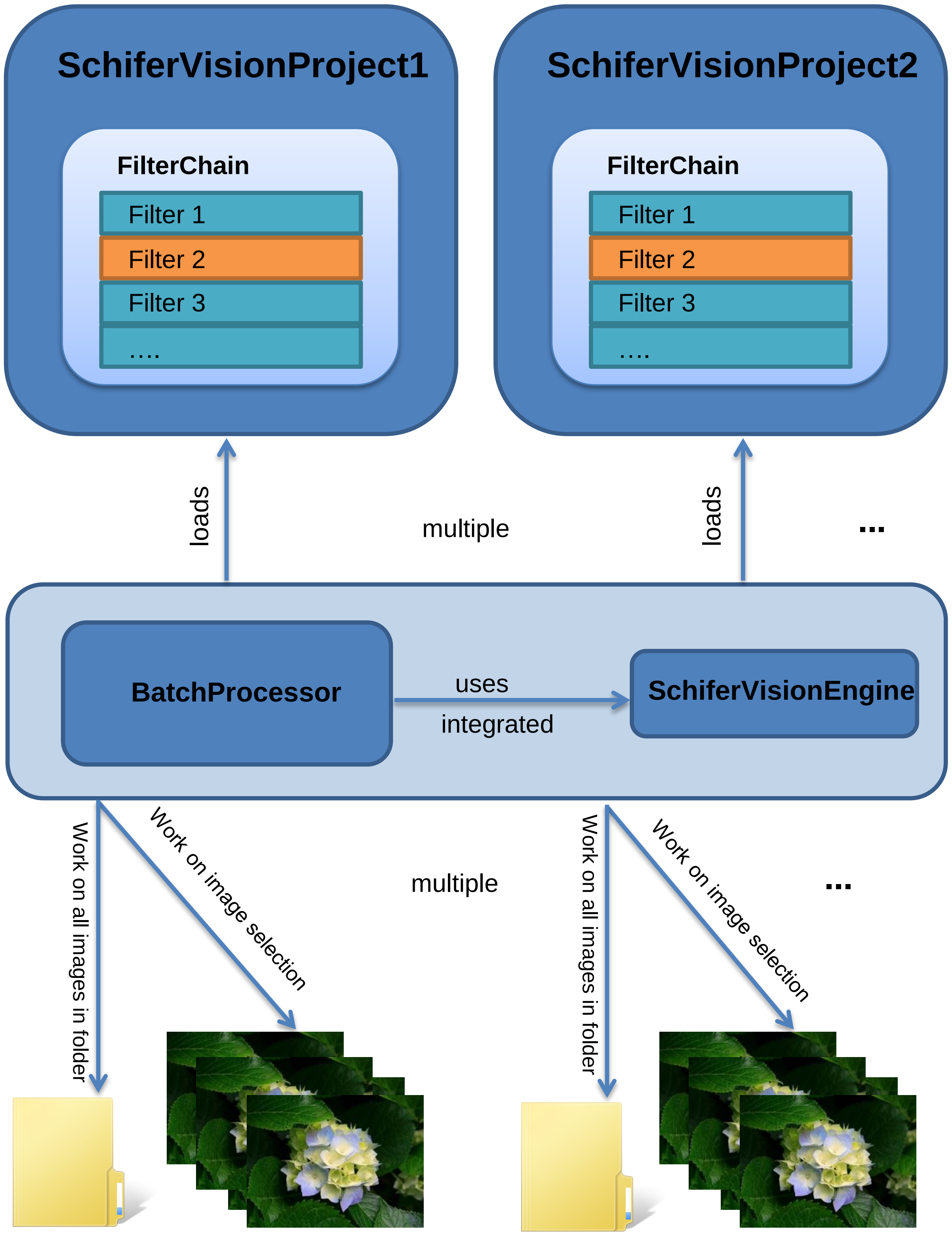


3.2 Second Step: Executing the project

3.2.1 First possibility: Using SchiferVisionProjectDeveloper



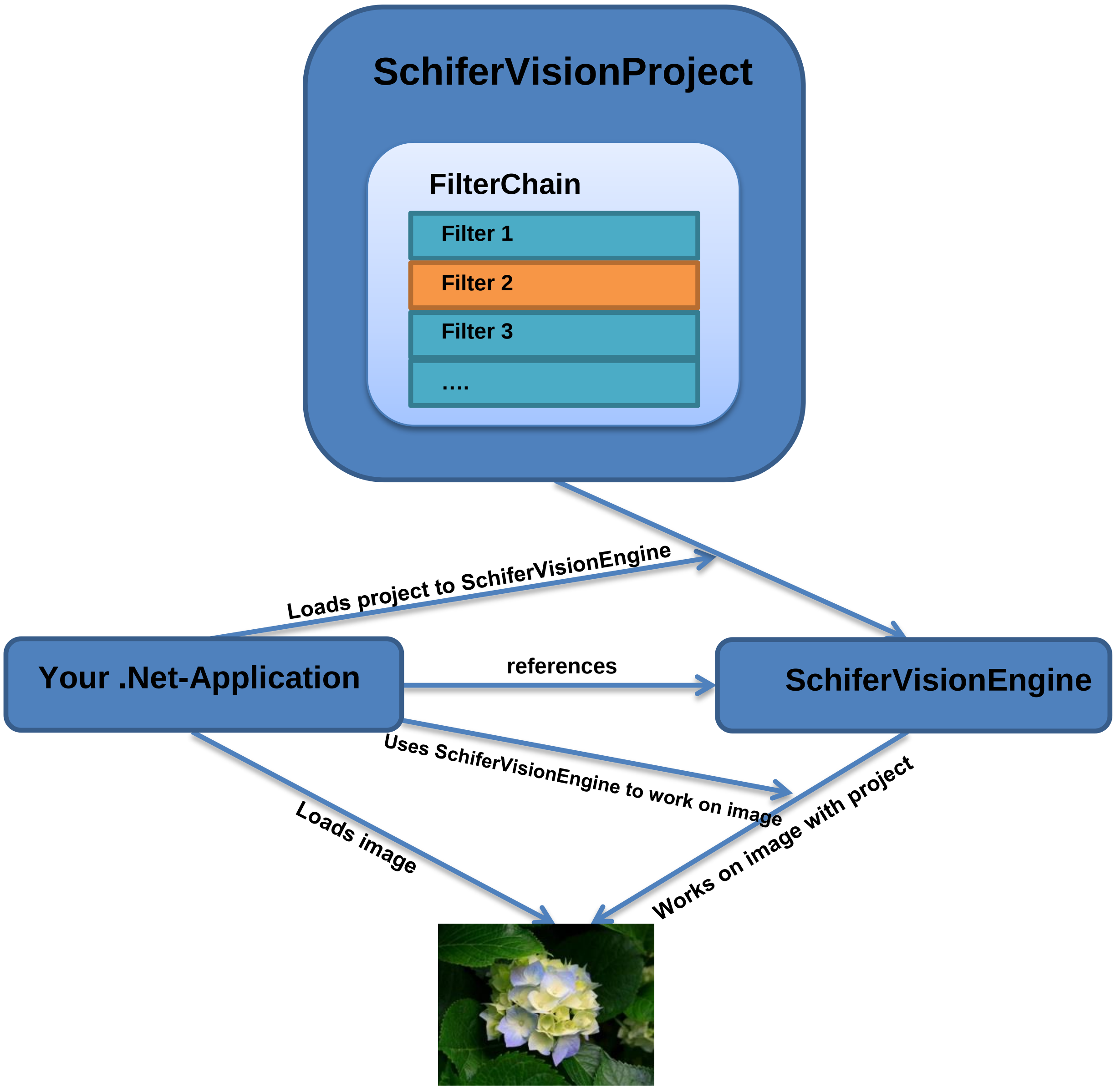
3.2.2 Second possibility: Using the BatchProcessor-Tool



3.2.3 Third possibility: Using the CommandLineProcessTool

If you want to start a project from the command line or from a not-.Net-application you may use the CommandLineProcessTool. The CommandLineProcessTool can work on single images or folders. You find the CommandLineTool.exe in the SchiferVision-folder of your program files or you use “Create Package for .Net Usage” where you find everything you need to use your project (Compare section Menu Project -> Create Package for .Net Usage).

3.2.4 Fourth possibility: Using your own .Net-Application



4 Key Features

- Use filters to calculate parameters of following filters
- Use subprojects to easily reuse your development

- An interactive developing Gui, which offers a lot of features for fast development of your imaging projects.
- The Gui offers live trying of filters for fast finding of accurate parameter values.
- View the output of each filter
- Visualize calculated objects
- Build your own ShortList projects and apply them with a single click
- Many sample projects which show a lot of different image processing techniques
- Encryption of the project to save your know how
- The plugin concept for filters allows building your own filters without compiling any source (for developers).
- More than 300 predefined filter types to meet a lot of needs of image processing. Many of them are licenced under various open source licences. So be careful when using them. (Compare section Menu Project -> Create Package for .Net Usage)
- Use the SchiferVision engine in your own .Net applications. Write your own applications and use your imaging projects directly in these applications (for developers).

5 Core Concept Basics

The main concept consists of four different base types of filters in the filter chain:

1. Direct manipulating steps
2. Calculating steps
3. Controlling steps
    a. Skip filter steps in certain cases (ConditionalSkipFilterStep)
    b. Jump to filter steps in certain cases (ConditionalJumpToFilterID)
    c. Loop filter steps (LoopOverObjectUntilFilterID)
    d. BreakLoop filter steps (BreakLoop)
    e. ContinueLoop filter steps (ContinueLoop)
4. Subprojects

The first one manipulates the image. However it can calculate values or “built-in-objects” as well.

The second one calculates values or “built-in-objects” which serve as parameter for direct manipulating steps or other calculating steps.

The third one offers the flexibility of skipping filter steps in certain cases or jumping to other filter steps. Furthermore you can loop certain steps over an array of “built-in-objects”.

The fourth one offers the possibility to execute another project in your current project as single filter step. This is very powerful to generate clearly arranged large projects. Of course you may use some projects in many other projects as subprojects. If you change this project, it will change wherever you use it as a subproject.

This flexible concept allows building powerful image processing projects and is, above all, simple to use.

To demonstrate this concept, let us take a look at some simple image processing tasks.

In our first sample we have images which we want to enhance and scale.

As enhancement we take a contrast correction with a fixed correction value.

The scaling should be a fixed factor of the original image.

So we need two filter steps:

ContrastCorrection -> Resize

Both filter steps manipulate the image directly and are therefore direct manipulating steps.

Now consider that not all of your input images look great after contrast correction with a fixed value. Fortunately SchiferVision offers the possibility of calculating input parameters for filter steps. To calculate the contrast correction value, we can use a calculating step, which individually calculates some statistics of every input image and so generates an individual contrast correction value for each input image.

StatisticsAsCalculation (output: calculated Parameter) -> ContrastCorrection (input: calculated Parameter) -> Resize

Now consider furthermore that some of your images may need saturation correction, but only a few of them. And this is where the third member, the controlling step, comes in. With this filter type it is possible to skip the saturation under certain circumstances. To calculate the condition, we can use a calculating step again. So we may have the following filter steps:

StatisticsAsCalculation (output: calculated parameter) -> ContrastCorrection (input: calculated parameter) -> StatisticsAsCalculation (output: calculated parameter) -> ConditionalSkipFilterStep (input: calculated parameter) -> SaturationCorrection -> Resize

So now we have a complete image-processing project which we can save for future use.

You can use it directly in SchiferVisionProjectDeveloper for application on single images or on an entire folder of images.

There is no need of programming know how; however, you can use the SchiferVisionEngine without the SchiferVisionProjectDeveloper directly in your .Net-Applications. For further information on this topic, refer to section Programming / Referencing SchiferVision Engine.

Furthermore you can build your own filters and easily plug them to SchiferVision. For further information on this topic refer to section Programming / Referencing SchiferVision Engine as well.

6 Core Concept

6.1 Common Objects

As described in section “Core Concept Basics”, there are these four types of filters. The calculating step type generates different types of objects, which we now want to have a look at.

There are simple types like:

- Integer values: e. g. 1234
- Decimal values (float and double precision): e.g. 1234.1234

- Strings: e.g. ABCD

There are advanced types like:

- A list of integer, double values (referred to as TSIntList or TSDoubleList): e.g. 1234, 5678, 4321, 8765 …
- A dictionary of double values (an integer Index and its corresponding double value, referred to as TSDictionaryList): e.g. [1, 232.2323], [2, 32.343] …
- Circle: represents a circle by its location and its radius (referred to as TSCircle)
- Ellipse: represents an ellipse by its central point, size and rotation angle (referred to as TSEllipse)
- Rectangle: represents a rectangle by its location and width, height and rotation angle (referred to as TSRectangle)
- Point: represents a point by its X- and Y- coordinates (referred to as TSPoint)
- Line: represents a line by its starting- and endpoint (referred to as TSLine)
- CircleList: represents a list of circles (referred to as TSCircleList)
- EllipseList: represents a list of ellipses (referred to as TSEllipseList)
- RectangleList: represents a list of circles (referred to as TSRectangleList)
- PointList: represents a list of points (referred to as TSPointList)
- PointListList: represents a list of PointLists, e.g. for several contours (referred to as TSPointListList)
- LineList: represents a list of lines (referred to as TSLineList)
- BlobContainer: represents a container for blobs (connected white objects in images with different background) (referred to as TSBlobContainer)
- HomMat2D: represents a homogenous translation matrix, e.g. for rectifying images. (referred to as TSHomMat2D)
- DenseHistogram: represents a DenseHistogram. (referred to as TSDenseHistogram). Only available with the Emgu-extension.
- Character: represents a single character, which was recognized with OCR. The advantage over a normal string is that possible alternatives and confidences are available for further use as well (referred to as TSCharacter)
- CharacterList: represents a list of characters (referred to as TSCharacterList)
- ImageList: represents a list of images (referred to as TSImageList)
- FloatMatrix: represents a two dimensional array of float values (referred to as TSFloatMatrix)
- LocatedString: represents a string and its location (referred to as TSLocatedString
- LocatedStringList: represents a list of LocatedStrings (referred to as TSLocatedStringList)
- CommonObjectList: represents a list of CommonObjects. The list can host different kinds of CommonObjects. (referred to as TSCommonObjectList)
- CommonObjectDictionary: similar to CommonObjectList, it hosts different kinds of CommonObjects. Here the list is not indexed, but a string is used as key to access the elements. (referred to as TSCommonObjectDictionary)
- StringDictionary: represents a dictionary of strings where the key is a string as well. (referred to as TSStringDictionary)
- StringList: represents a list of strings. (referred to as TSStringList)

There may be other types which only serve as interaction between different kinds of special filters. E.g. if you build your own filter which outputs a special type and another filter which uses this output as input. In that case the type is not known to the SchiferVisionProjectDeveloper or SchiferVisionEngine, which is no problem. SchiferVisionProjectDeveloper just displays its class name and SchiferVisionEngine passes it through to the desired filter.

Some of these types can be saved and reloaded for further use. Refer to section View Object.

The advanced types have one thing in common. They are referenced through their use in the filter chain. That means one filter creates an instance of an advanced type and later filters, which use this as calculated object, use this instance. Furthermore, that means, if this later executed filter changes the values of the instance, the values are changed even if you use the initial filter which created the instance as calculated object for a filter, which is executed after the filter which changed the values. E.g. a filter with index 2 creates a TSBlobContainer. A filter at index 3 filters the blobs in this container. A filter with an index of 4 and higher, which references the TSBlobContainer of filter at index 2 will get the filtered TSBlobContainer from index 3. If you need the initial values of the TSBlobContainer before the filtering, you have to execute this step before the filtering. E.g. index 2 creates the TSBlobContainer, index 3 gets all bounding rectangles (e.g. 400 rectangles). Index 4 filters the TSBlobContainer. Index 5 gets all bounding rectangles of the filtered TSBlobContainer (e.g. 100 rectangles), even if it references index 2!

6.2 A deeper look at controlling steps

As pointed out in the section about core concepts basics, there are three different types of controlling filter steps. We took a brief look at the first one (ConditionalSkipFilterStep) in this section as well. Now let us take a deeper one at these filter types.

The first two steps both share a field called CalculatedInputCondition where you can input a condition via Reverse Polish Notation[1] (RPN), e.g. $4 > 100 ? 1 : 0. This means if the output of the filter step with ID 4 is higher than 100, set the condition to true, else to false. Compare to section SchiferVisionProjectDeveloper – Project Steps Properties. If this condition is true, the following step will be skipped (ConditionalSkipFilterStep) or the execution will jump to the filter step with the given ID and skip all steps until then (ConditionalJumpToFilterID).

Hint: If you set the JumpToFilterID of a ConditionalJumpToFilterID to 999999, this will end the project execution if the condition is true.

Hint: If you want a filter which always jumps to a filter step with a given ID, just use a condition which is always true, e.g. 1 == 1.

The ConditionalJumpToFilterID step offers a second working mode. If you leave the CalculatedInputCondition empty and use the CalculatedSwitchToFilterID field instead, you can input multiple pairs of conditions and destinations. The format is as follows: CalculatedInputCondition1 , JumpToFilterID1 ; CalculatedInputCondition2 , JumpToFilterID2

[1] Compare http://en.wikipedia.org/wiki/Reverse_Polish_notation

… E. g.: $0 > 200 , 2 ; $0 > 150 , 4 ; $0 > 100 , 7 ; $0 < 101 , 9. The sorting of these pairs matters as they are executed top down. The first true condition wins. To solve this example: If the output of the filter step with ID 0 is higher than 200, the execution will jump to ID 2. If the output is bigger than 150 but smaller than 201, the execution will jump to ID 4 and so on.

The third controlling step gives you the power of repeating steps for calculated objects, which are returned as a list, e.g. a RectangleList. Imagine you are looking for rectangles on your image and want to save each of these rectangles as separate images. First, you will do some blobprocessing for example and then have a TSRectangleList. So let us take the LoopOverObjectUntilFilterID step and repeat the steps Crop (Cropping the Image via the rectangle) and ConvertImageFormatRenameAndSave (save the image in a certain format and with a certain name) for the number of rectangles in the TSRectangleList. It is possible to use several loops, even nested loops. However it is important that the loopUntilFilterID is never the same and the loopUntilFilterID of a nested loop is executed earlier to the loopUntilFilterID of the parent loop.

Hint: Sometimes it is useful to execute some filter steps only once, even if you apply the project several times to many images. E.g. you load a second image which serves as mask, or you load a classifier for OCR. These are time-consuming steps, which may be executed only once. To achieve that there is a calculating filter that returns a counter of project runs (GetProjectRuns), you can use this as input for a ConditionalSkipFilterStep-Filter or a ConditionalJumpToFilterID-Filter. There is also a calculating filter that returns a counter of loop runs (GetLoopRuns). You can use this to get elements of a list, e.g. you loop over a list of rectangles, but you have a list of points for the same things as well. So you use GetLoopRuns and then GetListElement and use the loop run counter as an index for the list element to return. Do not forget: A list starts with index 0 whereas the counter starts with 1. So you have to subtract 1 from the counter to get the right list index.

The fourth controlling step enables breaking a loop. E.g. you are looping a RectangleList and search for a special one. When you have found it, there is no need to loop further. So you can break the loop to speed things up.

The fifth controlling step enables continuing a loop. The steps after this one in the current loop run will not be executed and the loop will continue with the next loop run.

Hint: The LoopOverObjectUntilFilterID step is a so-called for each loop. There is no built-in function which loops for predefined times, let’s say 10 times. To achieve this, you can build a TSIntList and loop over this list (GenerateIntList is a fast way to do that). You can achieve the same by using a ConditionalJumpToFilterID-Filter and jump back if a condition is true. Be careful using loops and jumps as there is no check if they are possible or producing endless loops.

7 Installation Guide

Refer to the file readme.txt in the install folder.

8 System Requirements

CPU: Intel Core2 Quad Q6600 @ 2.4 Ghz (or equivalent) or higher

RAM: 4 GB

OS: Windows V/7/8/8.1 – 32 or 64-Bit, not tested on Win 10 yet

Free Disk Space: 5 GB

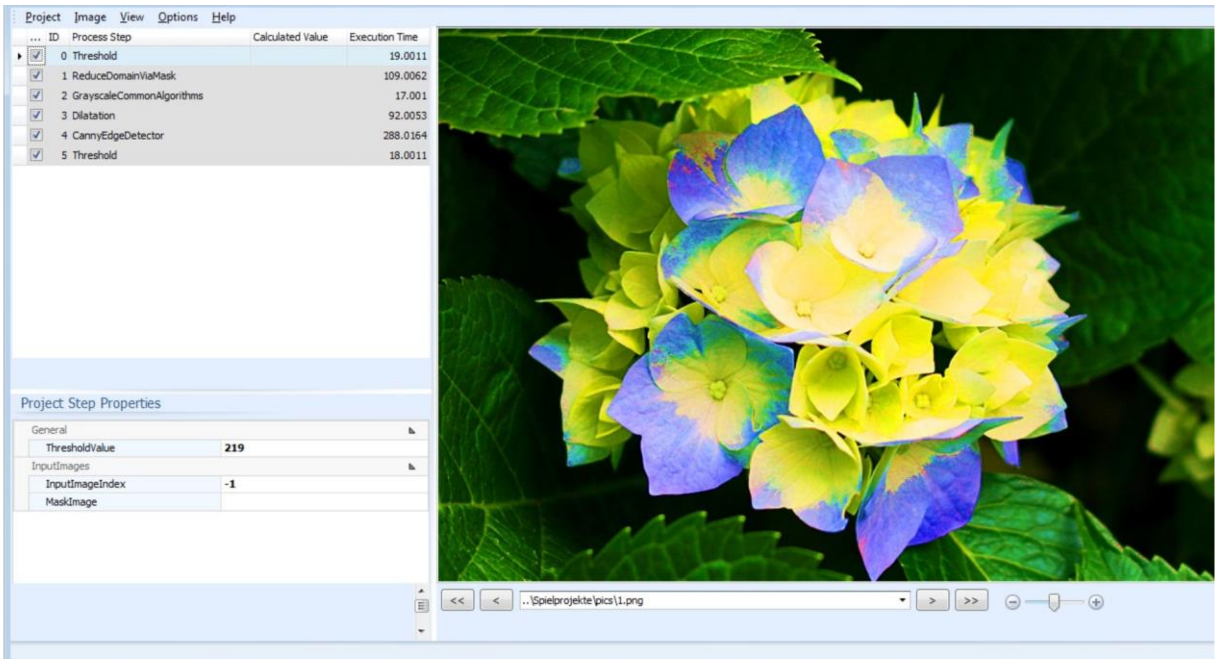
9 SchiferVisionProjectDeveloper

The SchiferVisionProjectDeveloper generates the filter chain (the project) and offers many useful tools to help and assist you in developing them.

9.1 Running SchiferVisionProjectDeveloper

Start -> Programs -> SchiferVision -> SchiferVisionProjectDeveloper

9.2 User Interface



...	ID	Process Step	Calculated Value	Execution Time
✓	0	Threshold		19.0011
✓	1	ReduceDomainViaMask		109.0062
✓	2	GrayscaleCommonAlgorithms		17.001
✓	3	Dilatation		92.0053
✓	4	CannyEdgeDetector		288.0164
✓	5	Threshold		18.0011

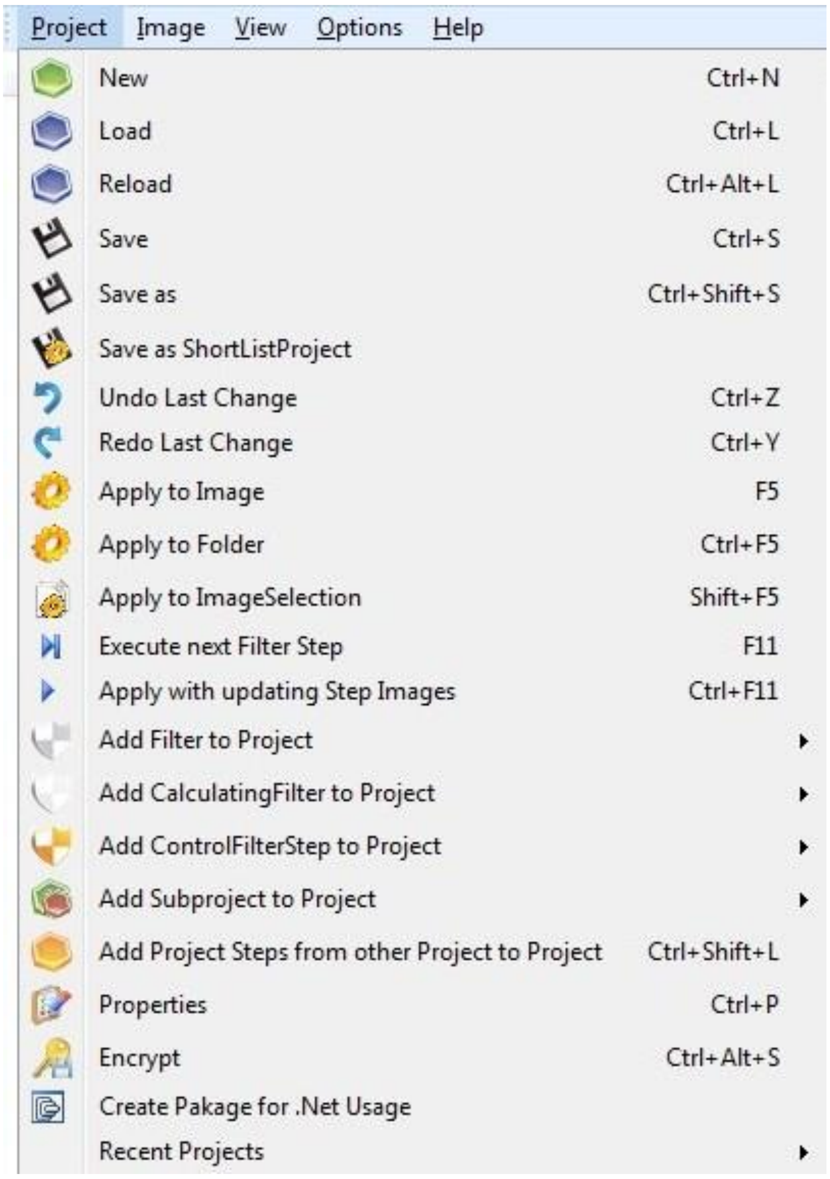
Project Step Properties

General	
ThresholdValue	219
InputImages	
InputImageIndex	-1
MaskImage	

The main menu consists of the following submenus:

- Project
- Image
- View
- Options
- Help

9.2.1 Menu Project


Project Image View Options Help
New Ctrl+N
Load Ctrl+L
Reload Ctrl+Alt+L
Save Ctrl+S
Save as Ctrl+Shift+S
Save as ShortListProject
Undo Last Change Ctrl+Z
Redo Last Change Ctrl+Y
Apply to Image F5
Apply to Folder Ctrl+F5
Apply to ImageSelection Shift+F5
Execute next Filter Step F11
Apply with updating Step Images Ctrl+F11
Add Filter to Project
Add CalculatingFilter to Project
Add ControlFilterStep to Project
Add Subproject to Project
Add Project Steps from other Project to Project Ctrl+Shift+L
Properties Ctrl+P
Encrypt Ctrl+Alt+S
Create Pakage for .Net Usage
Recent Projects

New: Asks to save the current project, clears everything in the current project and opens a new blank project.

Load: Opens a file browse dialog to search for a project file. By default, the file type is .tsi. Then loads the selected file as project. Note: You can do the same by double clicking the desired project file without opening the SchiferVisionProjectDeveloper first.

Reload: Reloads the project. E.g. if you use subprojects and made changes to these, you have to reload the main project to work with the changed subprojects.

Save: Saves the current project. If this is the first time and no name is specified, the “Save as” dialog will be opened.

Save as: Opens a file save dialog where you can specify a name and a location for your project. Saves the project then.

Save as ShortListProject: Saves the current project as ShortListProject. Compare “Apply ShortListProject” in the image context menu. Make sure the project has a direct manipulating output as it otherwise will be useless as ShortListProject.

Undo Last Change: Undoes the last change made to the project (adding, deleting, moving of steps, changing properties and so on). 10 steps are possible.

Redo Last Change: Redoes the change of an undo-operation.

Apply to Image: Applies (runs) the project for the currently loaded image.

Apply to Folder: Opens a folder browse dialog and applies (runs) the project over all images in the selected folder. Note: You will not see the images, but a notification about the current work in the notification bar at the left bottom of the SchiferVisionProjectDeveloper mask. For the output location, there are two different possibilities:

1. You use a saving filter, which does the converting and naming (ConvertImageFormatRenameAndSave)
2. You select an output folder where the images are stored as .png with the original name

Apply to ImageSelection: Runs the project to the image selection (Compare Menu Image->LoadImageSelection). The rest is similar to “Apply to Folder”.

Execute next Filter Step: Executes the selected filter step of the project steps grid. Works only if this is possible at the current state of project execution. A great tool for debugging, especially if you use loop filter steps.

Apply with updating Step Images: Executes the whole project but updates the picture box whenever a direct manipulating step is executed.

Add Filter to Project: Adds a direct manipulating filter to the project. Just click on the desired filter.

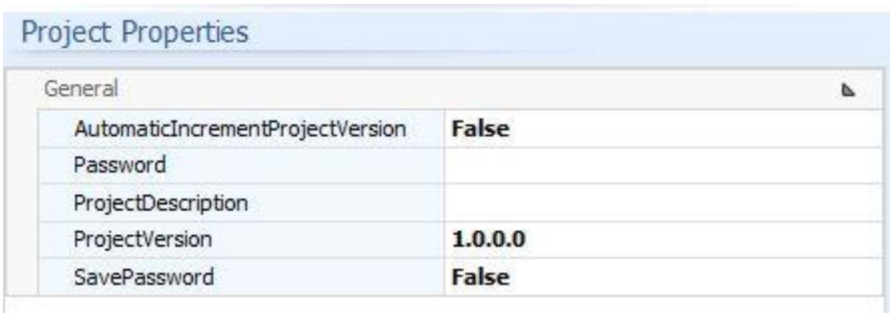
Add CalculatingFilter to Project: Adds a calculating filter to the project. Just click on the desired filter.

Add ControlFilterStep to Project: Adds a controlling filter to the project. Just click on the desired filter.

Add Subproject to Project: Adds a subproject to the project. This project will be executed like a single filter step. There are two types of subprojects. One that works like a direct manipulating step and one that works like a calculating filter step.

Add Project Steps from other Project to Project: Opens a file browse dialog and copies all filter steps from the selected project to the current project.

Properties: Shows the properties of the current project.



Project Properties

General

AutomaticIncrementProjectVersion	False
Password	
ProjectDescription	
ProjectVersion	1.0.0.0
SavePassword	False

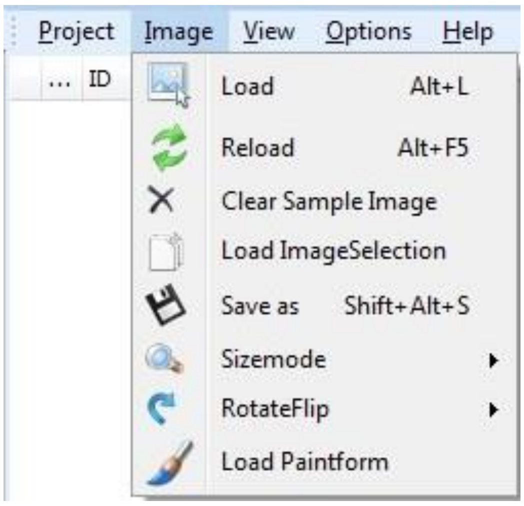
- **AutomaticIncrementProjectVersion**: Automatically increments the built-in number of the project version with each saving. E.g. first you have project version 2.2.1.1. After saving the project you will have 2.2.1.2
- **Password**: Here you can enter the password used for encryption (Compare Encrypt further down)
- **ProjectDescription**: Here you can enter a description for your project
- **ProjectVersion**: Here you can set a project version for handling different versions of the same project

Encrypt: SchiferVision provides the function of encrypting your projects. If you have entered a password in the project properties and click this button, a file with the same name but with the ending .tsx will be generated at the original location. If you try to open such a file, you will get a mask to enter your password. This is a powerful way to protect your work.

Create Package for .Net Usage: Opens a FolderBrowseDialog and copies all files needed to develop you own .Net Application with the current project. Compare section Programming/Referencing SchiferVisionEngine for further information. There is also a MessageBox, which informs you about the possibly needed licences and the licences will be copied as well.

Recent Projects: Shows a list of recently opened projects and loads one if you click on it

9.2.2 Menu Image



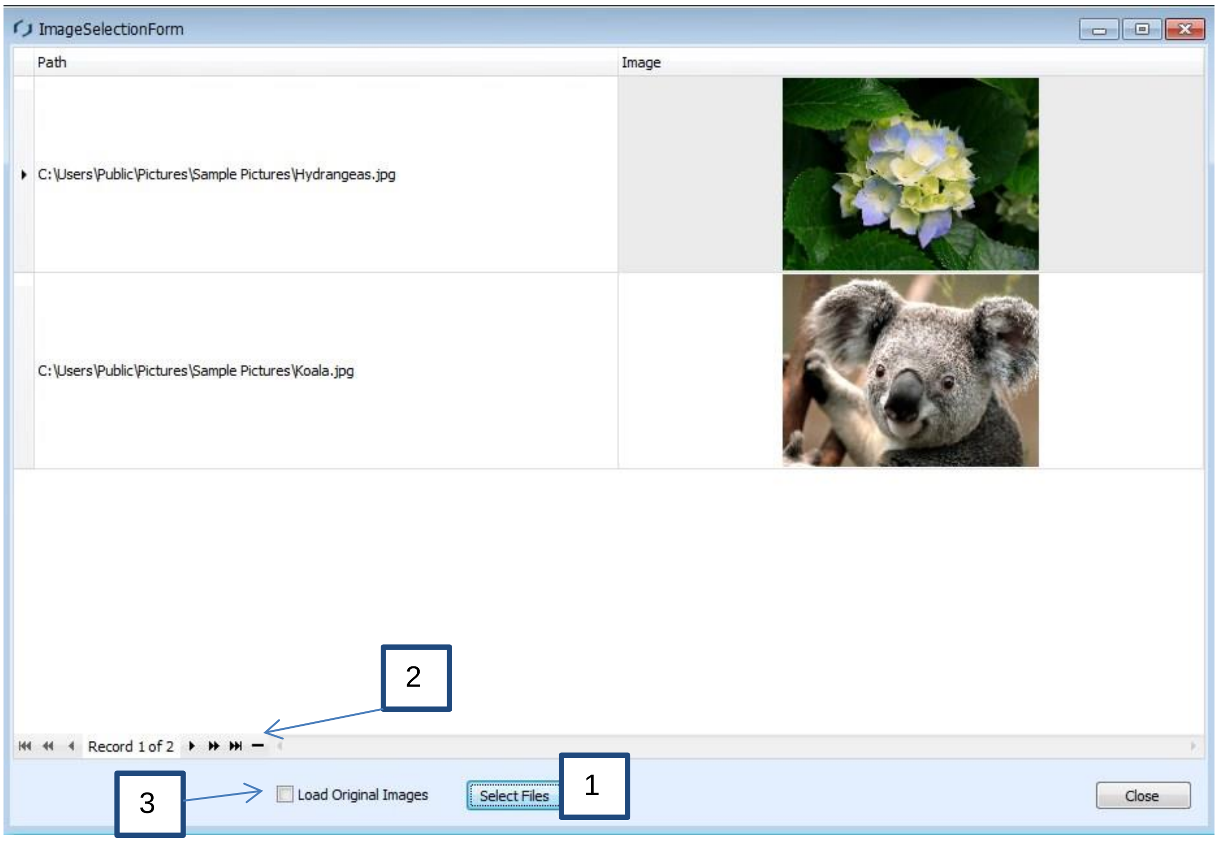
Methods for the sample image of the project

Load: Opens a file browse dialog and loads the selected image to the picture box. This image serves as initial image to which the project will be applied.

Reload: Reloads the image. E.g. after applying the project you want the original image. Only works if you inserted the image via Load or pasting an image path and not by pasting a bitmap.

Clear Sample Image: Removes the sample image from the project if you do not need it anymore.

Load Image Selection: Opens a form where you can load several images from different locations.



C:\Users\Public\Pictures\Sample Pictures\Hydrangeas.jpg
C:\Users\Public\Pictures\Sample Pictures\Koala.jpg

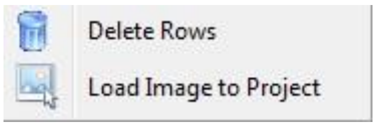
You can use the “Select File”-button (see 1.) which opens a FileBrowse-dialog where you can select multiple files of one folder. If you use it again, the images will be appended if the selection does not include the image. To delete an image, select it and press the minus at the bottom of the grid (see 2.). You can also use the context menu (compare below) to delete images from the selection. You can also drag and drop images from the explorer to the grid. If you close and reopen the form by pressing Load ImageSelection again, you will see your ImageSelection again. The project can be applied to this selection of images (Compare Menu Project->Apply to ImageSelection). You can sort the images in the selection by drag and drop. This is useful if you want to use the selection as TSImageList. The ImageSelection is available for the filters as TSImaglist with an ID of -1. Compare section Project Steps Properties -> Section Calculated Objects.

If you do not check “Load Original Image” (see 3.), only Thumbnails will be loaded. This will drastically reduce the memory usage and you can put far more images to the selection. When applying the project to the ImageSelection and “Load Original Image” is checked, the images will be taken from the memory. If it is not checked, each image will be loaded during project applying.

Hint: If you check “Load Original Image”, all the images are loaded to the memory, hence a selection of large images may cause memory problems. In this case it is better to uncheck it or to use the “Apply to Folder” method, where only one image is opened.

Hint: If you do not check “Load Original Image”, the TSImageList which is generated from the ImageSelection (Compare above) only includes the Thumbnails.

A right click to a row in the grid opens the following context menu:



Delete Rows: Deletes the selected rows from the image selection.

Load Image to Project: Loads the focused image to the project and closes the form.

Save as: Opens a file save dialog and saves the currently visible image to the selected path.

Sizemode: Sets the size mode for the currently visible image. The size of the image will not be changed!

RotateFlip: For rotating and flipping the currently visible image. If applied on the project, the initial image will be taken and not the rotated. If you save the image, the rotated image will be saved.

Load Paintform: Opens a form where you can paint basic objects directly into the image.

9.2.3 Menu View

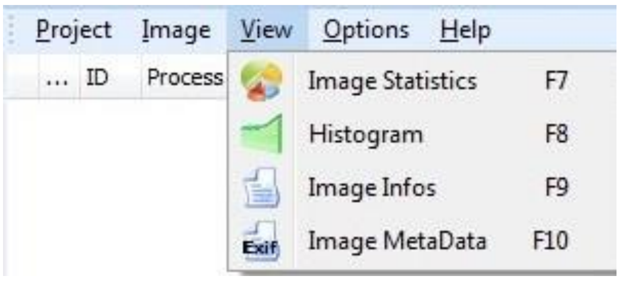
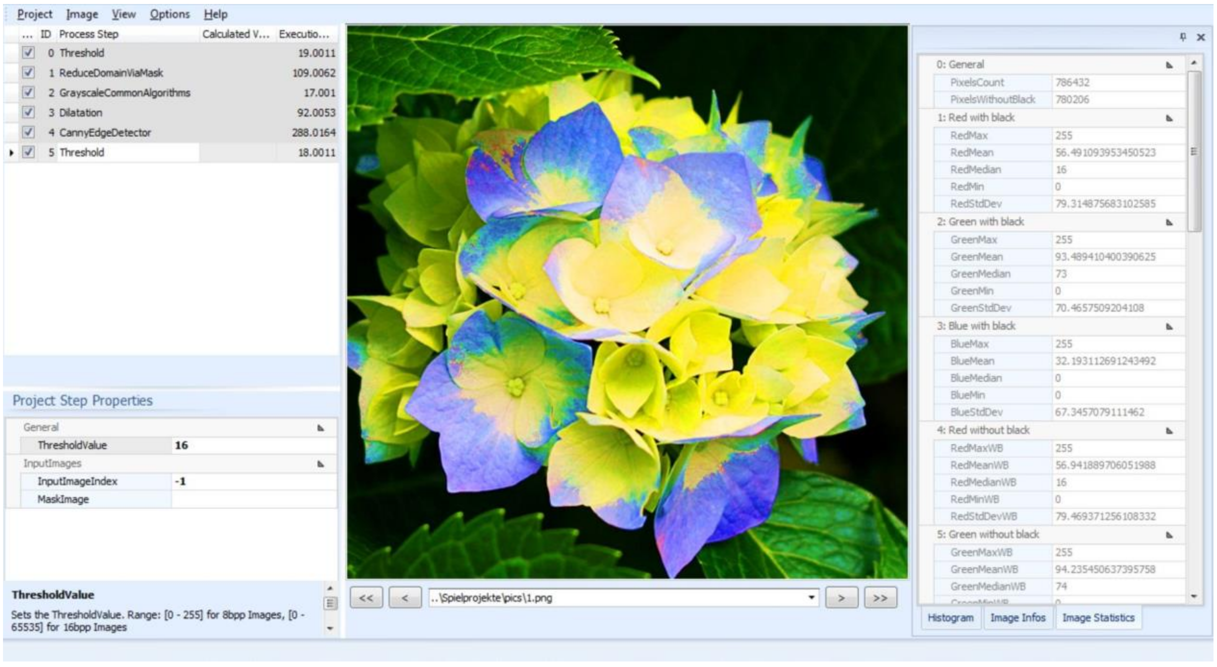


Image Statistics: Opens a dock-window to the right or ads a new tab to this window if it is already open. Some statistics of the currently visible image are shown. This is a great tool for analysing images.



...	ID	Process Step	Calculated V...	Executio...
✓	0	Threshold		19.0011
✓	1	ReduceDomainViaMask		109.0062
✓	2	GrayscaleCommonAlgorithms		17.001
✓	3	Dilatation		92.0053
✓	4	CannyEdgeDetector		288.0164
✓	5	Threshold		18.0011

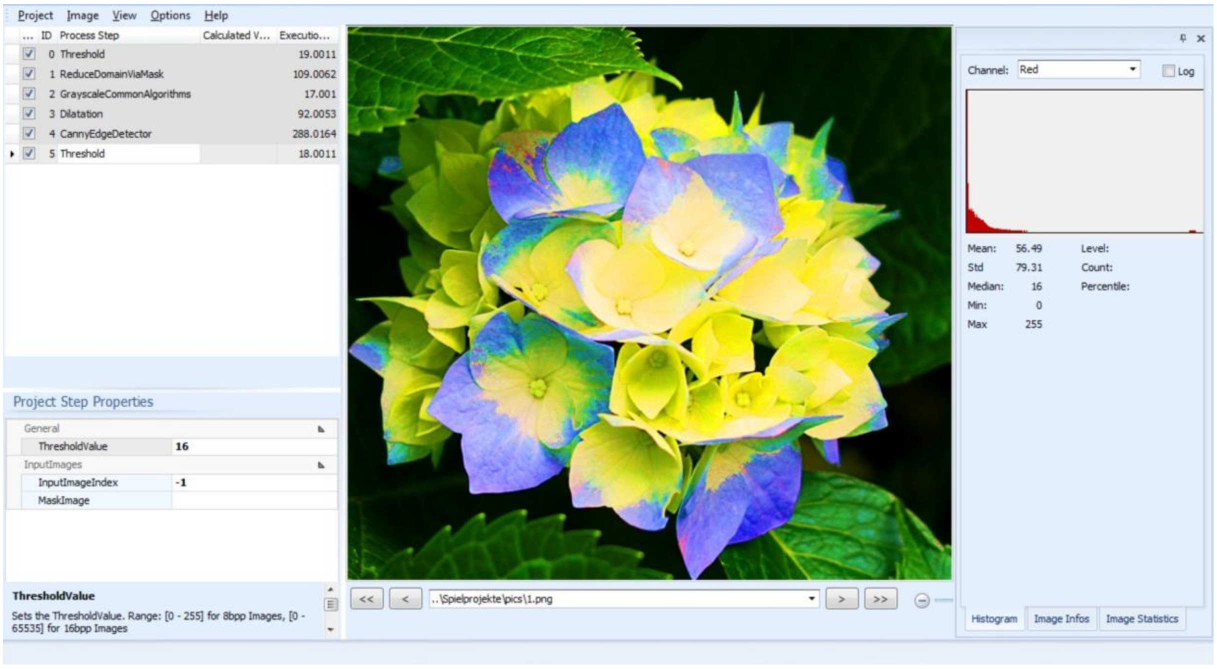
General	
ThresholdValue	16
InputImages	
InputImageIndex	-1
MaskImage	

ThresholdValue

Sets the ThresholdValue. Range: [0 - 255] for 8bpp Images, [0 - 65535] for 16bpp Images

0: General	
PixelsCount	786432
PixelsWithoutBlack	780206
1: Red with black	
RedMax	255
RedMean	56.491093953450523
RedMedian	16
RedMin	0
RedStdDev	79.314875683102585
2: Green with black	
GreenMax	255
GreenMean	93.489410400390625
GreenMedian	73
GreenMin	0
GreenStdDev	70.4657509204108
3: Blue with black	
BlueMax	255
BlueMean	32.193112691243492
BlueMedian	0
BlueMin	0
BlueStdDev	67.3457079111462
4: Red without black	
RedMaxWB	255
RedMeanWB	56.941889706051988
RedMedianWB	16
RedMinWB	0
RedStdDevWB	79.469371256108332
5: Green without black	
GreenMaxWB	255
GreenMeanWB	94.235450637395758
GreenMedianWB	74

Histogram: Opens a dock-window to the right or ads a new tab to this window if it is already open. Here the histogram of the currently visible image is shown.



...	ID	Process Step	Calculated V...	Executio...
✓	0	Threshold		19.0011
✓	1	ReduceDomainViaMask		109.0062
✓	2	GrayscaleCommonAlgorithms		17.001
✓	3	Dilatation		92.0053
✓	4	CannyEdgeDetector		288.0164
✓	5	Threshold		18.0011

General	
ThresholdValue	16
InputImages	
InputImageIndex	-1
MaskImage	

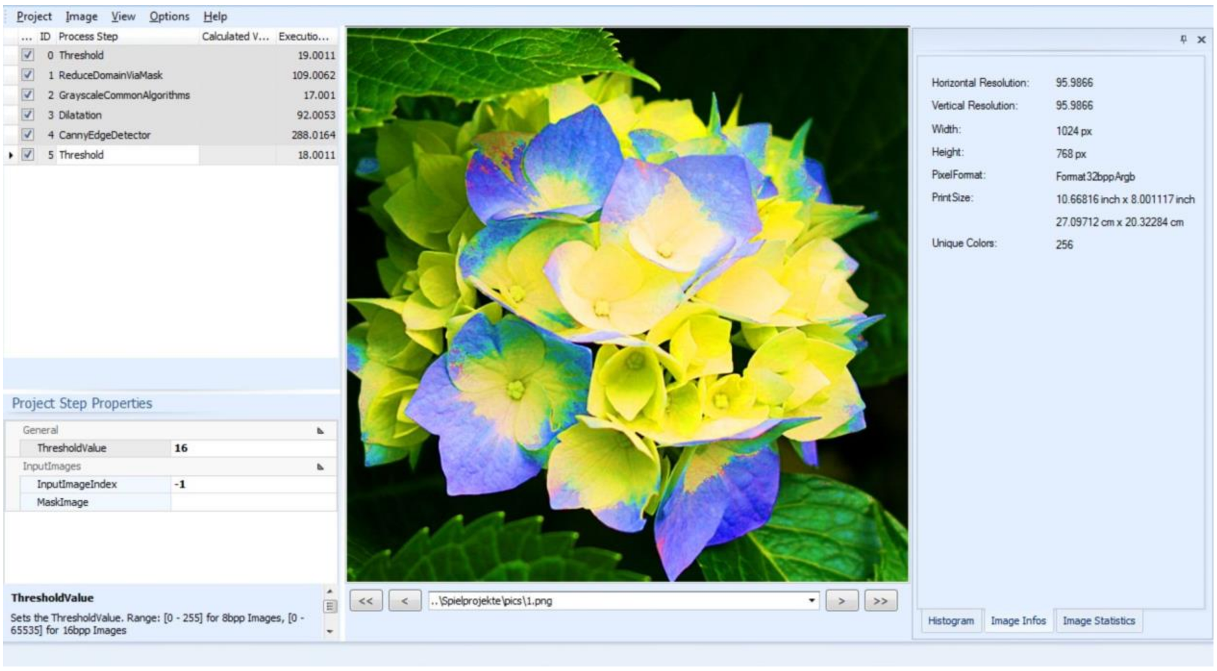
ThresholdValue

Sets the ThresholdValue. Range: [0 - 255] for 8bpp Images, [0 - 65535] for 16bpp Images

Channel: Red ☐ Log

Mean:	56.49	Level:	
Std	79.31	Count:	
Median:	16	Percentile:	
Min:	0		
Max	255		

Image Infos: Opens a dock-window to the right or ads a new tab to this window if it is already open. Some additional image information is displayed, like colour depth, unique colours, resolution and so on.



...	ID	Process Step	Calculated V...	Executio...
✓	0	Threshold		19.0011
✓	1	ReduceDomainViaMask		109.0062
✓	2	GrayscaleCommonAlgorithms		17.001
✓	3	Dilatation		92.0053
✓	4	CannyEdgeDetector		288.0164
✓	5	Threshold		18.0011

Project Step Properties

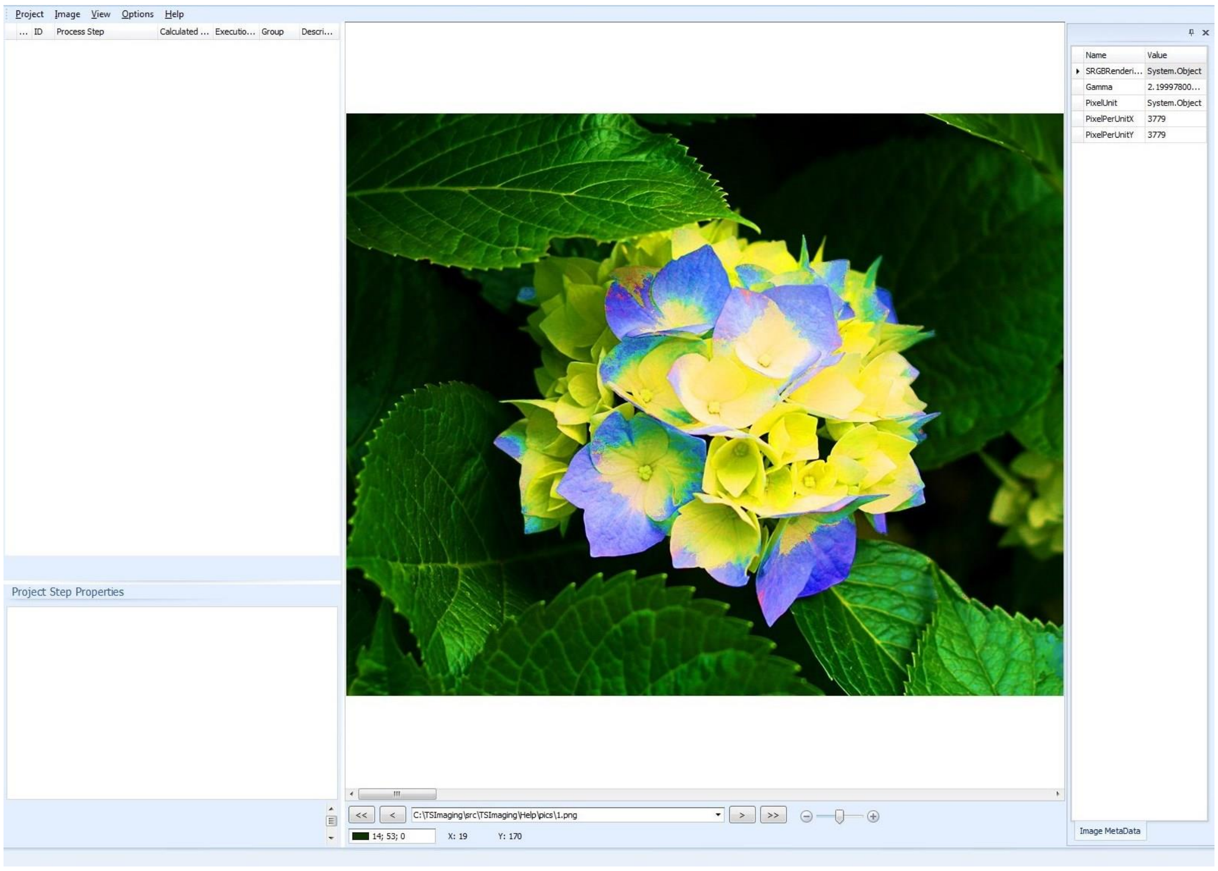
General	
ThresholdValue	16
InputImages	
InputImageIndex	-1
MaskImage	

ThresholdValue
Sets the ThresholdValue. Range: [0 - 255] for 8bpp Images, [0 - 65535] for 16bpp Images

..\Spielprojekte\pics\1.png

Horizontal Resolution:	95.9866
Vertical Resolution:	95.9866
Width:	1024 px
Height:	768 px
PixelFormat:	Format32bppArgb
PrintSize:	10.66816 inch x 8.001117 inch
	27.09712 cm x 20.32284 cm
Unique Colors:	256

Image MetaData: Opens a dock-window to the right or ads a new tab to this window if it is already open. The available meta data of the image is displayed.



...	ID	Process Step	Calculated ...	Executio...	Group	Descri...

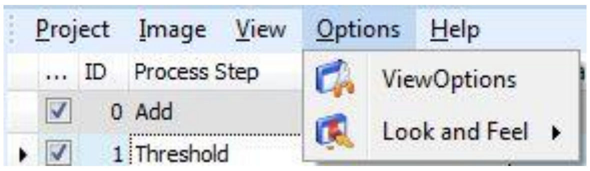
Project Step Properties

C:\TSImaging\src\TSImaging\Help\pics\1.png

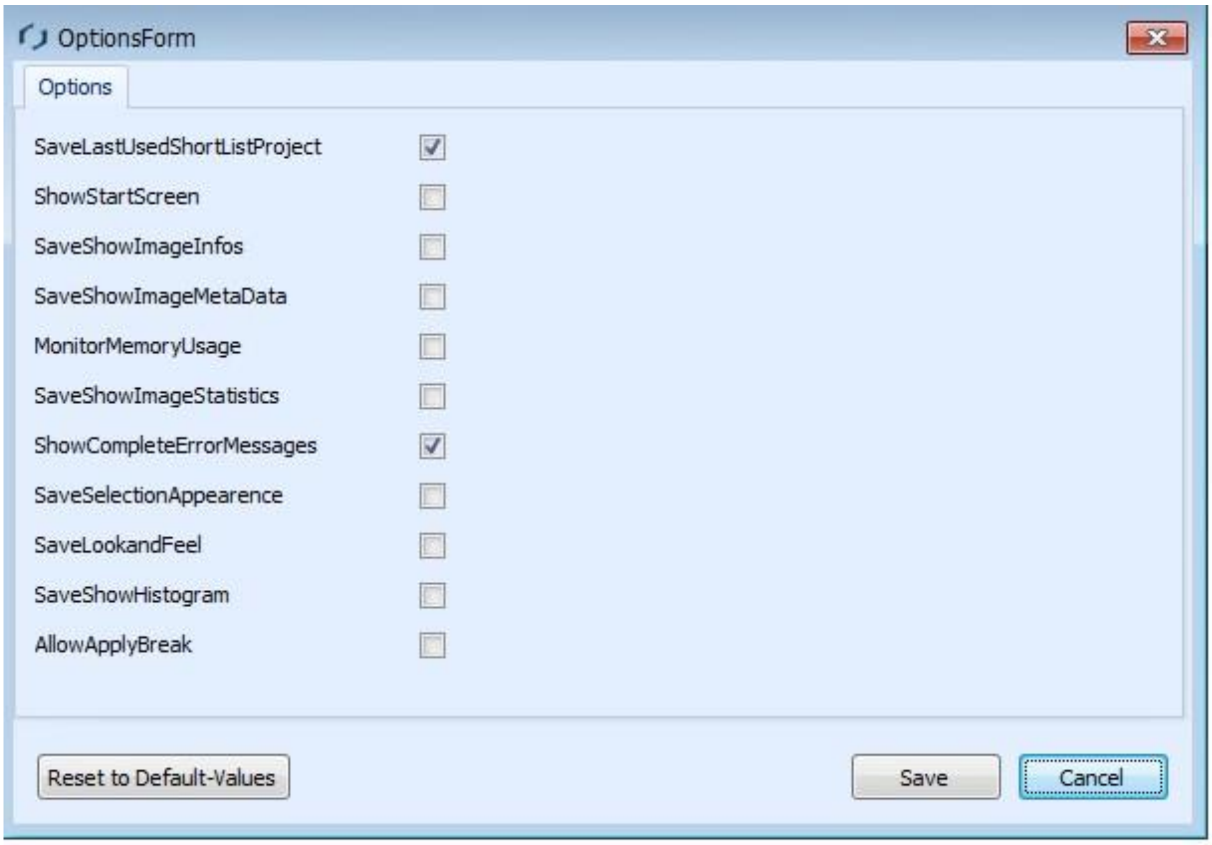
14; 53; 0 X: 19 Y: 170

Name	Value
SRGBRenderi...	System.Object
Gamma	2.19997800...
PixelUnit	System.Object
PixelPerUnitX	3779
PixelPerUnitY	3779

9.2.4 Menu Options



View Options: Opens a form to define some general options concerning SchiferVisionProjectDeveloper.



SaveLastUsedShortListProject	✓
ShowStartScreen	
SaveShowImageInfos	
SaveShowImageMetaData	
MonitorMemoryUsage	
SaveShowImageStatistics	
ShowCompleteErrorMessages	✓
SaveSelectionAppearence	
SaveLookandFeel	
SaveShowHistogram	
AllowApplyBreak	

SaveShowHistogram: If checked and the histogram dock-window is open while closing SchiferVisionProjectDeveloper, the histogram dock-window will be opened after the next start of SchiferVisionProjectDeveloper.

SaveLastUsedShortListProject: If you used a ShortListProject, this selection will be saved and available after restart.

SaveShowImageInfos: see SaveShowHistogram.

SaveShowImageMetaData: see SaveShowHistogram.

SaveLookAndFeel: Like SaveShowHistogram, but useful for the look and feel of SchiferVisionProjectDeveloper. Compare to section Look and Feel.

SaveShowImageStatistics: see SaveShowHistogram.

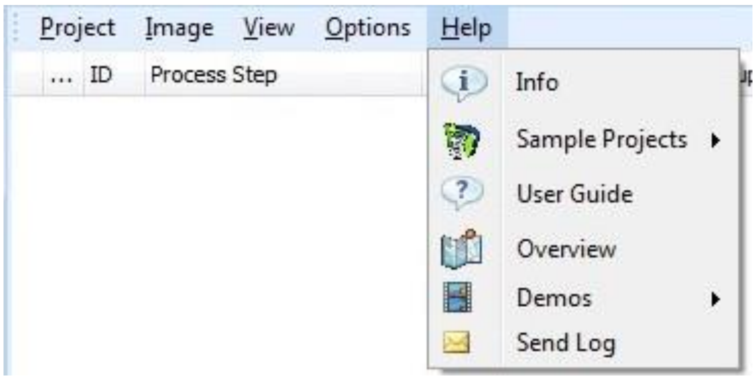
SaveSelectionAppearence: If you changed the appearance of the selection, it will be saved and it will look the same after a restart.

ShowCompleteErrorMessages: In case of an error, a window with a complete error message will be opened. Here you have the possibility to send the error to us. If not set, you will only get a small MessageBox. Compare section Configure SchiferVisionEgine->TraceLevel.

AllowApplyBreak: If set, it is possible to break the application by pressing “Esc”. Compare section Configure SchiferVisionEgine->AllowApplyBreak.

MonitorMemoryUsage: If you activate this, there will be an additional column in the project steps grid where you can see the memory usage (.Net working set) of each step in MB. This is useful for debugging memory problems.

9.2.5 Menu Help



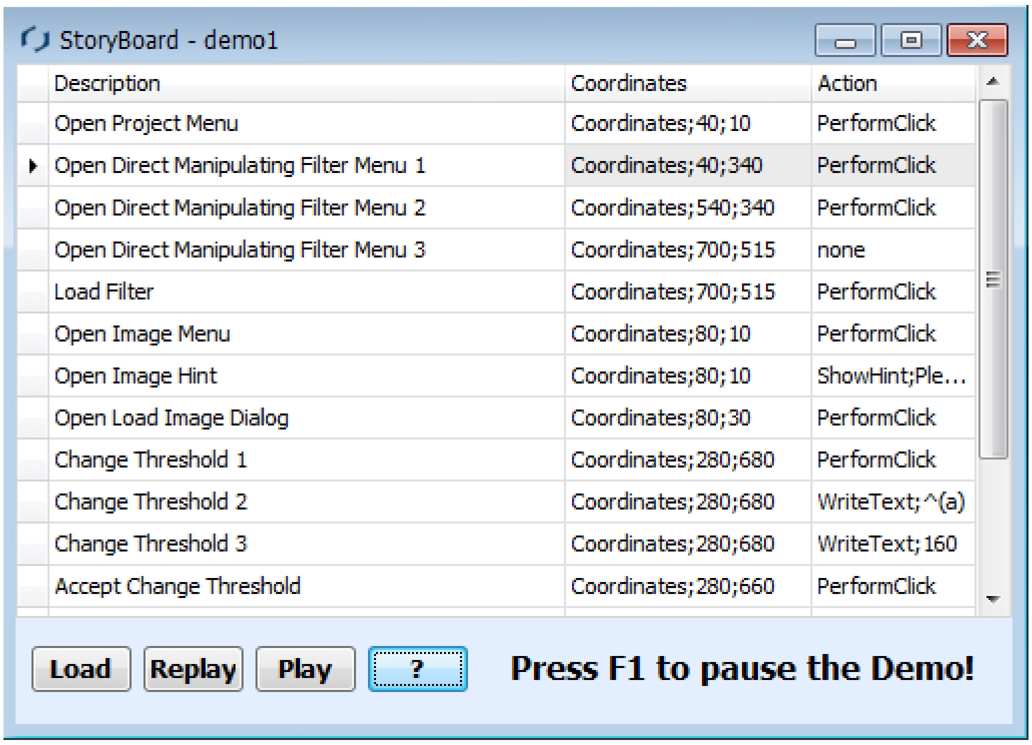
Info: Shows an info form of the application.

Sample Projects: Opens a sample project. Compare section Running Sample Projects.

User Guide: Opens this document.

Overview: Opens a document which gives an overview of the possibilities of SchiferVision

Demos: Opens the following form and plays a demo. The demo moves the mouse automatically and shows how several techniques are used.


StoryBoard - demo1

	Description	Coordinates	Action
	Open Project Menu	Coordinates;40;10	PerformClick
▸	Open Direct Manipulating Filter Menu 1	Coordinates;40;340	PerformClick
	Open Direct Manipulating Filter Menu 2	Coordinates;540;340	PerformClick
	Open Direct Manipulating Filter Menu 3	Coordinates;700;515	none
	Load Filter	Coordinates;700;515	PerformClick
	Open Image Menu	Coordinates;80;10	PerformClick
	Open Image Hint	Coordinates;80;10	ShowHint;Ple…
	Open Load Image Dialog	Coordinates;80;30	PerformClick
	Change Threshold 1	Coordinates;280;680	PerformClick
	Change Threshold 2	Coordinates;280;680	WriteText;^(a)
	Change Threshold 3	Coordinates;280;680	WriteText;160
	Accept Change Threshold	Coordinates;280;660	PerformClick

It is important that the SchiferVisionProjectDeveloper is visible. As the mouse moves automatically, you have to press F1 to pause the demo. The demos only work with a screen resolution of 1400 * 1000 pixels and above at 96 DPI.

Load: Opens a FileBrowseDialog to open a demo file (only used for loading demos which are not included in the initial shipment)

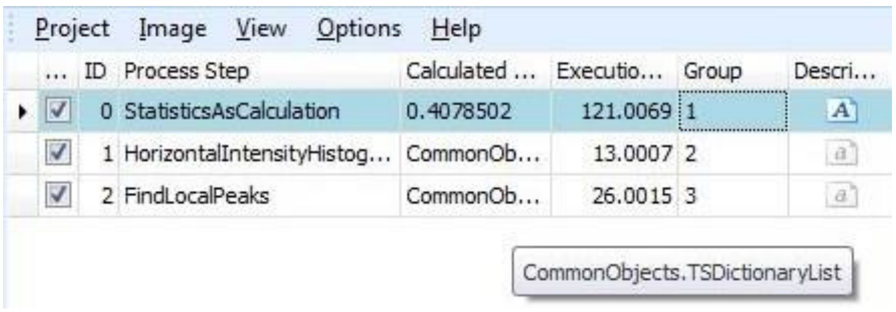
Replay: Replays the whole demo.

Play: Starts the demo at the focused row of the grid.

?: Shows a description of the demo.

Send Log: If you have turned on ShowCompleteErrorMessages (Compare Menu Options -> View Options) a log file will be generated. Here you can send the log file to us.

9.3 Project Steps Grid:



	…	ID	Process Step	Calculated …	Executio…	Group	Descri…
▸	☑	0	StatisticsAsCalculation	0.4078502	121.0069	1	
	☑	1	HorizontalIntensityHistog…	CommonOb…	13.0007	2	
	☑	2	FindLocalPeaks	CommonOb…	26.0015	3	

CommonObjects.TSDictionaryList

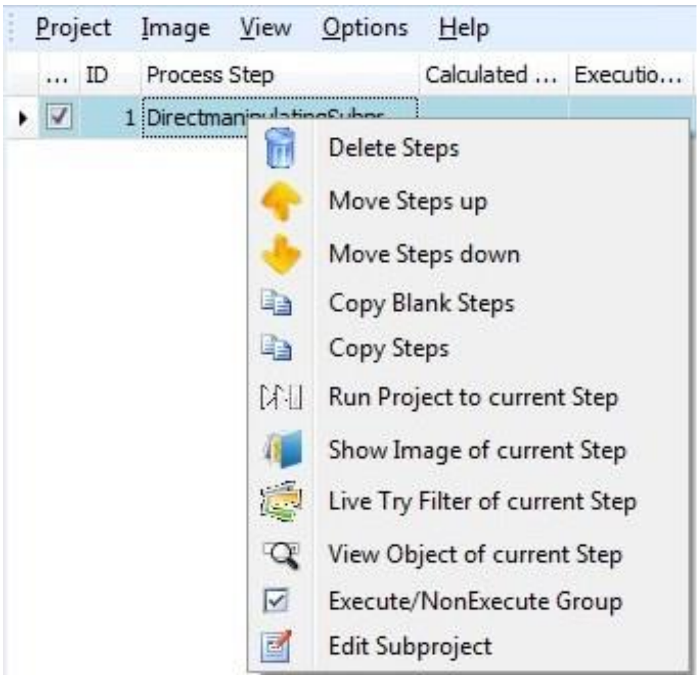
Here we can see a grid with all filter steps of the project. The columns are as follows:

- Execute/Nonexecute Step: A checkbox which determines whether this specific step should be executed or not. This offers a comfortable way of testing and prototyping. You may delete all the steps you do not need in the final project.

- ID: The unique ID of the filter step
- Process Step: The name of the filter
- Calculated Value: Here you can see the output value of calculating steps. If it is a simple type, you will see the real value. If it is an advanced type, you will see the type of output. Hint: Some direct manipulating steps may output a calculated value additionally to manipulating the image. You can say that they serve as both filter types. Controlling steps display the following things here:
    - ConditionalSkipFilter step: true or false
    - ConditionalJumpToFilterID: the JumpToFilterID or -1 if the condition is false, or nothing if JumpToFilterID is 999999 to end the execution of the project
    - LoopOverObjectUntilFilterID: the current object of the loop
    - BreakLoop and ContinueLoop: nothing
- Execution Time: The execution time of this filter step. Hint: LoopOverObjectUntilFilterID shows the total execution time of the entire loop.
- Group: You can input a free text to group some of your filter steps. You can then select and deselect the “Execute/NonExecute Group” checkbox for an entire group. Compare Project Step Context Menu for this feature.
- Description: Here you can enter a description for the filter step. This is without further use and just information for you.

9.3.1 Project Step Context Menu

If you right-click on a certain filter step in the project step grid, you will open the following context menu:



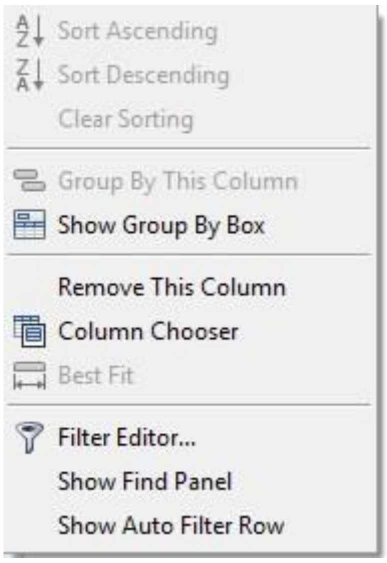
Here you can find some very useful features, which are as follows:

- Delete Steps: Deletes the selected steps
- Move Steps up: Moves the selected steps one position up. That means they will be executed earlier. Note: You can drag the steps with the mouse as well. The step will be inserted below the step where you dropped it. That means you cannot drag it to

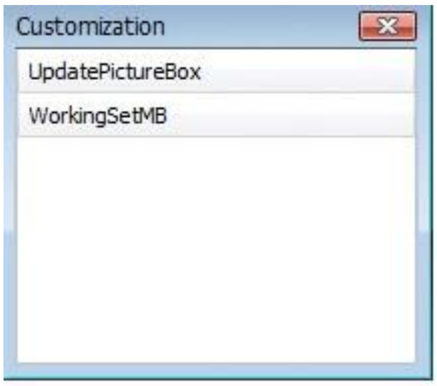
the first position, only to the second and then you have to use this button to move it to the first position.

- Move Steps Down: Moves steps down. See Move Steps up for information about dragging the step.
- Copy Blank Steps: Copies the selected steps and adds the new steps below the copied step. Note: The values of the properties of the new steps will be reset to default.
- Copy Steps: Copies the selected steps and adds the new steps below the copied step.
- Run Project to current Step: Runs the project only until the current step
- Show Image of current Step: Shows the output image of the current step, which is very useful for testing and prototyping. Only available for direct manipulating steps and after running the project
- Live Try Filter of Current Step: For many direct manipulating filters, a live try is available. That means you can comfortably test the effect of changing parameters of the certain filters on the images available. Take a look at section Live Try Filter.
- View Object of current Step: For many advanced output types from calculating filters a viewer is available. Take a look at section View Object.
- Execute/NonExecute Group: Sets the Execute/Nonexecute Step Checkbox for the entire group of the selected row. So for rows which have the same group (same text in group field) the Execute/Nonexecute Step Checkbox will be checked if this Checkbox of the selected row is currently unchecked and vice versa.
- Edit Subproject: If your project contains subprojects, you may want to edit them. Here you can load the subproject into a new SchiferVisionProjectDeveloper Window. All input variables will be passed to the subproject (compare section Using subprojects with InputVariables). A great tool for debugging subprojects.

9.3.2 Grid Context Menu



If you right-click on one of column headers, this context menu opens. There you can find several functions concerning the grid itself. One important point is the Column Chooser:

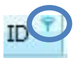


This serves as container for hidden columns. You can drag and drop them to the grid header to show a specific column and vice versa.

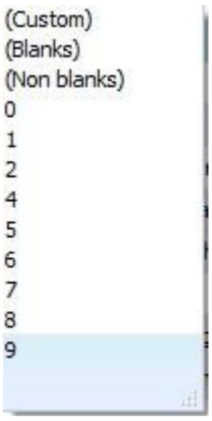
UpdatePictureBox: This column is important if you do not want that a filter step updates the image box. This is mainly used in debug mode (compare execute next filter step) for presentation projects for “smoother” running.

WorkingSetMB: Shows the used MB of the working set. This is mainly for monitoring the memory usage of each filter step (Compare Menu Options -> Monitor Memory Usage)

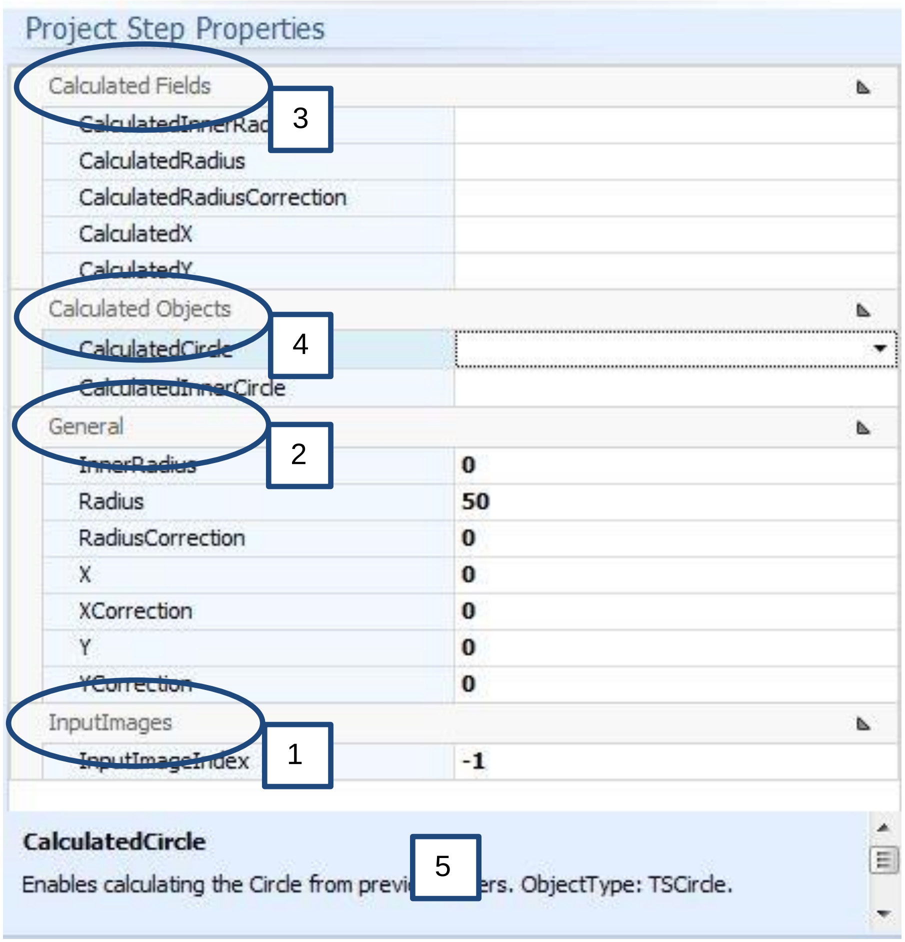
If you move the mouse over one of the column headers of the grid, you will see the following icon:



If you press this icon, you will get the following drop down menu where you can filter the grid. This is useful if you are searching for a special project filter step, especially in large projects which are not organised by the step ID.



9.4 Project Steps Properties


Project Step Properties

Calculated Fields	
CalculatedInnerRad	
CalculatedRadius	
CalculatedRadiusCorrection	
CalculatedX	
CalculatedY	
Calculated Objects	
CalculatedCircle	
CalculatedInnerCircle	
General	
InnerRadius	0
Radius	50
RadiusCorrection	0
X	0
XCorrection	0
Y	0
YCorrection	0
InputImages	
InputImageIndex	-1

CalculatedCircle
Enables calculating the Circle from previ ers. ObjectType: TSCircle.

9.4.1 Section Input Images

Here you can set the input image for the specific filter step. This may not be needed for several steps which work with advanced objects as input.

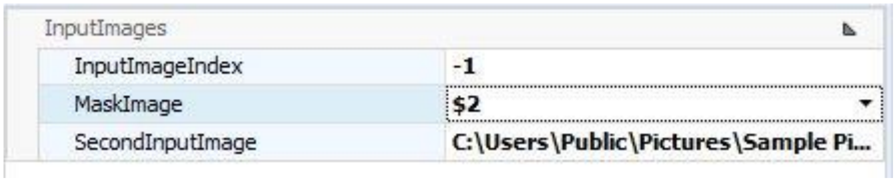
However there are three ways of specifying the input image:

-1: Takes the last available image. This may be the initial image for the first filter step or the last output of a direct manipulating filter step.

-2: Always takes the initial image.

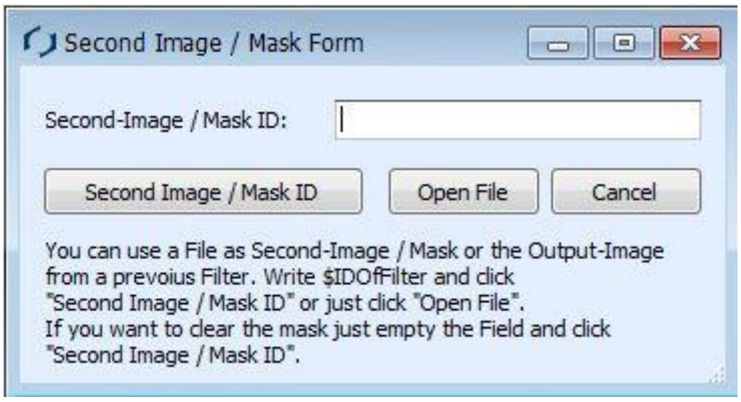
IDOfFilter: Takes the image which will be the output of the filter with the corresponding ID. This has to be a direct manipulating step and it has to be executed first. Otherwise you will get an error.

There are filters which take other images as input, e.g. two source filter for image merging and so on or a mask image. The mask image excludes several parts of an image from processing (compare section Filter Steps – General).



InputImages	
InputImageIndex	-1
MaskImage	$2
SecondInputImage	C:\Users\Public\Pictures\Sample Pi...

To input such images, there are two possibilities. Both work via the following mask:



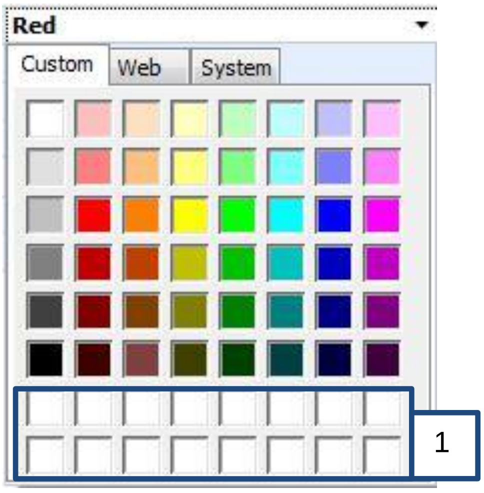
You can use a File as Second-Image / Mask or the Output-Image from a prevoius Filter. Write $IDOfFilter and click "Second Image / Mask ID" or just click "Open File".
If you want to clear the mask just empty the Field and click "Second Image / Mask ID".

Here you can enter a filter ID, e.g. $4 for the output of the filter with the ID 4. This has to be a direct manipulating step and it has to be executed first. Otherwise you will get an error. Or you click on Open File and insert a path to an image. You can see these two possibilities in the image above.

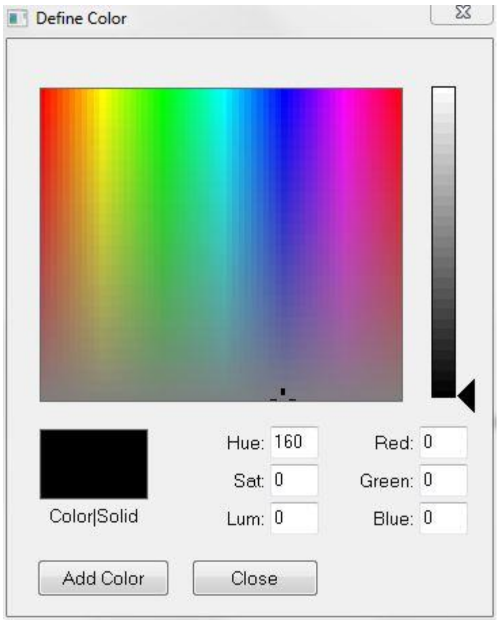
9.4.2 Section General

Here you can input values directly without calculating them. The allowed values can be seen in the information field below (compare section Info Field).

Some of these values are special ones with line colours. Here you can enter the rgb-value by writing r; g; b; or you open the following editor by clicking on the triangle on the far right side:



Here you can choose a colour or you right-click on one of the custom colour fields (see 1). This opens a further editor:



Hue: 160
Sat: 0
Lum: 0
Red: 0
Green: 0
Blue: 0

Here you can define a custom colour and add it to the list.

Some values may be paths and accept a string or open a Folder Browse / File Browse dialog.

9.4.3 Section Calculated Fields

In this section you can enter the output of calculating filter steps if this is a simple type. There are two possibilities:

If you want to input the value without manipulation, you can write: $4, whereas 4 is the ID of the filter step which produces the output. Make sure that this filter is executed first in your filter chain.

You can use Reverse Polish Notation[2] (RPN) Infix Notation to calculate the input value. E.g. you can write: $4 / 2 which inputs the output from step 4 divided by 2.

You can even use this method to calculate the value from multiple other filter steps, e.g.: $4 – ($2 / 2) or: $4 < $3 ? $3 : $4 …

The only thing you have to be aware of is the notation with blanks. Before and after each $IDOfInputFilter you have to put a blank, otherwise it will result in an error.

The simple object type can be seen in the corresponding property of the section General. It will be cast to this anyway. If the cast is not possible, it will result in an error.

List of possible operators:

Math operators: +, -, *, /, %

Compare operators: ==, !=, >, <, >=, <=

Bitwise operators: &, |, ^

[2] Compare http://en.wikipedia.org/wiki/Reverse_Polish_notation

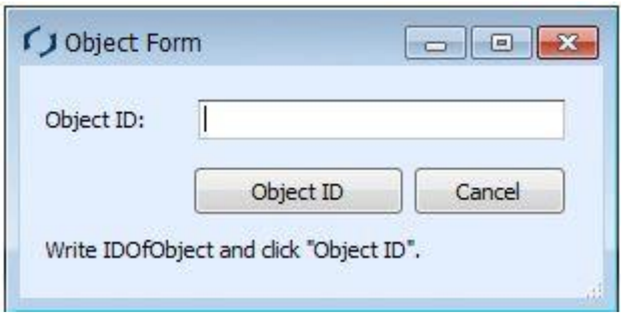
Logical operators: &&, ||

If you need more complex mathematical operators, you may use the filter MathOperator, which offers operators like sqrt, pow, sin, cos, tan, log and so on. This filter uses RPN as described above but uses a Postfix Notation. E.g. you can write: $4 sqrt 2 * which calculates the square root from the output from step 4 and multiplies it by 2.

9.4.4 Section Calculated Objects

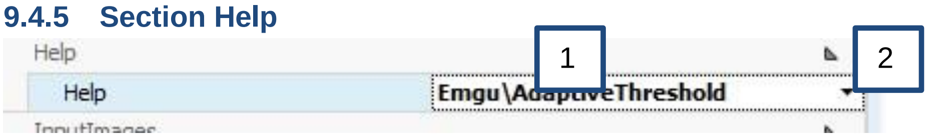
In this section you can enter the output of calculating filter steps if this is an advanced type. It works rather straight forward via a small mask (look below) where you have to enter the ID of the outputting filter. The allowed advanced type can be seen in the corresponding information field.

Hint: If the advanced type is a TSImageList, you can use the ImageSelection (Compare Menu Image) by using an ID of -1.



Write IDOfObject and click "Object ID".

9.4.5 Section Help



Help: Emgu\AdaptiveThreshold

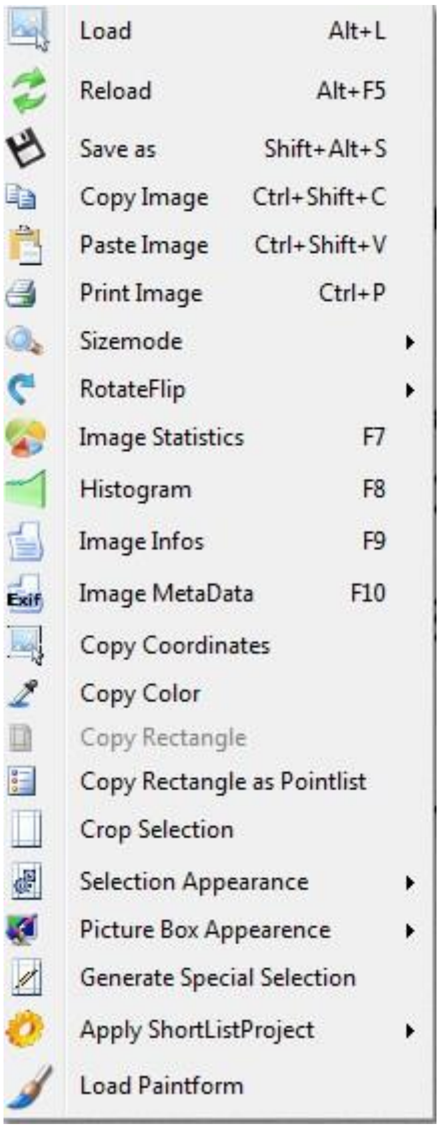
For many filters there is help available. First click into the field (see 1) and then click on the down arrow far right (see 2). This will open a pop-up with more information about the filter.

9.4.6 Info Field

If you click into the row of a filter property, you will see information about this property in an info field below (number 5 in the screenshot in section Project Step Properties).

9.5 Image Context Menu

Opened with a right mouse click on the image.



Methods for the sample image of the project

Load: Opens a file browse dialog and loads the selected image to the picture box. This image serves as an initial image to which the project will be applied.

Reload: Reloads the image. E.g. after applying the project you want the original image. Only works if you inserted the image via Load or pasting an image path and not by pasting a bitmap.

Save as: Opens a file save dialog and saves the currently visible image to the selected path.

Copy Image: Copies the image to the clipboard.

Paste Image: Pastes an image from the clipboard.

Print Image: Prints the image.

Sizemode: Sets the size mode for the currently visible image. The size of the image will not be changed!

RotateFlip: For rotating and flipping the currently visible image. If you apply the project, the initial image will be taken and not the rotated. If you save the image, the rotated image will be saved.

Image Statistics, Histogram, Image Infos, Image MetaData: See section Menu View

Copy Coordinates: Copies the coordinates of the current mouse position to the clipboard as follows: X, Y

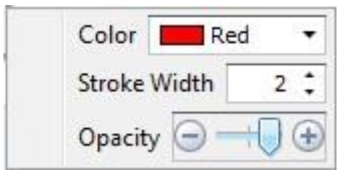
Copy Colour: Copies the colour of the current mouse position to the clipboard as follows: R; G; B

Copy Rectangle: If you press the left mouse button and move the mouse to another point in the image and then release the left mouse button, you can copy the properties of the so-generated rectangle to the clipboard. This will be visualized by a selection rectangle. The properties are as follows: StartPointXCoordinate, StartPointYCoordinate, Width, Height.

Copy Rectangle as PointList: Like Copy Rectangle, but generates a PointList (StartPoint; EndPoint) of the generated rectangle. Hint: You can use the filter GeneratePointList (Calculating Filters -> Common Objects) and then GenerateRectangle (Calculating Filters -> Common Objects) very fast that way.

Crop Selection: Crops the selection. Hold the left mouse button and move the mouse for a selection.

Selection Appearance: Opens the following submenu:

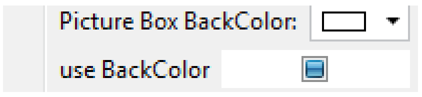


Color	🟥 Red
Stroke Width	2
Opacity	

This menu customizes the appearance of the selection. The selection generated via holding the left mouse button and moving the mouse helps to visualize the rectangle or the measured distance or the xDif and yDif of two points.

You can change the colour, the stroke width and the opacity of the selection here.

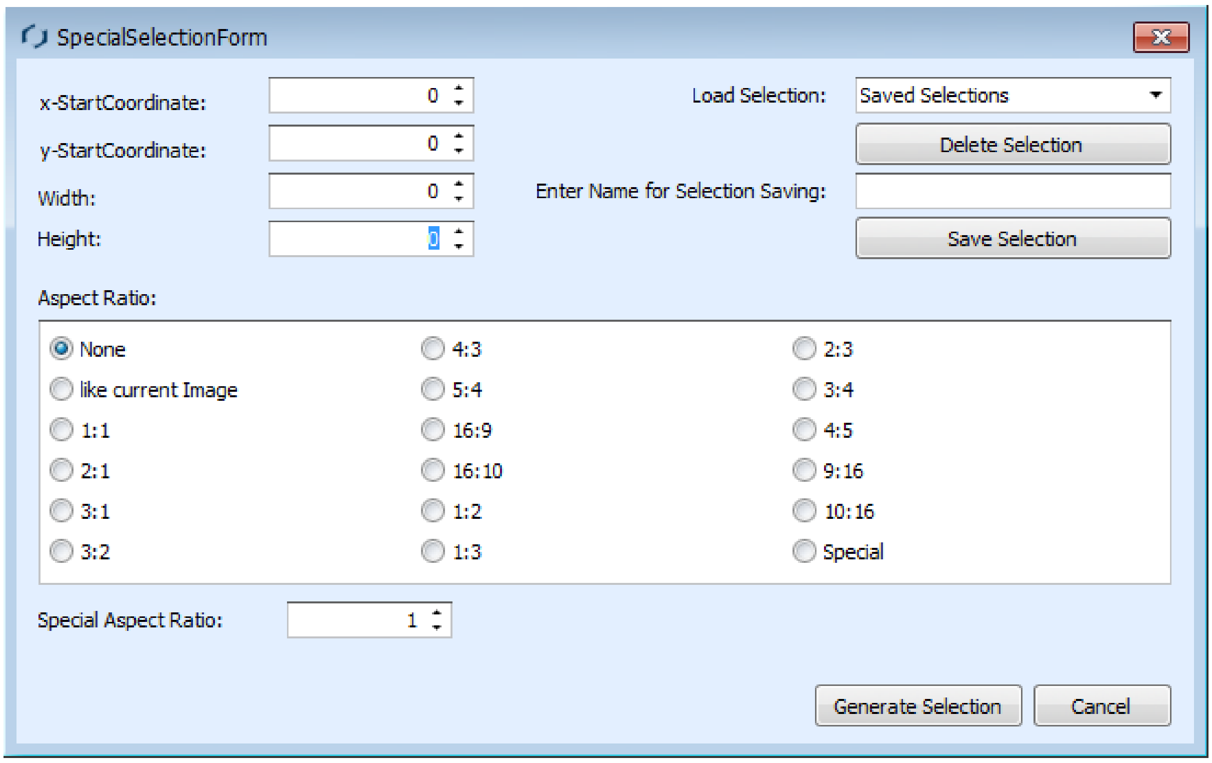
Picture Box Appearance: Opens the following submenu:



Picture Box BackColor:	⬜
use BackColor	

This menu customizes the appearance of the picture box. You can change the BackColor and decide whether to use it or not. This is useful for transparent images or images where the border has nearly the same colour as the picture box background.

Generate Special Selection: Opens the following mask:


SpecialSelectionForm

x-StartCoordinate:	0	Load Selection:	Saved Selections
y-StartCoordinate:	0		Delete Selection
Width:	0	Enter Name for Selection Saving:	
Height:	0		Save Selection

Aspect Ratio:

[✓ None]	4:3	2:3
like current Image	5:4	3:4
1:1	16:9	4:5
2:1	16:10	9:16
3:1	1:2	10:16
3:2	1:3	Special

Special Aspect Ratio: 1

Here you can manually enter a selection. E.g. x-StartCoordinate: 200, y-StartCoordinate: 200, Width: 500 and Height: 500 will draw a selection at the position 200, 200 which is 500 wide and 500 high. You can use an aspect ratio which corrects your width and height inputs to this aspect ratio. You can enter a name and save your selection for further use. All saved selections are available in the DropDown menu in the upper right corner. You can delete a saved selection by selecting it in the dropdown and pressing “Delete Selection”.

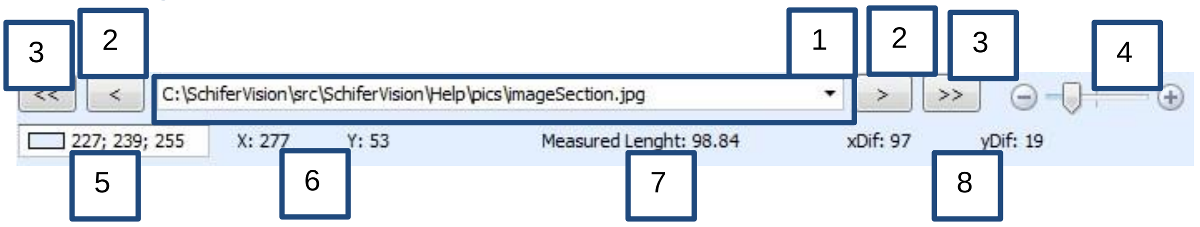
Hint: If you have generated a selection either by mouse or manually via “Generate Special Selection”, you can change this selection either again by the mouse or by using the arrow keys on your keyboard.

Apply ShortListProject: SchiferVision offers the possibility of generating predefined projects which can be applied to the current image with one single click. There is no need to load and apply it. This is very useful for many common actions you face during your work. Of course only projects which directly manipulate an image will work. To put a project to the ShortList you can use the built-in function “Save as ShortlistProject” in the project menu. By clicking one of the projects it will be directly applied to the current image. E.g. you know that you often need a higher contrast. So you can generate a project which does this and save it as described above. From now on you can apply it just by clicking it here.

Hint: Pressing F6 directly applies the last used ShortListProject.

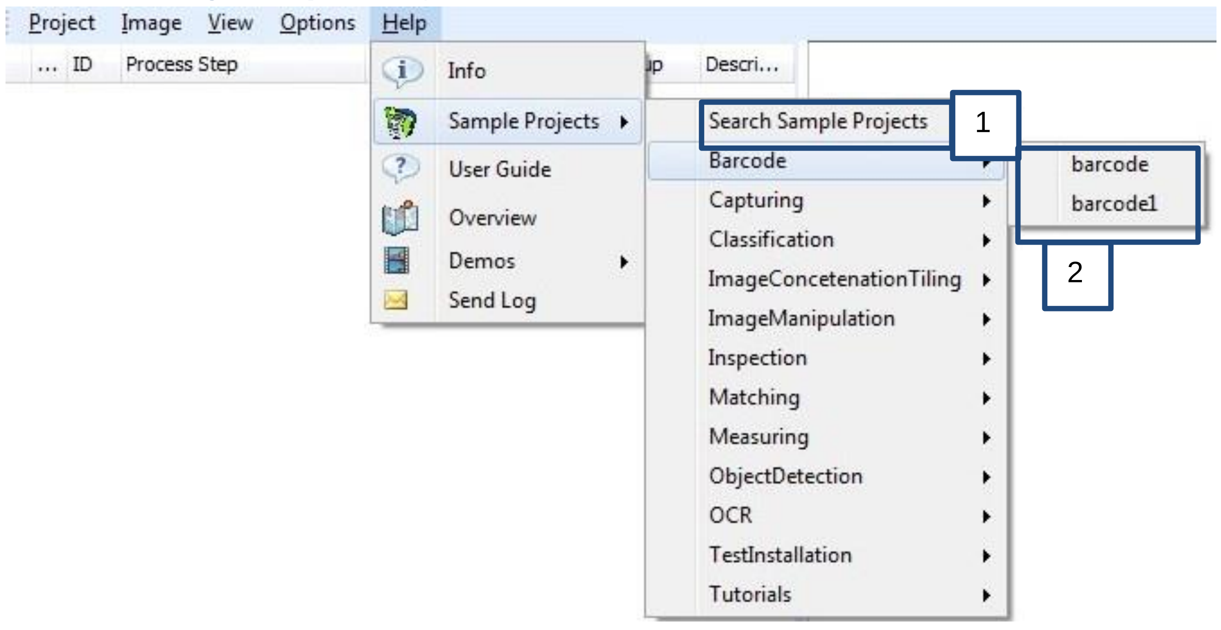
Load Paintform: Opens a form where you can paint basic objects directly on the image.

9.6 Image Section



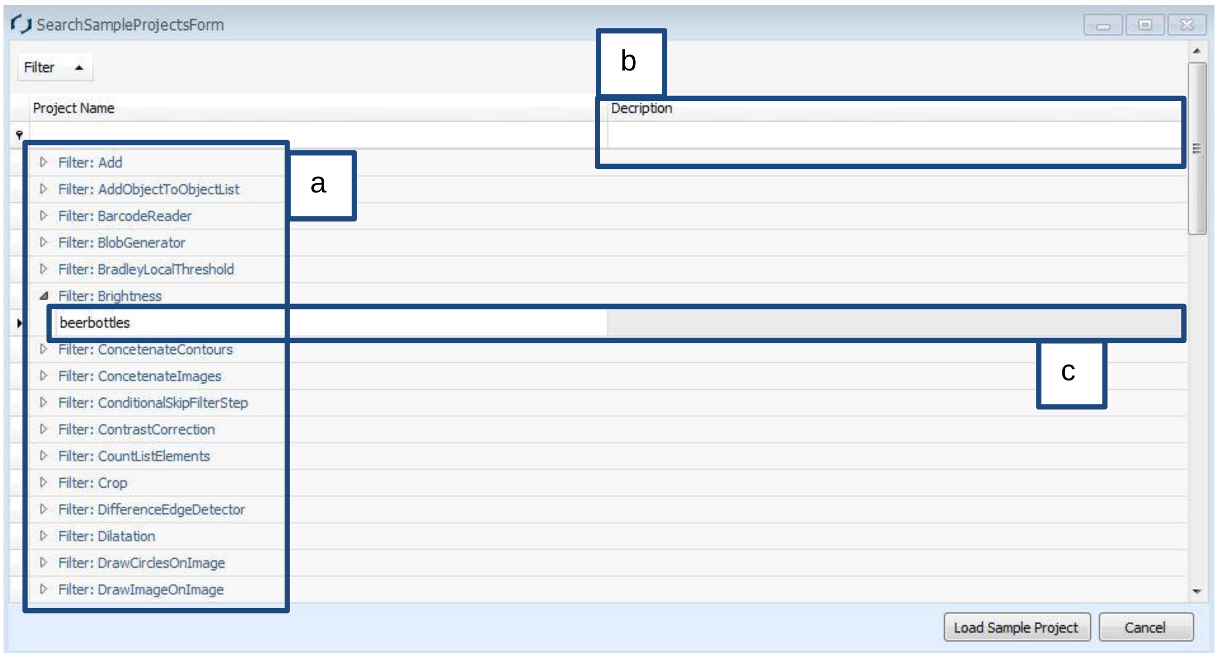
1. Shows the path of the image. Via Dropdown you can select all images which are in the same folder as the initially loaded image. If you use an image from the image selection (compare Menu Image -> Load Image Selection -> Load Image to Project), all paths of the images of the image selection will be available here.
2. Loads the previous or next image of the folder where the initially loaded image is located.
3. Loads the first or last image of the folder where the initially loaded image is located.
4. Zooms the image (Just the view in this control. The image itself will not be scaled). Only available if the size mode is set to Clip. Hint: A click with the right mouse button resets the zoom level to 100.
5. Shows the colour of the cursor position.
6. Shows the coordinates of the cursor position.
7. If you hold down the left mouse button and move the mouse, you can select a rectangle and measure the distance between the starting point (point where you first pressed the left mouse button) and your current cursor position. Here you see this distance.
8. Like 7., but does not show the length between the points but the difference of the x- and y-coordinates.

9.7 Running Sample Projects



There are two ways of running sample projects:

1) You can use the search form:



Filter: Add
Filter: AddObjectToObjectList
Filter: BarcodeReader
Filter: BlobGenerator
Filter: BradleyLocalThreshold
Filter: Brightness
beerbottles
Filter: ConcetenateContours
Filter: ConcetenateImages
Filter: ConditionalSkipFilterStep
Filter: ContrastCorrection
Filter: CountListElements
Filter: Crop
Filter: DifferenceEdgeDetector
Filter: Dilatation
Filter: DrawCirclesOnImage
Filter: DrawImageOnImage

This form initially shows all used filer types sorted alphabetically (compare a). If you want to know how to use a special filter type, you can see which sample projects use it. You can search in the description (hint: use %searchText to search within the text) as well (compare b). Then you can select the desired project and load it via the button “Load Sample Project” or by simply double-clicking on the desired project row.

2) You can directly choose one. They are grouped by varied imaging techniques.

After loading the project, a pdf-document describing the sample project will be loaded as well.

Now there are three ways of executing these sample projects.

1. Press F5 or choose “Apply Project to Image” from menu project: Applies the entire project to the image and shows the final result.
2. Press Ctrl+F11 or choose “Apply with Updating Step Images” from menu project: Applies the entire project to the image but shows every single image of direct manipulating steps.
3. Press F11 or choose “Execute next Step” from menu project: You can watch what happens step by step.

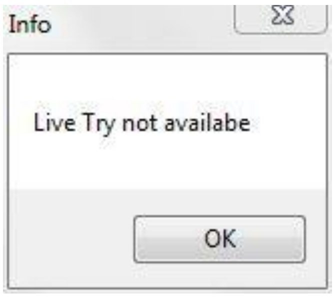
For some sample projects, e.g. capturing projects, the default cam of the computer is used. If such a camera does not exist, these projects return an error. In these projects it also does not make sense to apply them to the entire image with F5. You would only see the last captured image. So better use Ctrl+F11 for such projects.

Hint: For most sample project the size mode Squeeze works best. Go to Menu Image->Sizemode-> Squeeze, if not set.

Most sample projects offer more than one sample image, so navigate through the images to test them.

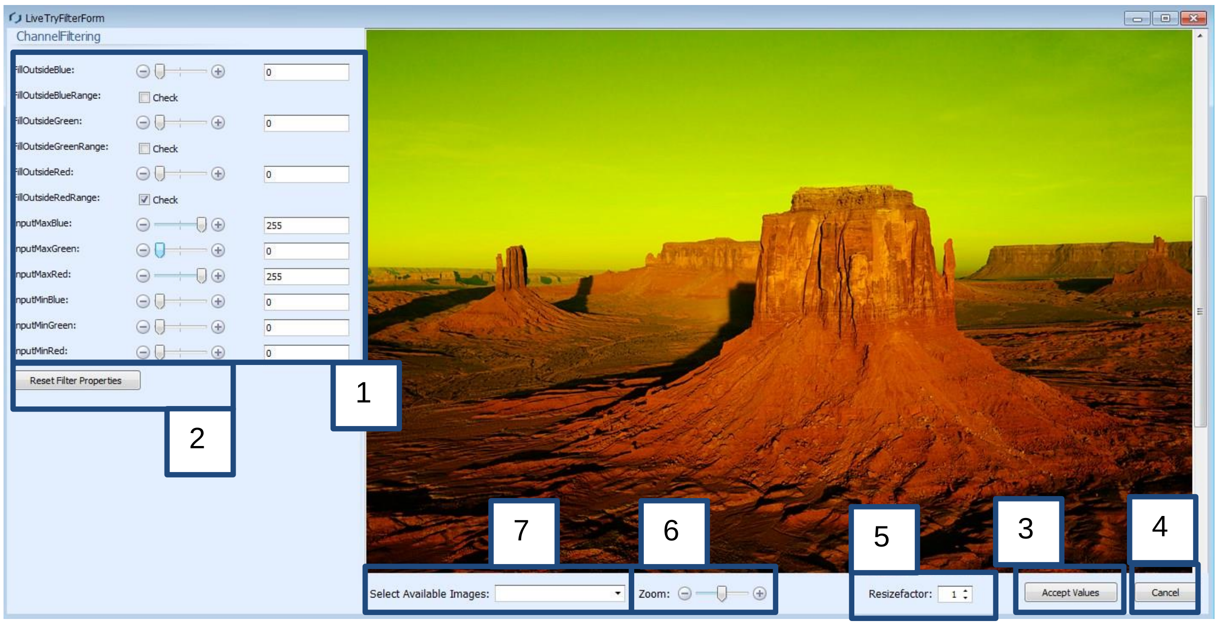
9.8 LiveTryFilter

A live try is available for many direct manipulating filters. That means you can comfortably test the effect of changing parameters of the specific filter on the images available. Just right-click on the filter step in the project step grid to show the project step context menu and click “Live try Filter of current Step”. If live try is available, the mask will be shown, otherwise you will get a message box:



Live Try not availabe

Live Try Mask:



illOutsideBlue:	⊖ ⊕	0
illOutsideBlueRange:	☐ Check	
illOutsideGreen:	⊖ ⊕	0
illOutsideGreenRange:	☐ Check	
illOutsideRed:	⊖ ⊕	0
illOutsideRedRange:	☑ Check	
nputMaxBlue:	⊖ ⊕	255
nputMaxGreen:	⊖ ⊕	0
nputMaxRed:	⊖ ⊕	255
nputMinBlue:	⊖ ⊕	0
nputMinGreen:	⊖ ⊕	0
nputMinRed:	⊖ ⊕	0

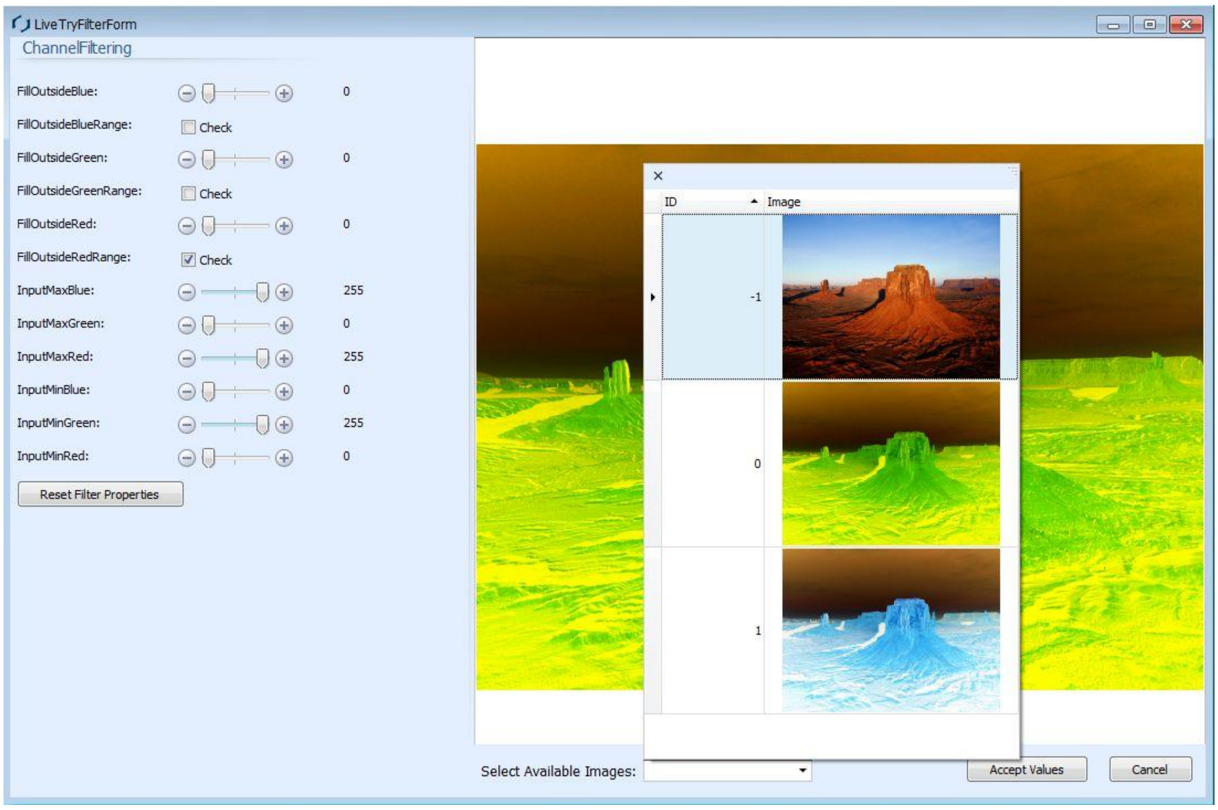
Select Available Images:

Zoom:

Resizefactor: 1

1. On the left side you can see every available property of the filter. You can play with the sliders or checkboxes and directly see the effect on the image.
2. The button “Reset Filter Properties” resets the values of the properties.
3. The button “Accept Values” closes the form and updates the values of the properties in your Project Steps Properties.
4. The button “Cancel” closes the form.
5. Resizes the current image. This is for larger images, where the filter application would be very slow. You can downsize these images here for a “smoother” LiveTry-performance. The resizing occurs only in the LiveTry.
6. Zooms the image (Just the view in this control. The image itself will not be scaled).
7. Select Available Images opens a drop down menu (see following mask) where all available images are shown. You then can select the desired image and test the filter as if this would be the input image for this filter. This is very useful for fast prototyping

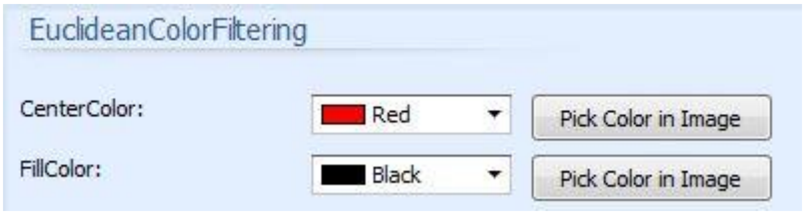
where you do not know how the filters interact best. Note: You have to run the project first to see all available images.


LiveTryFilterForm

ChannelFiltering

FillOutsideBlue:		0
FillOutsideBlueRange:	☐ Check	
FillOutsideGreen:		0
FillOutsideGreenRange:	☐ Check	
FillOutsideRed:		0
FillOutsideRedRange:	☑ Check	
InputMaxBlue:		255
InputMaxGreen:		0
InputMaxRed:		255
InputMinBlue:		0
InputMinGreen:		255
InputMinRed:		0

Hint: If the property is a colour, you will find the normal colour editor (compare section project steps properties / section general), on the right of this editor you will find a button “Pick Colour in Image”:


EuclideanColorFiltering

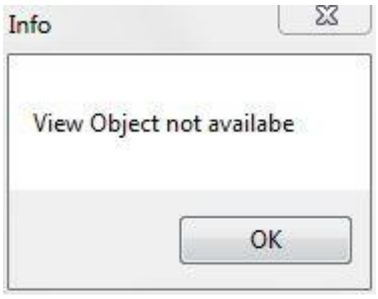
CenterColor:	🟥 Red	Pick Color in Image
FillColor:	⬛ Black	Pick Color in Image

If you click this button, you get an info message box and afterwards you can click directly into the image to pick the colour at the place you click. The colour editor will be automatically updated with the selected colour.

9.9 ShowObject

For many advanced output types of calculating filters a viewer is available. Imagine a list of points or a histogram. In the project step grid you only see the type but you cannot test it to see if it brings about the desired result. With the object viewer you have a powerful tool to visualize these types.

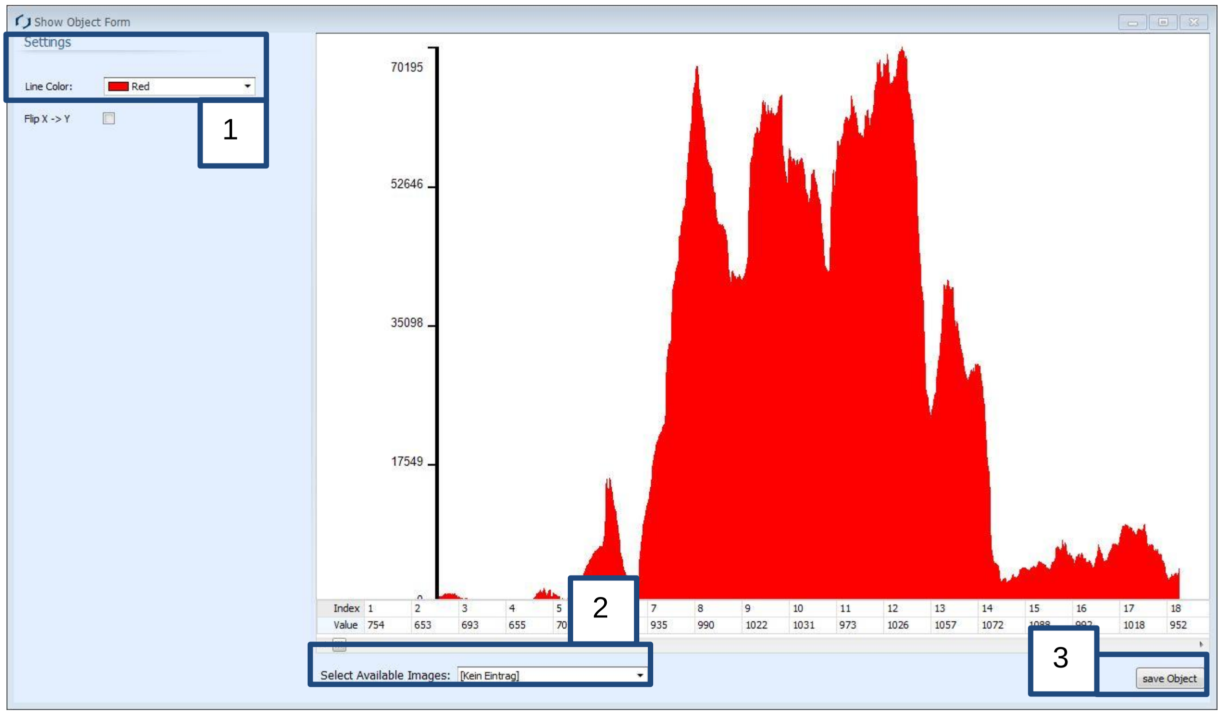
Just right-click on the filter step in the project step grid to show the project step context menu and click “View Object of current Step”. If View Object is available, the mask will be shown, otherwise you will get a message box:



Info

View Object not availabe

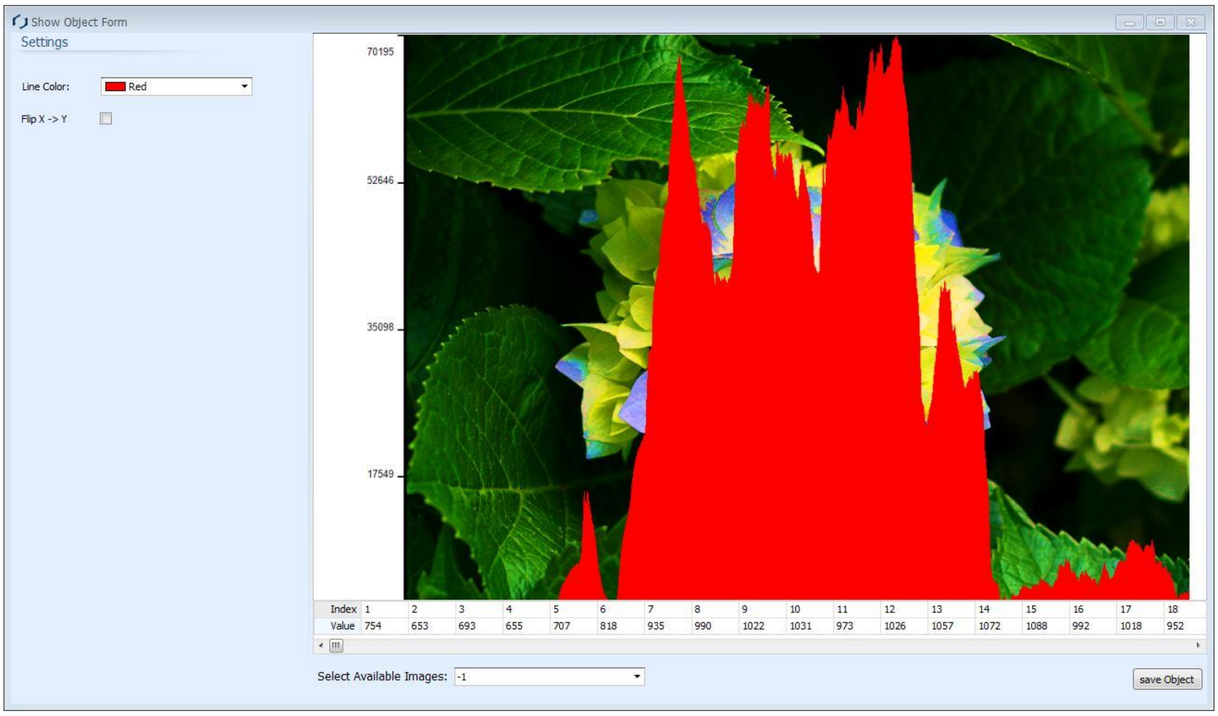
OK



Index	1	2	3	4	5	7	8	9	10	11	12	13	14	15	16	17	18
Value	754	653	693	655	70	935	990	1022	1031	973	1026	1057	1072	1088	992	1018	952

1. On the left side you can change the colour or sometimes you can use a look-up table (lut) to colour the objects. Here you can set the stroke width in certain cases or flip the x- and y- coordinate to visualize vertical histograms.
2. Select Available Images opens a drop down menu where all available images are shown. You can then select the desired image and show the object directly on this image. Note: You have to run the project first to see all available images.
3. Some objects can be saved and reloaded with the filter LoadObject for further use, e.g. a TSPointList as shape for shape based matching.

The following mask shows a horizontal intensity histogram directly on the image:



Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Value	754	653	693	655	707	818	935	990	1022	1031	973	1026	1057	1072	1088	992	1018	952

10 Filter Steps General

All filters serve as product on its own and they are not directly referenced by neither the SchiferVisionProjectDeveloper nor the SchiferVision Engine. This plugin function allows developers to develop their own filters and use it with SchiferVision without needing to compile SchiferVision.

However, the initially shipped filters share some common concepts for better understanding and a simpler use.

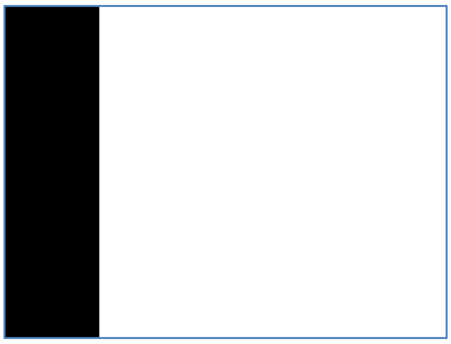
One of these concepts are masks.

10.1 Understanding masked filtering

Masks are black and white images, which define to which part of an image a filter should be applied. Thus it is possible for a lot of filters to apply them not to the whole image but to parts of it. To use this concept just load a mask to the specified filter (compare section filter steps properties). The mask has to meet the following requirements:

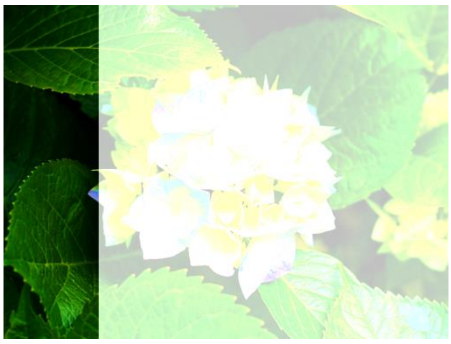
- The mask image has to be of the same size as the image to process.
- The white parts are the parts where the filter is applied.

A sample mask:



The original image:

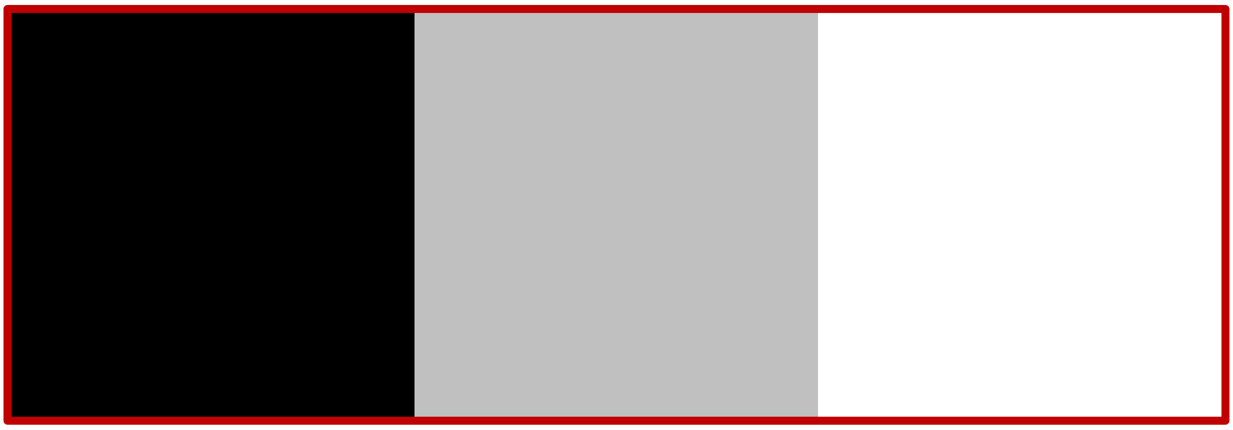
The filtered image with the mask and a brightness-filter:



10.2 Understanding Textures, Texturer and TexuredMerging

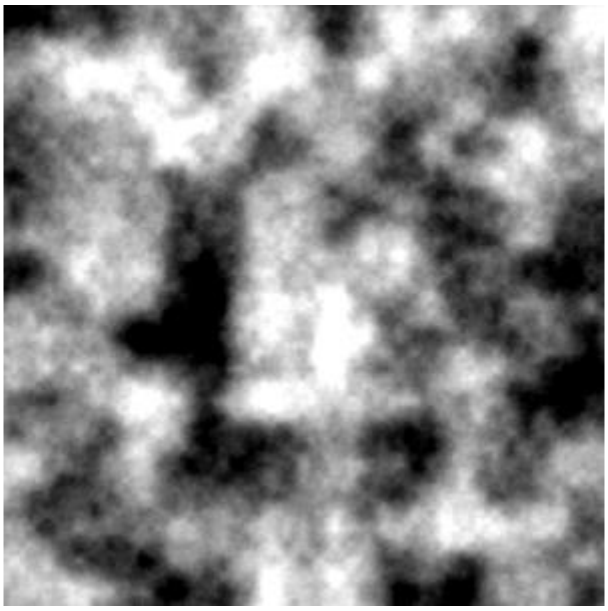
A texture is somehow an intensity mask - this means a TSFloatMatrix with values between 0 and 1 for each pixel of the mask. It may be generated with special texture generators or from a grey scale image (GenerateTextureFromGrayscaleImage). In this case, every grey value of the image will be divided by 256 to get a value between 0 and 1.

Hint: The filter ConvertImageToMatrix also generates a TSFloatMatrix but with the same values as the initial grey scale image. E.g. the value of 128 will convert to 128.0 at the same position. You can then reshape the Matrix (Filter ReshapeMatrix) to generate a row vector, for example. This is mainly used in machine learning filters.



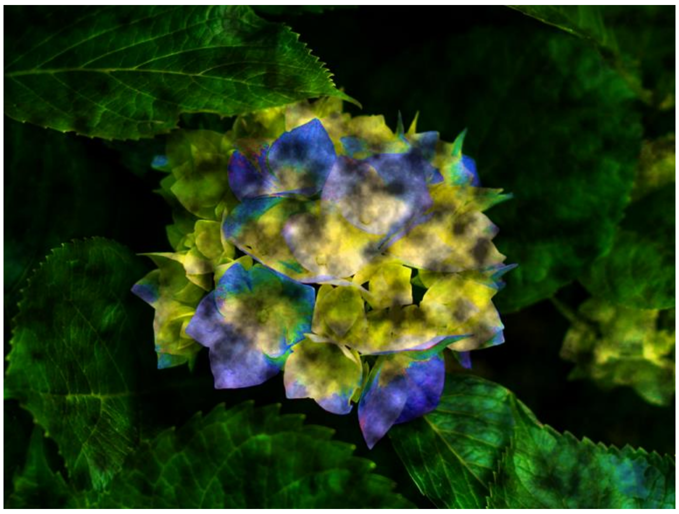
This image (without the red border) would produce a TSFloatMatrix with the image width and height and values of 0.0 for the black area, 0.5 for the grey area and 1.0 for the white area.

A more common example of a texture is the famous cloud texture as follows:



So let’s take a look what we can achieve with textures. We take the cloud texture as example. Don’t forget that the texture itself is a TSFloatMatrix, however you can easily translate it back to a grey scale image via the filter TextureToImage. The reverse filter for translating a grey scale image to a texture is GenerateTextureFromGrayscaleImage.

10.2.1 Texturer



This is the image filtered with the texture and a filter level of 0.8 and a preserve level of 0.2.

The Texturer works as follows:

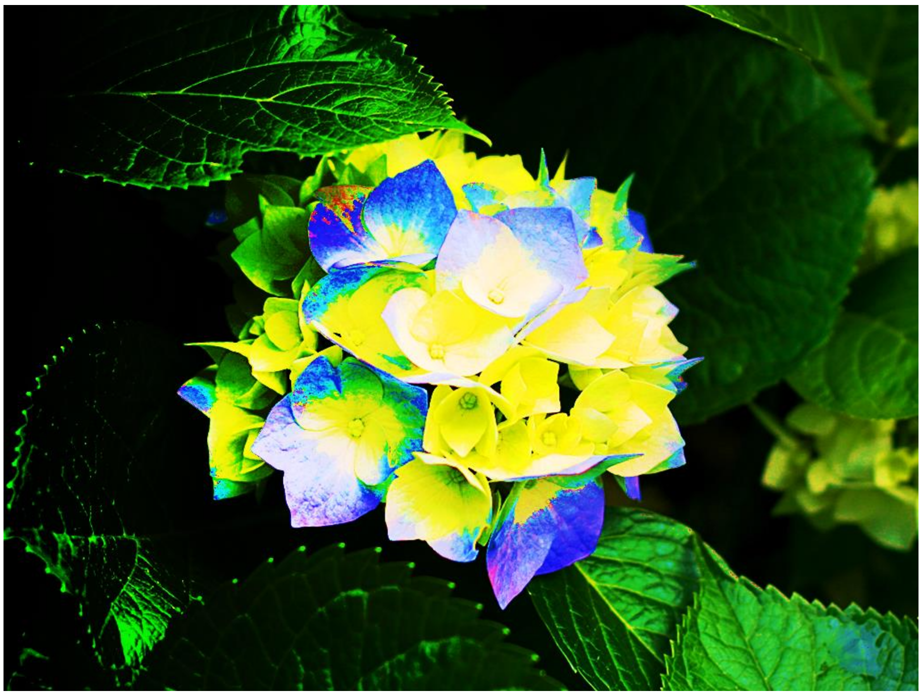
Destination = Source * PreserveLevel + Source * FilterLevel * textureValue

textureValue represents the value of the TSFloatMatrix!

10.2.2 TexturedMerge

Now this is a very powerful filter as you can use it to “scale” another filter. Take the masked filter described above. Here the filter is only used for certain areas of an image, but when it is used it is always used with the same intensity. TexturedMerge now offers the possibility to merge two images using a texture to determine which level of each image to use. If you therefore merge the original image with an image, which is the result of this original image applying a filter using a texture, you can handle the amount of filter application.

This is the image of our flowers where the contrast was drastically increased. Now we merge this using the gradient texture with the original flower image. This is the result:



The gradient texture used:

The areas where the texture is nearly black fully takes the picture with the high contrast. The areas where the texture is nearly white fully takes the original picture. The areas where the texture is grey take a mixture of both images.

The complete formula of this filter is as follows:

Destination = Source * textureValue + SecondImage * (1.0 - textureValue)

textureValue represents the value of the TSFloatMatrix!

10.3 Understanding Lists of CommonObjects

A lot of filters work with lists of CommonObjects, so it is vital to understand how they work. A List of CommonObjects is an indexed storage for many objects of the same type, e.g. TSCircleList, which is a list of circles:

Index	Object
0	TSCircle(Location: 200, 100; Radius: 50)
1	TSCircle(Location: 100, 50; Radius: 40)
…	…

It may also be an indexed storage for many objects of different types, e.g. TSCommonObjectList, which is a list of different CommonObjects:

Index	Object
0	TSCircle
1	TSPoint
2	TSPointList
3	TSFloatMatrix
…	…

All these lists start with index 0. There are filters which generate such lists and you can iterate through a list with the LoopOverObjectUntilFilterID-filter.

To get a single object from such a list, you can use the filter GetListElement where you have to use the index as key.

To get the amount of objects in the list, you can use the filter CountListElements. This may be important if it is not sure that a list contains elements.

If you want to use a list with a string as key you can use a dictionary of CommonObjects:

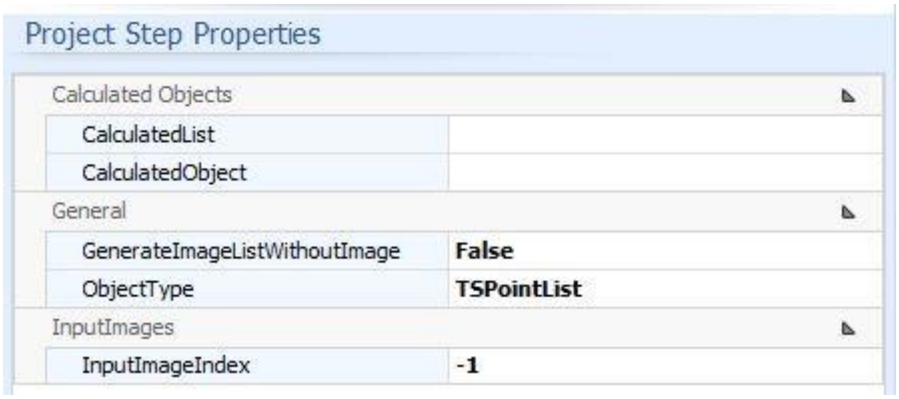
Key	Object
Whatever string you like	TSCircle
A different string	TSPoint
…	…

To get a single object here, you must use the GetCommonObjectDictionaryElement-filter.

10.4 Understanding AddObjectToObjectList-Filter

You can build a list of CommonObjects yourself via the filter AddObjectToObjectList. This filter is a little bit more complex and has three functions.

1. Generating a new list of CommonObjects
2. Adding an object to the end of a new list
3. Adding an object to the end of an existing list


Project Step Properties

Calculated Objects	
CalculatedList	
CalculatedObject	
General	
GenerateImageListWithoutImage	False
ObjectType	TSPointList
InputImages	
InputImageIndex	-1

You always have to set the correct ObjectType of the list. The GenerateImageListWithoutImage and InputImageIndex fields are only used in case of generating or working with a TSImageList.

If you leave the CalculatedList-field blank, the filter will generate a new list.

If you fill the CalculatedList field with a list that currently does not exist, it will also generate a new list.

There are some scenarios where the use of this filter generates different outcomes if you use it differently.

Imagine a loop which repeatedly executes an AddObjectToObjectList-filter: If you set the CalculatedList field, the filter will generate a new list at the first execution and afterwards use this generated list and add objects to it. If you do not set this field, the filter will always generate a new list and add the current object. The second technique is good for storing a single object.

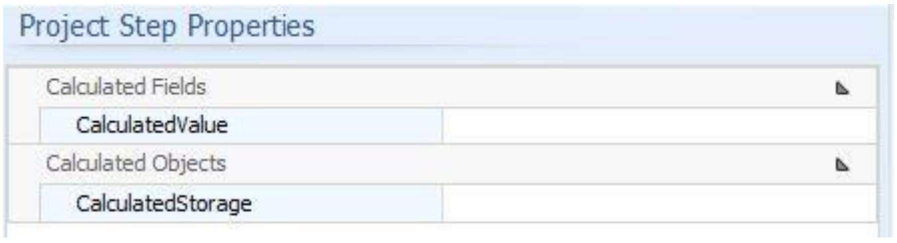
However, imagine a loop which possibly does not always execute the AddObjectToObjectList-filter due to skipping steps or breaking or continuing the loop or the like. If you want to reference the list later, this will lead to an error. In this case you have to place an AddObjectToObjectList-filter before the loop with a blank CalculatedList field and reference this list during the loop.

Hint: There are some sample projects in the tutorial section, which demonstrate the use of this filter.

Hint: You can create a TSIntlist with the GenerateIntList-filter in a single step.

10.5 Understanding SingleValueStorage

A single value storage stores a single double value. Its main purpose is to store a value for comparison during a loop. For example, you want to find an object with the lowest value. You can loop all of these objects and store the currently lowest value in the SingleValueStorage. If you find an even lower value, you will store this value.



If you leave the CalculatedStorage field blank, a new storage will be generated and returned.

If you set the CalculatedValue field, this value will be the new value of the storage and the storage will be returned.

If you leave the CalculatedValue field blank, the current value will be returned but not the storage.

Hint: Store the corresponding object with AddObjectToObjectList and leave the CalculatedList-field blank as described above. So you simulate a SingleValueStorage for this object. You could use AddObjectToObjectList as SingleValueStorage as well. However, the SingleValueStorage-filter is more convenient as it does not need a GetListElement-filter.

10.6 Difference between Subfilter and Subproject

A subproject is a project loaded to another project and executed as single filter step (Compare section Menu Project). This is an important thing for reusing your know how. Imagine several projects which all share a section of the same filter steps. This is the point where subprojects come into play. If you later alter some parameters or filters in this section, you just have to alter the subproject and this will affect all your projects.

A subfilter is a filter step which works as input for another filter step. This is primarily to apply a certain filter step to all images of a TSImageList. In this case you can, for instance, insert a resizing filter and afterwards ApplyFilterToImageList and reference the resizing filter there. The result is that all your images of your referenced TSImageList will be resized. You do not have to loop the TSImageList and resize in the loop.

Hint: See the tutorial section in the sample projects for understanding the use of these techniques.

10.7 Using a subproject with InputVariables

Sometimes it might be useful to pass objects and values to a subproject. E.g. a subproject may take a TSImageList which was generated by the main project as input and work on it.

You can pass such objects to the subproject using the field InputVariables. The syntax is as follows: IDOfTheSourceFilterStep , IDOfTheTargeFilterStep , NameOfTheTargetField ; IDOfTheSourceFilterStep , IDOfTheTargeFilterStep , NameOfTheTargetField ; …

IDOfTheSourceFilterStep is the ID of the filter step in the main project which outputs the object you want to pass to the subproject.

IDOfTheTargeFilterStep is the ID of the filter step in the subproject where you want to input the object.

NameOfTheTargetField is, as it says, the name of the field where you want to input the object at the target filter step.

Hint: The objects are referenced. Hence it follows that changing them in the subproject changes the object in the main project as well.

11 Programming / Referencing SchiferVision Engine

First you have to reference the SchiferVisionEngine.dll to your project. This is a .Net 4.0 C# dll.

If you want to use filters which use opencv (Emgu-Extension), you have to set the target platform to x86 as these are 32bit filters.

A sample implementation may be as follows. You can take a look at the attached SchiferVisionTestProgram for further source details.

```
using TSImaging;

SchiferVisionProject SchiferVisionProject;

LoadProjectReturnCodes loadProjectReturnCodes =
SchiferVisionProject.LoadProject(projectFileName);
switch (loadProjectReturnCodes)
{
       case LoadProjectReturnCodes.CannotLoadFile:
              MessageBox.Show("Cannot load Project");
             return;
       case LoadProjectReturnCodes.FileNotFound:
              MessageBox.Show("Cannot find ProjectFile");
              return;
       case LoadProjectReturnCodes.OK:
              break;
}

SchiferVisionProject.SampleImage = Bitmap.FromFile(imageFileName);

Image image = SchiferVisionProject.ApplyProjectToImage(out
applyProjectToImageReturnCode);

if (applyProjectToImageReturnCode != ApplyProjectToImageReturnCodes.OK)
    MessageBox.Show("Cannot apply Project to Image", "Error");
```

11.1 Setting Parameters

You can set the input parameters of calculating- and direct manipulating filters in the following way. This is useful if your application uses several SchiferVisionProjects and some of the parameters depend on previous program steps:

```
// sample usage (Sets the Parameter "InputMax" of the Filter with the ID 0 to 144:
Hashtable htID0 = new Hashtable();
htID0["InputMax"] = 144;
SchiferVisionProject.htInputVariables[0] = htID0;
```

You can read the output of calculating filters in the Hashtable htCalculatingFilters. Sample usage: Getting the output of the filter with ID 7:

```
CommonObjects.TSLineList lineList =
(CommonObjects.TSLineList)SchiferVisionProject.htCalculatingFilters[7];
```

Hint: To use CommonObjects in your application, you need to reference the CommonObjects.dll and Interfaces.dll as well.

11.2 Getting Images of the filter chain

You can use images which are generated during applying the project as well if you do not choose to delete them after applying (parameter for the apply-method!). Storing the images in applications with many project-runs may cause memory problems! Getting the image of the filter with ID 7:

```
Image f = (Image)SchiferVisionProject.htImagesAfterStep[7];
```

11.3 Setting the CountProjectRuns

If you execute your project more than once, you may need to know how often the project is executed directly in your project. You can get this number in your project with the filter GetProjectRuns. The following line sets the CountProjectRuns to 4.

```
SchiferVisionProject.CountProjectRuns = 4
```

11.4 Breaking the execution

You can tell the SchiferVisionProject to break its current execution. Compare section Configure SchiferVisionEngine->AllowApplyBreak.

```
SchiferVisionProject.breakApplying = true;
```

The execution will be broken and the current state of execution will be returned.

11.5 Building your own filter

As mentioned before, SchiferVision offers the possibility of producing your own filter and plug it into the SchiferVisionProjectDeveloper as well as the SchiferVisionEngine.

To do so, you first have to decide which kind of filter you want to develop: A direct manipulating or an object calculating filter.
You then have to reference the Interfaces.dll and make a new class which implements the desired interface, e.g. IFilterManipulateBitmapDirectly.
You need some properties which are necessary as follows:

The name of the filter:

```
private string name = "Blur";

[BrowsableAttribute(false)]
```

```
public string Name
{
    get { return name; }
    set { name = value; }
}
```

The name of the filter has to be the same as the name of the class and the namespace!

The category of the filter, where it should be plugged in the menu of the SchiferVisionProjectDeveloper:

```
private string filterManipulateBitmapDirectlyCategory = "Convolution";

[BrowsableAttribute(false)]
public string FilterManipulateBitmapDirectlyCategory
{
    get { return filterManipulateBitmapDirectlyCategory; }
    set { filterManipulateBitmapDirectlyCategory = value; }
}
```

The host (This is for the plugging mechanism):

```
private IFilterManipulateBitmapDirectlyHost host;

[BrowsableAttribute(false)]
public IFilterManipulateBitmapDirectlyHost Host
{
    get { return host; }
    set
    {
        host = value;
        host.Register(this);
    }
}
```

These are the mandatory properties. You may set the [BrowsableAttribute(false)] so that it cannot be seen in the property grid of the SchiferVisionProject developer.

In the following, a few other properties which are useful:

```
private int inputImageIndex = -1;

[Category("InputImages")]
[Description("Sets the InputImageIndex. -1 for previous Image, -2 for initial Image or
the index of the desired Step.")]
[BrowsableAttribute(true)]
public int InputImageIndex
{
    get { return inputImageIndex; }
    set { inputImageIndex = value; }
}
```

Here you see the inputImageIndex where `[BrowsableAttribute(true)]` is set to see this in the property grid of the SchiferVisionProjectDeveloper. Also the category and a description are set. This is where the property appears in the property grid and the description which is shown under the property grid.

If you want to use the built-in ExpressionParser for calculating values, you can do this by including Calculated in your property name, e.g. CalculatedAjustValue. You have to do this to reference the output of other filters as well:

```
private string calculatedAdjustValue = "";

[Category("Calculated Fields")]
[Description("Enables calculating the Brightness Correction from previous Filters
using RPN.")]
[BrowsableAttribute(true)]
public string CalculatedAdjustValue
{
    get { return calculatedAdjustValue; }
    set { calculatedAdjustValue = value; }
}
```

If you want to use the built-in ExpressionParser, you have to reference the ExpressionParser.dll and use it as follows:

```
if (calculatedAdjustValue != "")
        AdjustValue = Convert.ToInt32(Math.Round(Convert.ToDecimal(new
ExpressionParser.ExpressionParser().ParseExpression(calculatedAdjustValue))));
```

Lastly you need a method Apply. This has to return either a bitmap for direct manipulating filters or an object for calculating filters.

```
public Bitmap Apply(Bitmap source)
{
       // do something

       return source;
}
```

If you want to develop a direct manipulating filter which calculates an object as well, you can use the following property:

```
private object generatedOutputObjectValue = null;

[BrowsableAttribute(false)]
public object GeneratedOutputObjectValue
{
    get { return generatedOutputObjectValue; }
    set { generatedOutputObjectValue = value; }
}
```

Set it in your apply-method:

```
generatedOutputObjectValue = yourObject;
```

You can use the following property to set a short description which is shown as tooltip in the SchiferVisionProjectDeveloper filter menu:

```
private string filterShortDescription = "OpenCVs ThresholdAdaptive";

[BrowsableAttribute(false)]
public string FilterShortDescription
{
    get { return filterShortDescription; }
    set { filterShortDescription = value; }
}
```

```
You can use the HelpTextEditor to show a readonly help in the property grid of the
SchiferVisionProjectDeveloper:

private object help = "Implementation of OpenCVs ThresholdAdaptive method!";

[Category("Help")]
[Description("Describes the filter itself.")]
[BrowsableAttribute(true)]
[Editor(typeof(LoadHelpTextEditor.LoadHelpTextEditor),
typeof(System.Drawing.Design.UITypeEditor))]
public object Help
{
    get { return help; }
    set { help = value; }
}
```

You can use some other built-in editors, however you can develop your own editor as well, it just has to be a System.Drawing.Design.UITypeEditor.

11.5.1 Using LiveTry

If you want to use the LiveTry for direct manipulating filters, you can do that in the following way:

Make a Hashtable and name it htFilterProperties:

```
private Hashtable htFilterProperties;
```

and fill it in the following way:

```
htFilterProperties = new Hashtable();
htFilterProperties.Add("AdjustValue", new int[] { -255, 255, 0 });
```

AdjustValue is the name of the property, followed by the data type as array and some information about the boundaries and the start value. The example above generates a slider of intvalues, which goes from -255 to 255 and the start value is set to 0.

For a double value, it works in the following way:

```
htFilterProperties.Add("FullnessMax", new double[] { 0, 1, 1, 100 });
```

This generates a slider of double values, which goes from 0 to 1 with the start value set to 1. The last number here is a multiplier to make the slider more convenient. So the full range of values will be multiplied with 100 to get a slider with 100 steps in this example.

Some other data types:

```
htFilterProperties.Add("OrientationMin", new float[] { 0, 360, 0, 100 });
```

Floatvalue works as double value.
You can use byte-values and int16-values in the same way as int-values.

```
htFilterProperties.Add("AreaCheck", false);
```

A Boolean value, just write the value. This will generate a checkbox.

```
htFilterProperties.Add("FillColor", Color.Black);
```

This will generate a colour edit.

```
private short[,] se = new short[3, 3] {
                                       {1, 1, 1},
                                       {1, 1, 1},
                                       {1, 1, 1}};
htFilterProperties.Add("MorphologyMatrix", se);
```

This demonstrates a matrix of shortValues. You can use an int-matrix in the same way.

You can use enums the following way:

```
DataTable dtMorphologyMethod = new DataTable();
dtMorphologyMethod.Columns.Add("1");
foreach (MorphologyMethod ecc in Enum.GetValues(typeof(MorphologyMethod)))
{
    DataRow dr = dtMorphologyMethod.NewRow();
    dr[0] = ecc.ToString();
    dtMorphologyMethod.Rows.Add(dr);
}
htFilterProperties.Add("MorphologyMethod", dtMorphologyMethod);
```

This will produce a dropdown-list with the values.

If your filter needs to know if it is a normal execution or a LiveTry, you can use the following property which will be set to true during LiveTry.

```
bool isLiveTry = false;
[BrowsableAttribute(false)]
public bool IsLiveTry
{
    get { return isLiveTry; }
    set { isLiveTry = value; }
}
```

11.5.2 Serialization

If you want that your filter is saved in the project, the class has to implement the following interface: `System.Xml.Serialization.IXmlSerializable`

The reader works as follows:

```
public System.Xml.Schema.XmlSchema GetSchema() { return null; }

public void ReadXml(System.Xml.XmlReader reader)
{
      reader.MoveToContent();
      Name = reader.GetAttribute("Name");
      InputImageIndex  = Convert.ToInt32(reader.GetAttribute("InputImageIndex"));
      CalculatedInputMinRed   = reader.GetAttribute("CalculatedInputMinRed");
}
```

The writer:

```
public void WriteXml(System.Xml.XmlWriter writer)
{
      writer.WriteAttributeString("Name", Name);
      writer.WriteAttributeString("InputImageIndex", InputImageIndex.ToString());
      writer.WriteAttributeString("FillColor_R" , FillColor.R.ToString());
}
```

12 Configure SchiferVisionEngine

12.1 TraceLevel

Use SchiferVisionEngine.dll.config to set the TraceLevel. Possible values are:

- Error: Shows detailed error messages, exceptions and stack trace
- Off: Shows only a message box indicating an error and the filterid and project runs where the error has occurred.

Here is the section of the config file:

```
<setting name="TraceLevel" serializeAs="String">
        <value>Off</value>
</setting>
```

Hint: If you use the SchiferVisionProjectDeveloper option, “ShowCompleteErrorMessages” is taken and not the config file (Compare section Menu Options).

12.2 AllowApplyBreak

You can also set if it is allowed to break the current execution of the project. In SchiferVisionDeveloper you can press ESC and the current execution will be broken if the following value is set to “True”. This is a good thing if you test new projects and you are not sure whether all jumpToIDs and loopToIDs are set correctly. To break the project in the case of an infinite loop or if the execution lasts too long, set this to “True” and press ESC as mentioned. However, the execution time will be significantly slower if this is set to “True”.

```
<setting name=" AllowApplyBreak " serializeAs="String">
        <value> False </value>
</setting>
```

Hint: If you use the SchiferVisionProjectDeveloper, the option “AllowApplyBreak” is taken and not the config file (Compare section Menu Options).