TISSUEGNOSTICS

PRECISION THAT INSPIRES

V8.0

StrataQuest

Engines References

Table of Contents

1. Disclaimer

TissueFAXSplus is a microscope-based cell analysis system for cells in cryocut, paraffin-sections and/or TMAs. It consists of the software modules "TissueFAXS" combined with either "TissueQUEST" & "HistoQUEST" or with "StrataQUEST" and is used for acquisition of images in brightfield and/or fluorescence and/or confocal and/or multiplexing and/or multispectral mode, for counting the number of positive and negative cells and/or for quantification of staining intensities and/or classification of histological structures. TissueFAXSplus is used for the standardization of tissue analysis in combination with immunohistochemical and immunofluorescence staining.

TissueFAXSplus does not give any direct diagnosis and there is the possibility that the tissue specimen does not provide enough information for a proper analysis and/or diagnosis. The cell analysis system only measures cellular parameters. Such measurement parameters must in any case be reviewed and validated by a qualified human professional with profound knowledge of the sample under investigation and its cellular constituents as well as profound knowledge of the cell analysis system and who has received special training in the operation of cell analysis methods. The cell analysis system can under no circumstances make a decision on the therapy of patients.

The results of the analysis are purely statistical values. Users must re-evaluate images and likelihood of the statistical data. Pure interpretation of statistical data is not allowed.

Notes:

- TissueFAXS FLUO is similar to TissueFAXSplus but is for fluorescence samples only and must not be used with brightfield/immunohistochemical samples. TissueFAXS FLUO consists of the software modules "TissueFAXS" as well as "TissueQUEST" and/or "StrataQUEST FLUO".

- TissueFAXS HISTO is similar to TissueFAXSplus but is for brightfield/immunohistochenical samples only and must not be used with fluorescence samples. TissueFAXS HISTO consists of the software modules "TissueFAXS" as well as "HistoQUEST" and/or "StrataQUEST HISTO".

- TissueFAXS 200 and TissueFAXS SL are similar to TissueFAXSplus but for batch scanning of up to 200/120 brightfield/immunohistochenical and/or fluorescence slides. TissueFAXS 200 / TissueFAXS SL consist of the software modules "TissueFAXS 200" / "TissueFAXS SL" as well as "HistoQUEST" and/or "TissueQuest" and/or "StrataQUEST".

- TissueFAXS SPECTRA is similar to TissueFAXSplus but is for multispectral imaging of fluorescence samples only. Scanning of brightfield/immunohistochenical samples on TissueFAXS SPECTRA is possible with the colour camera, if any, but MUST NOT / CANNOT be done with the multi-spectral camera. TissueFAXS SPECTRA consists of the software modules "TissueFAXS SPECTRA" as well as "TissueQUEST" and/or "HistoQuest" and/or "StrataQUEST".

Using the unmixing procedure must be done with care as tissue-derived autofluorescence may interfere with the algorithm and cause inconclusive or even faulty output.

Selecting and correctly assigning proper reference spectra is critical for the accuracy of the unmixing result and is the sole responsibility of the user. ***Selecting wrong reference spectra and/or assigning any reference spectra inappropriately will cause inconclusive / faulty output and thus generate false measurement results!***

- TissueFAXS CHROMA is similar to TissueFAXSplus but is for multi-channel imaging of fluorescence samples through narrow band-pass filters only. Scanning of brightfield/immunohistochenical samples on TissueFAXS SPECTRA is possible with the colour camera. TissueFAXS CHROMA consists of the software modules "TissueFAXS" as well as "TissueQUEST" and/or "HistoQuest" and/or "StrataQUEST".

- StrataQuest provides a software tool for machine learning for the automatic classification of tissue structures, including detection of tumor areas in tissue sections. The results generated by the software have to be verified by a qualified human professional in any case (negative as well as positive results). In case the software does not detect specific histological structures and/or tumor cells, a human professional has to verify this result by other means, as it is possible that certain biological patterns and/or special types of (cancer) cells may not be detected by the automatic detection function. Measurement errors may also occur due to the fact that the cell-environment in which the software has to operate in is highly variable.

We point out to the fact that the following circumstances/factors might influence and/or impair the result of the analysis to the level where the result rendered might be inconclusive or even faulty:

- Quality of the tissue sample: In this context, especially the age of the tissue sample is relevant. Long durations between the harvesting and/or staining of the tissue sample and the analysis as well as storage errors can tamper with the outcome of the analysis.
- Quality of the preparation of the tissue sample for the analysis and the materials used: In this context, especially the type and quality of the reagents and the capability/precision of the person handling the reagents can be relevant and may lead to inconclusive/faulty results (for example: dilution-errors concerning the reagents).

These factors (as well as the capability of the human professional performing the validation of the test results) lie solely within the responsibility of the user of the software. TissueGnostics does not take any responsibility for test results that are influenced by one of the above mentioned factors.

Each and any product shall be used only after training performed by TissueGnostics or authorized distributors of TissueGnostics. A list of authorized distributors is available at the TissueGnostics website.

https://tissuegnostics.com/global-network/distributors

The TissueFAXS system and its software applications are FOR RESEARCH USE ONLY.

*By using only parts of the system, changing system components (hardware or software) or using the TissueFAXSplus instrument in any other than the intended way it was designed for, the CE-Declaration becomes **invalid**.*

Notices about symbols and labels on product

The following symbols appear on the labels of the product:

SN — Serial number	WEEE symbol / separate disposal of electrical and electronic equipment
Manufacturer	CE mark

2. Introduction

The Engines Reference document complements StrataQuest 8 User Manual and it describes the functions of the analytical engines that are included in the application.

2.1. About Experimental Engines

An experimental engine or measurement means it is out of IVD and requires special care from the users.

The user is responsible with the validation of the results generated by experimental engines and the way they are going to be used.

3. Assign classes to objects

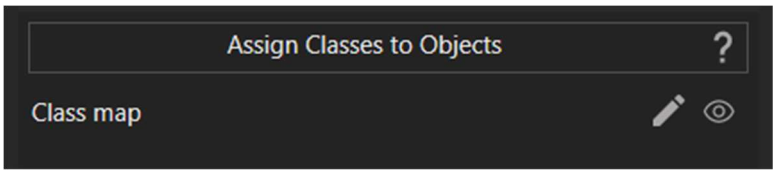


Figure 1 – Assign classes to objects

Where it can be found:

Layer's "Measurements" ► Add ► Measurements (Advanced) ►Assign classes to objects

Description:

To each object is assigned the id / label of the bigger / larger object, in which is it placed / positioned / found. This information can be used to find out how many objects belong to a bigger structure, e.g. how many nuclei are in macro-structure.

Assign classes to objects is a simple measurements engine that aids in using Classifier or Proximity Areas outputs. It will assign to each event the index of the class / proximity area it is located in. Using a histogram with this measurement eases the visualization / counting of objects in a certain class / area.

Parameters:

- Class map – grayscale image representing the map of the available classes (each pixel has the class id specific to its class).

Note	This engine is considered a measurements engine, the events to which classes are assigned are the ones detected in current layer's "Detection" section.

Effect:

Each event will be assigned to the class id in which it is located.

Example:

- Input:

Image below is a brightfield color image (8-bit, 3-channel) representing a small region from a colon sample, where nuclei and structures have been detected.

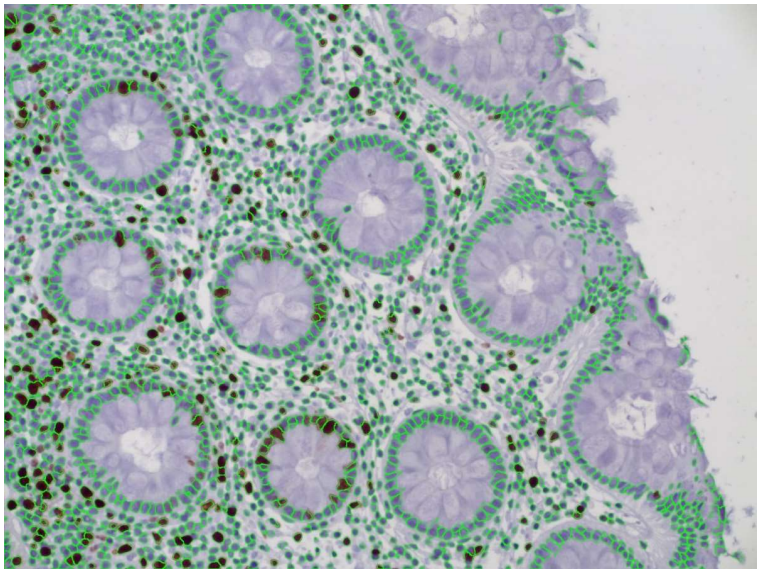


Figure 2 – Detected nuclei

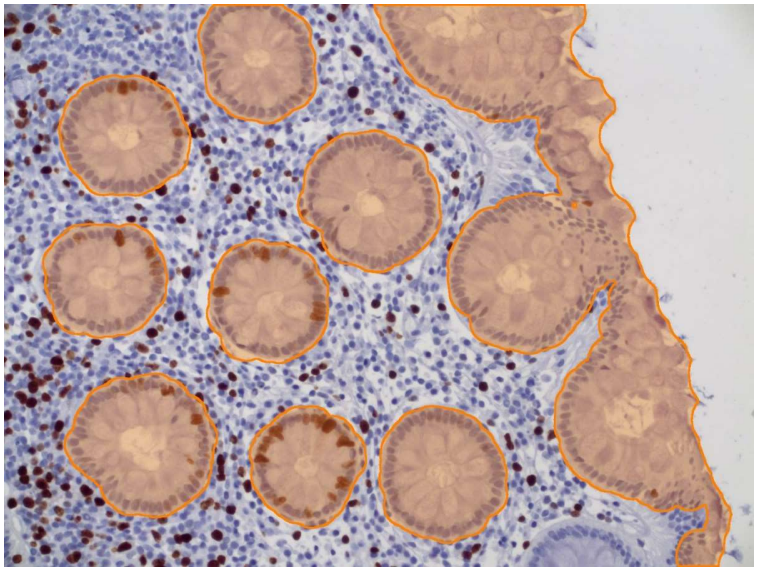


Figure 3 – Detected glands

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a measurement value associated with each event (nuclei).

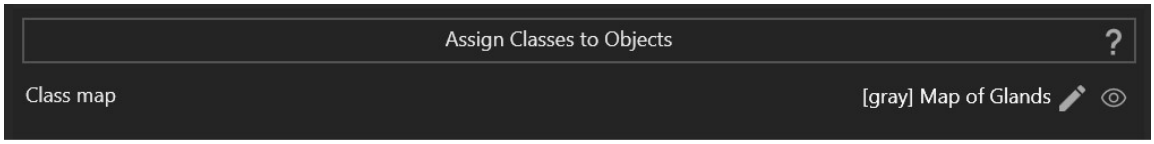


Figure 4 – Assign classes to objects

- Output:

The result can be visualized with the help of View Event Data functionality. Figure below shows the View Event Data activated for some events (nuclei and structures) to highlight the class (structure) id assignment to each nucleus.

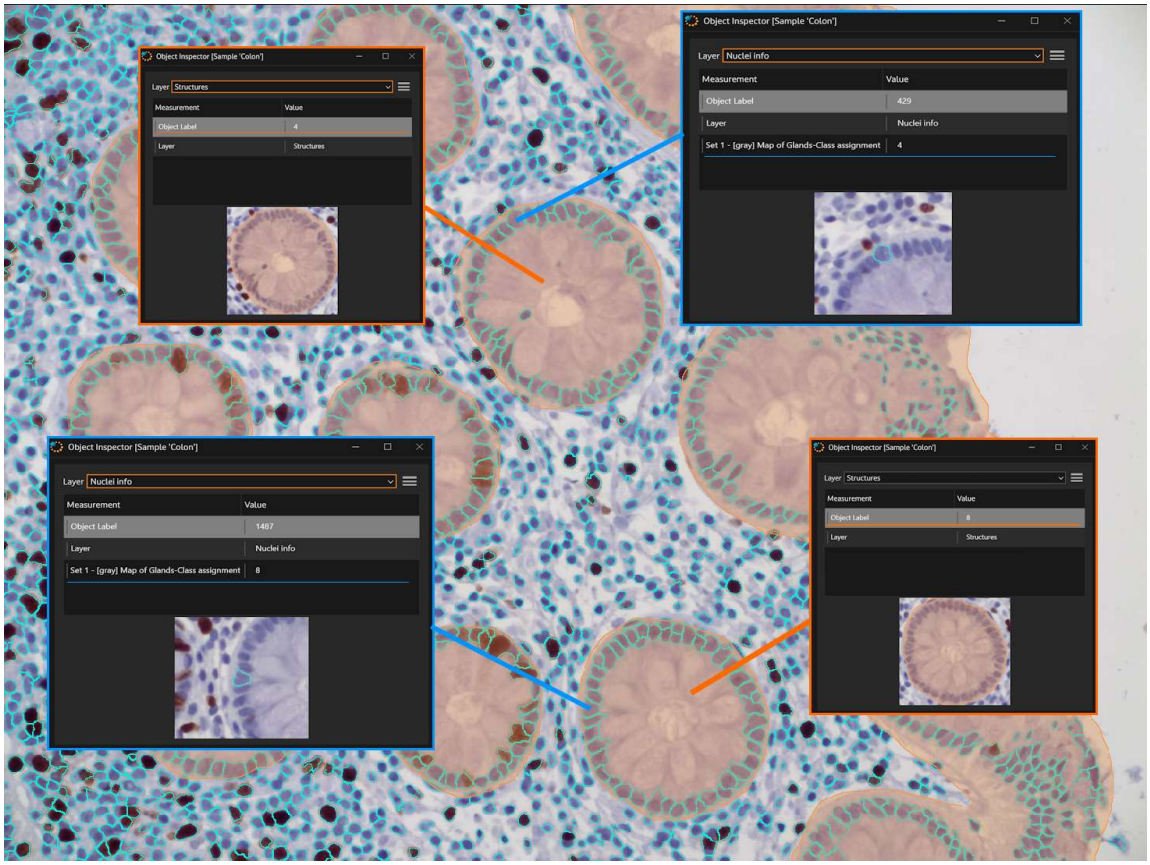


Figure 5 – View Event data with highlighted classes

4. Background Removal

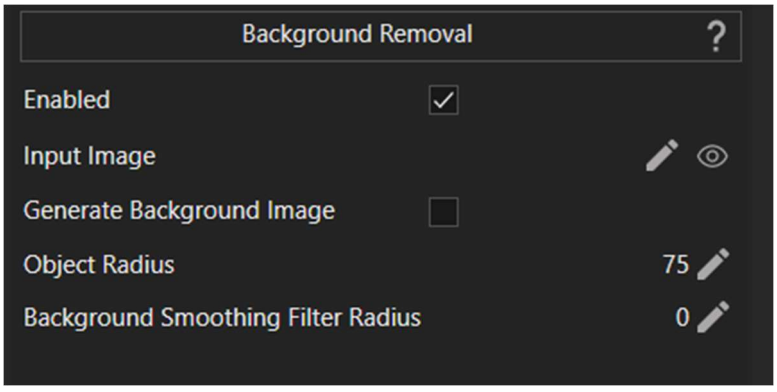


Figure 6 – Background removal

Where it can be found:

Layer's "Pre-Processing" ► Add ► Operations (Advanced) ►Background Removal

Description:

Background Removal is an engine used to remove / lower the impact of the high background in the region of interest. This is done by generating a background model which is subtracted from the original image. The effect is mostly noticeable in the 16-bit images, where the background can be associated with a white noise (random values with a specific average value and finite variance).

Parameters:

- Enabled - enables / disables the engine (while unchecked, the engine is skipped in the analysis workflow).
- Input Image - inputs a grayscale image.
- Generate background image - enables / disables the estimated background model image as output.
- Object Radius - the average radius of the larger objects that are present in the image. If this value is too small, larger objects tend to be affected by the background removal process.
- Background smoothing filter radius - the size of the smoothing filter applied on the estimated background model, before removal.

Effect:

Removes or decreases the impact of the high background in grayscale images.

Example:

For this example, we consider a small region of interest (ROI) in a kidney sample. Figure 2 shows the overlay of the original channels (DAPI, FITC, Texa, Cy5). The images have been acquired using 16-bit. The Background Removal engine has been applied to each original channel and the overlay of the resulting images can be seen in Figure 3.

Two different settings have been used:

- one for DAPI:

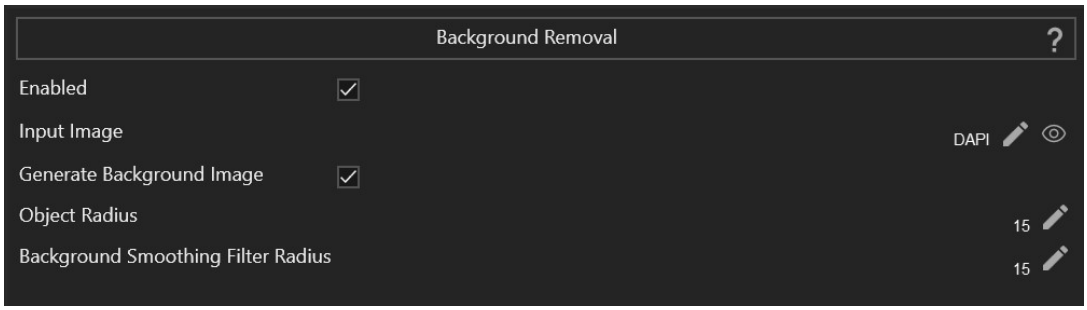


Figure 7 – Engines settings used for DAPI channel

- a second one for FITC, Texa and Cy5:

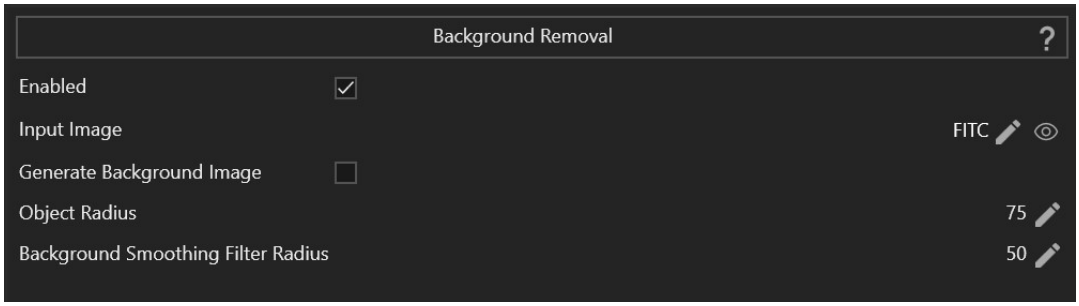


Figure 8 – Engines settings used for FITC, Texa and Cy channels

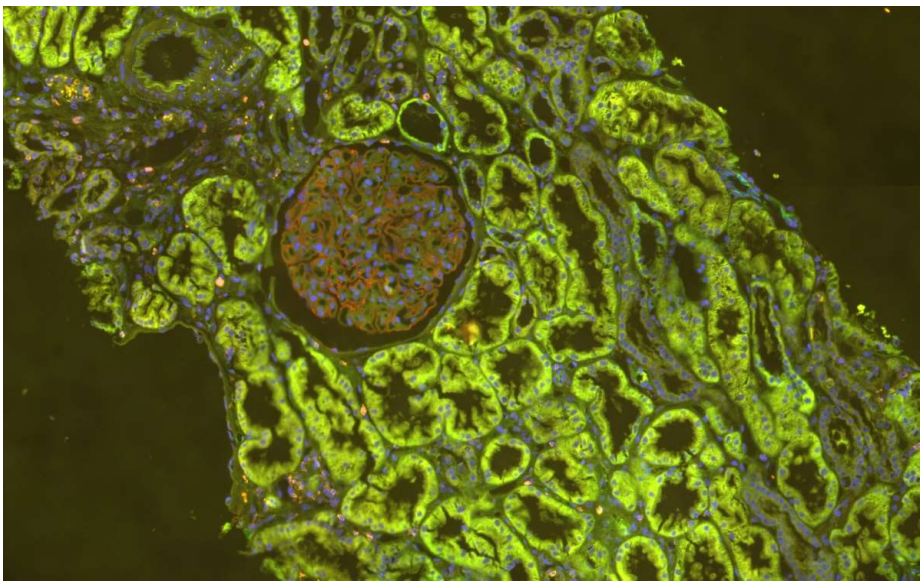


Figure 9 – Overlay of the original channels (DAPI, FITC, Texa, Cy5)

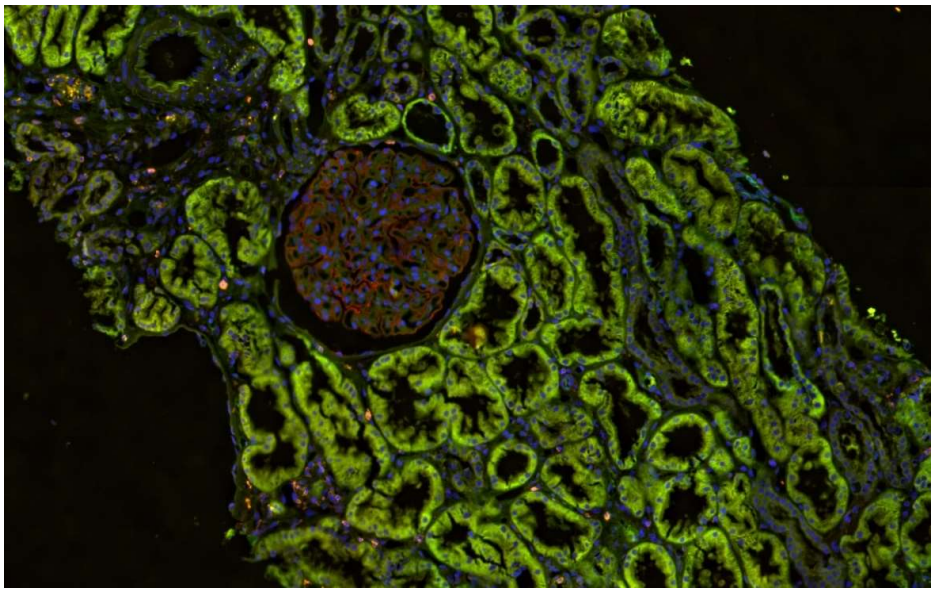


Figure 10 – Overlay of the resulting channels (after Background Removal)

5. Basic Operations Module

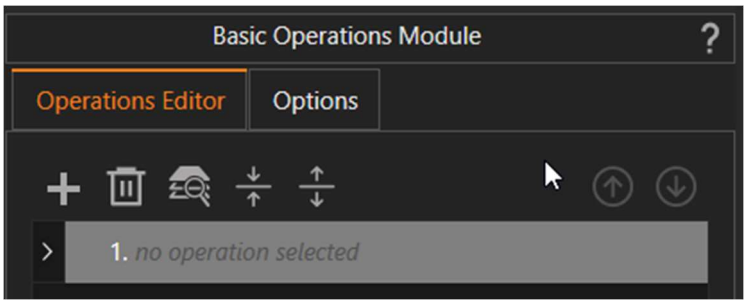


Figure 11 – Basic Operations Module: Operations Editor

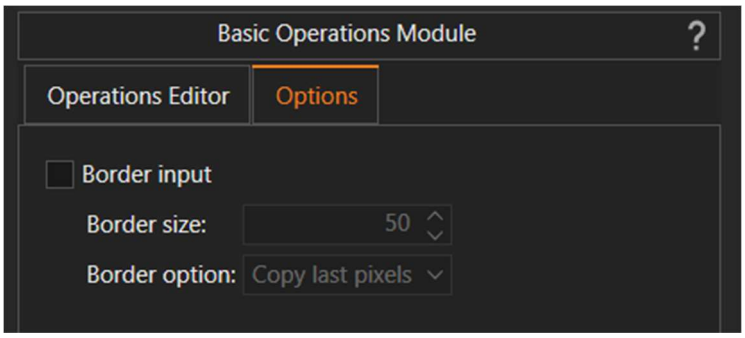


Figure 12 – Basic Operations Module: Options

Where it can be found:

Layer's "Pre-Processing" ► Add ► Operations (Basic) ► Basic Operations Module

Description:

Basic Operations Module is an engine which allows the combination of basic operations available in a list, to create more powerful and complex processing operations. A basic operation is a function which takes inputs (images and specific parameters) and generates outputs (e.g. add 2 images; apply a filter on an image, etc.) The input image parameters are not typed, meaning the input image can be 8-bit or 16-bit, 1-channel or 3-channels, so it's the user who has to choose the proper output image type, based on the input image type and the selected operation (e.g. Label engine requires a mask image as input and generates a coded image as output).

It is structured in columns:

- WBS: work breakdown structure shows the decomposition of the Basic Operations Module.
- Name: shows the names of the items composing an operation.
- Type: defines the type of operation.
- Value: defines the value of an operation.

Effect:

- The effect of this engine may vary based on the operation or the combination of operations selected.

Operations Editor:

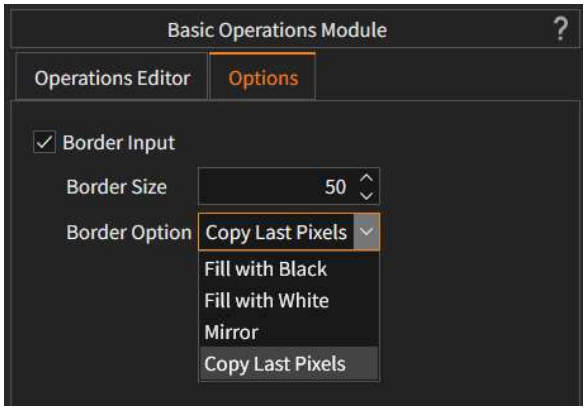
This is the main tab in the Basic Operations Module (see above images). It contains the basic controls needed to combine different operations to obtain the desired effect(s) / result(s).

- Add Operation - add a new operation
- Remove Operation - removes selected operation
- Move Up - moves the selected operation up in the list
- Move down - moves the selected operation down in the list

Note	An operation can use the output generated by previous operations as inputs. That is why the order of operations is very important. Only the outputs generated by operations performed before the current operation are available for input. Changing the order of operations on the list will affect the list of available inputs for each operation.

Options:

This is the second tab in the Basic Operations Module.



- Border Input - Enables / disables the usage of information from neighboring Fields of View (FOV). When this option is enabled, a border is added to the current processed FOV, using information from neighbor FOVs (the yellow area from figure below).
- Border Size - The number of pixels that will be used from the neighbor images to create the border. The maximum value allowed is the minimum between the width and height of the image (size of the yellow area from figure below).
- Border Option:

- **Fill with black**: the image is padded with black pixels (0 values);

- **Fill with white**: the image is padded with white values (e.g. 255 for 8-bit images);

- **Mirror**: the image is padded with the mirrored values of the pixels close to border (e.g. p1 p2 p3 | p3 p2 p1);

- **Copy last pixels**: the image is padded with pixels having the values of the closest pixel to border (e.g. p1 p2 p3 | p3 p3 p3).

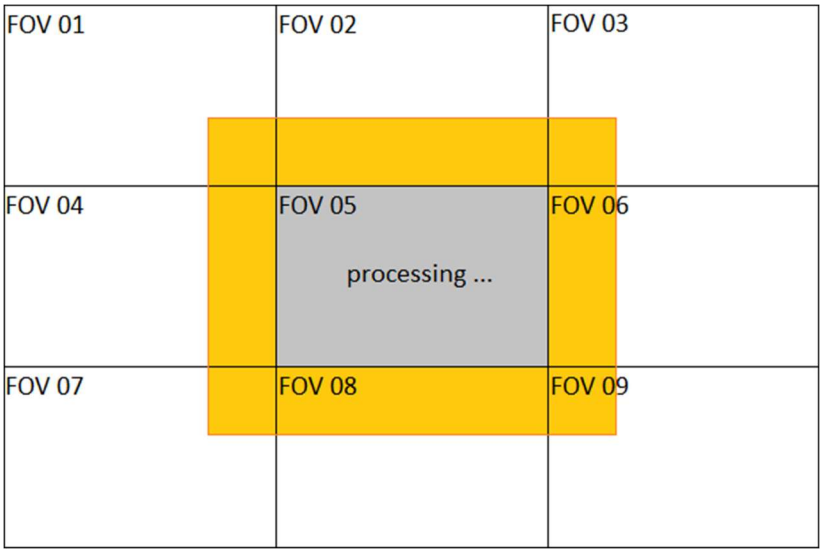


Figure 13 – Border example

Example:

The next example shows why it can be helpful to use information from the adjacent Fields of View (FOVs). A small Region of Interest (ROI) is considered, showing 2 nuclei stained with DAPI. The ROI is created at the border of two FOVs. The border is highlighted in Figure 8 by the white line.

Two cases have been considered:

- A filter applied on the DAPI channel with the border option enabled and with the border option disabled.
- Nuclear detection engine applied on the DAPI channel with the border option enabled and with the border option disabled.

From these two examples it can be observed that information that has continuity on neighboring FOVs is interpreted correctly when the border option is enabled. In other words, the information from the processed FOV is decoupled from the information from the neighboring FOVs when the border option is not used.

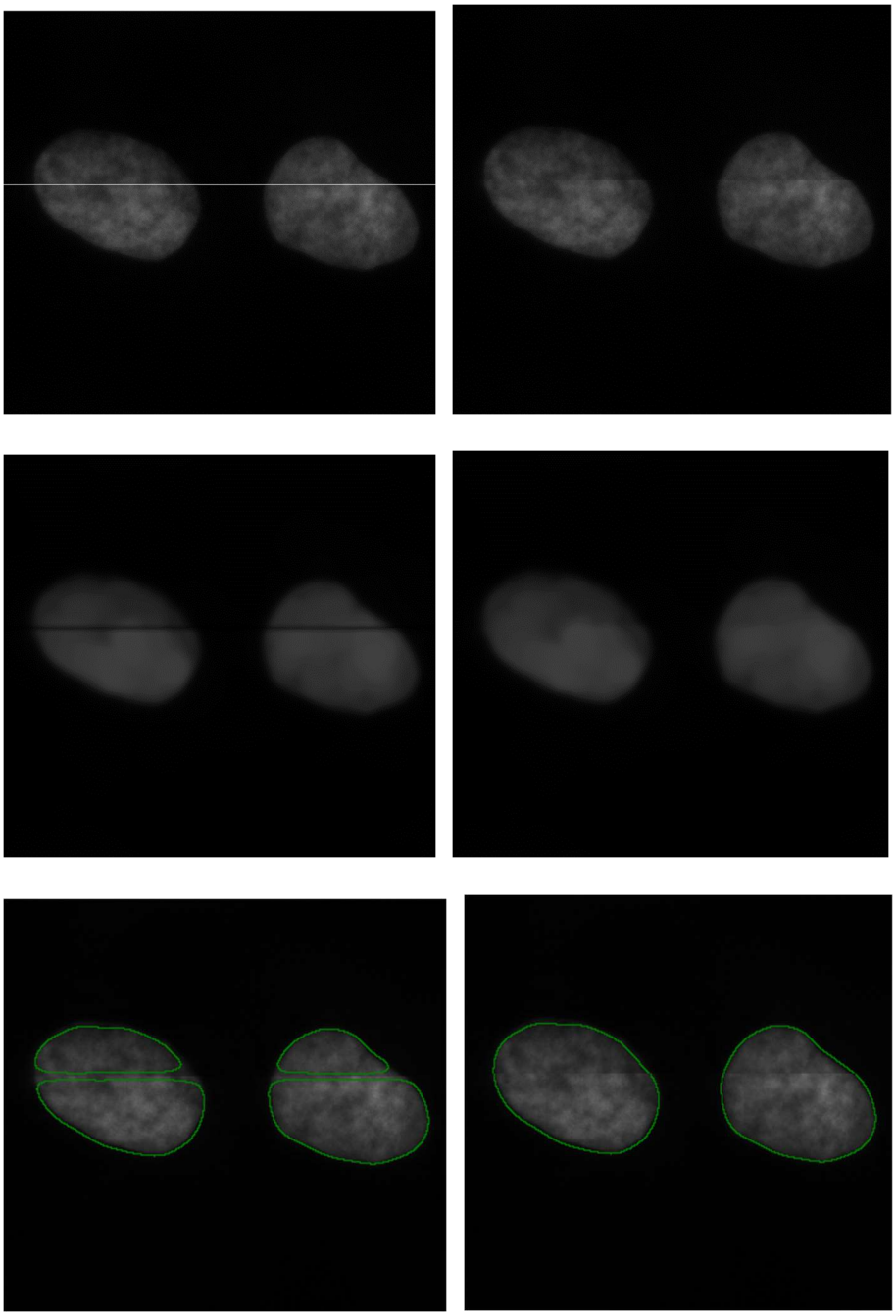


Figure 14 – Border results

5.1. Arithmetic

Add images

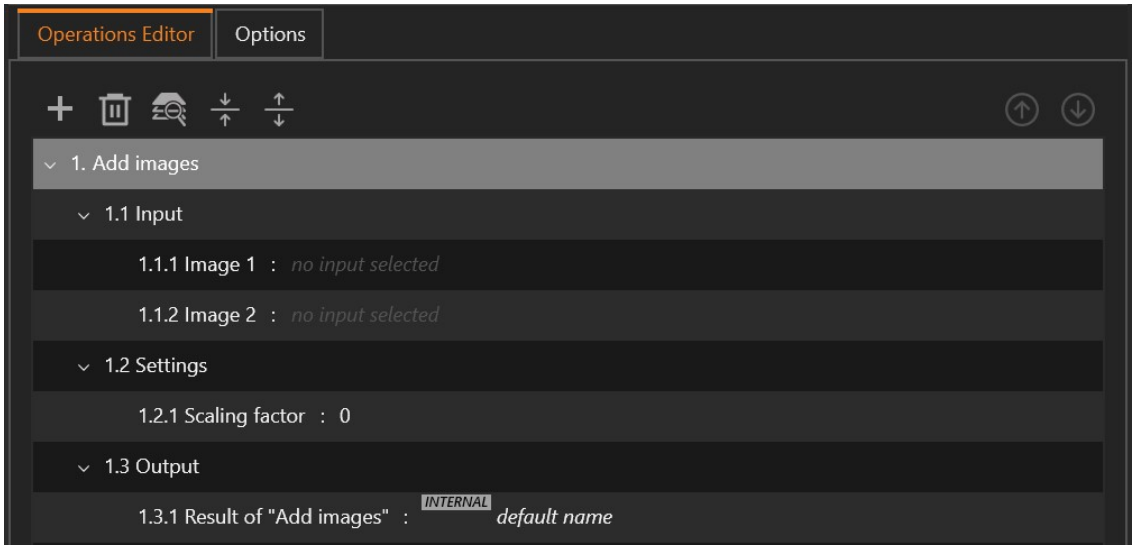


Figure 15 – Add Images

Where it can be found:

Basic Operations Module ► Arithmetic ► Add images

Description:

Add images is an arithmetic operation available in Basic Operations Module.

This operation adds the corresponding pixel of two images and places the result in the corresponding element of the output (performs the pixel-wise addition of two images).

$$Result = Image_1 + Image_2$$

Parameters:

- 1.1 Image 1 - the first input image
- 1.2 Image 2 - the second input image. Must have the same number of channels (1, 3) and the same data type (8-bit, 16-bit) as the first input image;
- 1.3 Scaling factor - scales the result by multiplying with 2^{value} (default = 0).
- 1.4 Result of "…" - the result of the operation

Effect:

- Combines the information from two images.

Example:

- Input:

Image 1 is a fluorescence grayscale image (8-bit, 1-channel) representing the DAPI channel of an UV irradiated skin sample (see below).

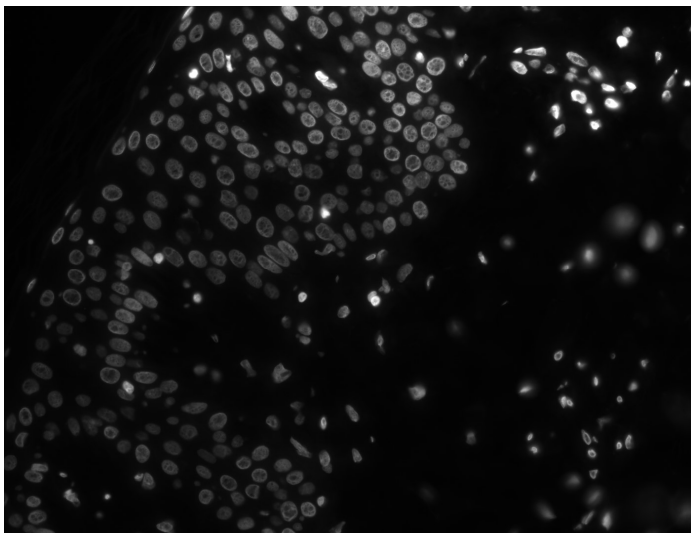


Figure 16 – UV irradiated skin sample – DAPI channel

Image 2 is a fluorescence grayscale image (8-bit, 1-channel) representing the Rhodamine channel of an UV irradiated skin sample (figure 3-7).

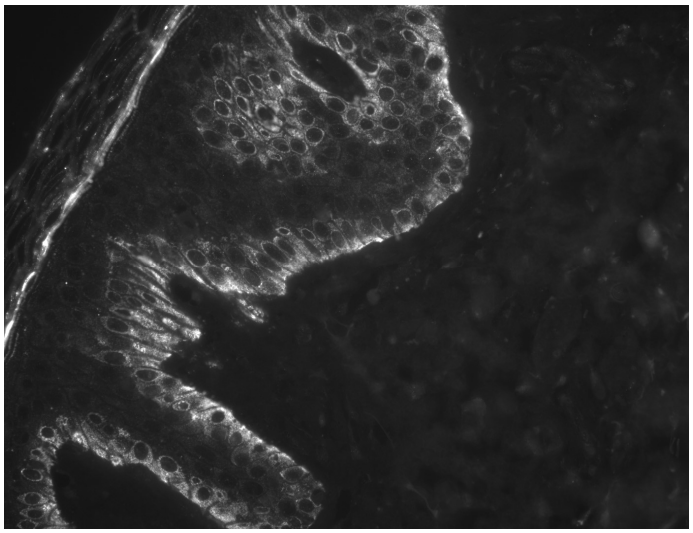


Figure 17 – UV irradiated skin sample – Rhodamine channel

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a grayscale image (8-bit, 1-channel), matching the two inputs.

Note	The two images used as input must have same bit-depth (8-bit / 16-bit) and the same number of channels (1-channel / 3-channels).

Figure 18 – Add Images engine settings

Output:

The result combines the information from the 2 input images, by summing corresponding pixel intensities.

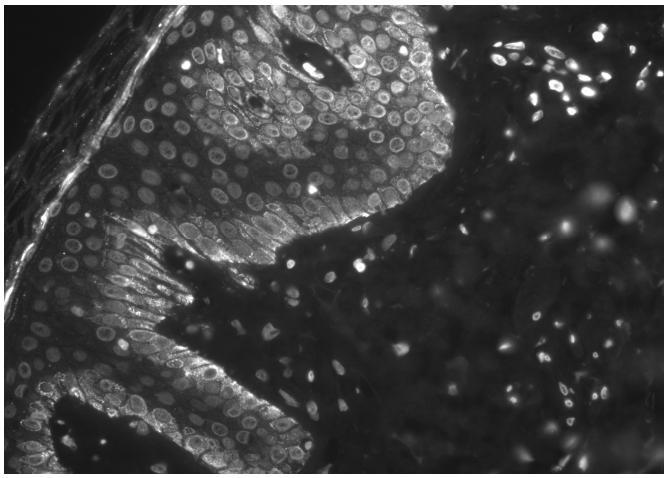


Figure 19 – Results of Add images engine

Add value

Figure 20 – Add value

Where it can be found:

Basic Operations Module ► Arithmetic ► Add value

Description:

Add value is an arithmetic operation available in Basic Operations Module.

This operation adds a numerical value to each pixel of the image and places the result in the corresponding element of the output.

$$Result = Image + value$$

Parameters:

- 1.1 Image - the input image
- 1.2. Scaling factor - scales the result by multiplying with (default = 0).
- 1.3 Value - the value added to each pixel from the image.
- 1.4 Result of "…" - the result of the operation

Effect:

Brightens (increases the intensity of) a grayscale image.

Example:

- Input:

The image is a fluorescence grayscale image (8-bit, 1-channel) representing the Rhodamine channel of an UV irradiated skin sample.

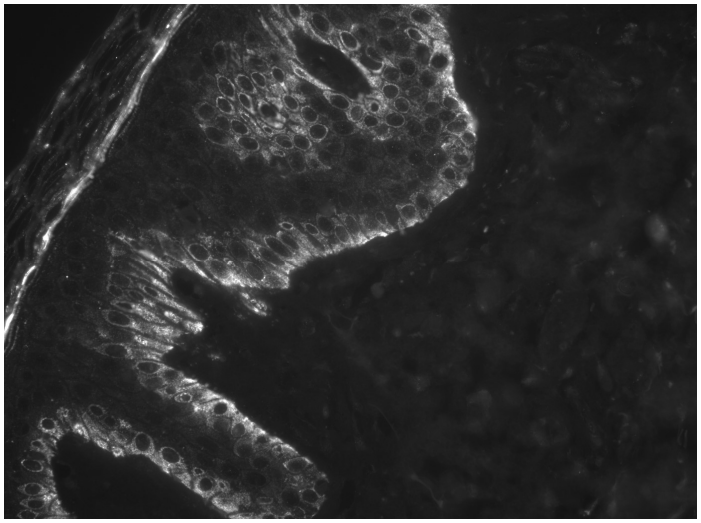


Figure 21 – UV irradiated skin sample – Rhodamine channel

- Engine settings:

Figure below shows the engine settings used for this example.

A value of 100 is added to the input image.

The engine's result is a grayscale image (8-bit, 1-channel), matching the input.

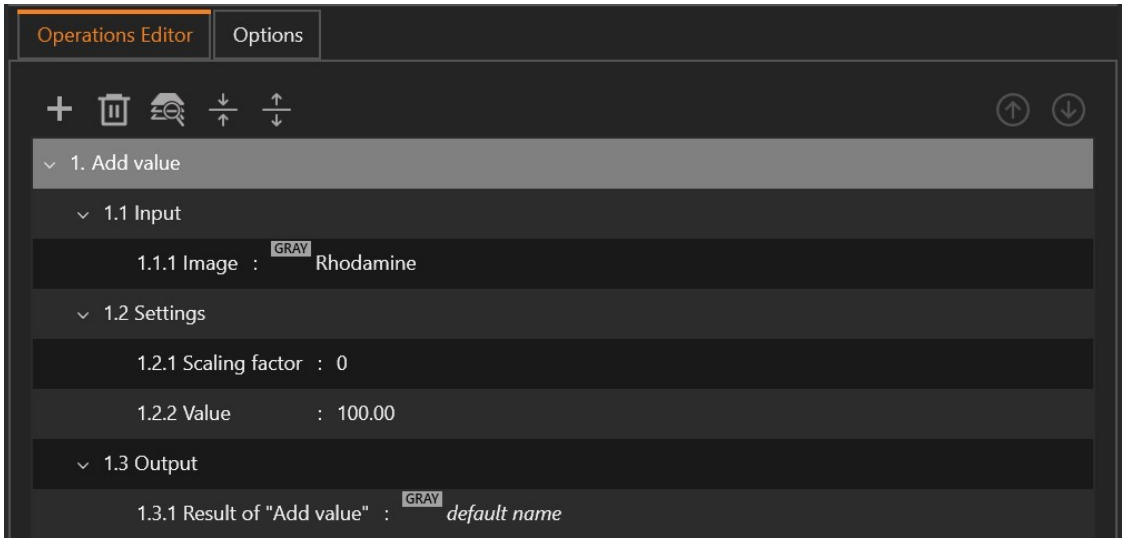


Figure 22 – Engine settings

- Output:

The result has the brightness increased by 100 compared with the input image.

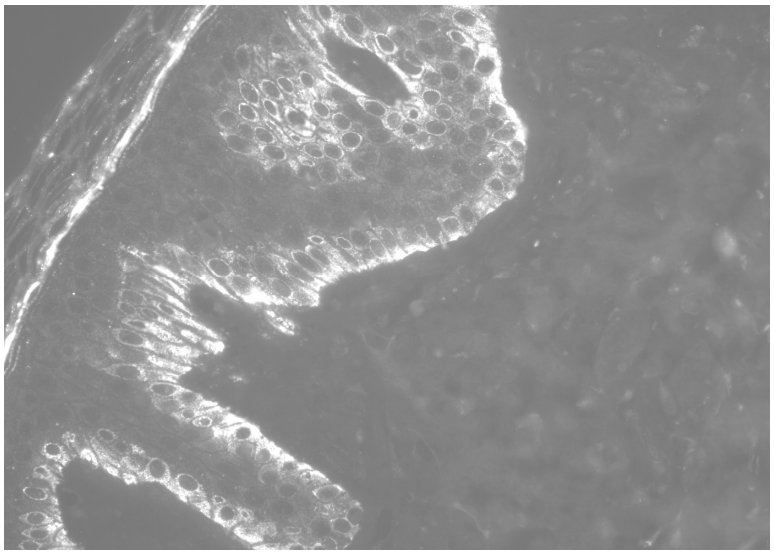


Figure 23 – Result of Add Value engine (+ 100)

Add value (RGB)

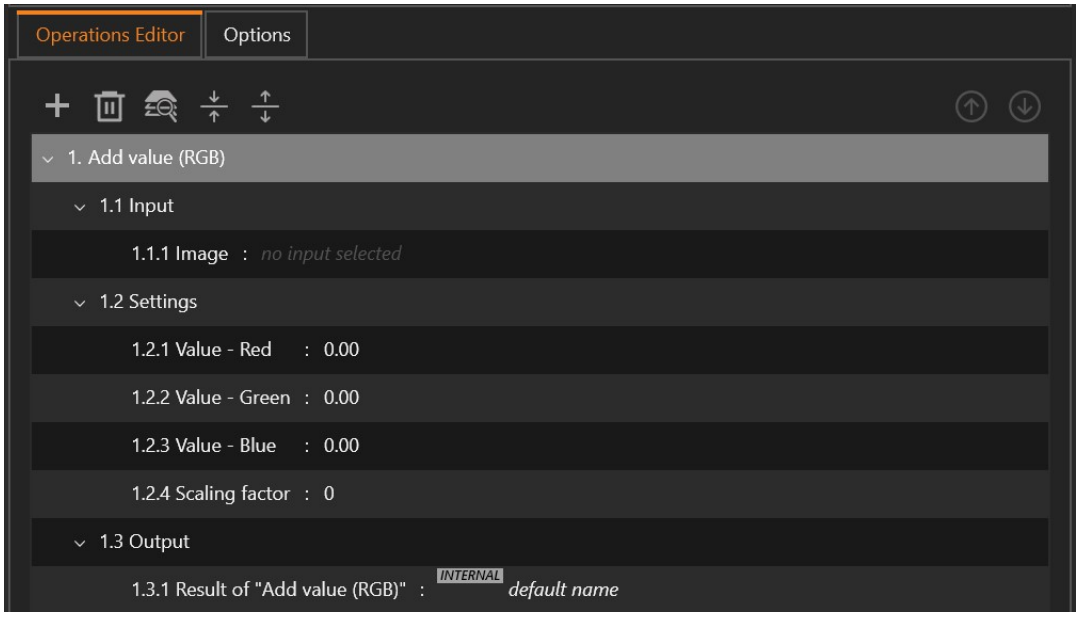


Figure 24 – Add value (RGB)

Where it can be found:

Basic Operations Module ► Arithmetic ► Add value (RGB)

Description:

Add value (RGB) is an arithmetic operation available in Basic Operations Module.

This operation adds the components of a color value (red, green, blue) to each pixel channel of the image and places the result in the corresponding element of the output.

$$\boldsymbol{Result}\,\langle r|g|b\rangle = \boldsymbol{Image}\,\langle r|g|b\rangle + \boldsymbol{value}\,\langle r|g|b\rangle$$

Parameters:

- 1.1 Image - the input image
- 1.2 Value - Red - the value added to the Red channel of the input image
- 1.3 Value - Green - the value added to the Green channel of the input image
- 1.4 Value - Blue - the value added to the Blue channel of the input image
- 1.5 Scaling factor - scales the result by multiplying with 2^{value} (default = 0).
- 1.6 Result of "…" - the result of the operation

Effect:

Brightens (increases the intensity of) a color image.

Example:

- Input:

The image is a brightfield color image (8-bit, 3-channel) representing a small region from a colon sample (figure below).

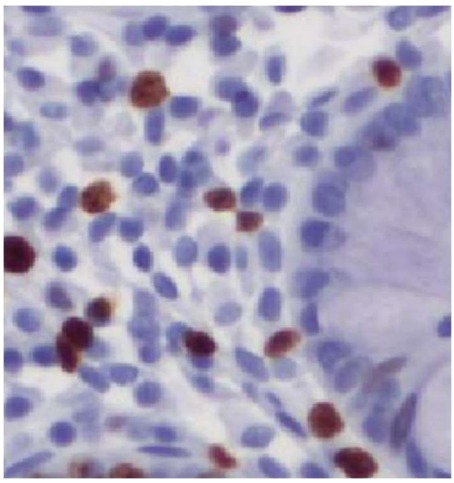


Figure 25 – Colon sample

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a color image (8-bit, 3-channel), matching the input.

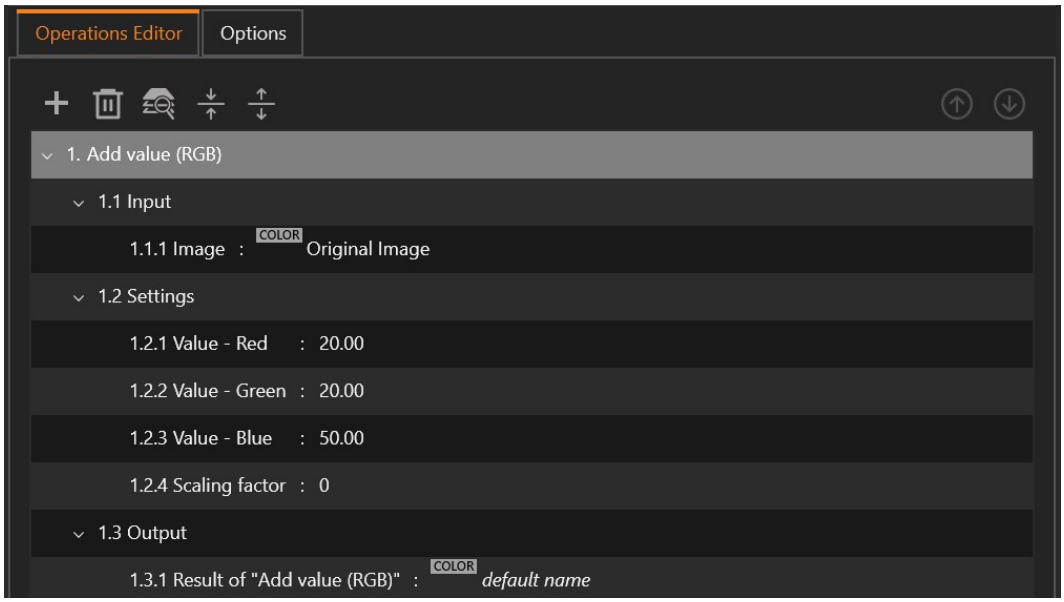


Figure 26 – Example of parameter values

- Output:

The result independently increases the brightness of the 3 channels of the input image (R=20 / G=20 / B=50).

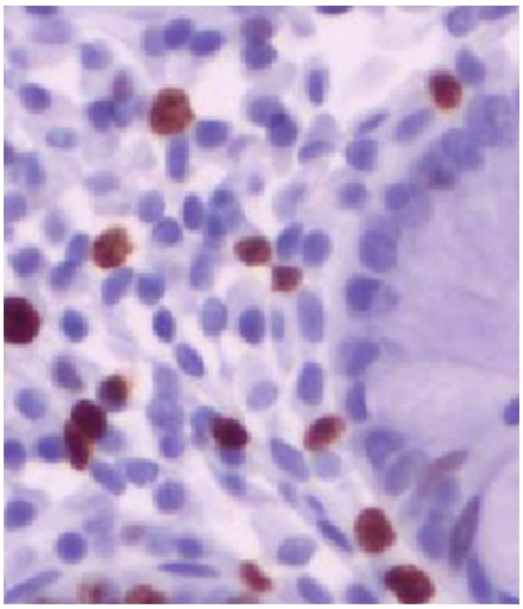


Figure 27 – Result of Add value (RGB) (R=20, G=20, B=50)

Add weighted images

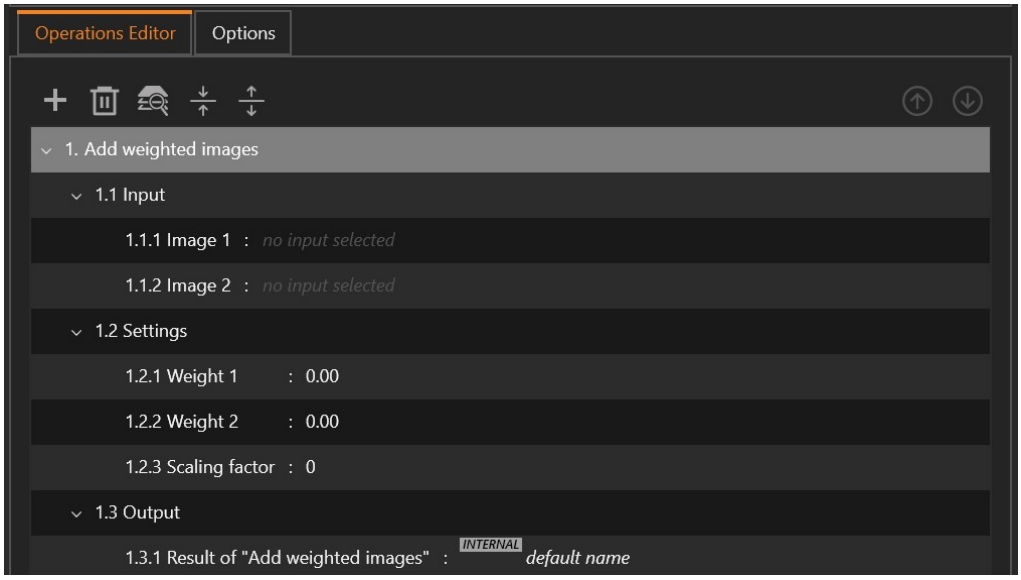


Figure 28 – Add Weighted Images

Where it can be found:

Basic Operations Module ► Arithmetic ► Add weighted images

Description:

Add weighted images is an arithmetic operation available in Basic Operations Module.

This operation adds the corresponding pixel of Image 1, multiplied by a weight factor weight 1, with the pixels of Image 2, multiplied by a weight factor weight 2, and places the result in the corresponding element of the output image (performs pixel-wise weighted addition of two images).

$$Result = weight_1 * Image_1 + weight_2 * Image_2$$

This operation is useful when the information from two images must be combined and the signal in one of the images is weaker or less important that the other one.

Parameters:

- 1.1 Image 1 - the first input image;
- 1.2 Image2 - the second input image;
- 1.3 Weight 1 - the weight value for the first image (default = 0);

- 1.4 Weight 2 - the weight value for the second image (default = 0);
- 1.5 Scaling factor - scales the result by multiplying with 2^{value} (default = 0).
- 1.6 Result of "…" - the result of the operation

Effect:

Combines the information from two images.

Example:

- Input:

In the image below there is a fluorescence grayscale image (8-bit, 1-channel) representing the DAPI channel of an UV irradiated skin sample.

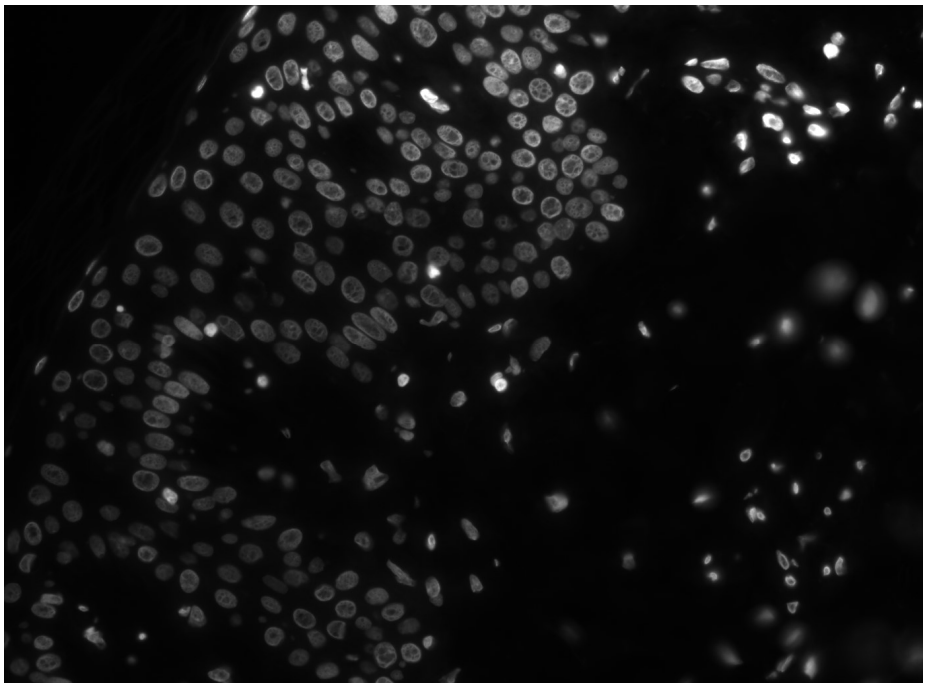


Figure 29 – UV irradiated skin sample – DAPI channel

In the image below there is a fluorescence grayscale image (8-bit, 1-channel) representing the Rhodamine channel of a UV irradiated skin sample.

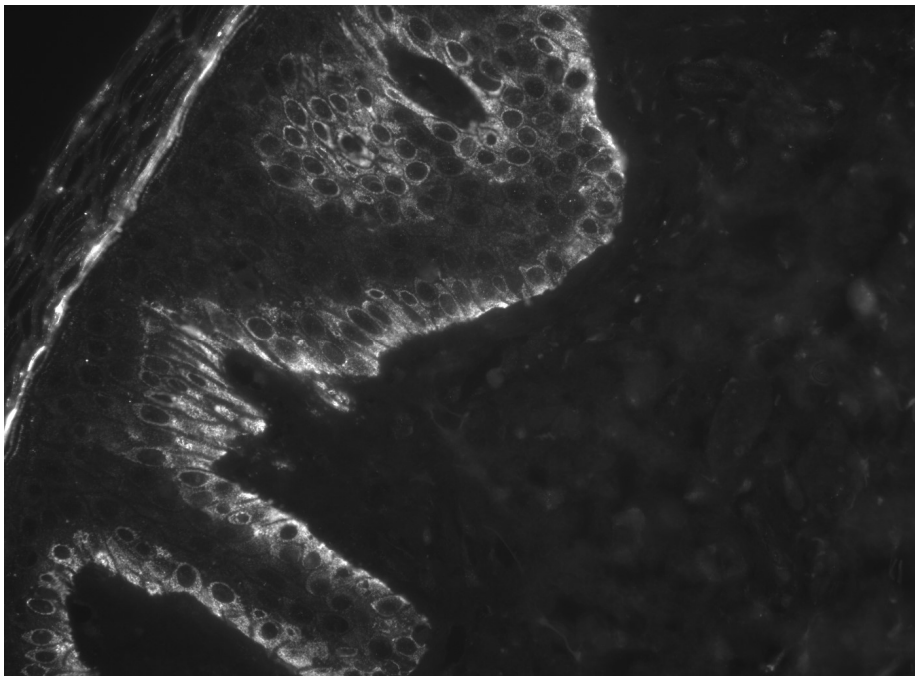


Figure 30 – UV irradiated skin sample - Rhodamine channel

- Engine settings:

Three examples are considered:

1. DAPI's weight = 1 and Rhodamine's weight = 1 (ratio 1:1)
2. DAPI's weight = 2 and Rhodamine's weight = 1 (ratio 2:1)
3. DAPI's weight = 1 and Rhodamine's weight = 2 (ratio 1:2)

The engine's result is a grayscale image (8-bit, 1-channel), matching the two inputs.

Note	If a negative weight is used for one of the images, a subtraction will take place.

Note	The two images used as input must have same bit-depth (8-bit / 16-bit) and the same number of channels (1-channel / 3-channels).

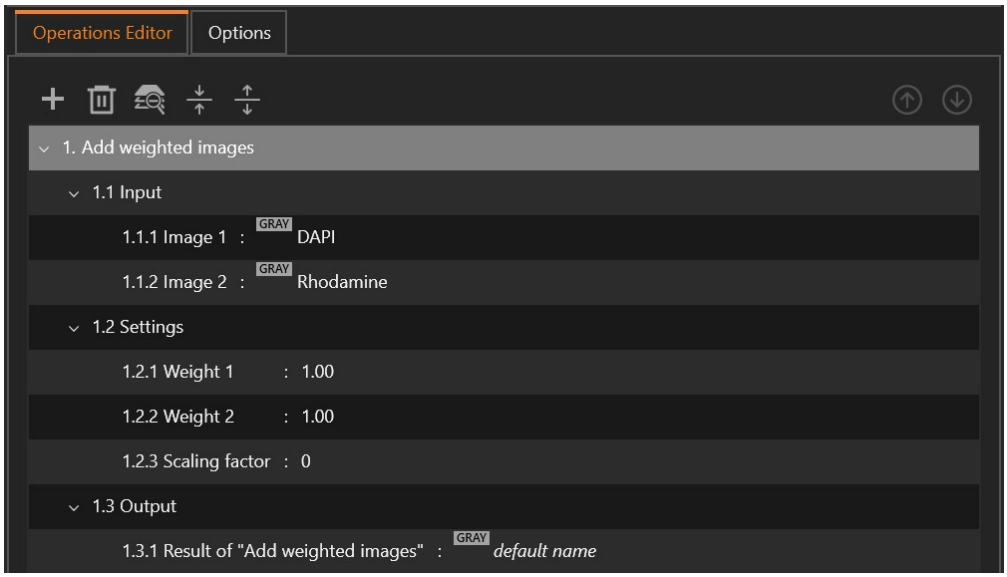


Figure 31 – Engine settings (example 1)

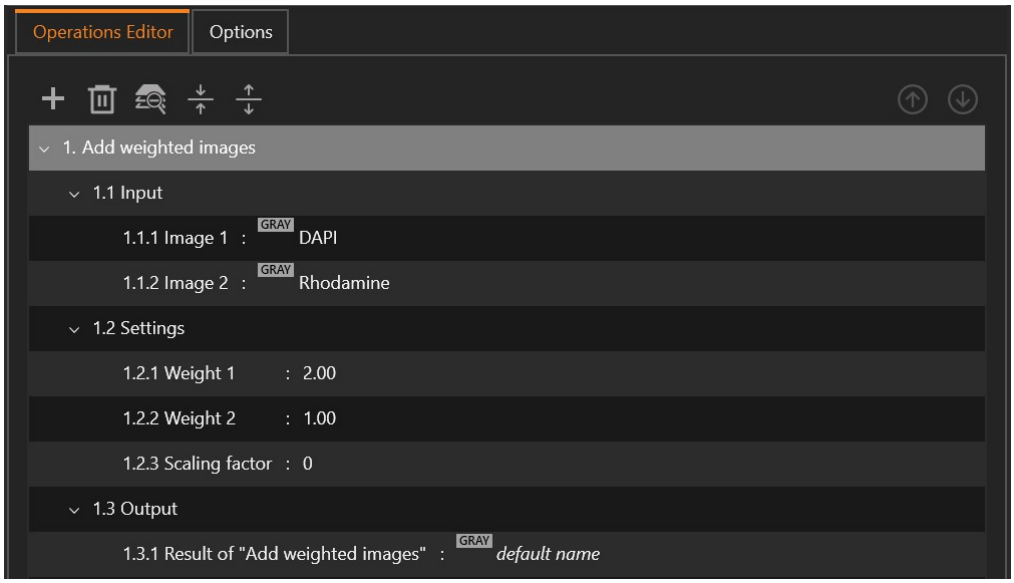


Figure 32 – Engine settings (example 2)

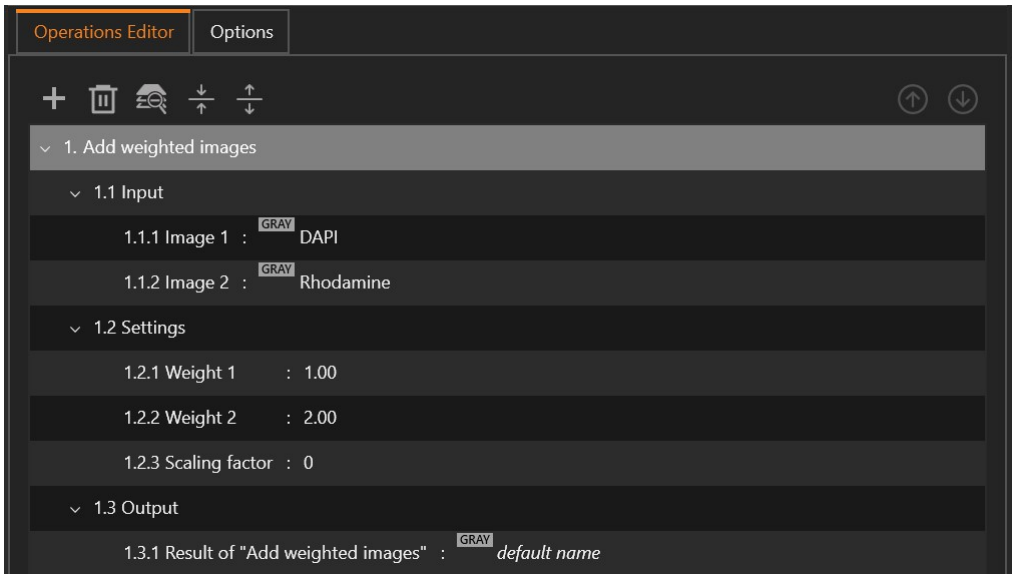


Figure 33 – Engine settings (example 3)

- Output:

The results combine the information from the 2 input images, by summing the corresponding pixel intensities using specified weights:

- Example 1 – 1.0 x DAPI + 1.0 x Rhodamine
- Example 2 – 2.0 x DAPI + 1.0 x Rhodamine
- Example 3 – 1.0 x DAPI + 2.0 x Rhodamine

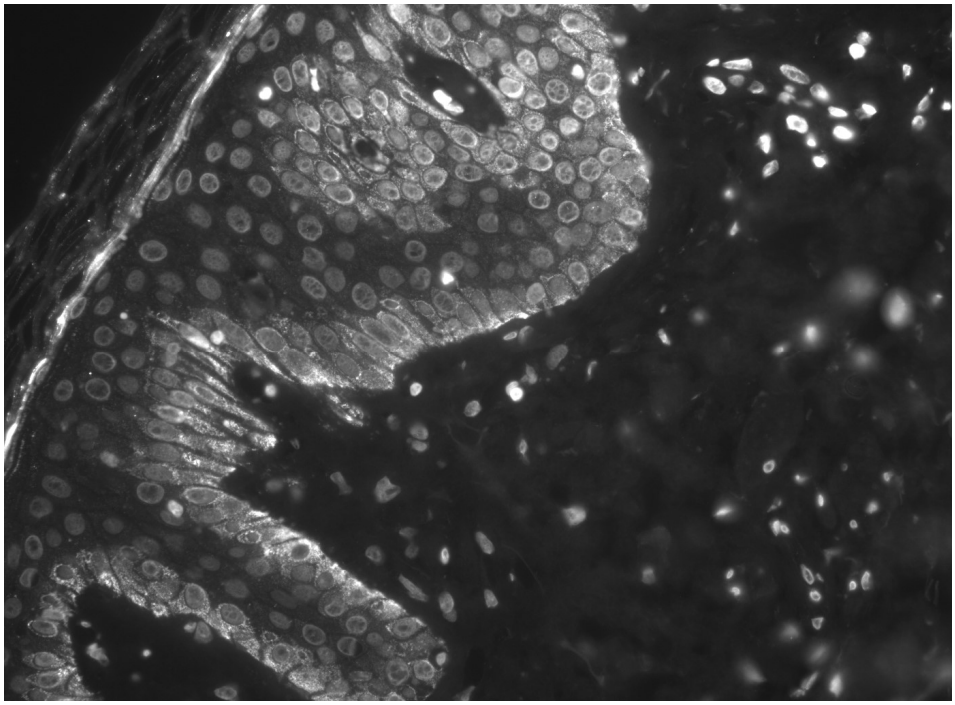


Figure 34 – Result of Add weighted images (example 1)

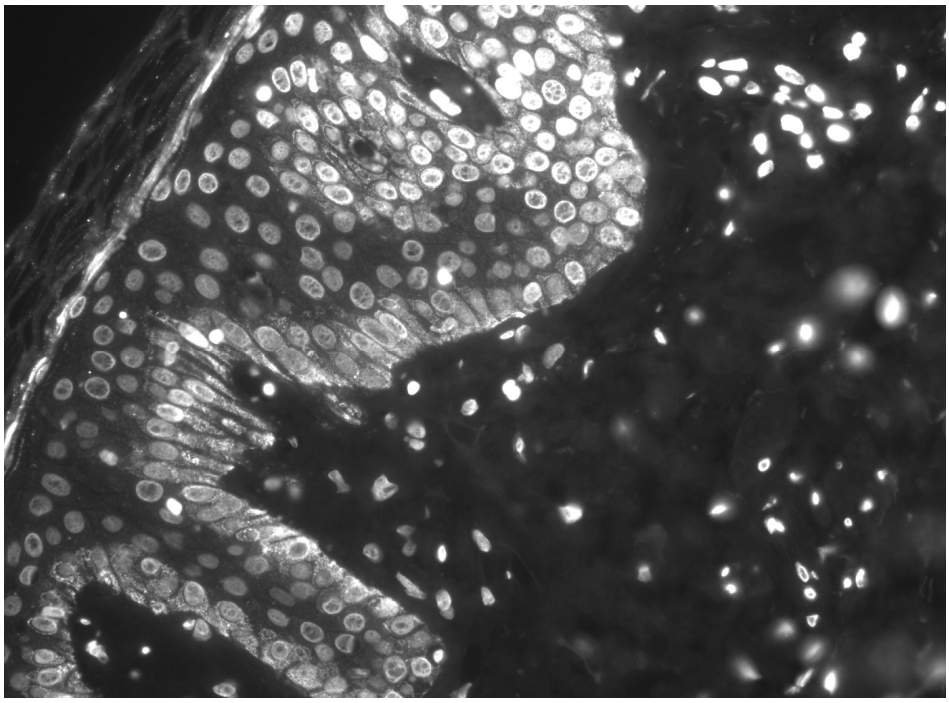


Figure 35 – Result of Add weighted images (example 2)

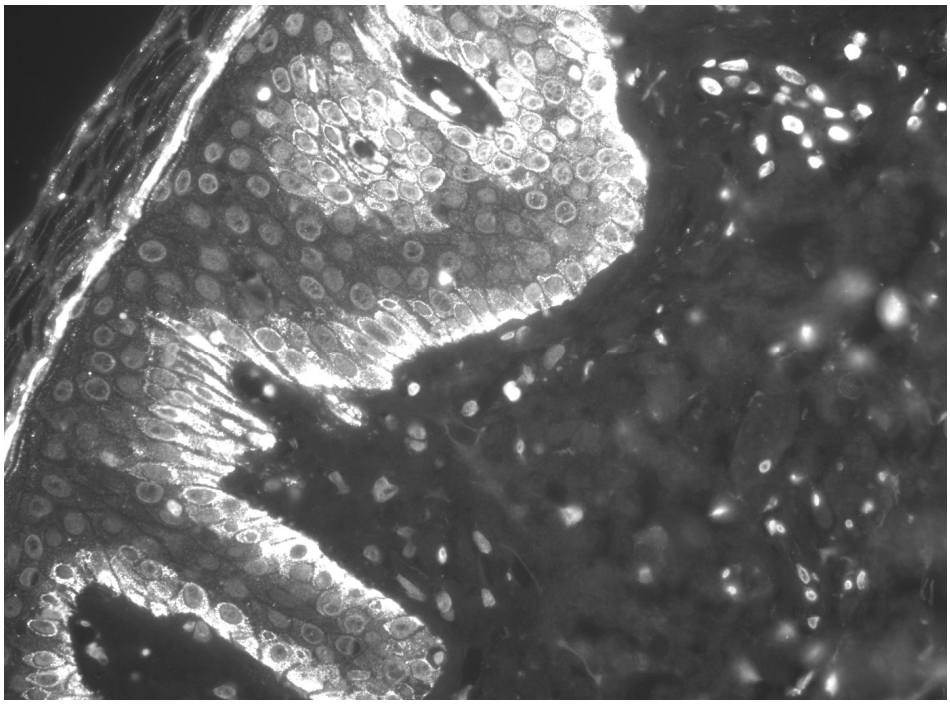


Figure 36 – Result of Add weighted images (example 3)

Complement image

Figure 37 – Complement image

Where it can be found:

Basic Operations Module ► Arithmetic ► Complement Image

Description:

Complement image is an arithmetic operation available in Basic Operations Module.

This operation computes the complement of each pixel from the image and places the result in the corresponding element of the output. The complement is computed in by subtracting the pixel value from the maximum value allowed (in case of 8-bit images, the maximum value allowed is 255).

$$Result = 255 - Image$$

Note	For a mask image (binary image with black and white areas), the black and white areas are inverted (black areas become white and vice-versa).

Parameters:

- 1.1 Image - the input image.
- 1.2 Result of "…" - the result of the operation.

Effect:

Invert the brightness of a grayscale image.

Example:

- Input:

The image is a fluorescence grayscale image (8-bit, 1-channel) representing the DAPI channel of an UV irradiated skin sample.

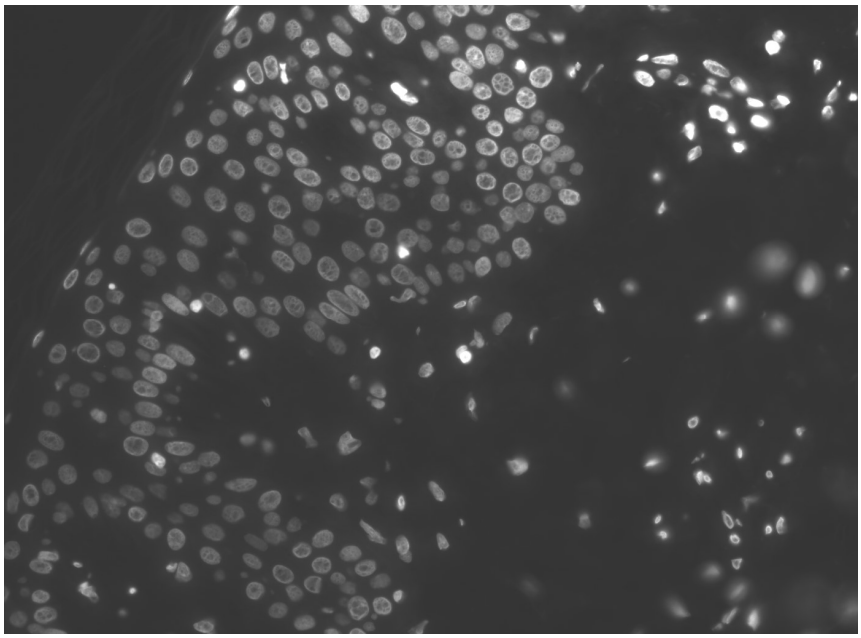


Figure 38 – Skin UV Treated - DAPI

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a grayscale image (8-bit, 1-channel), matching the input.

Figure 39 – Engine settings

- Output:

The result has all intensities of the input image complemented, “simulating” a brightfield image.

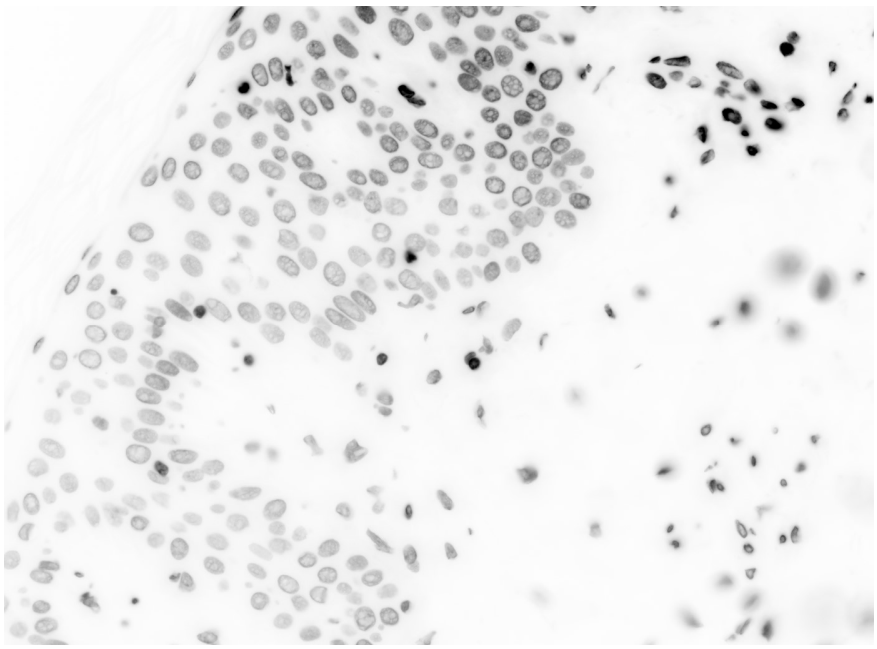


Figure 40 – Result of Complement image engine

Divide by value

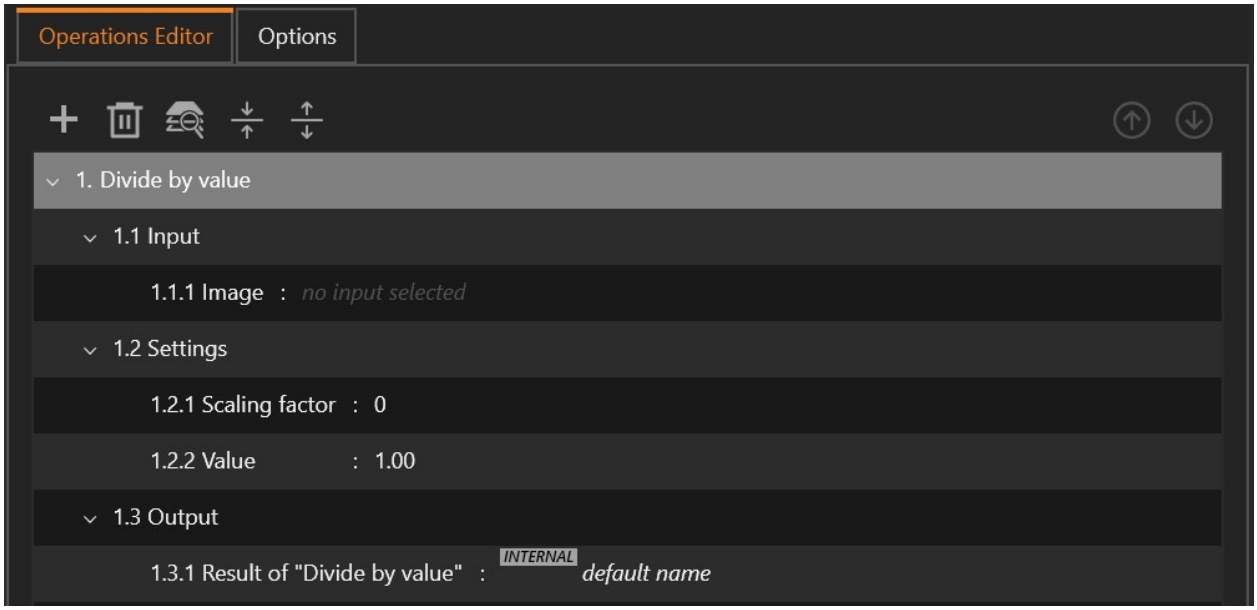


Figure 41 – Divide by value

Where it can be found:

Basic Operations Module ► Arithmetic ► Divide by value

Description:

Divide by value is an arithmetic operation available in Basic Operations Module.

This operation divides each pixel of the image by a numerical value and places the result in the corresponding element of the output.

$$Result = Image / value$$

Parameters:

- 1.1 Image - the input image
- 1.2. Scaling factor - scales the result by multiplying with 2^{value} (default = 0).
- 1.3 Value - the value used to divide each pixel from the image
- 1.4 Result of "…" - the result of the operation

Effect:

Changes the intensity of a grayscale image.

Example:

- Input: grayscale image representing the nuclear marker in a colon sample.

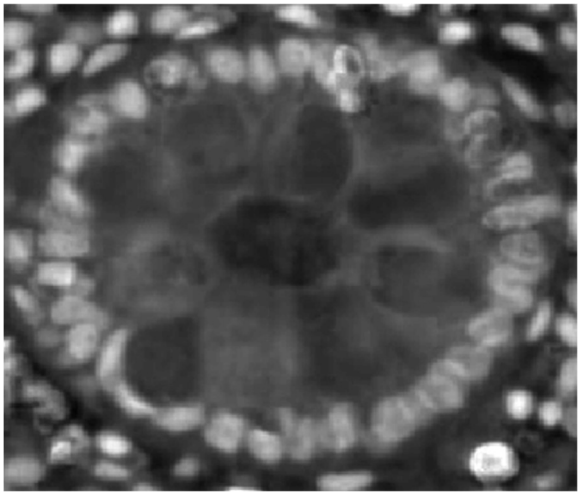


Figure 42 – Colon sample – nuclear marker

- Engine settings:

Figure below shows the engine settings used for this example.

All intensity values are divided by 2.

The engine's result is a grayscale image (8-bit, 1-channel), matching the two inputs.

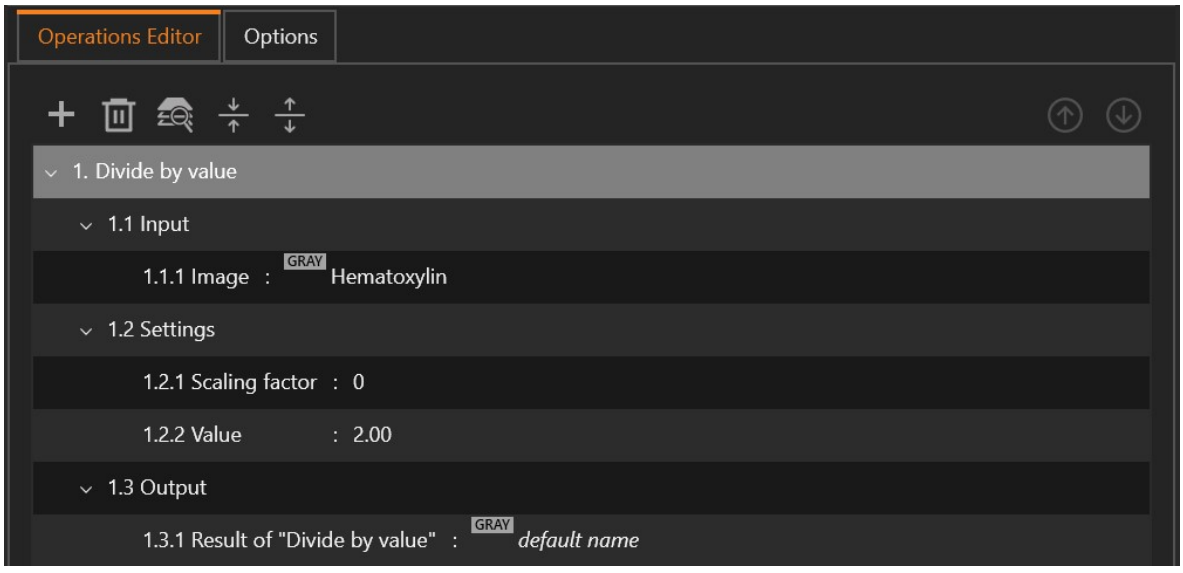


Figure 43 – Engine settings

- Output:

The result has the brightness lowered by half, compared with the input image.

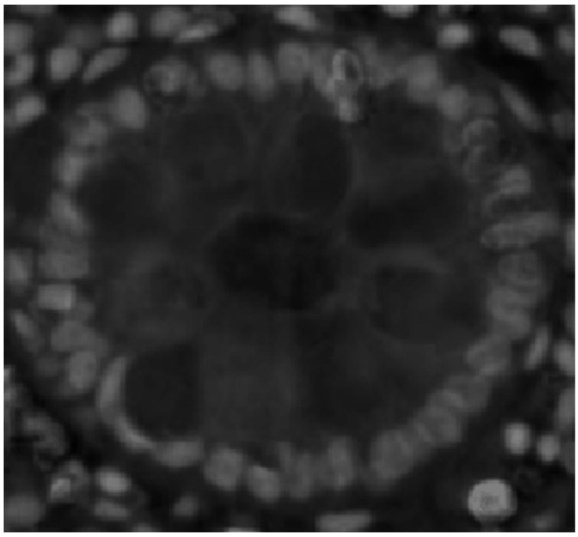


Figure 44 – Result of Divide by value engine

Divide by value (RGB)

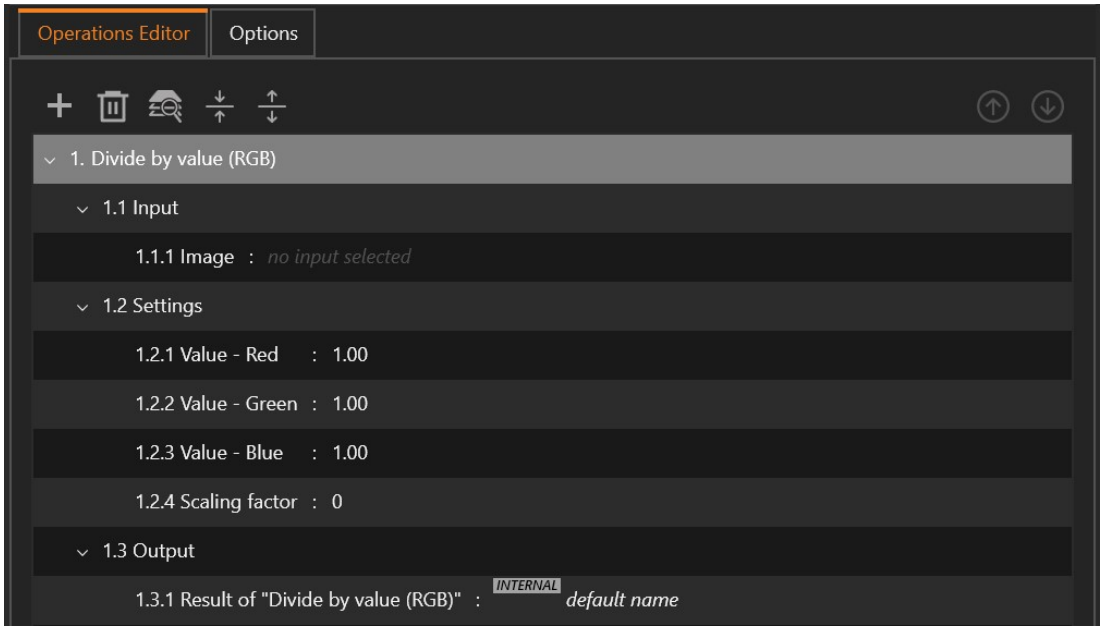


Figure 45 – Divide Constant RGB

Where it can be found:

Basic Operations Module ► Arithmetic ► Divide by value (RGB)

Description:

Divide by value (RGB) is an arithmetic operation available in Basic Operations Module.

This operation divides each pixel channel of the image by the corresponding component of a color value (red, green, blue) and places the result in the corresponding element of the output.

$$Result\,\langle r|g|b\rangle = Image\,\langle r\,|\,g\,|\,b\rangle \;/\; value\,\langle r|g|b\rangle$$

Parameters:

1.1 Image - the input image

1.2 Value - Red - the value used to divide the Red channel of the input image

1.3 Value - Green - the value used to divide the Green channel of the input image

1.4 Value - Blue - the value used to divide the Blue channel of the input image

1.5 Scaling factor - scales the result by multiplying with 2value (default = 0)

1.6 Result of "…" - the result of the operation

Effect:

Changes the intensity of a color image.

Example:

- Input:

The image below is a brightfield color image (8-bit, 3-channel) representing immunohistochemistry showing E-Cadherin in epithelial tissue.

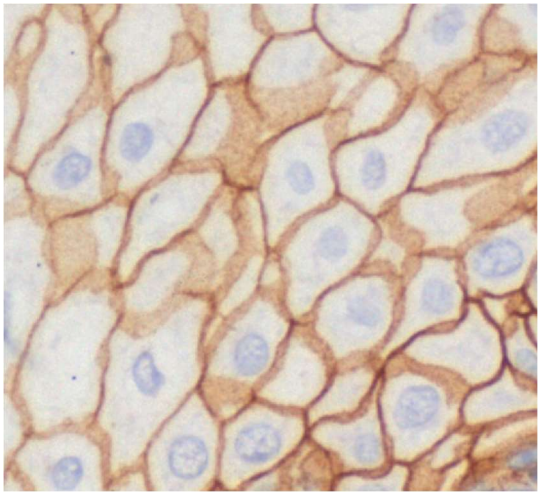


Figure 46 – Original image

- Engine settings:

Figure below shows the engine settings used for this example.

The intensity values for all channels are divided by 2.

The engine's result is a color image (8-bit, 3-channel), matching the two inputs

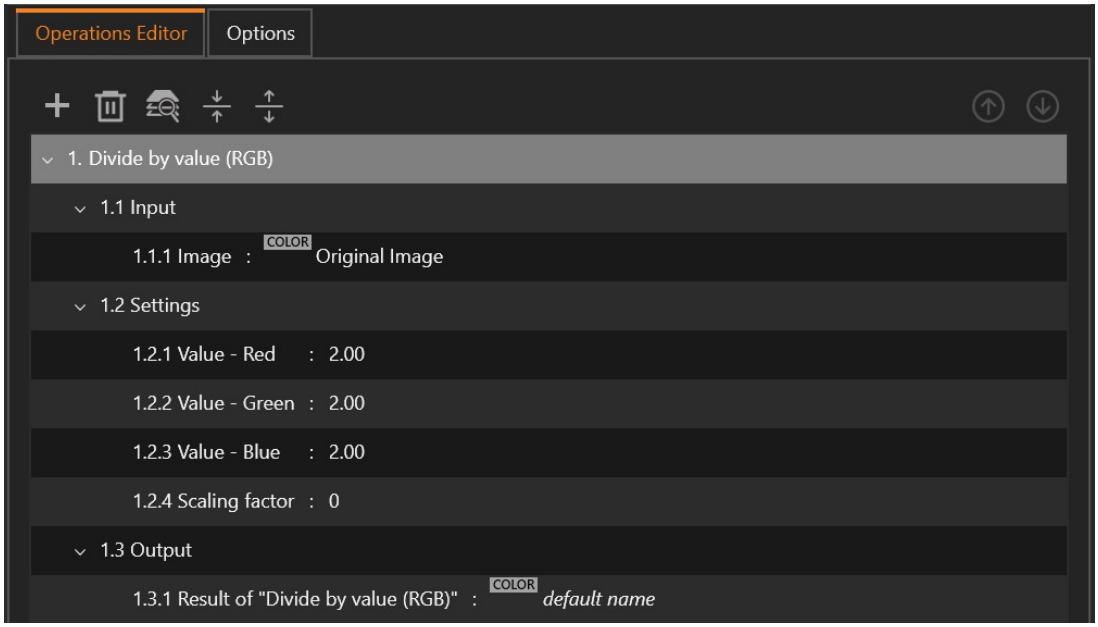


Figure 47 – Engine settings

- Output:

The result has the brightness (of all channels) lowered by half, compared with the input image.

The brightness of all channels in the resulting image is lowered by half (compared with the input image).

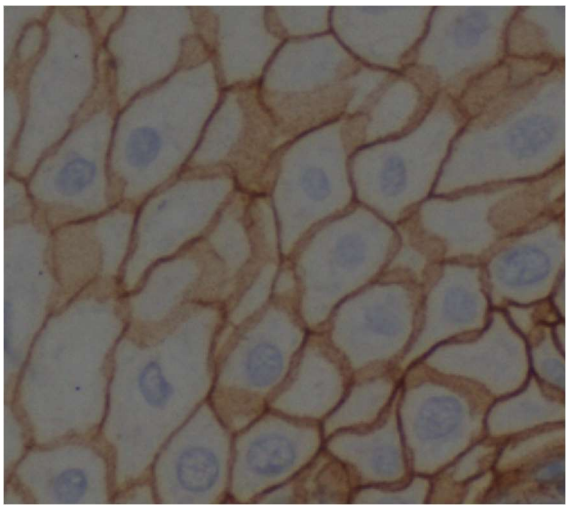


Figure 48 – Result of Divide by value (RGB) (R = 2, G = 2, B = 2)

Divide images

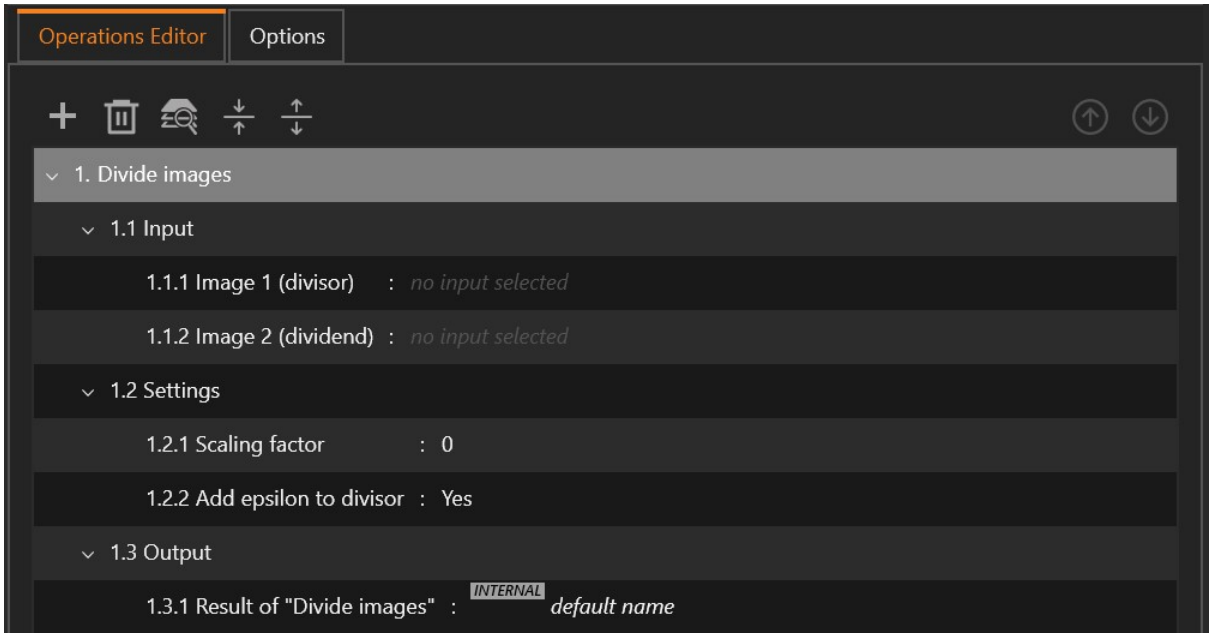


Figure 49 – Divide images

Where it can be found:

Basic Operations Module ► Arithmetic ► Divide images

Description:

Divide images is an arithmetic operation available in Basic Operations Module.

This operation divides each pixel from Image 2 by the corresponding pixel from Image 1 and places the result in the corresponding element of the output (performs the pixel-wise division of two images).

$$\boldsymbol{Result} = \frac{\boldsymbol{Image_2}}{\boldsymbol{Image_1}}$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \div \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} = \begin{pmatrix} a_{11}/\,b_{11} & a_{12}/\,b_{12} & a_{13}/\,b_{13} \\ a_{21}/\,b_{21} & a_{22}/\,b_{22} & a_{23}/\,b_{23} \\ a_{31}/b_{31} & a_{32}/\,b_{32} & a_{33}/\,b_{33} \end{pmatrix}$$

Parameters:

- 1.1 Image 1 (divisor) - the first input image representing the divisor
- 1.2 Image 2 (dividend) - the second input image representing the dividend

- 1.3 Scaling factor - the scales the result by multiplying with 2^{value} (default = 0)
- 1.4 Add epsilon to divisor - enables / disable the addition of a very small value to the divisor to avoid division by 0 (default = Yes).
- 1.5 Result of "…" - the result of the operation

Effect:

Divides two images.

Example:

Although it is not common to divide one image by another, there are some rare cases where this engine might be useful, e.g. to modify the input image brightness using a gradient image (generated from a distance map).

Exponential (Exp)

Figure 50 – EXP

Where it can be found:

Basic Operations Module ► Arithmetic ► Exponential (Exp)

Description:

Exponential (Exp) is an arithmetic operation available in Basic Operations Module.

This operation computes the exponential for each pixel of the image and places the result in the corresponding element of the output.

$$Result = e^{Image}$$

$$\exp\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} e^{a_{11}} & e^{a_{12}} & e^{a_{13}} \\ e^{a_{21}} & e^{a_{22}} & e^{a_{23}} \\ e^{a_{31}} & e^{a_{32}} & e^{a_{33}} \end{pmatrix}$$

Parameters:

- 1.1 Image - the input image
- 1.2 Scaling factor - scales the result by multiplying with 2^{value} (default = 0)
- 1.3 Result of "…" - the result of the operation

Effect:

Exponentially change the intensity of a grayscale image.

Example:

- Input: The image below is a fluorescence grayscale image (8-bit, 1-channel) representing the SpGold channel of a colorectal cancer sample.

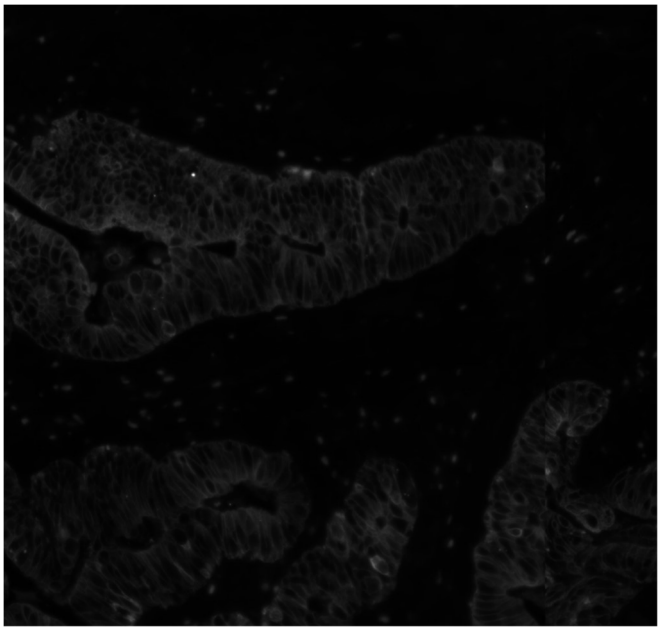


Figure 51 – Colorectal cancer sample – SpGold channel

- Engine settings:

Figure below shows the engine settings used for this example.

The input image brightness is increased exponentially. Due to the nature of the exponential function, the resulting intensity values are much higher than 8-bit limit (255). Therefore, the result is scaled down by 2^2 (divided by 4), to retain the high intensity details and discard low intensity details.

The engine's result is a grayscale image (8-bit, 1-channel), matching the input.

Note	A positive / negative value for scaling factor scales down / up the resulting value(s).

Figure 52 – Engine settings

- Output:

The result has the brightness increased exponentially compared with the input image.

Figure 53 – Result of Exponential (Exp) engine

Logarithm (Ln)

Figure 54 – LN

Where it can be found:

Basic Operations Module ► Arithmetic ► Logarithm (Ln)

Description:

Logarithm (Ln) is an arithmetic operation available in Basic Operations Module.

This operation computes the natural logarithm for each pixel of the image and places the result in the corresponding element of the output.

$$Result = Ln(Image)$$

$$\text{Ln}\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} Ln(a_{11}) & Ln(a_{12}) & Ln(a_{13}) \\ Ln(a_{21}) & Ln(a_{22}) & Ln(a_{23}) \\ Ln(a_{31}) & Ln(a_{32}) & Ln(a_{33}) \end{pmatrix}$$

Parameters:

- 1.1 Image - the input image
- 1.2 Scaling factor - scales the result by multiplying with 2^{value} (default = 0)
- 1.3 Result of "…" - the result of the operation

Effect:

Logarithmically changes the intensity of a grayscale image.

Example:

- Input:

Image below is a fluorescence grayscale image (8-bit, 1-channel) representing the SpGold channel of a colorectal cancer sample.

Figure 55 – Colorectal cancer sample – SpGold channel

- Engine settings:

Figure below shows the engine settings used for this example.

The input image brightness is decreased logarithmically. Due to the nature of the logarithm function, the resulting intensity values are too small to be distinguished, thus the result is scaled up by 2^4 (multiplied by 16), to amplify the values.

The engine's result is a grayscale image (8-bit, 1-channel), matching the input.

Note	A positive / negative value for scaling factor scales down / up the resulting value(s).

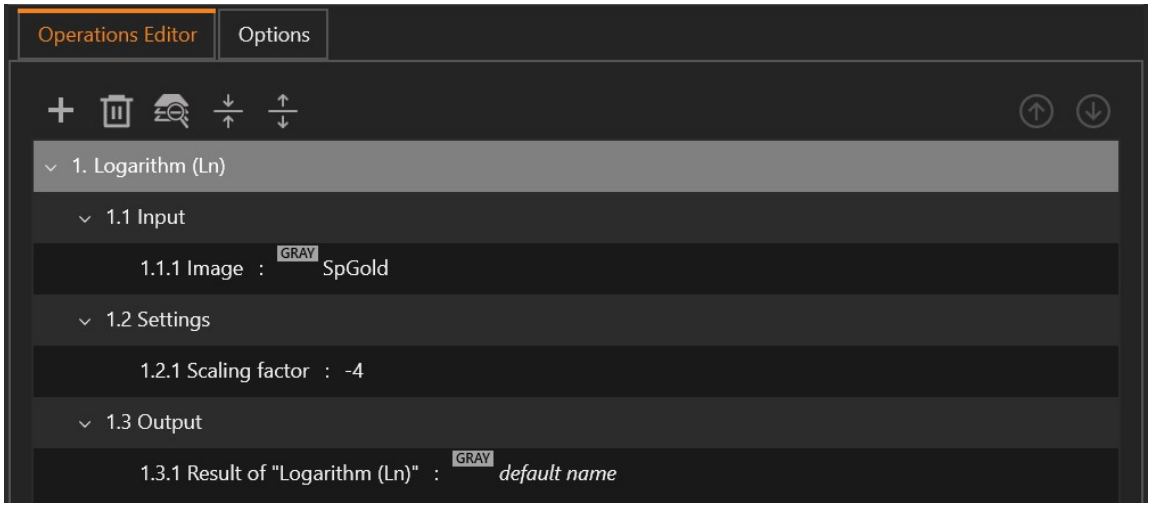


Figure 56 – Engine settings

- Output:

The result has the brightness decreased logarithmically compared with the input image.

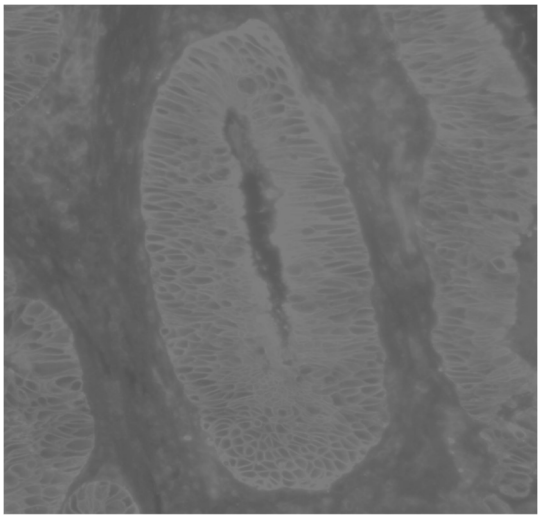


Figure 57 – Result of Logarithm (Ln) engine

Multiply images

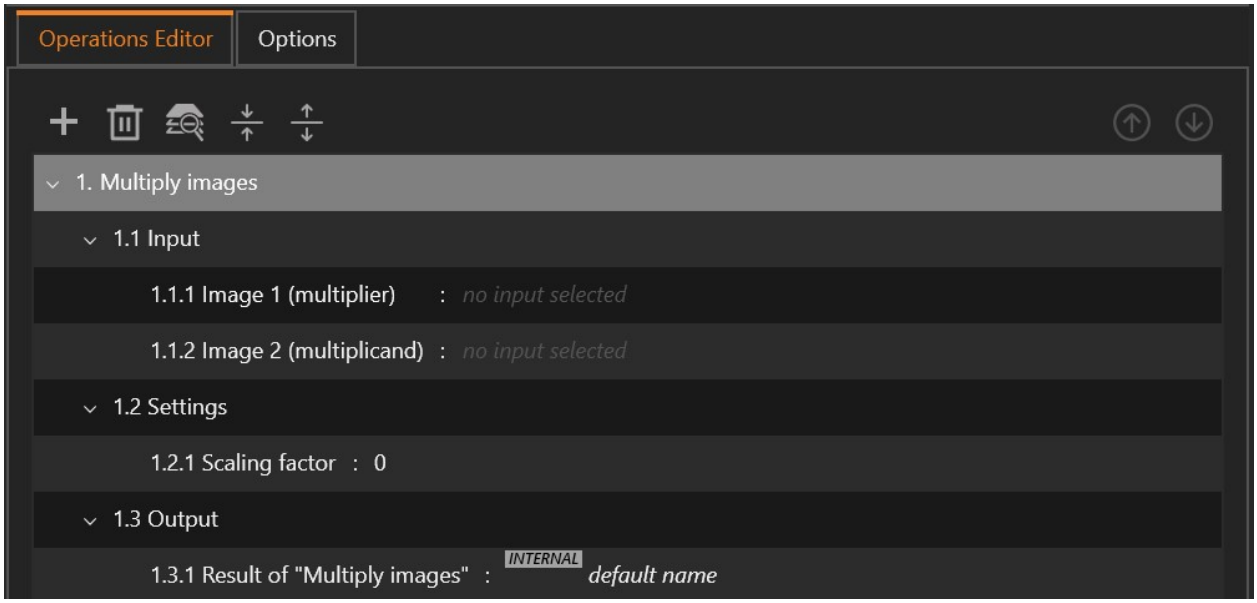


Figure 58 – Multiply images

Where it can be found:

Basic Operations Module ► Arithmetic ► Multiply images

Description:

Multiply images is an arithmetic operation available in Basic Operations Module.

This operation multiplies the pixel from Image 1 by the corresponding pixel from Image 2 and places the result in the corresponding element of the output (performs the pixel-wise multiplication of two images).

$$\boldsymbol{Result = Image_2 * Image_1}$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} * \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} = \begin{pmatrix} a_{11} * b_{11} & a_{12} * b_{12} & a_{13} * b_{13} \\ a_{21} * b_{21} & a_{22} * b_{22} & a_{23} * b_{23} \\ a_{31} * b_{31} & a_{32} * b_{32} & a_{33} * b_{33} \end{pmatrix}$$

Parameters:

- 1.1 Image 1 - the first input image representing the multiplier
- 1.2 Image 2 - the second input image representing the multiplicand
- 1.3 Scaling Factor - scales the result by multiplying with 2^{value} (default = 0)
- 1.4 Result of "…" - the result of the operation

Effect:

The pixel-wise multiplication of two images.

Example:

- Input:

Image below is a fluorescence grayscale image (8-bit, 1-channel) representing the SpGold channel of a colorectal cancer sample.

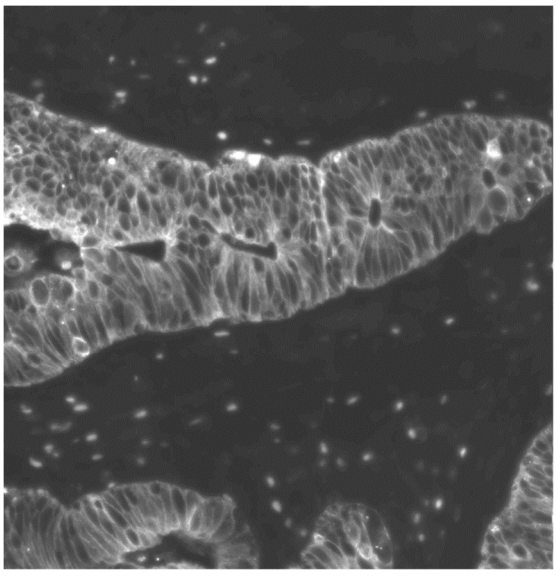


Figure 59 – Colorectal cancer sample - SpGold channel

Image below is a mask image (8-bit, 1-channel) representing the stained area.

Figure 60 – Mask image

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a grayscale image (8-bit, 1-channel), matching the two inputs.

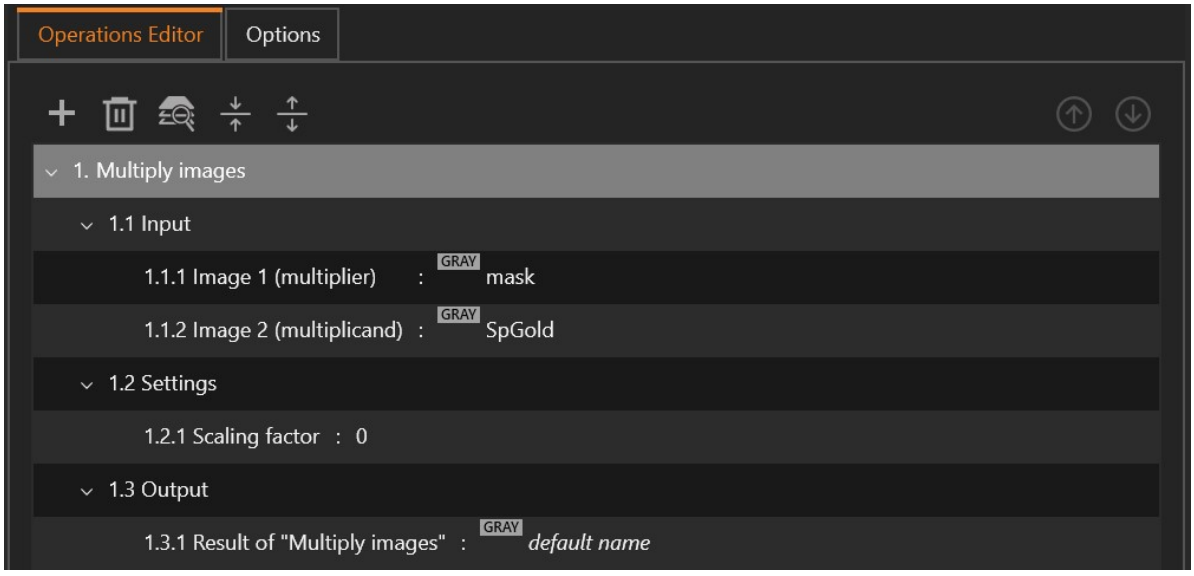


Figure 61 – Engine settings

- Output:

The result is the multiplication of corresponding pixel values from the two input images. In this example, the effect is the preservation of the intensity values indicated by the

mask's white area, while the other intensity values (corresponding to mask's black area) are set to 0.

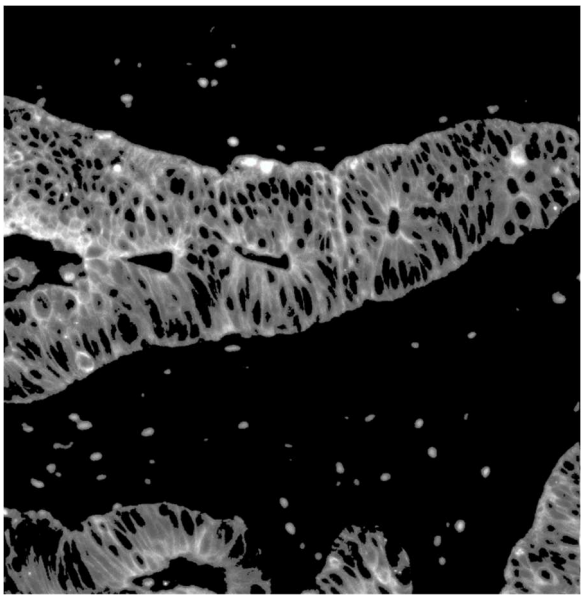


Figure 62 – Result of Multiply images engine

Multiply with value

Figure 63 – Multiply with value

Where it can be found:

Basic Operations Module ► Arithmetic ► Multiply with value

Description:

Multiply with value is an arithmetic operation available in Basic Operations Module.

This operation multiplies each pixel of the image by a numerical value and places the result in the corresponding element of the output.

Parameters:

- 1.1 Image - the input image
- 1.2. Scaling factor - scales the result by multiplying with 2^{value} (default = 0)
- 1.3 Value - the value used to multiply each pixel from the image
- 1.4 Result of "…" - the result of the operation

Effect:

Change the intensity of a grayscale image.

Example:

- Input: The image below is a fluorescence grayscale image (8-bit, 1-channel) representing the SpGold channel of a colorectal cancer sample.

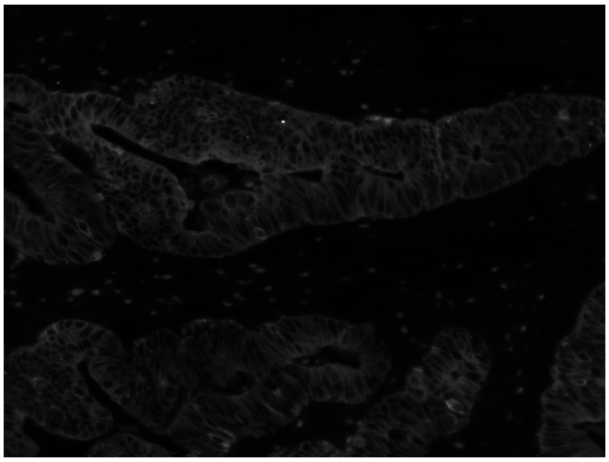


Figure 64 – Colorectal cancer sample – SpGold channel

- Engine settings:

Figure below shows the engine settings used for this example.

The brightness of the input image is increased 5 times.

The engine's result is a grayscale image (8-bit, 1-channel), matching the input.

Figure 65 – Engine settings

- Output:

The result has the brightness increased 5 times compared with the input image.

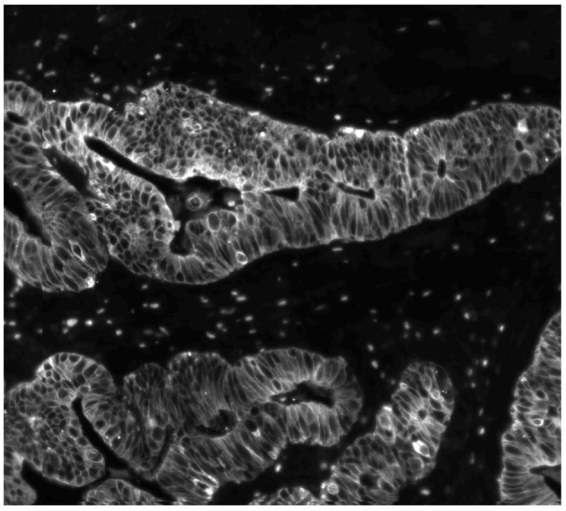


Figure 66 – Result of Multiply with value engine

Multiply with value (RGB)

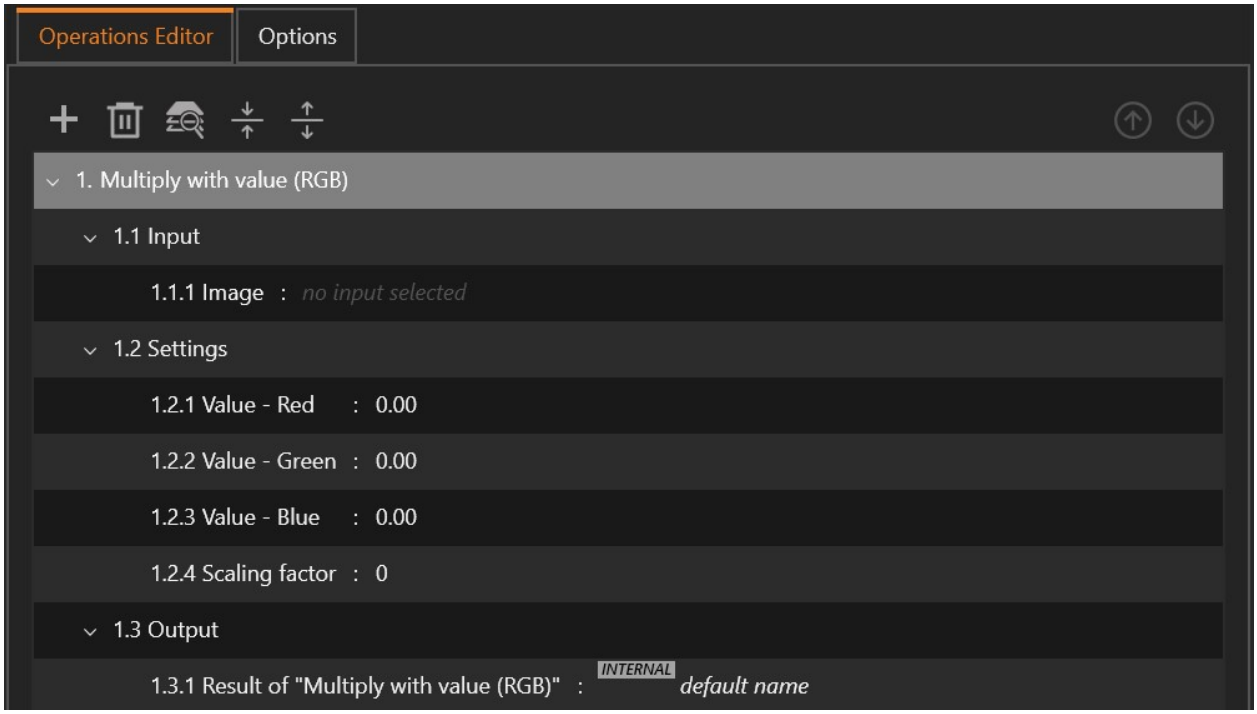


Figure 67 – Multiply constant RGB

Where it can be found:

Basic Operations Module ► Arithmetic ► Multiply with value (RGB)

Description:

Multiply with value (RGB) is an arithmetic operation available in Basic Operations Module.

This operation multiplies each pixel channel of the image with the corresponding component of a color value (red, green, blue) and places the result in the corresponding element of the output.

$$\boldsymbol{Result}\,\langle r|g|b\rangle = \boldsymbol{Image}\,\langle r\,|\,g\,|\,b\rangle * \boldsymbol{value}\,\langle r|g|b\rangle$$

Parameters:

- 1.1 Image - the input image
- 1.2 Value - Red - the value used to multiply the Red channel of the input image
- 1.3 Value - Green - the value used to multiply the Green channel of the input image
- 1.4 Value - Blue - the value used to multiply the Blue channel of the input image

- 1.5 Scaling factor - scales the result by multiplying with 2value (default = 0)
- 1.6 Result of "…" - the result of the operation

Effect:

Changes the intensity of a color image.

Example:

- Input: The image below is a brightfield color image (8-bit, 3-channel) representing a small region from a colon sample.

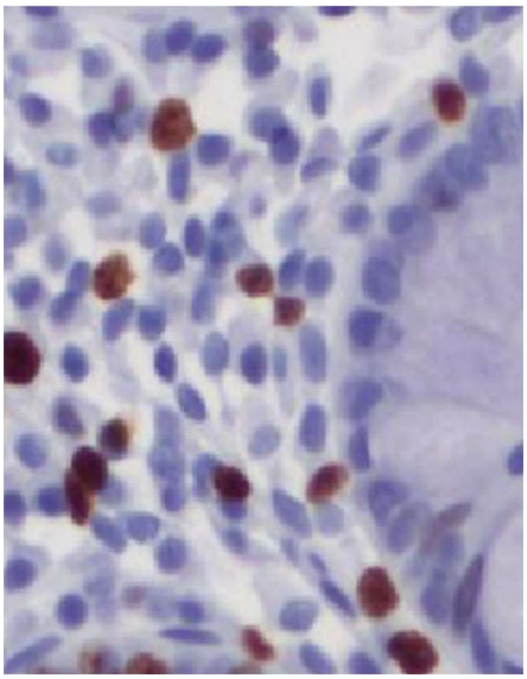


Figure 68 – Colon sample

- Engine settings:

Figure below shows the engine settings used for this example.

Different values are used to multiply the channels of the input image: red channel x 2, green channel x 1, blue channel x 2. The engine's result is a color image (8-bit, 3-channel), matching the input.

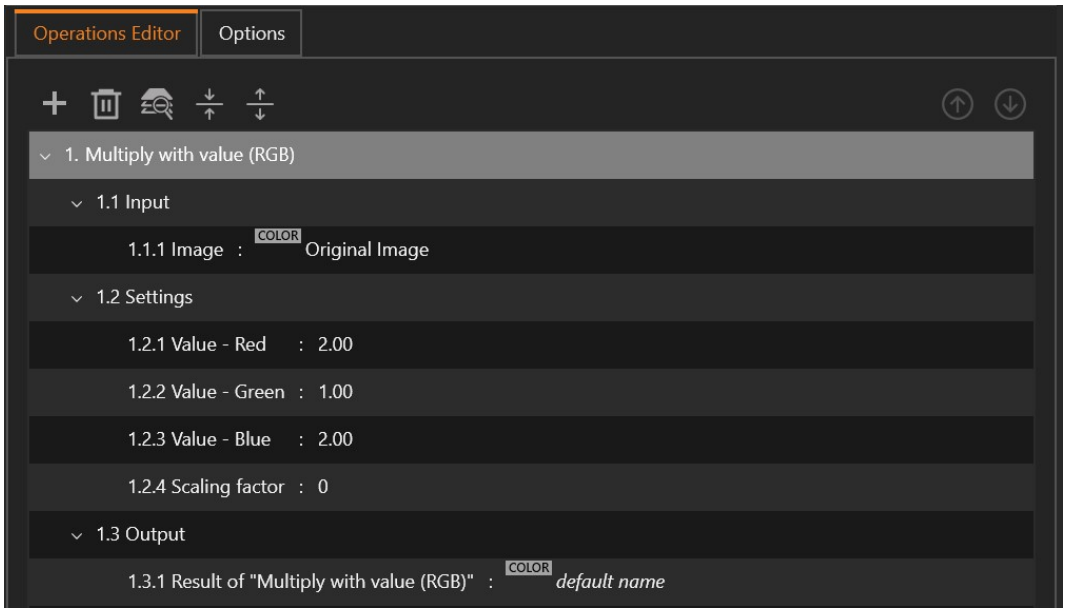


Figure 69 – Engine settings

Output:

The result has the brightness increased for red and blue channels, while the green channel remained unchanged.

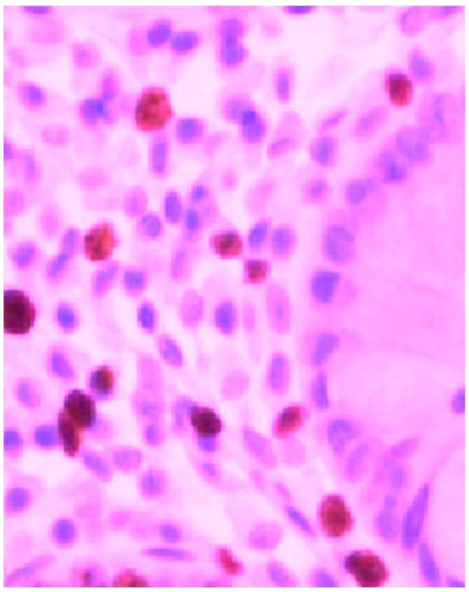


Figure 70 – Result of Multiply with value (RGB) engine

Polynomial - a + b * Image ^ c

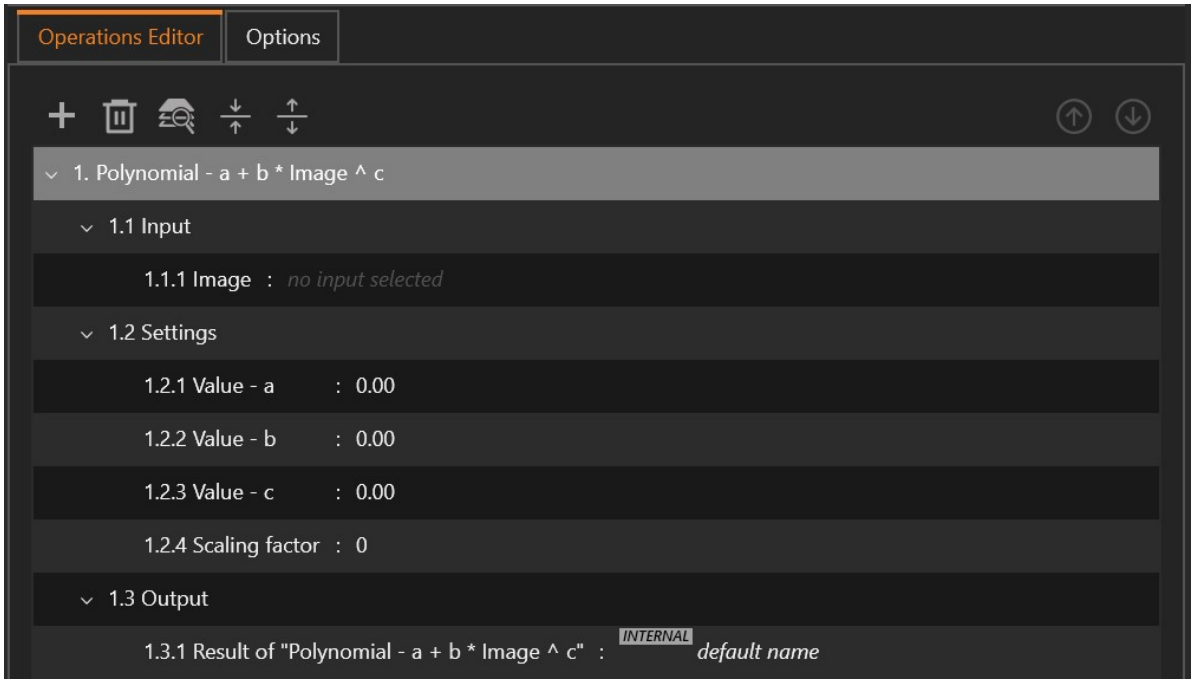


Figure 71 – Polynomial - a + b * Image ^ c

Where it can be found:

Basic Operations Module ► Arithmetic ► Polynomial -a + b * Image ^ c

Description:

Polynomial - a + b * Image ^ c is an arithmetic operation available in Basic Operations Module.

This operation applies a polynomial function to each pixel of the image and places the result in the corresponding element of the output.

$$Result = a + b * Image^{c}$$

Parameters:

- 1.1 Image - the input image
- 1.2 Value - “a” - the value of “a” in the polynomial function
- 1.3 Value - “b” - the value of “b” in the polynomial function
- 1.4 Value - “c” - the value of “c” in the polynomial function
- 1.5 Scaling factor - scales the result by multiplying with 2^{value} (default = 0)
- 1.6 Result of “…” - the result of the operation

Effect:

Changes the intensity of a grayscale image by applying a polynomial function.

Example:

- Input: The image below is a fluorescence grayscale image (8-bit, 1-channel) representing the SpGold channel of a colorectal cancer sample.

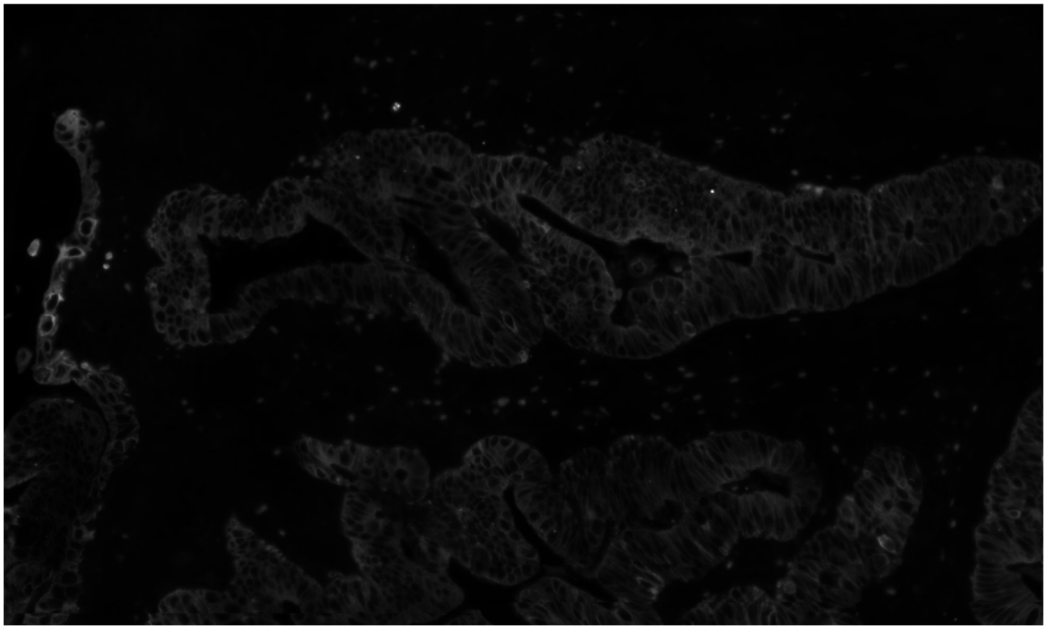


Figure 72 – Colorectal cancer sample – SpGold channel

- Engine settings:

Figure below shows the engine settings used for this example.

Input image brightness is modified using a polynomial function

$$Result = 50 + 1.5 * Image^1$$

The engine's result is a grayscale image (8-bit, 1-channel), matching the input.

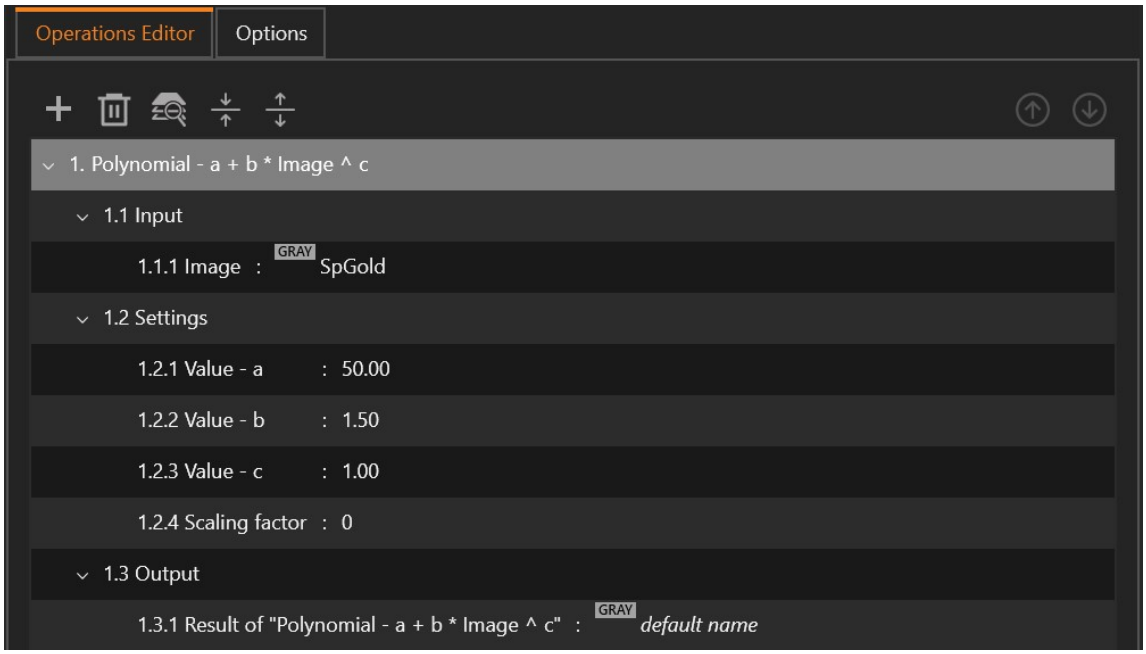


Figure 73 – Engine settings

- Output:

The result has an increased brightness compared with the input image.

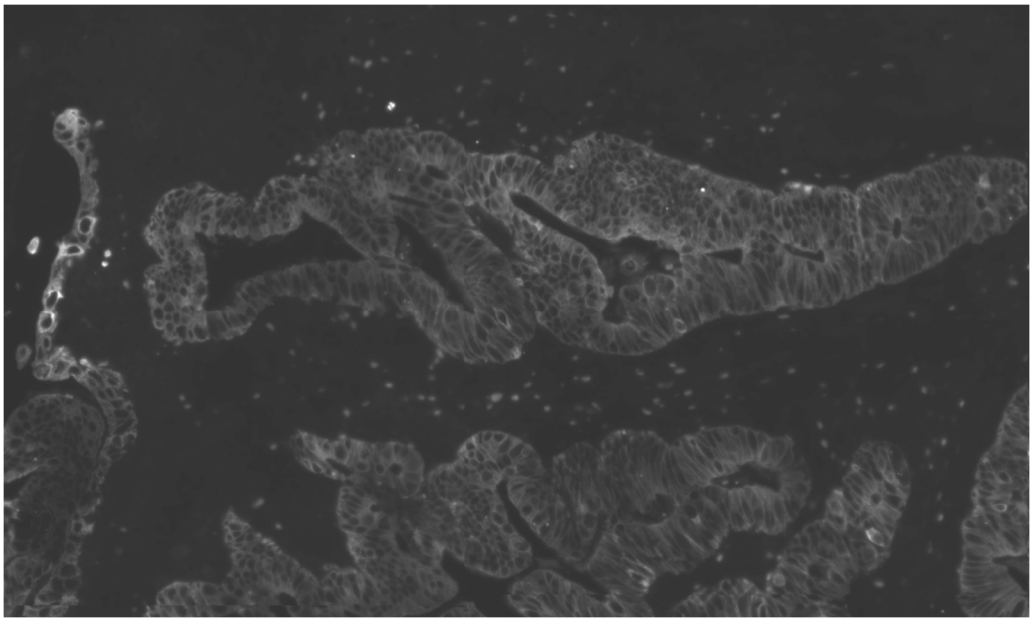


Figure 74 – Result of Polynomial – a + b * Image ^ c engine

Square (Sqr)

Figure 75 – SQR

Where it can be found:

Basic Operations Module ► Arithmetic ► Square (Sqr)

Description:

Square (Sqr) is an arithmetic operation available in Basic Operations Module.

This operation computes the square value for each pixel of the image and places the result in the corresponding element of the output.

$$Result = Image^2$$

$$\mathrm{Sqr}\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} a_{11}^{\,2} & a_{12}^{\,2} & a_{13}^{\,2} \\ a_{21}^{\,2} & a_{22}^{\,2} & a_{23}^{\,2} \\ a_{31}^{\,2} & a_{32}^{\,2} & a_{33}^{\,2} \end{pmatrix}$$

Parameters:

- 1.1 Image - the input image
- 1.2 Scaling factor - scales the result by multiplying with 2^{value} (default = 0)
- 1.3 Result of "…" - the result of the operation

Effect:

Change the intensity of a grayscale image.

Example:

- Input: The image is a fluorescence grayscale image (8-bit, 1-channel) representing the SpGold channel of a colorectal cancer sample.

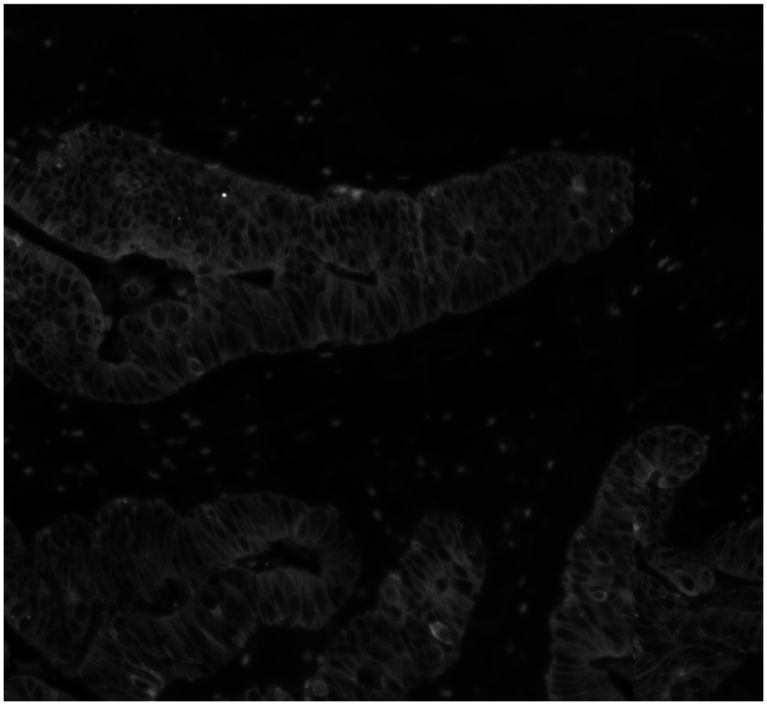


Figure 76 – SpGold channel

- Engine settings:

Figure below shows the engine settings used for this example.

The brightness of the input image is increased by squaring the intensity values. Due to the nature of the square function, the resulting intensity values can be much higher than 8-bit limit (255), thus the result is scaled down by 2^2 (divided by 4), to retain the high intensity details and discard low intensity details.

The engine's result is a grayscale image (8-bit, 1-channel), matching the input.

Note	A positive / negative value for scaling factor scales down / up the resulting value(s).

Figure 77 – Engine settings

- Output:

The result has an increased brightness compared with the input image.

Figure 78 – Result of Square (Sqr) engine

Square root (Sqrt)

Figure 79 – SQRT

Where it can be found:

Basic Operations Module ► Arithmetic ► Square root (Sqrt)

Description:

Square root (Sqrt) is an arithmetic operation available in Basic Operations Module.

This operation computes the square root value for each pixel of the image and places the result in the corresponding element of the output.

$$\boldsymbol{Result} = \sqrt{\boldsymbol{Image}}$$

$$\text{Sqrt}\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} \sqrt{a_{11}} & \sqrt{a_{12}} & \sqrt{a_{13}} \\ \sqrt{a_{21}} & \sqrt{a_{22}} & \sqrt{a_{23}} \\ \sqrt{a_{31}} & \sqrt{a_{32}} & \sqrt{a_{33}} \end{pmatrix}$$

Parameters:

- 1.1 Image - the input image
- 1.2 Scaling factor - scales the result by multiplying with 2^{value} (default = 0)
- 1.3 Result of "…" - the result of the operation

Effect:

Changes the intensity of a grayscale image.

Example:

- Input: The image below is a fluorescence grayscale image (8-bit, 1-channel) representing the SpGold channel of a colorectal cancer sample.

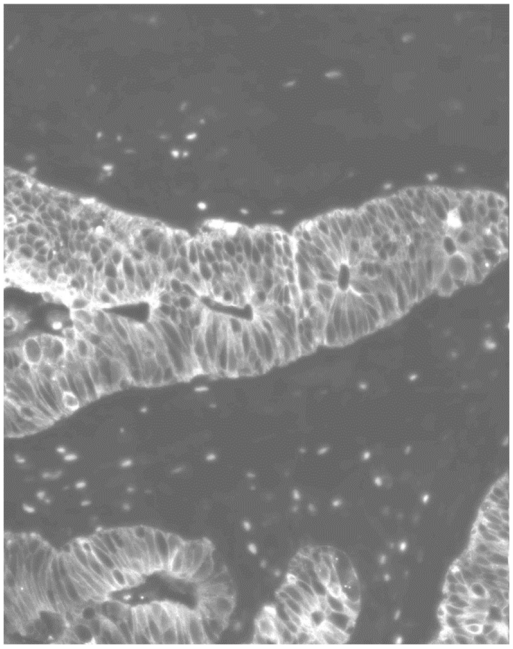


Figure 80 – Colon sample – SpGold channel

- Engine settings:

Figure below shows the engine settings used for this example.

The input image brightness is decreased by taking the square root of the intensity values. Due to the nature of the square root function, the resulting intensity values can be too small to be distinguished. Accordingly, the result is scaled up by 2^1 (multiplied by 2), to amplify the values.

The engine's result is a grayscale image (8-bit, 1-channel), matching the input.

Note	A positive / negative value for scaling factor scales down / up the resulting value(s).

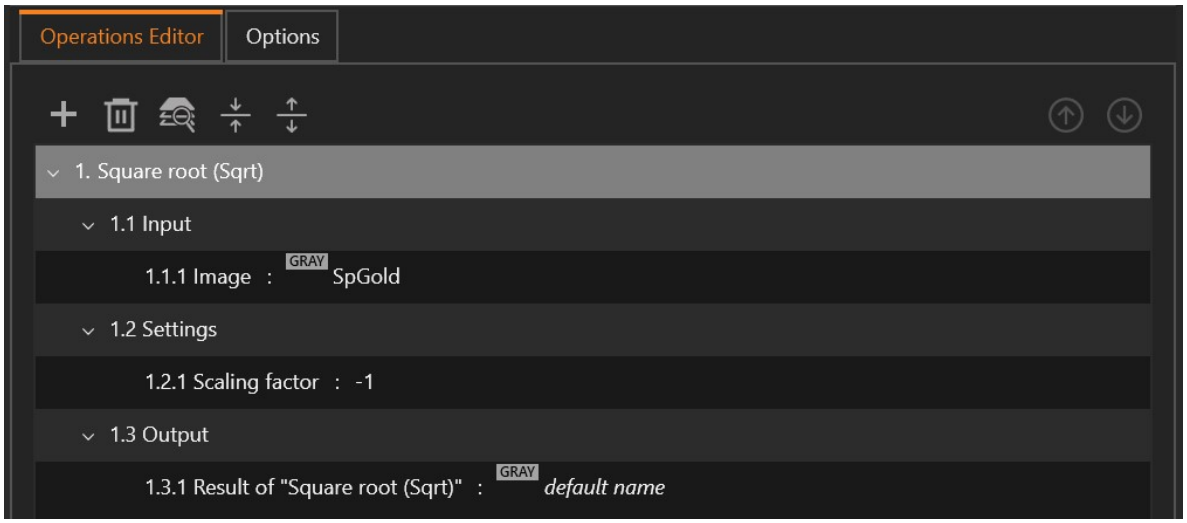


Figure 81 – Engine settings

- Output:

The result below has decreased brightness compared with the input image.

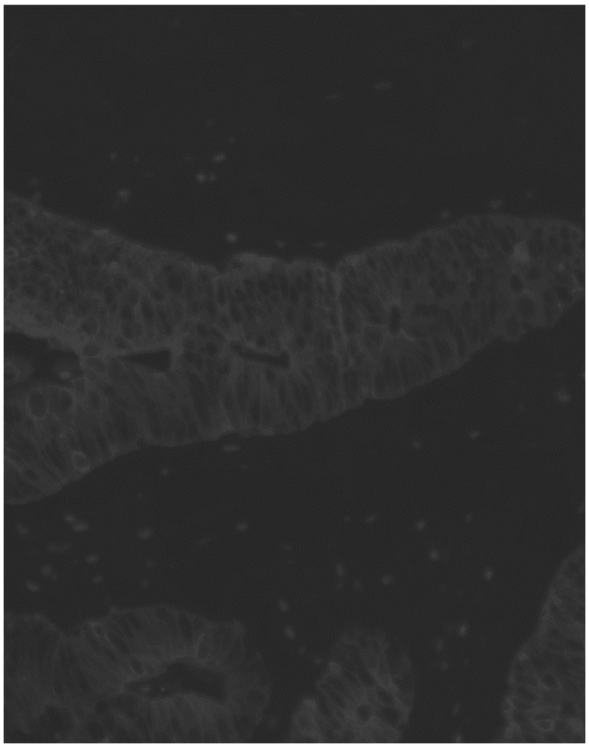


Figure 82 – Result of Square Root (Sqrt) engine

Subtract images

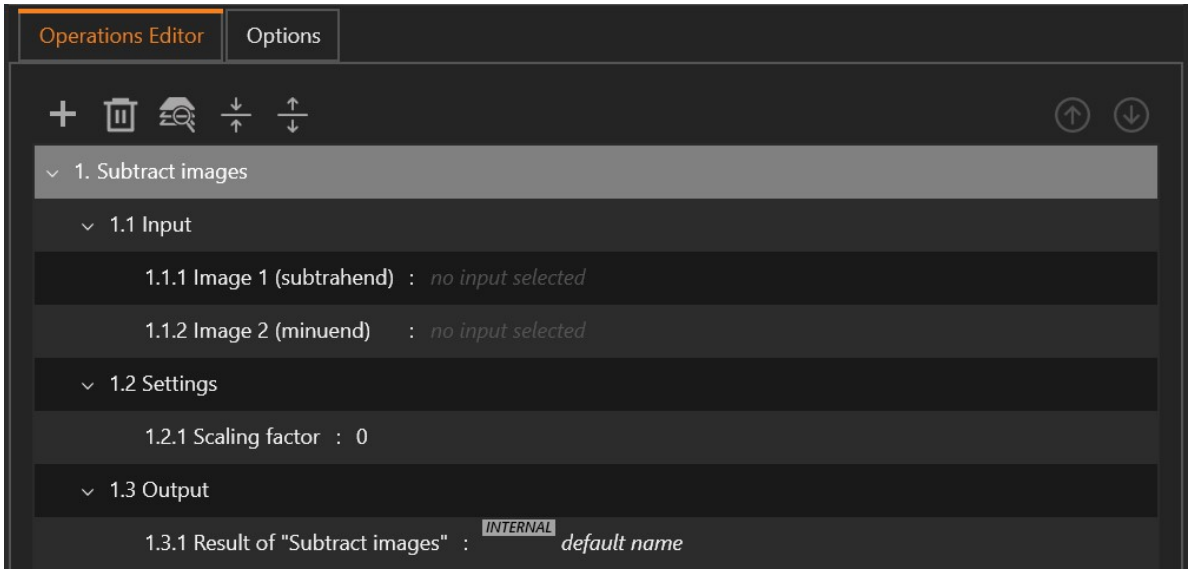


Figure 83 – Subtract images

Where it can be found:

Basic Operations Module ► Arithmetic ► Subtract images

Description:

Subtract images is an arithmetic operation available in Basic Operations Module.

This operation subtracts the pixel of Image 1 from the corresponding pixel of Image 2 and places the result in the corresponding element of the output (performs the pixel-wise subtraction of two images).

$$Result = Image_2 - Image_1$$

Parameters:

- 1.1 Image 1 (subtrahend) - the first input image representing the subtrahend
- 1.2 Image 2 (minuend) - the second input image representing the minuend. Must have the same number of channels (1, 3) and the same data type (8-bit, 16-bit) as the first input image
- 1.3 Scaling factor - scales the result by multiplying with 2^{value} (default = 0)
- 1.4 Result of "…" - the result of the operation

Effect:

Subtracts Image 1 information from Image 2.

Example:

- Input: Image below is a fluorescence grayscale image (8-bit, 1-channel) representing the DAPI channel of an UV irradiated skin sample.

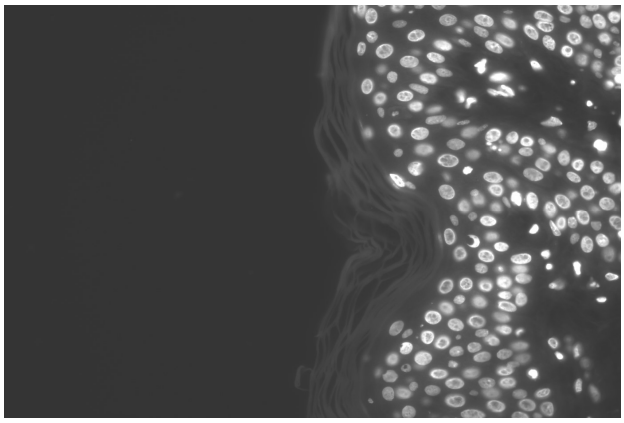


Figure 84 – UV irradiated skin sample – DAPI channel

Image below is a fluorescence grayscale image (8-bit, 1-channel) representing the Rhodamine channel of an UV irradiated skin sample.

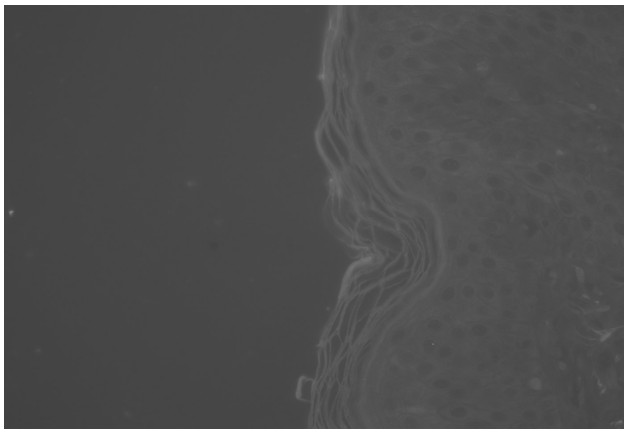


Figure 85 – UV irradiated skin sample – Rhodamine channel

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a grayscale image (8-bit, 1-channel), matching the two inputs.

Note	The two images used as input must have same bit-depth (8-bit / 16-bit) and the same number of channels (1-channel / 3-channels).

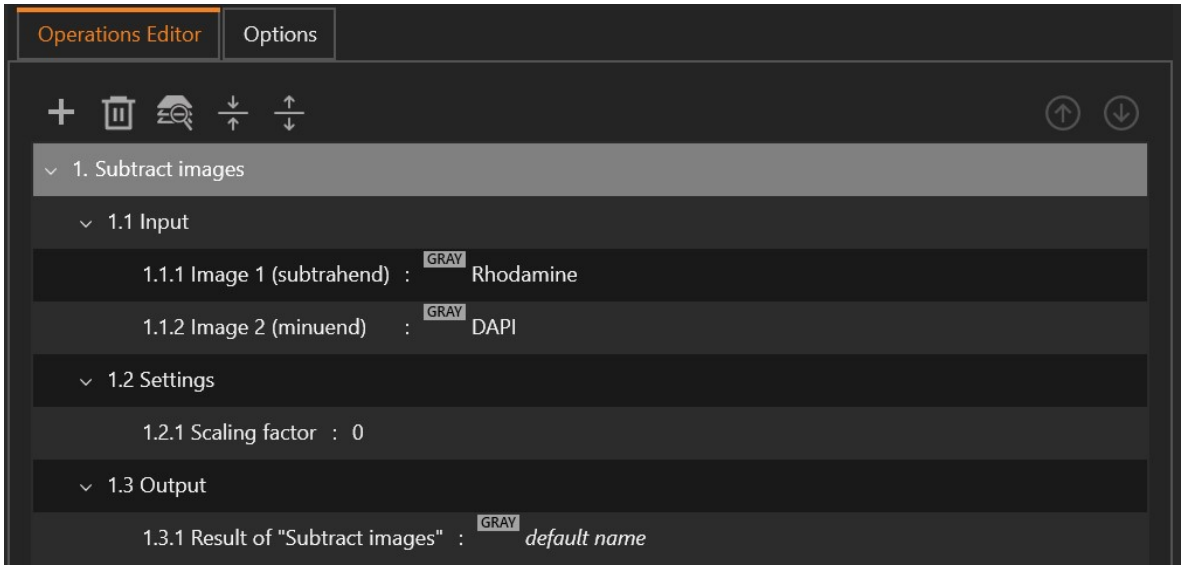


Figure 86 – Engine settings

- Output:

The result combines the information from the 2 input images, by deleting information in the Rhodamine channel from the DAPI channel.

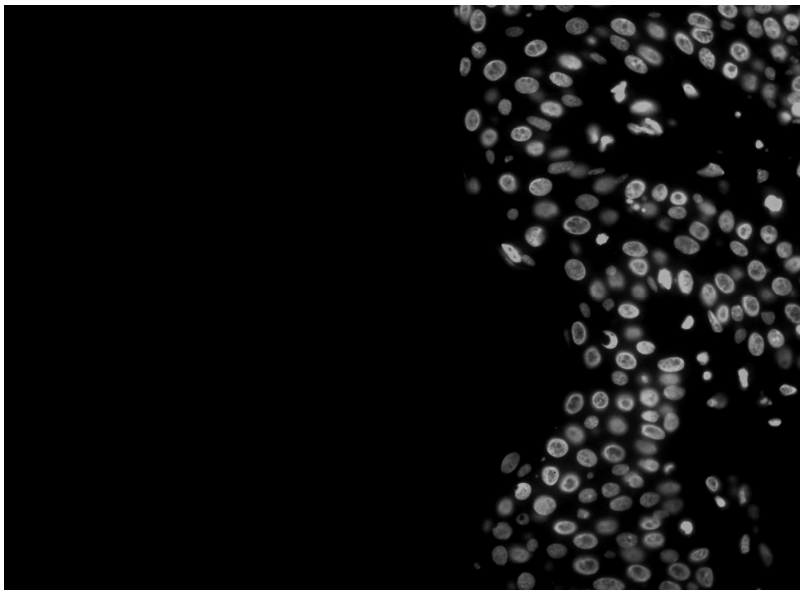


Figure 87 – Result of Subtract images engine

Subtract value

Figure 88 – Subtract value

Where it can be found:

Basic Operations Module ► Arithmetic ► Subtract value

Description:

Subtract value is an arithmetic operation available in Basic Operations Module.

This operation subtracts a numerical value from each pixel of the image and places the result in the corresponding element of the output.

$$Result = Image - value$$

Parameters:

- 1.1 Image - the input image
- 1.2. Scaling factor - scales the result by multiplying with 2^{value} (default = 0)
- 1.3 Value - the value subtracted from each pixel of the image
- 1.4 Result of "…" - the result of the operation

Effect:

Darkens (decreases the intensity of) a grayscale image.

Example:

- Input: The image below is a fluorescence grayscale image (8-bit, 1-channel) representing the SpGold channel of a colorectal cancer sample.

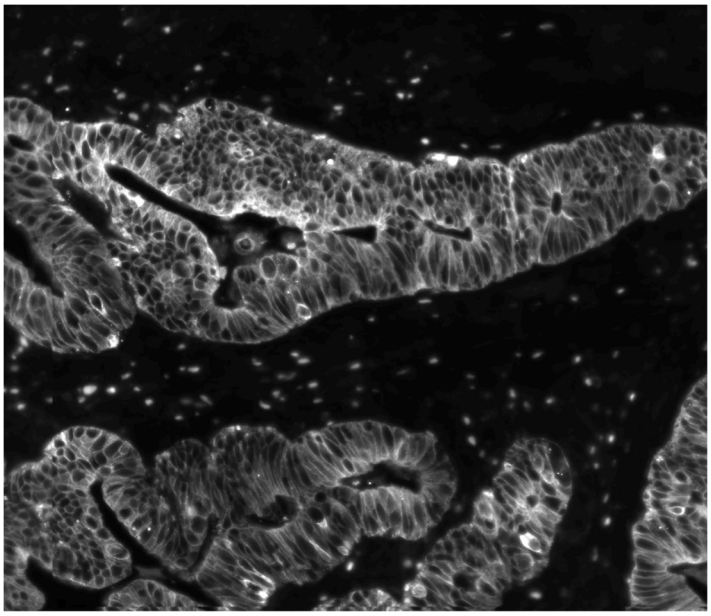


Figure 89 – Colorectal sample – SpGold channel

- Engine settings:

Figure below shows the engine settings used for this example.

A value of 50 is subtracted from the input image.

The engine's result is a grayscale image (8-bit, 1-channel), matching the input.

Figure 90 – Engine settings

- Output:

The result below has the brightness decreased by 50 compared with the input image.

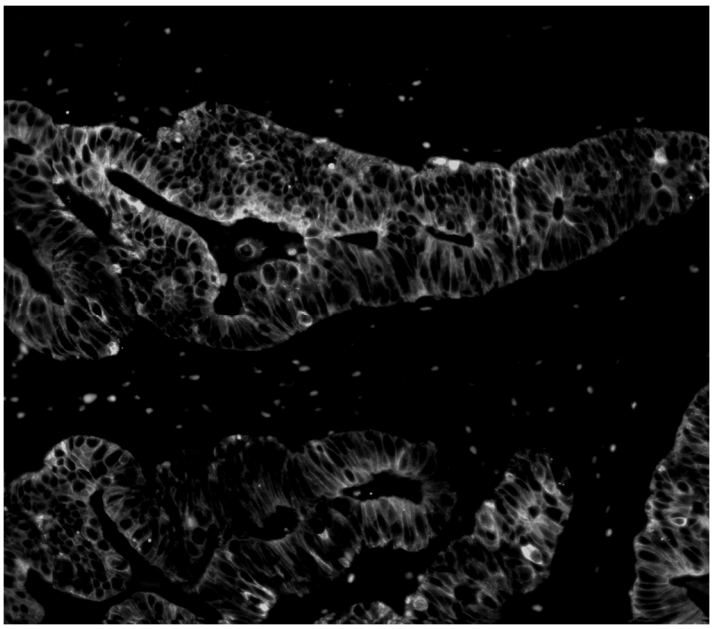


Figure 91 – Result of Subtract value engine

Subtract value (RGB)

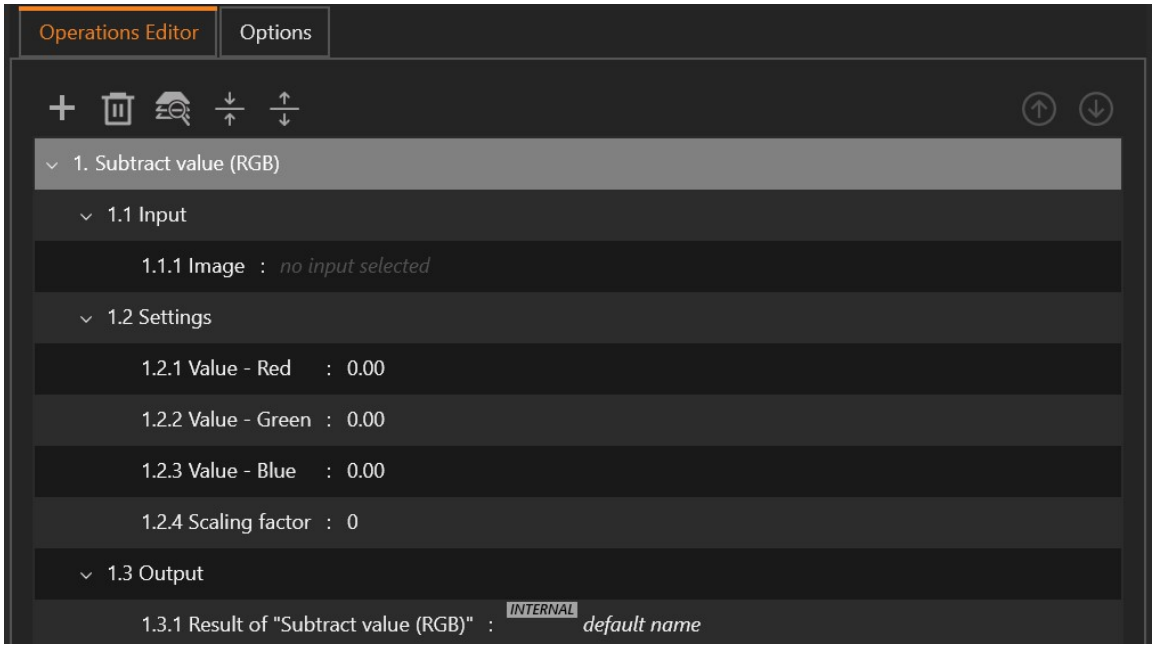


Figure 92 – Subtract value RGB

Where it can be found:

Basic Operations Module ► Arithmetic ► Subtract value (RGB)

Description:

Subtract value (RGB) is an arithmetic operation available in Basic Operations Module.

This operation subtracts the components of a color value (red, green, blue) from each pixel channel of the image and places the result in the corresponding element of the output.

$$\boldsymbol{Result}\,\langle r|g|b\rangle = \boldsymbol{Image}\,\langle r|g|b\rangle - \boldsymbol{value}\,\langle r|g|b\rangle$$

Parameters:

- 1.1 Image - the input image
- 1.2 Value - Red - the value subtracted from the Red channel of the input image
- 1.3 Value - Green - the value subtracted from the Green channel of the input image
- 1.4 Value - Blue - the value subtracted from the Blue channel of the input image
- 1.5 Scaling factor - scales the result by multiplying with 2^{value} (default = 0)
- 1.6 Result of "…" - the result of the operation

Effect:

Darkens (decreases the intensity) a color image.

Example:

- Input: The image below is a brightfield color image (8-bit, 3-channel) representing a small region from a colon sample.

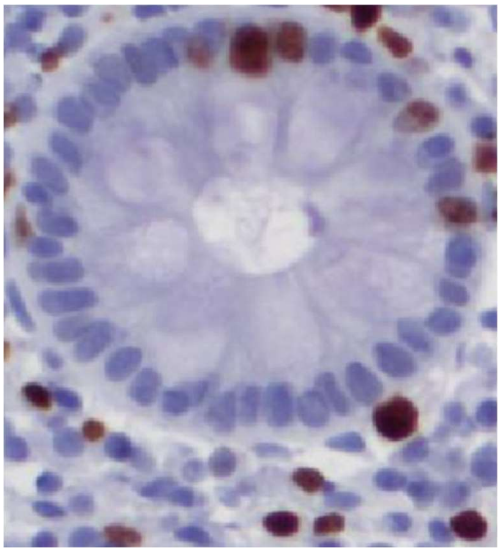


Figure 93 – Colon sample

- Engine settings:

Figure below shows the engine settings used for this example.

The same value is subtracted from each channel of the input image: red channel - 20, green channel - 20, blue channel - 20.

The engine's result is a color image (8-bit, 3-channel), matching the input.

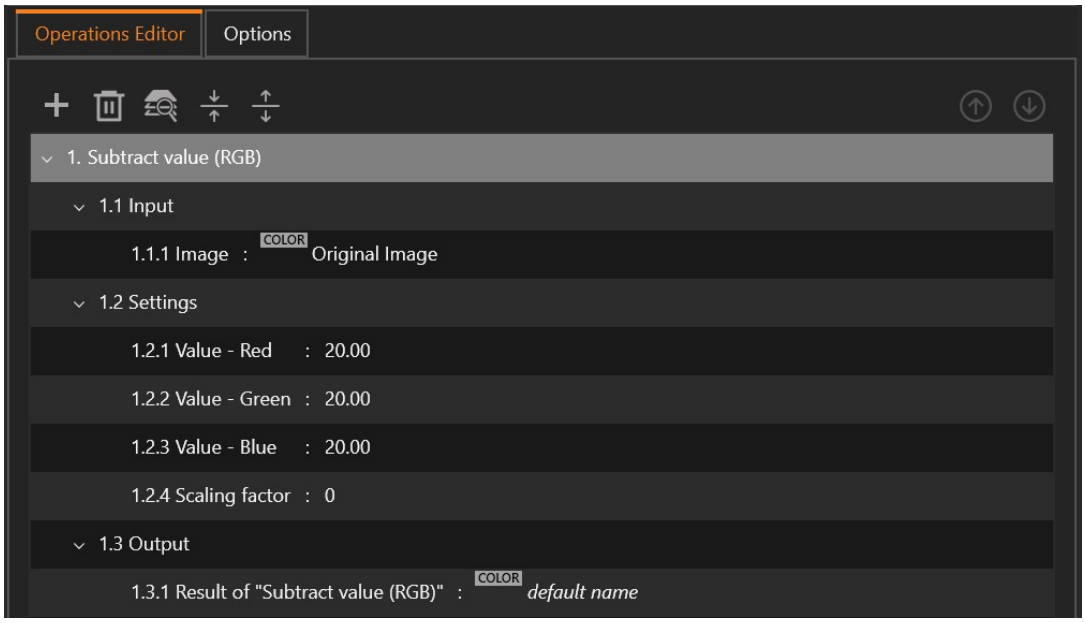


Figure 94 – Engine settings

- Output:

The result has the brightness lowered by half (identically for all channels) compared with the input image.

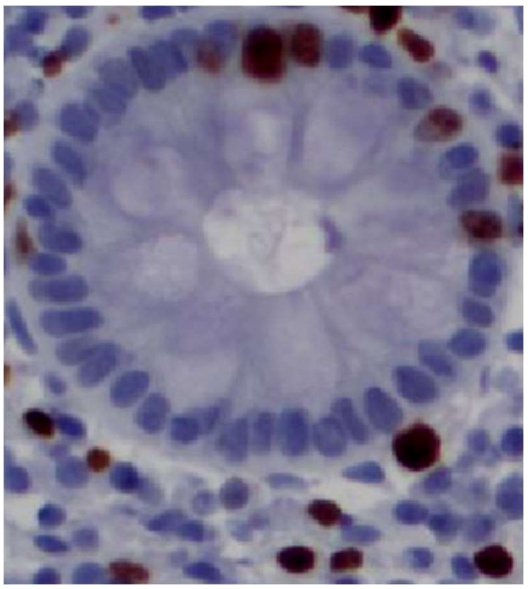


Figure 95 – Result of Subtract value (RGB) engine

5.2. Color Conversions

Convert BGR to HLS

Figure 96 – Convert BGR to HLS

Where it can be found:

Basic Operations Module ► Color Conversions ► Convert BGR to HLS

Description:

Convert BGR to HLS is a color conversion operation available in Basic Operations Module.

This operation converts an image from the RGB color-space to the HSL color-space and places the result in the output. The conversion is lossless, meaning it's a conversion without loss of information and the inverse conversion can be applied

The order of the letters in the color-space names used in the operation name is different from the color-space original names. The reason is to indicate the order of the channels:

- BGR -Blue (index = 1), Green (index = 2), Red (index = 3)
- HLS -Hue (index = 1), Luminance (index = 2), Saturation (index = 3)

Parameters:

- 1.1 Image - the input image
- 1.2 Result of "…" - the result of the operation

Effect:

Converts an image from the RGB color-space to the HSL color-space.

Example:

- Input: The image is a brightfield color image (8-bit, 3-channel) representing a small region from a colon sample.

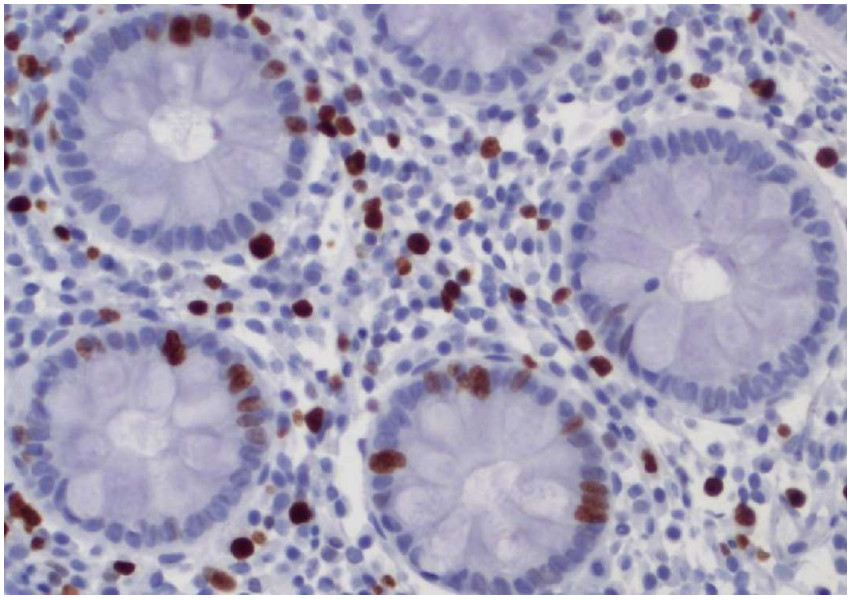


Figure 97 – Colon sample (RGB color-space)

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a color image (8-bit, 3-channel), matching the input.

Figure 98 – Engine settings

- Output:

The result represents the HSL representation of the input image.

If needed, particular channel(s) can be extracted from the HSL image.

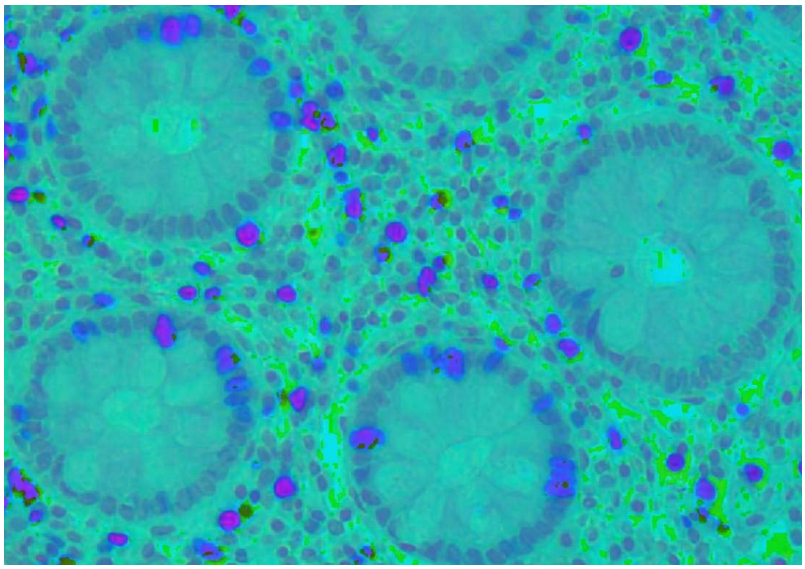


Figure 99 – Result of Convert BGR to HLS engine

Convert BGR to HSV

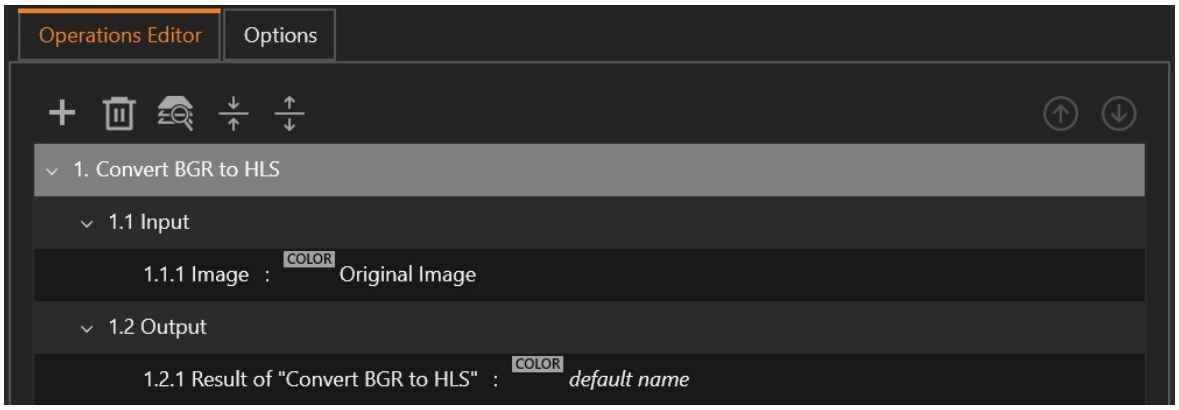


Figure 100 – Convert BGR to HSV

Where it can be found:

Basic Operations Module ► Color Conversions ► Convert BGR to HSV

Description:

Convert BGR to HSV is a color conversion operation available in Basic Operations Module.

This operation converts an image from the RGB color-space to the HSV color-space and places the result in the output. The conversion is lossless, meaning it's a conversion without loss of information and the inverse conversion can be applied

The order of the letters in the color-space names used in the operation name is different from the color-space original names. The reason is to indicate the order of the channels:

- BGR -Blue (index = 1), Green (index = 2), Red (index = 3)
- HSV -Hue (index = 1), Saturation (index = 2), Value (index = 3)

Parameters:

- 1.1 Image - the input image
- 1.2 Result of "…" - the result of the operation

Effect:

Converts an image from the RGB color-space to the HSV color-space

Example:

- Input: The image below is a brightfield color image (8-bit, 3-channel) representing a small region from a colon sample.

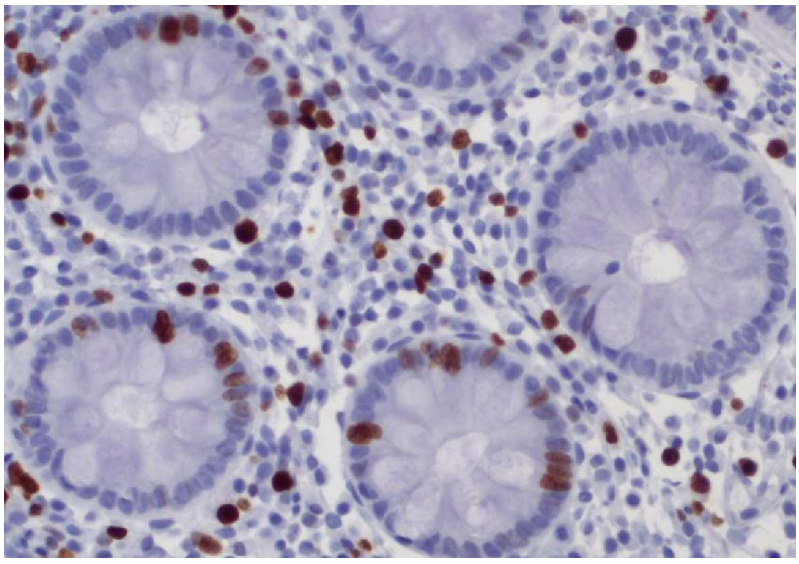


Figure 101 – Colon sample (RGB color-space)

- Engine settings:

Figure 3-92 shows the engine settings used for this example.

The engine's result is a color image (8-bit, 3-channel), matching the input.

Figure 102 – Engine settings

- Output:

The result represents the HSV representation of the input image.

If needed, particular channel(s) can be extracted from the HSV image.

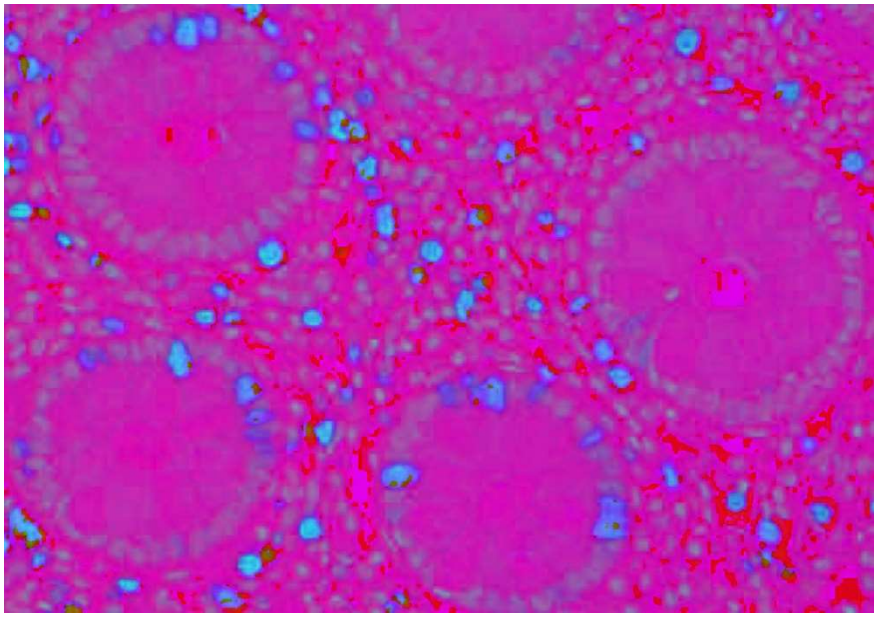


Figure 103 – Result of Convert BGR to HSV engine

Convert BGR to Lab

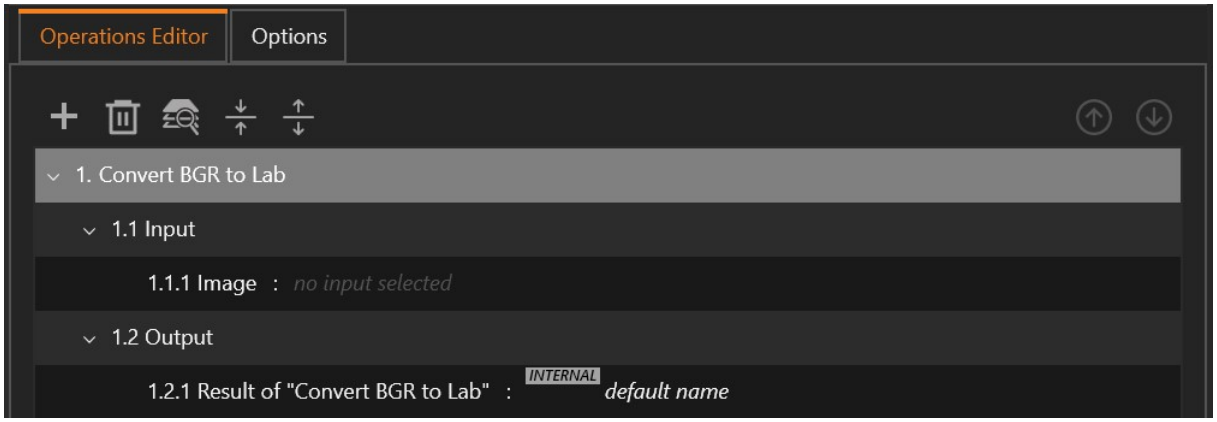


Figure 104 – Convert BGR to Lab

Where it can be found:

Basic Operations Module ► Color Conversions ► Convert BGR to Lab

Description:

Convert BGR to Lab is a color conversion operation available in Basic Operations Module.

This operation converts an image from the RGB color-space to the Lab color-space and places the result in the output. The conversion is lossless, meaning it's a conversion without loss of information and the inverse conversion can be applied

Parameters:

- 1.1 Image - the input image
- 1.2 Result of "…" - the result of the operation

Effect:

Converts an image from the RGB color-space to the Lab color-space

Example:

- Input: The image below is a brightfield color image (8-bit, 3-channel) representing a small region from a colon sample.

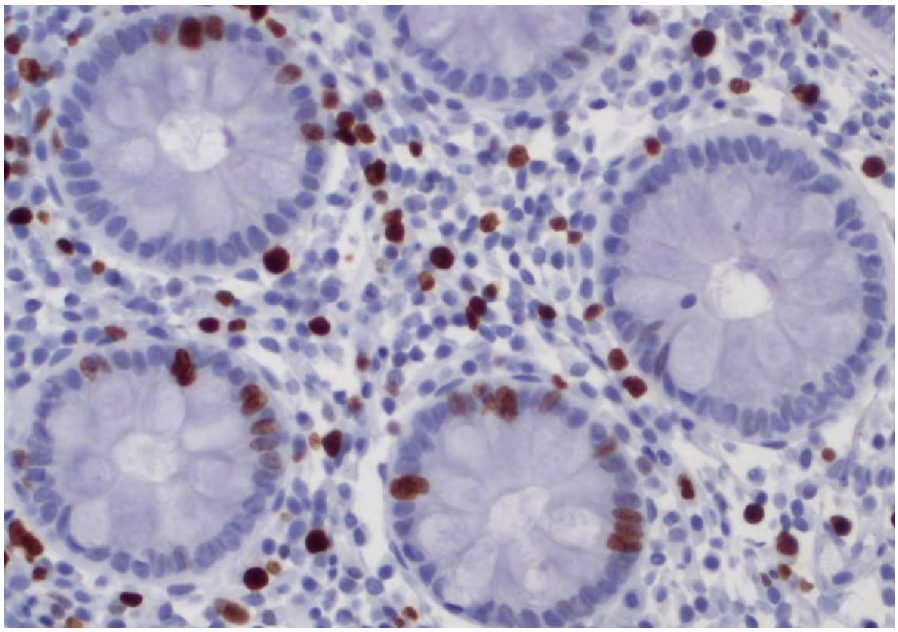


Figure 105 – Colon sample (RGB color-space)

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a color image (8-bit, 3-channel), matching the input.

Figure 106 – Engine settings

- Output:

The result represents the converted input into Lab color-space.

If needed, particular channel(s) can be extracted from the Lab image.

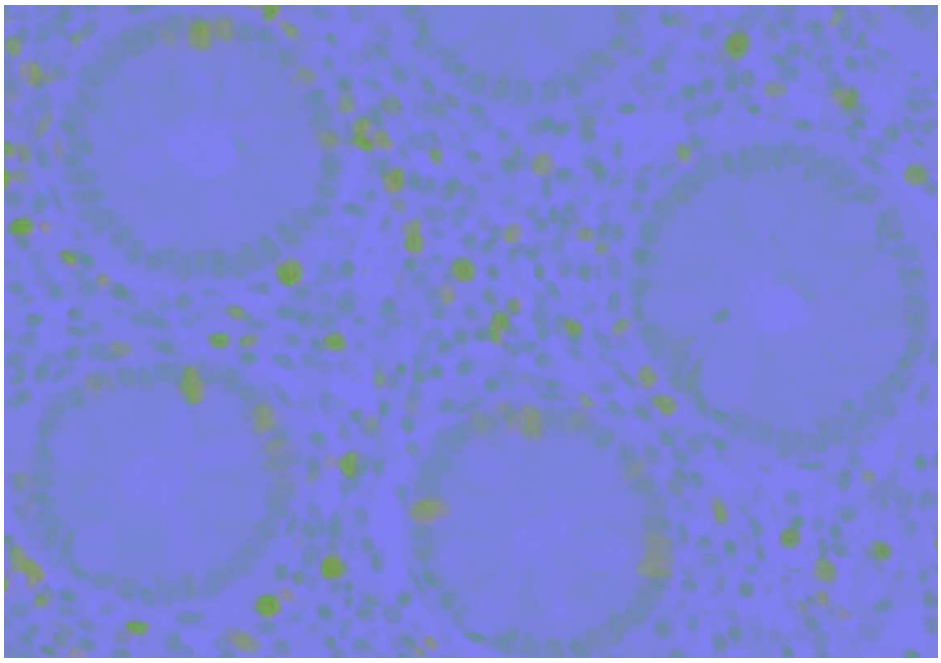


Figure 107 – Result of Convert BGR to Lab engine

Convert HLS to BGR

Figure 108 – Convert HLS to BGR

Where it can be found:

Basic Operations Module ► Color Conversions ► Convert HLS to BGR

Description:

Convert HLS to BGR is a color conversion operation available in Basic Operations Module.

This operation converts an image from the HSL color-space to the RGB color-space and places the result in the output. The conversion is lossless, meaning it's a conversion without loss of information and the inverse conversion can be applied

The order of the letters in the color-space names used in the operation name is different from the color-space original names. The reason is to indicate the order of the channels:

- BGR -Blue (index = 1), Green (index = 2), Red (index = 3)
- HLS -Hue (index = 1), Luminance (index = 2), Saturation (index = 3)

Parameters:

- 1.1 Image - the input image
- 1.2 Result of "…" - the result of the operation

Effect:

Converts an image from the RGB color-space to the HSL color-space.

Example:

- Input:

The image is a brightfield color image (8-bit, 3-channel) representing a small region from a colon sample.

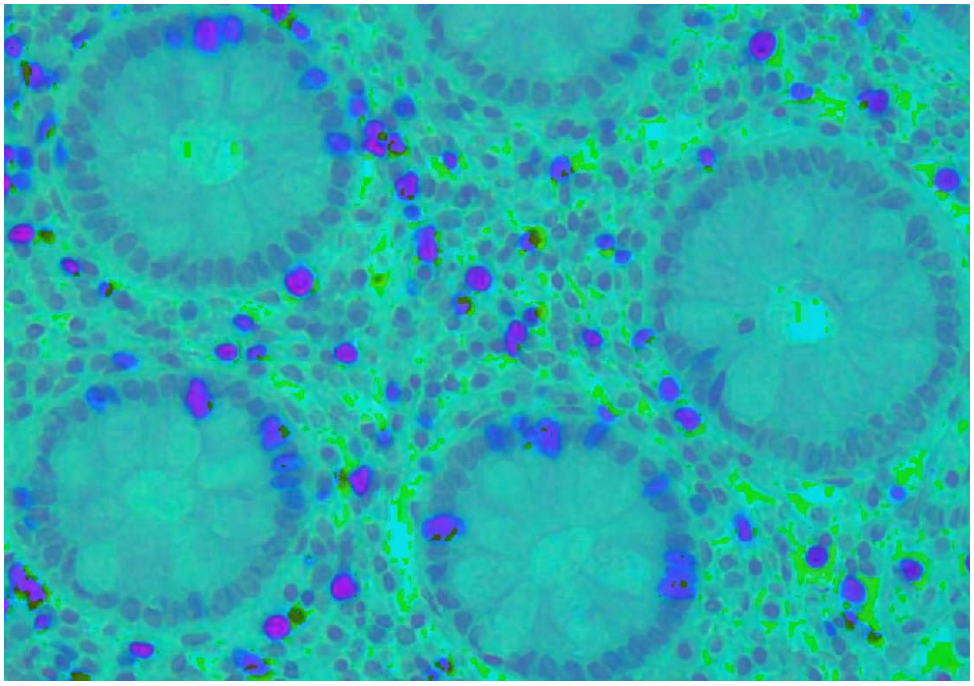


Figure 109 – Colon sample (HLS color-space)

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a color image (8-bit, 3-channel), matching the input.

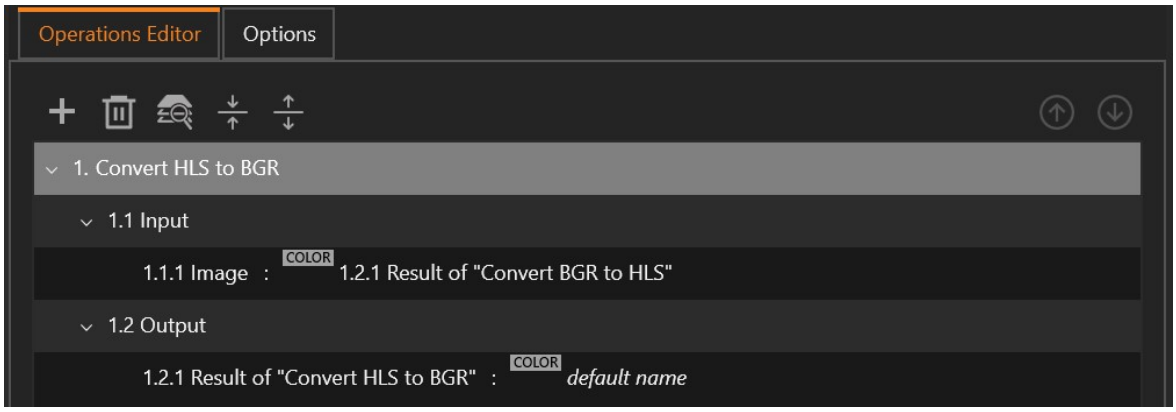


Figure 110 – Engine settings

- Output:

The result represents the converted input into HSL color-space.

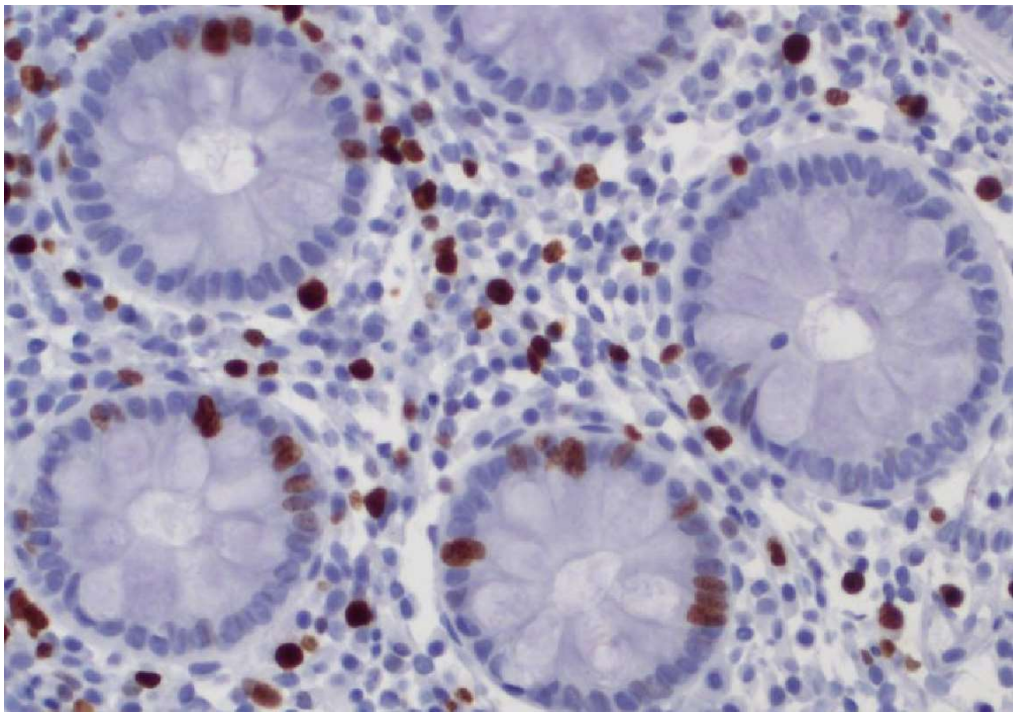


Figure 111 – Result of Convert HLS to BGR engine

Convert HSV to BGR

Figure 112 – Convert HSV to BGR

Where it can be found:

Basic Operations Module ► Color Conversions ► Convert HSV to BGR

Description:

Convert HSV to BGR is a color conversion operation available in Basic Operations Module.

This operation converts an image from the HSV color-space to the RGB color-space and places the result in the output. The conversion is lossless, meaning it's a conversion without loss of information and the inverse conversion can be applied

The order of the letters in the color-space names used in the operation name is different from the color-space original names. The reason is to indicate the order of the channels:

- BGR -Blue (index = 1), Green (index = 2), Red (index = 3)
- HSV -Hue (index = 1), Saturation (index = 2), Value (index = 3)

Parameters:

- 1.1 Image - the input image
- 1.2 Result of "…" - the result of the operation

Effect:

Converts an image from the HSV color-space to the RGB color-space.

Example:

- Input:

Image below is a brightfield color image (8-bit, 3-channel) representing a small region from a colon sample.

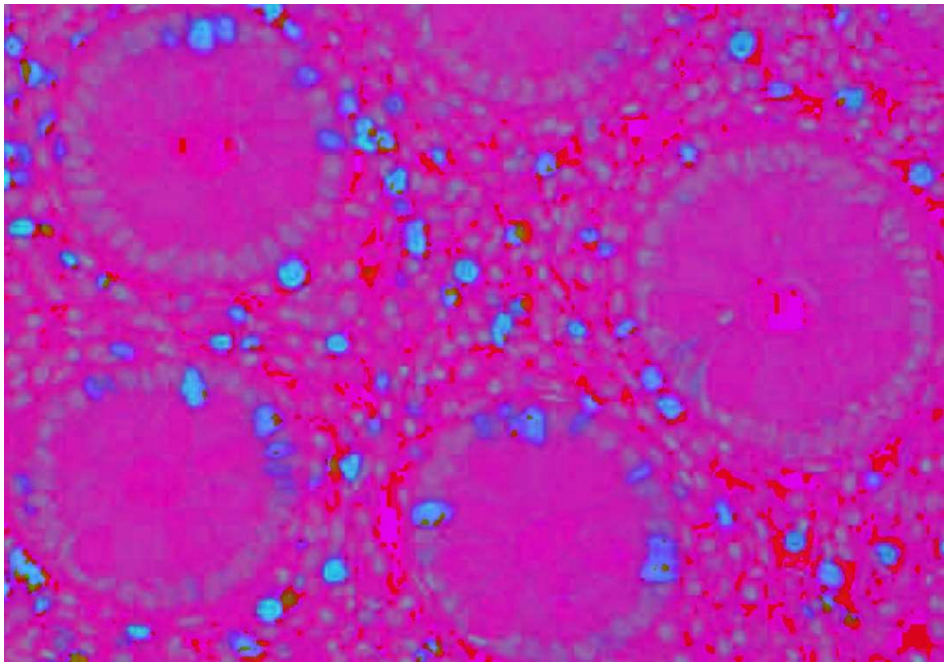


Figure 113 – Colon sample (HSV color-space)

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a color image (8-bit, 3-channel), matching the input.

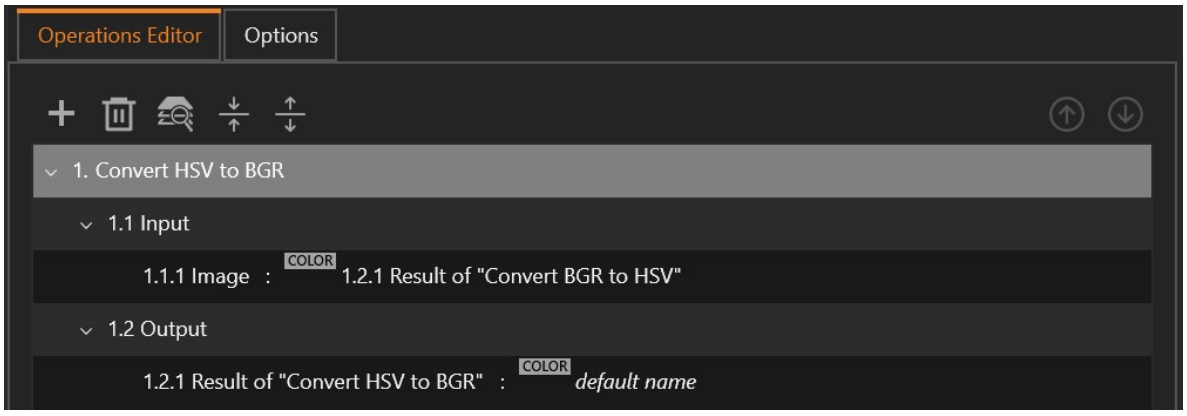


Figure 114 – Engine settings

- Output:

The result represents the converted input into HSL color-space.

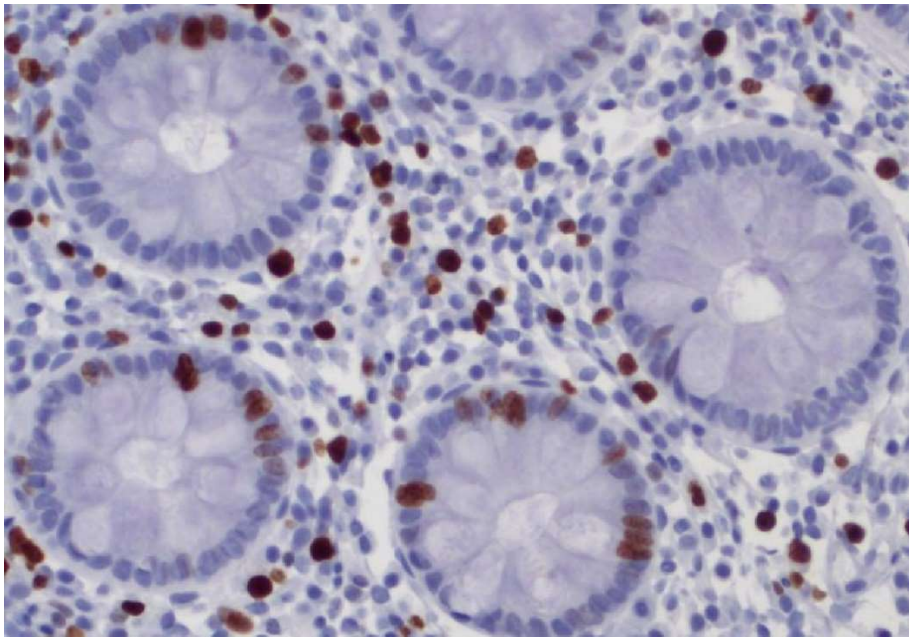


Figure 115 – Result of Convert HSV to BGR engine

Convert Lab to BGR

Figure 116 – Convert Lab To BGR

Where it can be found:

Basic Operations Module ► Color Conversions ► Convert Lab to BGR

Description:

Convert Lab to BGR is a color conversion operation available in Basic Operations Module.

This operation converts an image from the Lab color-space to the RGB color-space and places the result in the output. The conversion is lossless, meaning it's a conversion without loss of information and the inverse conversion can be applied.

Parameters:

- 1.1 Image - the input image
- 1.2 Result of "…" - the result of the operation

Effect:

Converts an image from the Lab color-space to the RGB color-space.

Example:

- Input:

The image is a brightfield color image (8-bit, 3-channel) representing a small region from a colon sample.

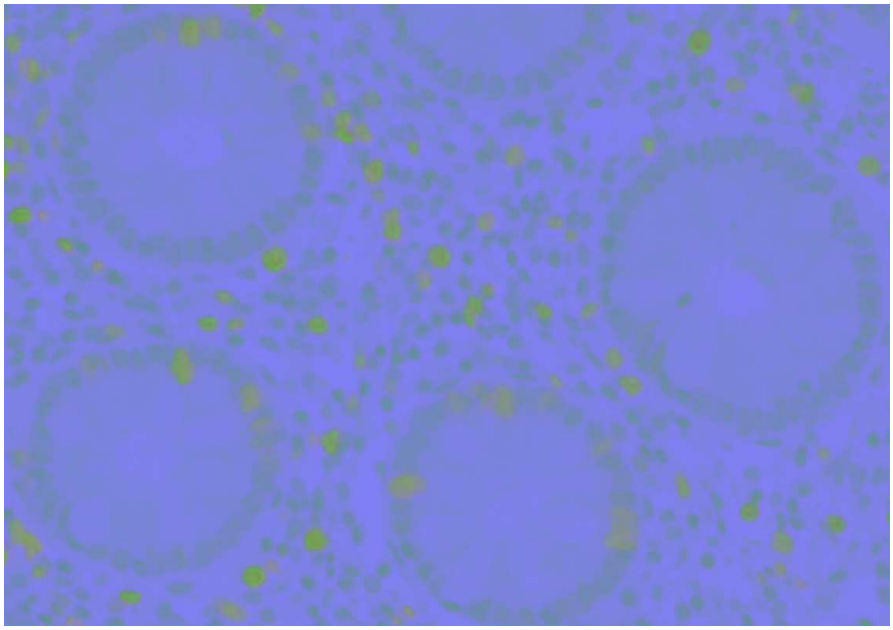


Figure 117 – Colon sample (Lab color-space)

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a color image (8-bit, 3-channel), matching the input.

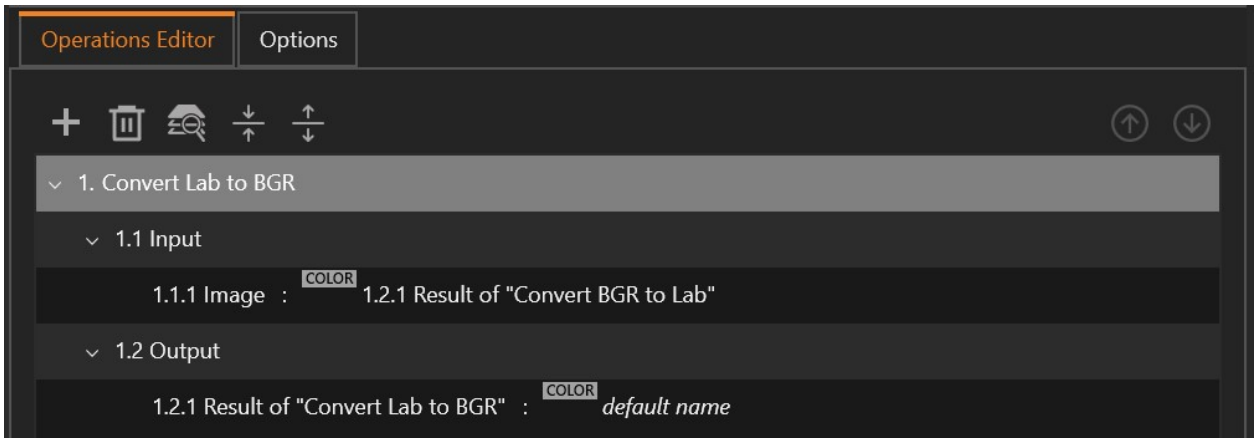


Figure 118 – Engine settings

- Output:

The result represents the converted input into BGR color-space.

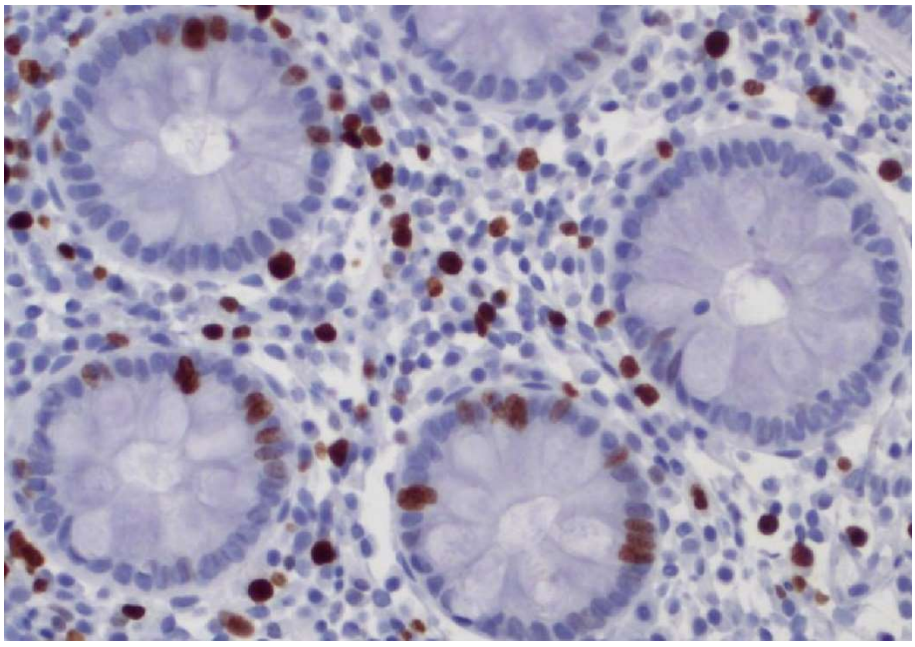


Figure 119 – Result of Convert Lab to BGR engine

Convert Optical Density (OD) to RGB

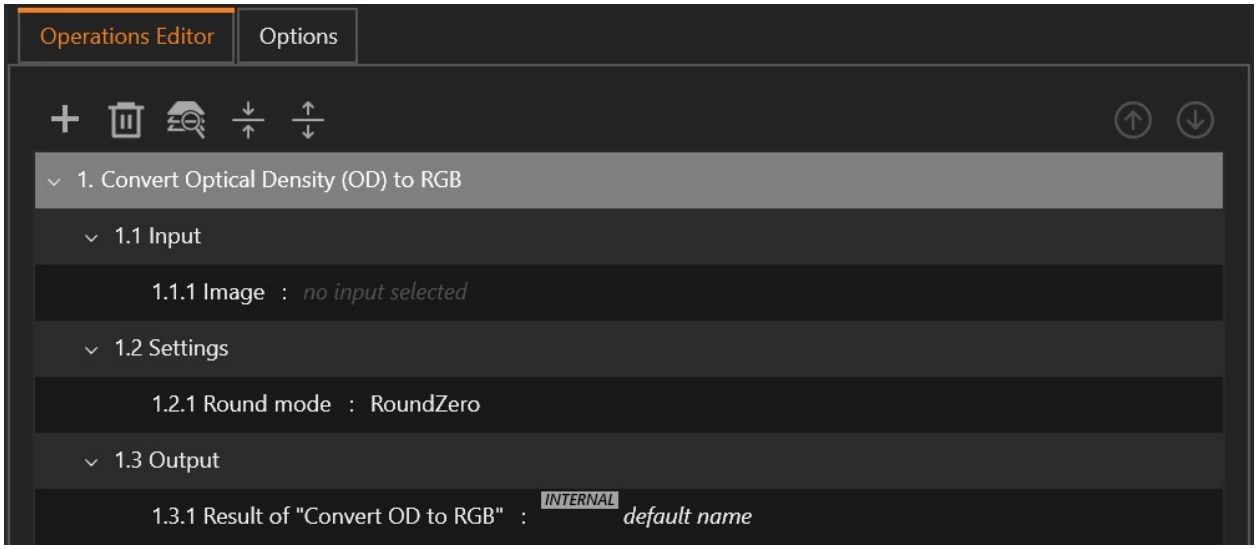


Figure 120 – Convert OD to RGB

Where it can be found:

Basic Operations Module ► Color Conversions ► Convert Optical Density (OD) to RGB

Description:

Convert Optical Density (OD) to RGB is a color conversion operation available in Basic Operations Module.

This operation converts an image from the Optical Density representation to the RGB color-space and places the result in the output. The conversion is lossless, meaning it's a conversion without loss of information and the inverse conversion can be applied.

The conversion is based on the inverse Beer-Lambert law. The Beer-Lambert law relates the attenuation of light (light source - lamp) to the properties of the material through which the light is traveling (tissue sample). In our case it is a representation of the tissue thickness based on the observed light (the acquired image).

Parameters:

1.1 Image - input image

1.2 Round mode - specifies the rounding method to be used in the conversion process (default = RoundZero)

1.3 Result of "…" - the result of the operation

Effect:

- Converts an image from the Optical Density representation to the RGB color-space.

Example:

- Input:

The image below is a brightfield color image (8-bit, 3-channel) representing a small region from a colon sample.

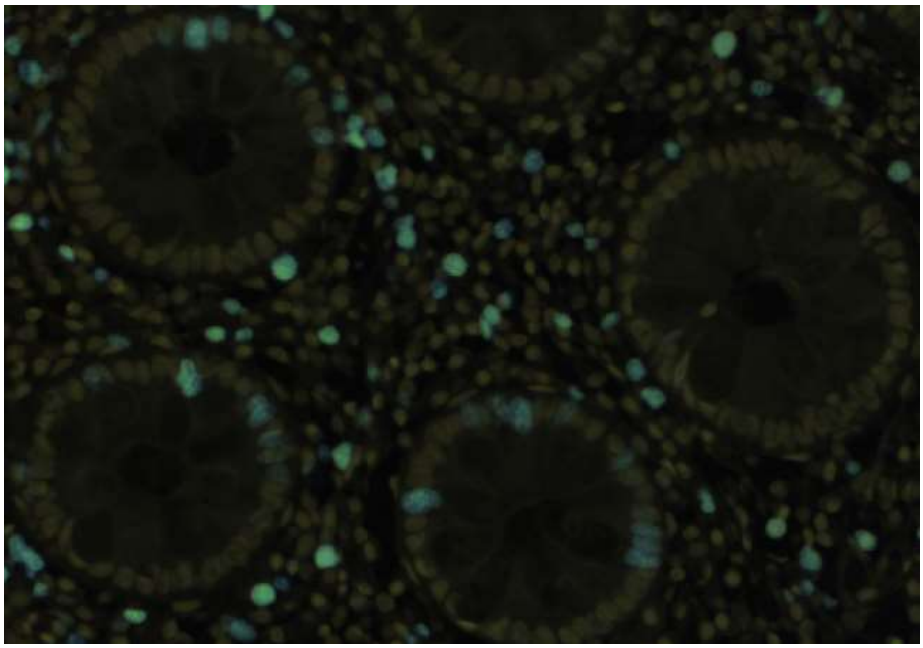


Figure 121 – Colon sample (OD representation)

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a color image (8-bit, 3-channel), matching the input.

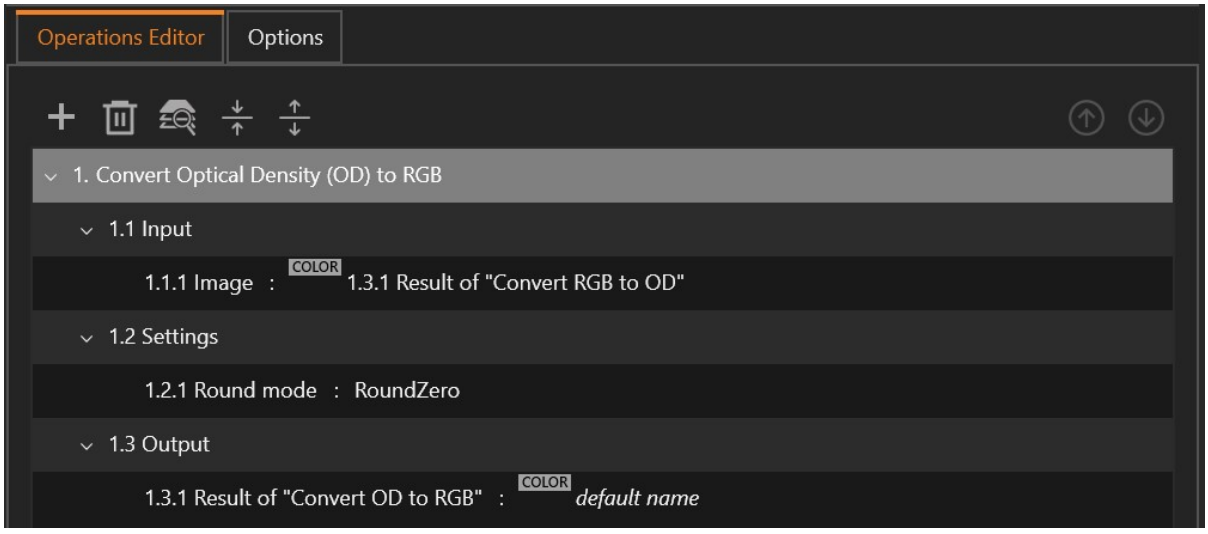


Figure 122 – Engine settings

- Output:

The result represents the converted input into RGB color-space.

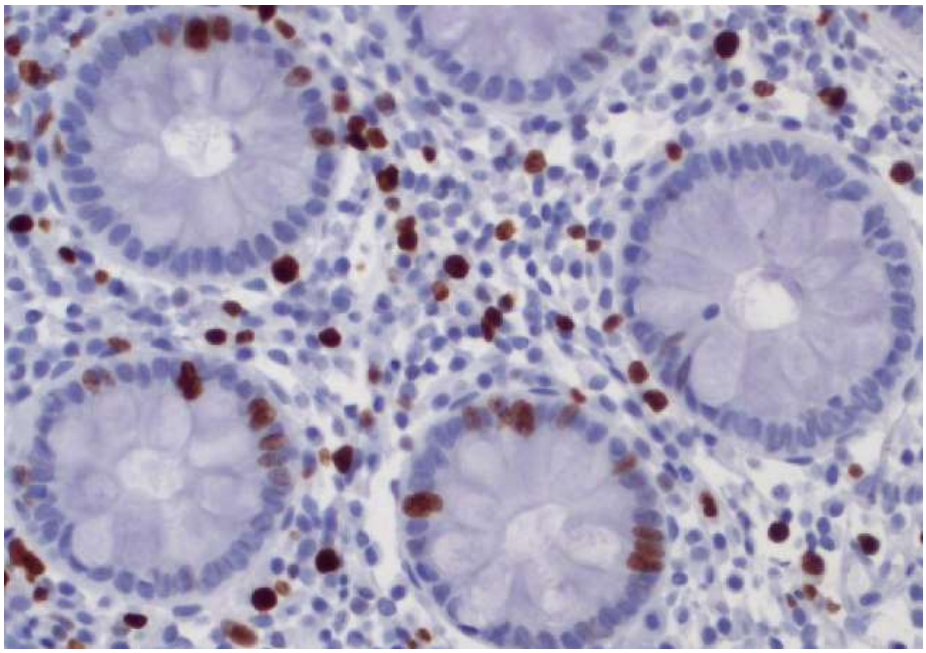


Figure 123 – Result of Convert Optical Density (OD) to RGB

Convert RGB to Grayscale

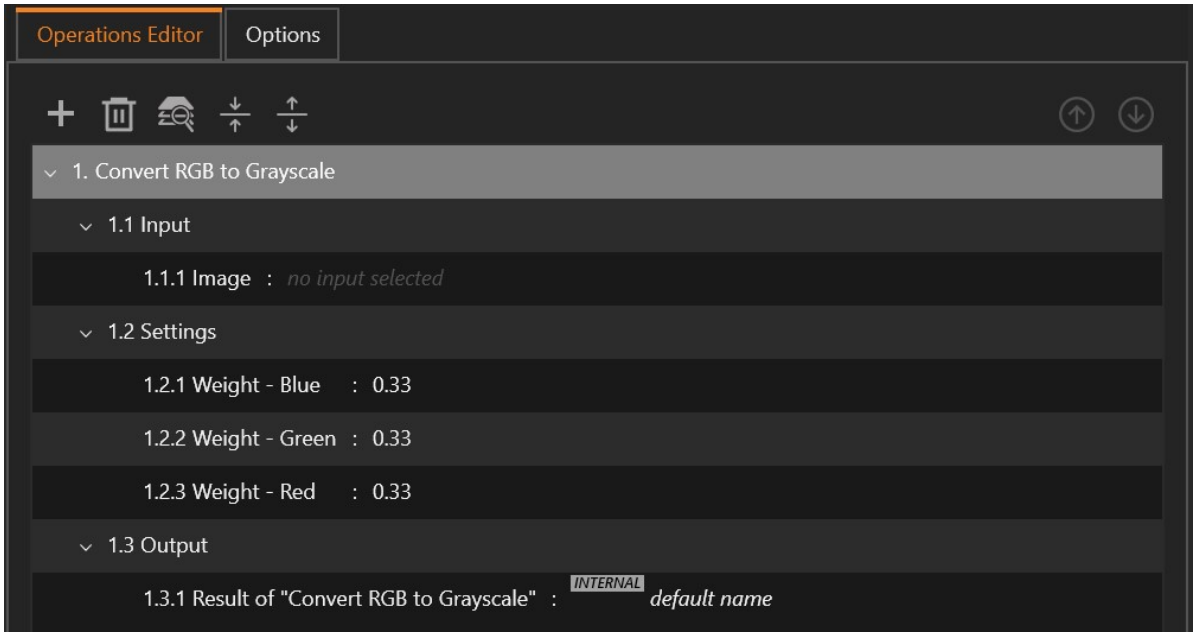


Figure 124 – Convert to gray

Where it can be found:

Basic Operations Module ► Color Conversions ► Convert RGB to Grayscale

Description:

Convert RGB to Grayscale is a color conversion operation available in Basic Operations Module.

This operation converts an image from the RGB color-space to Grayscale and places the result in the output. It is a conversion with loss of information and the inverse conversion cannot be applied.

The following equation is used to perform the conversion:

$$Result = weight_{Red} * Channel_{Red} + weight_{Green} * Channel_{Green} + weight_{Blue} * Channel_{Blue}$$

Where the weights must satisfy the condition:

$$weight_{Red} + weight_{Green} + weight_{Blue} \leq 1$$

Parameters:

- 1.1 Image - the input image

- 1.2 Weight -Blue - represents the coefficient for the Blue channel (default = 0.33)
- 1.3 Weight -Green - represents the coefficient for the Green channel (default = 0.33)
- 1.4 Weight -Red - represents the coefficient for the Red channel (default = 0.33)
- 1.5 Result of "…" - the result of the operation

Effect:

Converts an image from the RGB color-space to Grayscale.

Example:

- Input:

Image is a brightfield color image (8-bit, 3-channel) representing a small region from a colon sample.

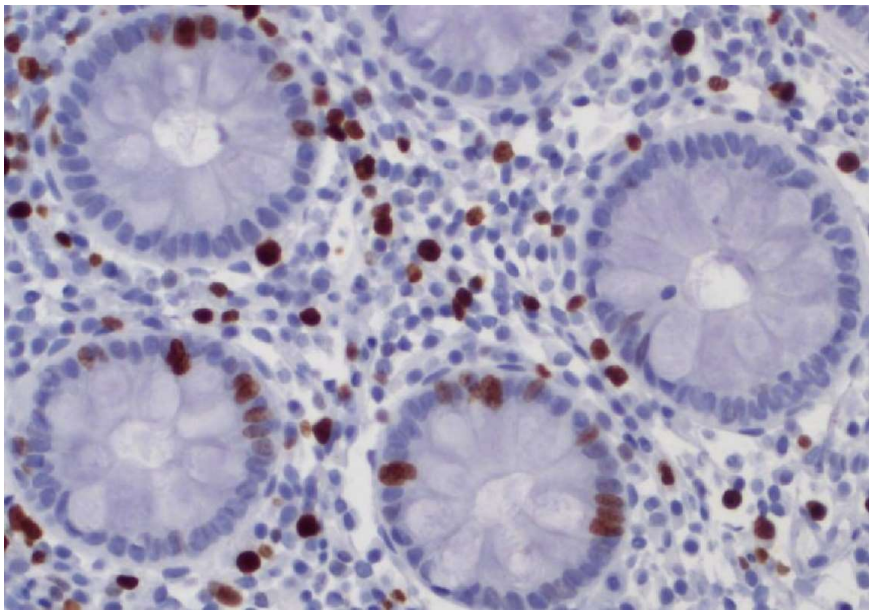


Figure 125 – Colon sample (RGB color-space)

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a grayscale image (8-bit, 3-channel), matching the input.

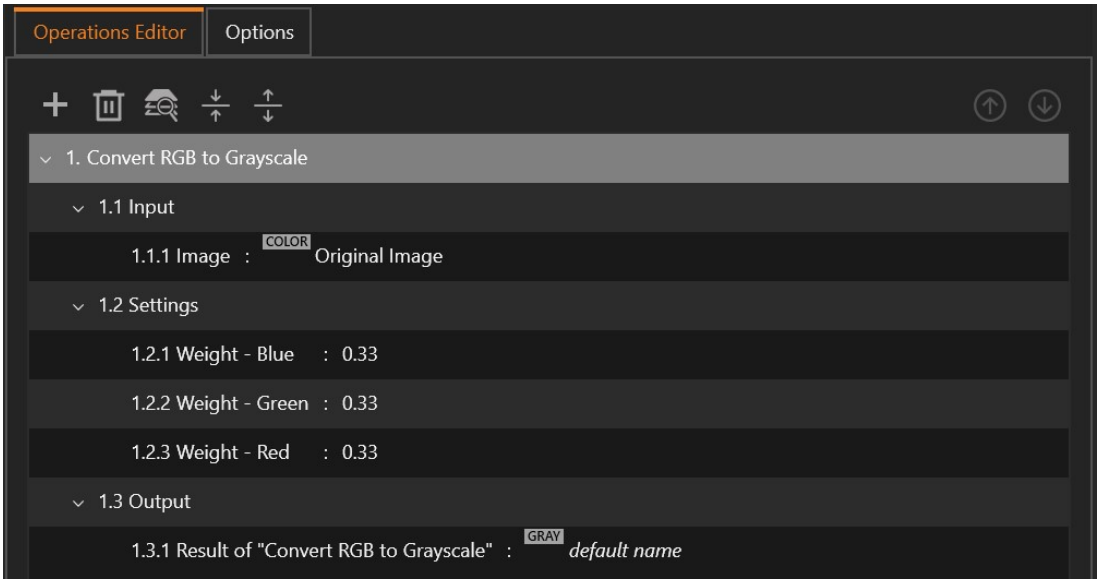


Figure 126 – Engine settings

- Output:

The result represents the converted input into grayscale color-space.

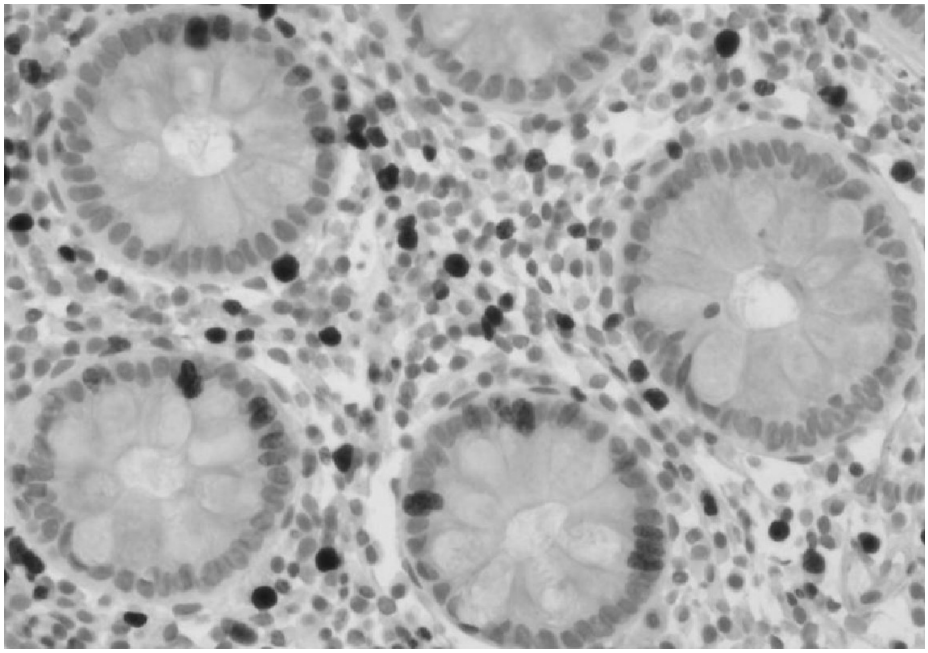


Figure 127 – Result of Convert RGB to Grayscale engine

Convert RGB to Optical Density (OD)

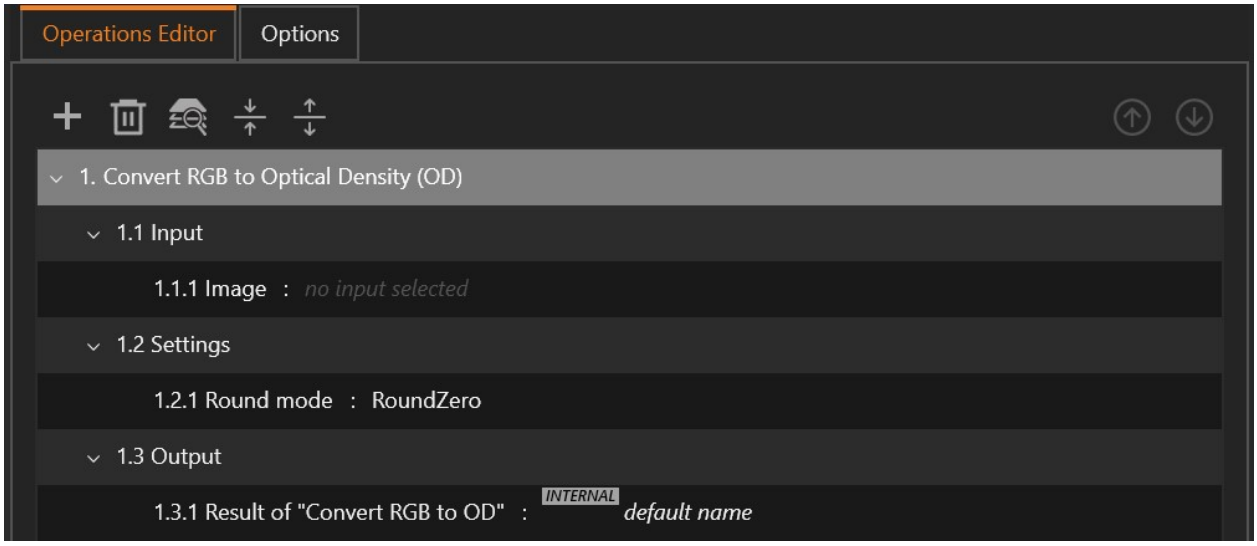


Figure 128 – Convert RGB to OD

Where it can be found:

Basic Operations Module ► Color Conversions ► Convert RGB to Optical Density (OD)

Description:

Convert RGB to Optical Density (OD) is a color conversion operation available in Basic Operations Module.

This operation converts an image from the RGB color-space to Optical Density representation and places the result in the output. The conversion is lossless, meaning it's a conversion without loss of information and the inverse conversion can be applied.

The conversion is based on the Beer-Lambert law, which relates the attenuation of light (light source - lamp) to the properties of the material through which the light is traveling (tissue sample). In our case it is a representation of the tissue thickness based on the observed light (the acquired image).

Parameters:

- 1.1 Image - the input image
- 1.2 Round mode - specify the rounding method to be used in the conversion process (default = RoundZero). This parameter is deprecated and will be removed in the future versions!

- 1.3 Result of "…" - the result of the operation

Effect:

- Converts an image from the RGB color-space to Optical Density representation.

Example:

- Input:

The image is a brightfield color image (8-bit, 3-channel) representing a small region from a colon sample.

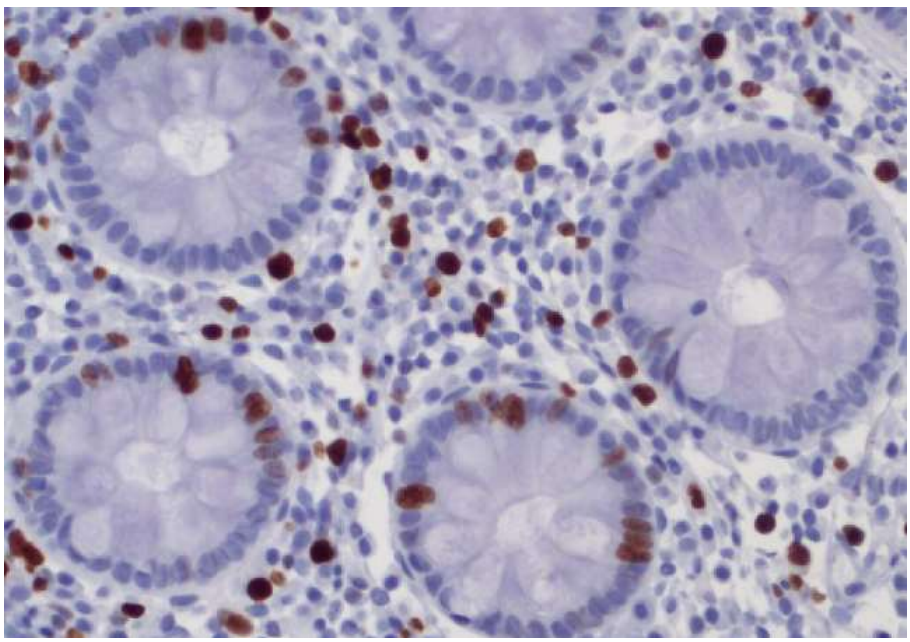


Figure 129 – Colon sample (RGB color-space)

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a color image (8-bit, 3-channel), matching the input.

Note: OD can be applied to RGB images because the formula is used on each channel (R, G, B), but theoretically it needs to be applied on a grayscale image (RGB converted to Grayscale). The engine allows applying the OD transformation on a 3 channels image, each channel being transformed independently.

Note	OD can be applied to RGB images because the formula is used on each channel (R, G, B), but theoretically it needs to be applied on a grayscale image (RGB converted to Grayscale). The engine allows applying the OD transformation on a 3 channels image, each channel being transformed independently.

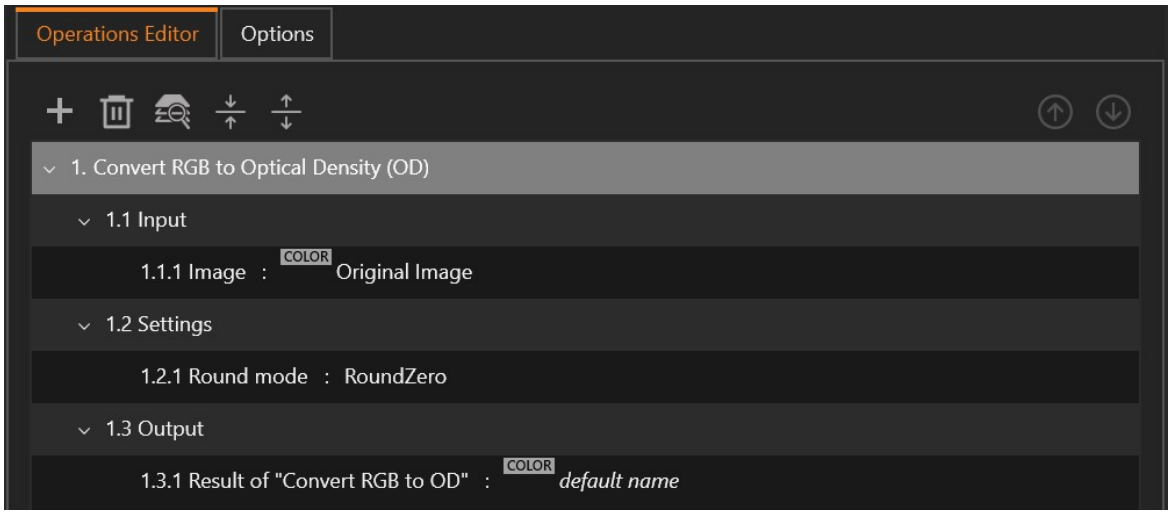


Figure 130 – Engine settings

- Output:

The result represents the converted input into OD representation.

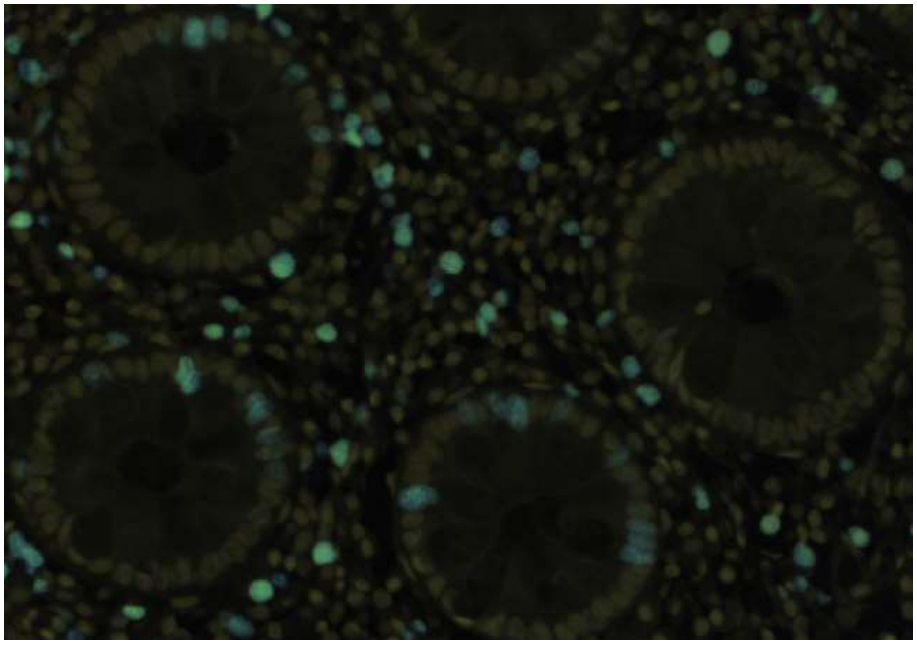


Figure 131 – Result of Convert RGB to Optical Density (OD) engine

5.3. Filters

Anisotropic filter

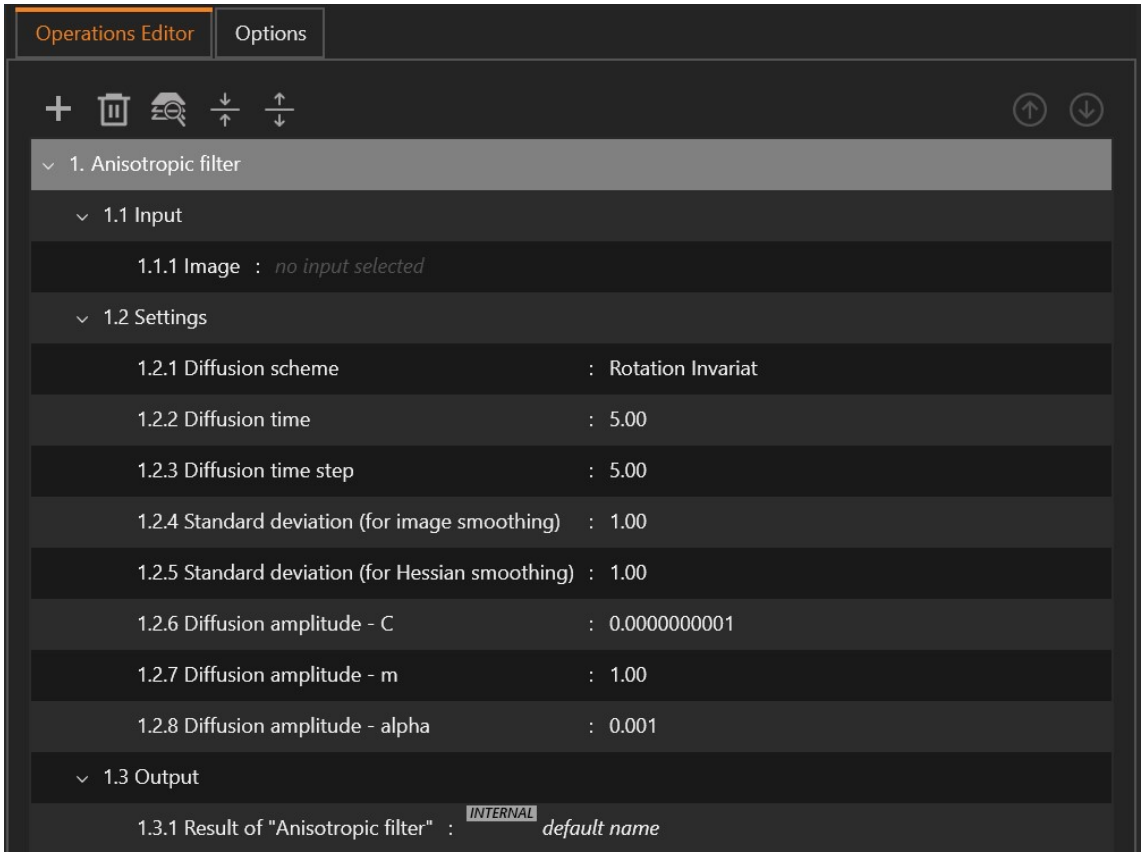


Figure 132 – Anisotropic filter

Where it can be found:

Basic Operations Module ► Filters ► Anisotropic filter

Description:

Anisotropic filter is a filter operation available in Basic Operations Module.

This operation performs anisotropic filtering on the image and places the result in the output.

Anisotropic filtering (or anisotropic diffusion) is a method used to reduce image noise while preserving significant information like region edges, lines, etc., and smooth along the image edges removing gaps caused by noise.

Parameters:

- 1.1 Image - the input image

- 1.2 Diffusion Scheme - specify the diffusion scheme used (default = Rotation Invariant)
- 1.3 Diffusion time - the total diffusion time. Using a small value may leave unfiltered some noise artefacts, while using a high value may results an image that have too much blur (default = 5);
- 1.4 Diffusion time step - the diffusion time step size (default = 5);
- 1.5 Standard deviation (for image smoothing) - the standard deviation of the gaussian smoothing before calculation of the image Hessian (default = 1);
- 1.6 Standard deviation (for Hessian smoothing - the standard deviation of the gaussian smoothing of the Hessian (default = 1);
- 1.7 Diffusion amplitude – C - represents the conductance parameter which controls how much each iteration smooths across edges (determines the amplitude of the diffusion smoothing). Smaller the value more strongly the filter will preserve edges while for higher values, the filter will cause smoothing across edges (default = 1e-10);
- 1.8 Diffusion amplitude - m - determines the amplitude of the diffusion smoothing (default = 1);
- 1.9 Diffusion amplitude - alpha - determines the amplitude of the diffusion smoothing (default = 0.001).
- 1.10 Result of "…" - the result of the operation

Effect:

Reduces the noise in the image while preserving details and smooths along image edges removing gaps caused by noise.

Example:

- Input:

The image is a fluorescence grayscale image (8-bit, 1-channel) representing the Rhodamine channel of an UV irradiated skin sample.

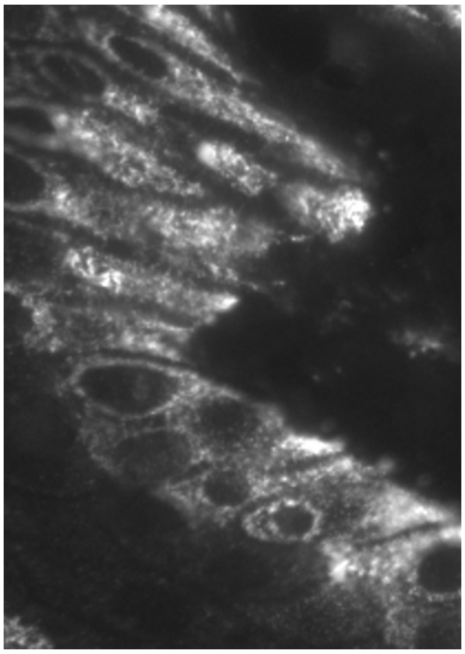


Figure 133 – UV irradiated skin sample – Rhodamine channel

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a grayscale image (8-bit, 1-channel), matching the input.

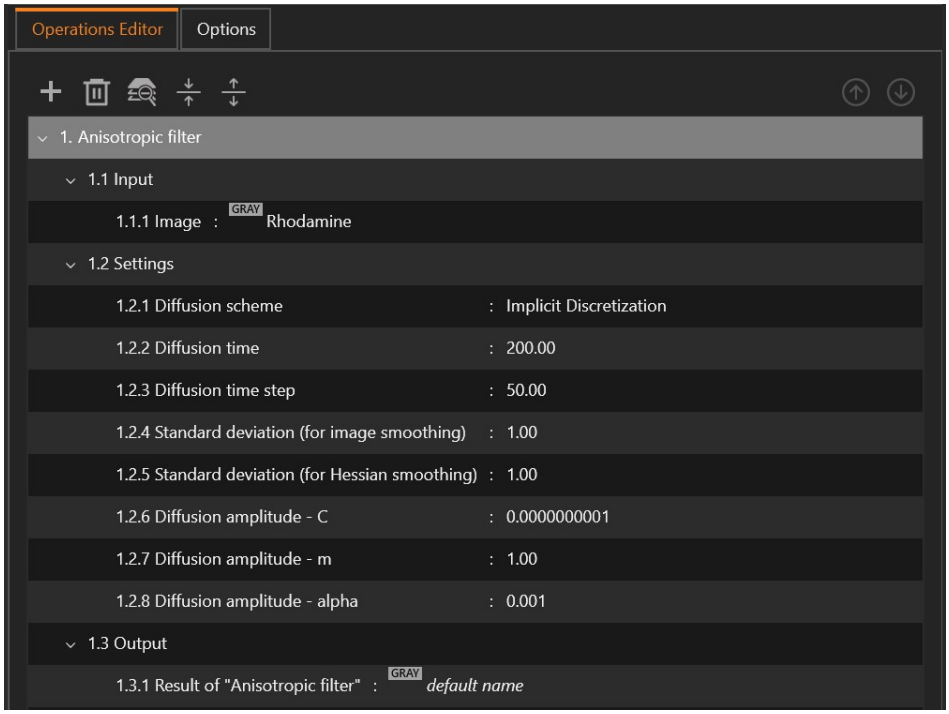


Figure 134 – Engine settings

- Output:

Figure below shows the result of anisotropic filter applied on the input image.

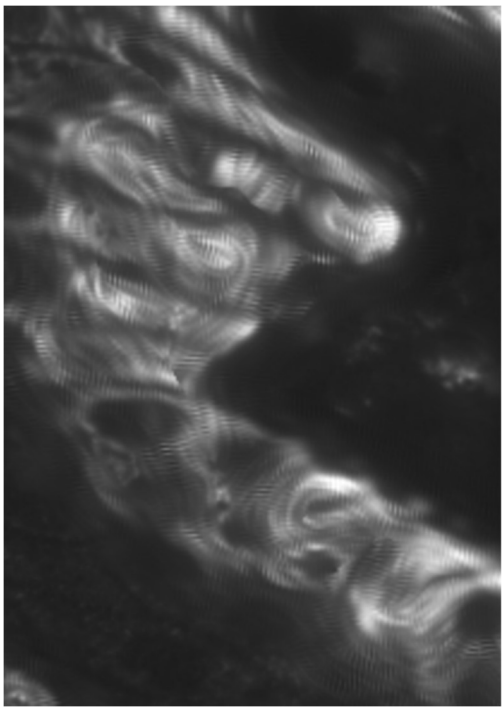


Figure 135 – Result of Anisotropic filter engine

Average filter

Figure 136 – Filter Average

Where it can be found:

Basic Operations Module ► Filters ► Average filter

Description:

Average filter is a filter operation available in Basic Operations Module.

This operation filters an image using an average kernel and places the result in the output.

Each pixel in the output image represents the average value of the entire input pixel taken in the neighborhood of the processed pixel.

$$\text{Kernel 3x3 (radius} = 1) \rightarrow \begin{pmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{pmatrix}$$

Parameters:

- 1.1 Image - the input image
- 1.2 Kernel radius (px) - determines the size of processing window (default = 1) using the formulas:

$$Kernel\ width = 2 * \boldsymbol{Kernel\ radius(px)} + 1$$

$$Kernel\ height = 2 * \boldsymbol{Kernel\ radius(px)} + 1$$

- 1.3 Kernel shape - specifies the shape of the kernel (default = Disk)
- 1.4 Result of "…" - the result of the operation

Effect:

Blurs a grayscale image using an average kernel.

Example:

- Input:

The image is a grayscale image (8-bit, 1-channel) representing a magnified nucleus from a colon sample. The image was generated using the Color Separation engine on a brightfield color image.

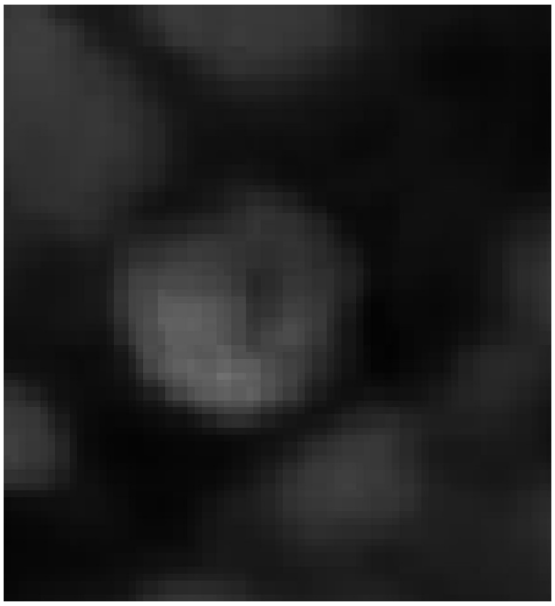


Figure 137 – Magnified nucleus (colon sample)

- Engine settings:

Figure below shows the engine settings used for this example.

An average filter with a radius of 3 is used (size = [7, 7]).

The engine's result is a grayscale image (8-bit, 1-channel), matching the input.

Figure 138 – Engine settings

- Output:

Figure below shows the result of average filter applied on the input image.

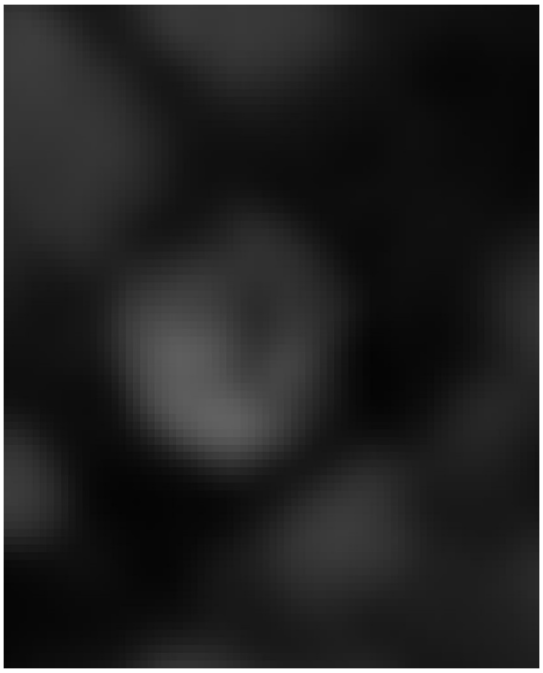


Figure 139 – Result of Average filter engine

Bilateral filter

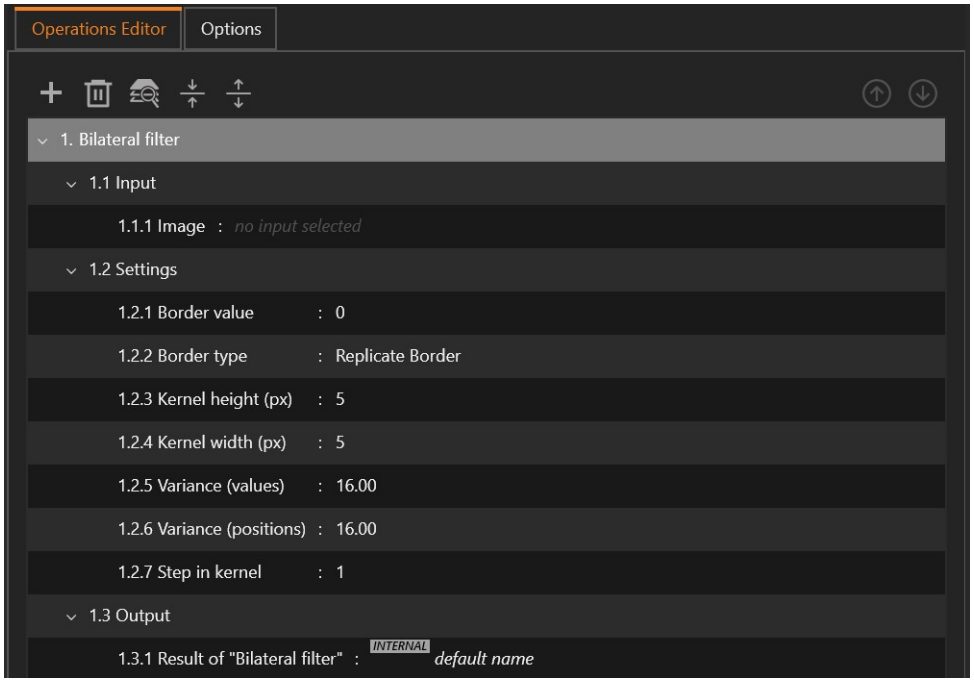


Figure 140 – Bilateral Filter

Where it can be found:

Basic Operations Module ► Filters ► Bilateral filter

Description:

Bilateral filter is a filter operation available in Basic Operations Module.

This operation performs bilateral filtering on the image and places the result in the output.

Bilateral filter is a non-linear filter used for attenuating image noise, with the advantage of preserving edges. Each resulted pixel intensity is defined by a weighted average of neighboring pixels intensities. The particularity of this filter is that the weights are adjusted based on the Euclidian distance of pixels as well as based on the intensity differences of pixels.

Parameters:

- 1.1 Image - the input image
- 1.2 Border Value - the value used to set to the pixels that are on border (default = 0);
- 1.3 Border type - specifies the method used that create the border (default = Wrap Border)
- 1.4 Kernel height (px) - the height of the processing window (kernel) (default = 5);
- 1.5 Kernel width (px) - the width of the processing window (kernel) (default = 5);
- 1.6 Variance (values) - the variance for differences of pixel values (default = 16);
- 1.7 Variance (positions) - the variance for pixels positions (default = 16);
- 1.8 Step in kernel - the processing step in the filter kernel (default = 1). This parameter is deprecated and will be removed in the future versions!
- 1.9 Result of ".." - the result of the operation

Using high values for Variance (values) and Variance (positions) will generate an image that will look "cartoonish", while using small values will generate a barely visible effect.

Effect:

Reduces the noise in the image and applies a smoothing effect while preserving relevant details.

Example:

- Input:

The image is a grayscale image (8-bit, 1-channel) showing a small region from the DAPI channel of a colorectal cancer sample.

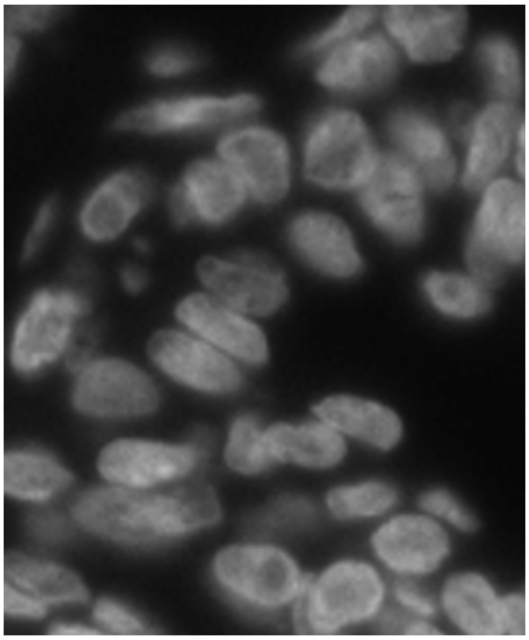


Figure 141 – Colorectal cancer sample – DAPI channel

- Engine settings:

Figure below shows the engine settings used for this example.

A bilateral filter with a size of 5x5 and variances of 750 is used.

The engine's result is a grayscale image (8-bit, 1-channel), matching the input.

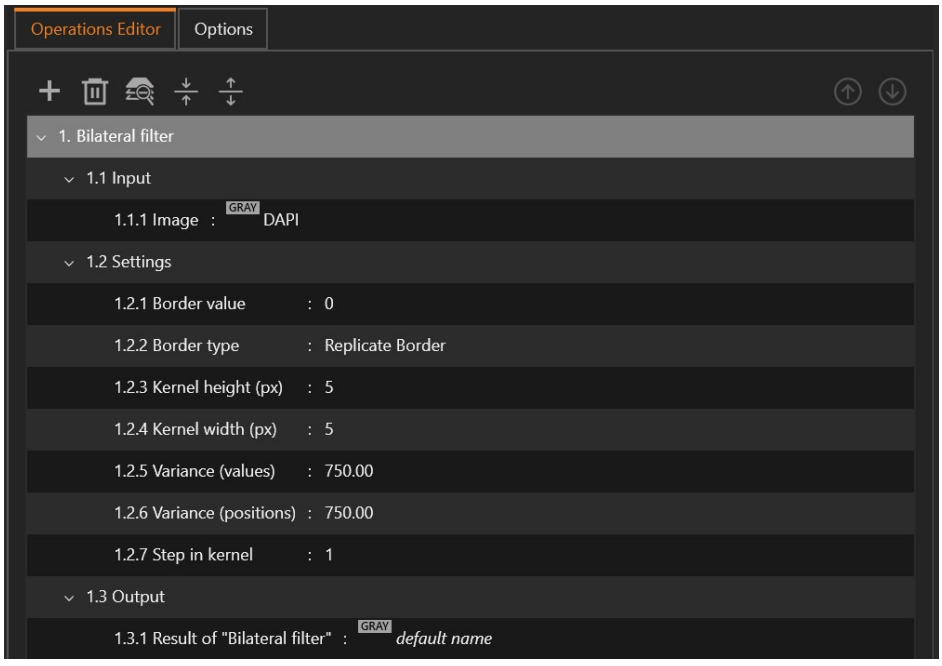


Figure 142 – Engine settings

- Output:

Figure below shows the result of bilateral filter applied on the input image.

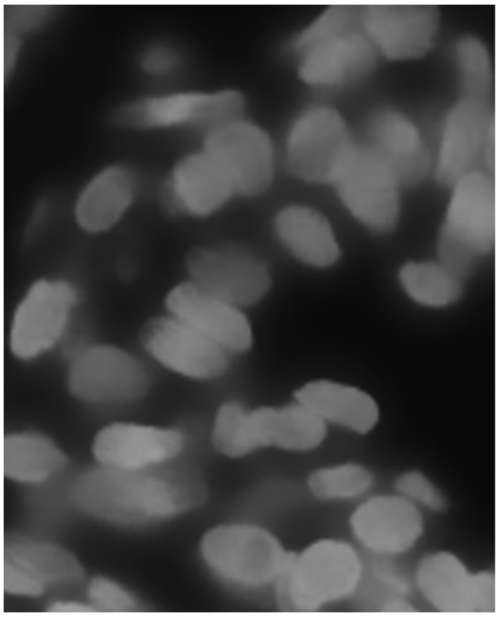


Figure 143 – Bilateral Filter (DAPI)

Canny edge detector

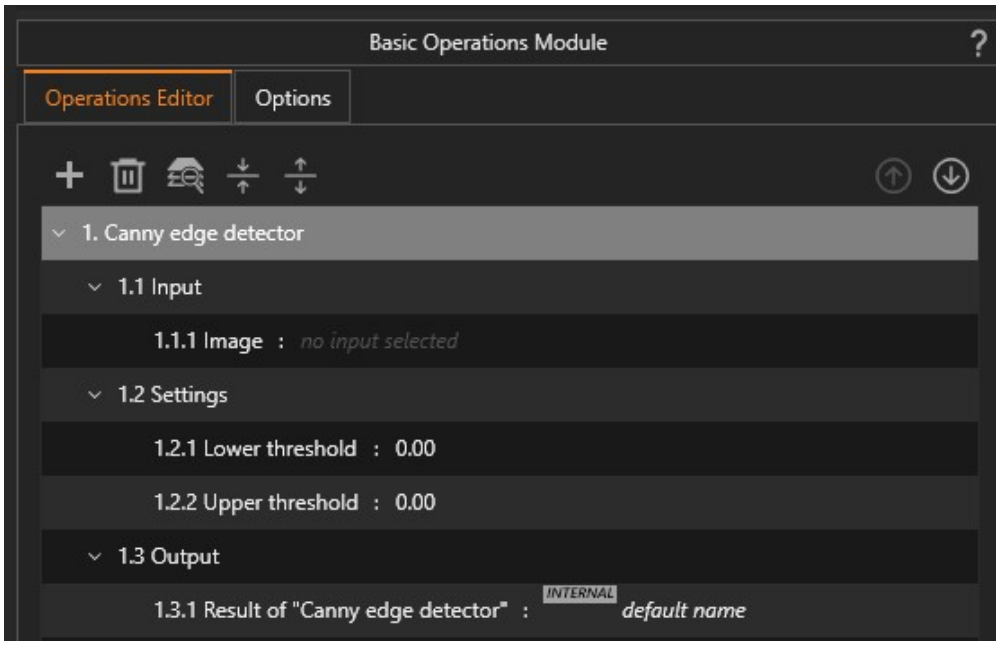


Figure 144 – Canny edge Detector

Where it can be found:

Basic Operations Module ► Filters ► Canny edge detector

Description:

Canny edge detector is a filter operation available in Basic Operations Module.

This operation uses the Canny algorithm to detect the edges present in the image and places the result in the output.

The process of the Canny edge detection algorithm can be broken in the following steps:

- Find the intensity gradients of the image;
- Apply non-maximum suppression to get rid of spurious response to edge detection;
- Apply double threshold to determine potential edges;
- Track edge by hysteresis - finalize the detection of the edges by suppressing all the other edges that are weak and not connected to strong edges.

Parameters:

- 1.1 Image - the input image
- 1.2 Lower threshold: - the lower threshold value for edges detection (default = 0);
- 1.3 Upper threshold: - the upper threshold value for edges detection (default = 0).
- 1.4 Result of "…" - the result of the operation

J. Canny recommends that the ratio of the high to low limit be in the range two or three to one, based on predicted signal-to-noise ratios.

Effect:

Finds edges present in a grayscale image.

Example:

- Input:

The image is a grayscale image (8-bit, 1-channel) showing a small region from the DAPI channel of a colorectal cancer sample.

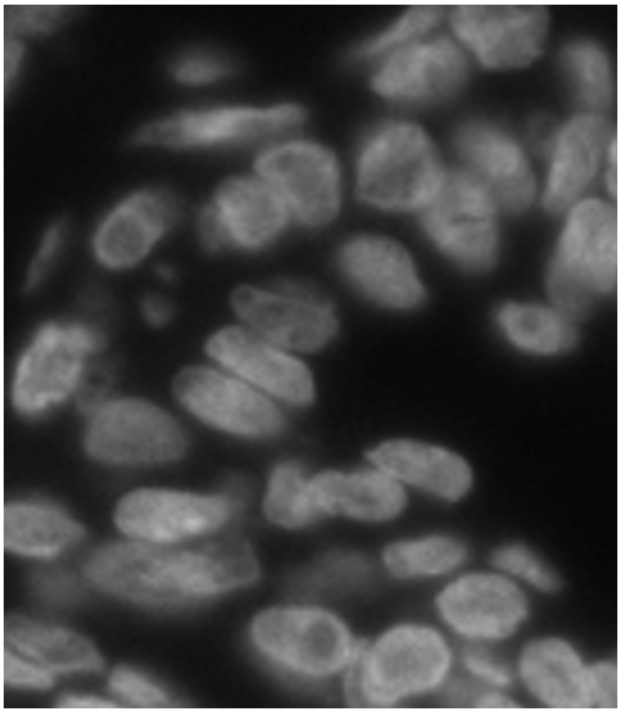


Figure 145 – MulCResult (DAPI)

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a grayscale image (8-bit, 1-channel), matching the input.

Figure 146 – Engine settings

- Output:

Figure below shows the result of the Canny edge detector applied on the input image.

Figure 147 – Canny edge Detector on Gray

Custom filter

Figure 148 – Custom Filter

Where it can be found:

Basic Operations Module ► Filters ► Custom filter

Description:

Custom filter is a filter operation available in Basic Operations Module.

This operation filters an image using a custom defined kernel and places the result in the output.

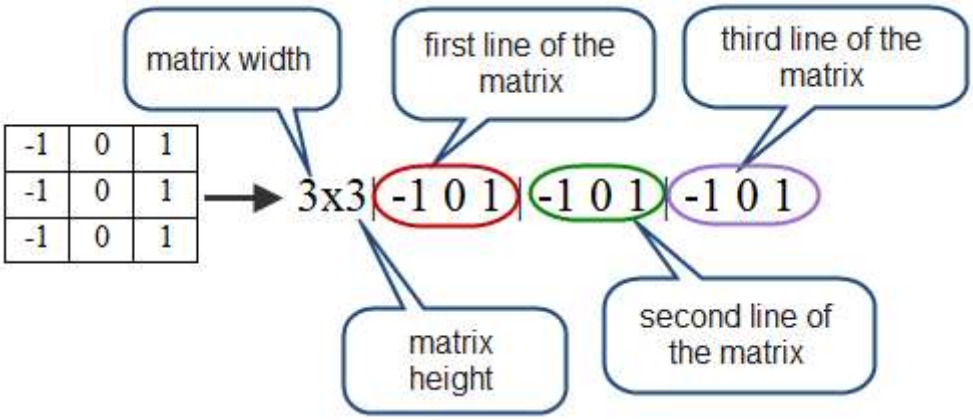
A custom filter consists of a string of all relevant information of the desired filter (kernel width and height and weights). The values used for the kernel size must always be integers, while the weights can take any value (usually 0 and 1, but other values are also eligible, like 0.5 or 1.25).

Parameters:

- 1.1 Image - the input image
- 1.2 Filter - the text defining the custom filter.
- 1.3 Result of "…" - the result of the operation

Example of a custom filter:

Filter = 3 x 3 | -1 0 1 | -1 0 1 | -1 0 1



First entry must be Width x Height followed by matrix elements. Each matrix line is delimited by "|" character.

Effect:

Performs image filtering using custom defined filters.

Example

- Input:

The image below is a fluorescence grayscale image (8-bit, 1-channel) showing a small region from the DAPI channel of a colorectal cancer sample.

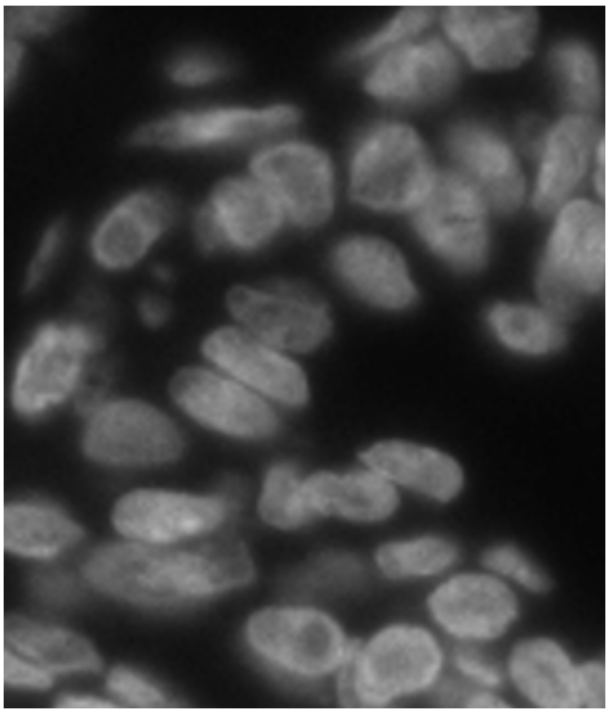


Figure 149 – Colorectal cancer sample – DAPI channel

- Engine settings:

Figure below shows the engine settings used for this example.

For this example, a custom filter with a 3x3 size is used:

$$\begin{pmatrix} 2 & 0 & 2 \\ 0 & -8 & 0 \\ 2 & 0 & 2 \end{pmatrix}$$

The engine's result is a grayscale image (8-bit, 1-channel), matching the input.

Figure 150 – Engine settings

- Output:

Figure below shows the result of the custom filter applied on the input image.

Figure 151 – Result of Custom filter engine

Gauss filter

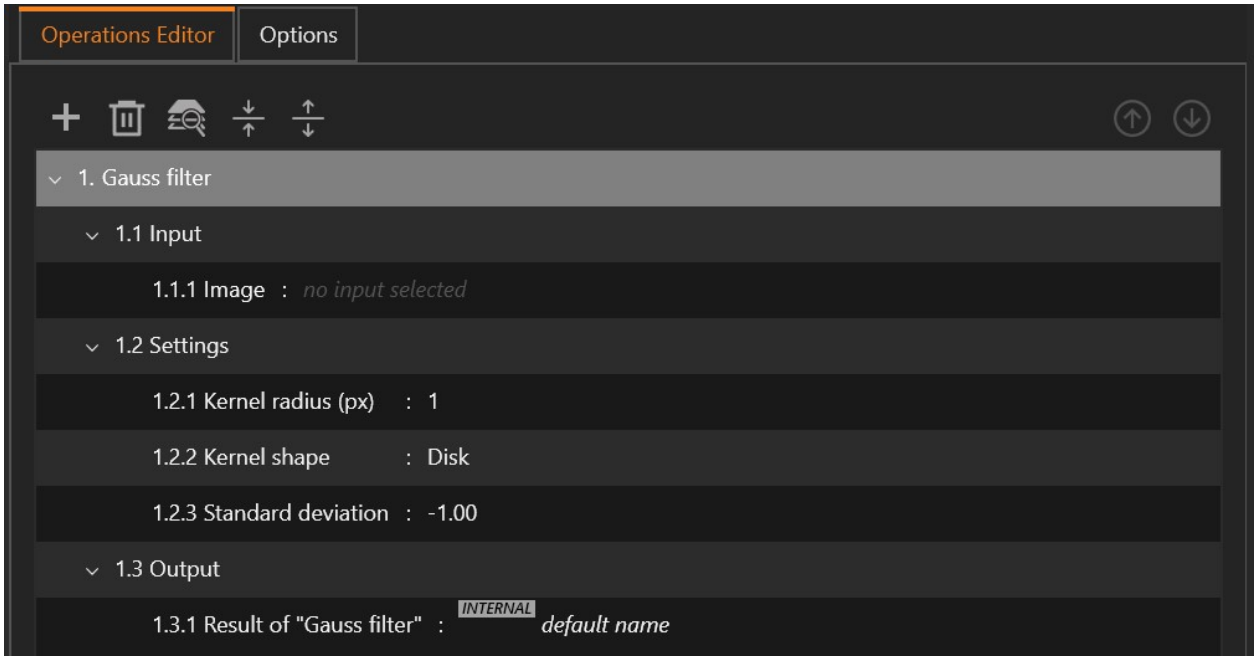


Figure 152 – Filter Gauss

Where it can be found:

Basic Operations Module ► Filters ► Gauss filter

Description:

Gauss filter is a filter operation available in Basic Operations Module.

This operation filters an image using a Gaussian kernel and places the result in the output.

Each pixel in the output image represents the weighted average value of the entire input pixel taken in the neighborhood of the processed pixel. The weights values are generated using the Gaussian distribution formula:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$\text{Kernel 5x5 } (\text{radius} = 2, \text{std} = 0.85) \rightarrow \begin{pmatrix} 0.0035 & 0.0123 & 0.0210 & 0.0123 & 0.0035 \\ 0.0123 & 0.0543 & 0.0910 & 0.0543 & 0.0123 \\ 0.0210 & 0.0910 & 0.2224 & 0.0910 & 0.0210 \\ 0.0123 & 0.0543 & 0.0910 & 0.0543 & 0.0123 \\ 0.0035 & 0.0123 & 0.0210 & 0.0123 & 0.0035 \end{pmatrix}$$

Parameters:

- 1.1 Image - the input image
- 1.2 Kernel radius (px) - determines the size of processing window (default = 1) using the formulas:

$$Kernel\ width = 2 * \boldsymbol{Kernel\ radius(px)} + 1$$

$$Kernel\ height = 2 * \boldsymbol{Kernel\ radius(px)} + 1$$

- 1.3 Kernel shape - specifies the shape of the kernel (default = Disk)
- 1.4 Standard deviation - indicates the amount of variation or dispersion of a data set. A value close to 0 indicates that the data points tend to be very close to the mean of the set, while a high value indicates that the data points are spread out over a wider range of values (default = 1)
- 1.4 Result of "…" - the result of the operation

Effect:

Blurs a grayscale image using a Gaussian kernel.

Example:

- Input:

Image below is a grayscale image (8-bit, 1-channel) representing a magnified nucleus from a colon sample. The image was generated using the Color Separation engine on a brightfield color image.

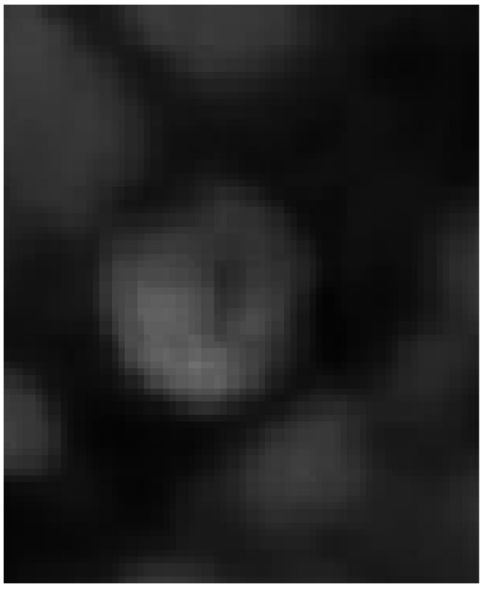


Figure 153 – Magnified nucleus (colon sample)

- Engine settings:

Figure below shows the engine settings used for this example.

A Gaussian filter with a radius of 3 is used (size = [7, 7]) and a standard deviation of 1.5 is used.

The engine's result is a grayscale image (8-bit, 1-channel), matching the input.

Figure 154 – Engine settings

- Output:

The figure below shows the result of gauss filter (size = 7x7) applied on the input image.

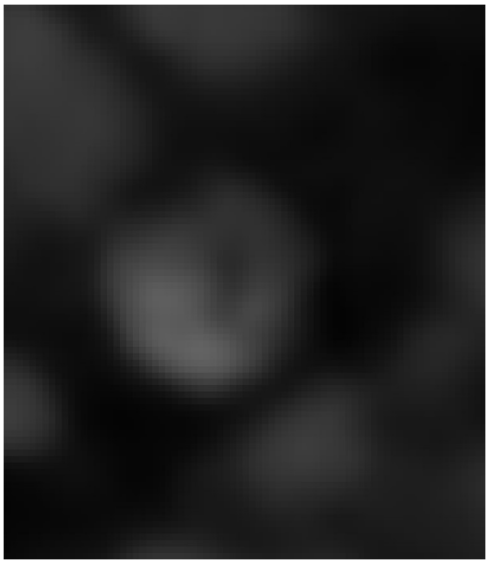


Figure 155 – Result of Gauss filter engine

Laplace filter

Figure 156 – Filter Laplace

Where it can be found:

Basic Operations Module ► Filters ► Laplace filter

Description:

Laplace filter is a filter operation available in Basic Operations Module.

This operation filters an image using a Laplacian kernel and places the result in the output.

The Laplace operator is a second-order differential used to highlight image details defined by quick intensity changes (e.g. edges).

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

where f denotes the image.

The operation allows the usage of two predefined kernel:

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} or \begin{pmatrix} -1 & -3 & -4 & -3 & -1 \\ -3 & 0 & 6 & 0 & -3 \\ -4 & 6 & 20 & 6 & -4 \\ -3 & 0 & 6 & 0 & -3 \\ -1 & -3 & -4 & -3 & -1 \end{pmatrix}$$

Source: Spatial Filters - Laplacian/Laplacian of Gaussian (ed.ac.uk)

Parameters:

- 1.1 Image - the input image
- 1.2 Kernel size (px) - specifies which of the two possible kernels to be used (default = 3x3)
- 1.3 Result of "…" - the result of the operation

Effect:

Highlights regions of rapid intensity change in a grayscale image.

Example:

- Input:

The image below is a fluorescence grayscale image (8-bit, 1-channel) showing a small region from the DAPI channel of a colorectal cancer sample.

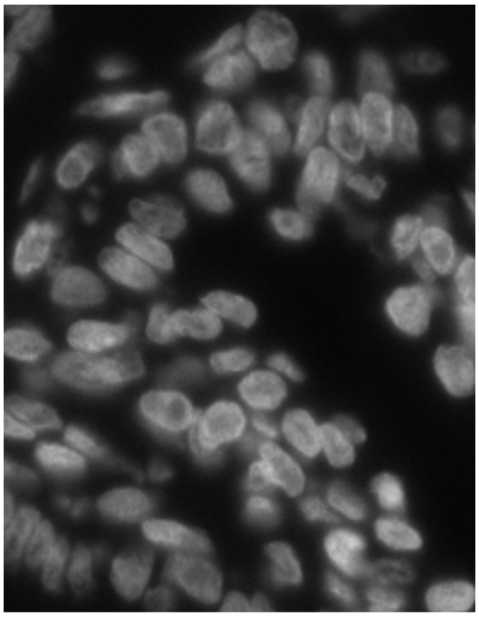


Figure 157 – Colorectal cancer sample – DAPI channel

- Engine settings:

Figure below shows the engine settings used for this example.

A Laplace filter with a size of 5x5 is used.

The engine's result is a grayscale image (8-bit, 1-channel), matching the input.

Figure 158 – Engine settings

- Output:

Figure below shows the result of a Laplace filter (5x5) applied on the input image.

Figure 159 – Result of Laplace filter engine

Median filter

Figure 160 – Filter Median

Where it can be found:

Basic Operations Module ► Filters ► Median filter

Description:

Median filter is a filter operation available in Basic Operations Module.

This operation filters an image using a median filter and places the result in the output.

Each pixel in the output image takes the median value of all the input pixel values taken in the neighborhood (defined by Kernel radius (px)) of the processed pixel.

Parameters:

- 1.1 Image - the input image
- 1.2 Kernel radius (px) - determines the size of processing window (default = 1) using the formulas:

$$Kernel\ width = 2 * \boldsymbol{Kernel\ radius(px)} + 1$$

$$Kernel\ height = 2 * \boldsymbol{Kernel\ radius(px)} + 1$$

- 1.3 Kernel shape - specifies the shape of the kernel (default = Disk)
- 1.4 Result of "…" - the result of the operation

Effect:

Reduces noise present in the image while it conserves certain forms of borders, but it does not generate new levels of gray.

Example:

- Input:

Image below is a fluorescence grayscale image (8-bit, 1-channel) showing a small region from the DAPI channel of a colorectal cancer sample.

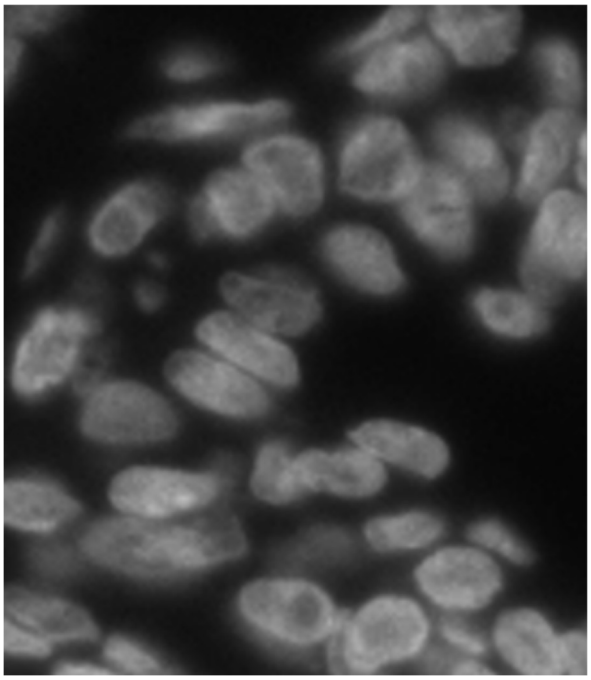


Figure 161 – Colorectal cancer sample - DAPI channel

- Engine settings:

Figure below shows the engine settings used for this example.

A median filter with radius = 2 (size = 5x5) is used.

The engine's result is a grayscale image (8-bit, 1-channel), matching the input.

Figure 162 – Engine settings

- Output:

Figure below shows the result of Median filter (5x5) applied on the input image.

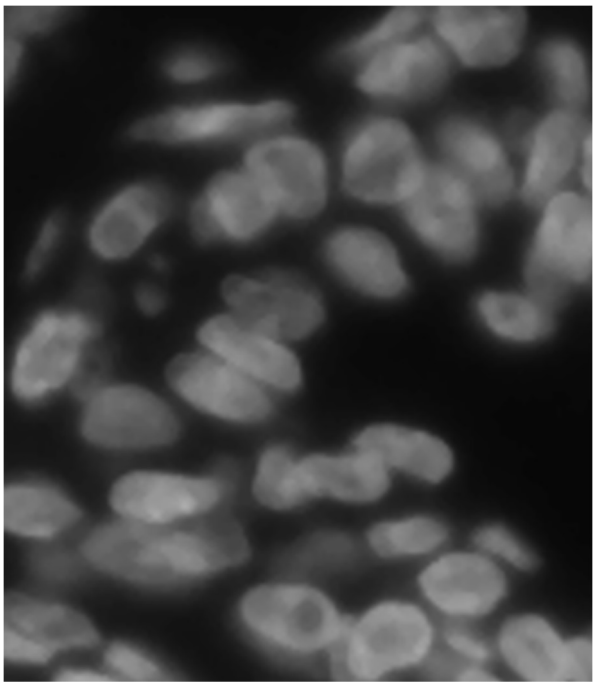


Figure 163 – Result of Median filter engine

Membrane detector

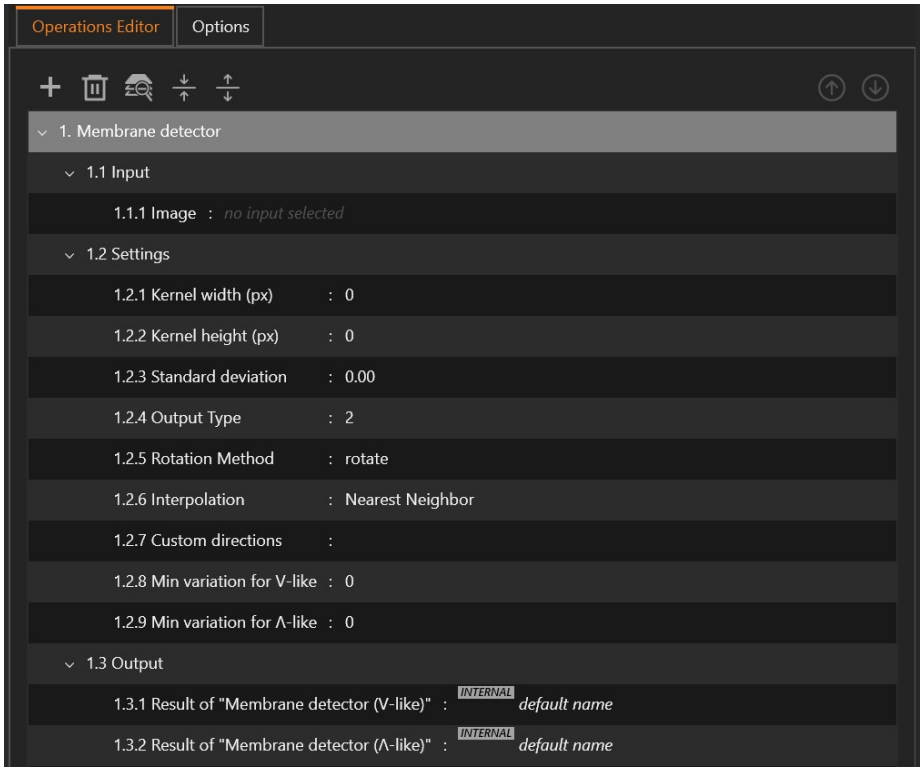


Figure 164 – Membrane detector

Where it can be found:

Basic Operations Module ► Filters ► Membrane detector

Description:

Membrane detector is a filter operation available in Basic Operations Module.

This operation uses Membrane detector algorithm to detect membranes present in the image and places the result in the output.

Membranes are usually associated with quick double variations of intensity:

- V-like intensity variations: a quick intensity drop from high to low followed by a quick intensity increase from low to high (e.g. it can be associated with dark membranes on a bright background).
- Λ-like intensity variations: quick intensity increase from low to high followed by a quick intensity drop from high to low (e.g. it can be associated with bright membranes on a dark background).

Parameters:

- 1.1 Image - the input image
- 1.2 Kernel width (px) - specifies the width of the kernel used by the algorithm (default = 0)
- 1.3 Kernel height (px) - specifies the height of the kernel used by the algorithm (default = 0);
- 1.4 Standard deviation - the standard deviation for the Gaussian model used by the algorithm (default = 0);
- 1.5 Output type - specifies the combination of results to be generated (V-like, ∧-like, V-like and ∧-like). The only value possible is 2 (V-like and ∧-like). This parameter is deprecated and will be removed in the future versions!
- 1.6 Rotation Method - specifies the rotation method used by the algorithm (default = rotate). This parameter is deprecated and will be removed in the future versions!
- 1.7 Interpolation - specifies the interpolation method used by the algorithm (default = Nearest Neighbor). This parameter is deprecated and will be removed in the future versions!
- 1.8 Custom Directions - specifies the directions to be used by the algorithm to search for membrane-like structures.

Example:

where:

- the first number (“4”) represents the number of directions
- the symbol (“|”) represents the separator between the number of directions and the enumeration of the directions
- the rest of the values (“0 90 45 135”) represents the angles used by the algorithm to search for membrane-like structures
- Min variation for V-like - specifies the minimum variation threshold for V-like membranes (default = 0);
- Min variation for ∧-like - specifies the minimum variation threshold for ∧-like membranes (default = 0);

Effect:

Detects membrane-like structures present in a grayscale image.

Example:

- Input:

The image below is a grayscale image (8-bit, 1-channel) representing the HER2 marker in breast cancer sample. The image was generated using the Color Separation engine on a brightfield color image.

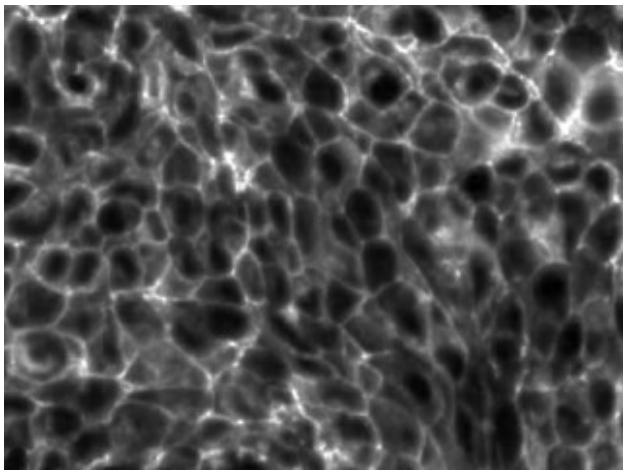


Figure 165 – Breast cancer sample – HER2 channel

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's results are mask images (8-bit, 1-channel).

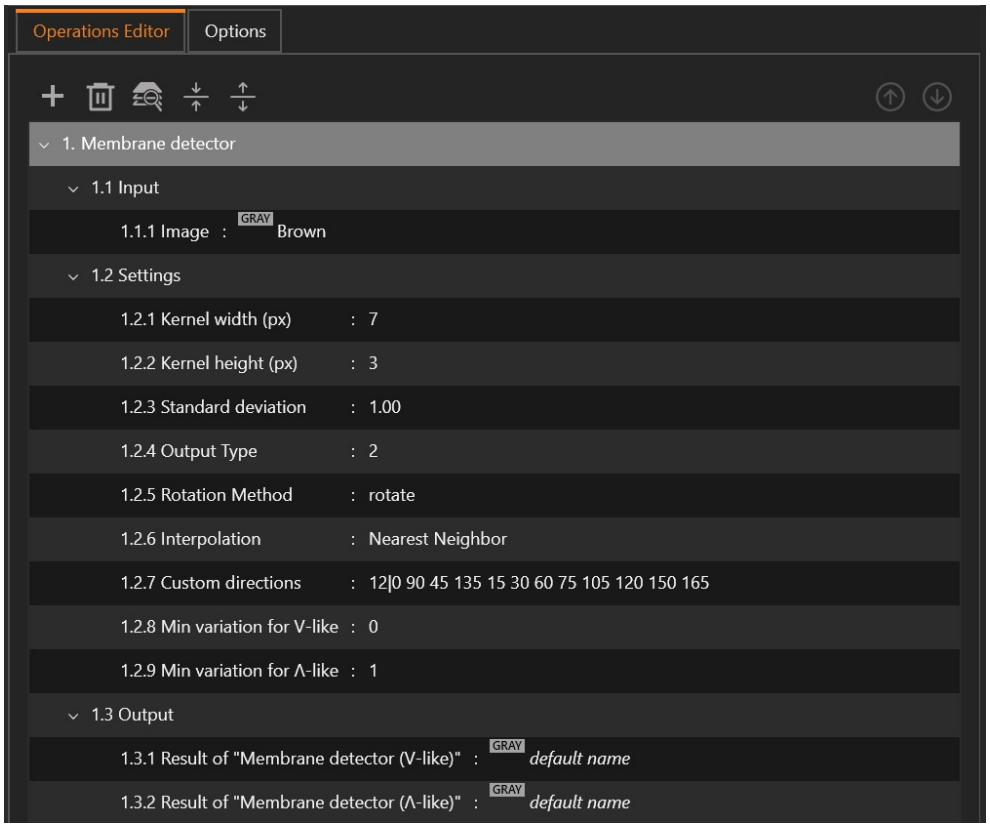


Figure 166 – Engine settings

- Output:

The results show the detected membrane structures overlaid on the input image.

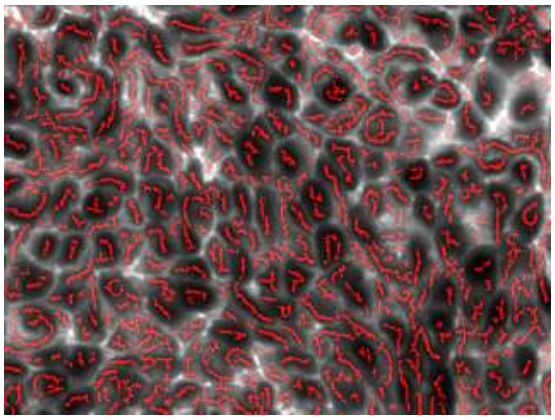


Figure 167 – Result of Membrane detector engine (V-like)

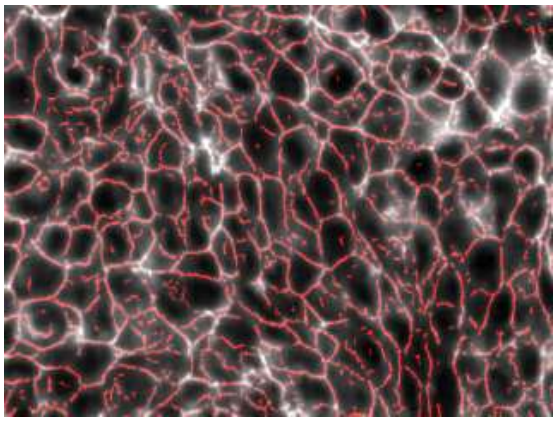


Figure 168 – Result of Membrane detector engine (Λ-like)

Sharp filter

Figure 169 – Sharp filter

Where it can be found:

Basic Operations Module ► Filters ► Sharp filter

Description:

Sharp filter is a filter operation available in Basic Operations Module.

This operation enhances image details by generating a better contrast. Can also be used to improve the quality of slightly out of focus images.

Parameters:

- 1.1 Image - the input image
- 1.2 Kernel radius (px) - determines the size of processing window (default = 7) using the formulas:

$$Kernel\ width = 2 * \boldsymbol{Kernel\ radius(px)} + 1$$

$$Kernel\ height = 2 * \boldsymbol{Kernel\ radius(px)} + 1$$

- 1.3 Result of "…" - the result of the operation

Effect:

Enhance image details.

Example:

- Input:

The image below is a brightfield color image (8-bit, 3-channel) representing a small region from a colon sample.

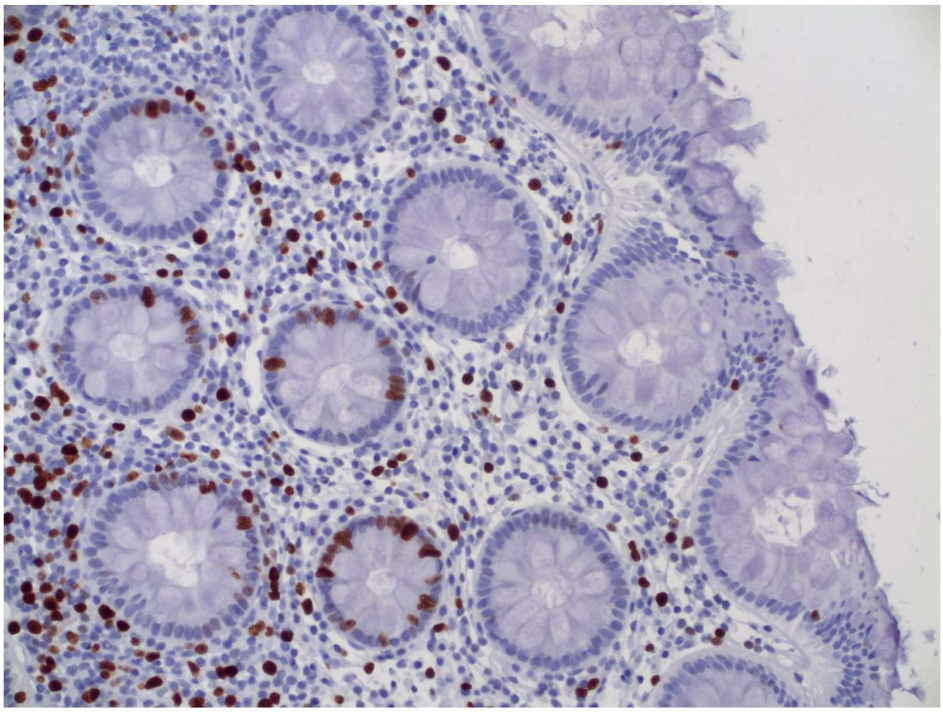


Figure 170 – Sharp: Input image

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a color image (8-bit, 3-channel), matching the input.

Figure 171 – Sharp: parameters

- Output:

The result has the details enhanced and a better contrast.

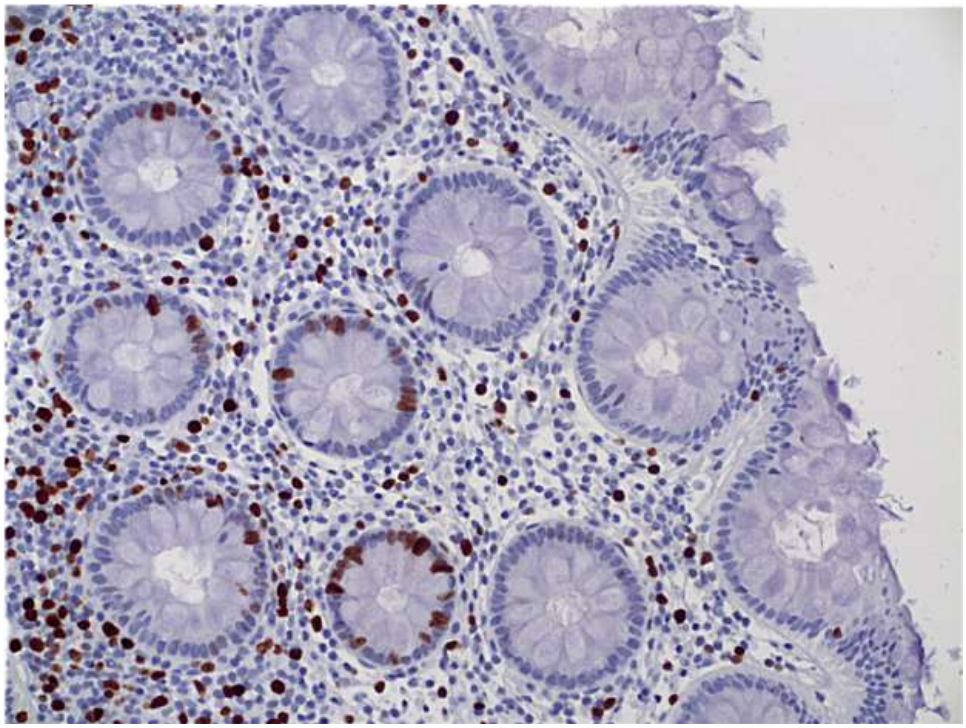


Figure 172 – Sharp: Output image

Sobel filter (cross)

Figure 173 – Sobel Cross Border Filter

Where it can be found:

Basic Operations Module ► Filters ► Sobel filter (cross)

Description:

Sobel filter (cross) is a filter operation available in Basic Operations Module.

This operation filters an image using a second cross derivative Sobel operator and places the result in the output.

The operation allows the usage of two predefined kernels:

$$\begin{pmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{pmatrix} or \begin{pmatrix} -1 & -2 & 0 & 2 & 1 \\ -2 & -4 & 0 & 4 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 4 & 0 & -4 & -2 \\ 1 & 2 & 0 & -2 & -1 \end{pmatrix}$$

They are convolved with the input image to calculate approximations of the second order derivatives (on diagonal). Pixels in the result emphasize high variations of intensity on diagonal.

Parameters:

- 1.1 Image - the input image
- 1.2 Border Value - the value used to set to the pixels that are on border (default = 0);
- 1.3 Border type - specifies the method used that create the border (default = Wrap Border)
- 1.4 Kernel size (px) - specifies which of the two possible kernels to be used (default = 3x3)
- 1.5 Result of "…" - the result of the operation

Effect:

Detects diagonal edges present in the image.

Example:

- Input:

The image below is a grayscale image (8-bit, 1-channel) showing a small region from DAPI channel of a colorectal cancer sample.

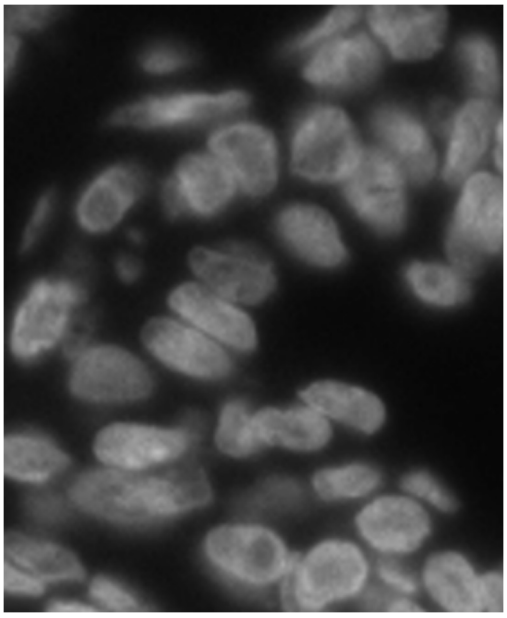


Figure 174 – DAPI channel

- Engine settings:

Figure below shows the engine settings used for this example.

A cross Sobel filter with a size of 5x5 is used.

The engine's result is a grayscale image (8-bit, 1-channel).

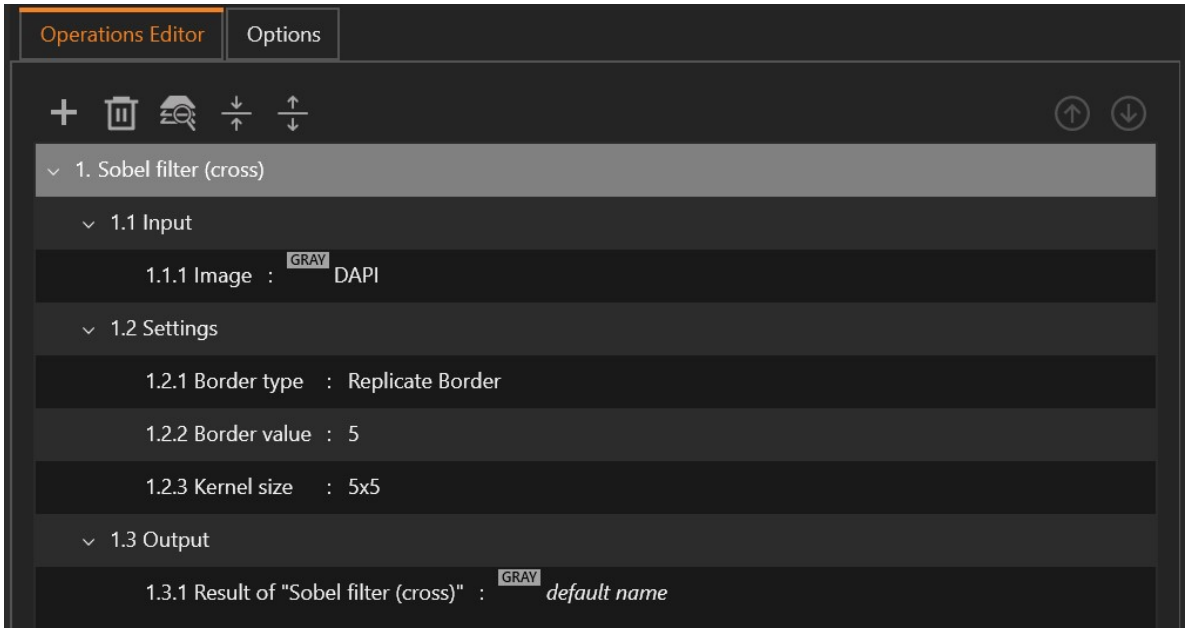


Figure 175 – Engine settings

- Output:

Figure below shows the results of a cross Sobel filter (5x5) applied on the input image.

Figure 176 – Result of Sobel filter (cross) engine

Sobel filter (horizontal)

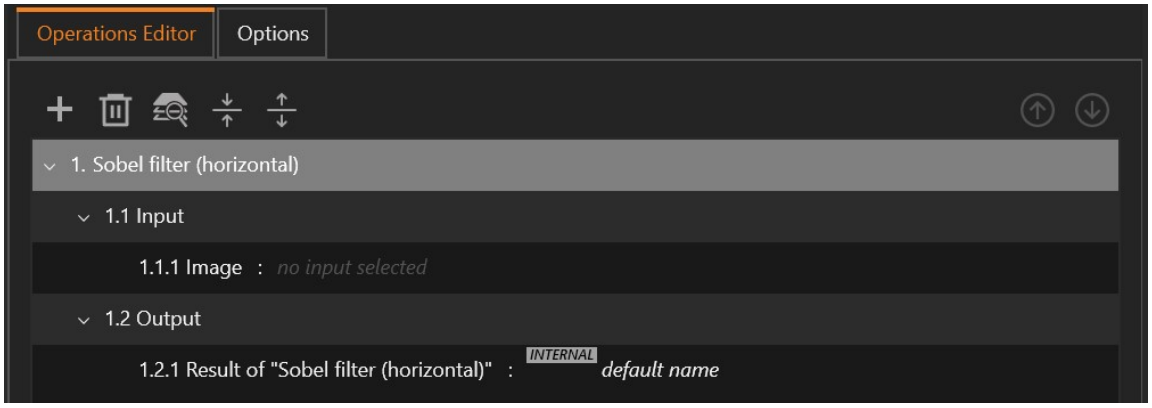


Figure 177 – Sobel Horizontal Filter

Where it can be found:

Basic Operations Module ► Filters ► Sobel filter (horizontal)

Description:

Sobel filter (horizontal) is a filter operation available in Basic Operations Module.

This operation filters an image using a horizontal Sobel operator and places the result in the output.

The operation uses a predefined 3x3 kernel:

$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

It is convolved with the input image to calculate approximations of the derivative (on horizontal). Pixels in the result emphasize high variations of intensity on the horizontal axis.

Parameters:

- 1.1 Image - the input image
- 1.2 Result of "…" - the result of the operation

Effect:

Detects horizontal edges present in the image.

Example:

- Input:

The image below is a grayscale image (8-bit, 1-channel) showing a small region from DAPI channel of a colorectal cancer sample.

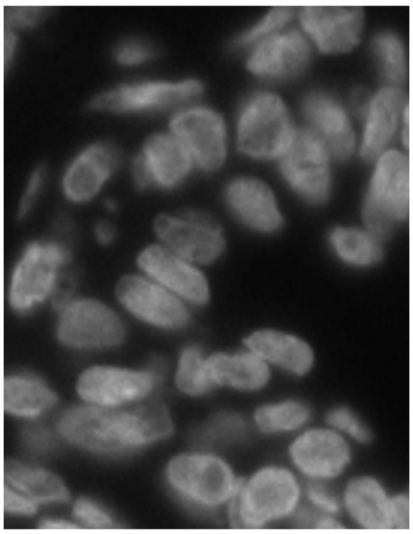


Figure 178 – Colorectal cancer sample - DAPI channel

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a grayscale image (8-bit, 1-channel).

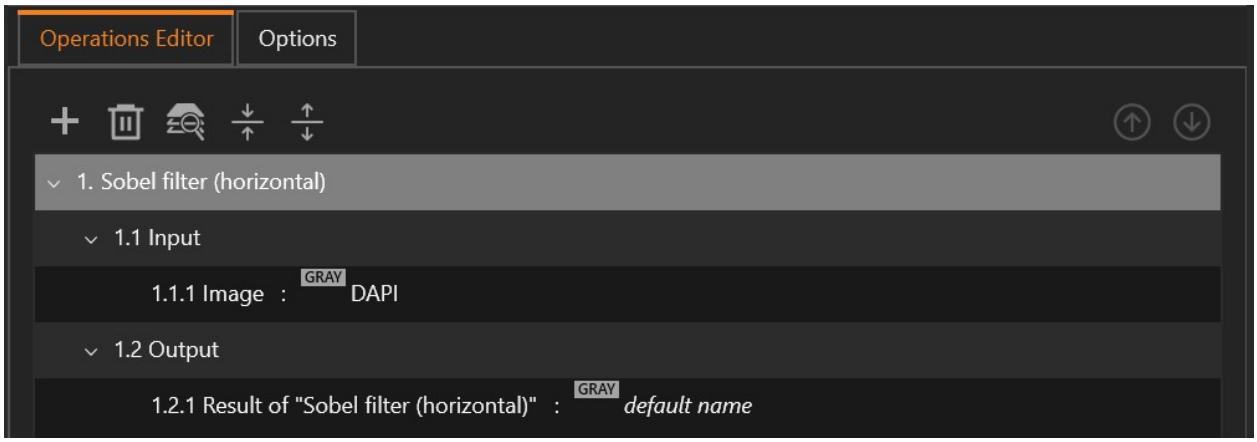


Figure 179 – Engine settings

- Output:

The result shows the results of a horizontal Sobel filter applied on the input image

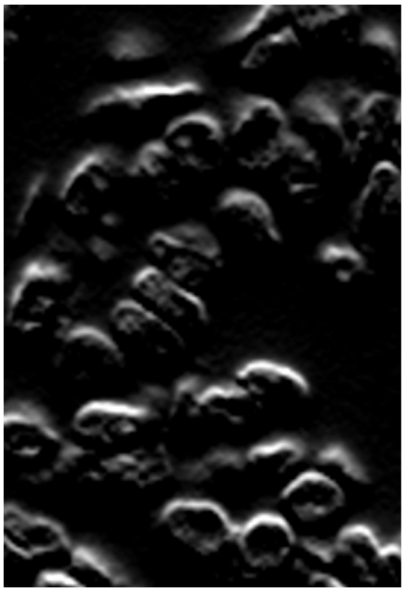


Figure 180 – Result of Sobel filter (horizontal) engine

Sobel filter (vertical)

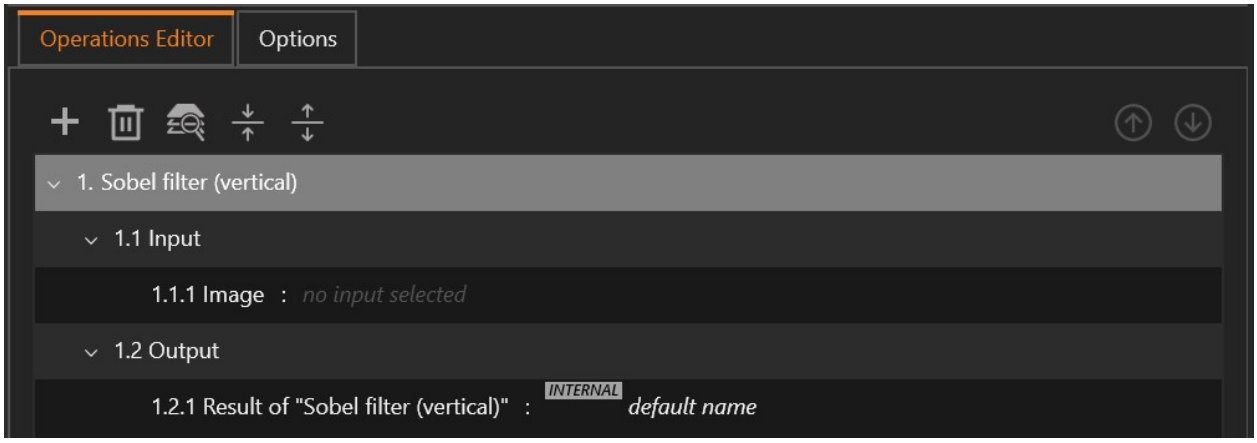


Figure 181 – Sobel Vertical Filter

Where it can be found:

Basic Operations Module ► Filters ► Sobel filter (vertical)

Description:

Sobel filter (vertical) is a filter operation available in Basic Operations Module.

This operation filters an image using a vertical Sobel operator and places the result in the output.

The operation uses a 3x3 predefined kernel:

$$\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

The kernel is convolved with the input image to calculate approximations of the derivative (on vertical). Pixels in the result emphasize high variations of intensity on vertical axis.

Parameters:

- 1.1 Image - the input image
- 1.2 Result of "…" - the result of the operation

Effect:

Detects diagonal edges present in the image.

Example:

- Input:

The image below is a grayscale image (8-bit, 1-channel) showing a small region from DAPI channel of a colorectal cancer sample.

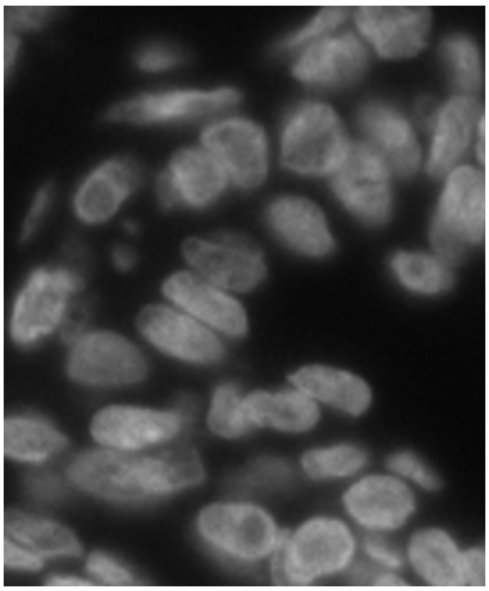


Figure 182 – Colorectal cancer sample - DAPI channel

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a grayscale image (8-bit, 1-channel).

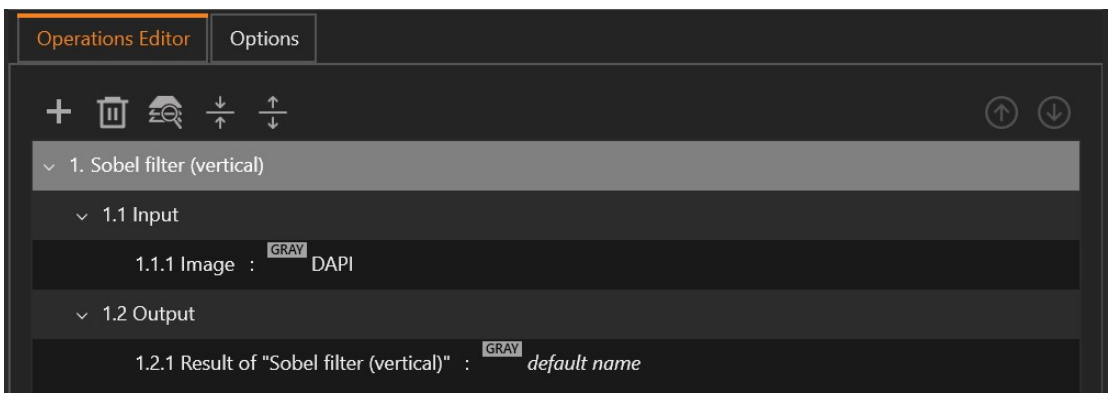


Figure 183 – Engine settings

- Output:

The figure below shows the results of a vertical Sobel filter applied on the input image.

Figure 184 – Result of Sobel filter (vertical) engine

Unsharp filter

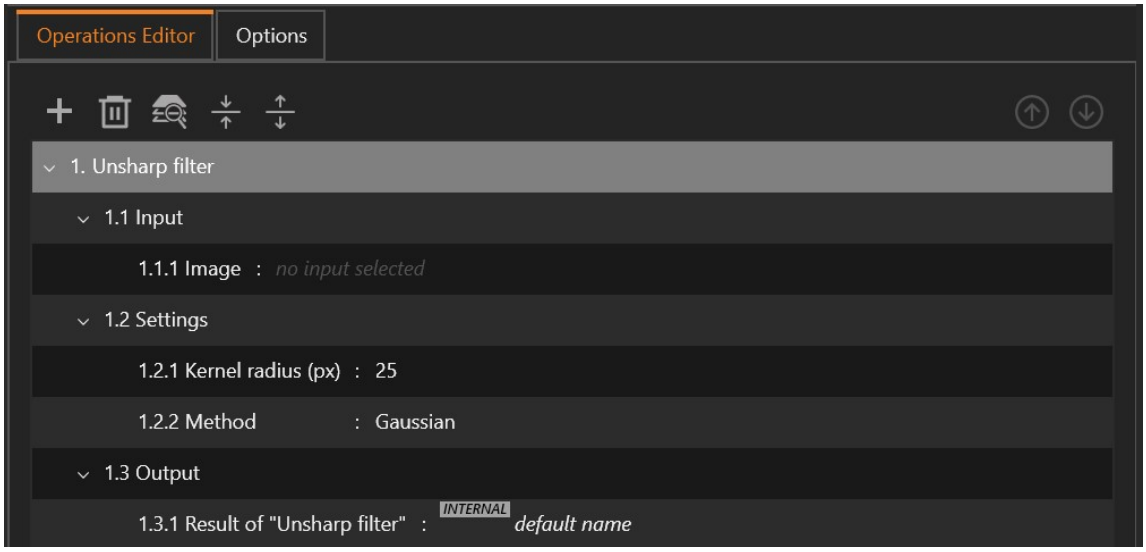


Figure 185 – Unsharp Filter

Where it can be found:

Basic Operations Module ► Filters ► Unsharp filter

Description:

Unsharp filter is a filter operation available in Basic Operations Module.

It removes or attenuates the effect of high background by generating a background model and subtracting it from the input image. The background model is generated by blurring the input image using an Average or Gaussian filter. This ensures that fine details are not present in the background model, thus the name Unsharp filter.

Parameters:

- 1.1 Image - the input image
- 1.2 Kernel radius (px) - determines the size of processing window (default = 25) using the formulas:

$$Kernel\ width = 2 * \boldsymbol{Kernel\ radius(px)} + 1$$

$$Kernel\ height = 2 * \boldsymbol{Kernel\ radius(px)} + 1$$

- 1.3 Method - specifies the blurring method used to generate the background model (default = Gaussian)
- 1.4 Result of "…" - the result of the operation

Effect:

Removes or decreases the impact of the high background in grayscale images.

Example:

- Input:

The image below is a grayscale image (8-bit, 1-channel) showing a small region from DAPI channel of a colorectal cancer sample.

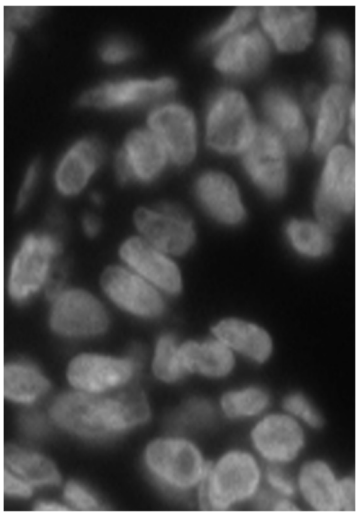


Figure 186 – Colorectal cancer sample – DAPI channel

- Engine settings:

The figure below shows the engine settings used for this example.

A Gaussian filter with a radius of 60 (size = [121 x 121]) is used to generate the background model.

The engine's result is a grayscale image (8-bit, 1-channel), matching the input.

Figure 187 – Engine settings

- Output:

Figure below shows the result of an unsharp filter applied on the input image. The high background has been attenuated (is darker).

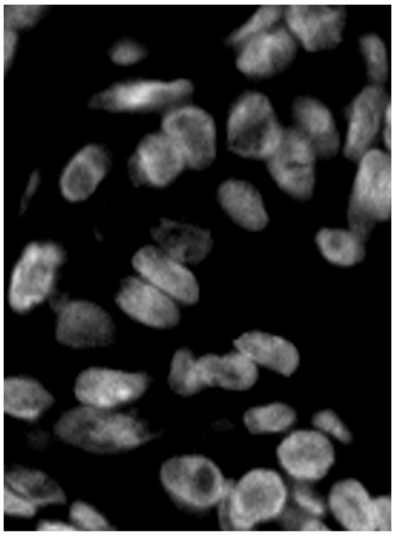


Figure 188 – Result of Unsharp filter engine

5.4. Fusing

Fuse events

Figure 189 – Fuse Events

Where it can be found:

Basic Operations Module ► Fuse ► Fuse events

Description:

Fuse events is a fuse operation available in Basic Operations Module.

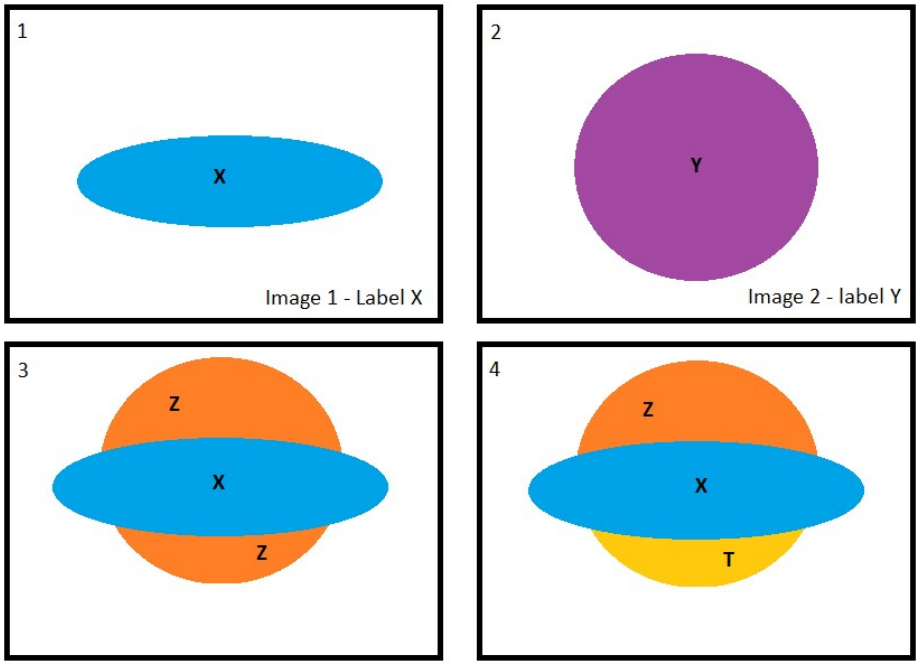
This operation generates the union of 2 sets of events presented in 2 different coded images and places the result in the output.

$$Result = Set\ of\ events\ A \cup Set\ of\ events\ B$$

where:

- Set of events A - events present in Image 1 (e.g. large nuclei detected using settings #1)
- Set of events B - events present in Image 2 (e.g. small nuclei detected using settings #2)

Events from Image 1 have priority over events from Image 2. In the output image, in case of overlapping, the events from Image 1 will be on top of the events from Image 2.



To avoid duplicate labels, the events from Image 2 will be relabeled starting with the maximum label value from Image 1 + 1. Figure 123124 -3 shows the fuse of event labeled “X” from Image 1 and the event labeled “Y” from Image 2. The new label assigned to the event from Image 2 is “Z”, where:

$$Z \geq \max(Image\ 1) + 1$$

Even so, it may happen in cases of partially overlapping events with different sizes that events from Image 2 are split in two or more disjunct bodies with the same label “Z”. This can be corrected by using the “Re-Relabel split labels” operation.

Parameters:

- 1.1 Image 1 (on top) - the first input coded image representing the 1st set of events. These events have priority over the 2nd set of events, meaning they will be on top in case of overlapping;
- 1.2 Image 2 - the second input coded image representing the 2nd set of events. These events do not have priority over the 1st set of events, meaning they will be covered by the 1st set of events in case of overlapping.
- 1.3 Result of “…” - the result of the operation

Effect:

Combines 2 sets of events.

Example:

- Input:

Two detections were run on the same sample using different settings, to find the small and the large nuclei.

Image 1 is a coded image representing 1st set of events (large nuclei). Figure below shows these events on a fluorescence colorectal cancer sample overlay.

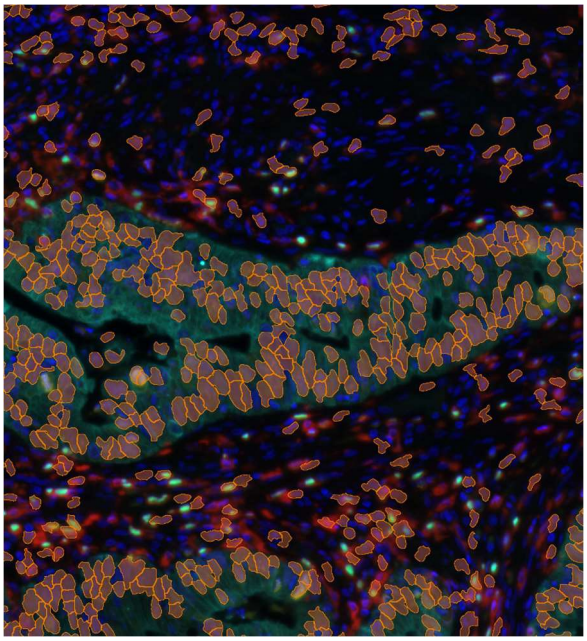


Figure 190 – Colorectal cancer sample - Large nuclei

Image 2 is a coded image representing the 2nd set of events (small nuclei). Figure below shows these events on the same fluorescence colorectal cancer sample overlay.

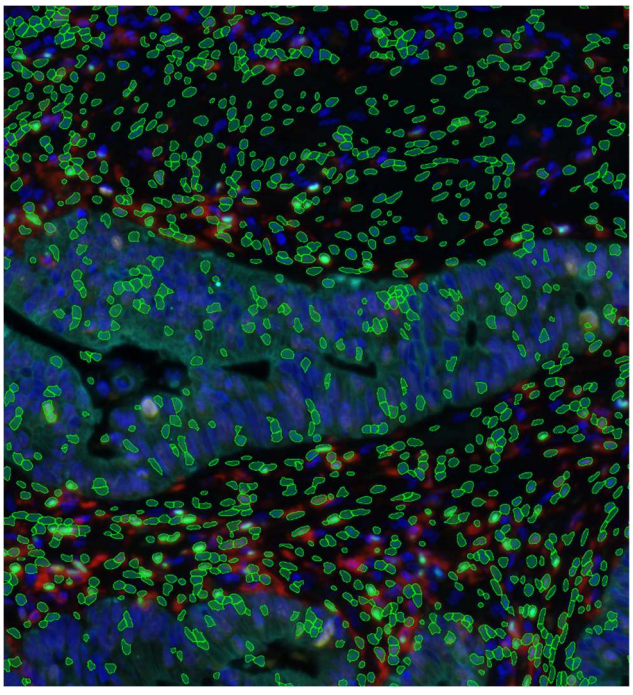


Figure 191 – Colorectal cancer sample - Small nuclei

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a coded image (events), matching the input.

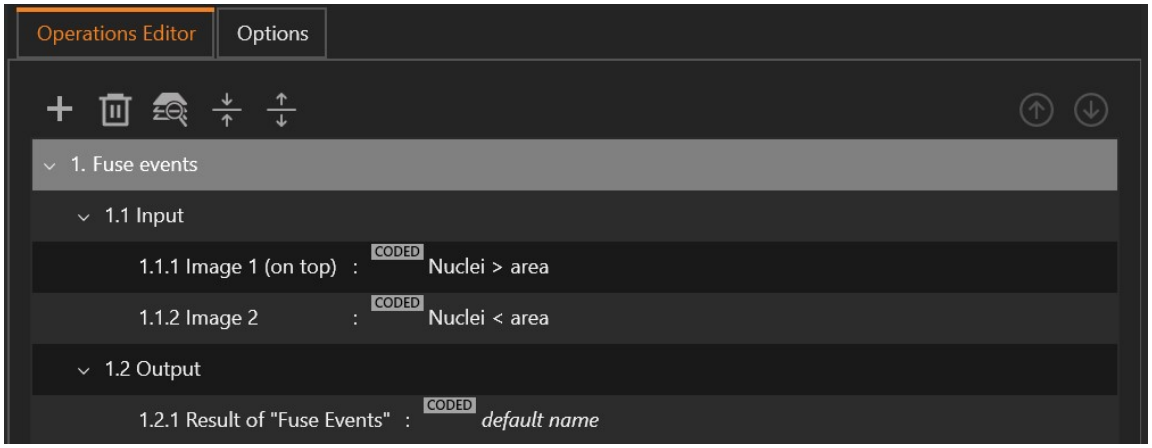


Figure 192 – Engine settings

- Output:

Figure below shows the fusion result of the 2 sets of events. Events coming from set 1 are highlighted in green while events coming from set 2 are highlighted in orange.

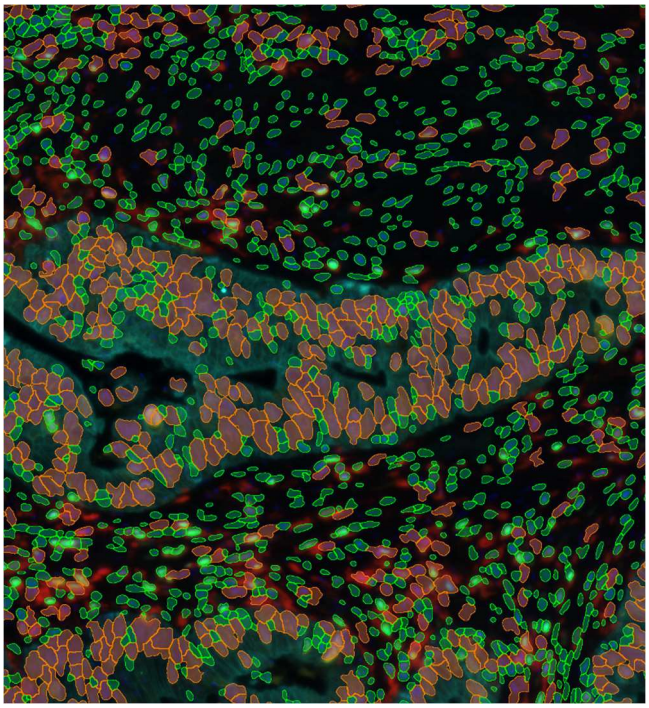


Figure 193 – Result of Fuse events engine

Fuse events (with mask)

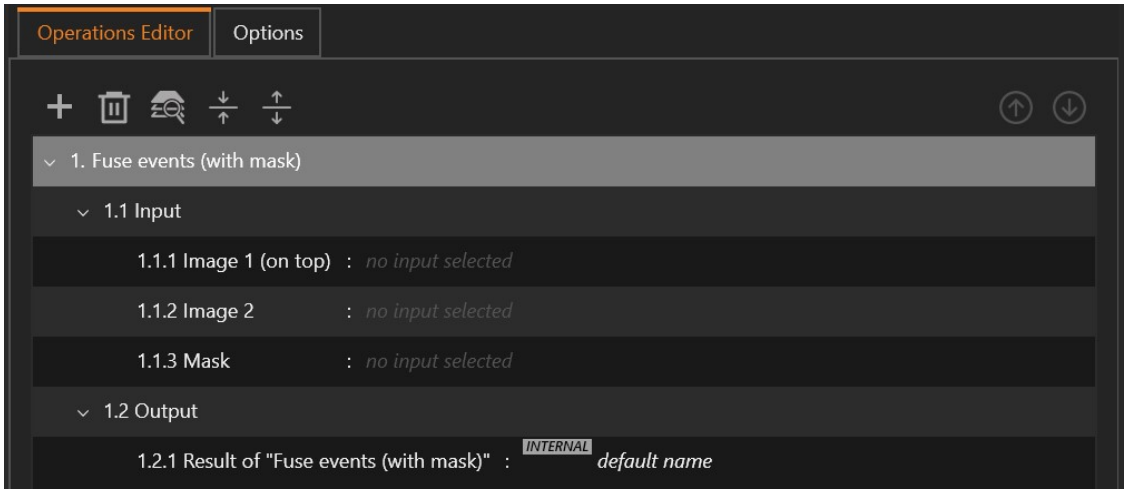


Figure 194 – Fuse Events with Mask

Where it can be found:

Basic Operations Module ► Fuse ► Fuse events (with mask)

Description:

Fuse events (with mask) is a fuse operation available in Basic Operations Module.

This operation generates the masked union of 2 sets of events presented in 2 different coded images and places the result in the output.

$$Result = (Set\ of\ events\ A \cup Set\ of\ events\ B) * Mask$$

where:

- Set of events A: - events present in Image 1 (e.g. large nuclei detected using settings #1)
- Set of events B: - events present in Image 2 (e.g. small nuclei detected using settings #2)
- Mask: - validation mask

This operation is similar to Fuse events. The difference is a validation mask applied to the fuse result. Only the pixels indicated by the mask (white) are kept.

The usage of “Re-Label split labels” is even more recommended after this operation than after Fuse events, because the validation mask can also split fused labels into disjoint bodies with the same label.

Parameters:

- 1.1 Image 1 (on top) - the first input coded image representing the 1st set of events. These events have priority over the 2nd set of events, meaning they will be on top in case of overlapping;
- 1.2 Image 2 - the second input coded image representing the 2nd set of events. These events don’t have priority over the 1st set of events, meaning they will be covered by the 1st set of events in case of overlapping.
- 1.3 Mask - the validation mask. Only the results pixels indicated by the mask (white) are kept, the rest are set on 0.
- 1.4 Result of “…” - the result of the operation

Effect:

Combines 2 sets of events in accordance with a validation mask.

Example:

- Input:

Two detections were run on the same sample using different settings, to find the small and the large nuclei.

Image 1 is a coded image representing the 1st set of events (large nuclei). Figure below shows these events on a fluorescence colorectal cancer sample overlay.

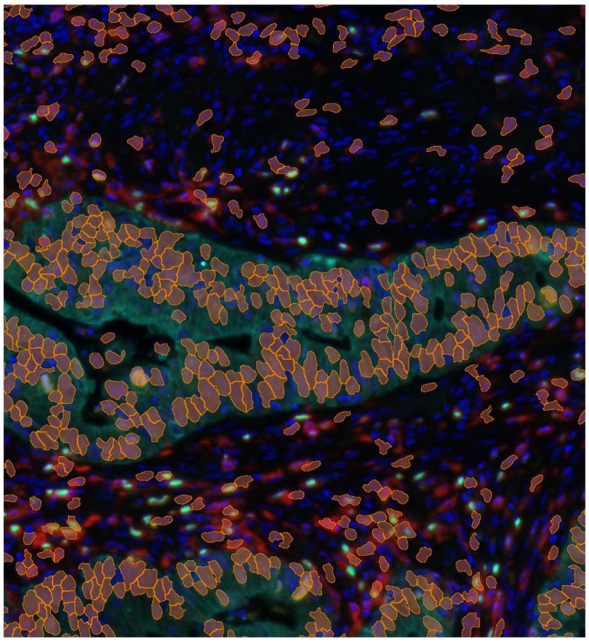


Figure 195 – Colorectal cancer sample - Large nuclei

The image below is a coded image representing the 2nd set of events (small nuclei). Figure shows these events on the same fluorescence colorectal cancer sample overlay.

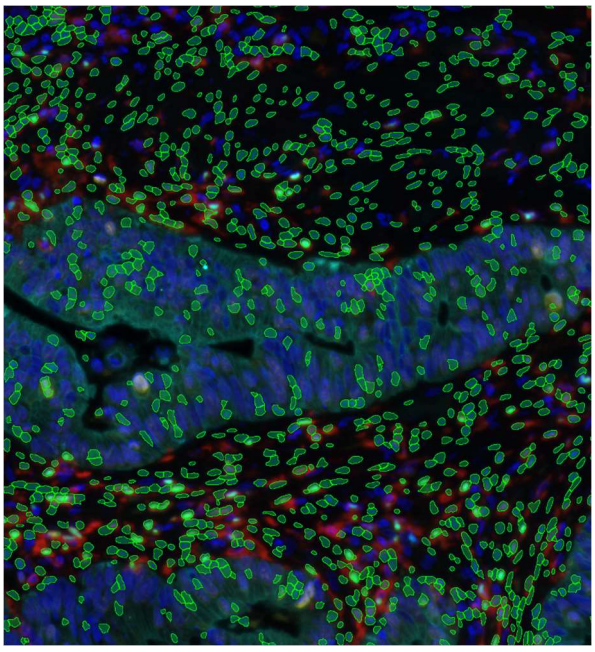


Figure 196 – Colorectal cancer sample - Small nuclei

Image 3 is a mask image (8-bit, 1-channel), representing the valid (white) areas were events fusion is performed.

Figure 197 – Mask image

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a coded image (events), matching the input.

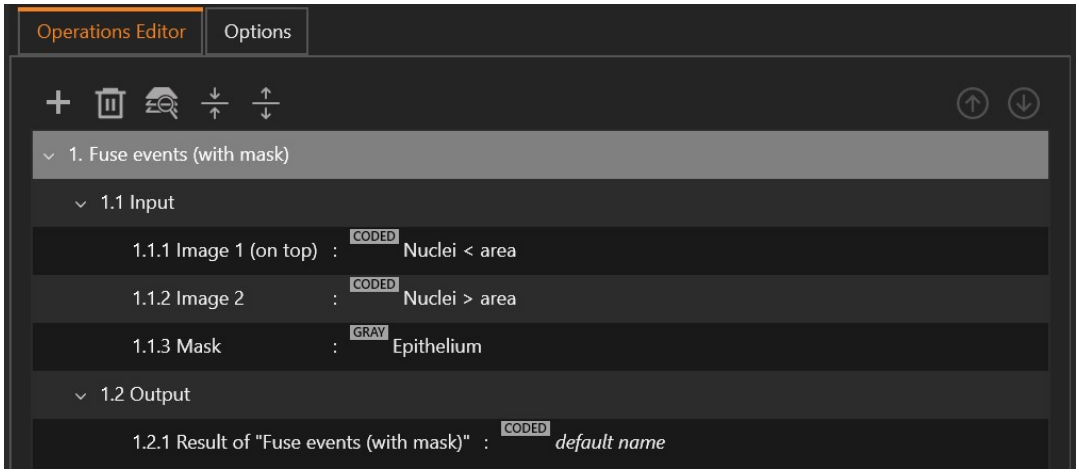


Figure 198 – Engine settings

- Output:

Figure below shows the fusion result of the 2 sets of events. Events coming from set 1 are highlighted in green while events coming from set 2 are highlighted in orange.

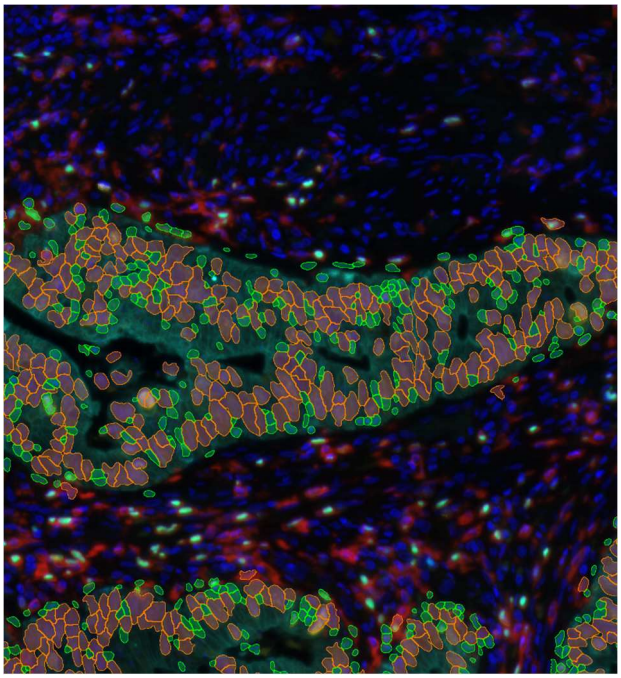


Figure 199 – Result of Fuse events with mask engine

5.5. Image Data Exchange and Initialization

Convert 16-bit to 8-bit

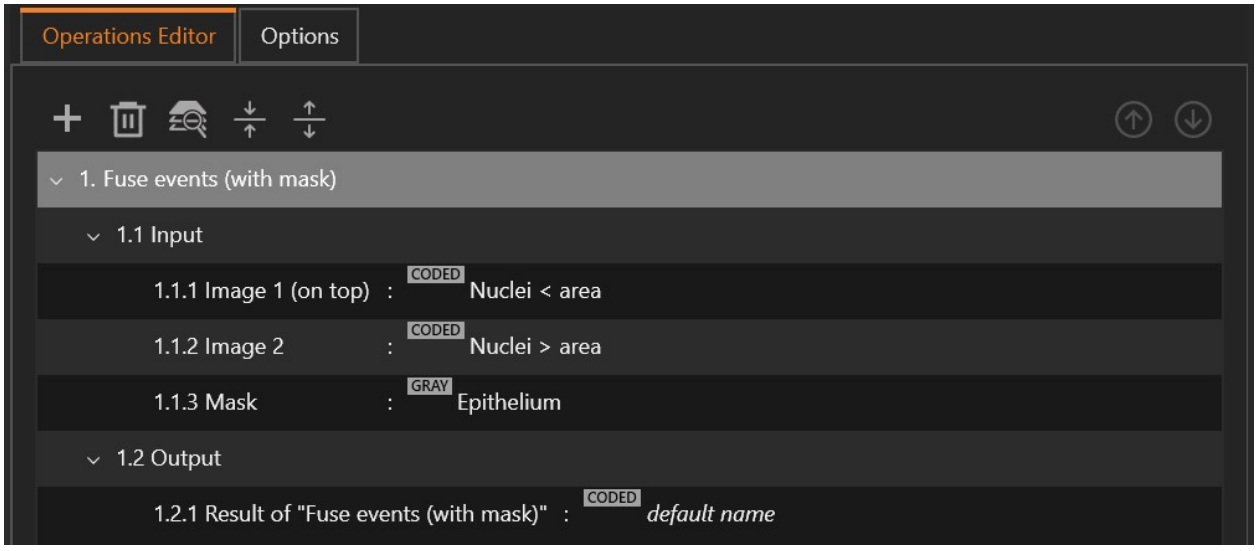


Figure 200 – Convert 16 bit to 8 bit

Where it can be found:

Basic Operations Module ► Image Data ► Convert 16-bit to 8-bit

Description:

Convert 16-bit to 8-bit is an image data operation available in Basic Operations Module.

This operation scales the pixel values of the grayscale 16-bit image and converts them to 8-bit. The result is placed in the output.

Most of the times, only a part of the entire 16-bit range (0 - 65.535) is used to represent the information. When the conversion to 8-bit (0 - 255) is made, it is important to consider only the relevant intensity interval in order to better scaling of the information. The interval is defined by the Lower Limit (minimum non-zero intensity value) and the Upper Limit (maximum intensity value).

Parameters:

- 1.1 Image - the input image
- 1.2 Lower limit - the lowest intensity of the information. This can be associated with the darkest area in the image
- 1.3 Upper limit - the highest intensity of the information. This can be associated with the brightest area in the image; except undesired overexposed areas
- 1.4 Result of "…" - the result of the operation

Effect:

Converts a 16-bit grayscale image to 8-bit.

Example:

- Input:

The image is fluorescence grayscale image (16-bit, 1-channel) representing the DAPI channel of a colorectal cancer sample. Figure shows the 8-bit converted image (because it's not possible to display the 16-bit image) and the 16-bit histogram of the input image is displayed in the top-left corner.

Note	Monitors do not have the capacity to display 16-bit images, thus images must be converted to 8-bit for display purposes.

- Engine settings:

Figure below shows the engine settings used for this example.

In this example, the lower limit and upper limit values are set using the histogram information.

The engine's result is an 8-bit grayscale image.

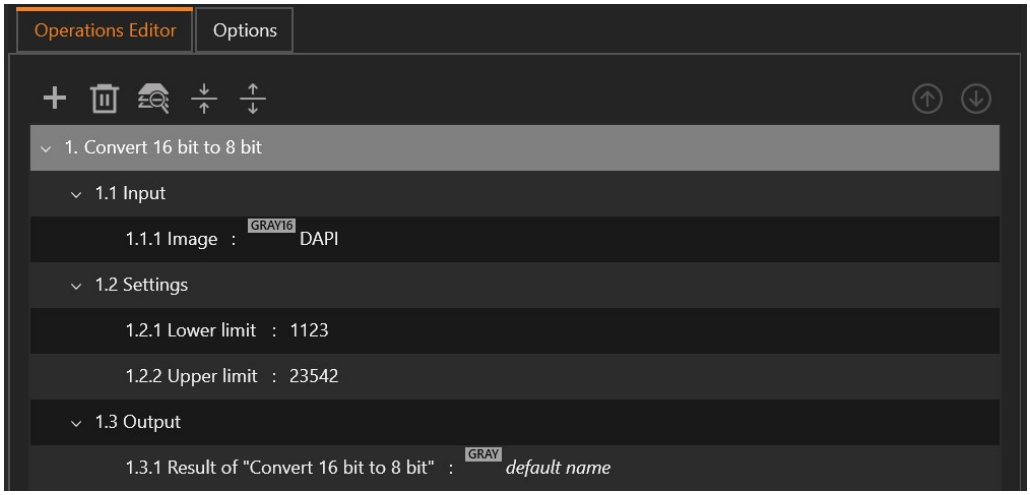


Figure 201 – Engine settings

- Output:

Figure below shows the result of 16-bit to 8-bit conversion, of the input image.

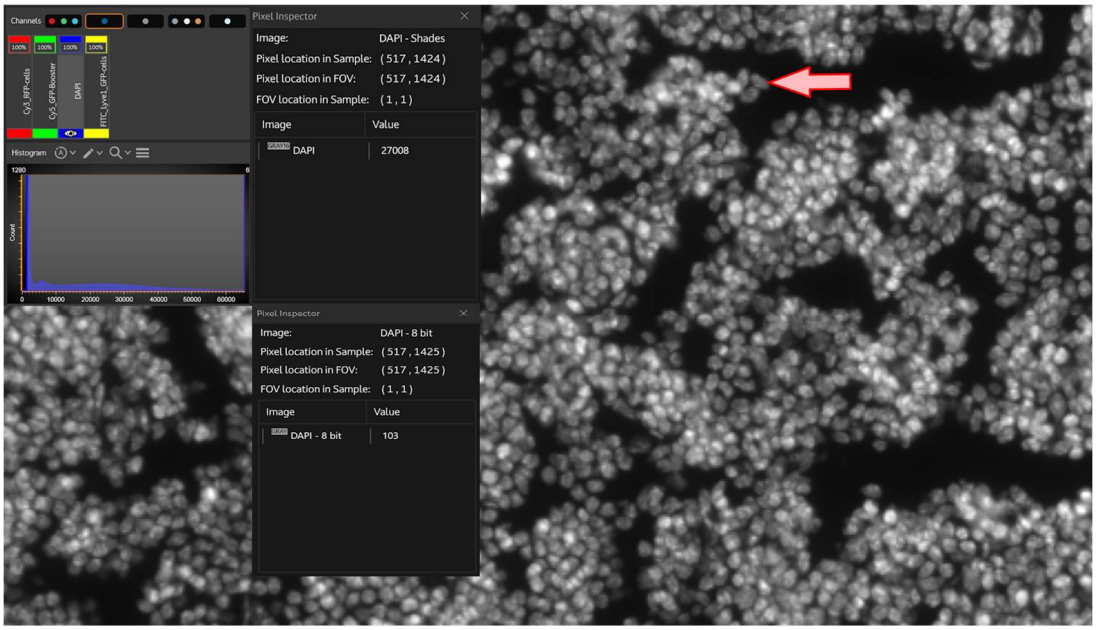


Figure 202 – Result of Convert 16-bit to 8-bit engine

Extract channel

Figure 203 – Extract channel

Where it can be found:

Basic Operations Module ► Image Data ► Extract channel

Description:

Extract channel is an image data operation available in Basic Operations Module.

This operation extracts the information present in a specified channel from a 3-channel image and places the result in the output.

Parameters:

- 1.1 Image - the input image
- 1.2 Channel index - specifies the index of the channel to be extracted (default = 1)
- 1.3 Result of "…" - the result of the operation

Effect:

Extracts a specified channel from a 3-channel image.

Example:

- Input:

The image below is a brightfield color image (8-bit, 3-channel) representing a small region from a colon sample.

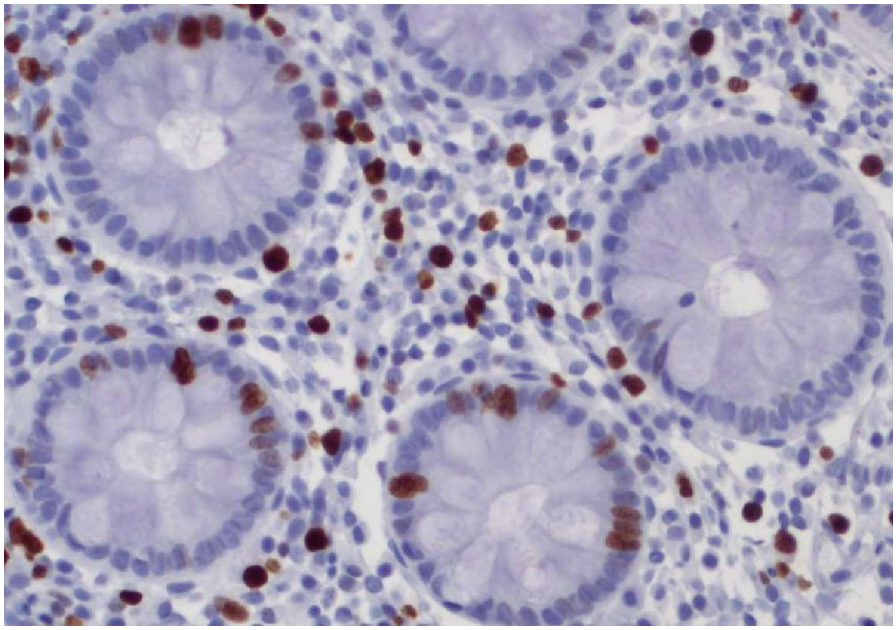


Figure 204 – Colon sample

- Engine settings:

Figure below shows the engine settings used for this example.

The green channel (2nd channel) is extracted.

The engine's result is a coded image (events), matching the input.

Figure 205 – Engine settings

- Output:

Figure below shows extracted green channel from the input image.

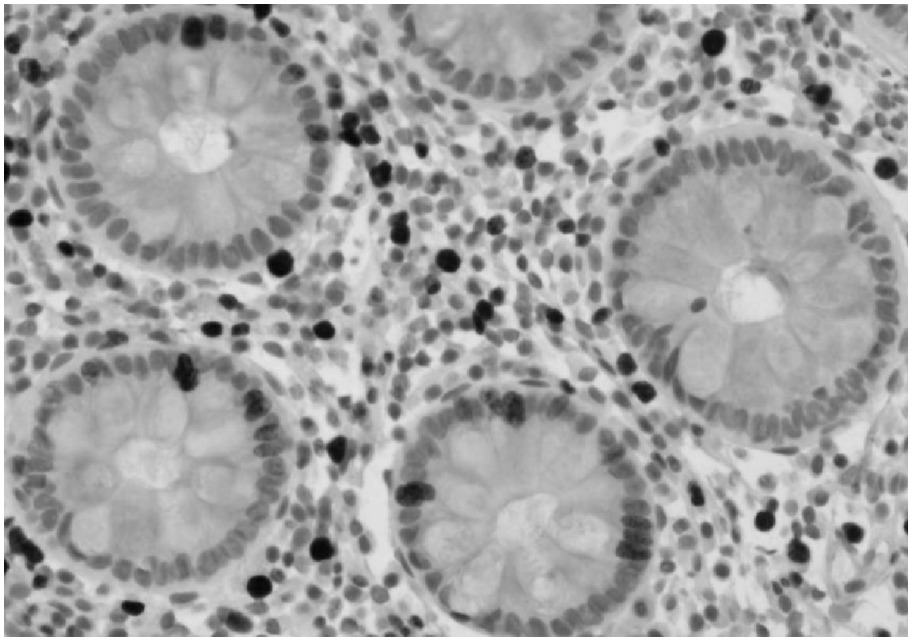


Figure 206 – Result of Extract channel engine

Extract strongest channel

Figure 207 – Extract Strongest Channel

Where it can be found:

Basic Operations Module ► Image Data ► Extract strongest channel

Description:

Extract strongest channel is an image data operation available in Basic Operations Module.

This operation extracts the information from the strongest intensity-wise channel present in a 3-channel image and places the result in the output (the color-space does not matter).

The selection of the strongest channel is based on the average value of intensities for each channel.

Parameters:

- 1.1 Image - the input image
- 1.2 Result of "…" - the result of the operation

Effect:

Generates a grayscale image from an RGB image by extracting the strongest channel.

Example:

- Input:

The image below is a brightfield color image (8-bit, 3-channel) representing a small region from a colon sample.

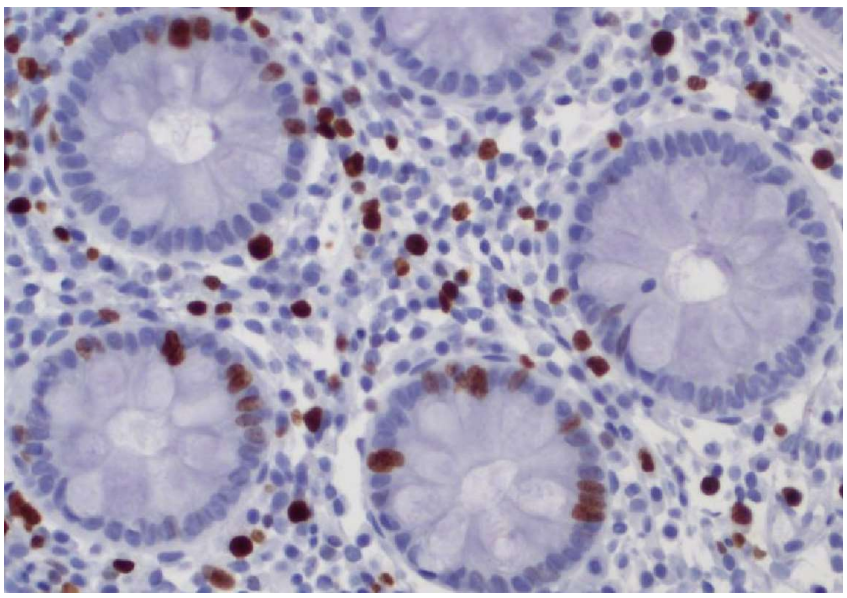


Figure 208 – Colon sample

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a coded image (events), matching the input.

Figure 209 – Engine settings

- Output:

Figure below shows the extracted channel, the one with the highest brightness intensity.

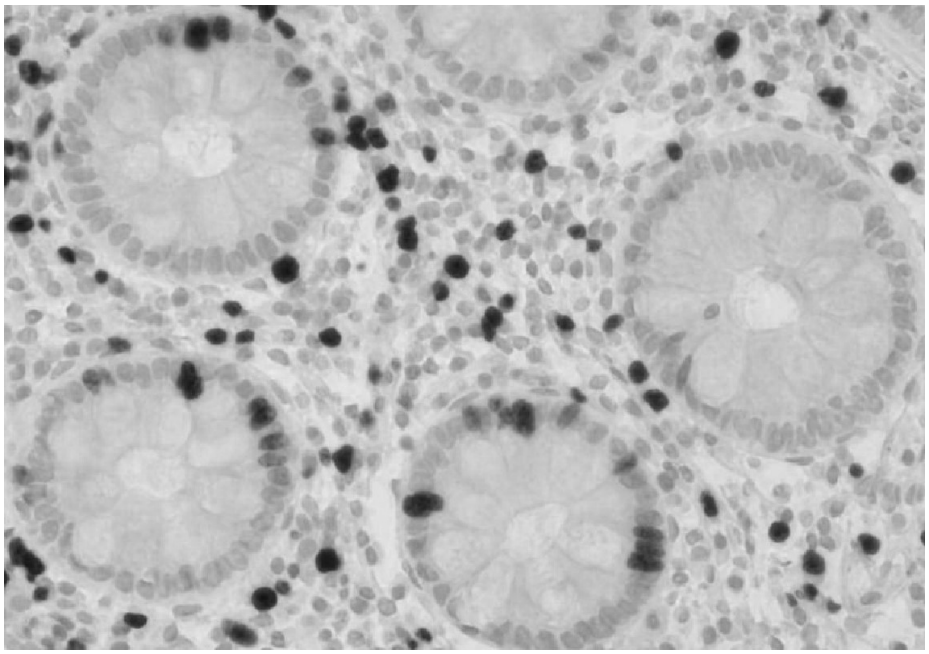


Figure 210 – Result of Extract strongest channel engine

Mask

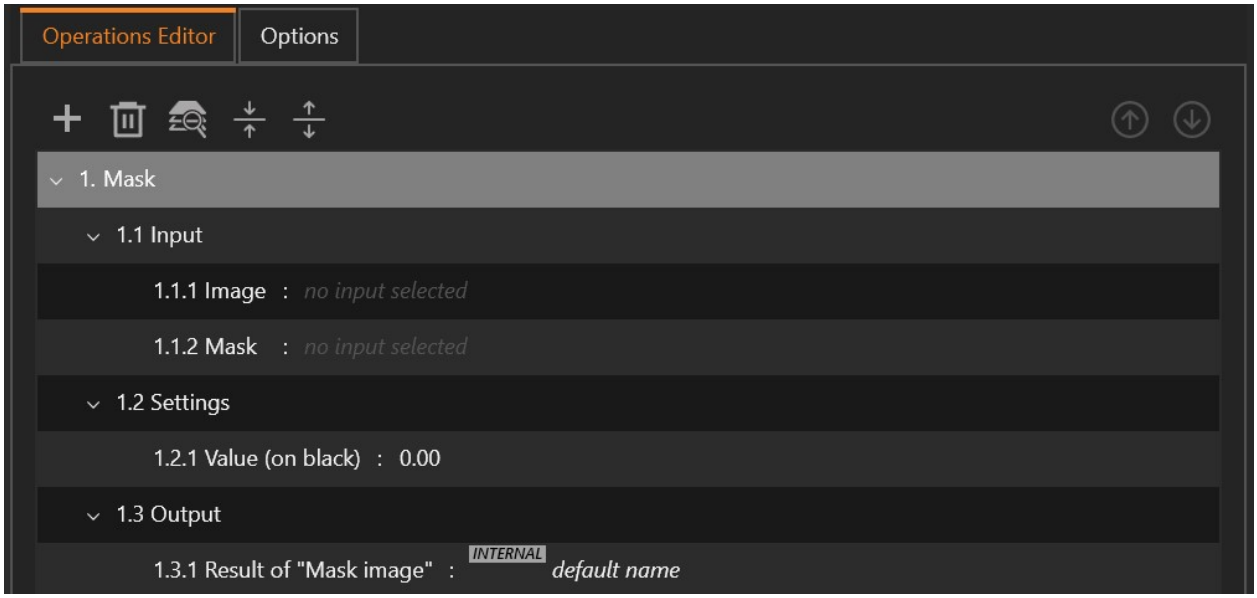


Figure 211 – Mask

Where it can be found:

Basic Operations Module ► Image Data ► Mask

Description:

Mask is an image data operation available in Basic Operations Module.

This operation masks the information present in the image and places the result in the output.

Image pixels indicated by the mask (white) are preserved while the other ones are set to the new specified value. With a binary mask (values 0 = black and 1 = white), the operation can be expressed by the formula:

$$Result = Image .* Mask + value$$

where ".*" denotes pixel-wise multiplication between Image and Mask.

Parameters:

- 1.1 Image - the input image
- 1.2 Mask - the input mask
- 1.3 Value (on black) - the new value for the pixels indicated by mask (black) (default = 0)

- 1.4 Result of "…" - the result of the operation

Effect:

Masks the information present in an image.

Example:

- Input:

The image is a brightfield color image (8-bit, 3-channel) representing a small region from a colon sample.

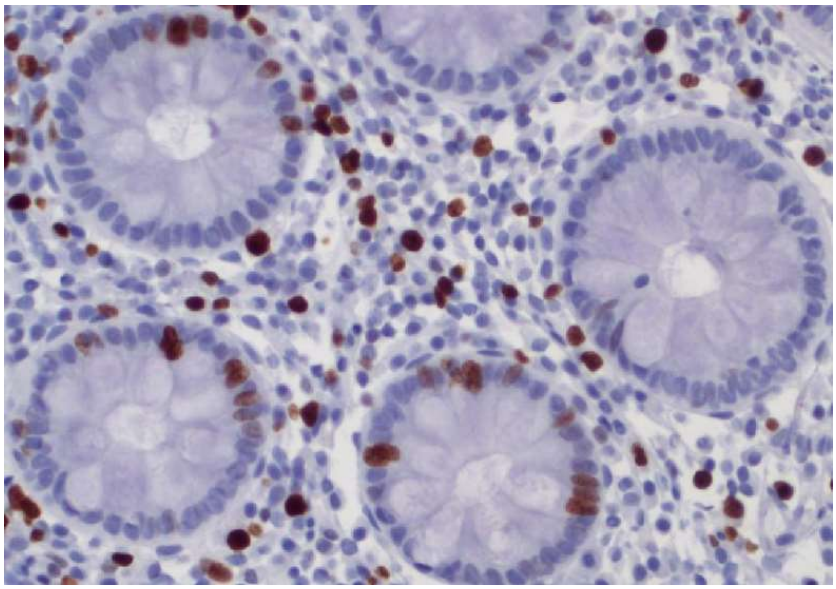


Figure 212 – Colon sample

The mask is a binary image (black and white) which indicates what pixel values are preserved (indicated by white).

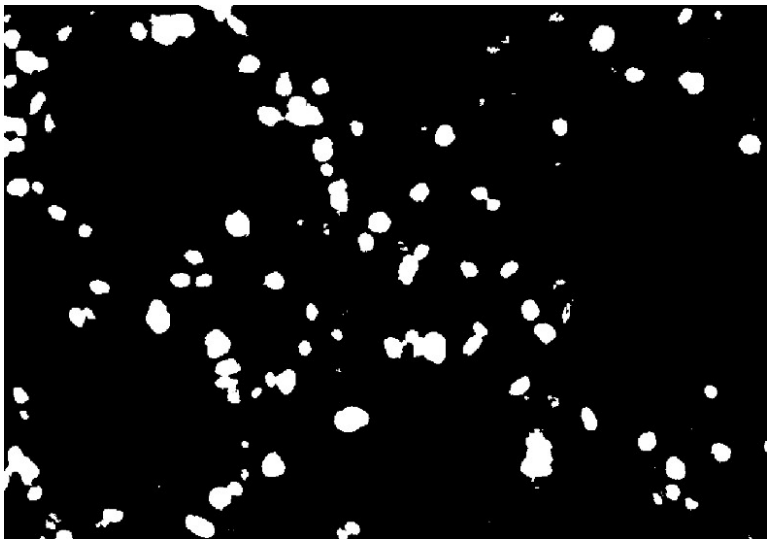


Figure 213 – Mask

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a color image (8-bit, 3-channels), matching the input.

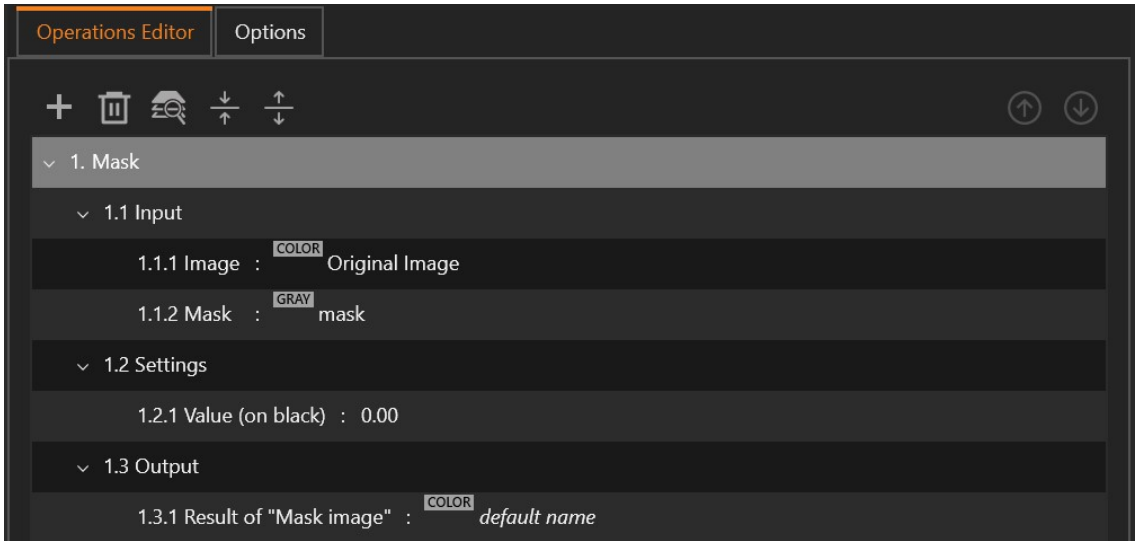


Figure 214 – Engine settings

- Output:

Figure below shows masked input image. Only pixels indicated by the white area of the mask preserved their color. The rest of the pixels (indicated by the black area of the mask) where set to the specified value (0).

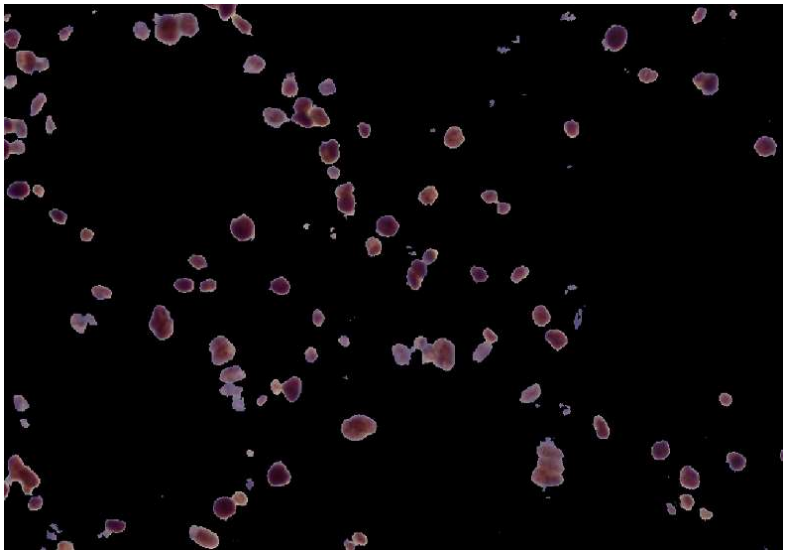


Figure 215 – Result of Mask engine

Set channels to RGB

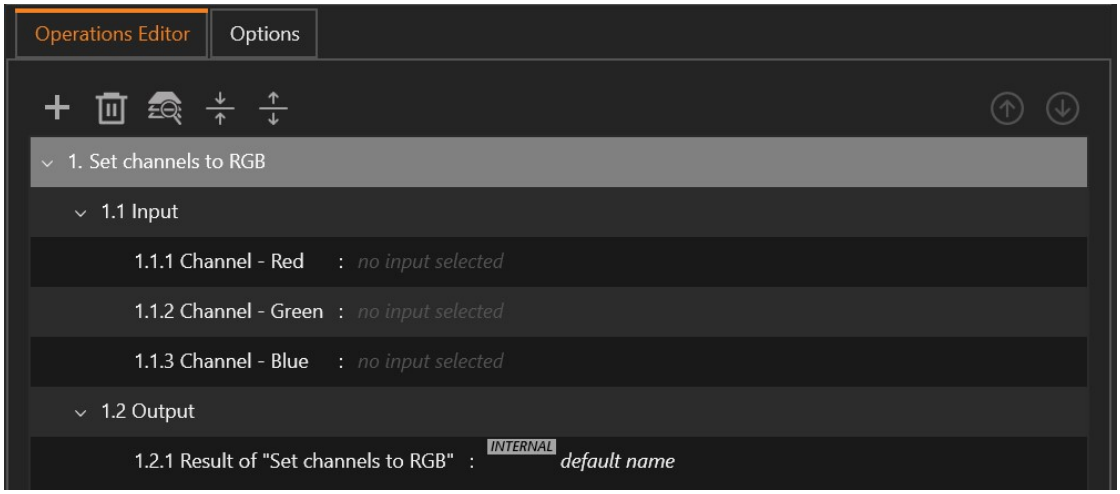


Figure 216 – Copy channels to color

Where it can be found:

Basic Operations Module ► Image Data ► Set channels to RGB

Description:

Set channels to RGB is an image data operation available in Basic Operations Module.

This operation copies three grayscale images to the specified channels of an RGB image and places the result in the output.

Parameters:

- 1.1 Channel - Red - the grayscale input image representing the red channel of the result
- 1.2 Channel - Green - the grayscale input image representing the green channel of the result
- 1.3 Channel - Blue - the grayscale input image representing the blue channel of the result

Effect:

Combines 3 grayscale images into an RGB image.

Example:

- Input:

The images are brightfield grayscale images (8-bit, 1-channel), representing the color channels of a small region of a colon sample.

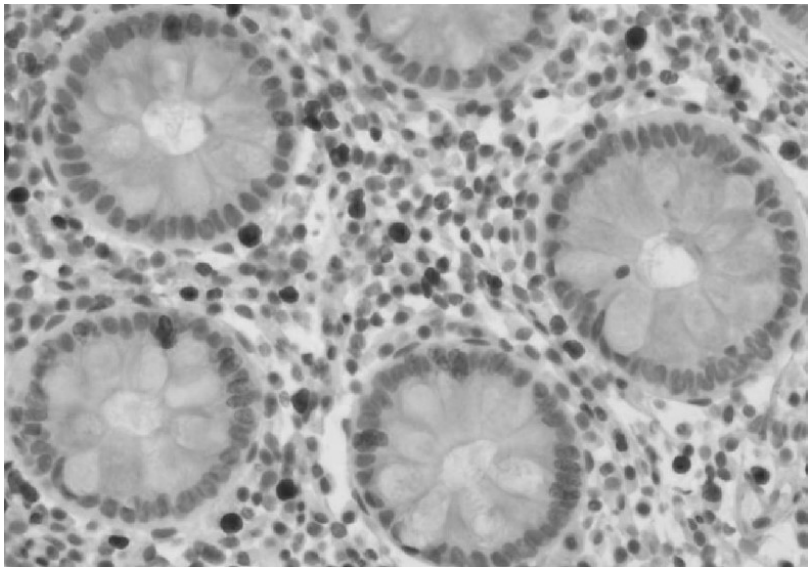


Figure 217 – Colon sample (red channel)

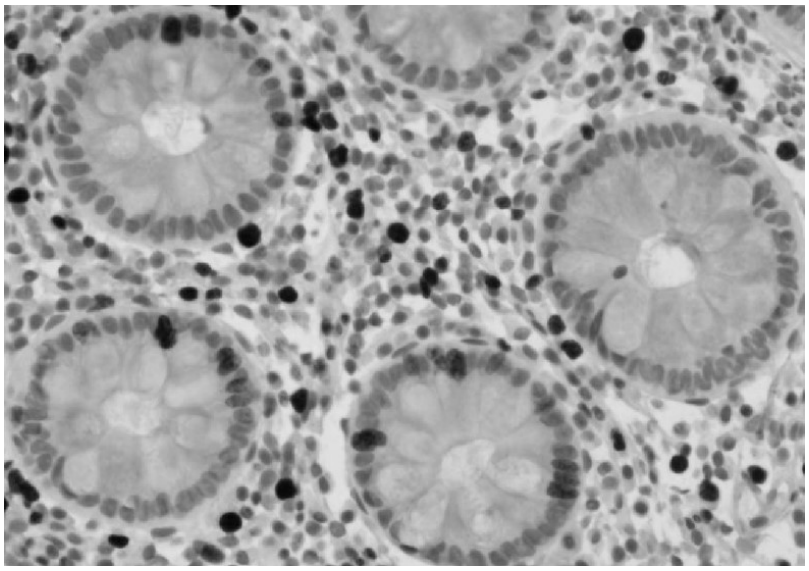


Figure 218 – Colon sample (green channel)

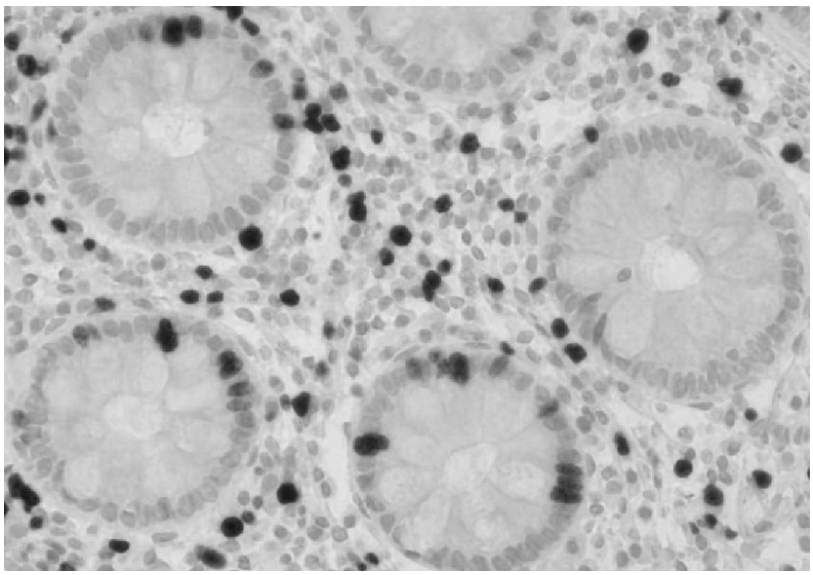


Figure 219 – Colon sample (blue channel)

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a color image (8-bit, 3-channels).

Figure 220 – Engine settings

- Output:

Figure below shows the result of 16-bit to 8-bit conversion, of the input image.

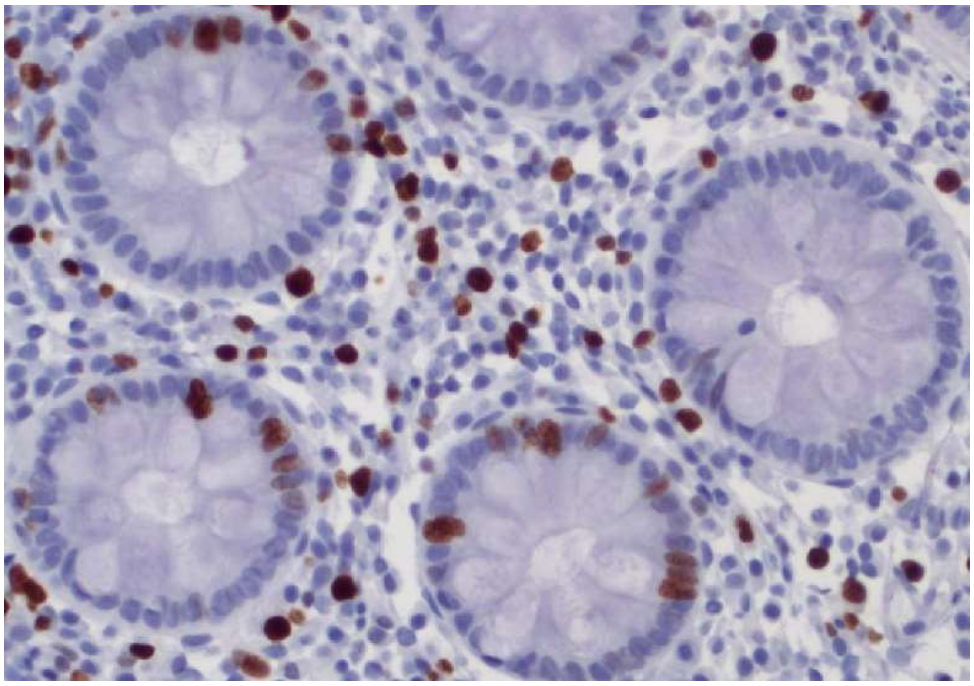


Figure 221 – Result of Set channels to RGB engine

Swap channels

Figure 222 – Swap channels

Where it can be found:

Basic Operations Module ► Image Data ► Swap channels

Description:

Swap channels is an image data operation available in Basic Operations Module.

This operation changes the order of the channels in a 3-channel image and places the result in the output.

Parameters:

- 1.1 Image - the input image
- 1.2 Channel 1 - the index of the input image channel to be copied to channel 1
- 1.3 Channel 2 - the index of the input image channel to be copied to channel 2
- 1.4 Channel 3 - the index of the input image channel to be copied to channel 3
- 1.5 Result of "…" - the result of the operation

Effect:

Changes the order of channels in a 3-channel image.

Example:

- Input:

The image is a brightfield color image (8-bit, 3-channel) representing a small region from a colon sample.

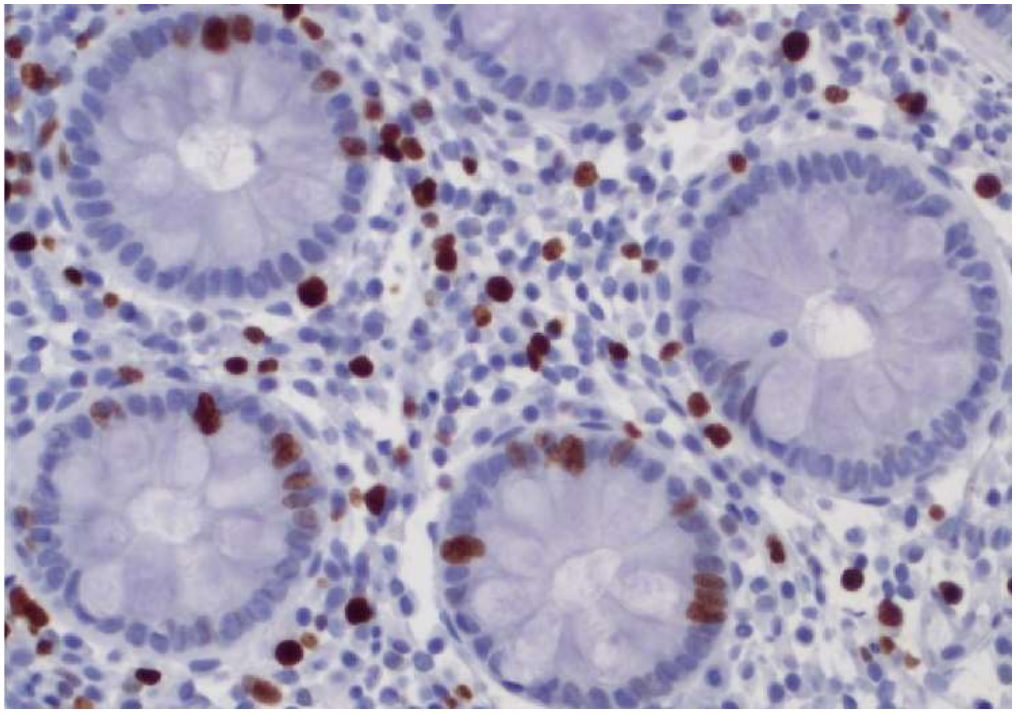


Figure 223 – Original

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a color image (8-bit, 3-channels), matching the input.

Figure 224 – Engine settings

- Output:

Figure below shows the result of input image channel swap (red and blue).

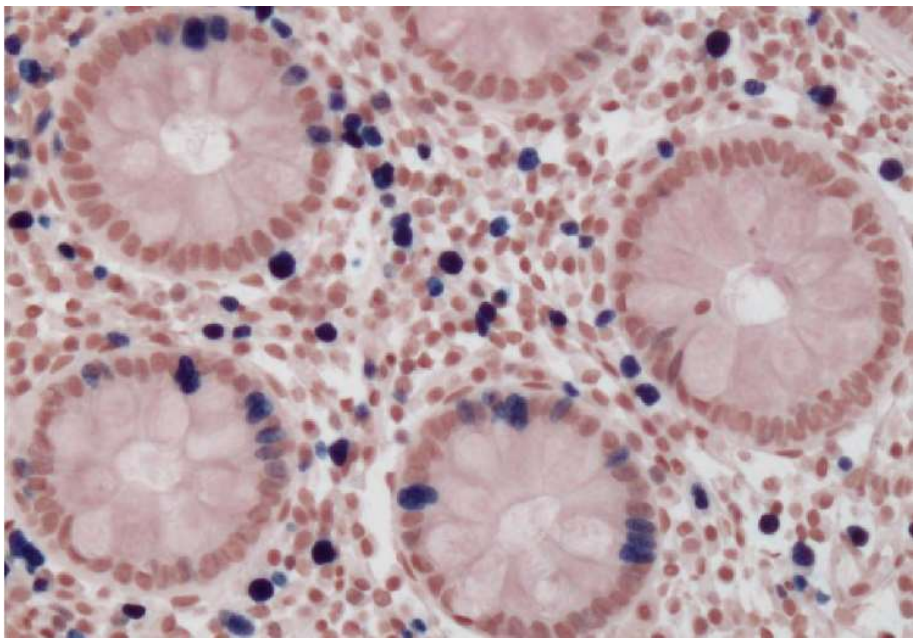


Figure 225 – Results of Swap channels engine

5.6. Label

Cut bottlenecks

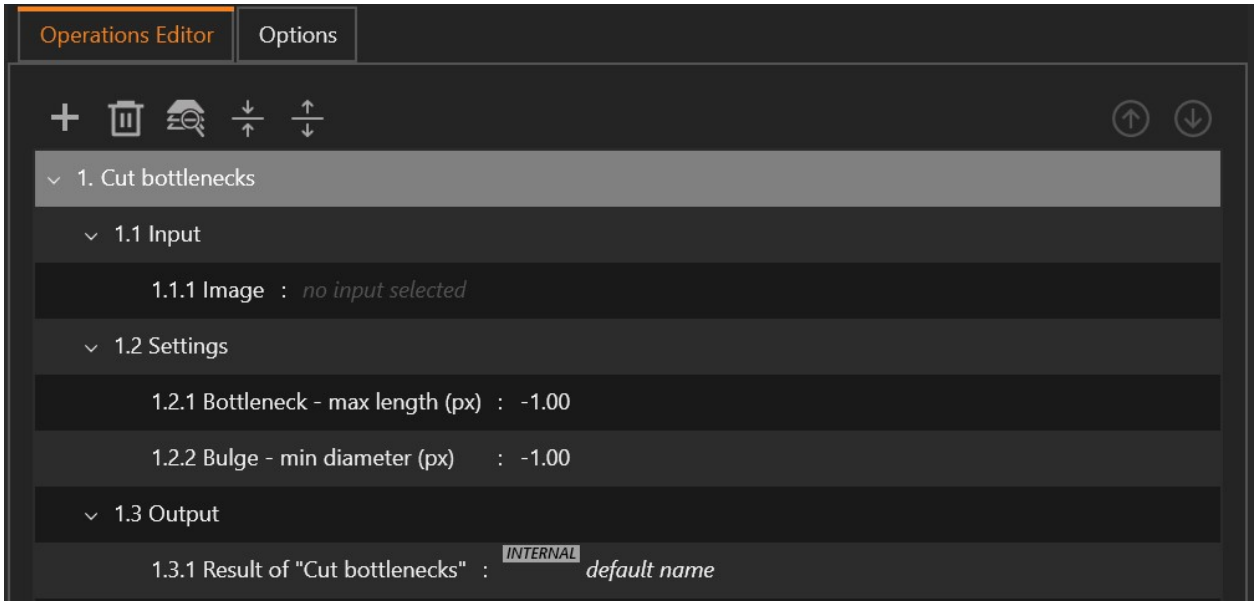


Figure 226 – Cut Bottlenecks

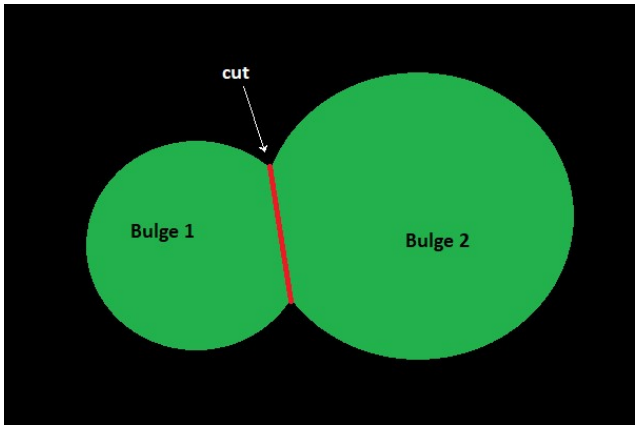
Where it can be found:

Basic Operations Module ► Labels ► Cut bottlenecks

Description:

Cut bottlenecks is a label operation available in Basic Operations Module.

This operation separates distinct structures detected as a single event and connected by a thin area resembling a bottleneck. The result is placed in the output.



This operation will not work for events spread across multiple FOVs.

Parameters:

- 1.1 Image - the input image
- 1.2 Bottleneck -max length (px) - the maximum length of the cut needed to separate the two structures (default = -1)
- 1.3 Bulge -min area (px) - the minimum area allowed for the event resulting from the cut (default = -1)
- 1.4 Result of "…" - the result of the operation

Effect:

Corrects detection results by separating structures detected as single events (under-segmentation) and which are merged by a thin area.

Example:

- Input:

The image is a coded image (events) representing the detected structures in a colon sample (figure 3-213, highlighted in orange). The 2 structures were detected as a single event. Additional steps are required to correct the result.

The annotations are present for easily visualize the 2 bulges and the bottleneck.

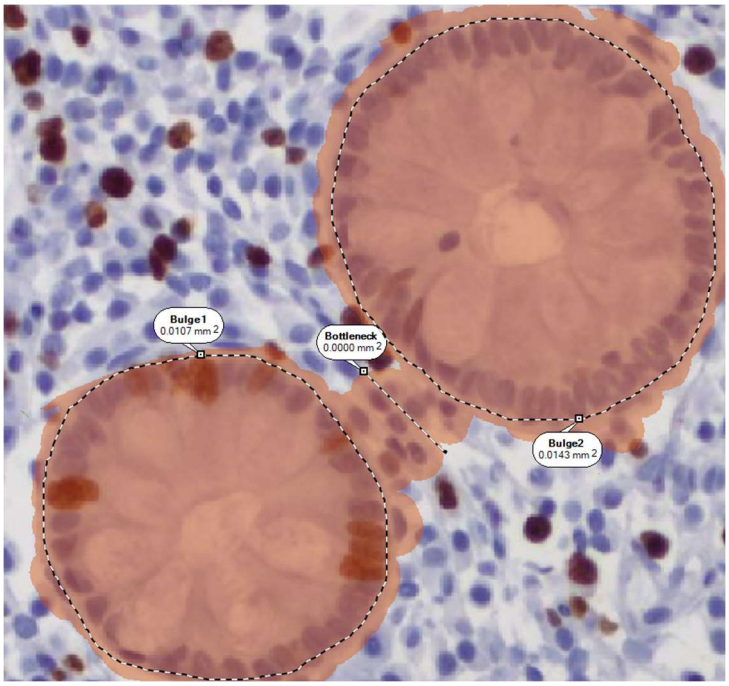


Figure 227 – Detected structures on colon sample

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a coded image (events), matching the input.

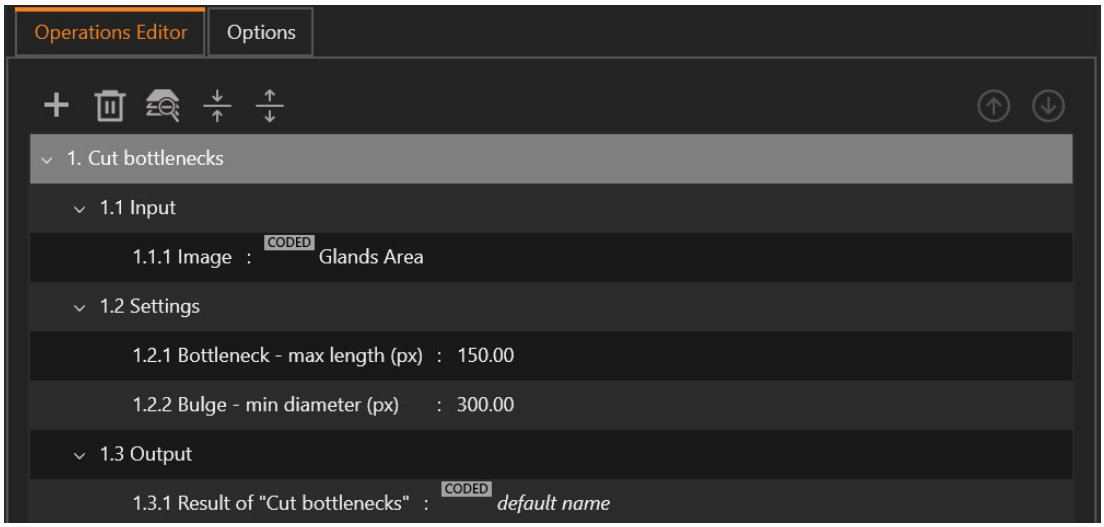


Figure 228 – Engine settings

- Output:

Figure below shows the 2 merged structures split, each new event highlighted with a different color (green and orange).

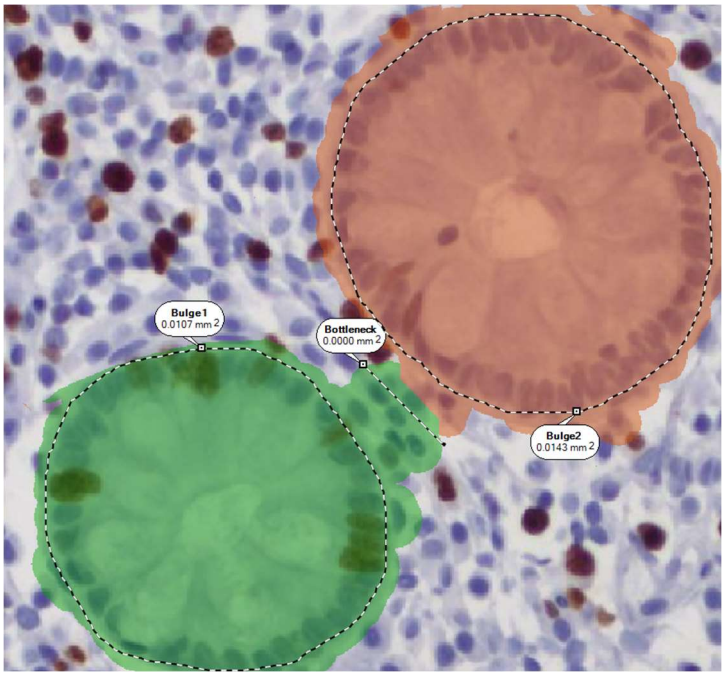


Figure 229 – Result of Cut bottlenecks engine

Fill labels

Figure 230 – Fill Labels

Where it can be found:

Basic Operations Module ► Labels ► Fill Labels

Description:

Fill Labels is a label operation available in Basic Operations Module.

This operation fills holes inside labels (labeled objects) present in image and places the result in the output.

Parameters:

- 1.1 Image - the input image
- 1.2 Fill labels on border - enables / disables filling for holes touching the border of the image
- 1.3 Result of "…" - the result of the operation

Effect:

Fills holes in labels.

Example:

- Input:

Image is a coded image (events) representing the detected structures (interior) in a colon sample (highlighted in green). To improve the detection result, the interior holes need to be filled.

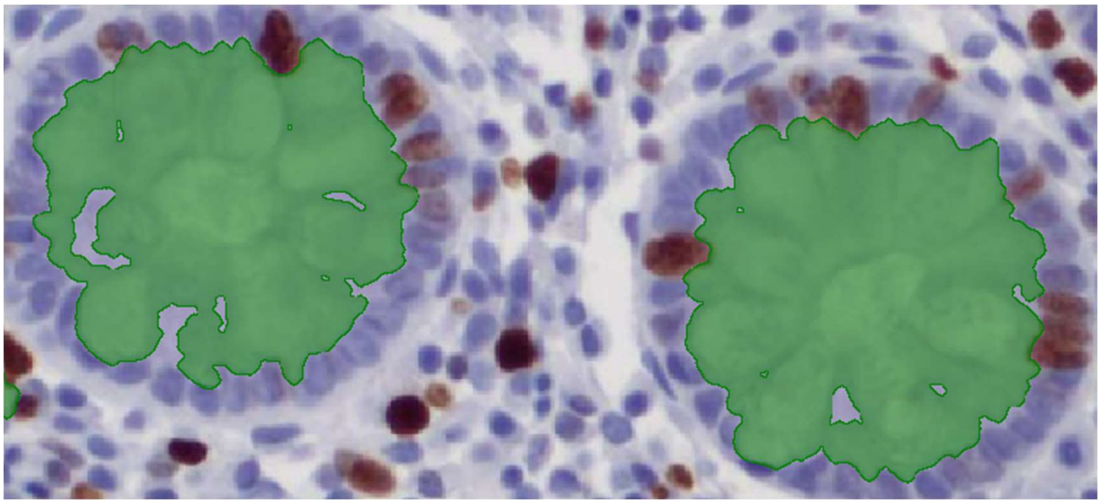


Figure 231 – Colon sample – partially detected structures

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a coded image (events), matching the input.

Figure 232 – Engine settings

- Output:

Figure below shows the improved detection result with the holes filled.

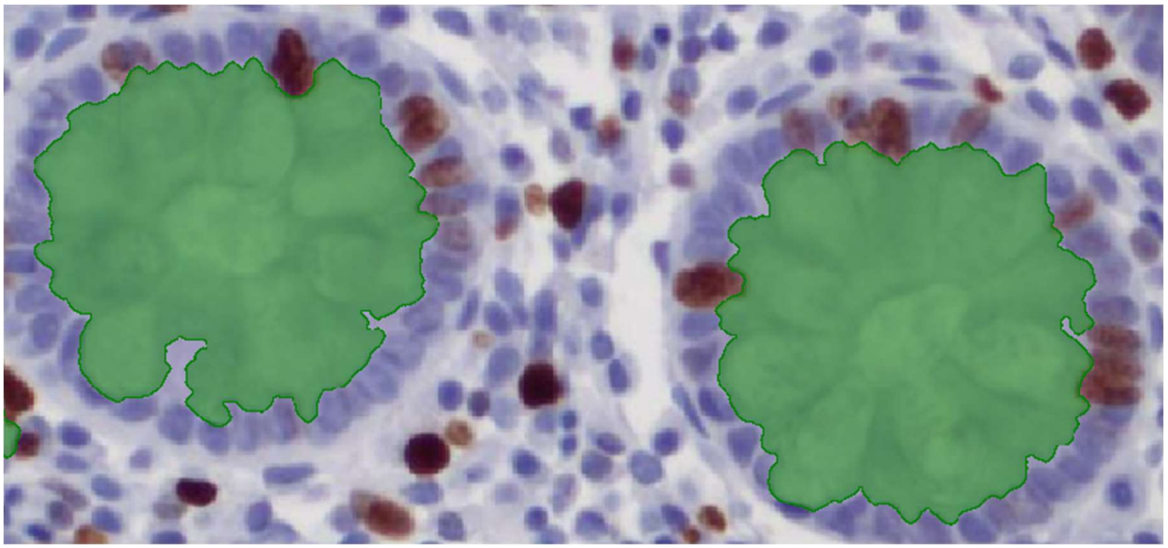


Figure 233 – Result of Fill labels engine

Graph from labels

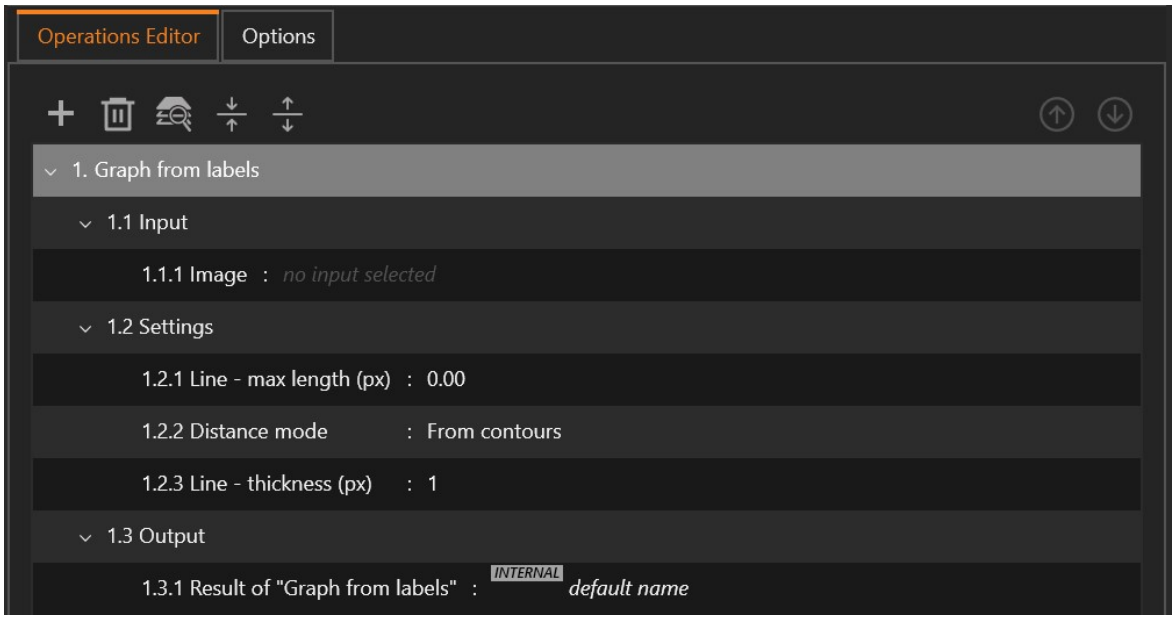


Figure 234 – Label Graph from Coded V3

Where it can be found:

Basic Operations Module ► Labels ► Graph from labels

Description:

Graph from labels is a label operation available in Basic Operations Module.

This operation generates a line-type connection (graph) between each two events closer than a specified distance and places the result in the output.

Parameters:

- 1.1 Image - the input image
- 1.2 Line - max length (px) - the maximum distance allowed between two events to be connected by a (default = 0);
- 1.3 Distance mode - specifies the reference points used to compute the distances: centroids or contours (default = 0)
- 1.4 Line - thickness (px) - represents the generated line thickness (default = 1)
- 1.5 Result of "…" - the result of the operation

Effect:

Generates a graph structure connecting close events.

Example:

- Input:

The image is a coded image (events) representing the detected nuclei in a breast cancer sample (highlighted in green).

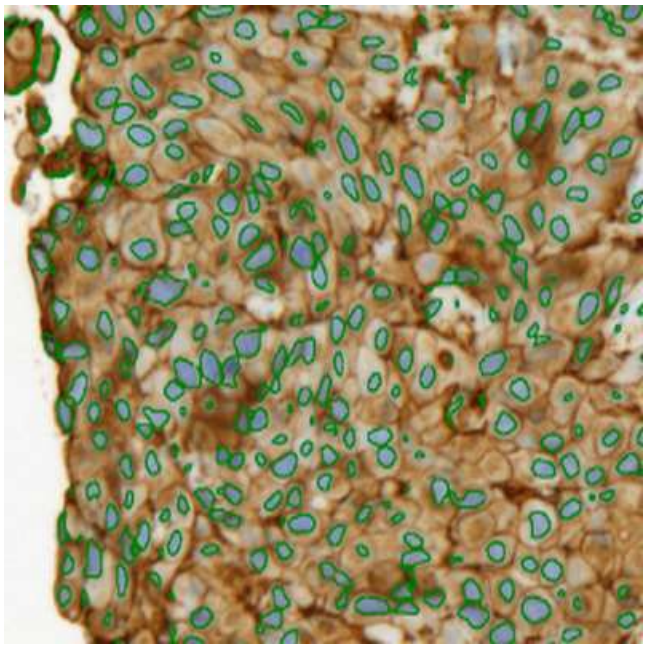


Figure 235 – Breast cancer sample - Detected nuclei

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a mask image (8-bit, 1-channel).

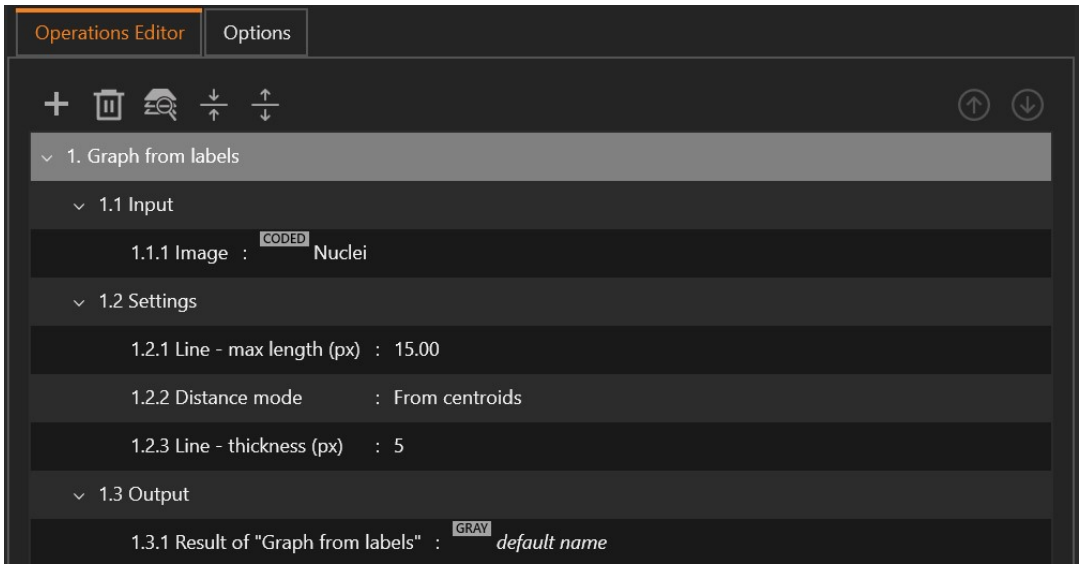


Figure 236 – Engine settings

- Output:

First figure shows the generated graph connecting events closer than the specified distance (15 pixels).

Second figure shows the generated graph highlighted in yellow, overlaid on the original color image and detected events (nuclei) highlighted in green.

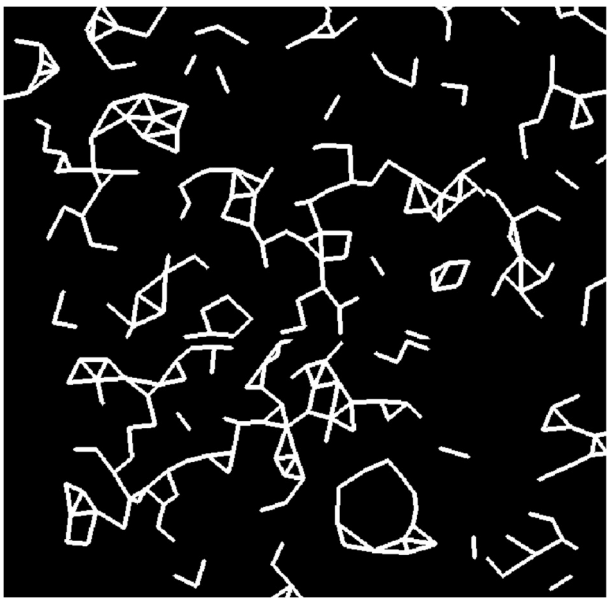


Figure 237 – Result of Graph from labels engine

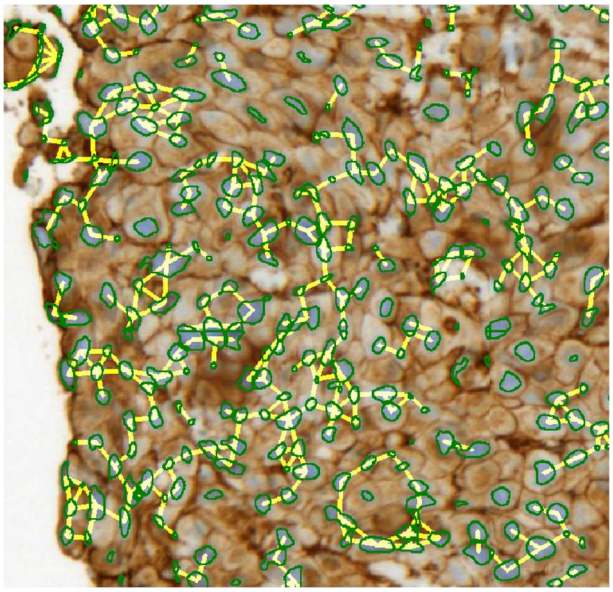


Figure 238 – Result of Graph from labels engine

Grow labels

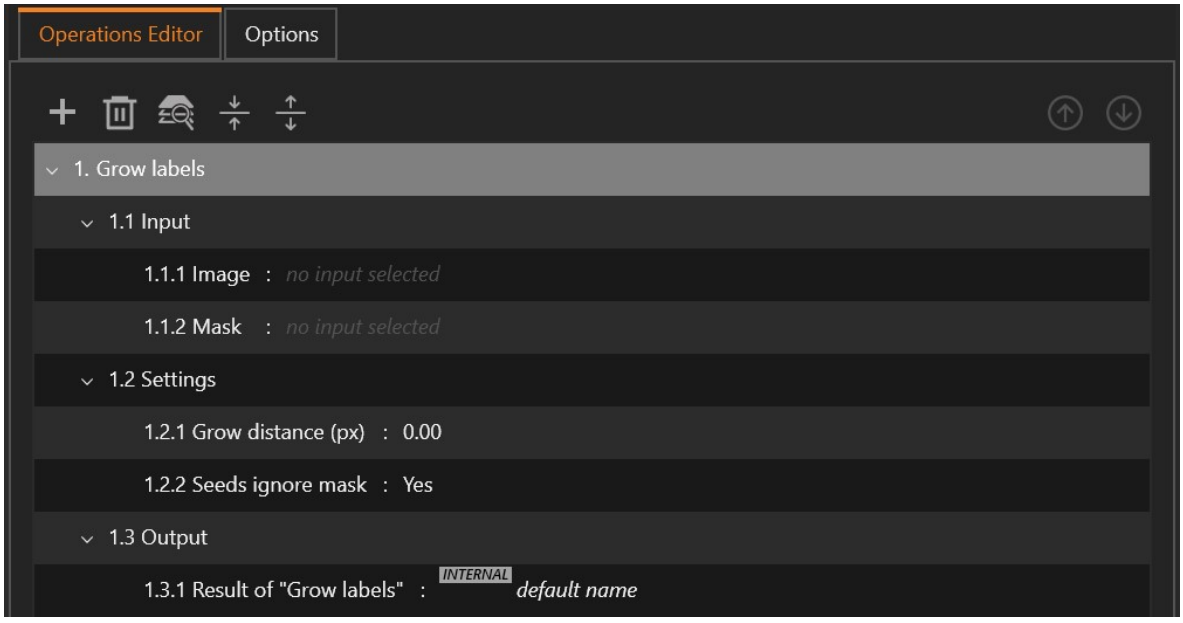


Figure 239 – Labels Grow - V2

Where it can be found:

Basic Operations Module ► Labels ► Grow Labels

Description:

Grow Labels is a label operation available in Basic Operations Module.

This operation grows the labels (labelled objects) present in image and places the result in the output.

The growing process uses labels to assimilate the background pixels (unlabeled pixels) in the proximity. The assimilation is prioritized by distance and each pixel is assimilated by the closest label. The process is limited by a maximum growing distance provided as input. Additionally, the growing process can be restricted by a validation mask (only pixels indicated by the mask are used in the growing process).

Parameters:

- 1.1 Image - the input image
- 1.2 Mask - the input mask. Allows labels growing only in the specified areas (white)
- 1.3 Grow distance (px) - the growing distance in pixels (default = 0);
- 1.4 Seeds ignore mask - enables / disables mask constraint on seeds (default = Yes).

Effect:

Performs constrained (distance, growing mask) labels growing.

Example:

- Input:

The image is a coded image (events) representing the detected structures (seeds) in a colon sample. Figure below shows the structure seeds highlighted in green.

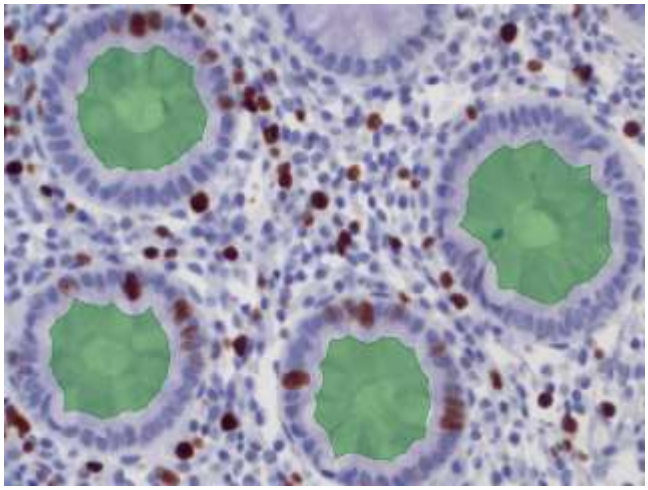


Figure 240 – Colon sample – partially detected structure

The growth is performed using a mask image. Growing is allowed only in the area(s) indicated by white.

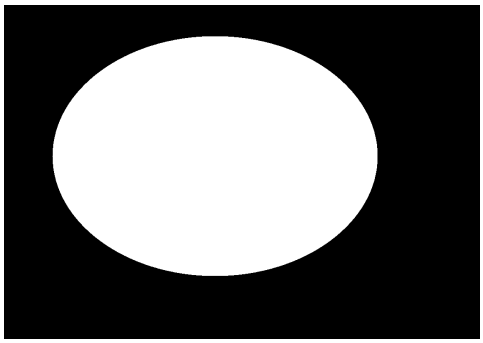


Figure 241 – Mask image used for growing

- Engine settings:

Two case have been considered for this example:

- case 1: Seeds Ignores Mask = no
- case 2: Seeds Ignores Mask = yes

The engine's results are coded images (events), matching the input.

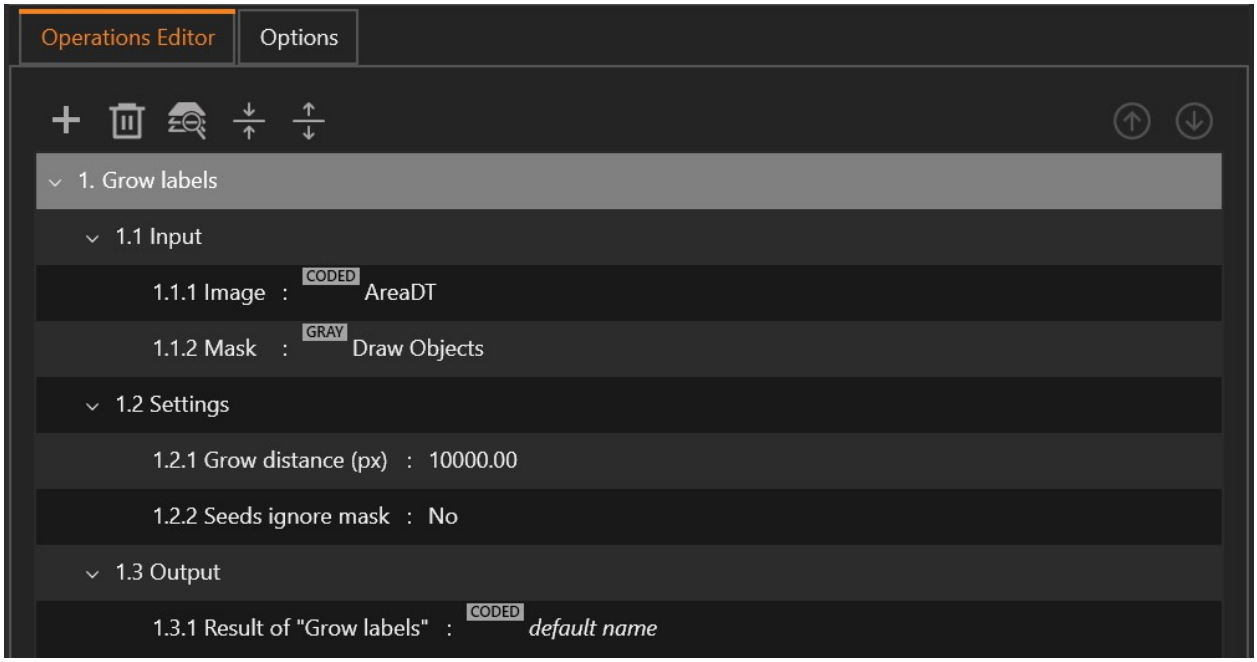


Figure 242 – Engine settings (case 1)

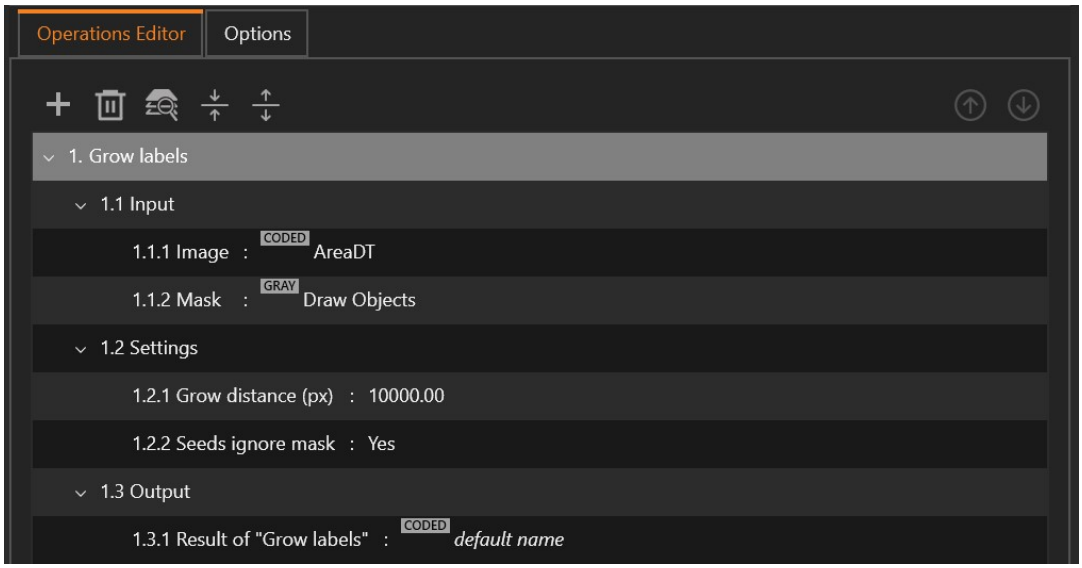


Figure 243 – Engine settings (case2)

- Output:

The 2 figures show the input events (structure seeds highlighted in cyan) and the grown events (a different color for each event: green, red, orange, brown) overlaid on the original color image.

First figure shows the growing results for the settings used in case 1. Everything found outside of the allowed growing area (mask) is deleted.

Second figure shows the growing results for the settings used in case 2. The full seeds bodies are preserved, even if they are partially outside of the allowed growing area (mask).

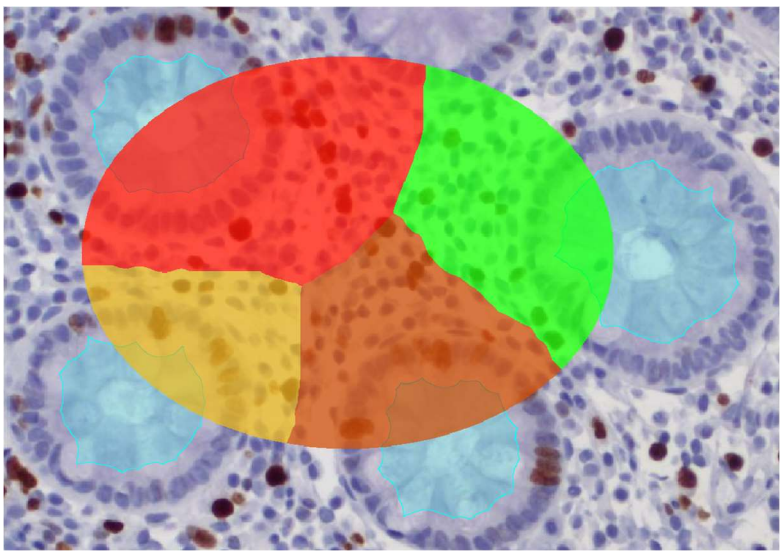


Figure 244 – Result of Grow labels engine (case 1)

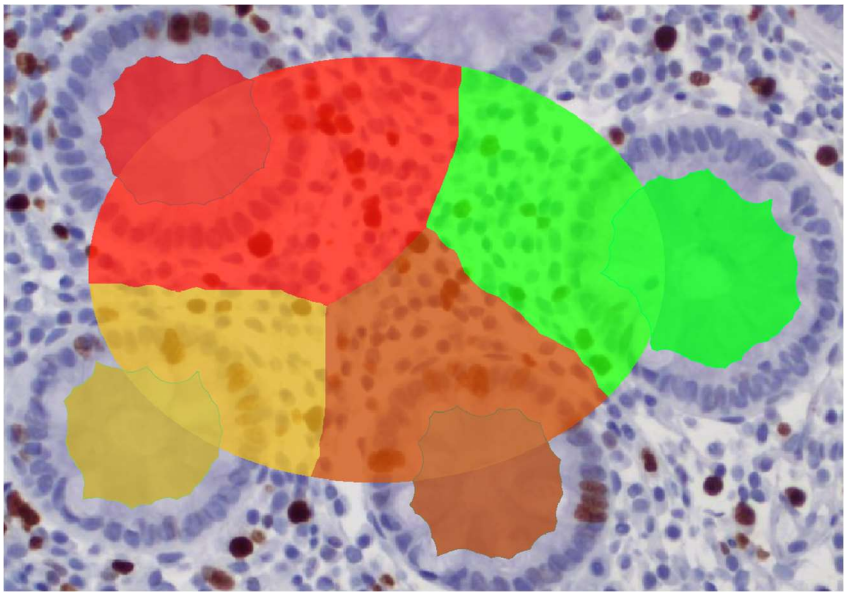


Figure 245 – Result of Grow labels engine (case 2)

Label

Figure 246 – Label Markers

Where it can be found:

Basic Operations Module ► Labels ► Label

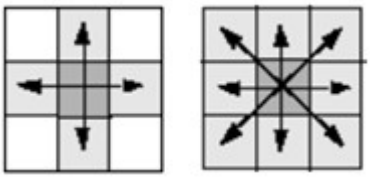
Description:

Label is a label operation available in Basic Operations Module.

This operation labels markers in a mask image (black & white) with different values (labels) and places the result in the output.

Each connected set of non-zero image pixels is treated as a separate marker. The markers are identified based on the selected connectivity:

- 4-connected -pixels are connected if their edges touch. Two adjacent pixels are part of the same marker if they are connected horizontally or vertically.
- 8-connected -pixels are connected if their edges touch. Two adjacent pixels are part of the same object if they are connected horizontally, vertically or diagonally.



All pixels belonging to the same marker are set to the same value.

Parameters:

- 1.1 Image - the input image
- 1.2 Connectivity - specifies the connectivity method (default = 4-conn)
- 1.3 Result of "…" - the result of the operation

Effect:

Labels an image by assigning unique IDs to each disjointed object.

Example:

- Input:

The image is mask image (black and white, 8-bit, 1-channel) representing the detected seed mask of fat cells. First figure shows a brightfield fat cells sample and second figure shows the fat cells seeds mask generated by applying a threshold on the original image.

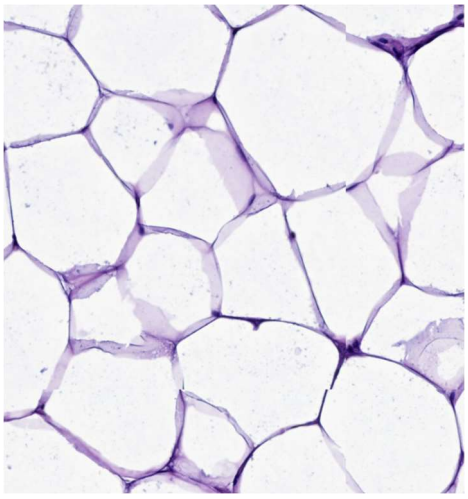


Figure 247 – Fat cells

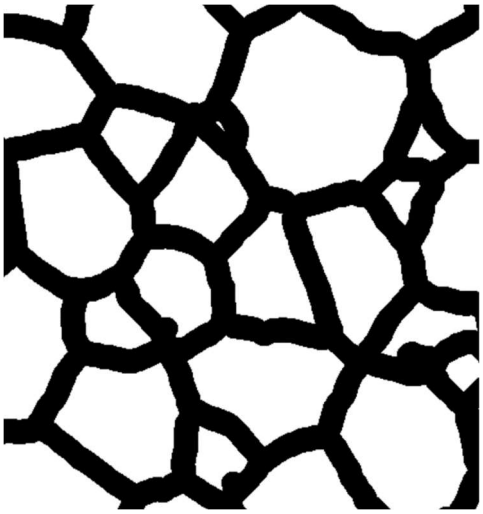


Figure 248 – Fat cells seeds (mask image)

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a coded image (events).

Figure 249 – Engine settings

- Output:

Figure below shows events generated (highlighted with different colors) from the input mask, overlaid on the original color image.

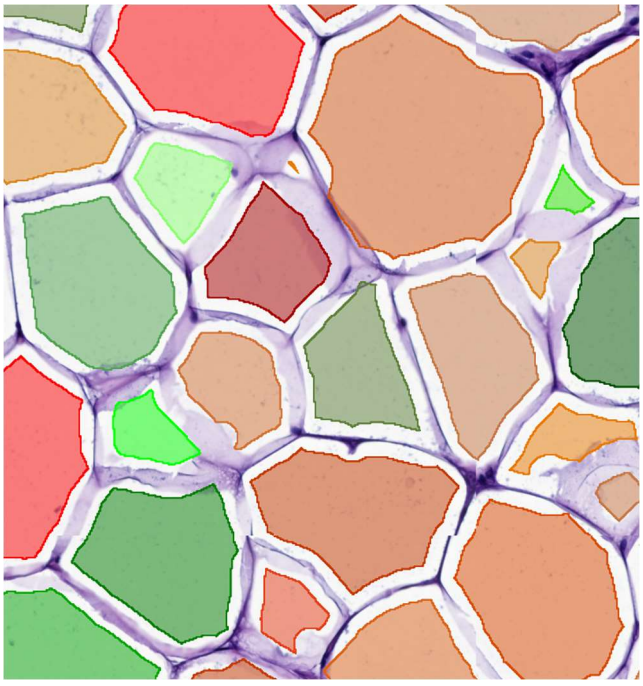


Figure 250 – Result of Label engine

Re-Label (missing)

Figure 251 – Relabel (missing)

Where it can be found:

Basic Operations Module ► Labels ► Re-Label (missing)

Description:

Re-Label (missing) is a label operation available in Basic Operations Module.

This operation re-labels an already labeled image and places the result in the output.

Combining different engines and operations can generate missing labels (e.g. Remove labels (…)). To avoid undesired problems generated by missing labels, it is recommended to use the Re-Label (missing) operation to correct the labels list.

Parameters:

- 1.1 Image - the input image
- 1.2 Result of "…" - the result of the operation

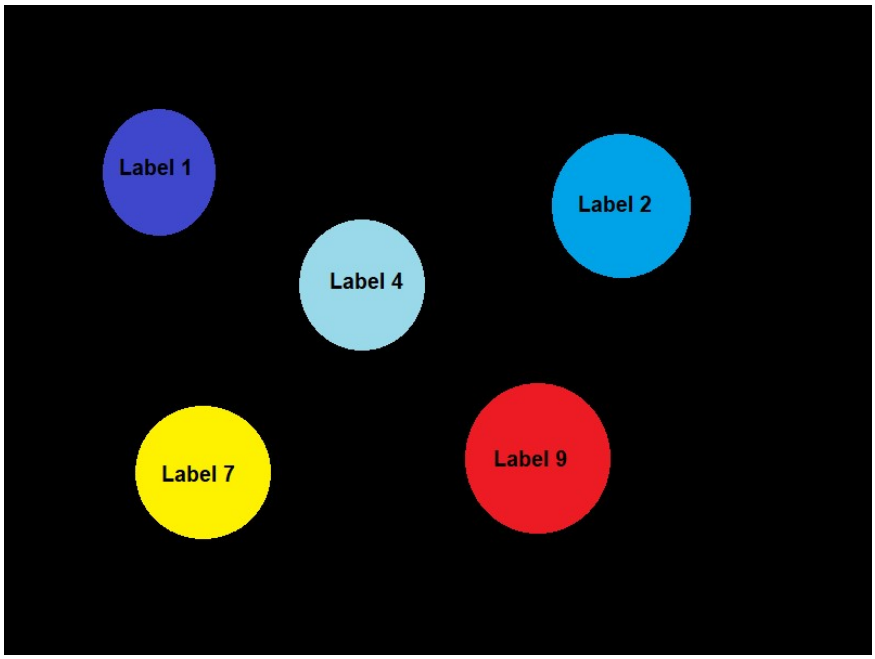
Effect:

Re-labels an image to correct missing labels.

Example:

- Input:

The image shows some non-consecutive labels generated artificially.



- Engine settings:

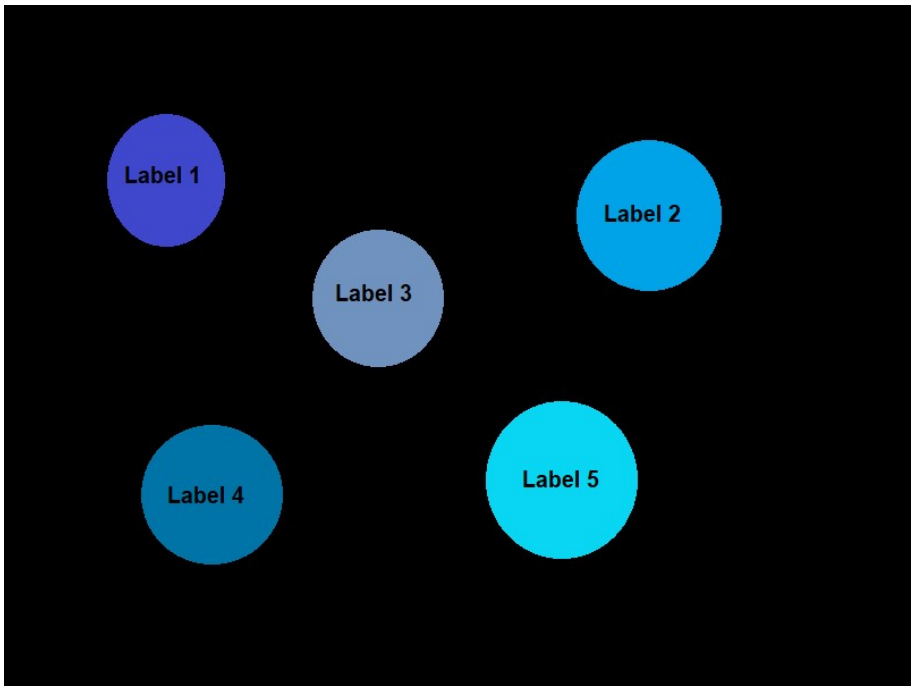
Figure below shows the engine settings used for this example.

The engine's result is a coded image (events), matching the input.

Figure 252 – Engine settings

- Output:

Figure below shows the result relabeling the input image. Some labels ended with a different value, ensuring the labels are consecutive values.



Re-Label (missing, duplicates)

Figure 253 – Relabel Split Labels

Where it can be found:

Basic Operations Module ► Labels ► Re-Label (missing, duplicates)

Description:

Re-Label (missing, duplicates) is a label operation available in Basic Operations Module.

This operation re-labels an already labelled image and places the result in the output.

Combining different engines and operations can generate missing labels (e.g. Remove labels (…) or duplicate labels (disjointed objects with the same ID) (e.g. Fuse events). To avoid these issues generated by missing and / or duplicate labels, it is recommended to use the Re-Label (missing, duplicates) operation to correct the labels list.

Parameters:

- 1.1 Image - the input image
- 1.2 Result of "…" - the result of the operation

Effect:

Re-labels an image to correct missing and / or duplicate labels.

Example:

- Input:

Although this engine corrects missing and duplicate labels, this example explains the duplicate labels correction only.

The image is a coded image (events) showing detected fat cells overlaid on a brightfield color image. The input set of events has duplicate labels, which may be the result of split labels, or other engines. Figure below shows two different events having the same label (ID).

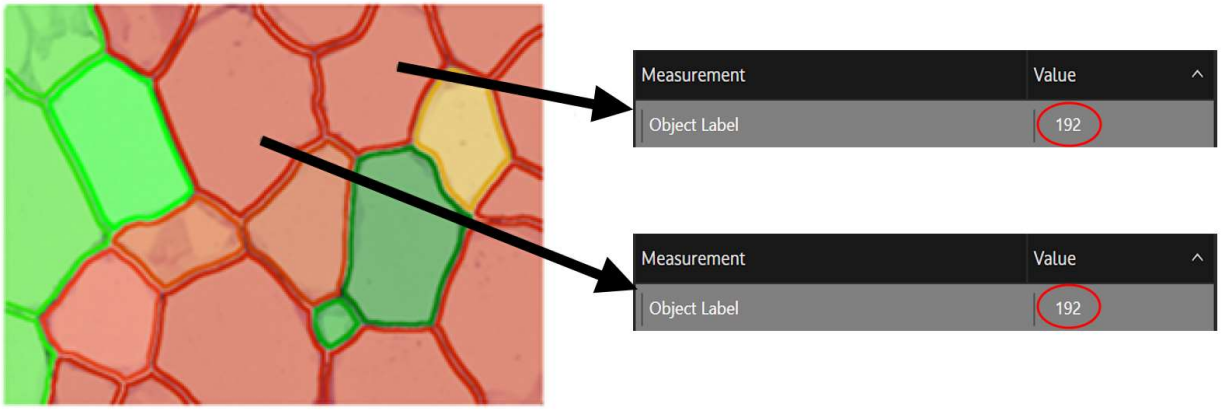


Figure 254 – Fat cells (duplicate labels)

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a coded image (events), matching the input.

Figure 255 – Engine settings

- Output:

Figure below shows the same events as before, but with corrected (different) labels (ID).

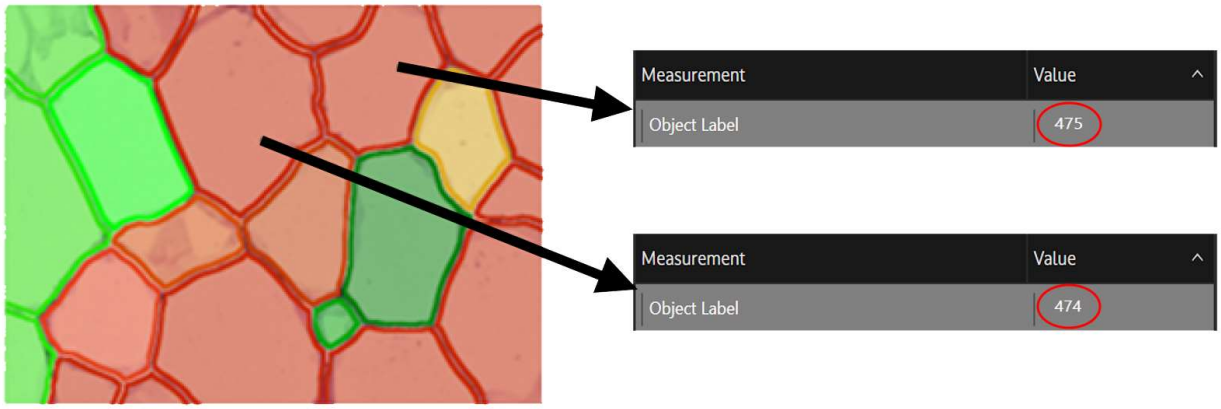


Figure 256 – Result of Re-Label (missing, duplicates) engine

Shrink labels

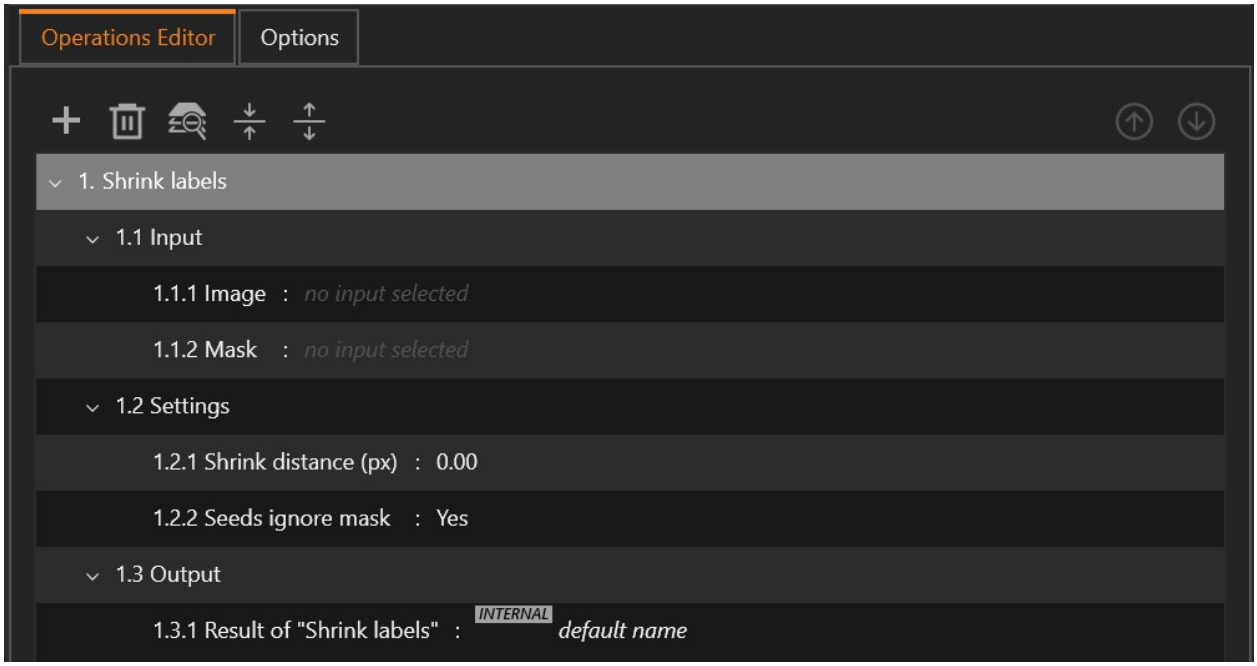


Figure 257 – Shrink Labels

Where it can be found:

Basic Operations Module ► Labels ► Shrink labels

Description:

Shrink Labels is a label operation available in Basic Operations Module.

This operation shrinks the labels (labelled objects) present in image and places the result in the output.

This operation is the inverse of Grow labels, the effect being the decrease of events bodies / areas.

Parameters:

- 1.1 Input - the input image
- 1.2 Mask - the input mask. Allows labels shrink only in the specified areas (white)
- 1.3 Shrink distance (px) - the shrink distance in pixels (default = 0);
- 1.4 Seeds ignore mask - enables / disables mask constraint on seeds (default = Yes).
- 1.5 Result of "…" - the result of the operation

Effect:

Performs constrained (distance, growing mask) labels shrinking.

Example:

- Input:

The image is a coded image (events) representing the partially detected structures (interior) in a colon sample (highlighted in green).

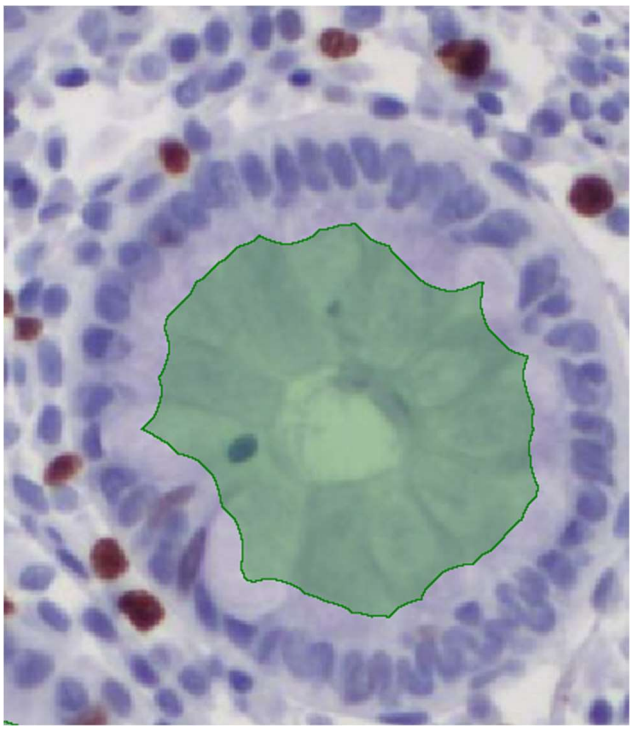


Figure 258 – Colon sample - partially detected structure

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a coded image (events), matching the input.

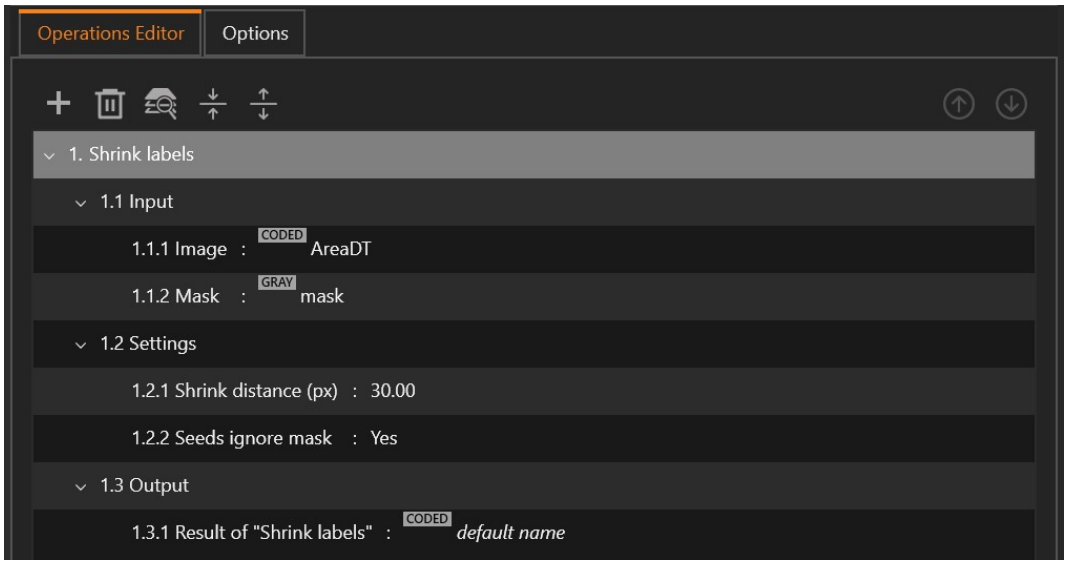


Figure 259 – Engine settings

- Output:

Figure below shows the original event (highlighted in green) and the shrinking event (highlighted in orange) overlaid on the original color image.

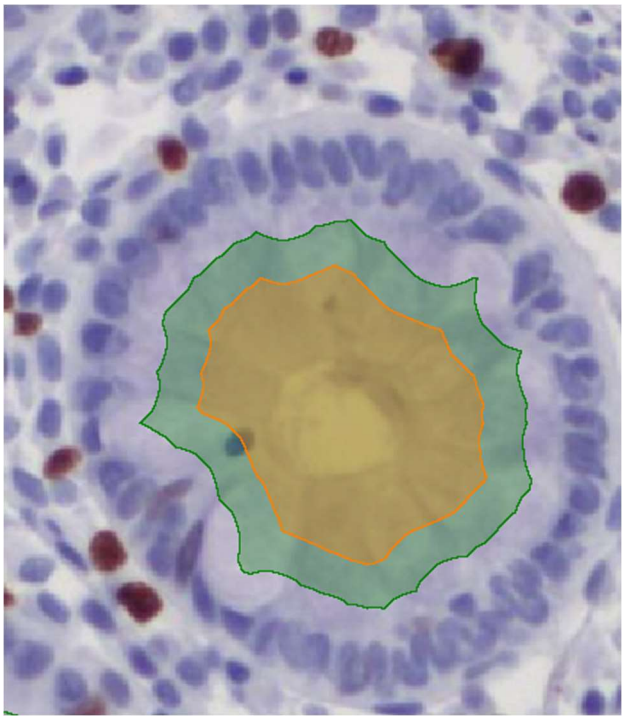


Figure 260 – Result of Shrink labels engine

5.7. Logical

And

Figure 261 – AND

Where it can be found:

Basic Operations Module ► Logical ► And

Description:

And is a logical operation available in Basic Operations Module.

This operation performs bitwise AND operations between corresponding pixels of two images and places the result in the output.

$$Result = Image\ 1 \wedge Image\ 2$$

A bitwise AND is a binary operation that takes two equal-length binary representations and performs a logical AND operation on each of the corresponding bits, which is equivalent to multiplying them. Thus, if both bits in the compared position are 1, the bit in the resulting binary representation is 1 (1 x 1 = 1). Otherwise, the result is 0 (1 x 0 = 0 and 0 x 0 = 0). For example:

$$\begin{array}{rcccccccc} & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & (decimal\ 178) \\ \boldsymbol{AND} & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & (decimal\ 37) \\ = & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & (decimal\ 32) \end{array}$$

Parameters:

- 1.1 Image 1 - the first input image
- 1.2 Image 2 - the second input image
- 1.3 Result of "…" - the result of the operation

Effect:

Performs a bitwise AND for two images.

Example:

- Input:

Image 1 is a mask image (black & white, 8-bit, 1-channel) representing an elongate shape (object A) over a vertical axis.

Image 2 is a mask image (black & white, 8-bit, 1-channel) representing an elongate shape (object B) over a horizontal axis.

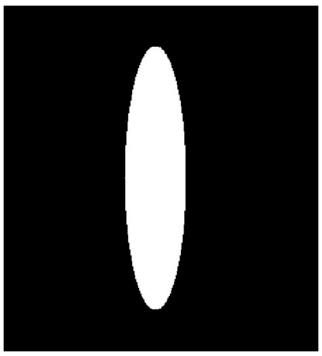


Figure 262 – Mask image 1 (object A)

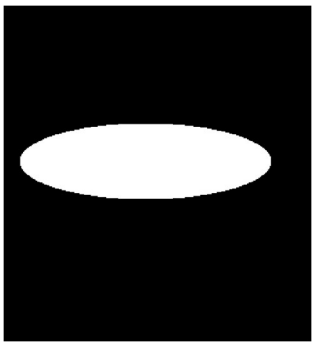


Figure 263 – Mask image 2 (object B)

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a mask image (black & white, 8-bit, 1-channel), matching the input.

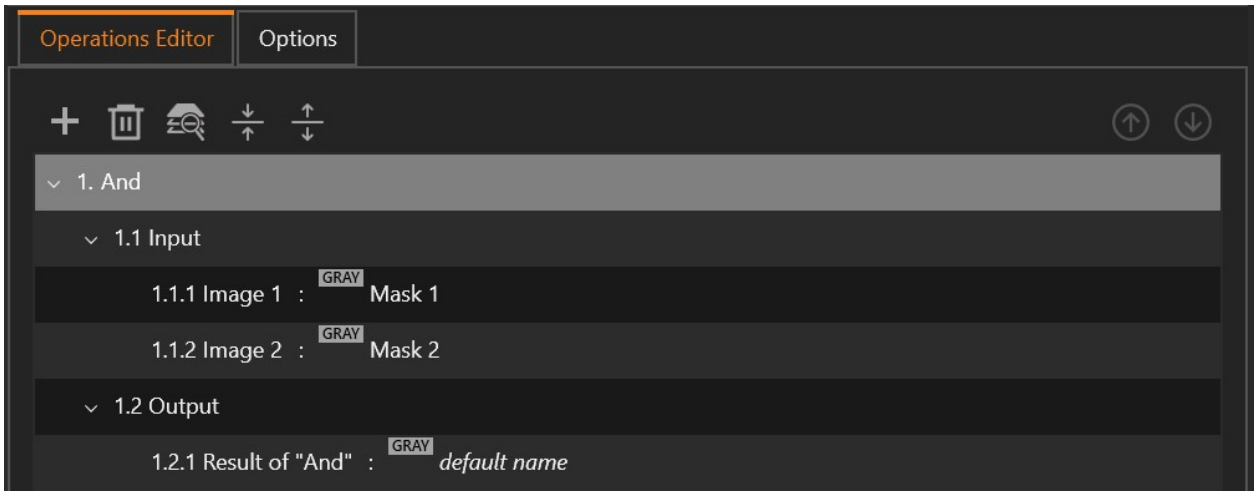


Figure 264 – Engine settings

- Output:

First figure shows the overlay of the input masks to visualize the intersection area.

First figure shows the intersection of the two input masks.

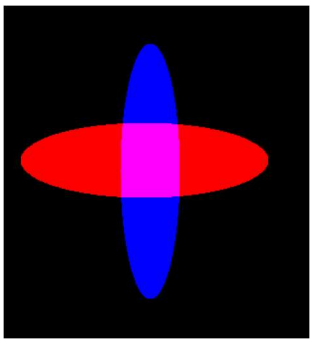


Figure 265 – Overlay of input masks

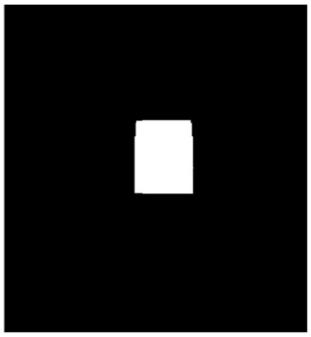


Figure 266 – Result of And engine

And (with value)

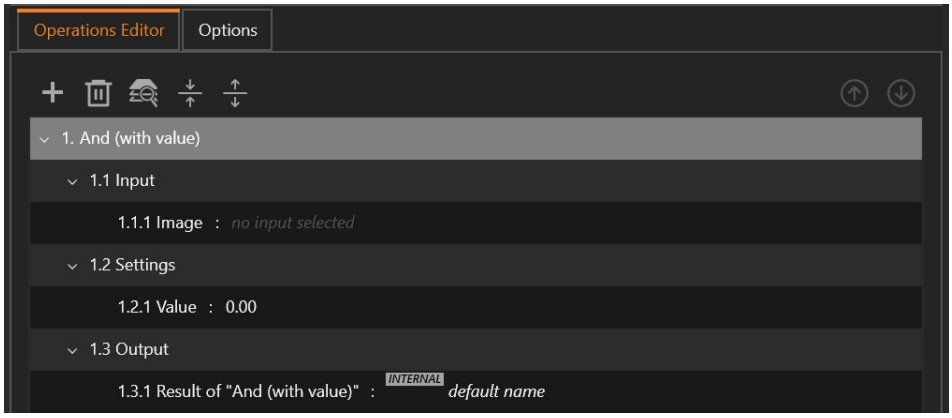


Figure 267 – ANDC

Where it can be found:

Basic Operations Module ► Logical ► And (with value)

Description:

And (with value) is a logical operation available in Basic Operations Module.

This operation performs bitwise AND operations between the pixels of an image and a numerical value and places the result in the output.

$$Result = Image \wedge value$$

A bitwise AND is a binary operation that takes two equal-length binary representation and performs logical AND operation on each of the corresponding bits, which is equivalent to multiplying them. Thus, if both bits in the compared position are 1, the bit in the resulting binary representation is 1 (1 x 1 = 1); otherwise, the result is 0 (1 x 0 = 0 and 0 x 0 = 0). For example:

$$\begin{array}{rcccccccc} & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & (decimal\ 178) \\ \boldsymbol{AND} & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & (decimal\ 37) \\ = & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & (decimal\ 32) \end{array}$$

Parameters:

- 1.1 Image - the input image
- 1.2 Value - the value used in combination with all image pixels
- 1.3 Result of "…" - the result of the operation

Effect:

Perform bitwise AND of all image pixels with a numerical value.

Example:

- Input:

Figure below (Mask image) shows a mask image representing a graph. A filter was applied on the graph image to find the number of neighbors for each pixel. Figure below (Score image) shows the score image, where:

- score = 1 for endpoints (light blue)
- score = 2 for segment points (green)
- score = 3 for intersection of 3 segments (orange)
- score = 4 for intersection of 4 segments (yellow)

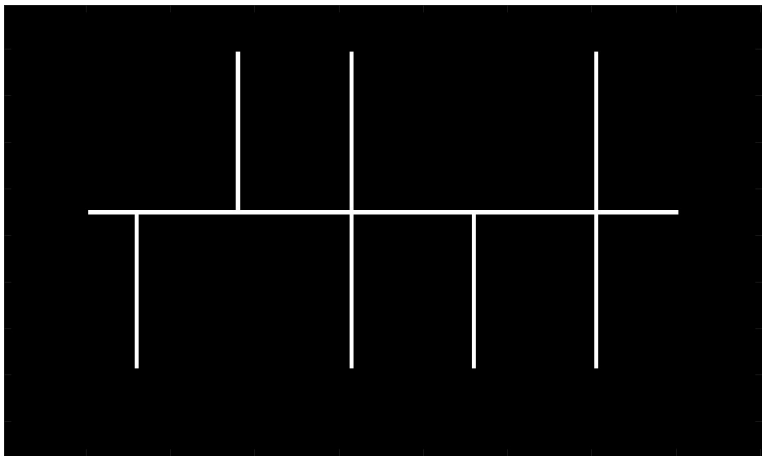


Figure 268 – Mask image

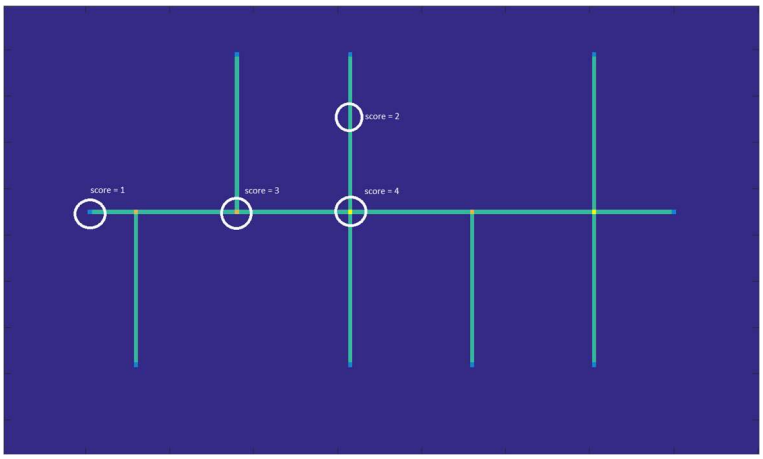


Figure 269 – Score image

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a grayscale image (8-bit, 1-channel), matching the input.

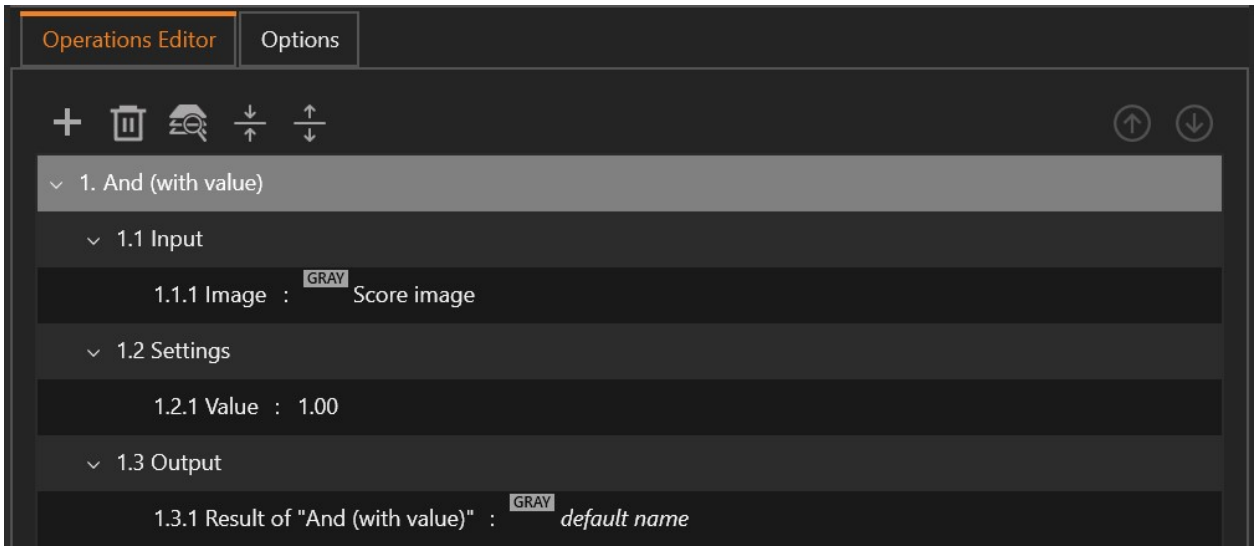


Figure 270 – Engine settings

- Output:

Figure below shows the result of performing bitwise AND operation between the score image and value = 1. The result is the identification of all points that have an odd score value:

- intersection points of 3 segments (score = 3)
- endpoints (score = 1)

Binary representation of input values:

$$
\begin{array}{lllllll}
0 & = & 0 & 0 & 0 & 0 & (background) \\
1 & = & 0 & 0 & 0 & 1 & (endpoints) \\
2 & = & 0 & 0 & 1 & 0 & (segment\ points) \\
3 & = & 0 & 0 & 1 & 1 & (intersection\ points\ of\ 3\ segments) \\
4 & = & 0 & 1 & 0 & 0 & (intersection\ points\ of\ 4\ segments)
\end{array}
$$

Binary representation of resulting values:

$$
\begin{array}{lcccccc}
\boldsymbol{AND}(0,1) & = & 0 & 0 & 0 & 0 & (decimal\ 0) \\
\boldsymbol{AND}(1,1) & = & 0 & 0 & 0 & 1 & (decimal\ 1) \\
\boldsymbol{AND}(2,1) & = & 0 & 0 & 0 & 0 & (decimal\ 0) \\
\boldsymbol{AND}(3,1) & = & 0 & 0 & 0 & 1 & (decimal\ 1) \\
\boldsymbol{AND}(4,1) & = & 0 & 0 & 0 & 0 & (decimal\ 0)
\end{array}
$$

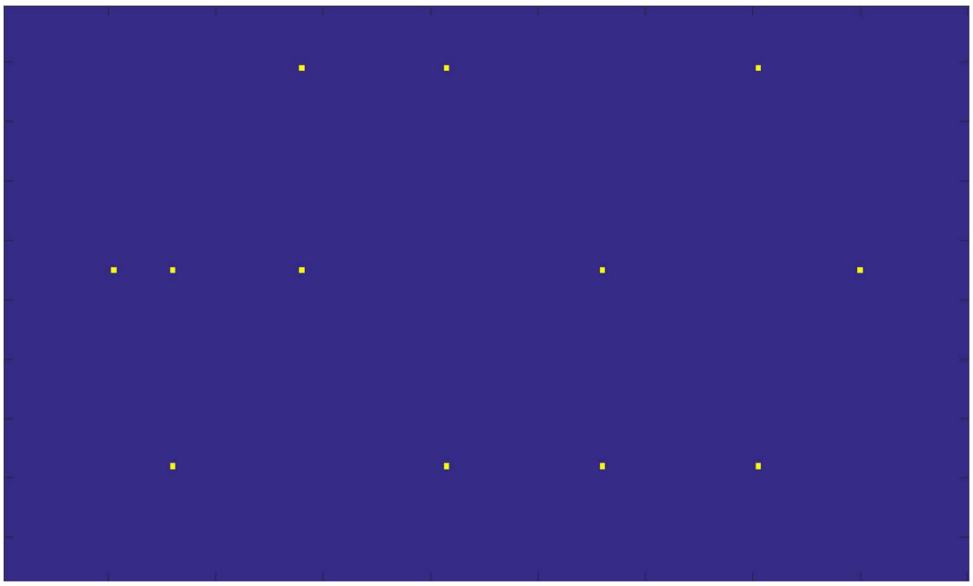


Figure 271 – Result of And (with value) engine

Not

Figure 272 – NOT

Where it can be found:

Basic Operations Module ► Logical ► Not

Description:

Not is a logical operation available in Basic Operations Module.

This operation performs bitwise NOT operations on all pixels of the image and places the result in the output.

$$Result = \sim Image$$

The bitwise NOT, or complement, is a unary operation that performs logical negation on each bit, forming the ones' complement of the given binary value. Bits that are 0 become 1, and those that are 1 become 0. For example:

$$\begin{array}{lllllllll} \boldsymbol{NOT} & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & (decimal\ 178) \\ = & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & (decimal\ 115) \end{array}$$

Parameters:

- 1.1 Image - the input image
- 1.2 Result of "…" - the result of the operation

Effect:

Perform bitwise NOT on the images.

Example:

- Input:

The image is a mask image (black & white, 8-bit, 1-channel) representing a white shape (object) on a black background (figure 3-259).

Binary representation of image values:

$$\begin{array}{rcccccccc} 0 & = & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 255 & = & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}$$

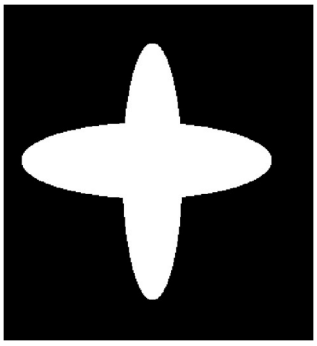


Figure 273 – Mask image

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a grayscale image (8-bit, 1-channel), matching the input.

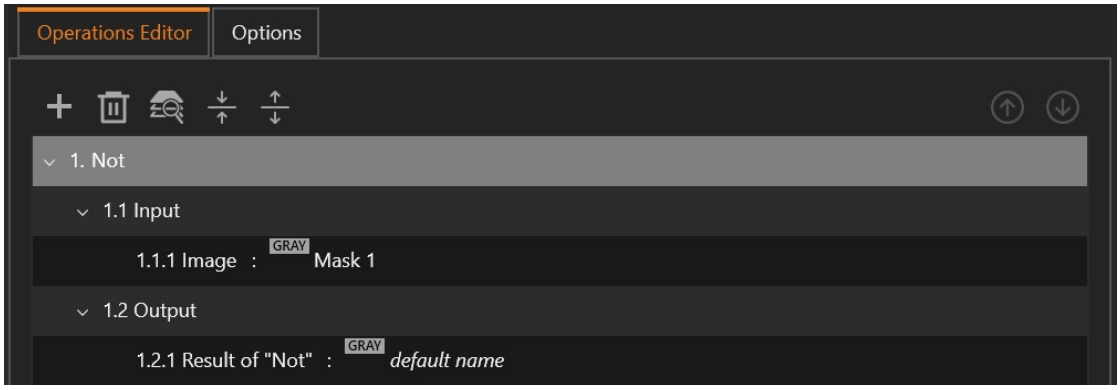


Figure 274 – Engine settings

- Output:

Figure below shows the result of the Not engine applied on the input image. The colors are inverted.

Binary representation of image values:

$$\begin{array}{lcccccccc} \boldsymbol{NOT}(0) & = & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \boldsymbol{NOT}(255) & = & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

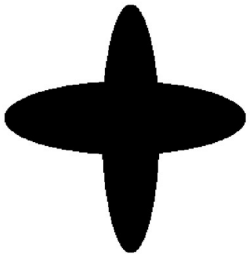


Figure 275 – Result of Not engine

Or

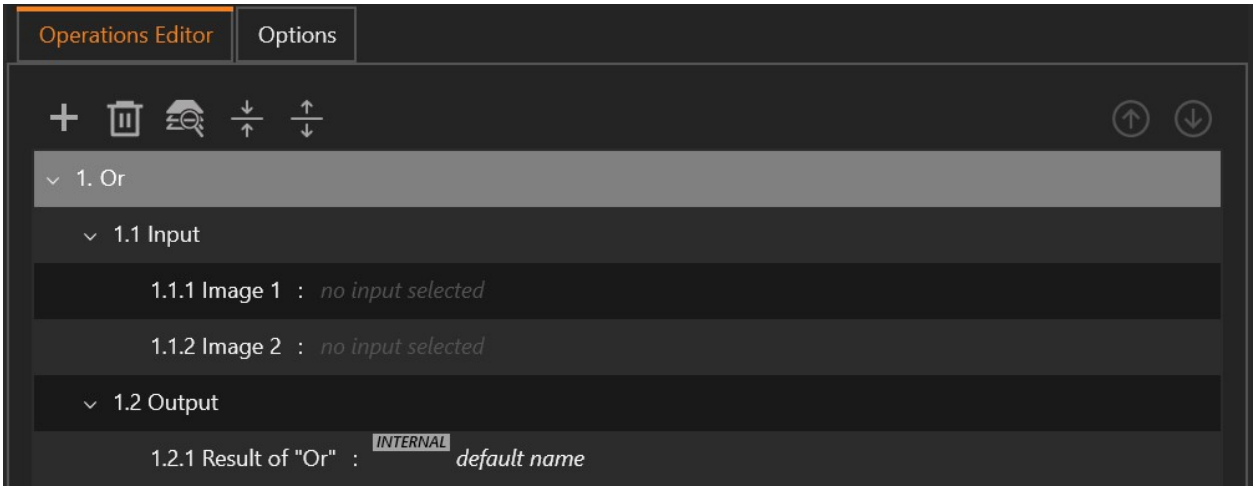


Figure 276 – OR

Where it can be found:

Basic Operations Module ► Logical ► Or

Description:

Or is a logical operation available in Basic Operations Module.

This operation performs bitwise OR operations between corresponding pixels of two images and places the result in the output.

$$Result = Image\,1 \vee Image\,2$$

A bitwise OR is a binary operation that takes two bit patterns of equal length and performs the logical inclusive OR operation on each pair of corresponding bits. The result in each position is 0 if both bits are 0. Otherwise the result is 1. For example:

$$\begin{array}{lllllllll} & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & (decimal\ 178) \\ \boldsymbol{OR} & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & (decimal\ 37) \\ = & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & (decimal\ 183) \end{array}$$

Parameters:

- 1.1 Image 1 - the first input image
- 1.2 Image 2 - the second input image
- 1.3 Result of "…" - the result of the operation

Effect:

Performs a bitwise OR for two images.

Example:

- Input:

Image 1 is a mask image (black & white, 8-bit, 1-channel) representing an elongate shape (object A) over vertical axis.

Image 2 is a mask image (black & white, 8-bit, 1-channel) representing an elongate shape (object B) over horizontal axis.

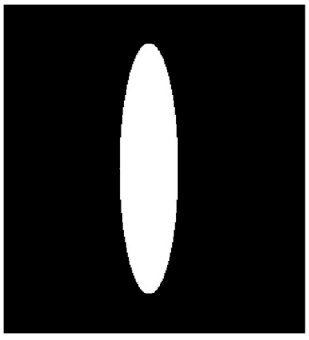


Figure 277 – Mask image 1 (object A)

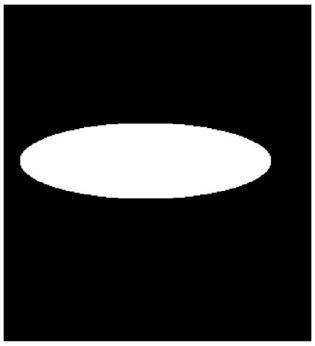


Figure 278 – Mask image 2 (object B)

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a grayscale image (8-bit, 1-channel), matching the input images.

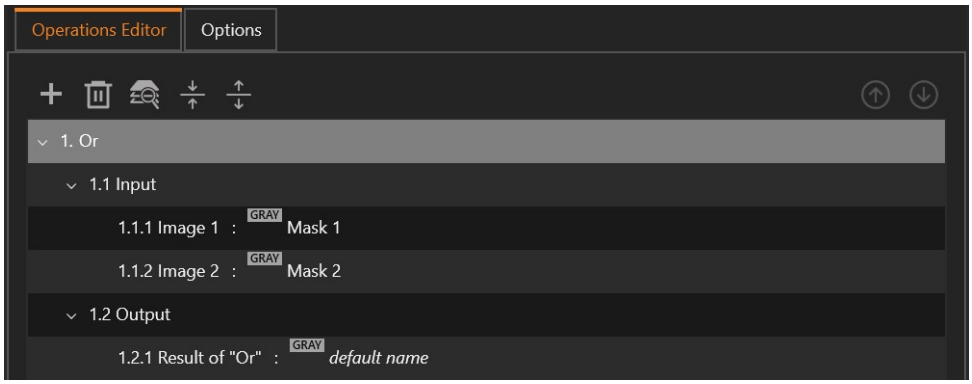


Figure 279 – Engine settings

- Output:

Figure below shows the union of the two input masks.

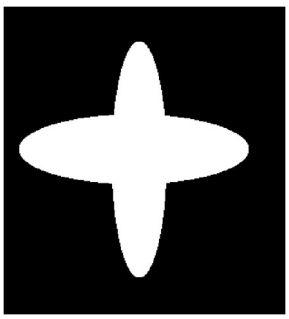


Figure 280 – Result of Or engine

Or (with value)

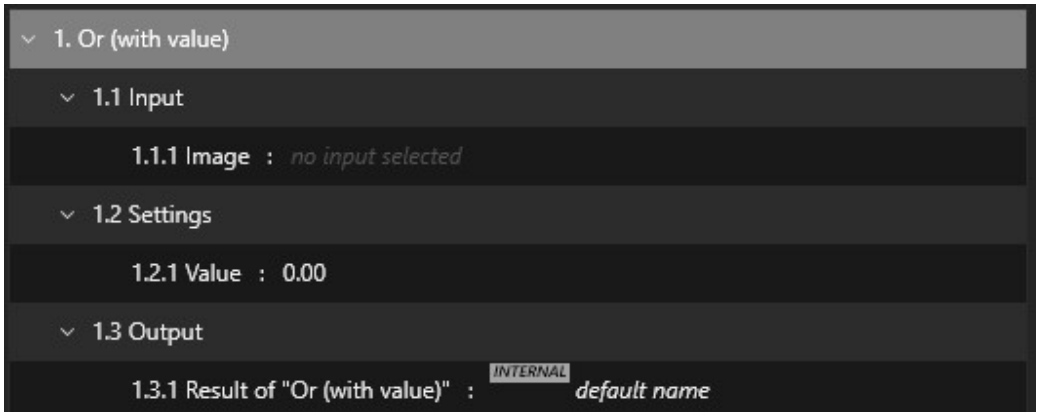


Figure 281 – Or

Where it can be found:

Basic Operations Module ► Logical ► Or (with value)

Description:

Or (with value) is a logical operation available in Basic Operations Module.

This operation performs bitwise OR operations between the pixels of an image and a numerical value and places the result in the output.

$$Result = Image \vee value$$

A bitwise OR is a binary operation that takes two bit patterns of equal length and performs the logical inclusive OR operation on each pair of corresponding bits. The result in each position is 0 if both bits are 0. Otherwise the result is 1. For example:

$$\begin{array}{lcccccccc} & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & (decimal\ 178) \\ \boldsymbol{OR} & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & (decimal\ 37) \\ = & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & (decimal\ 183) \end{array}$$

Parameters:

- 1.1 Image - the input image
- 1.2 Value - the value used in combination with all image pixels
- 1.3 Result of "…" - the result of the operation

Effect:

Performs a bitwise OR of all image pixels with a numerical value.

Example:

- Input:

Figure below (Mask image) shows a mask image representing a graph. A filter was applied on the graph image to find the number of neighbors for each pixel. Figure below (Score image) shows the score image, where:

- score = 1 for endpoints (light blue)
- score = 2 for segment points (green)
- score = 3 for intersection of 3 segments (orange)
- score = 4 for intersection of 4 segments (yellow)

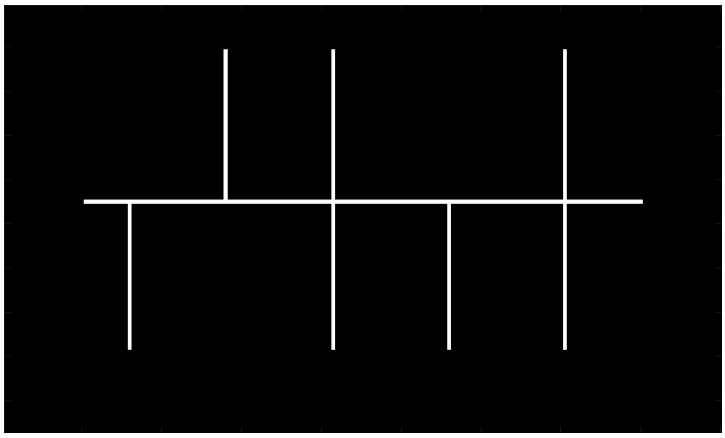


Figure 282 – Mask image

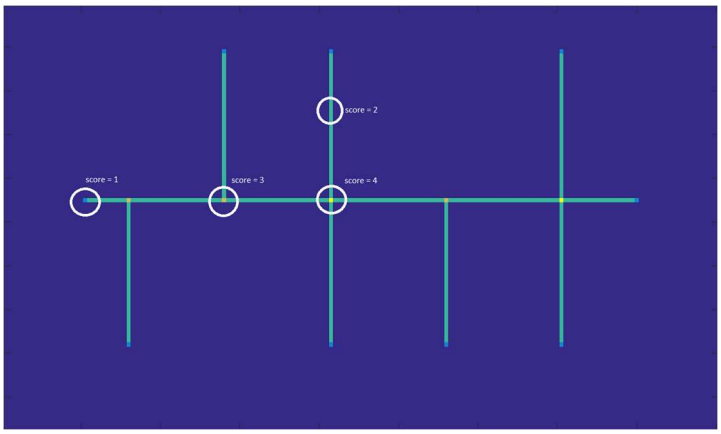


Figure 283 – Score image

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a grayscale image (8-bit, 1-channel), matching the input image.

Figure 284 – Engine settings

- Output:

Figure below shows the result of performing bitwise OR operation between the score image and value = 3. The result is the identification of all points that have a score value higher than 3 (intersecting points of 4 segments).

Binary representation of input values:

$$
\begin{array}{llllll}
0 & = & 0 & 0 & 0 & 0 \quad (background) \\
1 & = & 0 & 0 & 0 & 1 \quad (endpoints) \\
2 & = & 0 & 0 & 1 & 0 \quad (segment\ points) \\
3 & = & 0 & 0 & 1 & 1 \quad (intersection\ points\ of\ 3\ segments) \\
4 & = & 0 & 1 & 0 & 0 \quad (intersection\ points\ of\ 4\ segments)
\end{array}
$$

Binary representation of resulting values:

$$
\begin{array}{llllll}
\boldsymbol{OR}(0,3) & = & 0 & 0 & 1 & 1 \quad (decimal\ 3) \\
\boldsymbol{OR}(1,3) & = & 0 & 0 & 1 & 1 \quad (decimal\ 3) \\
\boldsymbol{OR}(2,3) & = & 0 & 0 & 1 & 1 \quad (decimal\ 3) \\
\boldsymbol{OR}(3,3) & = & 0 & 0 & 1 & 1 \quad (decimal\ 3) \\
\boldsymbol{OR}(4,3) & = & 0 & 1 & 1 & 1 \quad (decimal\ 5)
\end{array}
$$

Figure 285 – Result of Or (with value) engine

Xor

Figure 286 – XOR

Where it can be found:

Basic Operations Module ► Logical ► Xor

Description:

Xor is a logical operation available in Basic Operations Module.

This operation performs bitwise XOR operations between the corresponding pixels of two images and places the result in the output.

$$Result = Image\ 1 \oplus Image\ 2$$

A bitwise XOR is a binary operation that takes two bit patterns of equal length and performs the logical exclusive OR operation on each pair of corresponding bits. The result in each position is 1 if only one of the bits is 1, but will be 0 if both are 0 or both

are 1. If comparing the two bits, the result is 1 if the two bits are different, and 0 if they are the same. For example:

$$\begin{array}{rcccccccc} & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & (decimal\ 178) \\ \boldsymbol{XOR} & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & (decimal\ 37) \\ = & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & (decimal\ 151) \end{array}$$

Parameters:

- 1.1 Image 1 - the first input image
- 1.2 Image 2 - the second input image
- 1.3 Result of "…" - the result of the operation

Effect:

Perform bitwise XOR for two images.

Example:

- Input:

Image 1 is a mask image (black & white, 8-bit, 1-channel) representing an elongate shape (object A) over vertical axis.

Image 2 is a mask image (black & white, 8-bit, 1-channel) representing an elongate shape (object B) over horizontal axis.

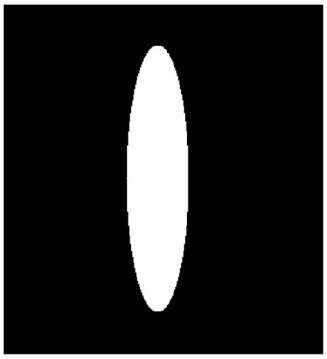


Figure 287 – Mask image 1 (object A)

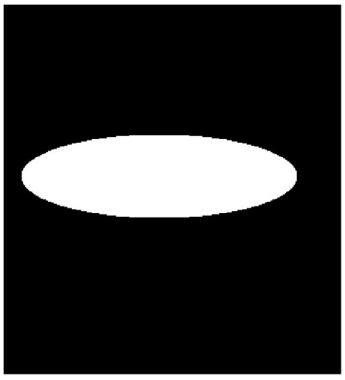


Figure 288 – Mask image 2 (object B)

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a grayscale image (8-bit, 1-channel), matching the input images.

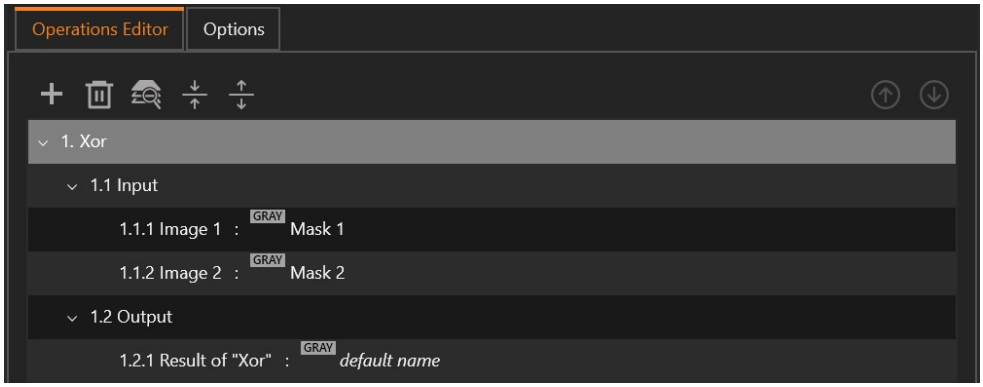


Figure 289 – Engine settings

- Output:

Figure below shows the exclusive disjunction of the two input masks.

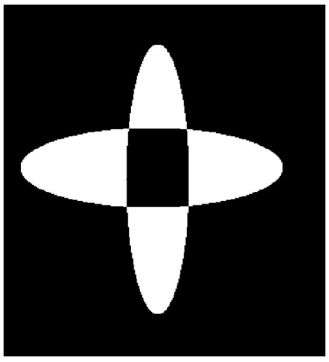


Figure 290 – Result of Or engine

Xor (with value)

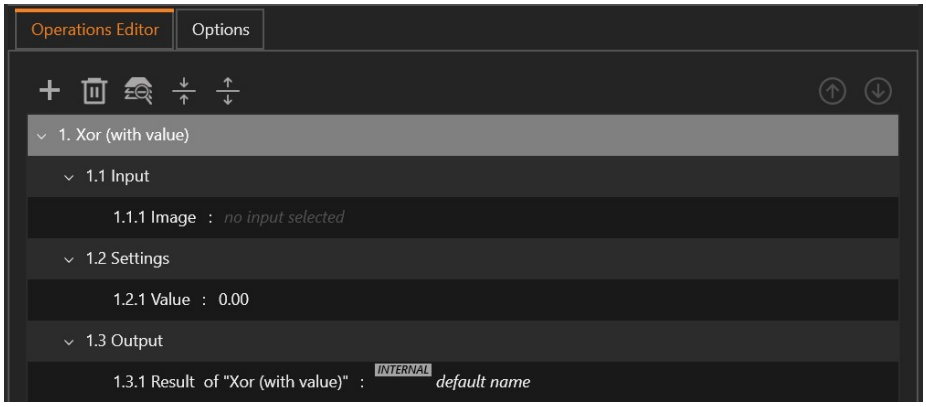


Figure 291 – XOR

Where it can be found:

Basic Operations Module ► Logical ► Xor (with value)

Description:

Xor (with value) is a logical operation available in Basic Operations Module.

This operation performs bitwise XOR operations between the pixels of an image and a numerical value and places the result in the output.

$$Result = Image \oplus value$$

Bitwise XOR (exclusive or) is a binary / logical operation which returns 1 / true if the inputs have different values and returns 0 / false if the inputs have equal values. The operation can be extended to big number, by representing the numbers in binary format and applying the XOR operation for each pair of bits.

When comparing the two bits, the result is 1 if the two bits are different, and 0 if they are the same. For example:

$$\begin{array}{lllllllll} & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & (decimal\ 178) \\ \boldsymbol{XOR} & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & (decimal\ 37) \\ = & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & (decimal\ 151) \end{array}$$

Parameters:

- 1.1 Image - the input image
- 1.2 Value - the value used in combination with all image pixels
- 1.3 Result of "…" - the result of the operation

Effect:

Perform bitwise AND of all image pixels with a numerical value.

Example:

- Input:

Figure below (Mask image) show a mask image representing a graph. A filter was applied on the graph image to find the number of neighbors for each pixel. Figure below (Score image) shows the score image, where:

- score = 1 for endpoints (light blue)
- score = 2 for segment points (green)
- score = 3 for intersection of 3 segments (orange)
- score = 4 for intersection of 4 segments (yellow)

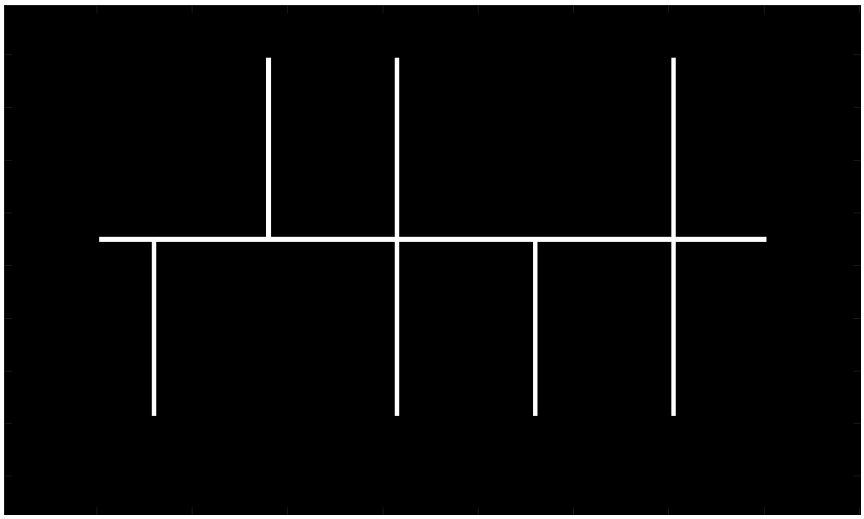


Figure 292 – Mask image

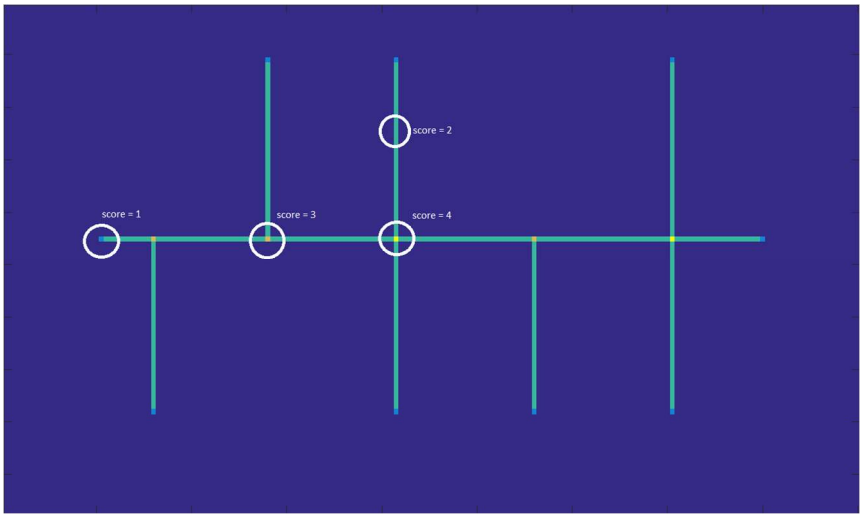


Figure 293 – Score image

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a grayscale image (8-bit, 1-channel), matching the input image.

Figure 294 – Engine settings

- Output:

Figure below shows the result of performing bitwise XOR operation between the score image and value = 2. The result is the identification of all elements except segment points (score = 2). Even the background is amplified.

Binary representation of input values:

$$
\begin{array}{llllll}
0 & = & 0 & 0 & 0 & 0 \quad (background) \\
1 & = & 0 & 0 & 0 & 1 \quad (endpoints) \\
2 & = & 0 & 0 & 1 & 0 \quad (segment\ points) \\
3 & = & 0 & 0 & 1 & 1 \quad (intersection\ points\ of\ 3\ segments) \\
4 & = & 0 & 1 & 0 & 0 \quad (intersection\ points\ of\ 4\ segments)
\end{array}
$$

Binary representation of resulting values:

$$
\begin{array}{llllll}
\boldsymbol{XOR}(0,2) & = & 0 & 0 & 1 & 0 \quad (decimal\ 2) \\
\boldsymbol{XOR}(1,2) & = & 0 & 0 & 1 & 1 \quad (decimal\ 3) \\
\boldsymbol{XOR}(2,2) & = & 0 & 0 & 0 & 0 \quad (decimal\ 0) \\
\boldsymbol{XOR}(3,2) & = & 0 & 0 & 0 & 1 \quad (decimal\ 1) \\
\boldsymbol{XOR}(4,2) & = & 0 & 1 & 1 & 0 \quad (decimal\ 6)
\end{array}
$$

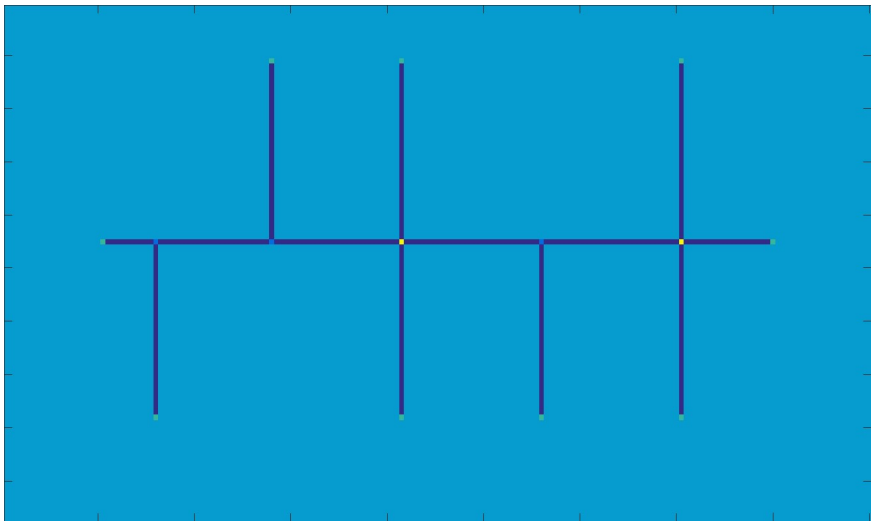


Figure 295 – Result of Xor (with value) engine

5.8. Morphological

Close (dilate, erode)

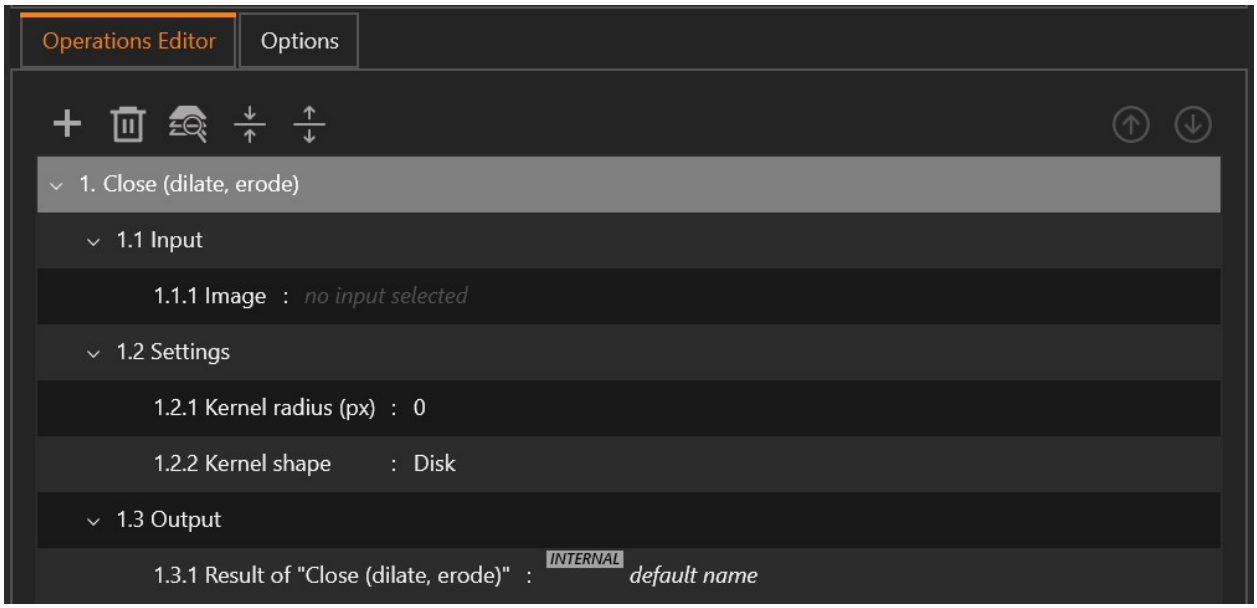


Figure 296 – Morphological - Close (dilate, erode)

Where it can be found:

Basic Operations Module ► Morphological ► Close (dilate, erode)

Description:

Close (dilate, erode) is a morphological operation available in Basic Operations Module.

This operation performs morphological closing of an image and places the result in the output.

Morphological close of an image represents the succession of dilation and erosion of the image, using the same structuring element (kernel).

$$Result = Image \bullet K = (Image \oplus K) \ominus K$$

where:

- $\bullet$ - denotes the close operation
- $\oplus$ - denotes the dilation operation
- $\ominus$ - denotes the erosion operation
- K -denotes the structuring element (kernel)

- 1.1 Image - the input image
- 1.2 Kernel radius (px) - determines the size of processing window (default = 0) using the formulas:

$$Kernel\ width = 2 * \boldsymbol{Kernel\ radius(px)} + 1$$

$$Kernel\ height = 2 * \boldsymbol{Kernel\ radius(px)} + 1$$

- 1.3 Kernel shape - specifies the shape of the kernel (default = Disk)
- 1.4 Result of "…" - the result of the operation

Effect:

Performs a morphological close of an image to remove noise, remove small holes, etc.

Example:

- Input:

Two cases are considered:

- case 1: Image is mask image (black & white, 8-bit, 1-channel)
- case 2: Image is a fluorescence grayscale image (8-bit, 1-channel) representing the DAPI channel of an UV irradiated skin sample

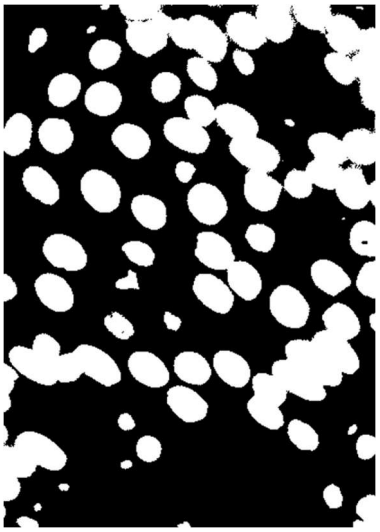


Figure 297 – Mask image (case 1)

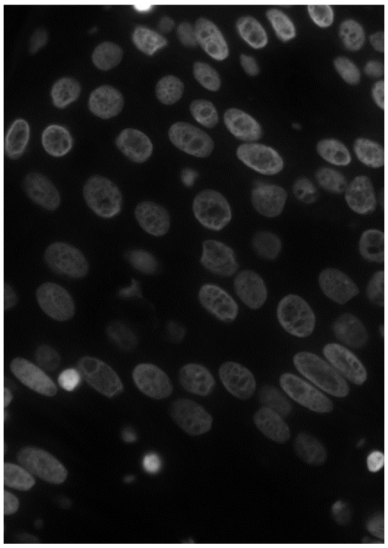


Figure 298 – UV irradiated skin sample - DAPI channel (case 2)

- Engine settings:

The engine settings used for the 2 examples are presented in case 1 and case 2.

The engine's result is a mask image (black & white, 8-bit, 1-channel) for case 1, and a grayscale image (8-bit, 1-channel) for case 2. In each case, the output matches the input.

Figure 299 – Engine settings (case 1)

Figure 300 – Engine settings (case 2)

- Output:

First figure shows the result of a morphological close operation applied on the input mask image (case 1).

Second figure shows the result of a morphological close operation applied on the input grayscale image (case 2).

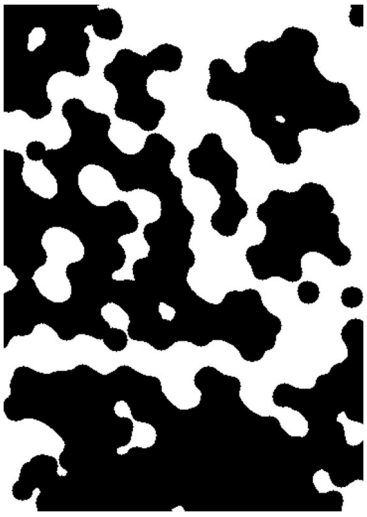


Figure 301 – Result of Close (dilate, erode) engine (case 1)

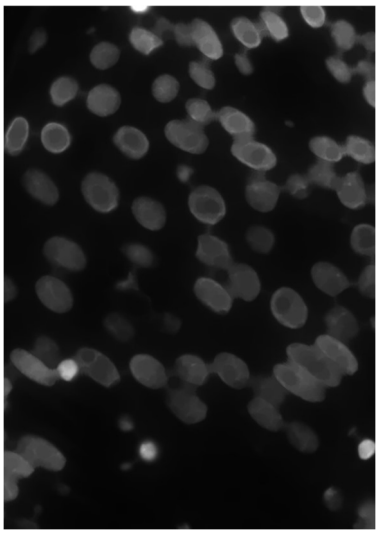


Figure 302 – Result of Close (dilate, erode) engine (case 2)

Dilate

Figure 303 – Morphological - Dilate

Where it can be found:

Basic Operations Module ► Morphological ► Dilate

Description:

Dilate is a morphological operation available in Basic Operations Module.

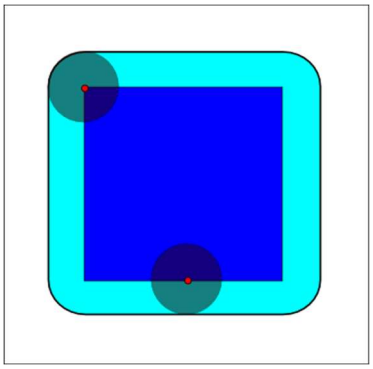
This operation performs morphological dilation of an image and places the result in the output.

$$Result = Image \oplus K$$

where:

- $\oplus$ - denotes the dilation operation
- K -denotes the structuring element (kernel)

The dilation of an Object by a structuring element K can be understood as the locus of the points covered by K when the center of K moves inside Object. The dilation of a square of size = 10 by a disk of radius = 2 is a square of side = 14, with rounded corners, centred at the origin. The radius of the rounded corners is 2.



The dilation of a dark-blue square by a disk, resulting in the light-blue square with rounded corners.

Parameters:

- 1.1 Image - the input image
- 1.2 Kernel radius (px) - determines the size of processing window (default = 0) using the formulas:

$$Kernel\ width = 2 * \boldsymbol{Kernel\ radius(px)} + 1$$

$$Kernel\ height = 2 * \boldsymbol{Kernel\ radius(px)} + 1$$

- 1.3 Kernel shape - specifies the shape of the kernel (default = Disk)
- 1.4 Result of "…" - the result of the operation

Effect:

Performs morphological dilation to inflate the shapes contained in the image.

Example:

- Input:

Two cases are considered:

- case 1: Image is mask image (black & white, 8-bit, 1-channel)
- case 2: Image is a fluorescence grayscale image (8-bit, 1-channel) representing the DAPI channel of an UV irradiated skin sample

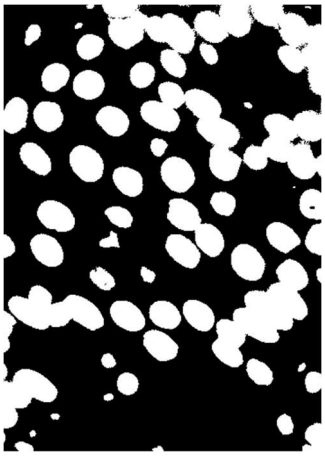


Figure 304 – Mask image (case 1)

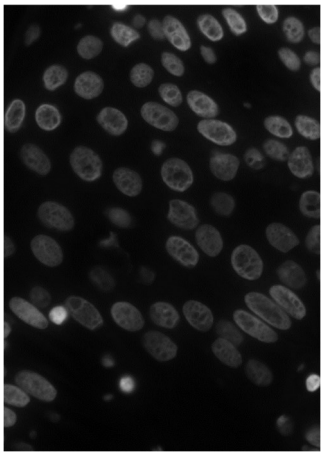


Figure 305 – UV irradiated skin sample - DAPI channel (case 2)

- Engine settings:

The engine settings used for the 2 examples are presented in case 1 and case 2.

The engine's result is a mask image (black & white, 8-bit, 1-channel) for case 1, respectively a grayscale image (8-bit, 1-channel) for case 2. In each case, the output matches the input.

Figure 306 – Engine settings (case 1)

Figure 307 – Engine settings (case 2)

- Output:

First figure shows the result of morphological close operation applied on the input mask image (case 1).

Second figure shows the result of morphological close operation applied on the input grayscale image (case 2).

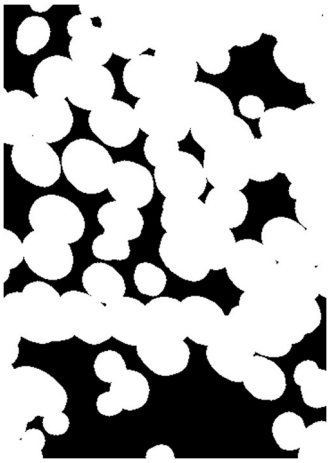


Figure 308 – Result of Dilate engine (case 1)

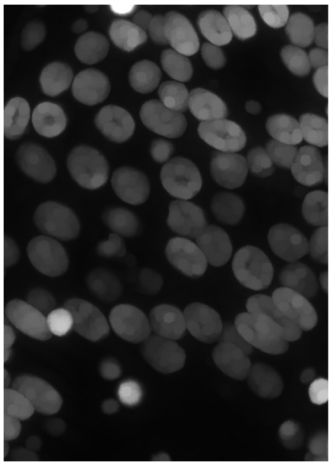


Figure 309 – Result of Dilate engine (case 2)

Dilate (with black border)

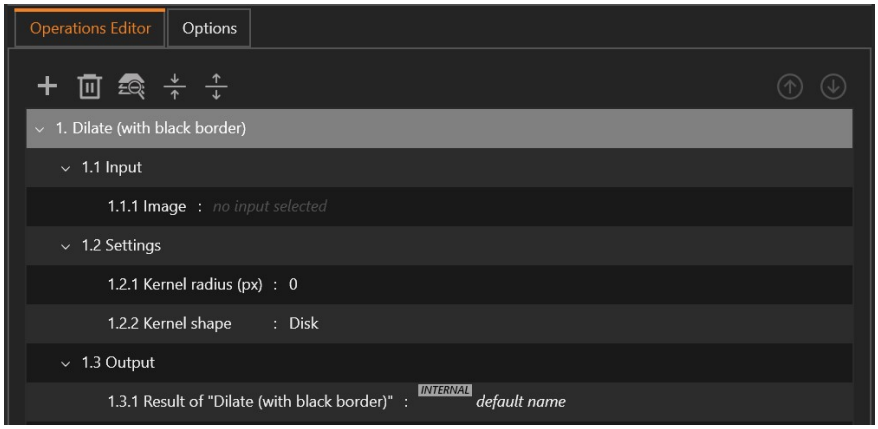


Figure 310 – Morphological - Dilate Const0

Where it can be found:

Basic Operations Module ► Morphological ► Dilate (with black border)

Description:

Dilate (with black border) is a morphological operation available in Basic Operations Module.

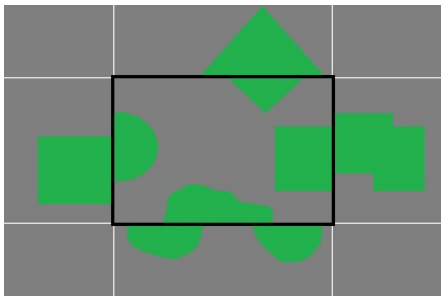
This operation adds a black border to the image, performs morphological dilation on the bordered image and places the result in the output.

$$Result = Image \oplus K$$

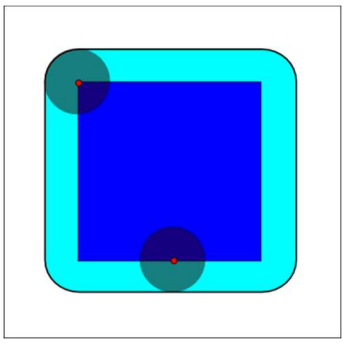
where:

- $\oplus$ - denotes the dilation operation
- K -denotes the structuring element (kernel)

Note	When Border Input option is enabled, this engine has a similar effect as Dilate (see above).



The dilation of an Object by a structuring element K can be understood as the locus of the points covered by K when the center of K moves inside the Object. The dilation of a square of size = 10 by a disk of radius = 2 is a square of side = 14, with rounded corners, cantered at the origin. The radius of the rounded corners is 2.



The dilation of the dark-blue square by a disk, resulting in the light-blue square with rounded corners.

Parameters:

- 1.1 Image - the input image
- 1.2 Kernel radius (px) - determines the size of the processing window (default = 0) using the formulas:

$$Kernel\ width = 2 * \boldsymbol{Kernel\ radius(px)} + 1$$

$$Kernel\ height = 2 * \boldsymbol{Kernel\ radius(px)} + 1$$

- 1.3 Kernel shape - specifies the shape of the kernel (default = Disk)
- 1.4 Result of "…" - the result of the operation

Effect:

Performs a morphological dilation to inflate the shapes contained in the image.

Example:

This engine has identical results as the Dilate engine (see above), with the Border Input option disabled.

Erode

Figure 311 – Morphological - Erode

Where it can be found:

Basic Operations Module ► Morphological ► Erode

Description:

Erode is a morphological operation available in Basic Operations Module.

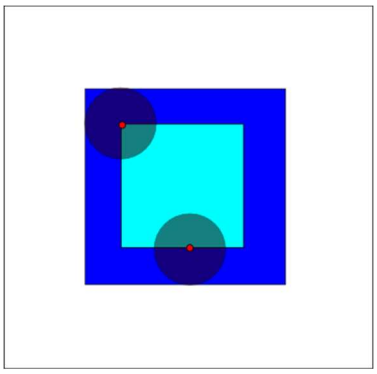
This operation performs a morphological erosion of an image and places the result in the output.

$$Result = Image \ominus K$$

where:

- $\ominus$ - denotes the erosion operation
- K - denotes the structuring element (kernel)

The erosion of an Object by a structuring element K can be understood as the locus of the points reached by the center of K when K moves inside the Object. The erosion of a square of size = 10 by a disk of radius = 2 is a square of side = 6.



The erosion of the dark-blue square by a disk, resulting in the light-blue square.

Parameters:

- 1.1 Image - the input image
- 1.2 Kernel radius (px) - determines the size of the processing window (default = 0) using the formulas:

$$Kernel\ width = 2 * \boldsymbol{Kernel\ radius(px)} + 1$$

$$Kernel\ height = 2 * \boldsymbol{Kernel\ radius(px)} + 1$$

- 1.3 Kernel shape - specifies the shape of the kernel (default = Disk)
- 1.4 Result of "…" - the result of the operation

Effect:

Performs morphological erosion to deflate the shapes contained in the image.

Example:

- Input:

Two cases are considered:

- case 1: the image is a mask image (black & white, 8-bit, 1-channel)
- case 2: the image is a fluorescence grayscale image (8-bit, 1-channel) representing the DAPI channel of an UV irradiated skin sample

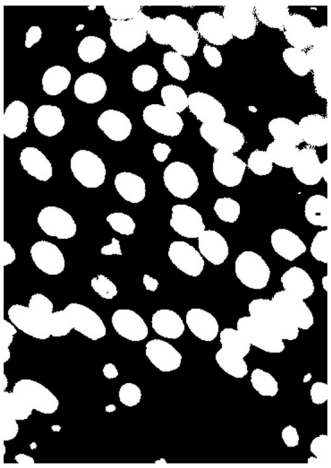


Figure 312 – Mask image (case 1)

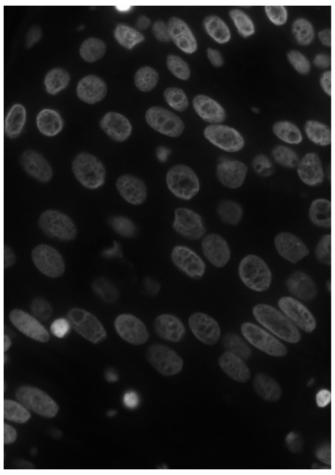


Figure 313 – UV irradiated skin sample - DAPI channel (case 2)

- Engine settings:

The engine settings used for the 2 examples are presented in case 1 and case 2.

The engine's result is a mask image (black & white, 8-bit, 1-channel) for case 1 and a grayscale image (8-bit, 1-channel) for case 2. In each case, the output matches the input.

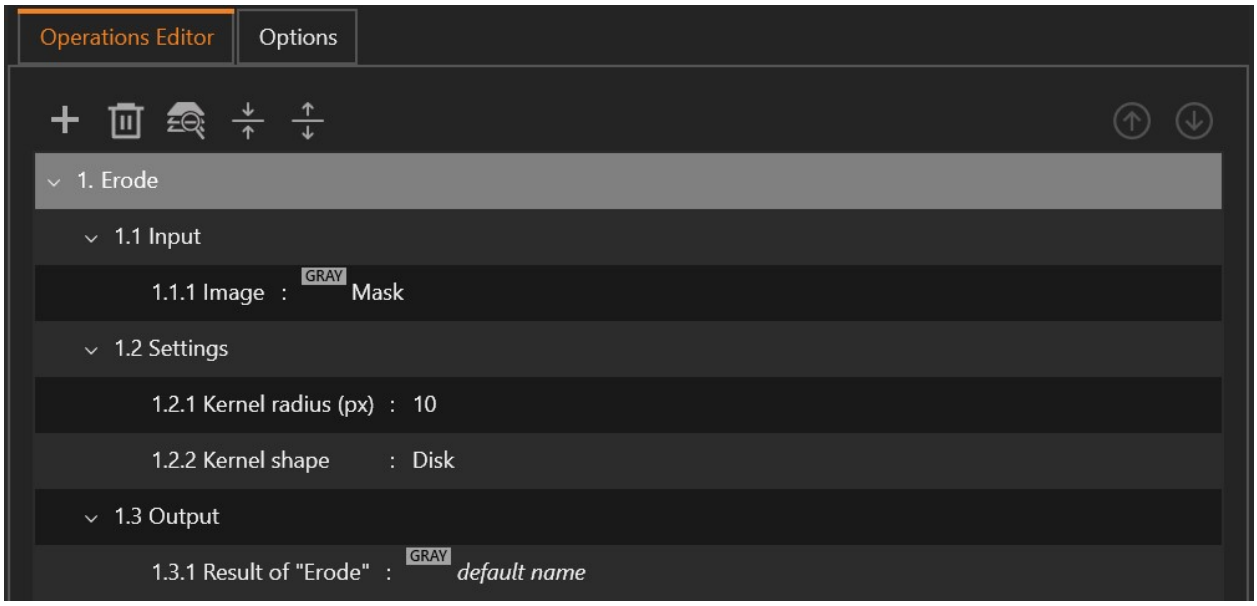


Figure 314 – Engine settings (case 1)

Figure 315 – Engine settings (case 2)

- Output:

First figure shows the result of a morphological close operation applied on the input mask image (case 1).

Second figure shows the result of a morphological close operation applied on the input grayscale image (case 2).

Figure 316 – Result of Erode engine (case 1)

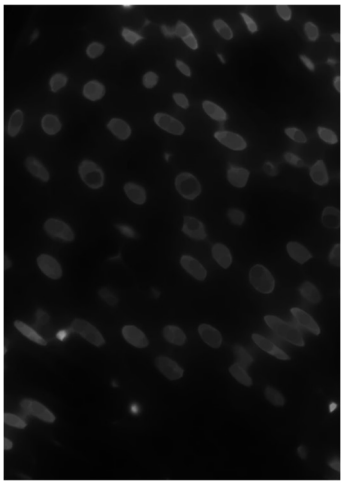


Figure 317 – Result of Erode engine (case 2)

Erode (with black border)

Figure 318 – Morphological - Erode Const0

Where it can be found:

Basic Operations Module ► Morphological ► Erode (with black border)

Description:

Erode is a morphological operation available in Basic Operations Module.

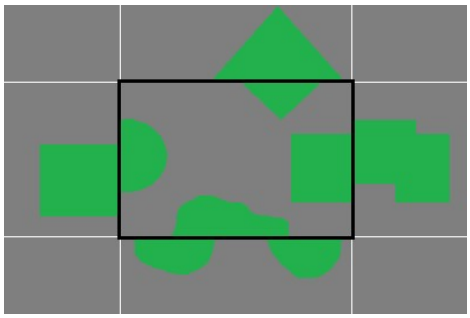
This operation performs a morphological erosion of an image and places the result in the output.

$$Result = Image \ominus K$$

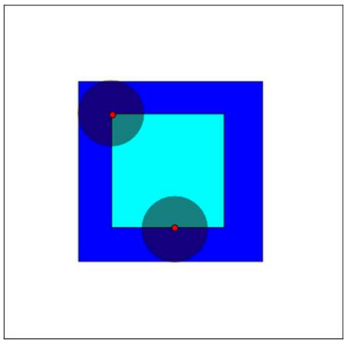
where:

- $\ominus$ - denotes the erosion operation
- K - denotes the structuring element (kernel)

Note	When the Border Input option is enabled, this engine has a similar effect as Erode (see 3.8.4).



The erosion of an Object by a structuring element K can be understood as the locus of the points reached by the center of K when K moves inside the Object. The erosion of a square of size = 10 by a disk of radius = 2 is a square of side = 6.



The erosion of the dark-blue square by a disk, resulting in the light-blue square.

Parameters:

- 1.1 Image - the input image
- 1.2 Kernel radius (px) - determines the size of processing window (default = 0) using the formulas:

$$Kernel\ width = 2 * \boldsymbol{Kernel\ radius(px)} + 1$$

$$Kernel\ height = 2 * \boldsymbol{Kernel\ radius(px)} + 1$$

- 1.3 Kernel shape - specifies the shape of the kernel (default = Disk)
- 1.4 Result of "…" - the result of the operation

Effect:

Performs morphological erosion to deflate the shapes contained in the image.

Example:

This engine has the same effect as the Erode engine with the border option disabled.

Fill holes

Figure 319 – Morphological BWFillHoles

Where it can be found:

Basic Operations Module ► Morphological ► Fill holes

Description:

Fill holes is a morphological operation available in Basic Operations Module.

This operation fills holes in the binary image and places the result in the output.

A hole is a set of background pixels that cannot be reached by filling in the background.

Parameters:

- 1.1 Image - the input image
- 1.2 Allow fill on border - enables / disables filling of holes touching image border
- 1.3 Result of "…" - the result of the operation

Effect:

Fills holes in a binary image.

Example:

- Input:

Images are mask images (black & white, 8-bit, 1-channel). The masks are full white (e.g. may represent a small interior region from a tissue detection output).

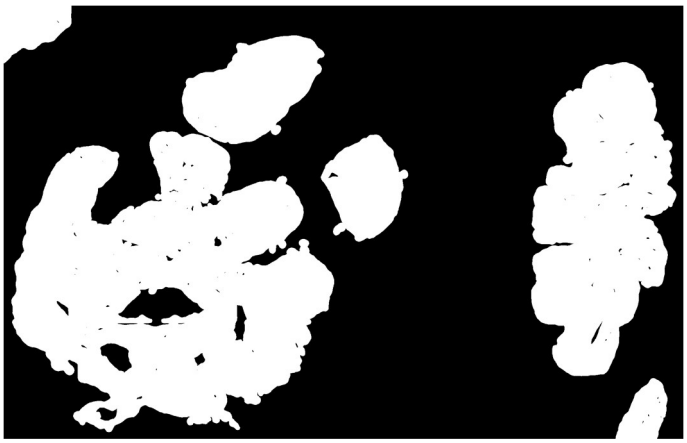


Figure 320 – Mask image

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a mask image (black & white, 8-bit, 1-channel), matching the input.

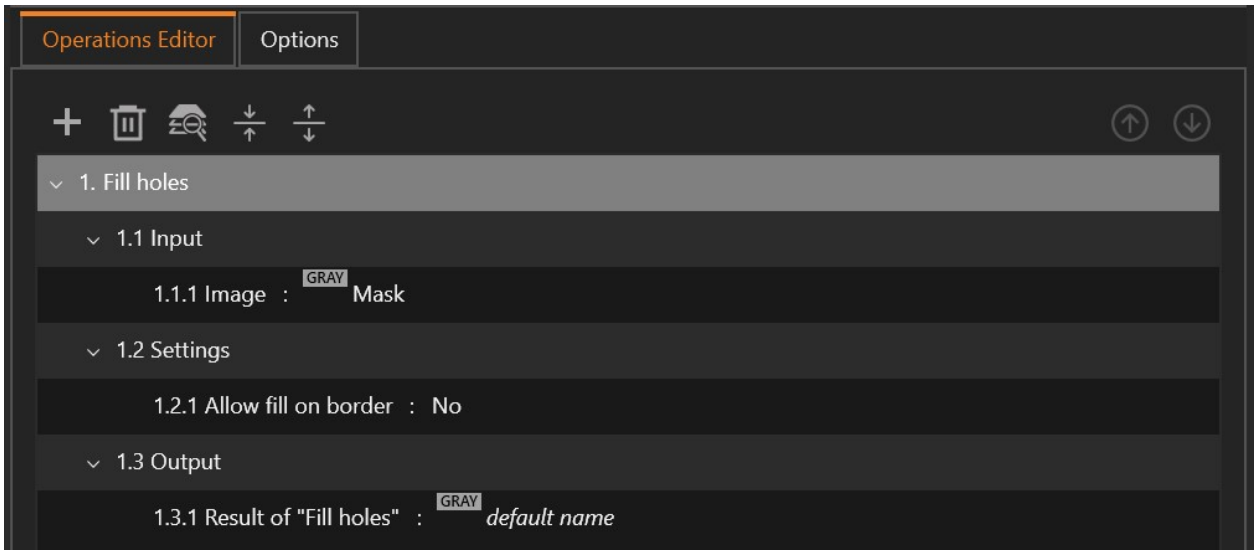


Figure 321 – Engine settings

- Output:

Figure below shows the results, where the structures look compact, without holes.

Figure 322 – Result Fill Holes engine

Open (erode, dilate)

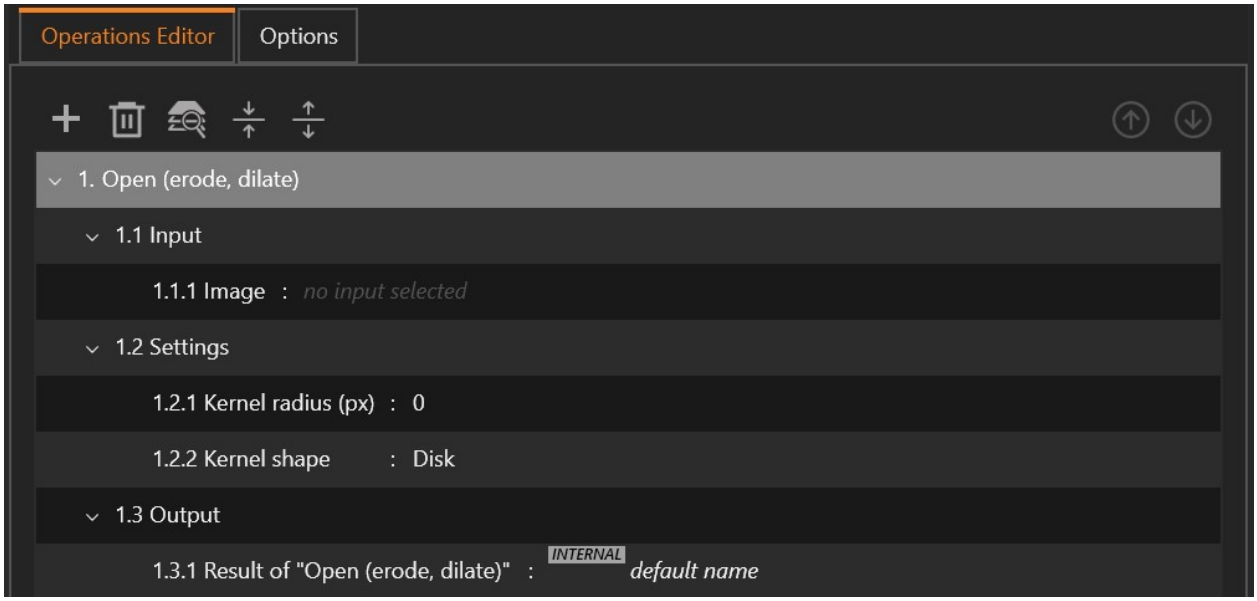


Figure 323 – Morphological - Open (erode, dilate)

Where it can be found:

Basic Operations Module ► Morphological ► Open (erode, dilate)

Description:

Open (erode, dilate) is a morphological operation available in Basic Operations Module.

This operation performs a morphological opening of an image and places the result in the output.

Morphological Open of an image represents the succession of erosion and dilation of the image, using the same structuring element (kernel).

$$Result = Image \circ K = (Image \ominus K) \oplus K$$

where:

- $\circ$ - denotes the open operation
- $\ominus$ - denotes the erosion operation
- $\oplus$ - denotes the dilation operation
- K - denotes the structuring element (kernel)

Parameters:

- 1.1 Image - the input image
- 1.2 Kernel radius (px) - determines the size of processing window (default = 0) using the formulas:

$$Kernel\ width = 2 * \boldsymbol{Kernel\ radius(px)} + 1$$

$$Kernel\ height = 2 * \boldsymbol{Kernel\ radius(px)} + 1$$

- 1.3 Kernel shape - specifies the shape of the kernel (default = Disk)
- 1.4 Result of "…" - the result of the operation

Effect:

Performs morphological open of an image to remove noise, remove small objects from the foreground, etc.

Example:

- Input:

Two cases are considered:

- case 1: Image is a mask image (black & white, 8-bit, 1-channel)
- case 2: Image is a fluorescence grayscale image (8-bit, 1-channel) representing the DAPI channel of an UV irradiated skin sample

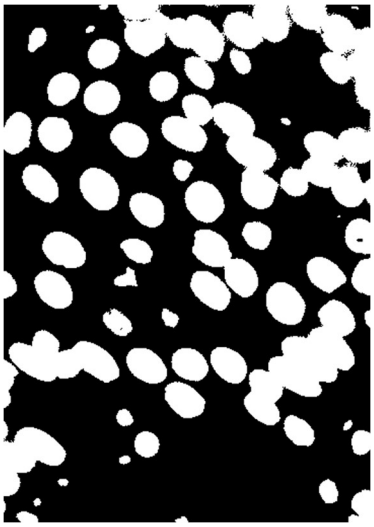


Figure 324 – Mask image (case 1)

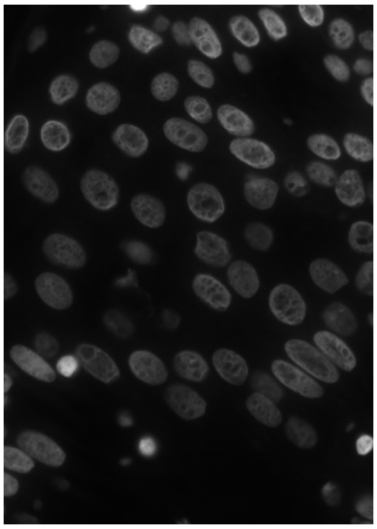


Figure 325 – UV irradiated skin sample - DAPI channel (case 2)

- Engine settings:

The engine settings used for the 2 examples are presented in case 1 and case 2.

The engine's result is a mask image (black & white, 8-bit, 1-channel) for case 1, respectively a grayscale image (8-bit, 1-channel) for case 2. In each case, the output matches the input.

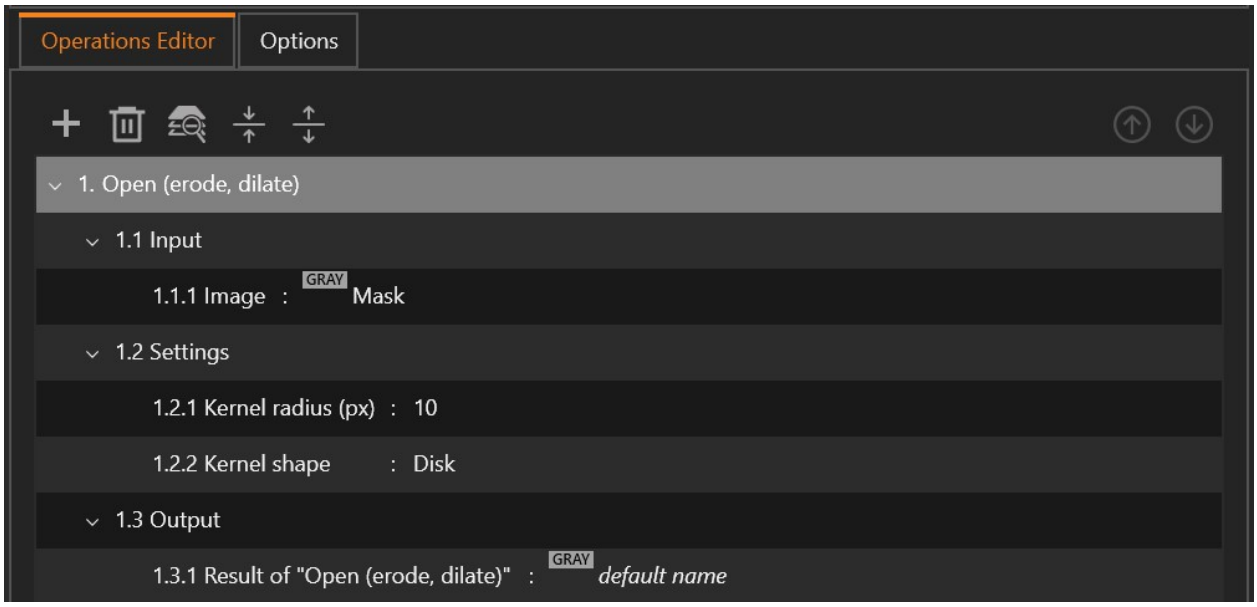


Figure 326 – Engine settings (case 1)

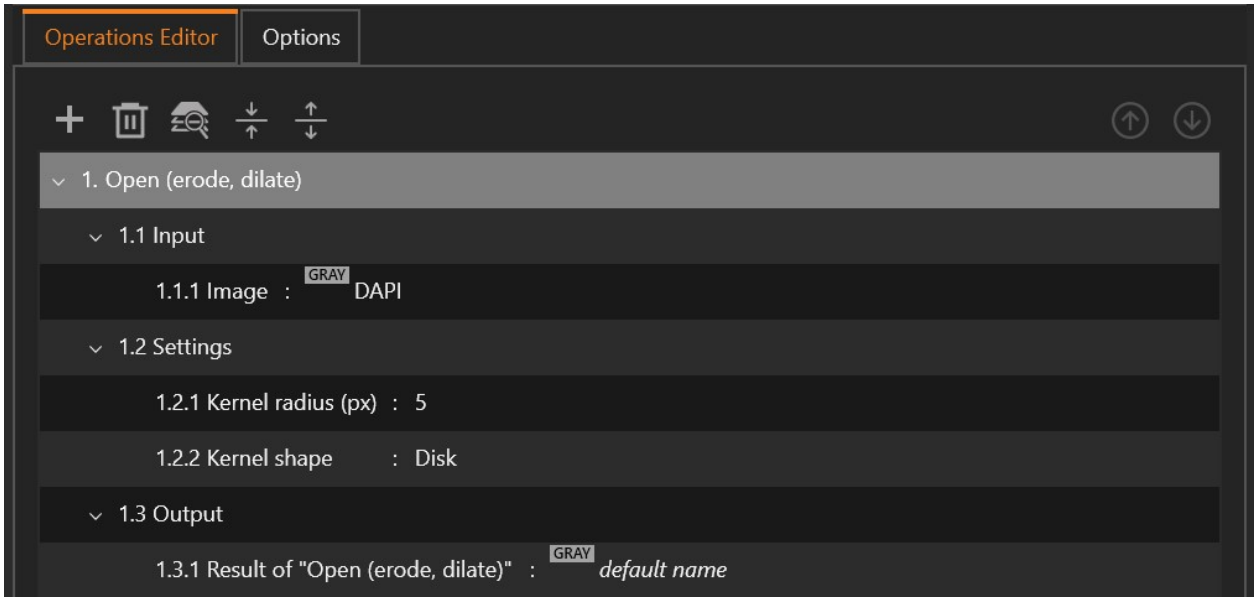


Figure 327 – Engine settings (case 2)

- Output:

First figure shows the result of a morphological open operation applied on the input mask image (case 1).

Second figure shows the result of a morphological open operation applied on the input grayscale image (case 2).

Figure 328 – Result of Open (erode, dilate) engine (case 1)

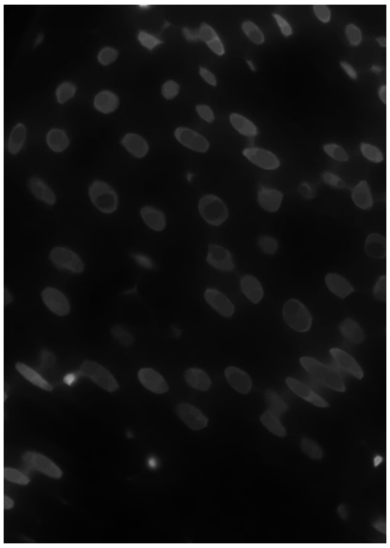


Figure 329 – Result of Open (erode, dilate) engine (case 2)

Remove small objects (from mask)

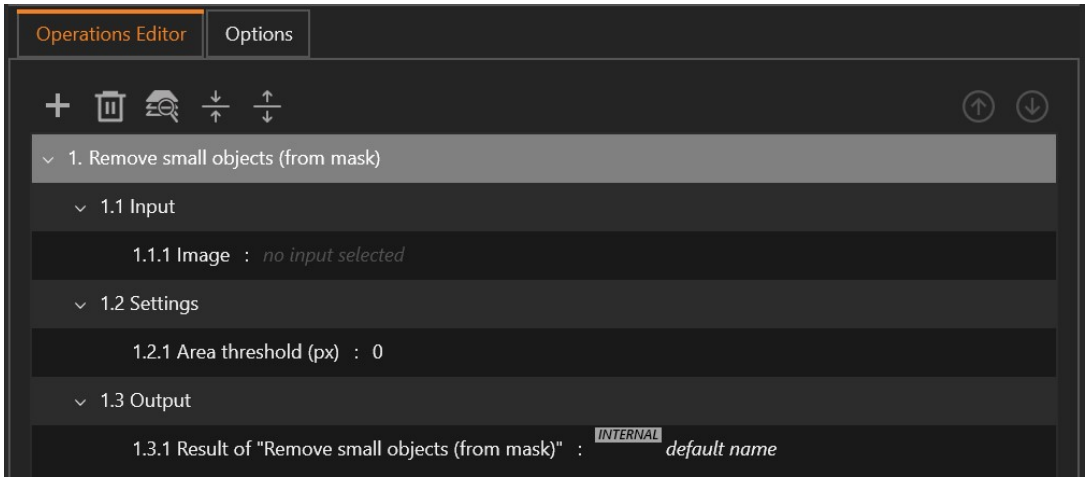


Figure 330 – Morphological BWArea Open

Where it can be found:

Basic Operations Module ► Morphological ► Remove small objects (from mask)

Description:

Remove small objects (from mask) is a morphological operation available in Basic Operations Module.

This operation removes all connected components (objects) that have fewer pixels than threshold value from a binary image (mask) and places the result in the output. This operation is known as an area opening.

Parameters:

- 1.1 Image - the input image
- 1.2 Area threshold (px) - maximum number of pixels defining small objects which must be removed (default = 0)
- 1.3 Result of "…" - the result of the operation

Effect:

Removes small objects from a binary image.

Example:

- Input:

The image is a mask image (black & white, 8-bit, 1-channel) representing the nuclei marker mask, generated by applying a threshold on DAPI channel.

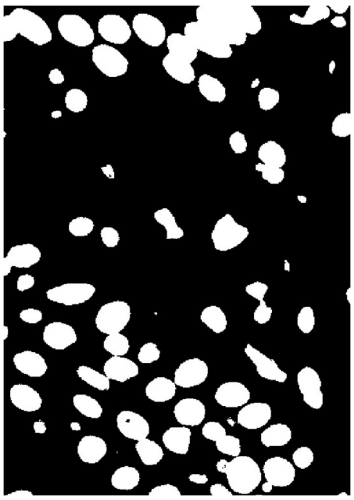


Figure 331 – Mask image

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a mask image (black & white, 8-bit, 1-channel), matching the input.

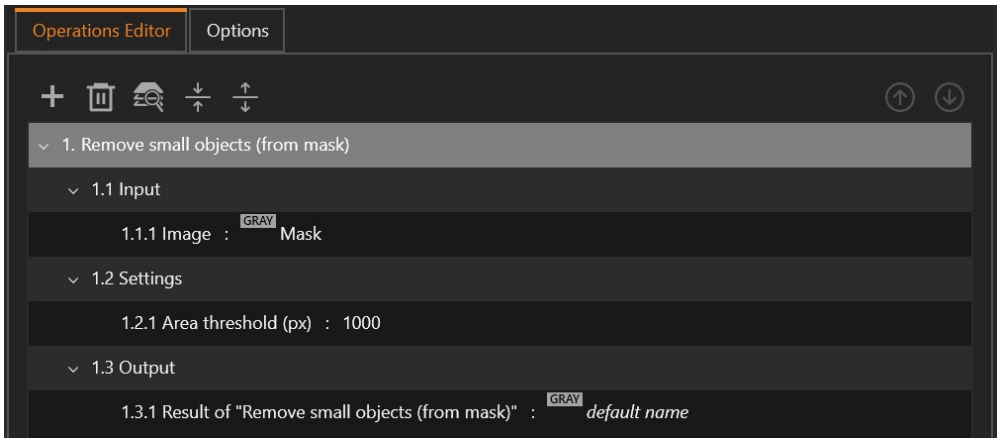


Figure 332 – Engine settings

- Output:

Figure below shows the object remained after the removal of objects having an area smaller than 1000 pixels.

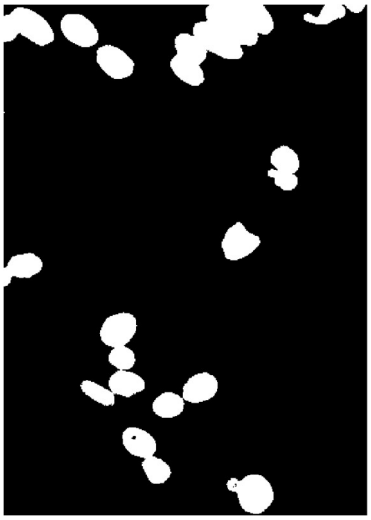


Figure 333 – Result of Remove small objects (from mask) engine

Skeleton

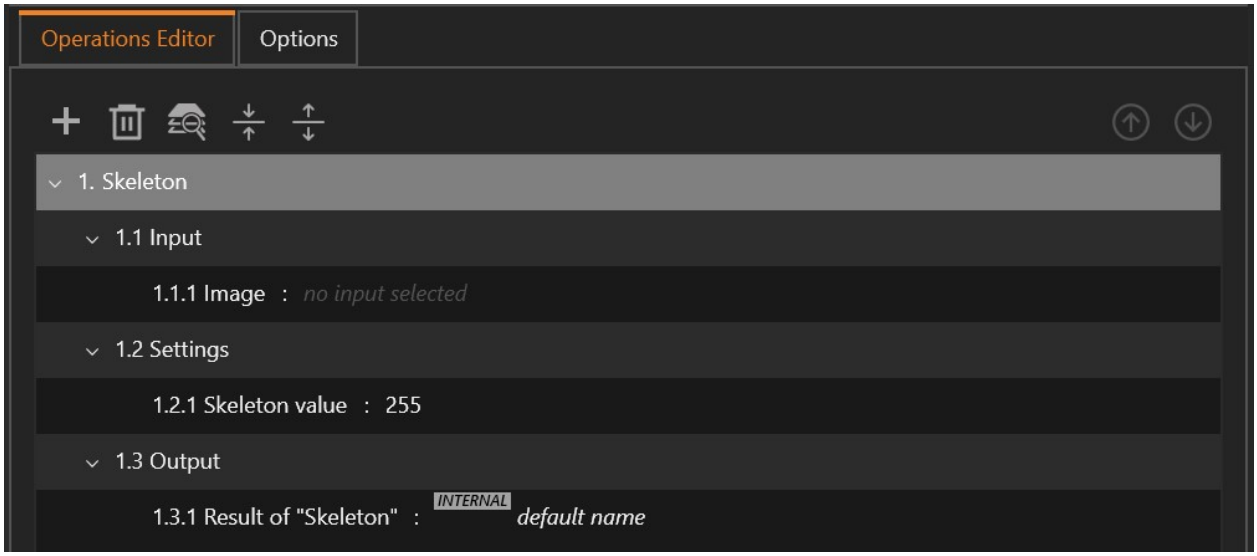


Figure 334 – Morphological - Skeleton

Where it can be found:

Basic Operations Module ► Morphological ► Skeleton

Description:

Skeleton is a morphological operation available in Basic Operations Module.

This operation skeletonizes all objects in the image and places the result in the output.

The skeleton is the reduction of an object in a binary image to a 1-pixel wide curved line, without changing the essential structure of the image. The skeletonization process extracts the centerline while preserving the topology and Euler number of the objects.

Parameters:

- 1.1 Image - the input image
- 1.2 Skeleton value - the value used to represent the skeleton: 1 or 255 (default = 255)
- 1.3 Result of "…" - the result of the operation

Effect:

Skeletonization of a binary image.

Example:

- Input:

Image is mask image (black & white, 8-bit, 1-channel) representing the membrane area detected using the information from image presented first figure. The color image shows an immunohistochemical small region of a fat cells sample.

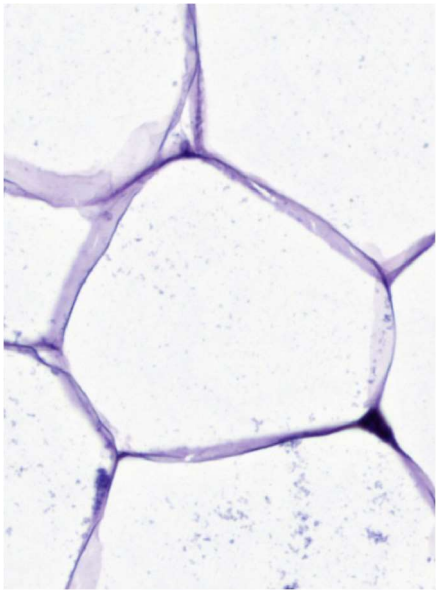


Figure 335 – Fat cells sample

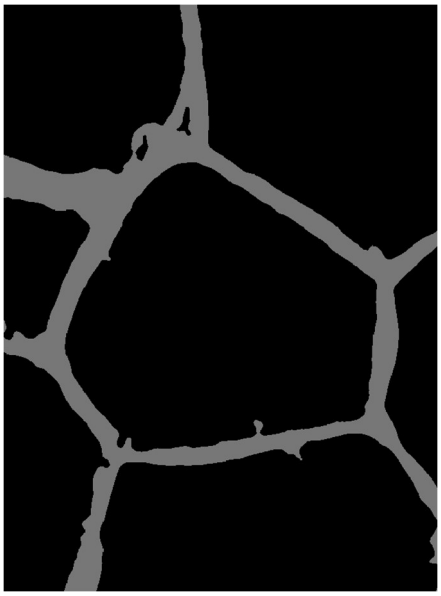


Figure 336 – Mask image

- Engine settings:

Figure below shows the engine settings used for this example.

The value used to generate the skeleton is set to 255.

The engine's result is a mask image (8-bit, 1-channel), matching the input.

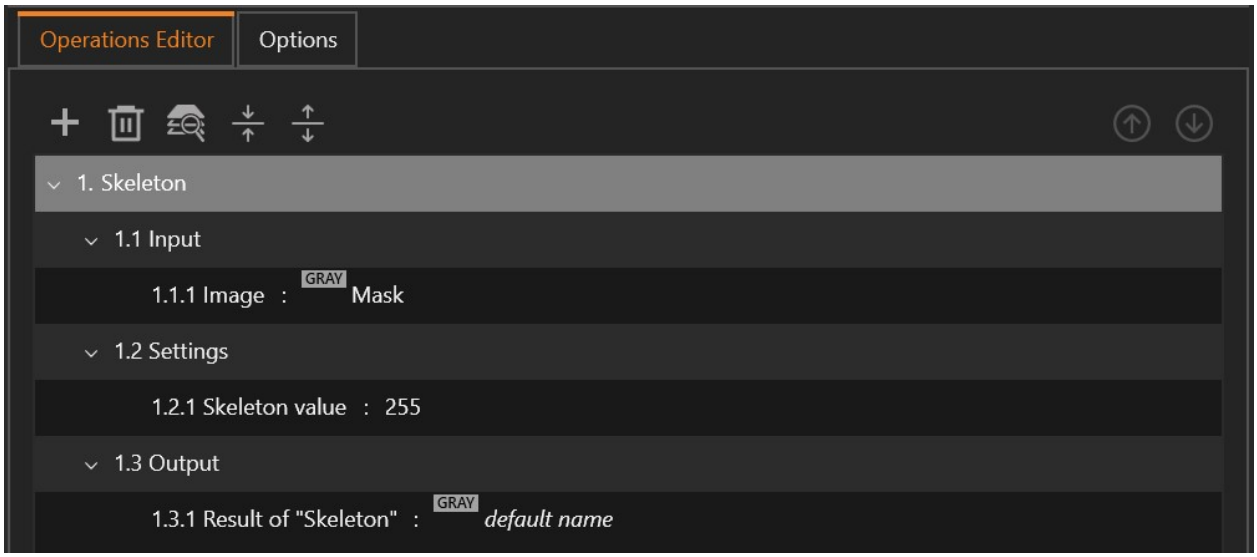


Figure 337 – Engine settings

- Output:

Figure below shows the resulting skeleton (highlighted in white), overlaid on the input mask (highlighted in gray).

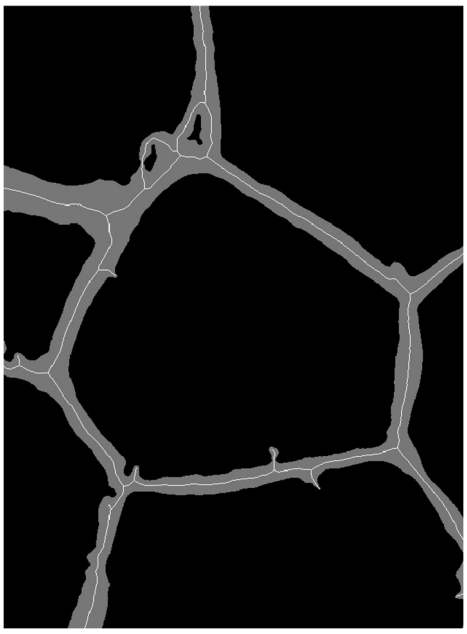


Figure 338 – Result of Skeleton engine

Skeleton - branch points

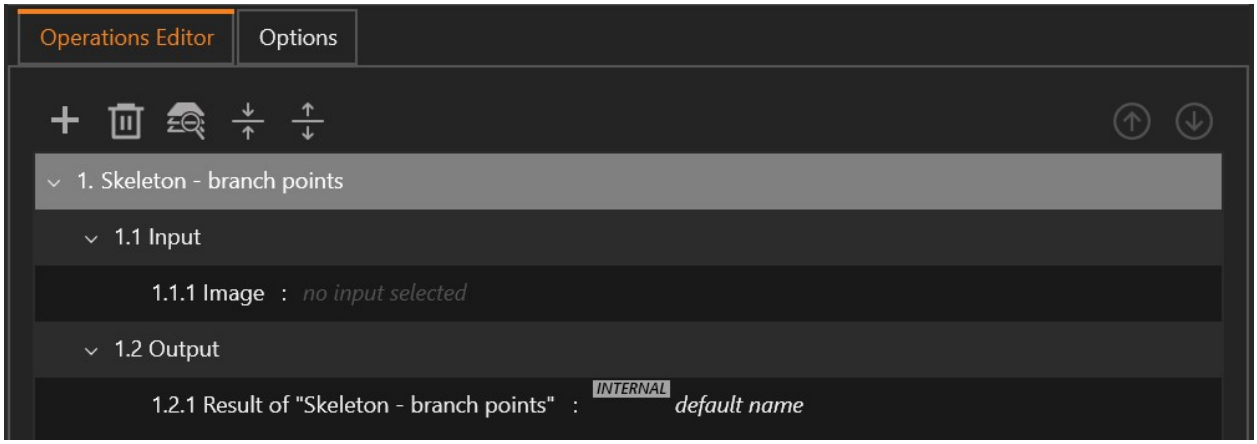


Figure 339 – Branch Points

Where it can be found:

Basic Operations Module ► Morphological ► Skeleton -branch points

Description:

Skeleton - branch points is a morphological operation available in Basic Operations Module.

This operation finds the intersection points in a skeleton image and places the result in the output.

Parameters:

- 1.1 Image - the input image
- 1.2 Result of "…" - the result of the operation

Effect:

Finds the branching points of a skeleton.

Example:

- Input:

Input Image is mask image (black & white, 8-bit, 1-channel), representing a skeleton structure.

Second figure shows the skeleton (highlighted in white) of the membrane area mask (highlighted in gray). The mask was generated using the information from the immunohistochemical small region of a fat cells sample presented in the first figure.

Figure 340 – Fat cells sample

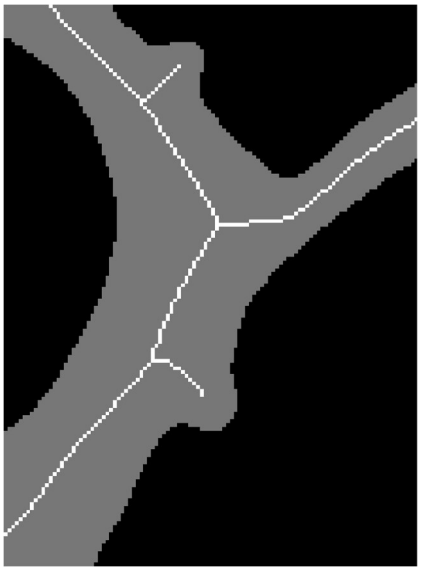


Figure 341 – Skeleton mask (white)

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a mask image (8bit, 1-channel), matching the input.

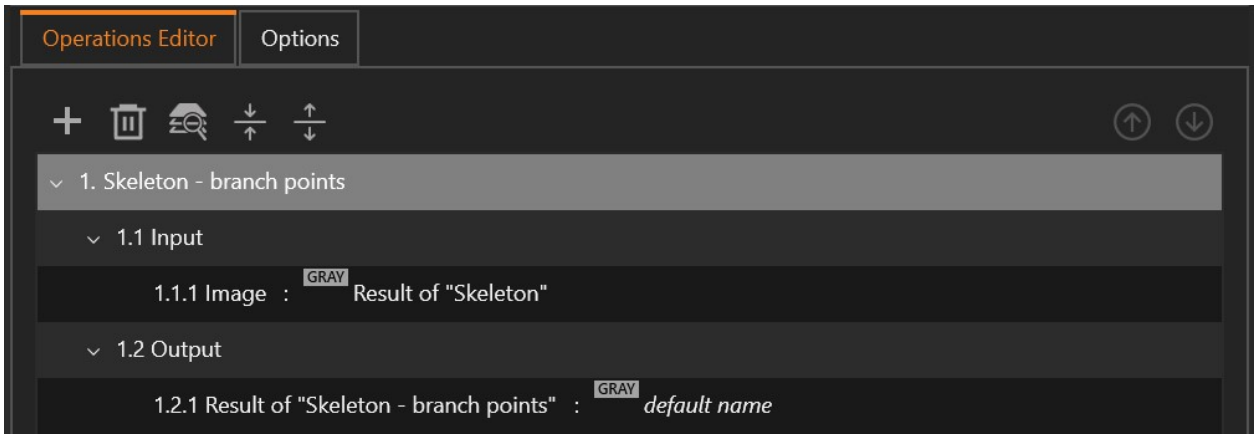


Figure 342 – Engine settings

- Output:

Figure below shows the identified branch points of the skeleton. The branch points (highlighted in red) were inflated (see Dilate) for better visualization.

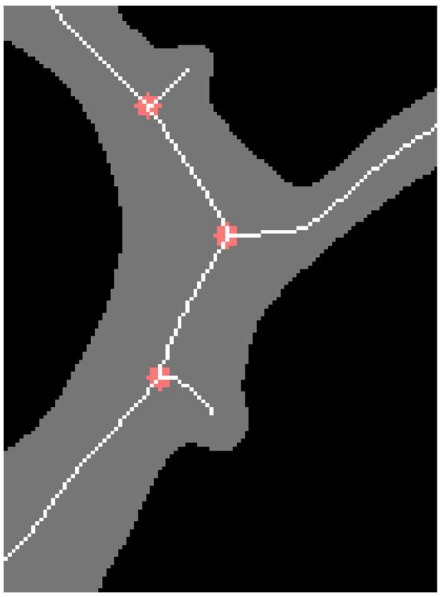


Figure 343 – Result of Skeleton - branch points engine

Skeleton - end points

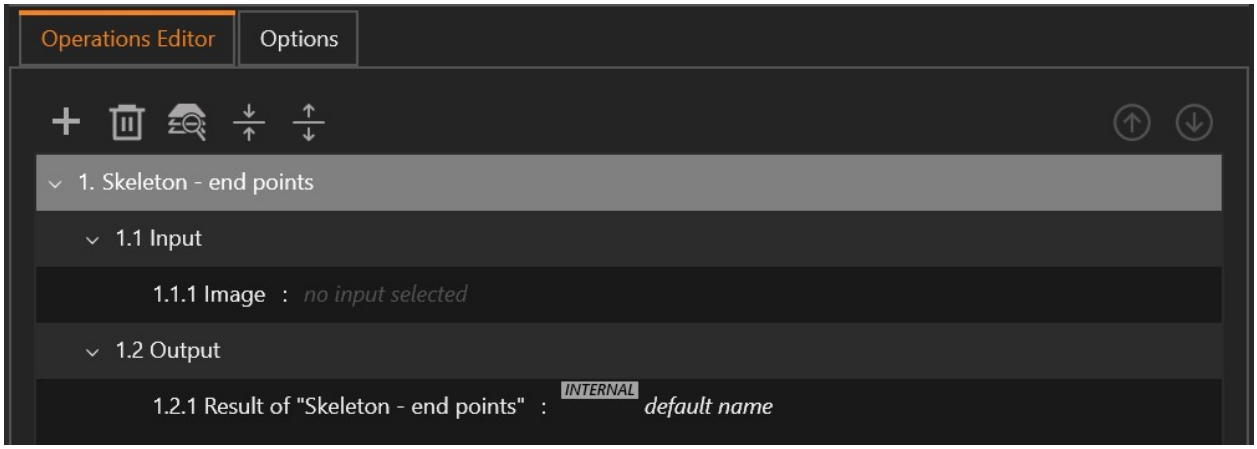


Figure 344 – End points

Where it can be found:

Basic Operations Module ► Morphological ► Skeleton -end points

Description:

Skeleton -end points is a morphological operation available in Basic Operations Module.

This operation finds the end points in a skeleton image and places the result in the output.

Parameters:

- 1.1 Image - the input image
- 1.2 Result of "…" - the result of the operation

Effect:

Finds the end points of a skeleton.

Example:

- Input:

The input Image is mask image (black & white, 8-bit, 1-channel), representing a skeleton structure.

Second figure shows the skeleton (highlighted in white) of the membrane area mask (highlighted in gray). The mask was generated using the information from the immunohistochemical small region of a fat cells sample presented in first figure.

Figure 345 – Fat cells sample

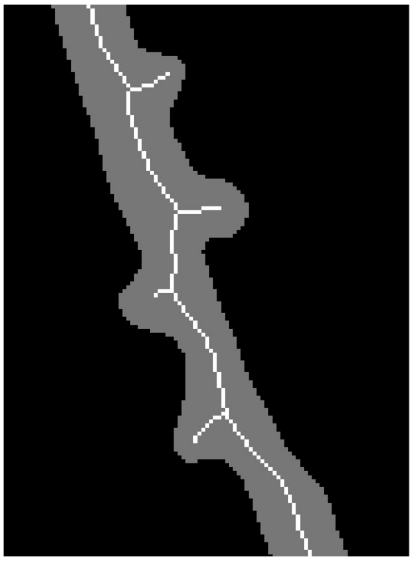


Figure 346 – Skeleton mask (white)

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a mask image (8bit, 1-channel), matching the input.

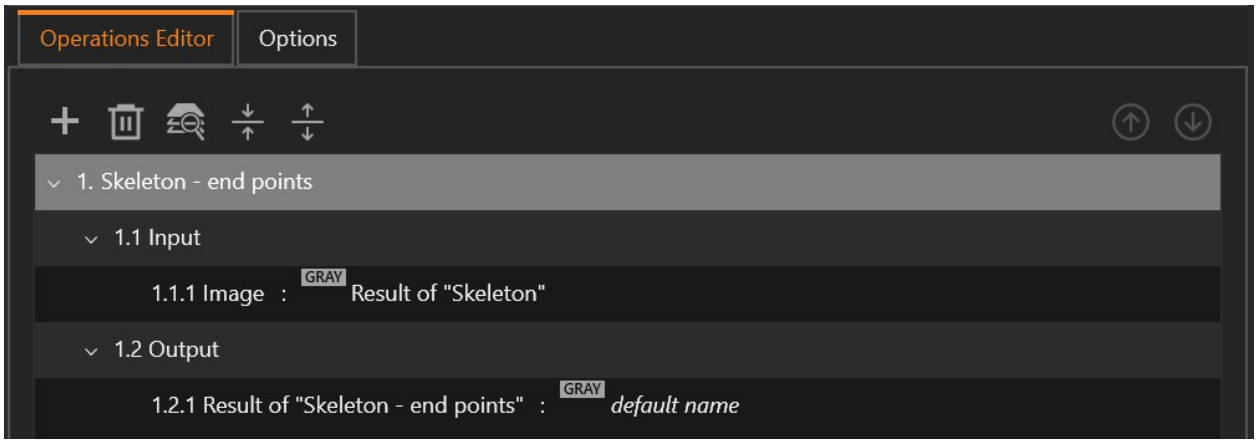


Figure 347 – Engine settings

- Output:

Figure below shows the identified end points of the skeleton. The end points (highlighted in red) were inflated (see Dilate) for better visualization.

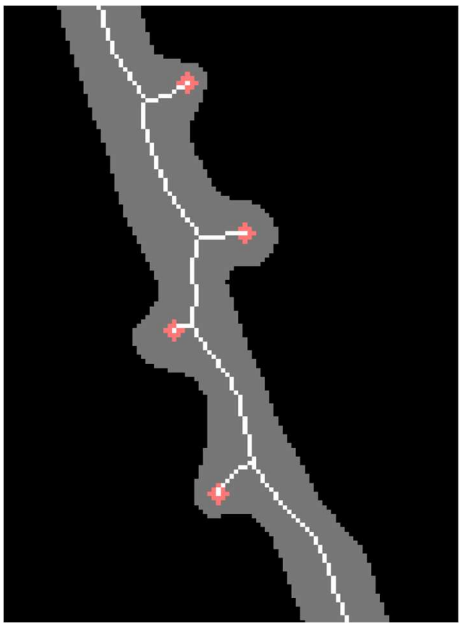


Figure 348 – Result of Skeleton - end points engine

Skeleton - extend spurs

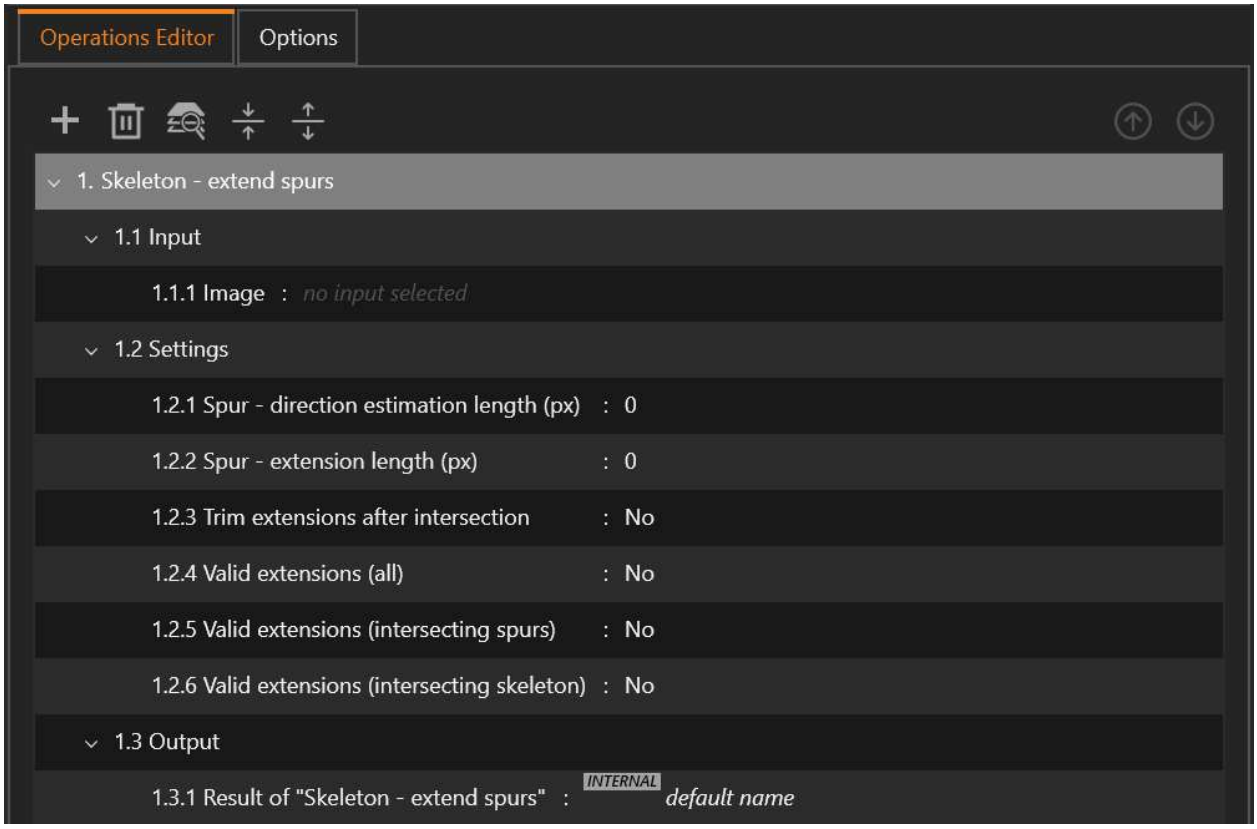


Figure 349 – Extended spurs V2

Where it can be found:

Basic Operations Module ► Morphological ► Skeleton -extend spurs

Description:

Skeleton - extend spurs is a morphological operation available in Basic Operations Module.

This operation extends the skeleton spurs and places the result in the output.

A spur is considered the set of connected pixels found between an end-point and the nearest connected branch-point.

Due to poor contrast, noise, etc., it may happen to have discontinuities in the resulting skeleton (e.g. grayscale image → binarization → skeletonization). In some cases, extending the spurs can correct the discontinuities.

Parameters:

- 1.1 Image - the input image;
- 1.2 Spur - direction estimation length (px) - the number of pixels (from end-point) used to estimate the direction of the spurs (default = 0)
- 1.3 Spur - extension length (px) - the pixel number representing the extension size (default = 0)
- 1.4 Trim extensions after intersection - enables / disables the removal of extensions beyond the intersection point (default = No).
- 1.5 Valid extensions (all) - enables / disables validation of all extensions, without restrictions (default = No)
- 1.6 Valid extensions (intersecting spurs) - enables / disables validation only for extensions intersecting other extensions (default = No)
- 1.7 Valid extensions (intersecting skeleton) - enables / disables validation only for extensions intersecting other skeleton elements (default = No)
- 1.8 Result of "…" - the result of the operation

Effect:

Extends skeleton spurs.

Example:

- Input:

Input Image is mask image (black & white, 8-bit, 1-channel), representing a skeleton structure.

Second figure shows the skeleton (highlighted in white) of the membrane area mask (highlighted in gray). The mask was generated using the information from the immunohistochemical small region of a fat cells sample presented in the first figure.

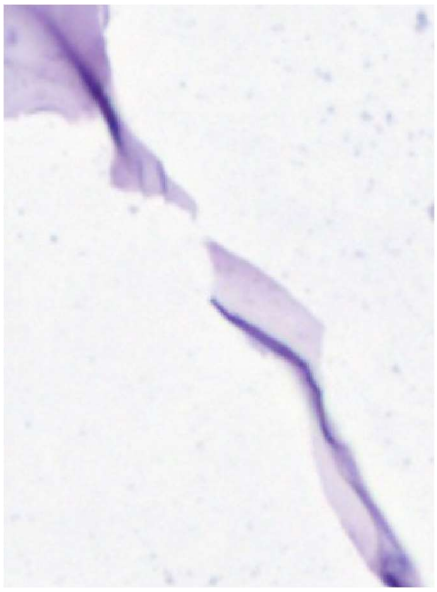


Figure 350 – Fat cells sample

Figure 351 – Skeleton mask (white)

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a mask image (8bit, 1-channel), matching the input.

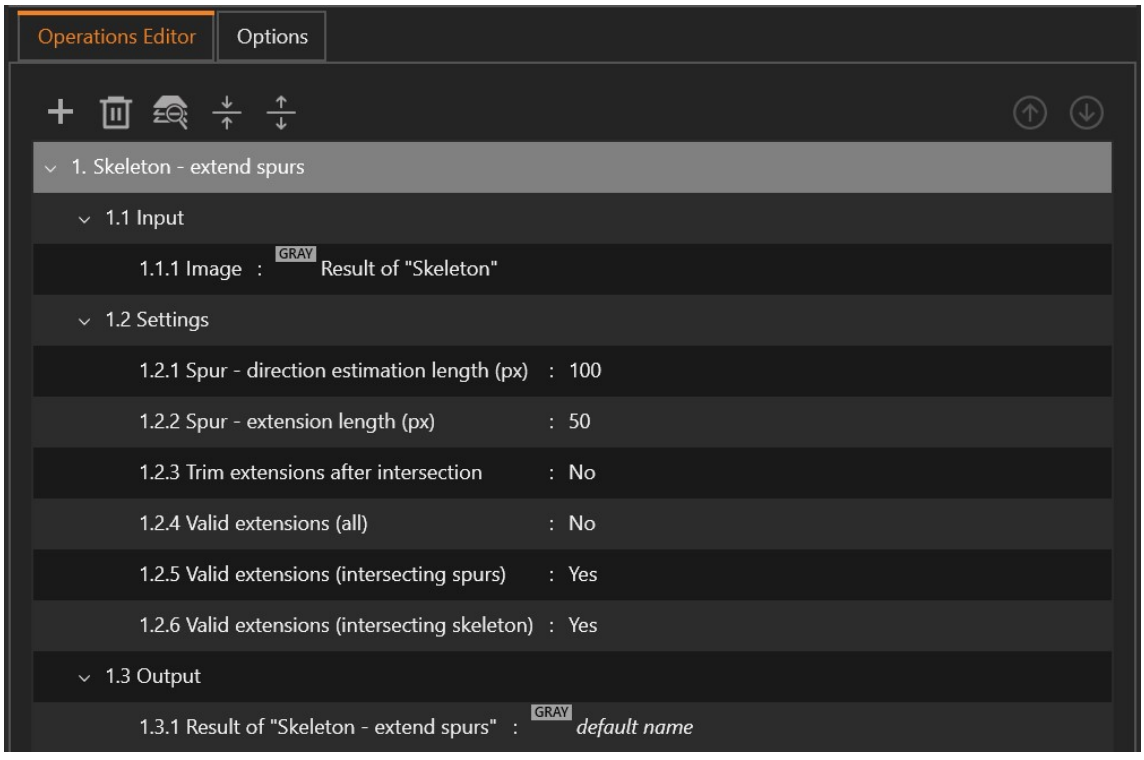


Figure 352 – Engine settings

- Output:

Figure below shows the result of skeleton spurs extension.

For all spurs, a 50-pixel extension is generated on the direction estimated by the last 100 pixels of the spur. Only the extensions which intersect another extension or the original skeleton are validated.

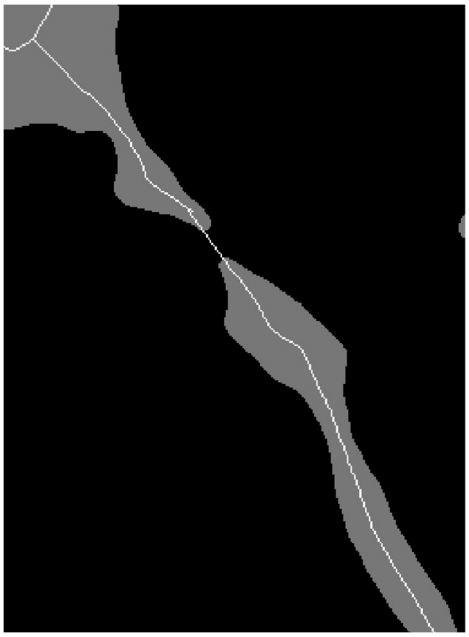


Figure 353 – Result of Skeleton - extend spurs engine

Skeleton - remove spurs

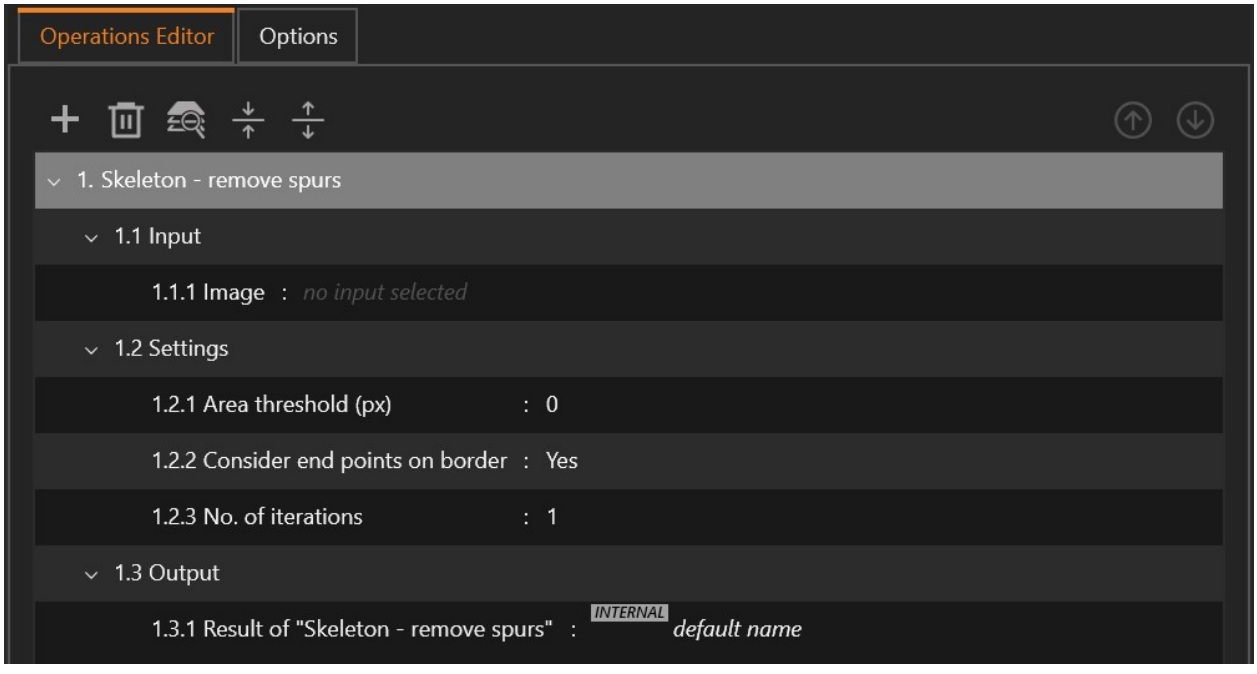


Figure 354 – Remove spurs

Where it can be found:

Basic Operations Module ► Morphological ► Skeleton -remove spurs

Description:

Skeleton -remove spurs is a morphological operation available in Basic Operations Module.

This operation cleans the skeleton in the image by removing spurs smaller than the threshold and places the result in the output. The process is iterative, meaning it can be run a successive number of times.

A spur is a set of connected pixels found between an end-point and the nearest connected branch-point.

Parameters:

- 1.1 Image - the input image;
- 1.2 Area threshold (px) - the area threshold for spurs. Spurs with an area smaller than the threshold will be deleted (default = 0).
- 1.3 Consider end points on border - enables / disables the possibility a spur touches the border. It can also be interpreted as the possibility the skeleton has continuity in the next FOV (default = Yes).
- 1.4 No. of iterations - specifies the number of run cycles (the number of times the operation is applied) (default = 1)
- 1.5 Result of "…" - the result of the operation

Effect:

Cleans the skeleton by removing small-sized spurs.

Example:

- Input:

Input Image is mask image (black & white, 8-bit, 1-channel), representing a skeleton structure.

Second figure shows the skeleton (highlighted in white) of the membrane area mask (highlighted in gray). The mask was generated using the information from the immunohistochemical small region of a fat cells sample presented in first figure.

Figure 355 – Fat cells sample

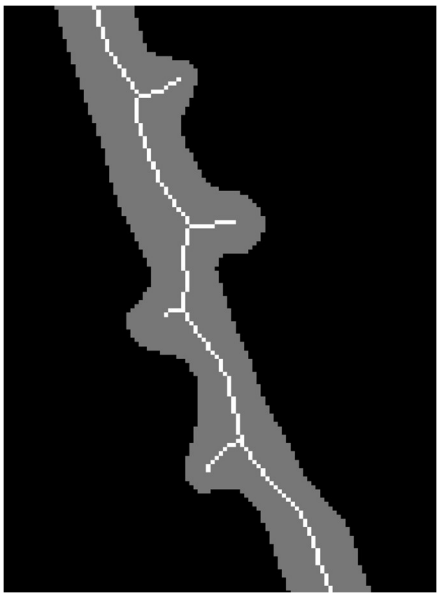


Figure 356 – Skeleton mask (white)

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a mask image (8bit, 1-channel), matching the input.

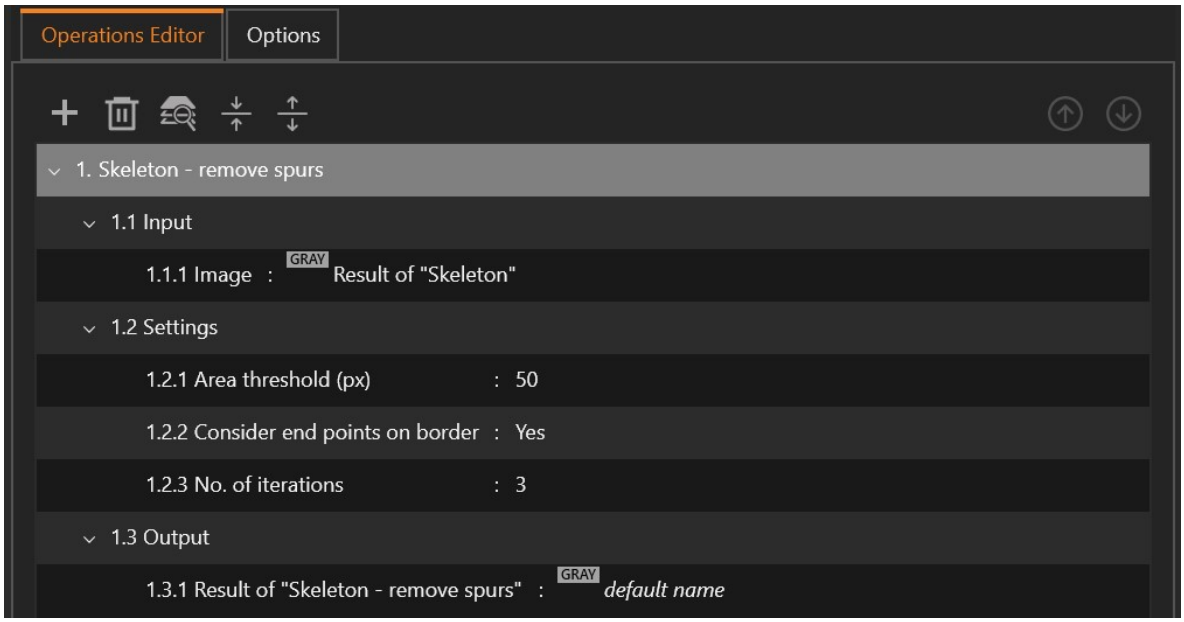


Figure 357 – Engine settings

- Output:

Figure below shows the result of skeleton spurs removal, by length.

All spurs shorter than 50 pixels have been removed. The operation has been run 3 times (1 iteration can generate new spurs).

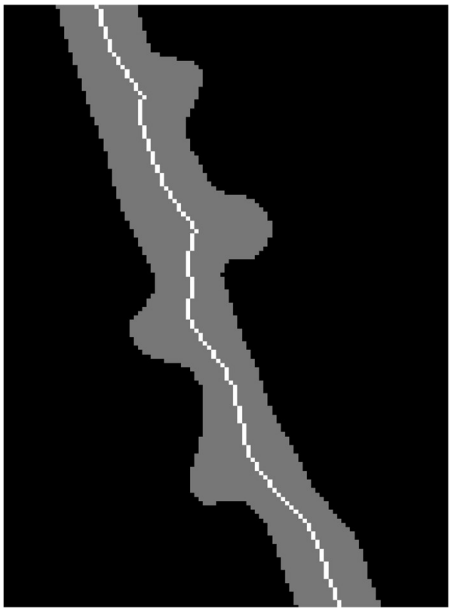


Figure 358 – Result of Skeleton - remove spurs engine

Skeleton - remove spurs (by angle)

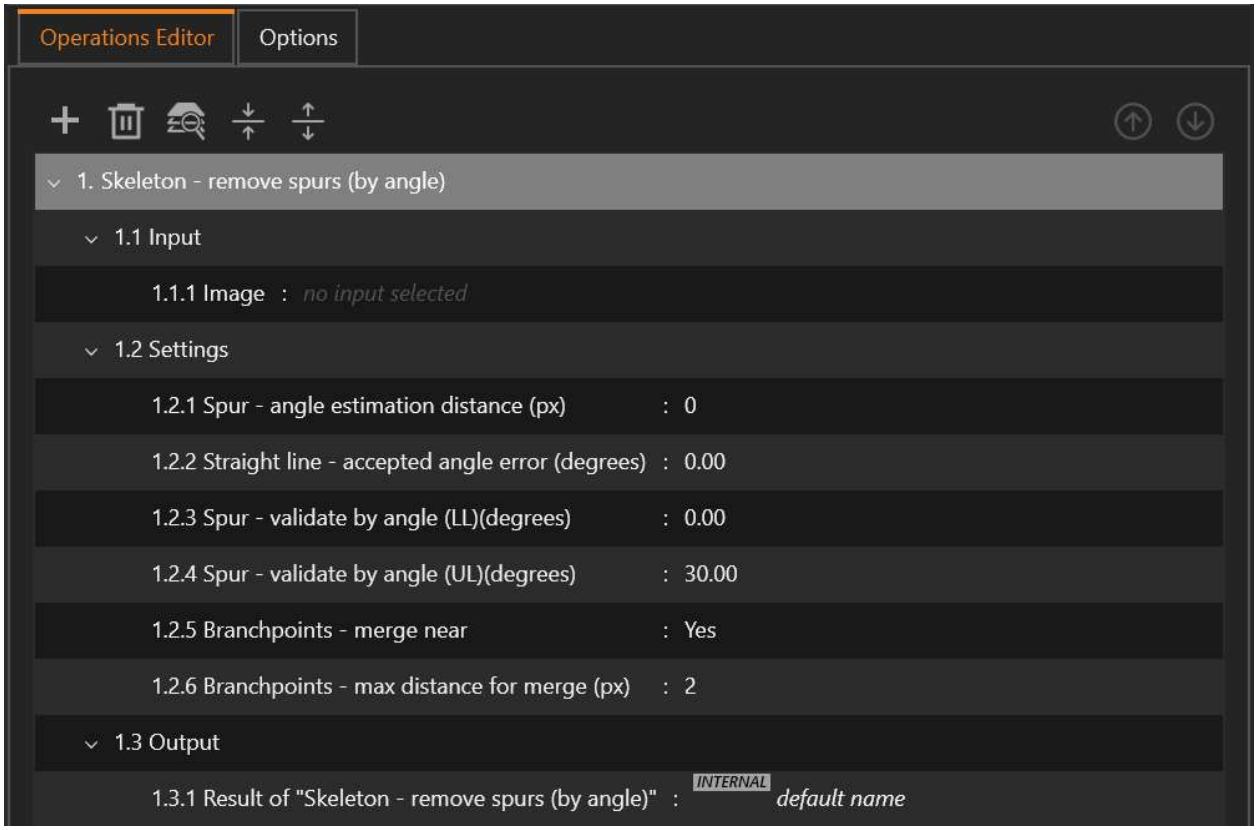


Figure 359 – Clean skeleton spurs by angle

Where it can be found:

Basic Operations Module ► Morphological ► Skeleton -remove spurs (by angle)

Description:

Skeleton - remove spurs (by angle) is a morphological operation available in Basic Operations Module.

This operation cleans the skeleton in the image by removing spurs based on the resulted angles and places the result in the output.

For each branch-point having a connected spur, the angles of all the intersecting skeleton elements are evaluated. If the angles made by the spur and with the rest of intersecting skeleton elements meet the defined conditions, the spur is removed.

Parameters:

- 1.1 Image - the input image

- 1.2 Spur -angle estimation distance (px) - the number of pixels (from a branch-point) used to estimate the angle of the intersecting skeleton element / spur (default = 0)
- 1.3 Straight line -accepted angle error (degrees) - the accepted angle error made by two intersecting skeleton elements to be considered a straight line (err. = 180º -angle) (default = 0)
- 1.4 Spur -validate by angle (LL) (degrees) - the lower limit for the angle made by a spur with an intersecting skeleton element (default = 0). Spurs with angles in the range [LL..UL] will not be removed
- 1.5 Spur -validate by angle (LL) (degrees) - the upper limit for the angle made by a spur with an intersecting skeleton element (default = 30). Spurs with angles in the range [LL..UL] will not be removed
- 1.6 Branchpoints -merge near - enables / disables the merging of close branch-points. When two branch-points are too close, they generate a skeleton element which is too short to correctly evaluate the angles around the branch-points (default = Yes)
- 1.7 Branch-points -max distance for merge (px) - the maximum distance allowed between two branch-points to be merged (default = 2).
- 1.8 Result of "…" - the result of the operation

Effect:

Cleans the skeleton by removing spurs by angles.

Example:

- Input:

The input Image is a mask image (black & white, 8-bit, 1-channel), representing a skeleton structure.

Second figure shows the skeleton (highlighted in white) of the membrane area mask (highlighted in gray). The mask was generated using the information from the immunohistochemical small region of a fat cells sample presented in first figure.

Figure 360 – Fat cells sample

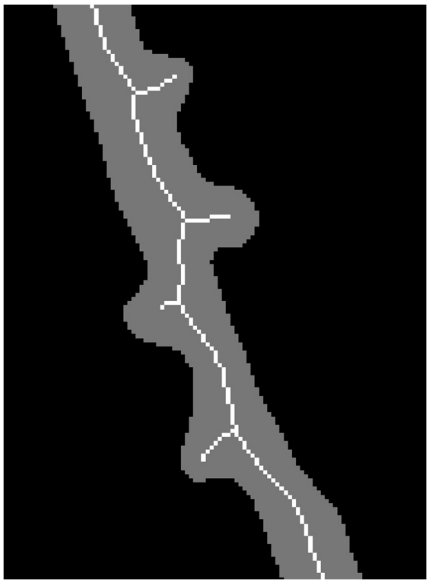


Figure 361 – Skeleton mask (white)

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a mask image (8bit, 1-channel), matching the input.

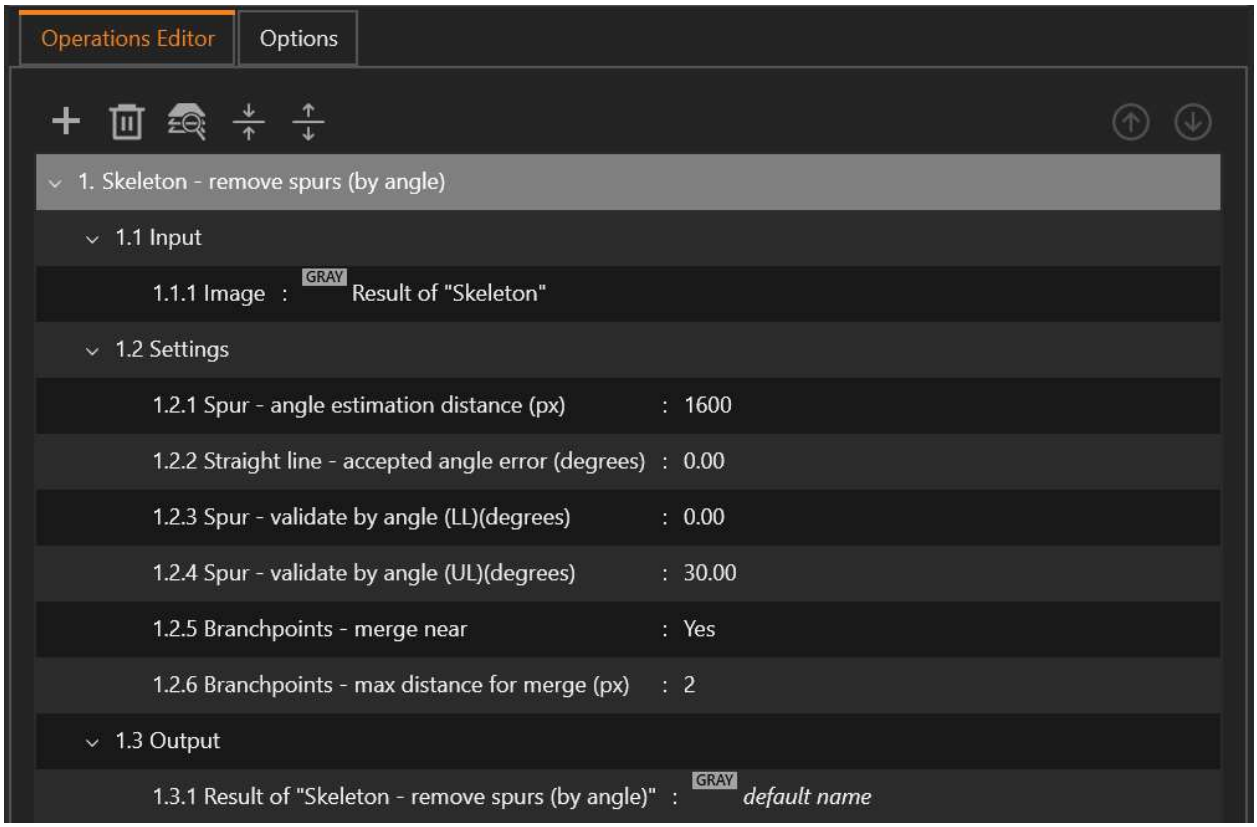


Figure 362 – Engine settings

- Output:

Figure below shows the result of skeleton spurs removal, by angle.

All spurs forming an angle between 0 and 30 degrees with the main skeleton segments, have been removed. The angles have been estimated using up to 1600 pixels from skeleton segments.

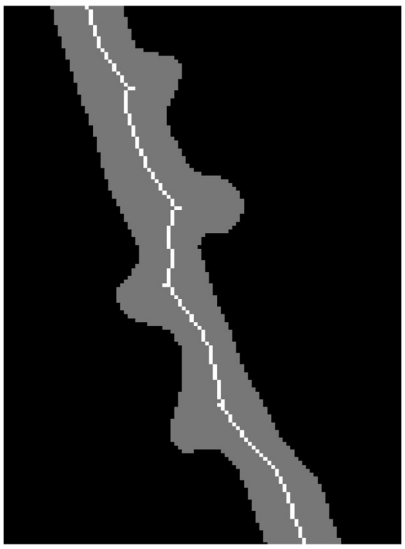


Figure 363 – Result of Skeleton - remove spurs (by angle) engine

Skeleton - shorten spurs

Figure 364 – Shorten spurs

Where it can be found:

Basic Operations Module ► Morphological ► Skeleton -shorten spurs

Description:

Skeleton -shorten spurs is a morphological operation available in Basic Operations Module.

This operation decreases the skeleton spurs length by cutting a defined number of pixels and places the result in the output.

A spur is a set of connected pixels found between an end-point and the nearest connected branch-point.

Parameters:

- 1.1 Image - the input image
- 1.2 Cut length (px) - the number of pixels to be cut from each spur (default = 5)
- 1.3 Consider end points on border - enables / disables the possibility a spur touches the border. It can also be interpreted as the possibility the skeleton has continuity in the next FOV (default = Yes)
- 1.4 Result of "..." - the result of the operation

Effect:

Shortens skeleton spurs.

Example:

- Input:

Input Image is mask image (black & white, 8-bit, 1-channel), representing a skeleton structure.

Second figure shows the skeleton (highlighted in white) of the membrane area mask (highlighted in gray). The mask was generated using the information from the immunohistochemical small region of a fat cells sample presented in first figure.

Figure 365 – Fat cells sample

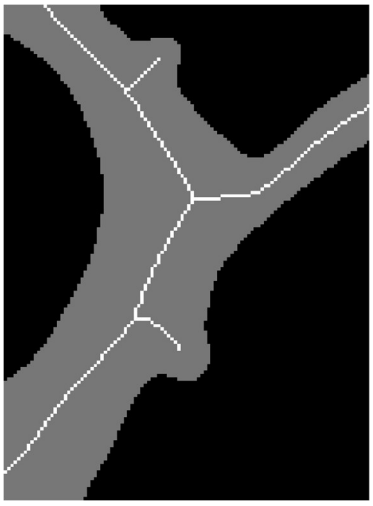


Figure 366 – Skeleton mask (white)

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a mask image (8bit, 1-channel), matching the input.

Figure 367 – Engine settings

- Output:

Figure below shows the shortened skeleton spurs. The resulting skeleton is highlighted in white and the removed parts of the spurs are highlighted in red.

All spurs have been shortened up to 30 pixels. The operation has been run 3 times (1 iteration can generate new spurs)

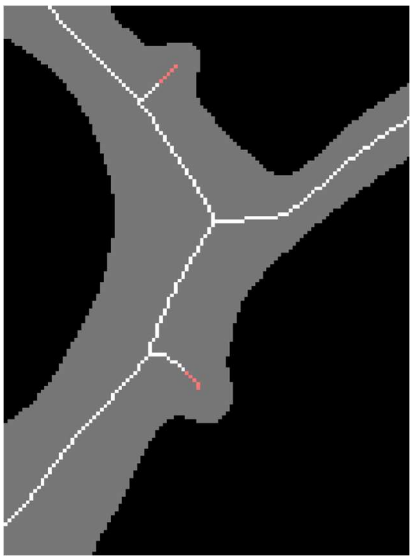


Figure 368 – Result of Skeleton - shorten spurs engine

Skeleton - split

Figure 369 – Split skeleton

Where it can be found:

Basic Operations Module ► Morphological ► Skeleton -split

Description:

Skeleton – split is a morphological operation available in Basic Operations Module.

This operation splits the skeleton an image into equal-size parts and places the result in the output.

Parameters:

- 1.1 Image - the input image
- 1.2 Split distance (px) - the length of each split part (default = 0)
- 1.3 Undersized rejection percent - specifies the minimum size percent allowed for a part to be considered valid (applicable for rests) (default = 0)
- 1.4 Result of "…" - the result of the operation

Effect:

Splits skeleton into equal-size parts

Example:

- Input:

The input Image is a mask image (black & white, 8-bit, 1-channel), representing a skeleton structure. Figure below shows the skeleton (highlighted in white) of the mask area (highlighted in dark gray).

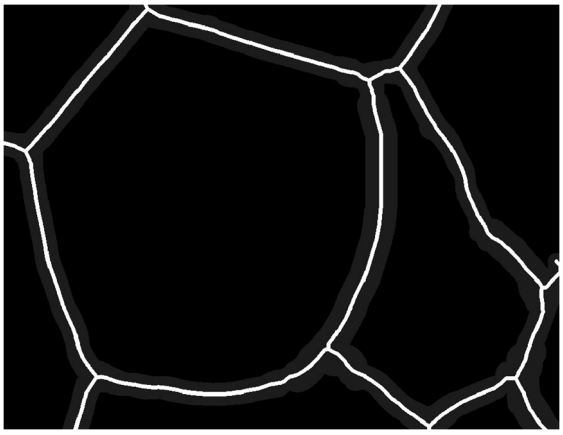


Figure 370 – Skeleton mask (white)

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a coded image (events). Each event represents a skeleton segment.

Figure 371 – Engine settings

- Output:

Figure below shows the resulting skeleton segments. Each segment is highlighted with a different color for better visualization.

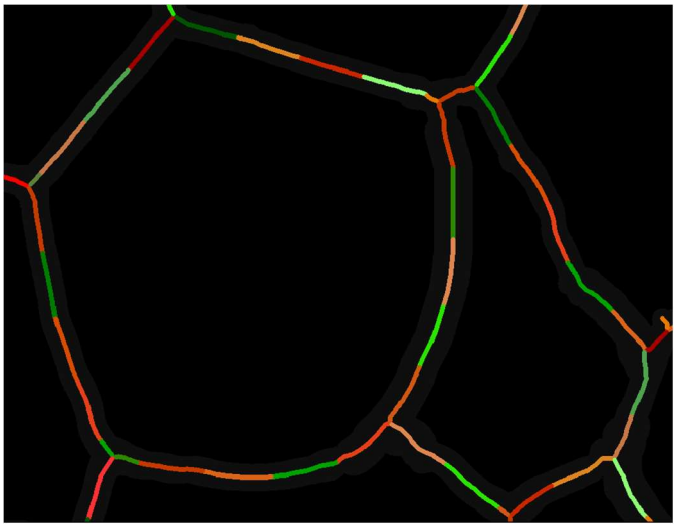


Figure 372 – Result of Skeleton - split engine

Top-hat

Figure 373 – Morph top hat border

Where it can be found:

Basic Operations Module ► Morphological ► Top-hat

Description:

Top-hat is a morphological operation available in Basic Operations Module.

This operation performs top-hat on an image and places the result in the output.

Top-hat is a morphological operation that extracts image details having the scale defined by the kernel size. The operation is defined as the difference between the image and the morphologically opened image (which can be interpreted as an "estimated" background).

$$\boldsymbol{Result = Image - Image \circ K}$$

where:

- K – denotes the kernel used to open the image (see Open (erode, dilate))
- ○ - denotes the opening operation

Parameters:

- 1.1 Image - the input image
- 1.2 Kernel radius (px) - determines the size of the processing window (default = 1) using the formulas:

$$Kernel\ width = 2 * \boldsymbol{Kernel\ radius(px)} + 1$$

$$Kernel\ height = 2 * \boldsymbol{Kernel\ radius(px)} + 1$$

- 1.3 Result of "…" - the result of the operation

Effect:

The Top-hat operation can be used for feature extraction, background equalization, image enhancements, etc.

Example:

- Input:

The image is fluorescence grayscale image (8-bit, 1-channel) representing the DAPI channel of a colorectal cancer sample.

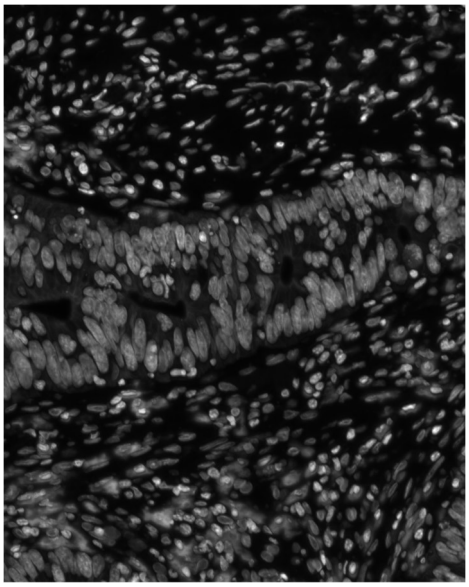


Figure 374 – Colorectal cancer sample - DAPI channel

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a grayscale image (8-bit, 1-channel), matching the input image.

Figure 375 – Engine settings

- Output:

Figure below shows the result of morphological top-hat engine. The effect is an attenuation of the high background.

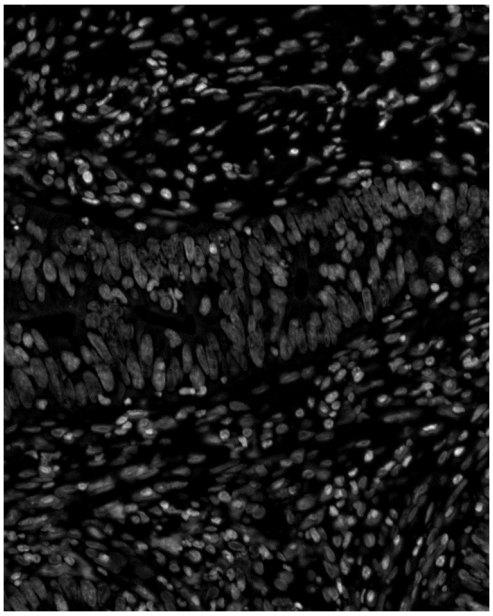


Figure 376 – Result of Top-hat engine

5.9. Post Processing

Remove labels (larger)

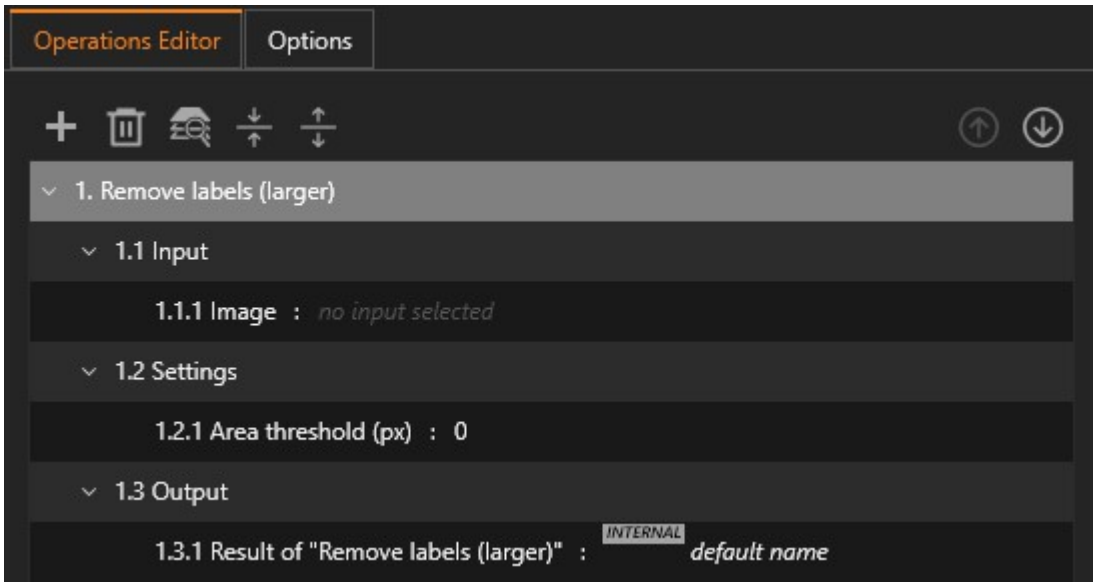


Figure 377 – Remove labels larger than area

Where it can be found:

Basic Operations Module ► Post Processing ► Remove labels (larger)

Description:

Remove labels (larger) is a post-processing operation available in Basic Operations Module.

This operation deletes events from the input image, that have an area greater than the threshold value and places the result in the output.

$$Result = Events < Area\ threshold\ (px)$$

Parameters:

- 1.1 Image - the input image
- 1.2 Area threshold (px) - the minimum area value, in pixels, of the events to be removed (default = 0)
- 1.3 Result of "…" - the result of the operation

Effect:

Removes events having an area greater than the threshold.

Example:

- Input:

The input is a coded image (events), representing the detected nuclei in a breast cancer sample (highlighted in green).

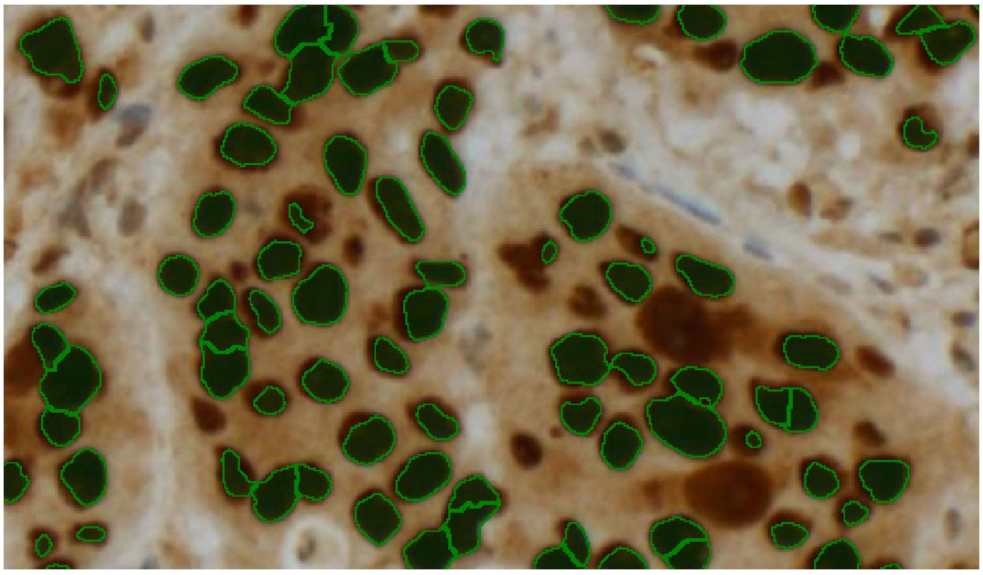


Figure 378 – Breast cancer sample - Detected nuclei

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a coded image (events), matching the input.

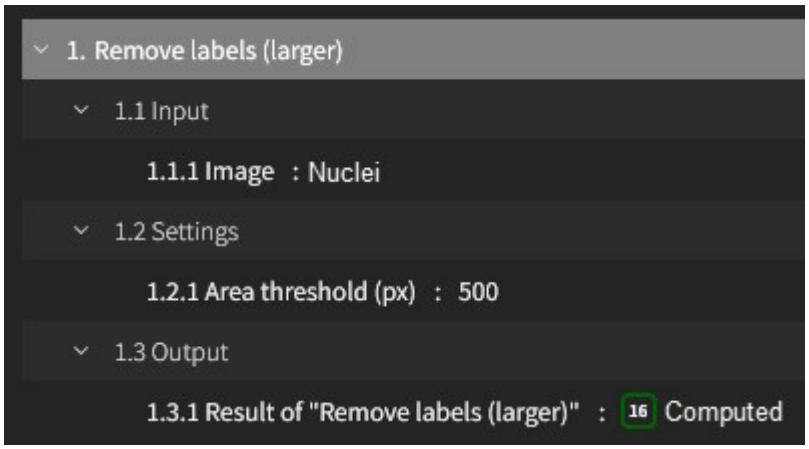


Figure 379 – Engine settings

- Output:

Figure below shows:

- The engine's result (highlighted in green) – all nuclei having area ≤ 500
- Removed events (highlighted in red) – all nuclei having area > 500

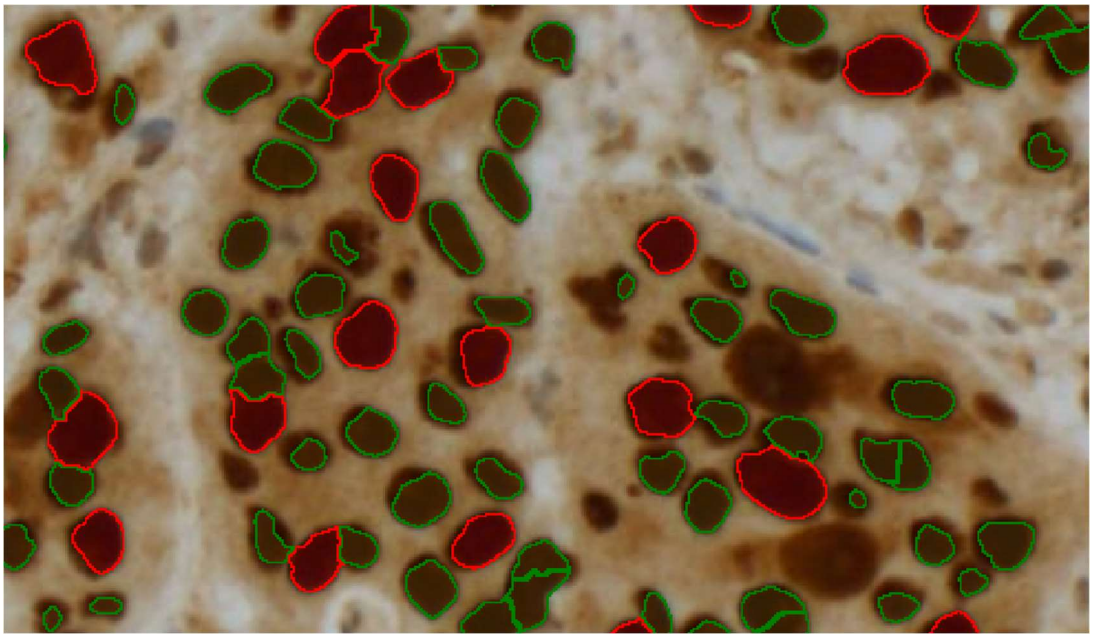


Figure 380 – Result of Remove labels (larger) engine

Remove labels (smaller)

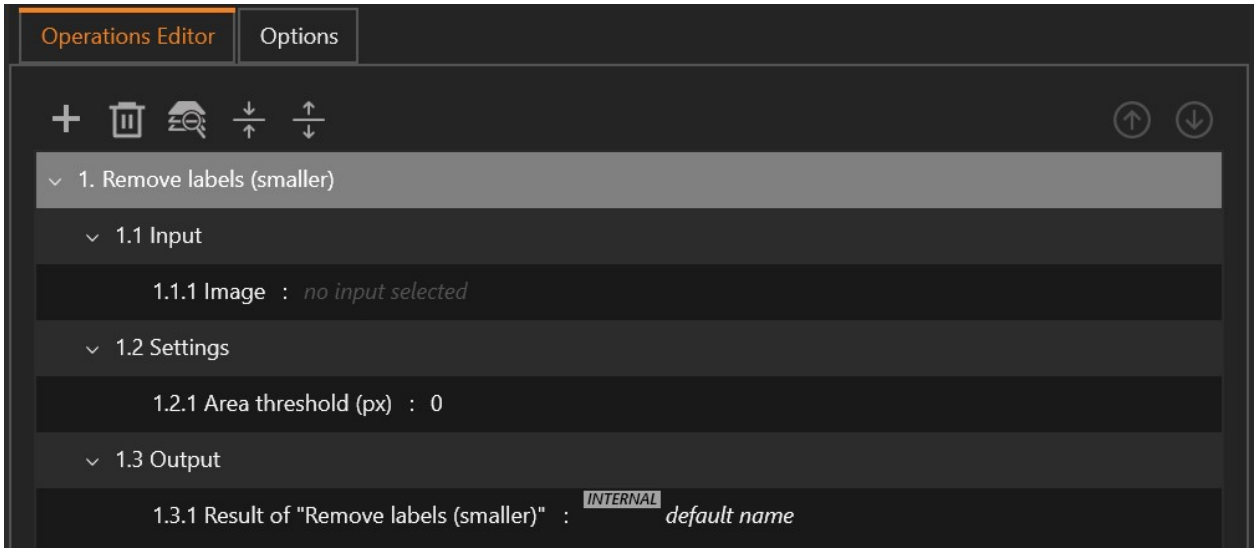


Figure 381 – Remove labels smaller than area

Where it can be found:

Basic Operations Module ► Post Processing ► Remove labels (smaller)

Description:

Remove labels (smaller) is a post-processing operation available in Basic Operations Module.

This operation deletes events from the input image, that have an area less than the threshold value and places the result in the output.

$$Result = Events > Area\ threshold\ (px)$$

Parameters:

- 1.1 Image - the input image
- 1.2 Area threshold (px) - the max area value, in pixels, of the events to be removed (default = 0)
- 1.3 Result of "…" - the result of the operation

Effect:

Removes events having an area smaller than the threshold.

Example:

- Input:

The input is a coded image (events), representing the detected nuclei in a breast cancer sample (highlighted in green).

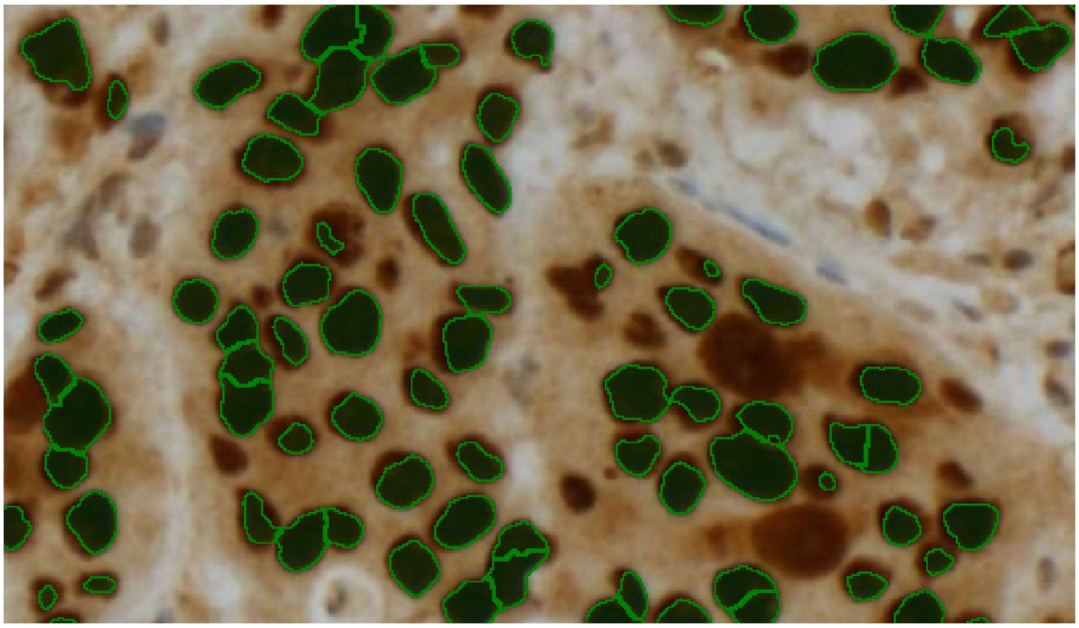


Figure 382 – Breast cancer sample - Detected nuclei

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a coded image (events), matching the input.

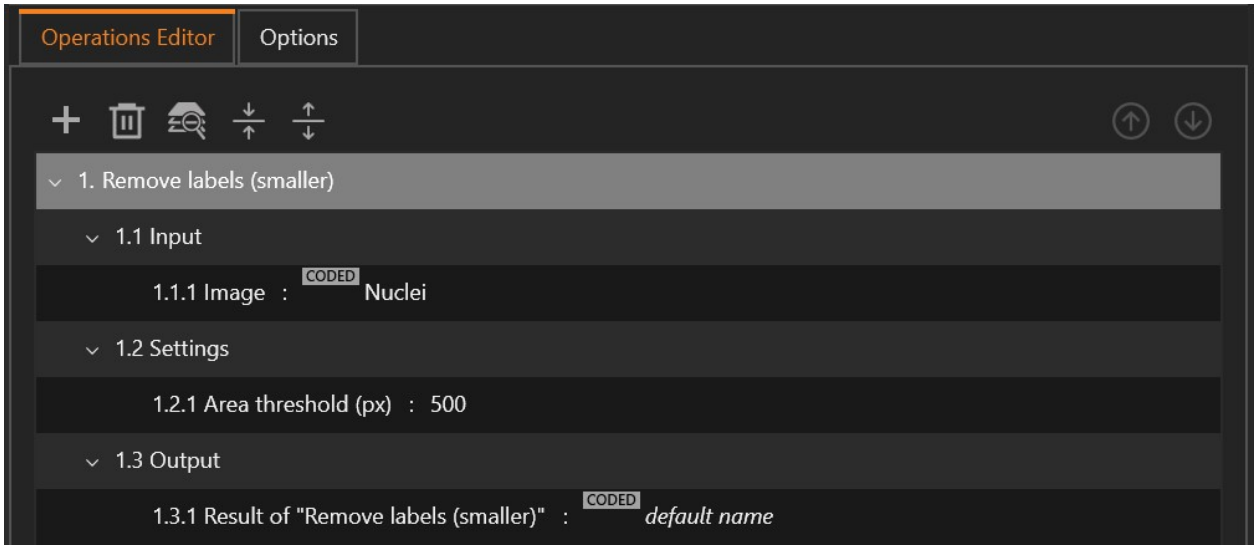


Figure 383 – Engine settings

- Output:

Figure below shows:

- The engine's result (highlighted in green) – all nuclei having area ≥ 500
- Removed events (highlighted in red) – all nuclei having area < 500

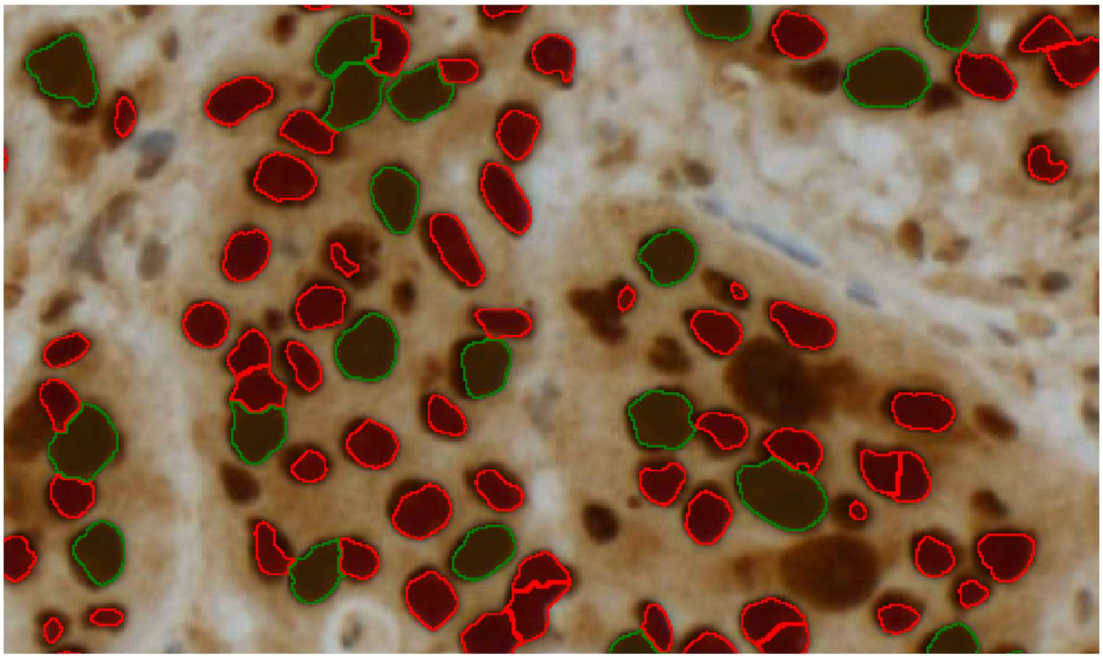


Figure 384 – Result of Remove labels (smaller) engine

Remove labels (stronger)

Figure 385 – Remove labels stronger than intensity

Where it can be found:

Basic Operations Module ► Post Processing ► Remove labels (stronger)

Description:

Remove labels (stronger) is a post-processing operation available in Basic Operations Module.

This operation deletes events from the input image, that have a mean intensity greater than the threshold value and places the result in the output.

$$Result = Events > Intensity\ threshold$$

The mean intensity of each event is computed by averaging the intensity of the pixels from Image 2 (gray) corresponding to the event's mask (event's body) from Image 1(coded).

Parameters:

- 1.1 Image 1 (coded) - the first input image, representing the events contours (bodies)
- 1.2 Image 2 (gray) - the second input image, representing the gray values
- 1.3 Intensity threshold - the minimum average intensity value of the events to be removed (default = 0)
- 1.4 Result of "…" - the result of the operation

Effect:

Removes events having a mean intensity higher than the threshold.

Example:

- Input:

The input is a coded image (events), representing the detected nuclei in a fluorescence sample (highlighted in green).

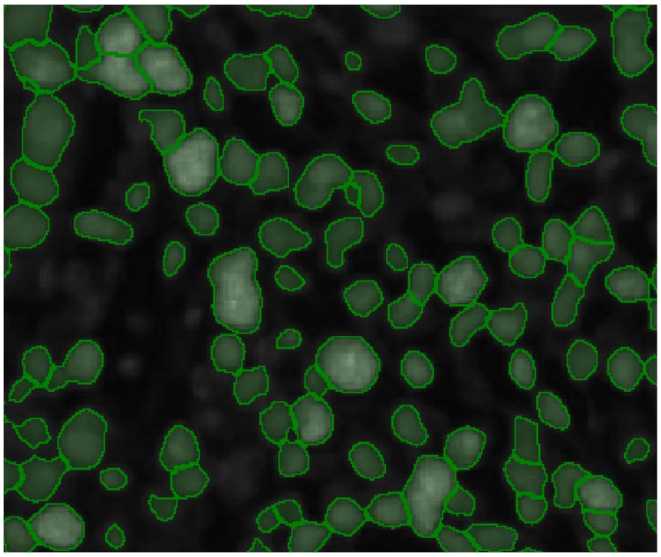


Figure 386 – Detected nuclei on DAPI channel

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a coded image (events), matching the input.

Figure 387 – Engine settings

- Output:

Figure below shows:

- The engine's result (highlighted in green) – all nuclei having an average intensity ≤ 75
- Removed events (highlighted in red) – all nuclei having an average intensity > 75

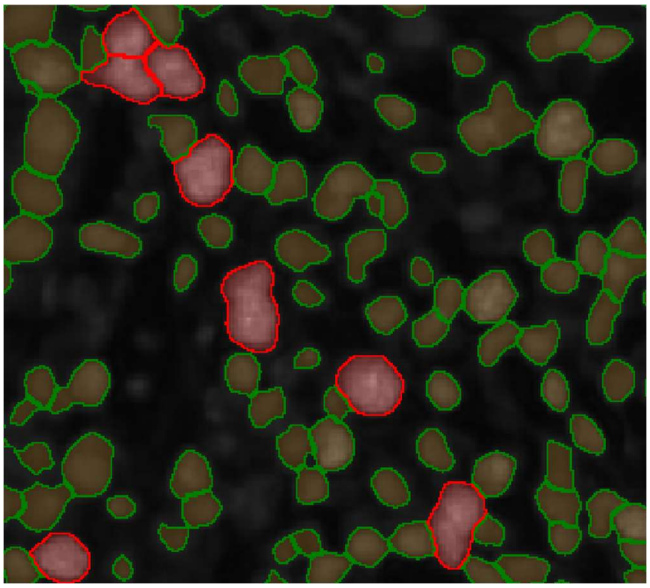


Figure 388 – Result of Remove labels (stronger) engine

Remove labels (weaker)

Figure 389 – Remove labels weaker than intensity

Where it can be found:

Basic Operations Module ► Post Processing ► Remove labels (weaker)

Description:

Remove labels (weaker) is a post-processing operation available in Basic Operations Module.

This operation deletes events from the input image, that have a mean intensity less than the threshold value and places the result in the output.

$$Result = Events < Intensity\ threshold$$

The mean intensity of each event is computed by averaging the intensity of the pixels from Image 2 (gray) corresponding to the event's mask (event's body) from Image 1(coded).

Parameters:

- 1.1 Image 1 (coded) - the first input image, representing the events contours (bodies)
- 1.2 Image 2 (gray) - the second input image, representing the gray values
- 1.3 Intensity threshold - the minimum average intensity value of the events to be removed (default = 0)
- 1.4 Result of "…" - the result of the operation

Effect:

Removes events having a mean intensity lower than the threshold.

Example:

- Input:

The input is a coded image (events), representing the detected nuclei in a fluorescence sample (highlighted in green).

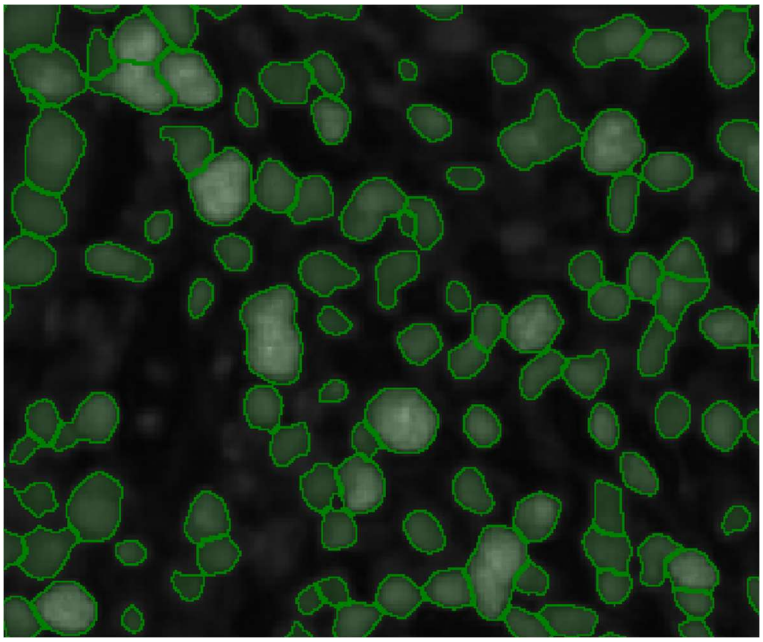


Figure 390 – Detected nuclei on DAPI channel

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a coded image (events), matching the input.

Figure 391 – Engine settings

- Output:

Figure below shows:

- The engine's result (highlighted in green) – all nuclei having an average intensity ≥ 75
- Removed events (highlighted in red) – all nuclei having an average intensity < 75

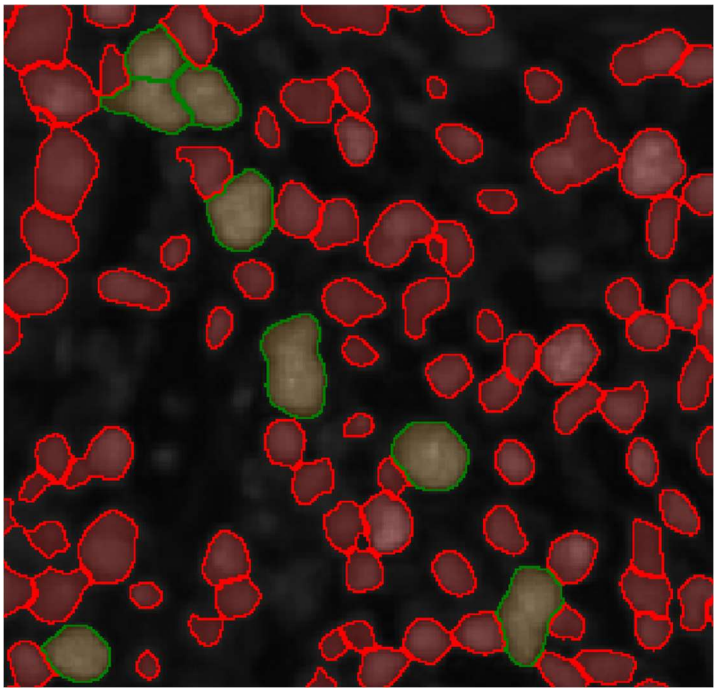


Figure 392 – Result of Remove labels (weaker) engine

Remove objects (with seeds)

Figure 393 – Remove Objects with Seeds

Where it can be found:

Basic Operations Module ► Post Processing ► Remove objects (with seeds)

Description:

Remove objects (with seeds) is a post-processing operation available in Basic Operations Module.

This operation deletes objects from the input Image 1, which have a seed in Image 2 and places the result in the output. An object from Image 2 is considered a seed for an object from Image 1 if their intersection has at least 1 pixel.

Parameters:

- 1.1 Image 1 - the first input image, representing the evaluated objects
- 1.2 Image 2 (seeds) - the second input image, representing the seeds for the object from Image 1
- 1.3 Result of "…" - the result of the operation

Effect:

Removes objects from Image 1 intersecting objects from Image 2.

Example:

- Input:

The two input images are mask images (8-bit, 1-channel).

Two cases are considered:

- case 1: objects = mask 1 & seeds = mask 2
- case 2: objects = mask 2 & seeds = mask 1

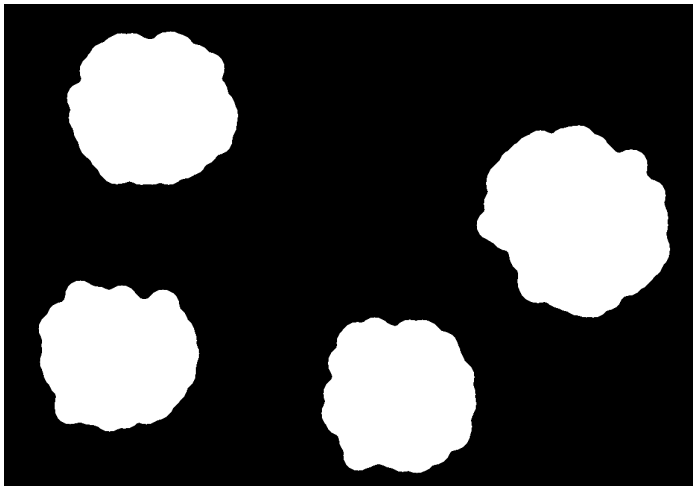


Figure 394 – Mask image 1

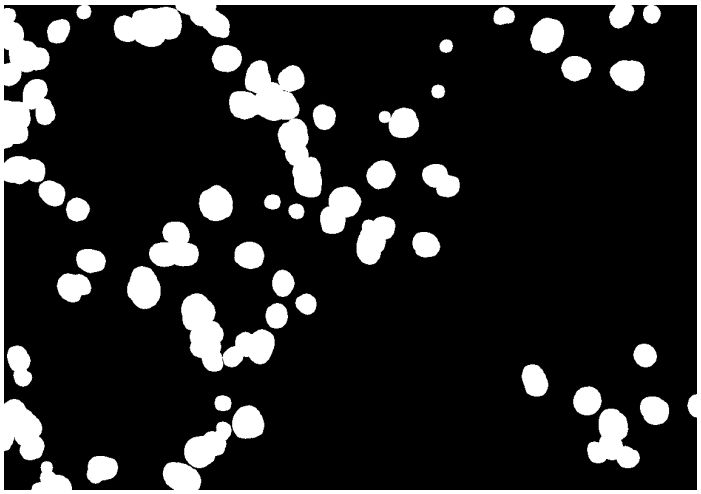


Figure 395 – Mask image 2

- Engine settings:

First figure shows the engine settings used for case 1 example.

Second figure shows the engine settings used for case 2 example.

The engine's result is a mask image (8-bit, 1-channel), matching the image representing the objects subject to removal.

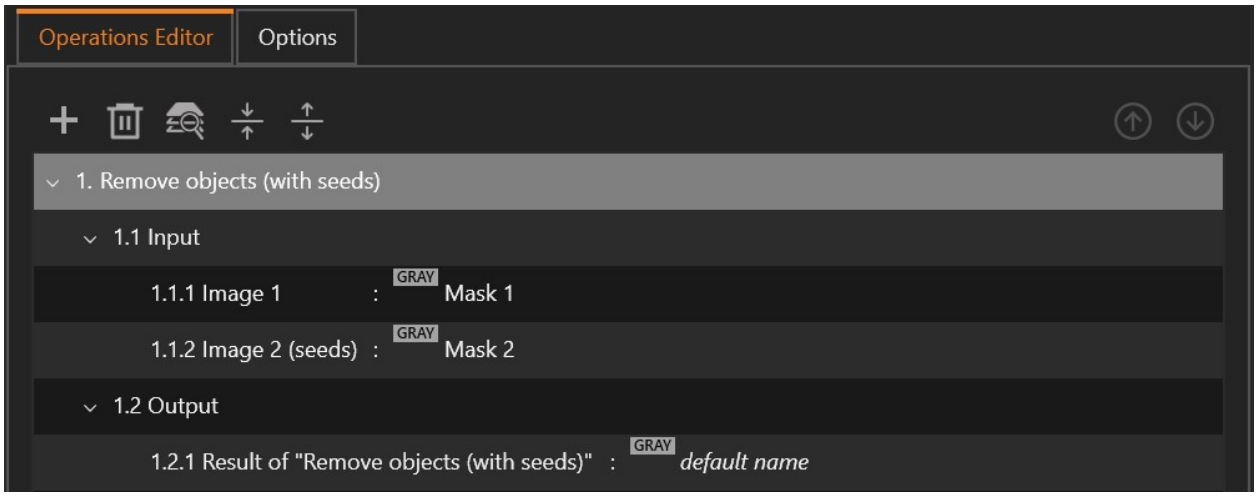


Figure 396 – Engine settings (case 1)

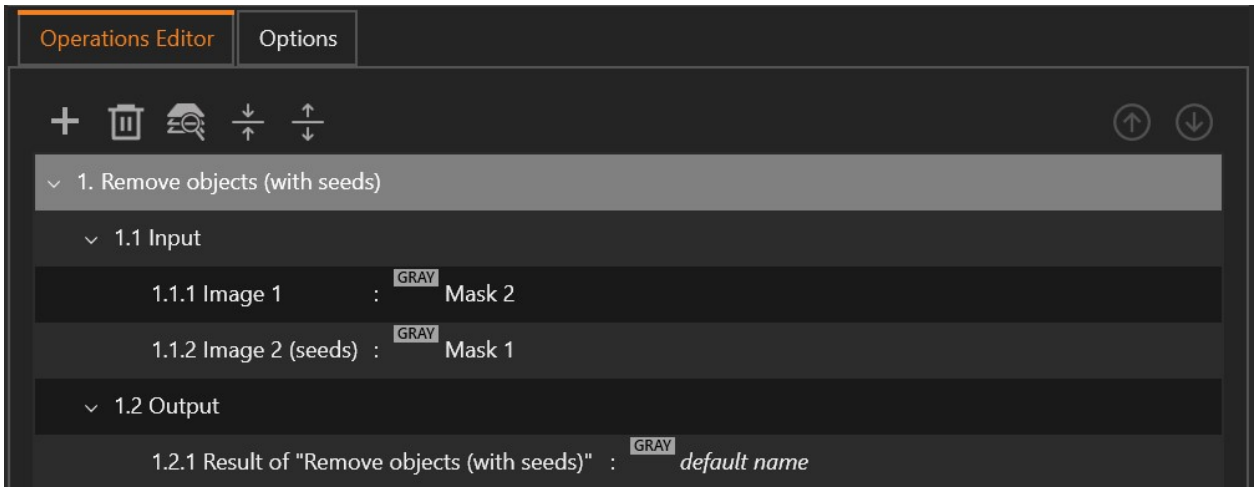


Figure 397 – Engine settings (case 2)

- Output:

First figure (case 1), representing the overlay of the 2 input masks where:

- mask 1 represents the objects highlighted in gray
- mask 2 represents the seeds highlighted in red

Second figure shows the result for case 1, representing the objects without seeds. Objects with seed have been removed.

Third figure shows case 2, representing the overlay of the 2 input masks where:

- mask 2 represents the objects highlighted in gray
- mask 1 represents the seeds highlighted in red

Fourth figure shows the result for case 2, representing the objects without seed. Objects with seed have been removed.

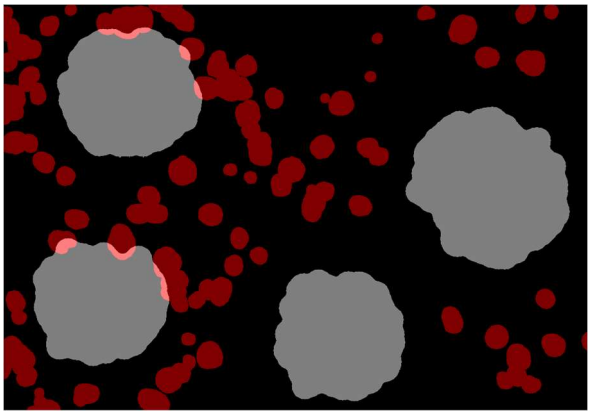


Figure 398 – Input images overlay (case 1)

Figure 399 – Result of Remove objects (with seeds) engine (case 1)

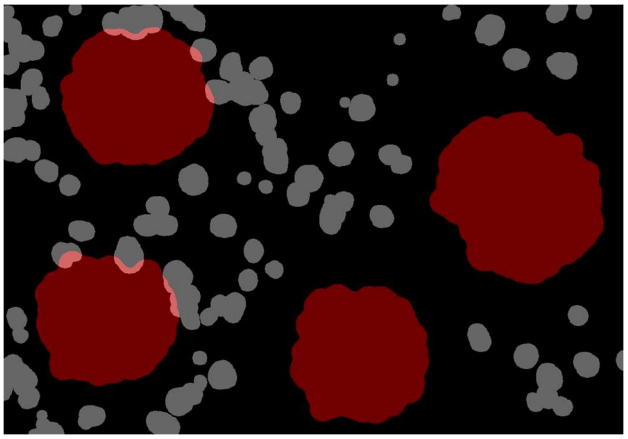


Figure 400 – Input images overlay (case 2)

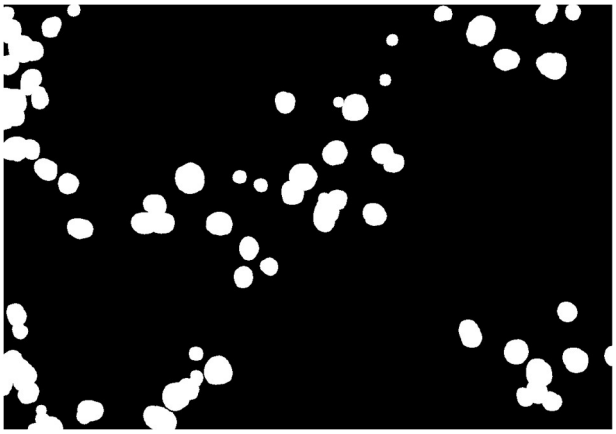


Figure 401 – Result of Remove objects (with seeds) engine (case 2)

Remove objects (without seeds)

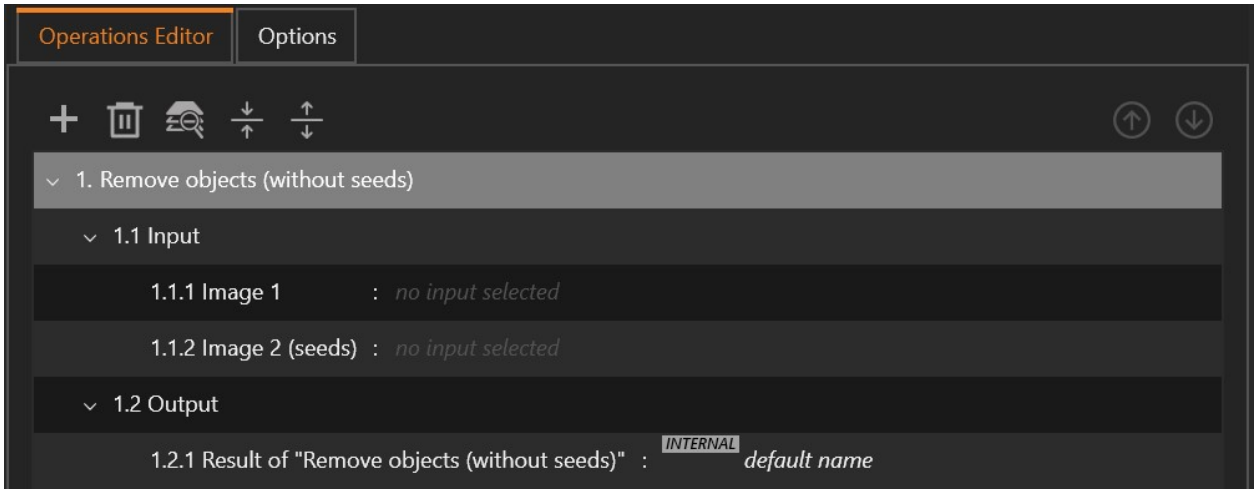


Figure 402 – Remove Objects without Seeds

Where it can be found:

Basic Operations Module ► Post Processing ► Remove objects (without seeds)

Description:

Remove objects (without seeds) is a post-processing operation available in Basic Operations Module.

This operation deletes objects from Image 1, which don't have a seed in Image 2 and places the result in the output. An object from Image 2 is considered seed for an object from Image 1 if the intersection of the two has at least 1 pixel.

Parameters:

- 1.1 Image 1 - the first input image, representing the evaluated objects
- 1.2 Image 2 (seeds) - the second input image, representing the seeds for the object from Image 1
- 1.3 Result of "…" - the result of the operation

Effect:

Removes objects from Image 1 not intersecting objects from Image 2.

Example:

- Input:

The two input images are mask images (8-bit, 1-channel).

Two cases are considered:

- case 1: objects = mask 1 & seeds = mask 2
- case 2: objects = mask 2 & seeds = mask 1

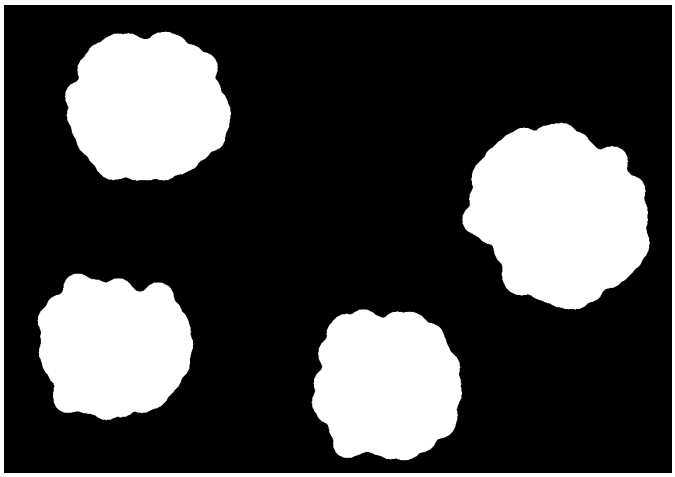


Figure 403 – Mask image 1

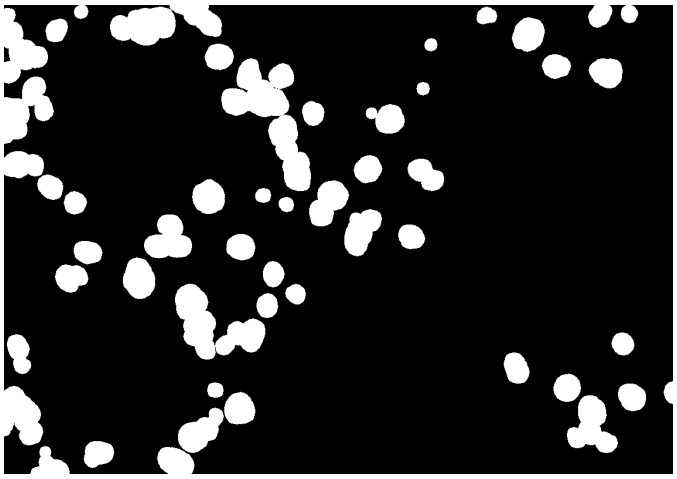


Figure 404 – Mask image 2

- Engine settings:

First figure shows the engine settings used for case 1 example.

Second figure shows the engine settings used for case 2 example.

The engine's result is a mask image (8-bit, 1-channel), matching the image representing the objects subject to removal.

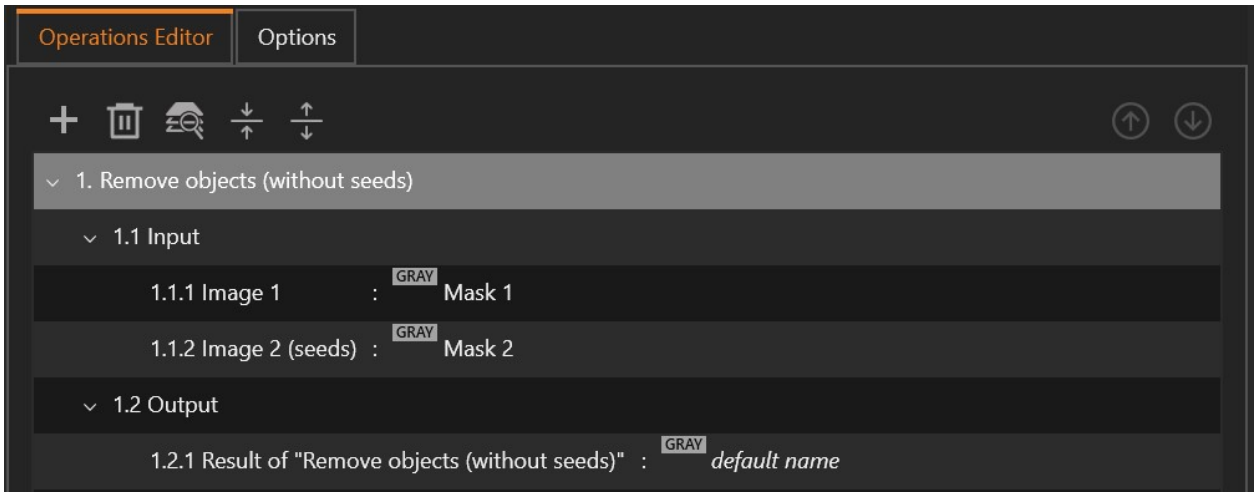


Figure 405 – Engine settings (case 1)

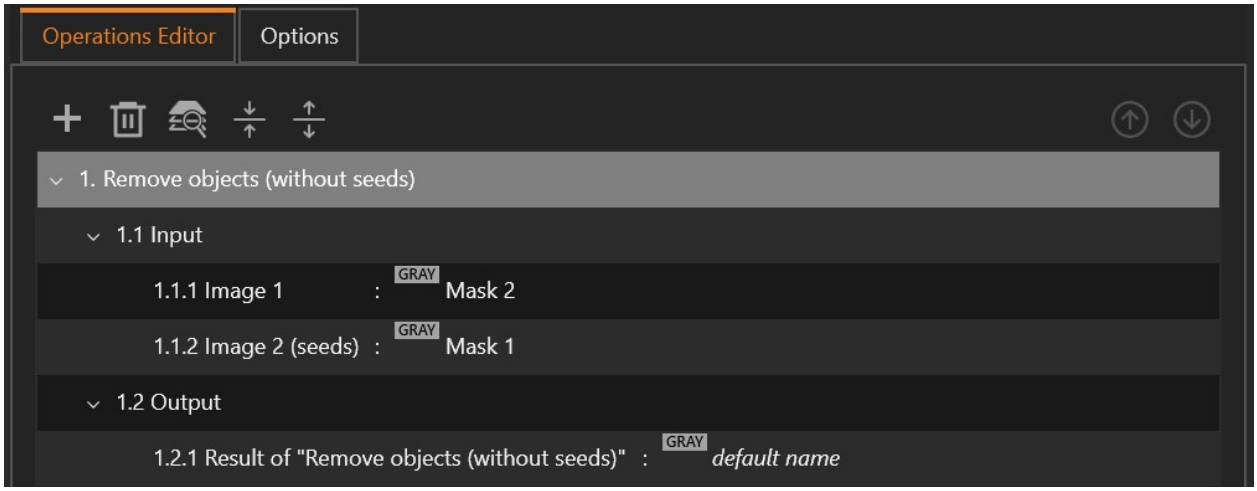


Figure 406 – Engine settings (case 2)

- Output:

First figure shows case 1, representing the overlay of the 2 input masks where:

- mask 1 represents the objects highlighted in gray
- mask 2 represents the seeds highlighted in red

Second figure shows the result for case 1, representing the objects with seed. Objects without seed have been removed.

Third figure shows case 2, representing the overlay of the 2 input masks where:

- mask 2 represents the objects highlighted in gray
- mask 1 represents the seeds highlighted in red

Fourth figure shows the result for case 2, representing the objects with seed. Objects without seed have been removed.

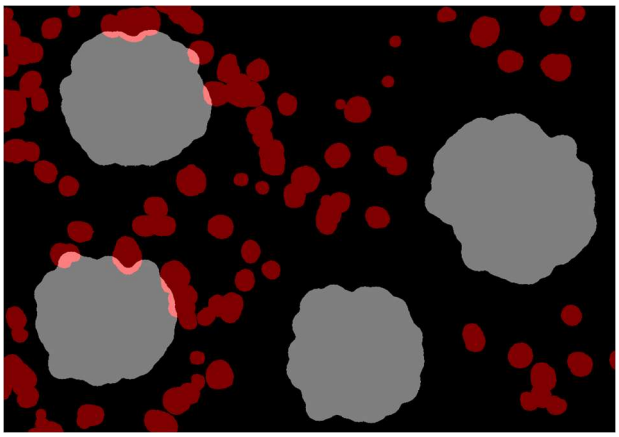


Figure 407 – Input images overlay (case 1)

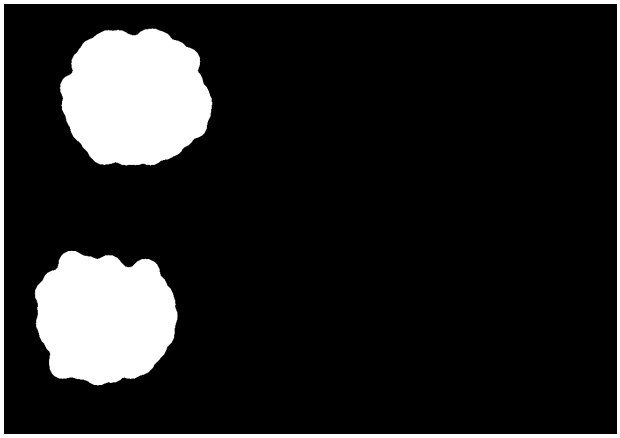


Figure 408 – Result of Remove objects (without seeds) engine (case 1)

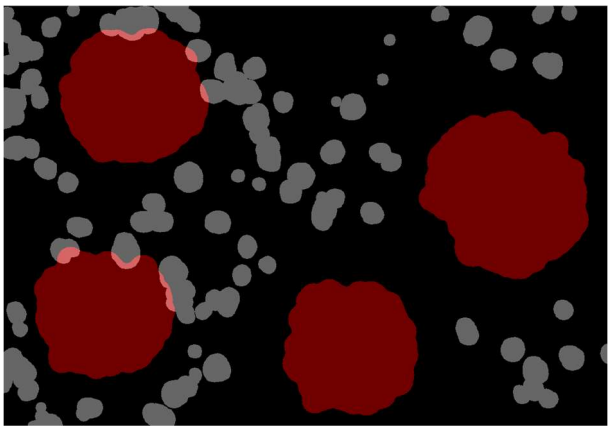


Figure 409 – Input images overlay (case 2)

Figure 410 – Result of Remove objects (without seeds) engine (case 2)

5.10. Statistics

Max (every pixel)

Figure 411 – Max Every Pixel

Where it can be found:

Basic Operations Module ► Statistics ► Max (every pixel)

Description:

Max (every pixel) is a statistical operation available in Basic Operations Module.

This operation computes the maximum value for each pair of pixels from two images and places the result in the output (maximum of two images pixel-wise).

$$Result\ (x, y) = \max\ (Image\ 1(x, y), Image\ 2(x, y))$$

Parameters:

- 1.1 Image 1 - the first input image
- 1.2 Image 2 - the second input image
- 1.3 Result of "…" - the result of the operation

Effect:

Computes the pixel-wise maximum of two images.

Example:

- Input:

The images are the marker channels resulting after color separation of an immunohistochemical colon sample.

First figure shows the quantification of the Hematoxylin maker and second figure shows the quantification of Ki67 marker.

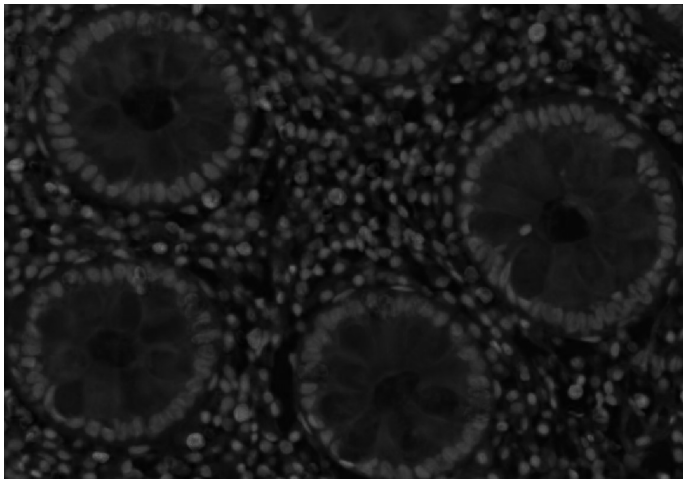


Figure 412 – Colon sample (separated Hematoxylin marker)

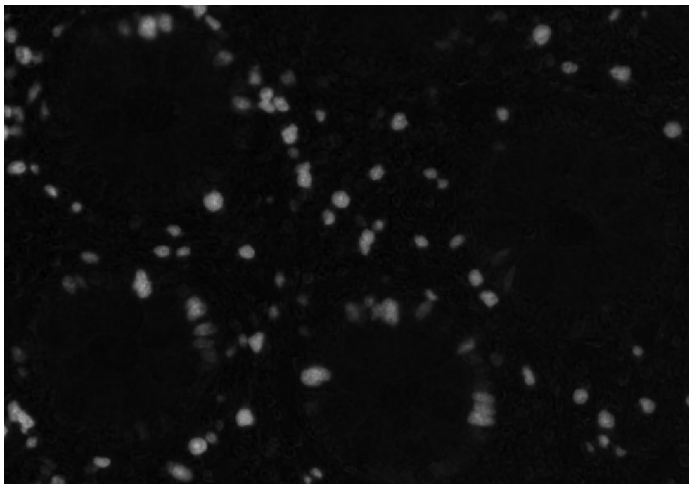


Figure 413 – Colon sample (separated Ki67 marker)

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a grayscale image (8-bit, 1-channel), matching the input.

Figure 414 – Engine settings

- Output:

Figure below shows the maximum value from the 2 input images for each pixel.

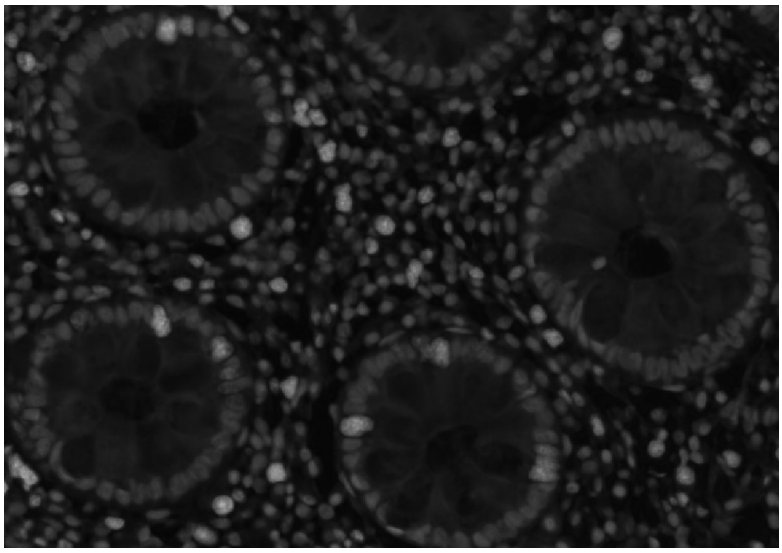


Figure 415 – Result of Max (every pixel) engine

Min (every pixel)

Figure 416 – Min Every Pixel

Where it can be found:

Basic Operations Module ► Statistics ► Min (every pixel)

Description:

Min (every pixel) is a statistical operation available in Basic Operations Module.

This operation computes the minimum value for each pair of pixels from two images and places the result in the output (maximum of two images pixel-wise).

$$Result\ (x, y) = \min\ (Image\ 1(x, y), Image\ 2(x, y))$$

Parameters:

- 1.1 Image 1 - the first input image
- 1.2 Image 2 - the second input image
- 1.3 Result of "…" - the result of the operation

Effect:

Computes the pixel-wise maximum of two images.

Example:

- Input:

The images are the marker channels resulting from the color separation of an immunohistochemical colon sample.

First figure shows the quantification of the Hematoxylin maker and second figure shows the quantification of Ki67 marker

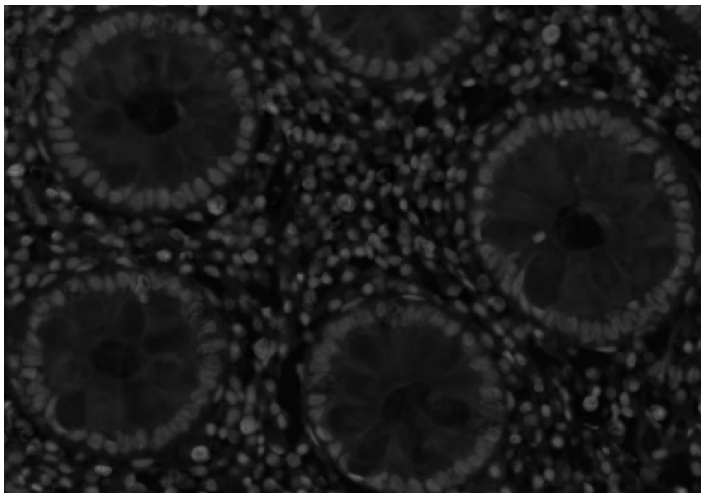


Figure 417 – Colon sample (separated Hematoxylin marker)

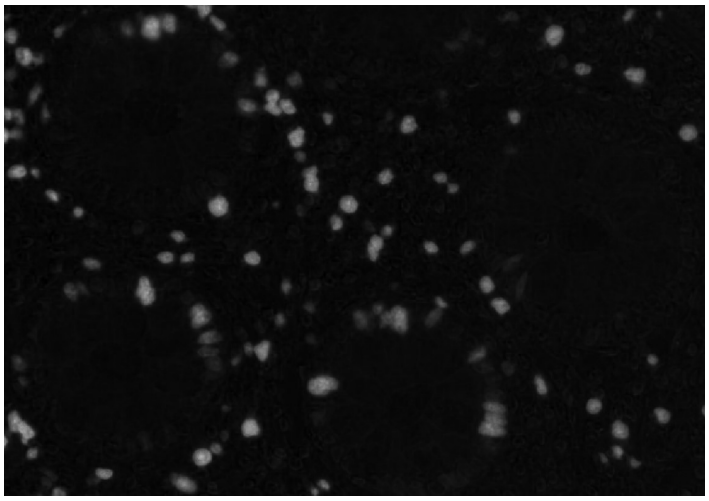


Figure 418 – Colon sample (separated Ki67 marker)

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a grayscale image (8-bit, 1-channel), matching the input.

Figure 419 – Engine settings

- Output:

Figure below shows the minimum value from the 2 input images for each pixel.

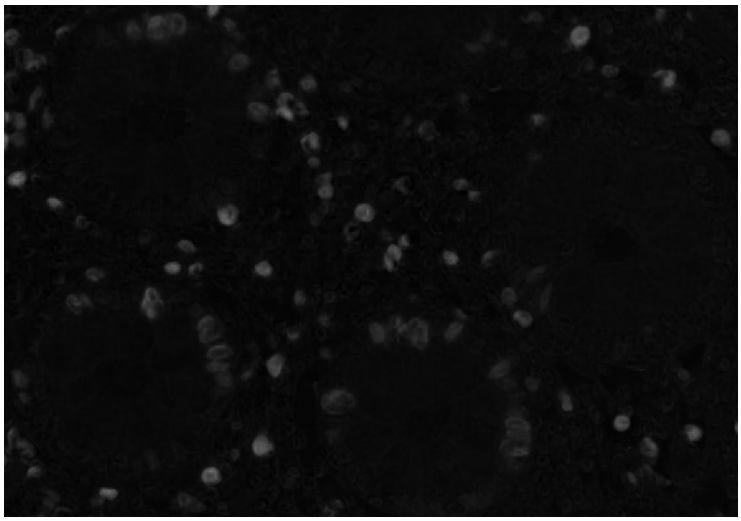


Figure 420 – Result of Min (every pixel) engine

5.11. Threshold and Compare

Compare

Figure 421 – Compare

Where it can be found:

Basic Operations Module ► Threshold and Compare ► Compare

Description:

Compare is a compare operation available in Basic Operations Module.

This operation compares pixels of the source image to a given threshold value, using the specified compare mode and places the result in the output. If the result of the compare operation is true, the corresponding pixel of the output is set to 255 (white). Otherwise, it is set to 0 (black).

Example for Compare Mode = “Greater”:

$$Result(x,y) = \begin{cases} 255, & Image(x,y) > Threshold \\ 0, & otherwise \end{cases}$$

The output of the operation is always a mask image.

Parameters:

- 1.1 Image - input image
- 1.2 Threshold - threshold value (default = 0)
- 1.3 Compare Mode - specify the compare mode to be used (default = Less). There are 6 options:
  - Less
  - Less or equal
  - Greater or Equal
  - Greater
  - Not equal
- 1.4 Result of “…” - result of the operation

Effect:

Generates a mask (binarization) from a grayscale image using a threshold value.

Example:

- Input:

The input is a grayscale image (8-bit, 1-channel) representing the combined information from Hematoxylin and Ki67 markers, in an immunohistochemical small region of a colon sample. The Hematoxylin and Ki67 images were generated using the Color separation engine.

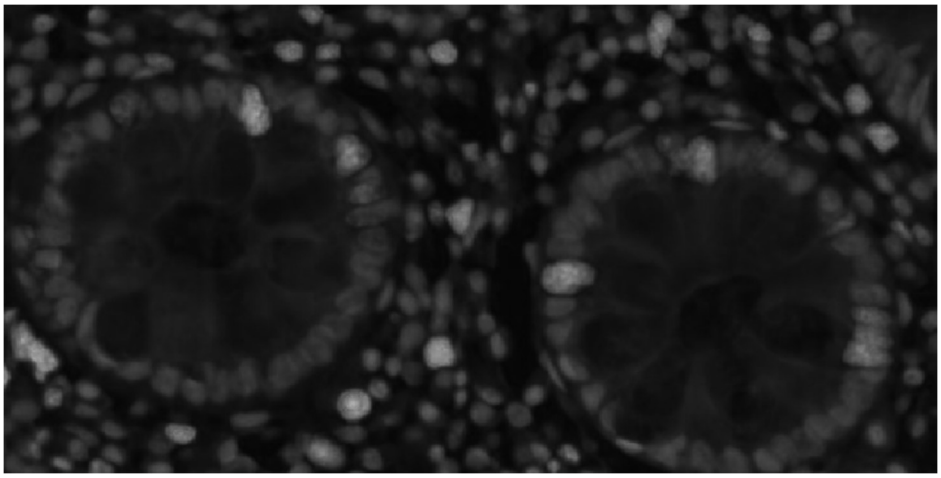


Figure 422 – Colon sample

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a mask image (8-bit, 1-channel).

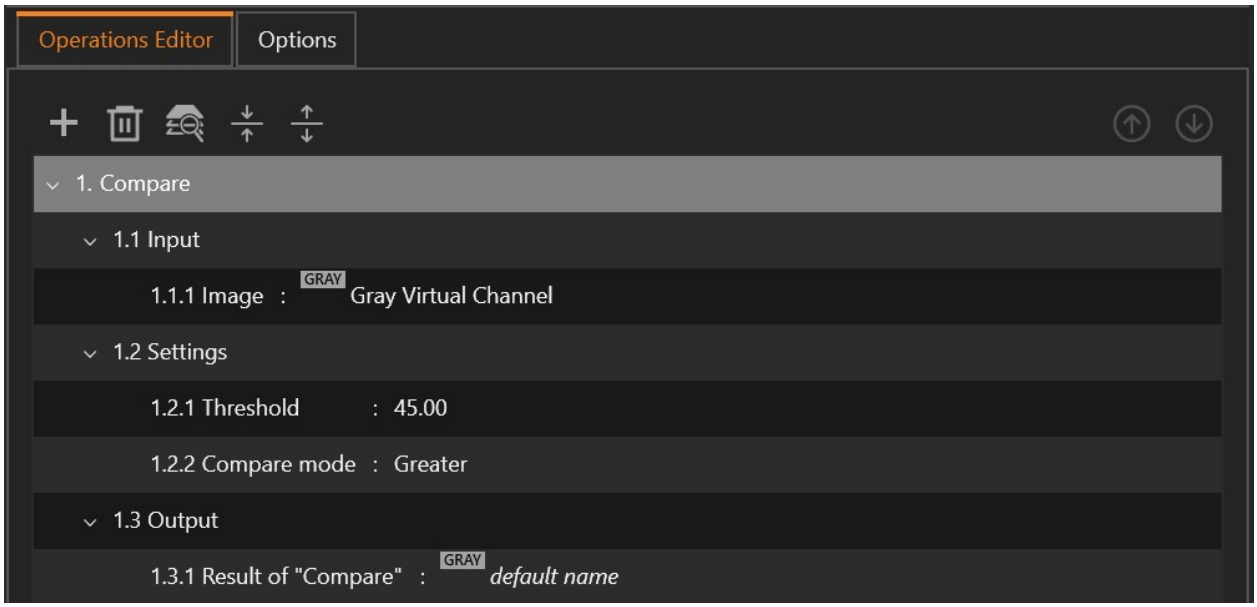


Figure 423 – Engine settings

- Output:

Figure below shows the result of the selected compare operation (Compare mode = Greater) of all pixels with the threshold value (Threshold = 45). All pixels meeting the condition are highlighted in white.

Figure 424 – Result of Compare engine

Compare (RGB)

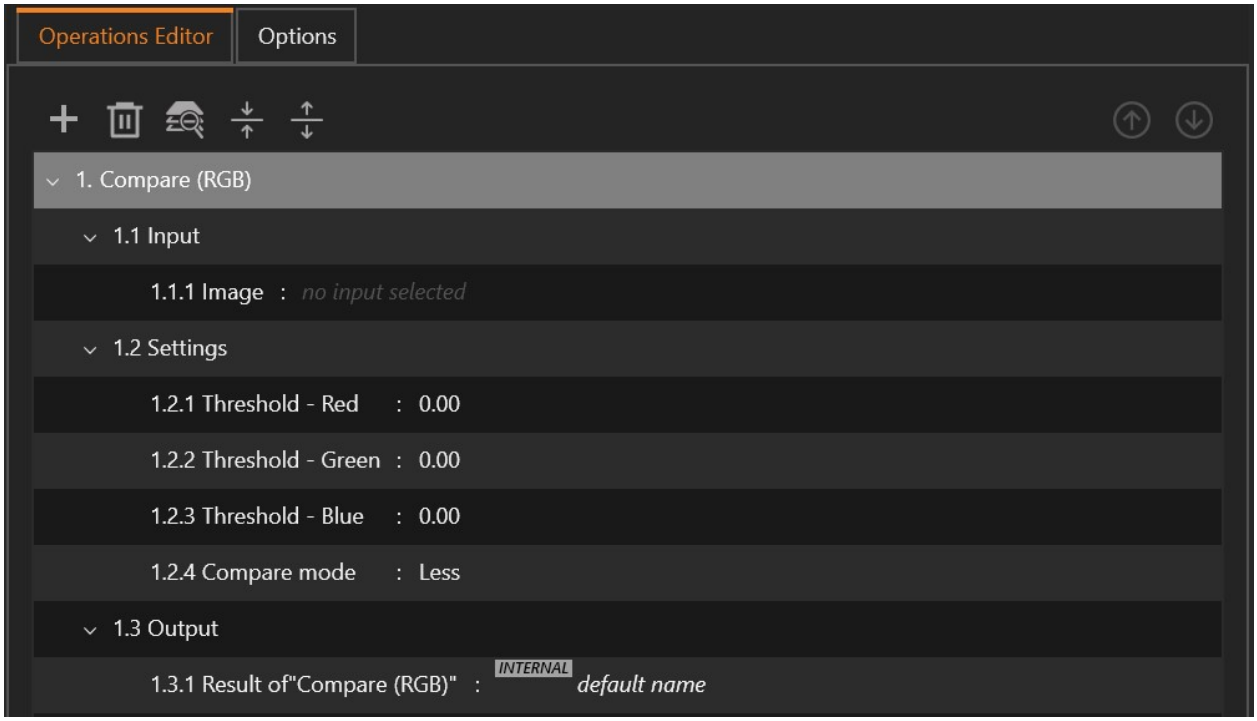


Figure 425 – Compare RGB

Where it can be found:

Basic Operations Module ► Threshold and Compare ► Compare (RGB)

Description:

Compare (RGB) is a compare operation available in Basic Operations Module.

This operation compares pixels from each channel of the source image to a given threshold value (with R, G and B components), using the specified compare mode and places the result in the output. If the result of the compare operation is true, meaning all pixel values of all channels meet the specified condition, the corresponding pixel of the output is set to 255 (white); otherwise, it is set to 0 (black).

Example for Compare Mode = “Greater”:

$$Result(x,y) = \begin{cases} 255, & Image(x,y)\,\langle \boldsymbol{r}|\boldsymbol{g}|\boldsymbol{b}\rangle > Threshold\,\langle \boldsymbol{r}|\boldsymbol{g}|\boldsymbol{b}\rangle \\ 0, & otherwise \end{cases}$$

The output of the operation if always a mask image.

Parameters:

- 1.1 Image - the input image
- 1.2 Threshold - Red - the threshold value for red channel (default = 0)
- 1.3 Threshold - Green - the threshold value for green channel (default = 0)
- 1.4 Threshold - Blue - the threshold value for blue channel (default = 0)
- 1.5 Compare Mode - specifies the compare mode to be used (default = Less). There are 6 options:
  - Less
  - Less or equal
  - Greater or Equal
  - Greater
  - Not equal
- 1.4 Result of “…” - the result of the operation

Effect:

Generates a mask (binarization) from a color (RGB) image using a threshold value (with R, G, B components).

Example:

- Input:

The input is a brightfield color image (8-bit, 3-channel) representing a small region of an immunohistochemical colon sample.

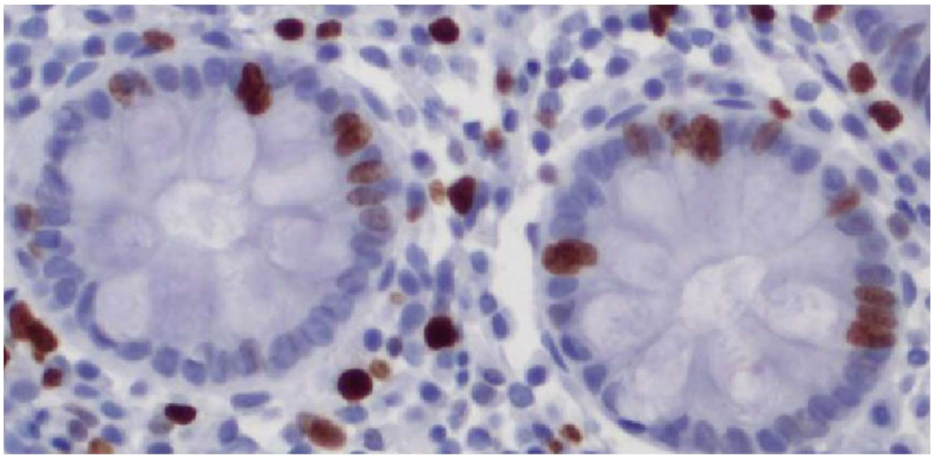


Figure 426 – Colon sample

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a mask image (8-bit, 1-channel).

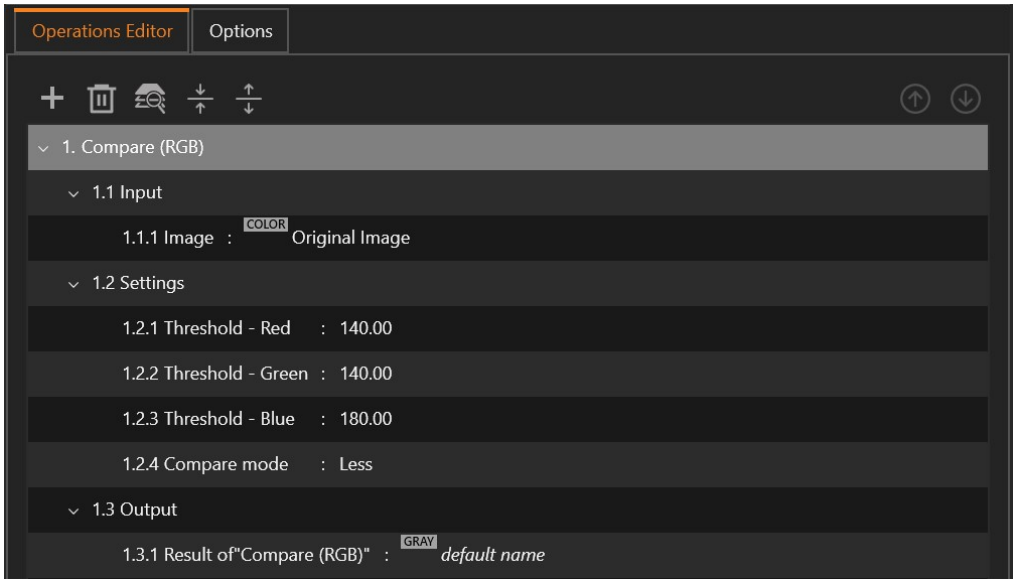


Figure 427 – Engine settings

- Output:

Figure below shows the result of the selected compare operation (Compare mode = Less) of all pixels with the threshold values:

$$Result(x,y) = \begin{cases} 255, & Image(x,y)\ \langle \boldsymbol{r}|\boldsymbol{g}|\boldsymbol{b}\rangle < \langle \mathbf{140}|\mathbf{140}|\mathbf{180}\rangle \\ 0, & otherwise \end{cases}$$

All pixels meeting the conditions are highlighted in white.

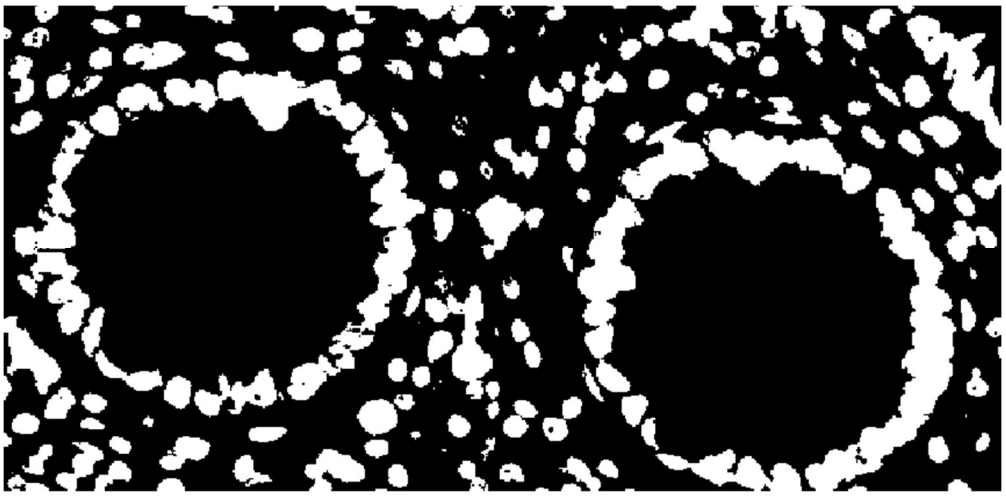


Figure 428 – Result of Compare (RGB) engine

Compare - inside interval

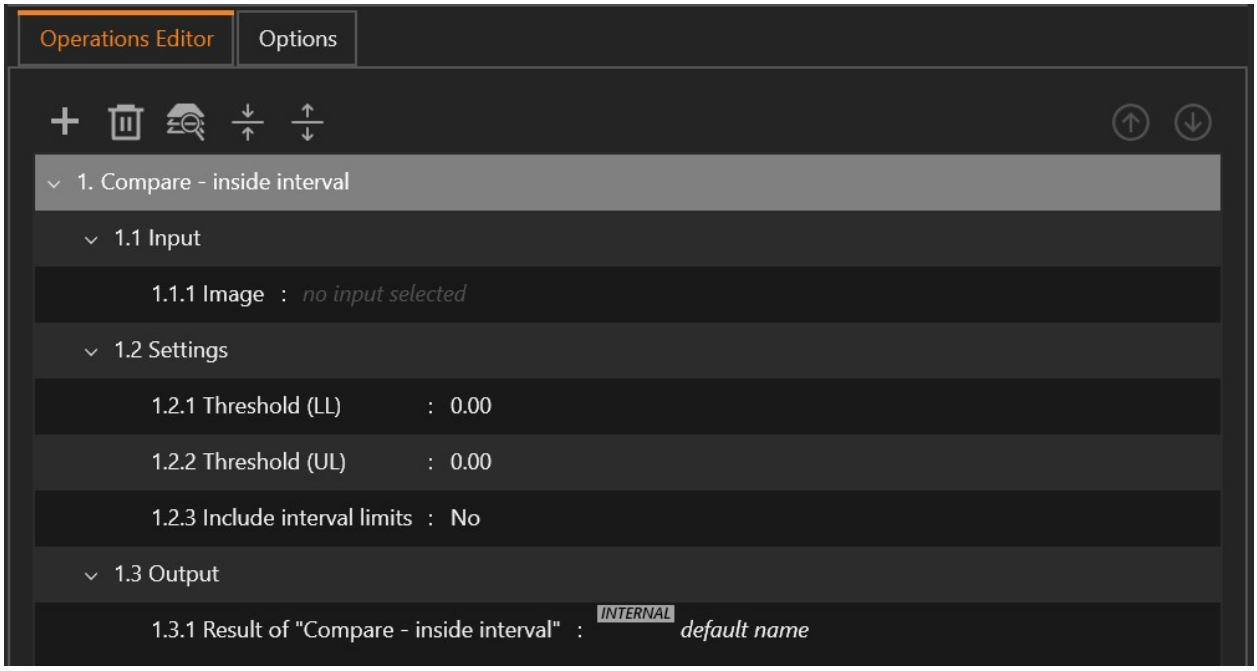


Figure 429 – Compare - Inside Interval

Where it can be found:

Basic Operations Module ► Threshold and Compare ► Compare -inside interval

Description:

Compare -inside interval is a compare operation available in Basic Operations Module.

This operation compares pixels of the source image with the limits of a given range and places the result in the output. If the result is true, meaning the pixel value is inside the range, the corresponding pixel of the output is set to 255 (white). Otherwise, it is set to 0 (black).

Example for Include interval limits = “No”:

$$Result(x,y) = \begin{cases} 255, & Threshold\ (LL) < Image(x,y) < Threshold\ (UL) \\ 0, & otherwise \end{cases}$$

The output of the operation is always a mask image.

Parameters:

- 1.1 Image - the input image
- 1.2 Threshold (LL) - the lower limit of the range (default = 0)
- 1.3 Threshold (UL) - the upper limit of the range (default = 0)
- 1.4 Include interval limits - switches from “<” and “>” to “≤” and “≥” (default = No)
- 1.5 Result of “…” - the result of the operation

Effect:

Generates a mask (binarization) from a grayscale image using two thresholds.

Example:

- Input:

The input is a grayscale image (8-bit, 1-channel) representing the combined information from Hematoxylin and Ki67 markers, in an immunohistochemical small region of a colon sample. The Hematoxylin and Ki67 images were generated using the Color separation engine.

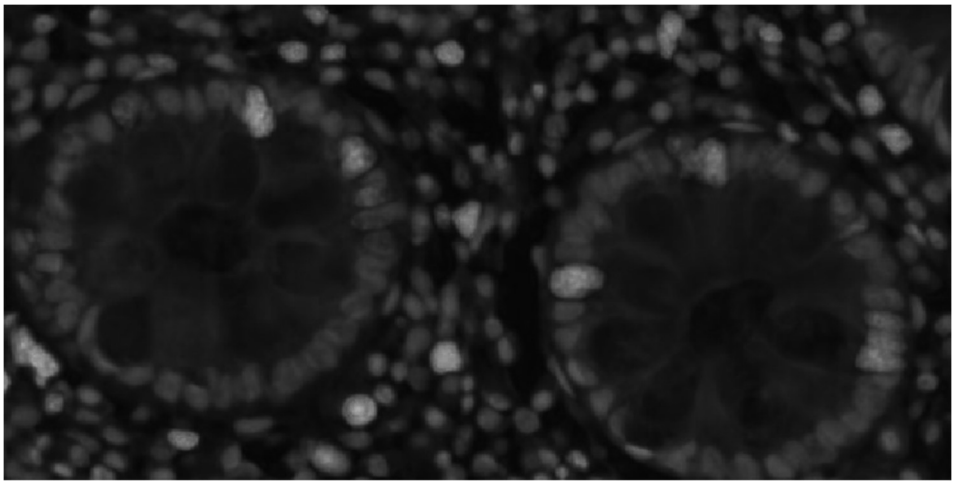


Figure 430 – Colon sample

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a mask image (8-bit, 1-channel).

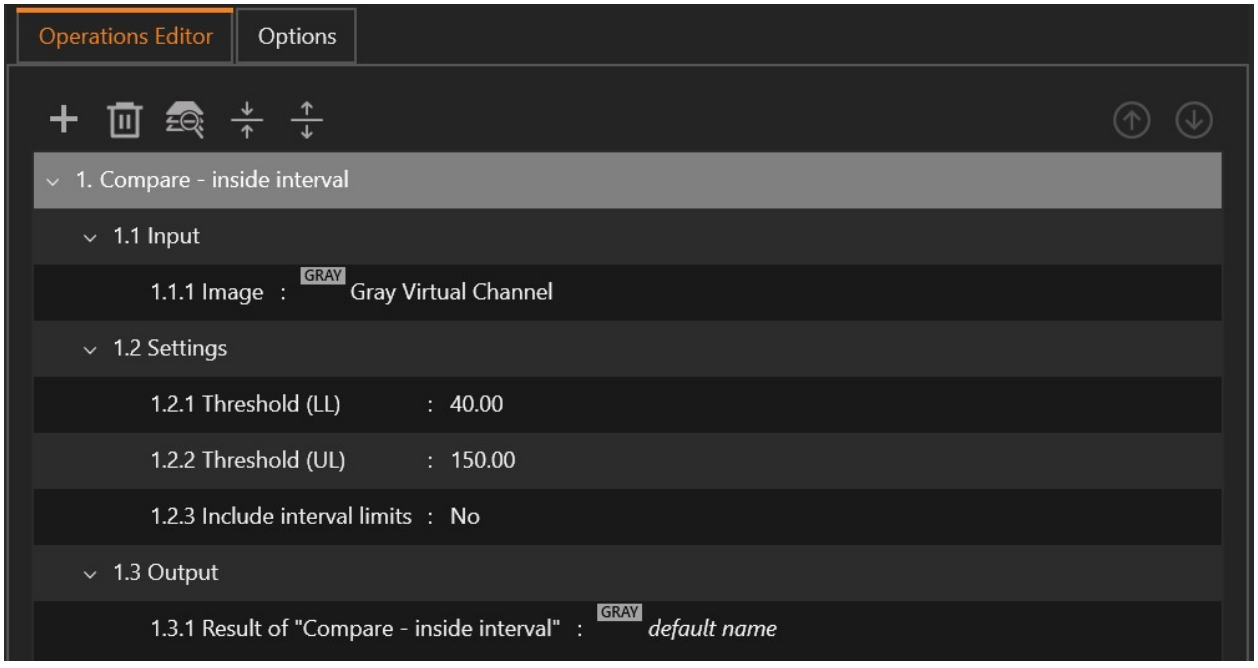


Figure 431 – Engine settings

- Output:

Figure 3-418 shows the result of double compare of all pixels with the two defined thresholds. All pixels with an intensity value inside the range defined by the following two thresholds are highlighted in white.

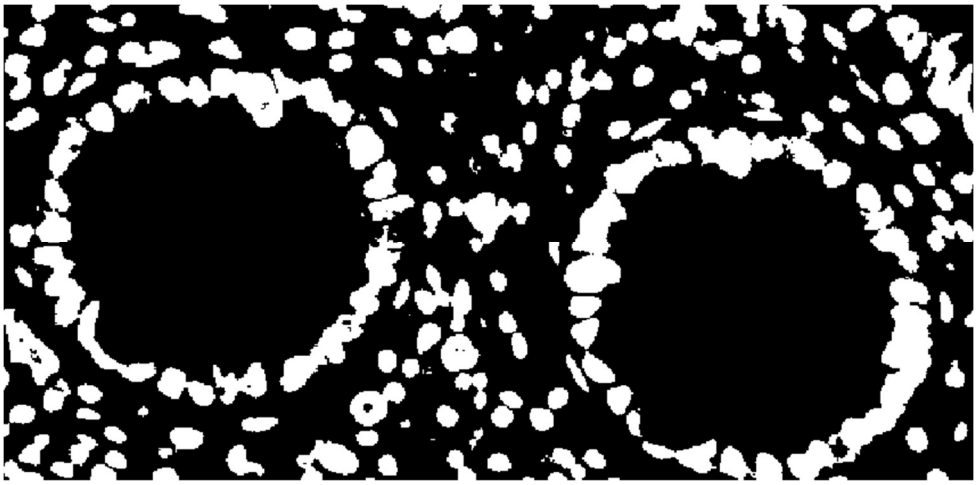


Figure 432 – Result of Compare - inside interval engine

Compare - inside interval (RGB)

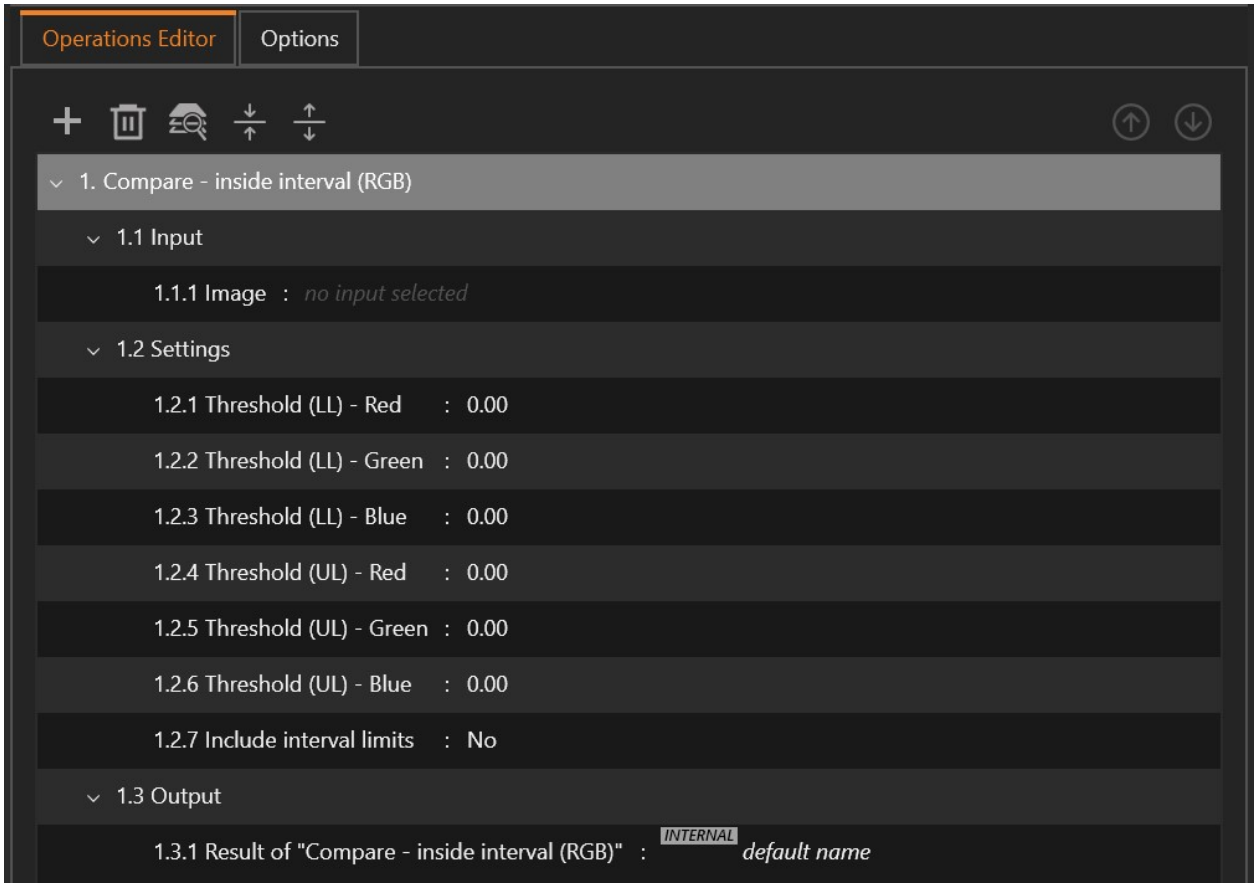


Figure 433 – Compare RGB Inside Interval

Where it can be found:

Basic Operations Module ► Threshold and Compare ► Compare - inside interval (RGB)

Description:

Compare -inside interval (RGB) is a compare operation available in Basic Operations Module.

This operation compares pixels of each channel of the source image with the limits of a given range and places the result in the output. If the result of the compare is true, meaning all pixel values of all channels are inside their interval, the corresponding pixel of the output is set to 255 (white). Otherwise, it is set to 0 (black).

Example for Include interval limits = “No”:

$$Result(x,y) = \begin{cases} 255, & Threshold\ (LL)\ \langle \boldsymbol{r}|\boldsymbol{g}|\boldsymbol{b} \rangle\ <\ Image(x,y)\ \langle \boldsymbol{r}|\boldsymbol{g}|\boldsymbol{b} \rangle\ <\ Threshold\ (UL)\ \langle \boldsymbol{r}|\boldsymbol{g}|\boldsymbol{b} \rangle \\ 0, & otherwise \end{cases}$$

The output of the operation if always a mask image.

Parameters:

- 1.1 Image - the input image
- 1.2 Threshold (LL) - Red - the lower limit of the range for red channel (default = 0);
- 1.3 Threshold (LL) - Green - the lower limit of the range for green channel (default = 0);
- 1.4 Threshold (LL) - Blue - the lower limit of the range for blue channel (default = 0);
- 1.5 Threshold (UL) - Red - the upper limit of the range for red channel (default = 0);
- 1.6 Threshold (UL) - Green - the upper limit of the range for green channel (default = 0);
- 1.7 Threshold (UL) - Blue - the upper limit of the range for blue channel (default = 0);
- 1.8 Include interval limits - switches compare mode from “<” and “>” to “≤” and “≥” (default = No)
- 1.9 Result of “…” - the result of the operation

Effect:

Generates a mask (binarization) from a color (RGB) image using two thresholds for each channel.

Example:

- Input:

The input is a brightfield color image (8-bit, 3-channel) representing a small region of an immunohistochemical colon sample.

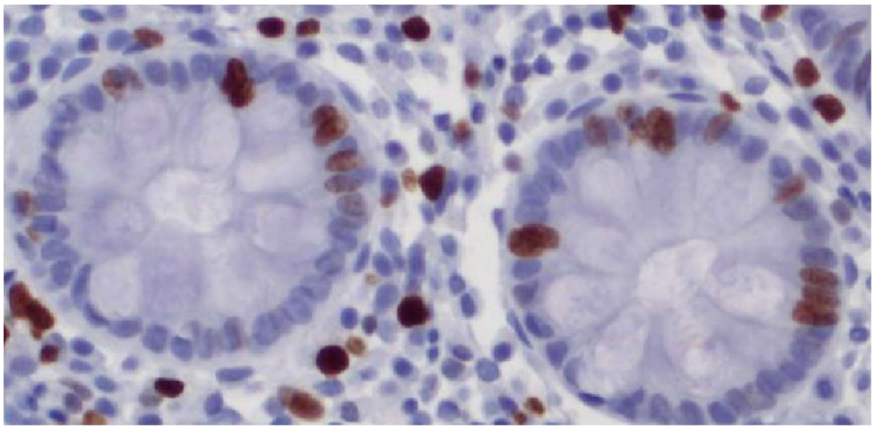


Figure 434 – Colon sample

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a mask image (8-bit, 1-channel).

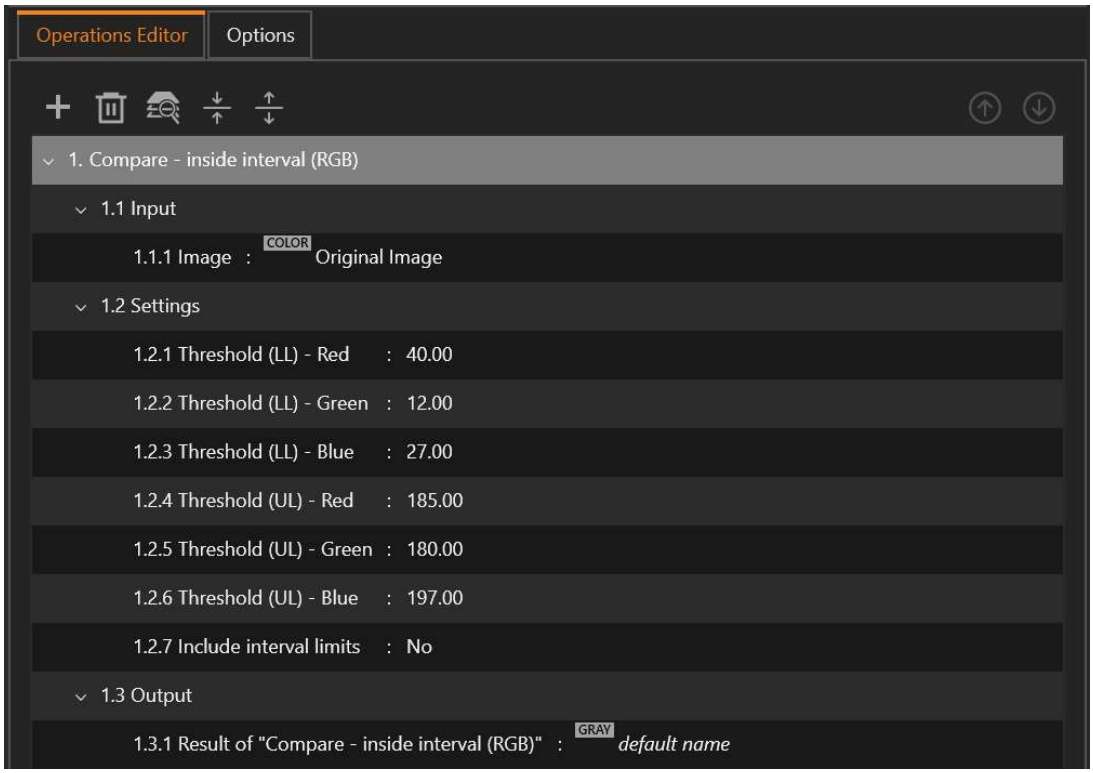


Figure 435 – Engine settings

- Output:

Figure below shows the result of double compare of all pixels with the two defined thresholds, for each channel. All pixels with an intensity value inside the range defined by the following two thresholds for each channel are highlighted in white.

$$Result(x,y) = \begin{cases} 255, & \langle \mathbf{40}|\mathbf{12}|\mathbf{27}\rangle < Image(x,y)\,\langle \boldsymbol{r}|\boldsymbol{g}|\boldsymbol{b}\rangle < \langle \mathbf{185}|\mathbf{180}|\mathbf{197}\rangle \\ 0, & otherwise \end{cases}$$

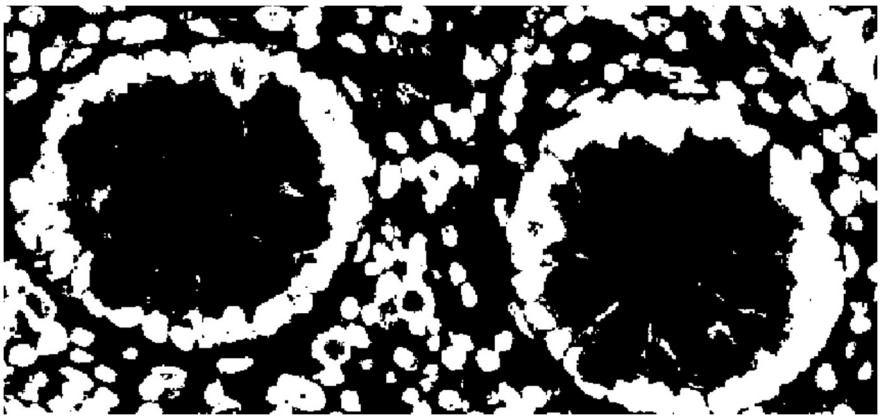


Figure 436 – Result of Compare - inside interval (RGB) engine

Compare - outside interval

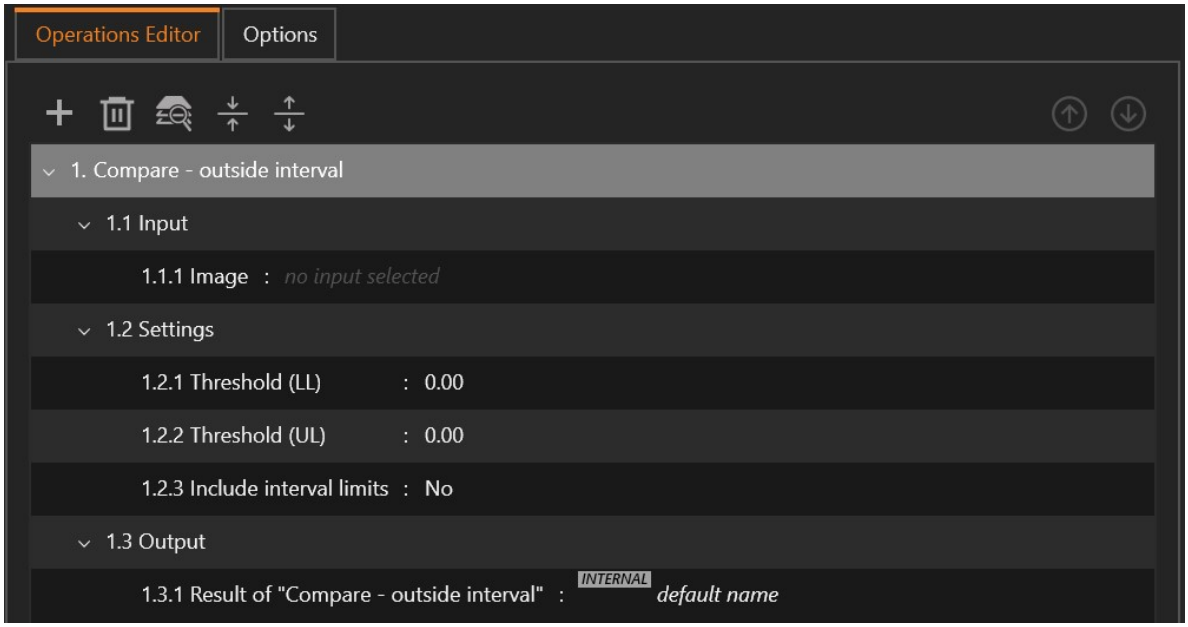


Figure 437 – Compare - outside Interval

Where it can be found:

Basic Operations Module ► Threshold and Compare ► Compare -outside interval

Description:

Compare -outside interval is a compare operation available in Basic Operations Module.

This operation compares pixels of the source image with the limits of a given range and places the result in the output. If the result is true, meaning the pixel value is outside the range, the corresponding pixel of the output is set to 255 (white). Otherwise, it is set to 0 (black).

Example for Include interval limits = “Yes”:

$$Result(x,y) = \begin{cases} 255, & Image(x,y) \leq Threshold\ (LL) \\ 255, & Image(x,y) \geq Threshold\ (UL\ ) \\ 0, & otherwise \end{cases}$$

The output of the operation is always a mask image.

Parameters:

- 1.1 Image - the input image
- 1.2 Threshold (LL) - the lower limit of the range (default = 0)
- 1.3 Threshold (UL) - the upper limit of the range (default = 0)
- 1.4 Include interval limits - switches from “<” and “>” to “≤” and “≥” (default = No)
- 1.5 Result of “…” - the result of the operation

Effect:

Generates a mask (binarization) from a grayscale image using two thresholds.

Example:

- Input:

The input is a grayscale image (8-bit, 1-channel) representing the combined information from Hematoxylin and Ki67 markers, in an immunohistochemical small region of a colon sample. The Hematoxylin and Ki67 images were generated using the Color separation engine.

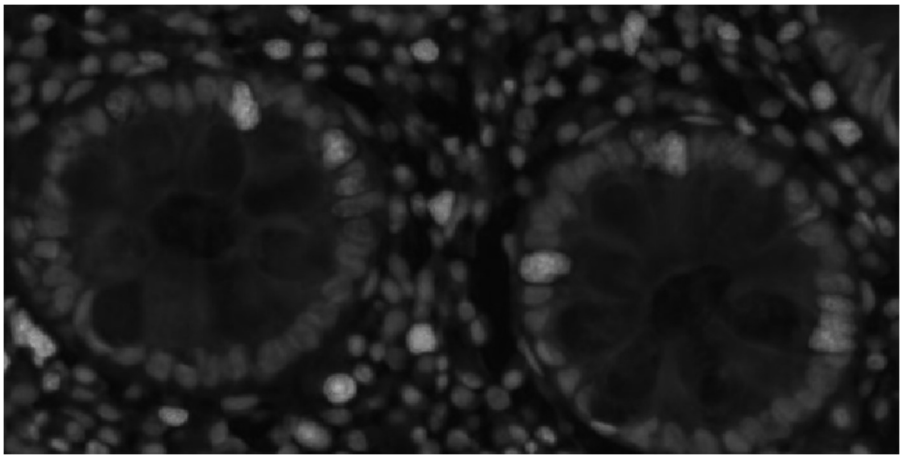


Figure 438 – Colon sample

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a mask image (8-bit, 1-channel).

Figure 439 – Engine settings

- Output:

Figure below shows the result of a double compare of all pixels with the two defined thresholds. All pixels with an intensity value outside the range defined by the following two thresholds are highlighted in white.

$$Result(x,y) = \begin{cases} 255, & Image(x,y) < 35 \\ 255, & Image\ (x,y)\ > 150 \\ 0, & otherwise \end{cases}$$

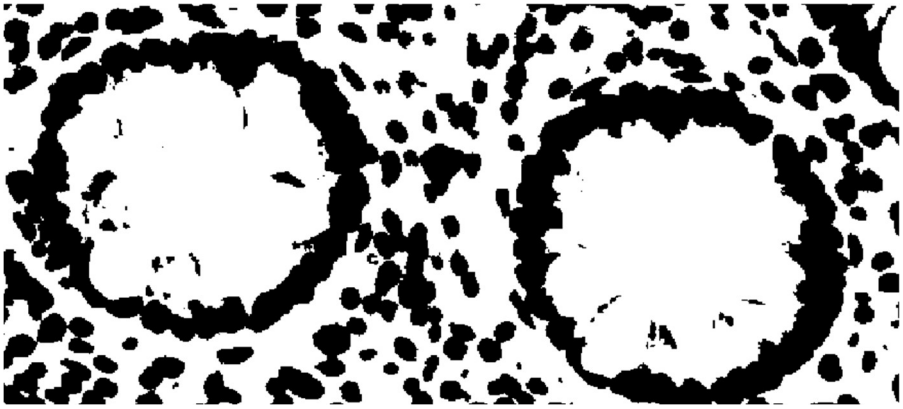


Figure 440 – Result of Compare - outside interval engine

Compare - outside interval (RGB)

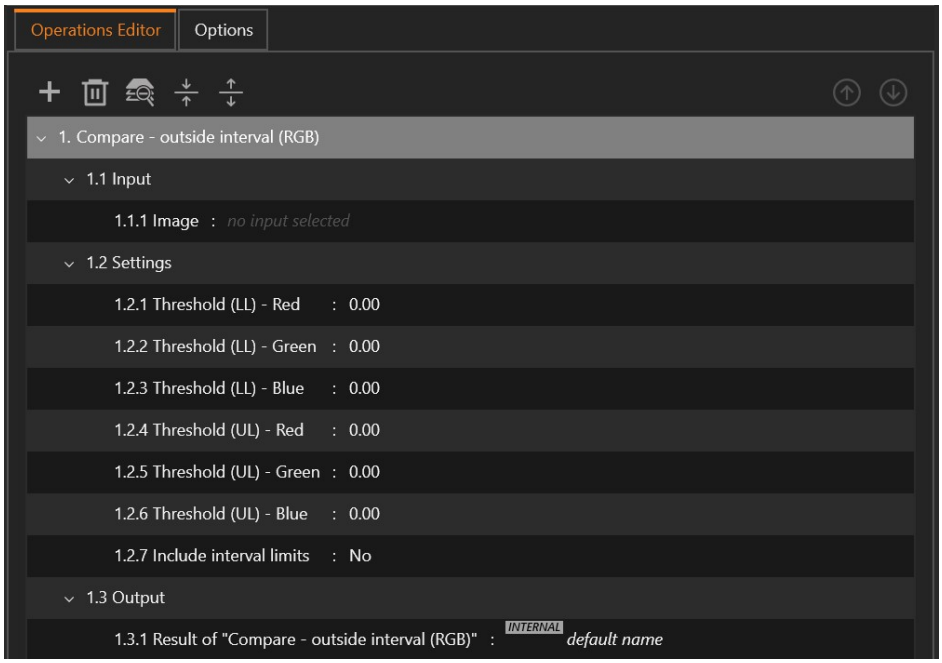


Figure 441 – Compare RGB Outside Interval

Where it can be found:

Basic Operations Module ► Threshold and Compare ► Compare -outside interval (RGB)

Description:

Compare -outside interval (RGB) is a compare operation available in Basic Operations Module.

This operation compares pixels of each channel of the source image with the limits of a given range and places the result in the output. If the result of the compare operation is true, meaning all pixel values of all channels are outside their interval, the corresponding pixel of the output is set to 255 (white). Otherwise, it is set to 0 (black).

Example for Include interval limits = “Yes”:

$$Result(x,y) = \begin{cases} 255, & Image(x,y)\,\langle \boldsymbol{r}|\boldsymbol{g}|\boldsymbol{b}\rangle \leq Threshold\,(LL)\,\langle \boldsymbol{r}|\boldsymbol{g}|\boldsymbol{b}\rangle \\ 255, & Image(x,y)\,\langle \boldsymbol{r}|\boldsymbol{g}|\boldsymbol{b}\rangle \geq Threshold\,(UL)\,\langle \boldsymbol{r}|\boldsymbol{g}|\boldsymbol{b}\rangle \\ 0, & otherwise \end{cases}$$

The output of the operation is always a mask image.

Parameters:

- 1.1 Image - the input image
- 1.2 Threshold (LL) - Red - the lower limit of the range for red channel (default = 0);
- 1.3 Threshold (LL) - Green - the lower limit of the range for green channel (default = 0);
- 1.4 Threshold (LL) - Blue - the lower limit of the range for blue channel (default = 0);
- 1.5 Threshold (UL) - Red - the upper limit of the range for red channel (default = 0);
- 1.6 Threshold (UL) - Green - the upper limit of the range for green channel (default = 0);
- 1.7 Threshold (UL) - Blue - the upper limit of the range for blue channel (default = 0);
- 1.8 Include interval limits - switches compare mode from “<” and “>” to “≤” and “≥” (default = No)
- 1.9 Result of “…” - the result of the operation

Effect:

Generates a mask (binarization) from a color (RGB) image using two thresholds for each channel.

Example:

- Input:

The input is a brightfield color image (8-bit, 3-channel) representing a small region of an immunohistochemical colon sample.

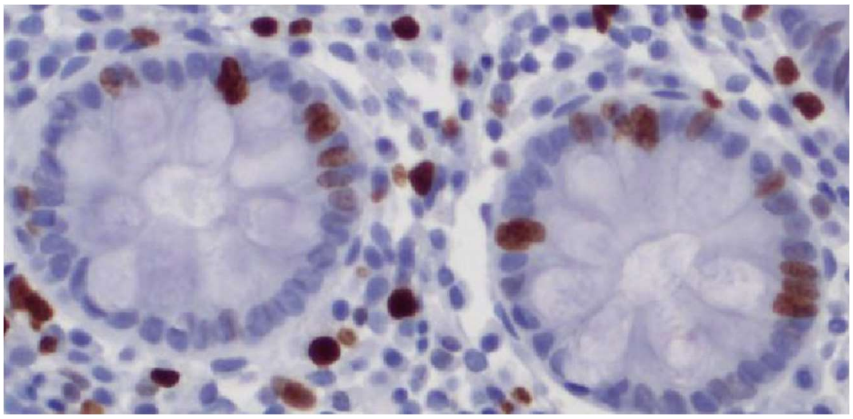


Figure 442 – Colon sample

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a mask image (8-bit, 1-channel).

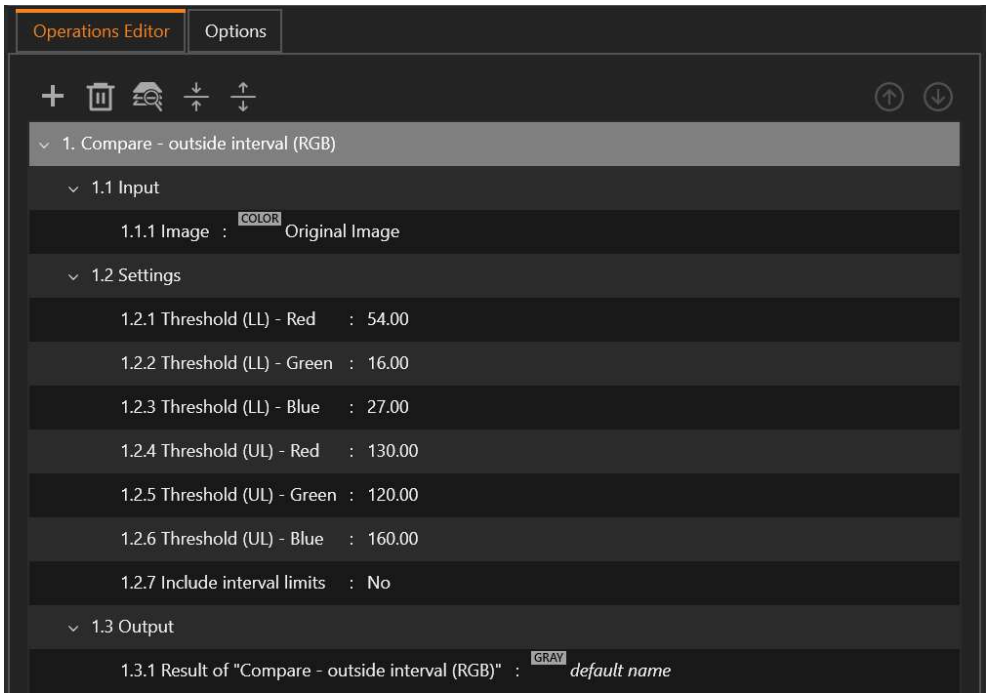


Figure 443 – Engine settings

- Output:

Figure below shows the result of double compare of all pixels with the two defined thresholds, for each channel. All pixels with an intensity value outside the range defined by the following two thresholds for each channel are highlighted in white.

$$Result\ (x,y) = \begin{cases} 255, & Image(x,y)\ \langle \boldsymbol{r}|\boldsymbol{g}|\boldsymbol{b}\rangle \leq \langle \mathbf{54}|\mathbf{16}|\mathbf{27}\rangle \\ 255, & Image(x,y)\ \langle \boldsymbol{r}|\boldsymbol{g}|\boldsymbol{b}\rangle \geq \langle \mathbf{130}|\mathbf{120}|\mathbf{160}\rangle \\ 0, & otherwise \end{cases}$$

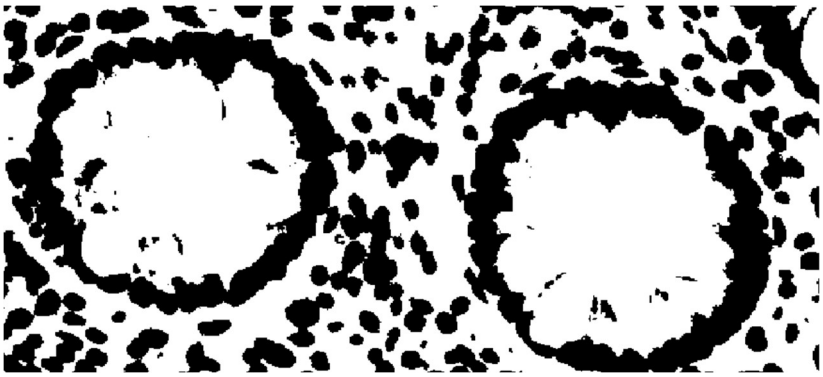


Figure 444 – Result of Compare - outside interval (RGB) engine

Otsu

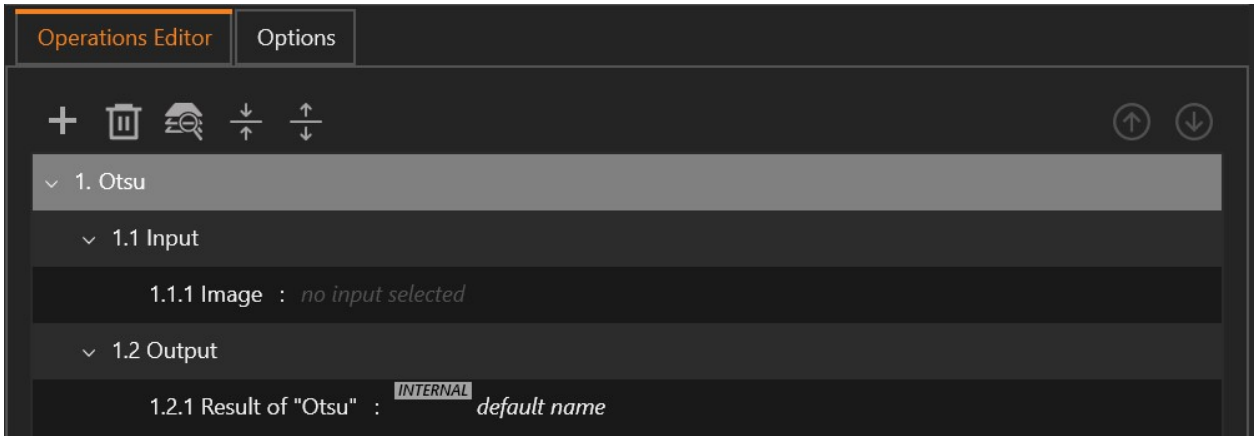


Figure 445 – Threshold Otsu

Where it can be found:

Basic Operations Module ► Threshold and Compare ► Otsu

Description:

Otsu is a threshold operation available in Basic Operations Module.

This operation performs automatic thresholding of a source image and places the results in the output. Each source pixel value is compared with a threshold level, which is automatically detected using Otsu algorithm. If the pixel value is greater than the threshold value, the corresponding pixel of the output is set to 255 (white). Otherwise, it is set to 0 (black).

$$Result(x,y) = \begin{cases} 255, & Image(x,y) > Threshold \\ 0, & otherwise \end{cases}$$

The Otsu algorithm returns a single intensity threshold that separates pixels in two classes, foreground and background. This threshold is determined by minimizing intra-class intensity variance (or by maximizing inter-class variance).

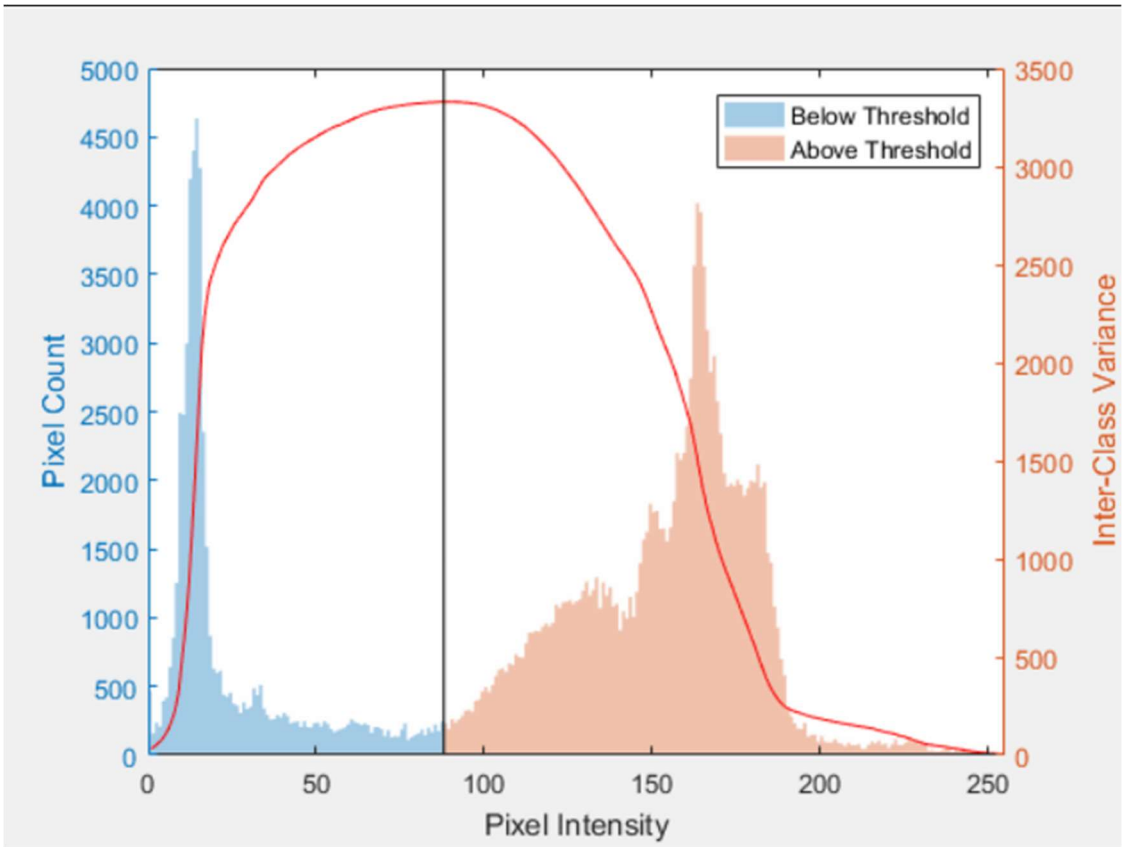


Figure 446 – Otsu threshold

Parameters:

- 1.1 Image - the input image
- 1.2 Result of "…" - the result of the operation

Effect:

Generates a mask (binarization) from a grayscale image using an automatically detected threshold value.

Example:

- Input:

Image is fluorescence grayscale image (8-bit, 1-channel) representing the SpGold channel of a colorectal cancer sample.

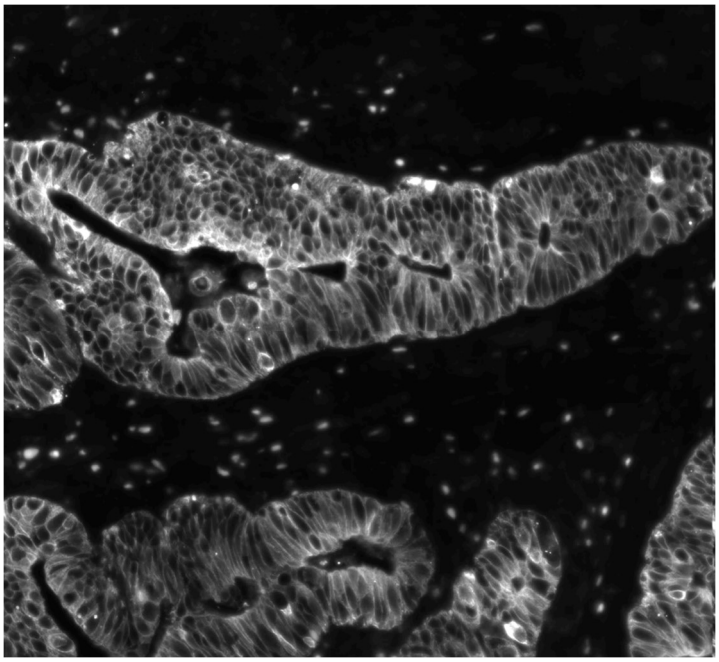


Figure 447 – Colorectal cancer sample (SpGold channel)

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a mask image (8-bit, 1-channel).

Figure 448 – Engine settings

- Output:

Figure below shows the result of the input image binarization, using automated threshold computed by the Otsu method.

Figure 449 – Result of Otsu engine

Otsu (with range)

Figure 450 – Threshold Otsu with range

Where it can be found:

Basic Operations Module ► Threshold and Compare ► Otsu (with range)

Description:

Otsu (with range) is a threshold operation available in Basic Operations Module.

This operation performs automatic thresholding of a source image and places the results in the output. Each source pixel value is compared with a threshold level, which is automatically detected using the Otsu algorithm on a defined intensity range. If the pixel value is greater than threshold value, the corresponding pixel of the output is set to 255 (white). Otherwise, it is set to 0 (black).

$$Result(x,y) = \begin{cases} 255, & Image(x,y) > Threshold \\ 0, & otherwise \end{cases}$$

The engine computes an optimal intensity value, using the Otsu algorithm, which can be used as an intensity threshold to separate the information in two classes, foreground (pixels with intensity higher than threshold) and background (pixels with intensity lower than or equal to threshold).

The threshold value is determined by minimizing the intensity variance of each of the classes.

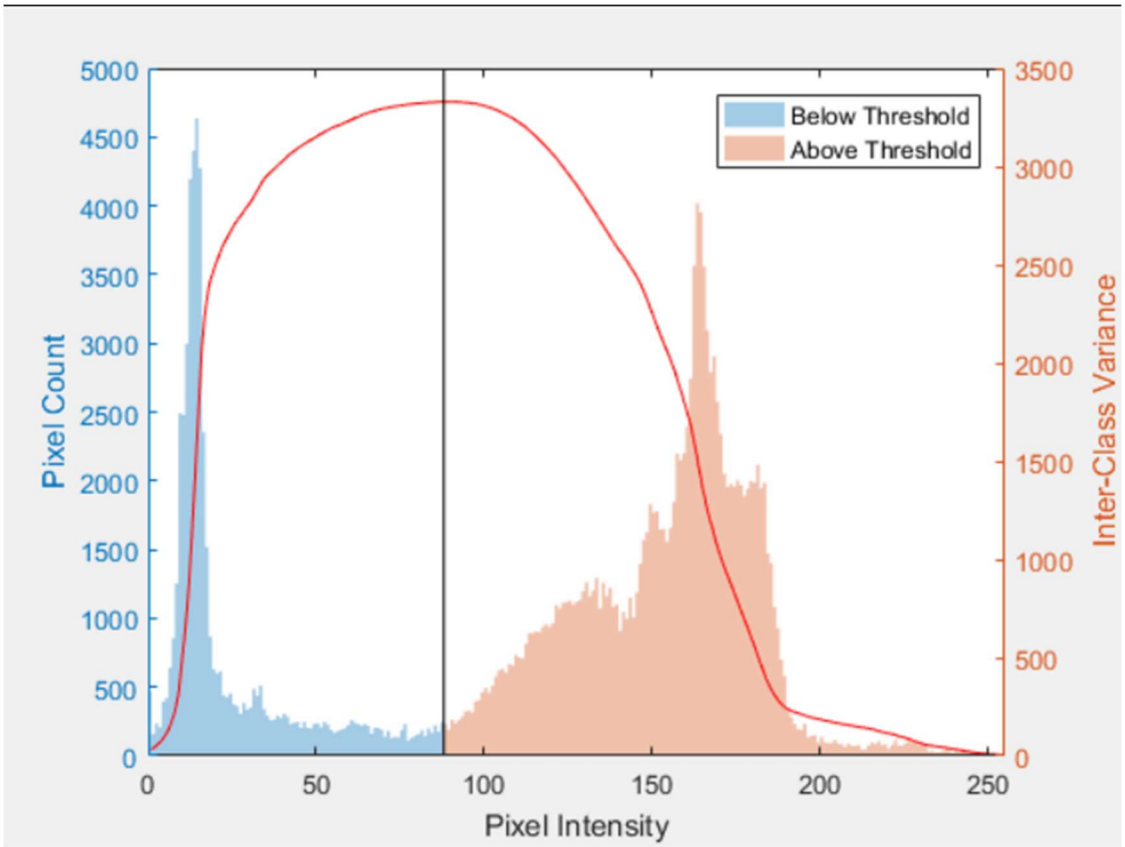


Figure 451 – Otsu threshold

The effectiveness of the threshold value returned by Otsu engine is determined by the assumption that the information can be represented by a bimodal distribution and assumed to possess a deep valley (transition values between background and foreground) between two peaks (concentrated distribution of background and foreground pixels). Noise or shading can change the intensity distribution and result in segmentation error. In this case, the intensity range used by the algorithm can be limited by Lower Limit and Upper Limit

Parameters:

- 1.1 Image - the input image
- 1.2 Lower limit - the lower limit of intensity range used by Otsu algorithm
- 1.3 Upper limit - the upper limit of intensity range used by Otsu algorithm
- 1.4 Result of "…" - the result of the operation

Effect:

Generates a mask (binarization) from a grayscale image using an automatically detected threshold value.

Example:

- Input:

Image is fluorescence grayscale image (8-bit, 1-channel) representing the SpGold channel of a colorectal cancer sample.

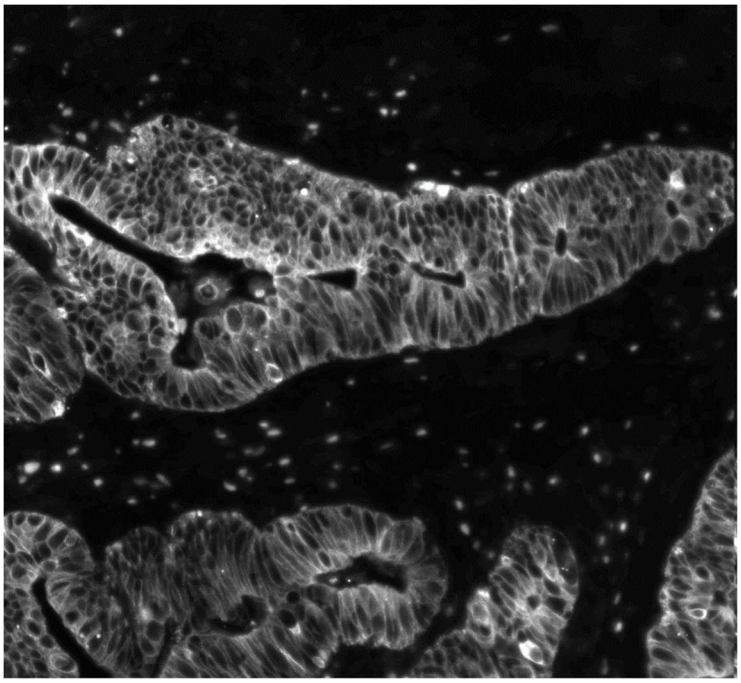


Figure 452 – Colorectal cancer sample (SpGold channel)

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a mask image (8-bit, 1-channel).

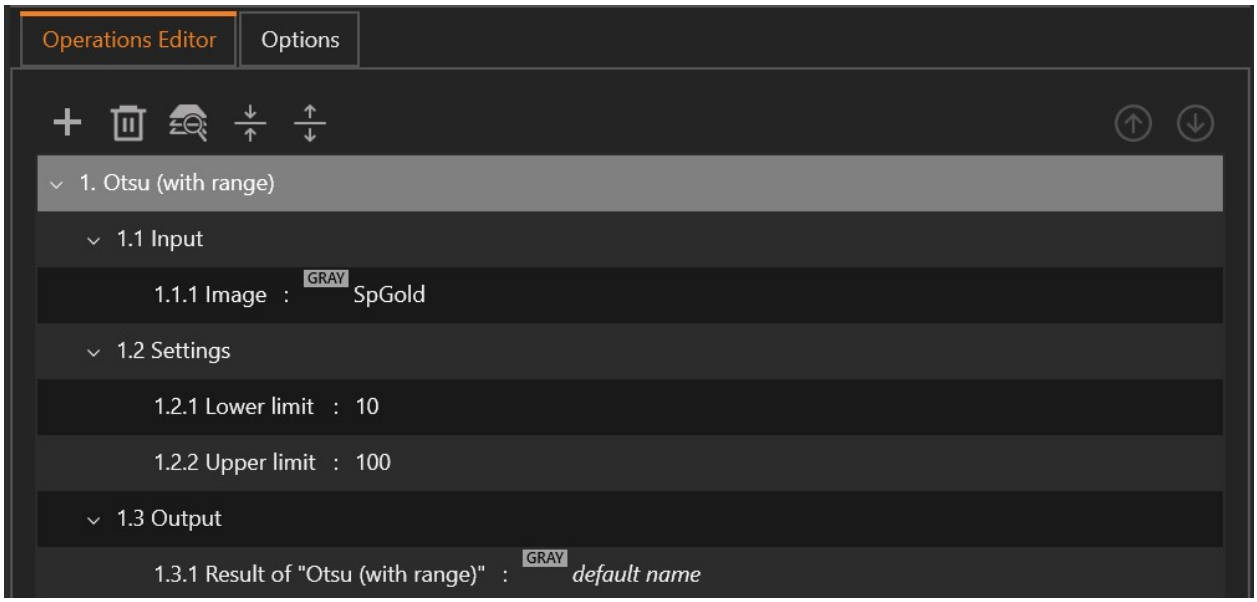


Figure 453 – Engine settings

- Output:

Figure below shows the result of the input image binarization, using an automated threshold computed by Otsu's method. The threshold value is restricted in the range defined by specified limits (lower limit, upper limit).

Figure 454 – Result of Otsu (with range) engine

Threshold

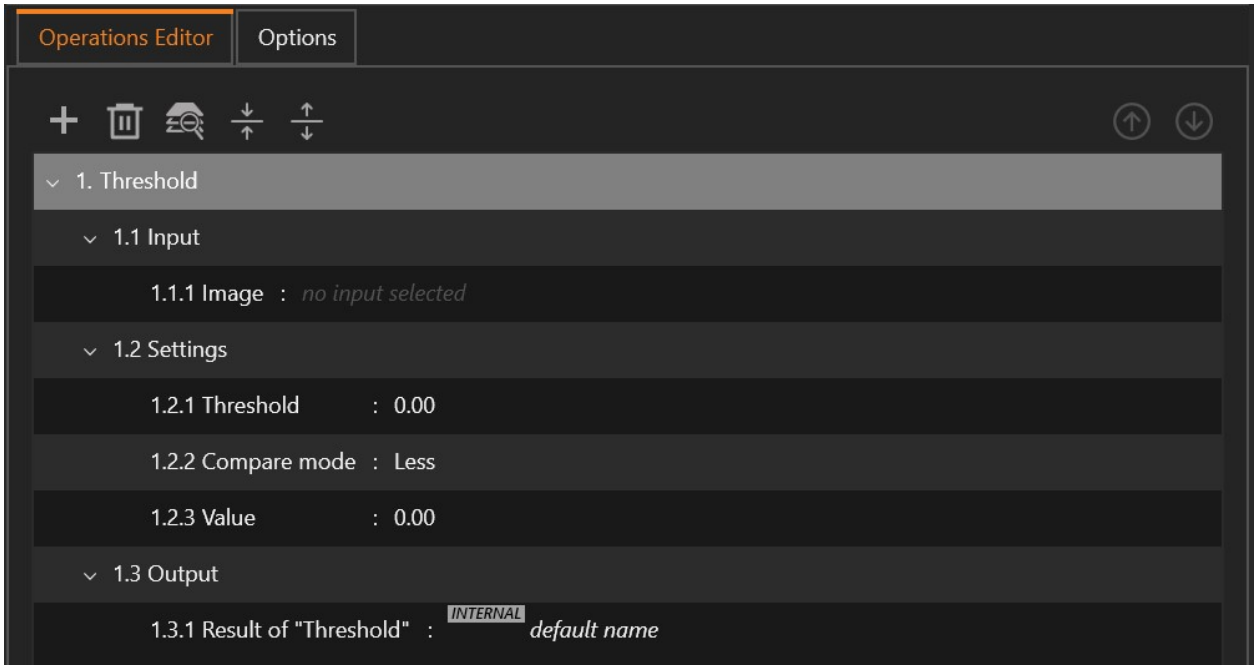


Figure 455 – Threshold

Where it can be found:

Basic Operations Module ► Threshold and Compare ► Threshold

Description:

Threshold is a threshold operation available in Basic Operations Module.

This operation performs thresholding of pixels from the source image and places the results in the output. Each source pixel value is compared with a Threshold level, using a selected Compare Mode. If the result of the compare operation is true, the corresponding pixel of the output is set to Value. Otherwise, it is set to the source pixel value.

Example for Compare Mode = “Greater”:

$$Result(x,y) = \begin{cases} Value, & Image(x,y) > Threshold \\ Image(x,y), & otherwise \end{cases}$$

Parameters:

- 1.1 Image - the input image
- 1.2 Threshold - the threshold level (default = 0);

- 1.3 Compare Mode - specifies the comparison operation to be used (default = Less). There are 2 options:
  - Less
  - Greater
- 1.4 Value - the new value for pixels meeting the condition
- 1.5 Result of "…" - the result of the operation

Effect:

Caps image values to a specified minimum or maximum value. Source pixels values not meeting the conditions are set to a new specified value.

Example:

- Input:

Image is grayscale image (8-bit, 1-channel) representing the combined information from Hematoxylin and Ki67 markers, in an immunohistochemical small region of a colon sample. The Hematoxylin and Ki67 images were generated using the Color separation engine.

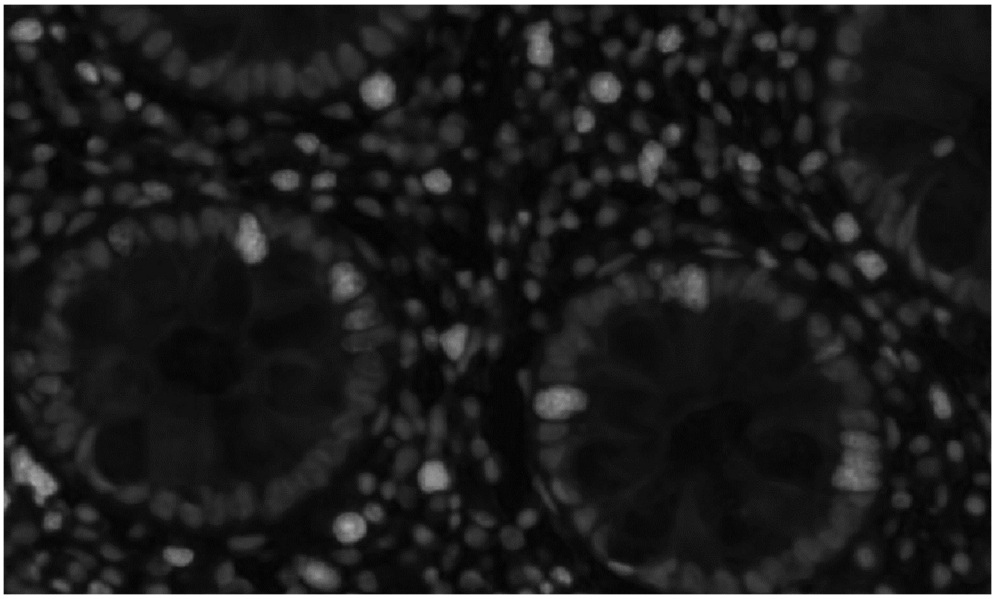


Figure 456 – Colon sample

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a grayscale image (8-bit, 1-channel), matching the input.

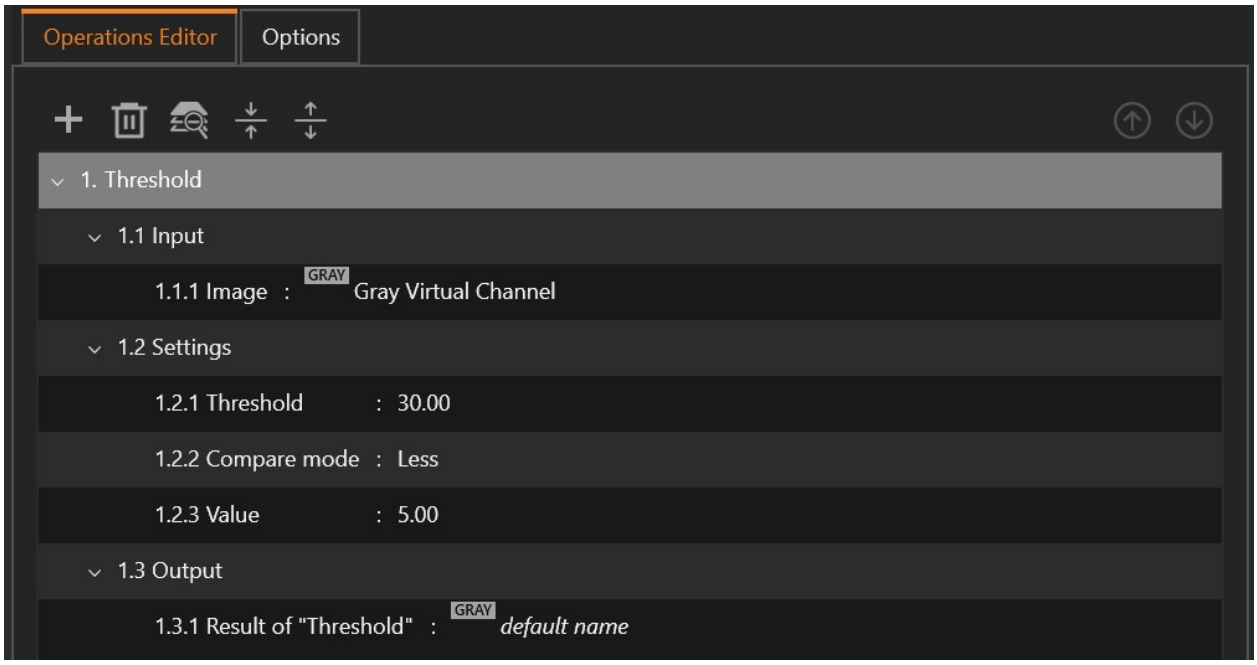


Figure 457 – Engine settings

- Output:

Figure below shows the result of thresholding the input image, using the formula.

$$Result\,(x,y) = \begin{cases} Image\,(x,y), & Image(x,y) \geq 30 \\ 5, & Image(x,y) < 30 \end{cases}$$

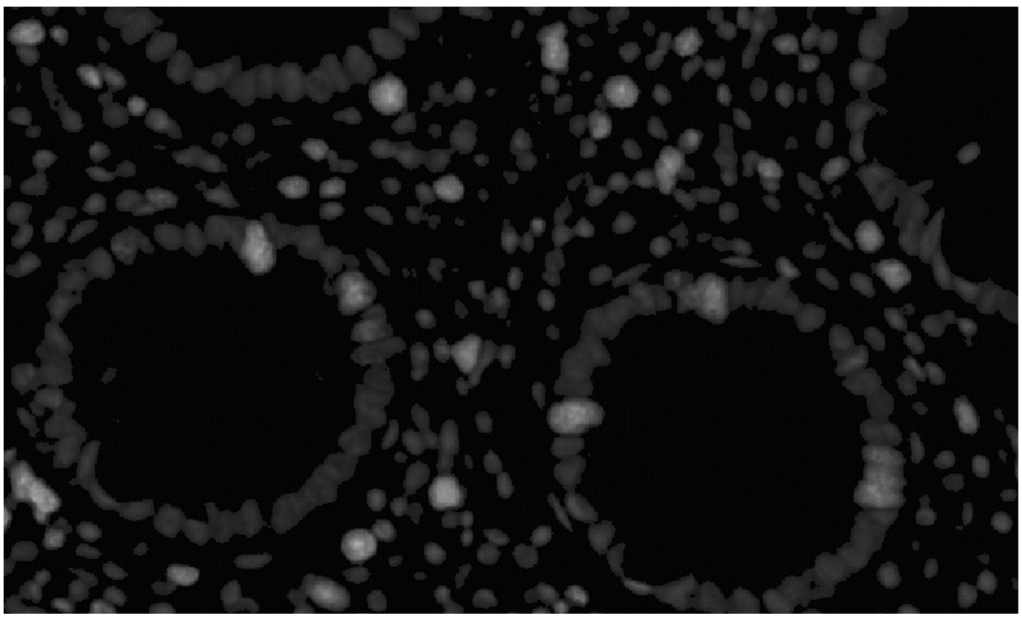


Figure 458 – Result of Threshold engine

Threshold (RGB)

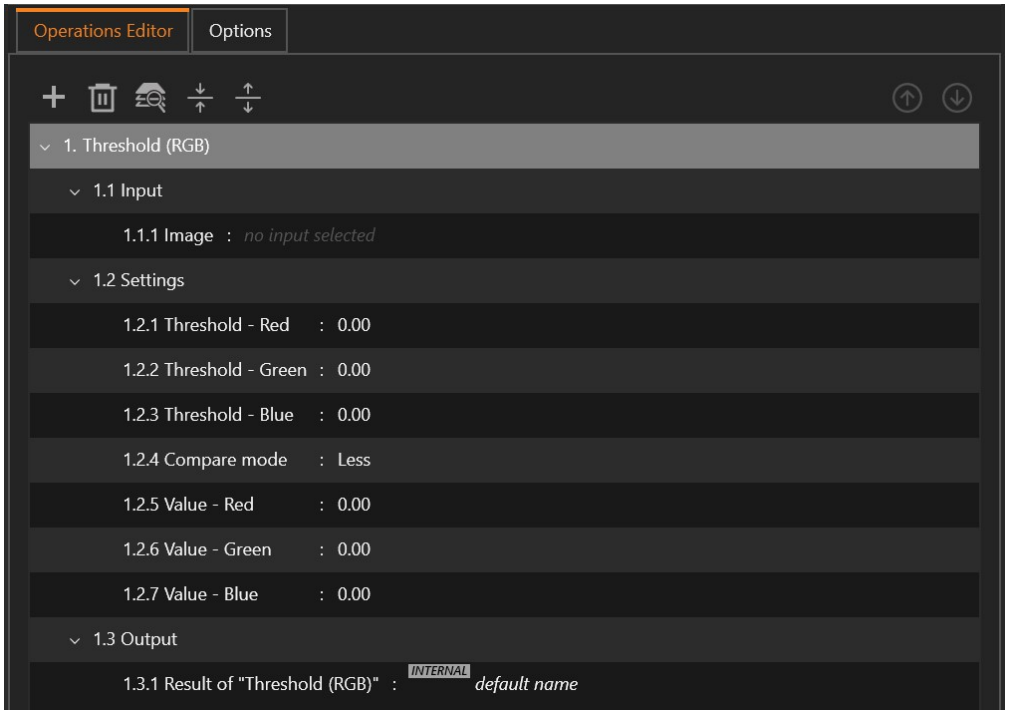


Figure 459 – Threshold (RGB)

Where it can be found:

Basic Operations Module ► Threshold and Compare ► Threshold (RGB)

Description:

Threshold (RGB) is a threshold operation available in Basic Operations Module.

This operation performs thresholding of pixels for each source image channel and places the results in the output. For each channel, the pixel value of the source image is compared with a Threshold level, using a selected Compare Mode. If the result of the compare is true, the corresponding pixel value of the output is set to Value. Otherwise, it is set to the source pixel channel value.

Example for Compare Mode = “Greater”:

$$Result(x,y,c) = \begin{cases} value(c), & Image(x,y,c) > threshold(c) \\ Image(x,y,c), & otherwise \end{cases}$$

where: x, y - represents the spatial coordinates of pixels and c is the channel.

Threshold (RGB) can be seen as the Threshold operation applied on each channel of a color (RGB) image.

Parameters:

- 1.1 Image - the input image
- 1.2 Threshold - Red - the threshold value for red channel (default = 0)
- 1.3 Threshold - Green - the threshold value for green channel (default = 0)
- 1.4 Threshold - Blue - the threshold value for blue channel (default = 0)
- 1.5 Compare Mode - specifies the comparison operation to be used (default = Less). There are 2 options:
  - Less
  - Greater
- 1.4 Value - Red - the new value for red channel pixels meeting the condition (default = 0)
- 1.4 Value - Green - the new value for green channel pixels meeting the condition (default = 0)
- 1.4 Value - Blue - the new value for blue channel pixels meeting the condition (default = 0)
- 1.5 Result of "…" - the result of the operation

Effect:

Caps values of each image channel to a specified minimum or maximum value. Source pixels values not meeting the condition are set to a new specified value.

Example:

- Input:

Image is brightfield color image (8-bit, 3-channel) representing a small region of an immunohistochemical colon sample.

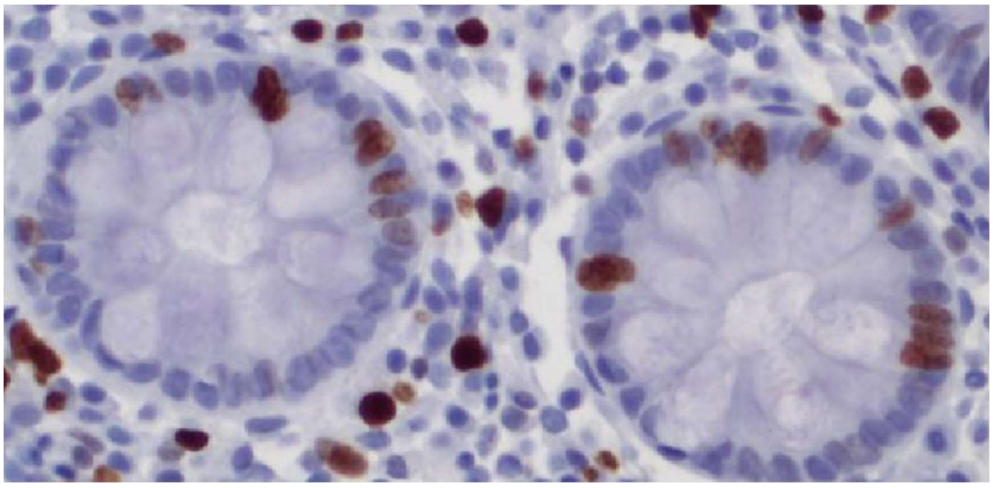


Figure 460 – Colon sample

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a color image (8-bit, 3-channel), matching the input.

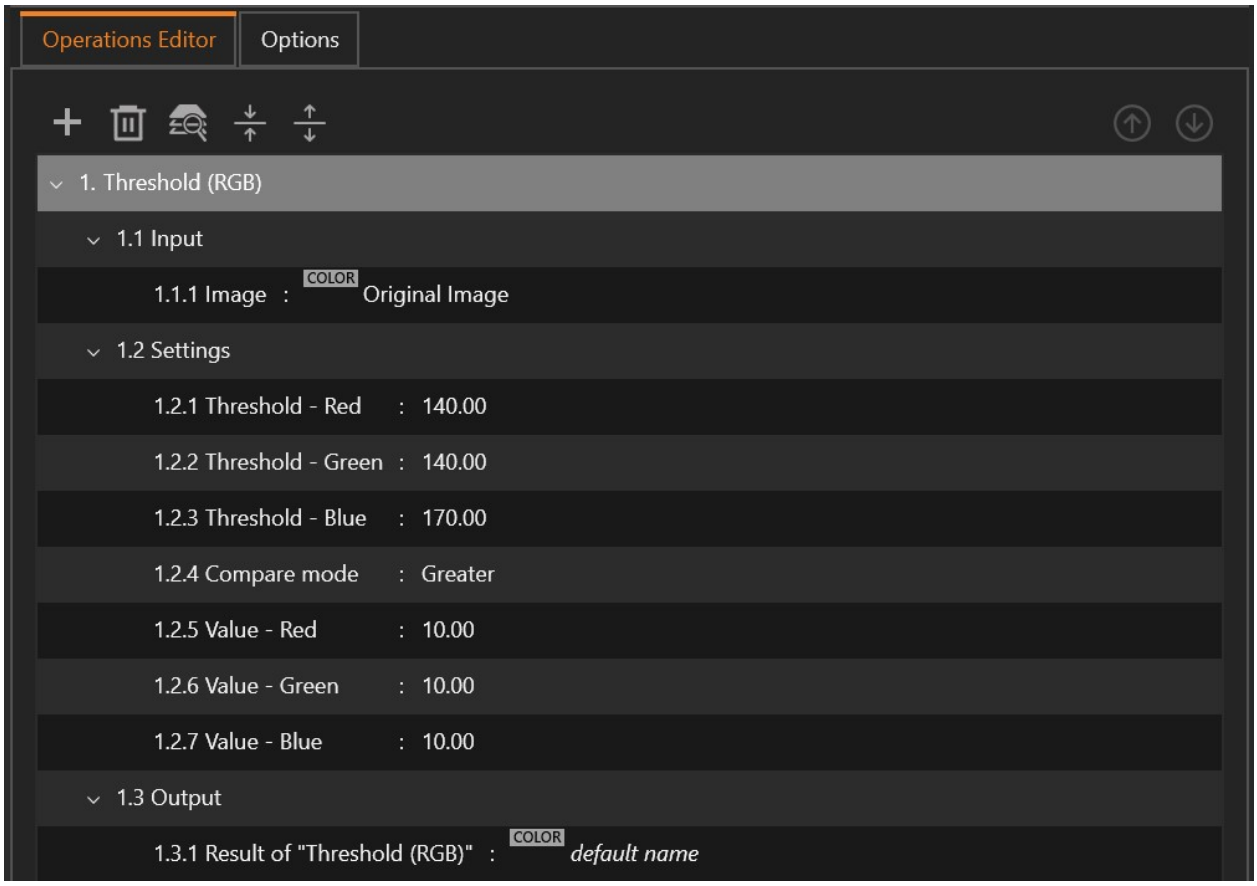


Figure 461 – Engine settings

- Output:

Figure below shows the result of thresholding the input image, using the formula:

$$Result\ (x,y,red) = \begin{cases} Image\ (x,y,red), & Image(x,y,red) \geq 140 \\ 10, & Image(x,y,red) < 140 \end{cases}$$

$$Result\ (x,y,green) = \begin{cases} Image\ (x,y,green), & Image(x,y,green) \geq 140 \\ 10, & Image(x,y,green) < 140 \end{cases}$$

$$Result\ (x,y,blue) = \begin{cases} Image\ (x,y,blue), & Image(x,y,blue) \geq 170 \\ 10, & Image(x,y,blue) < 170 \end{cases}$$

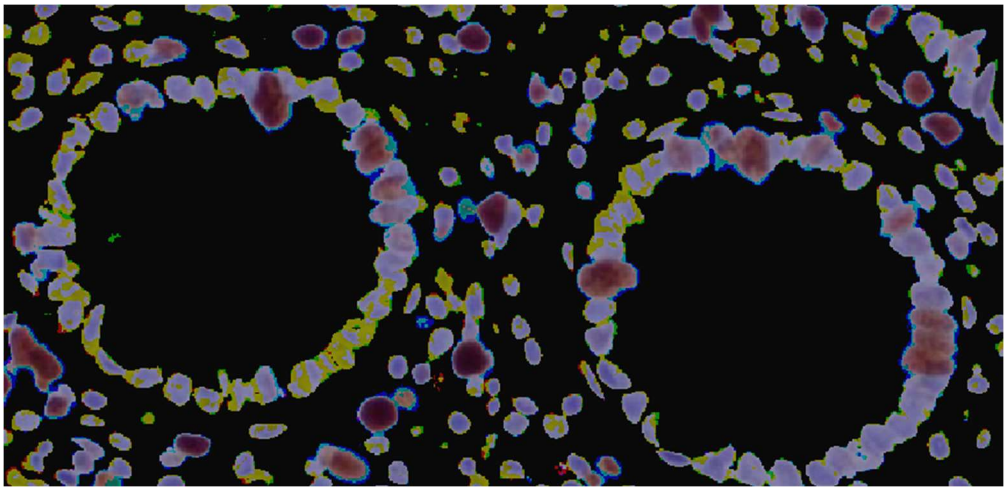


Figure 462 – Result of Threshold (RGB) engine

Threshold - LT, GT

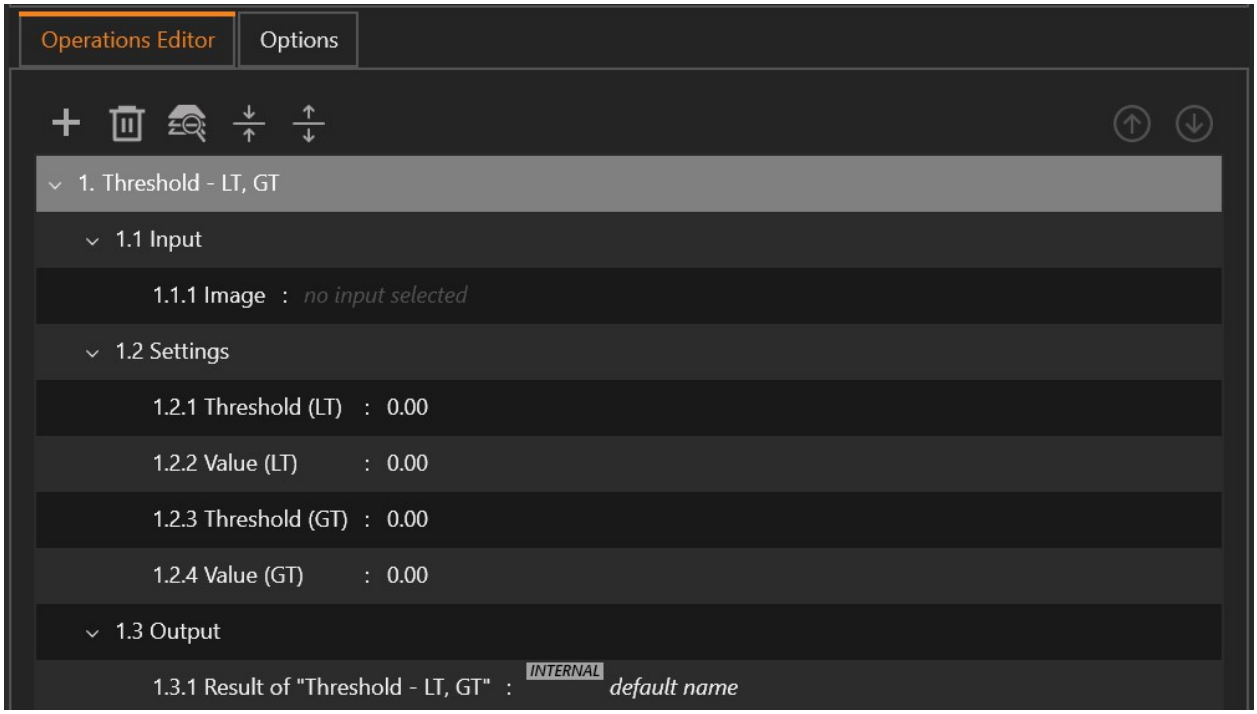


Figure 463 – Threshold - LT, GT

Where it can be found:

Basic Operations Module ► Threshold and Compare ► Threshold -LT, GT

Description:

Threshold -LT, GT is a threshold operation available in Basic Operations Module.

This operation performs double thresholding of pixels from the source image and places the results in the output. If the source pixel value is lower than Threshold (LT), the corresponding output pixel is set to Value (LT). If the source pixel value is higher than Threshold (GT), the corresponding output pixel is set to Value (GT). Otherwise, it is set to the source pixel value.

$$Result(x,y) = \begin{cases} Value\ (LT), & Image(x,y) \leq Threshold\ (LT) \\ Value\ (GT), & Image(x,y) \geq Threshold\ (GT) \\ Image(x,y), & otherwise \end{cases}$$

Required condition:

- Threshold (LT) ≤ Threshold (GT)

Parameters:

- 1.1 Image - the input image
- 1.2 Threshold (LT) - the lower threshold level (default = 0);
- 1.3 Value (LT) - the new value for pixels lower than Threshold (LT)
- 1.4 Threshold (GT) - the upper threshold level (default = 0);
- 1.4 Value (GT) - the new value for pixels greater than Threshold (GT)
- 1.5 Result of "…" - the result of the operation

Effect:

Caps image values to a specified minimum and maximum value. Source pixels values outside the range are set to new specified values.

Example:

- Input:

The input is a grayscale image (8-bit, 1-channel) representing the combined information from Hematoxylin and Ki67 markers, in an immunohistochemical small region of a colon sample. The Hematoxylin and Ki67 images were generated using the Color separation engine.

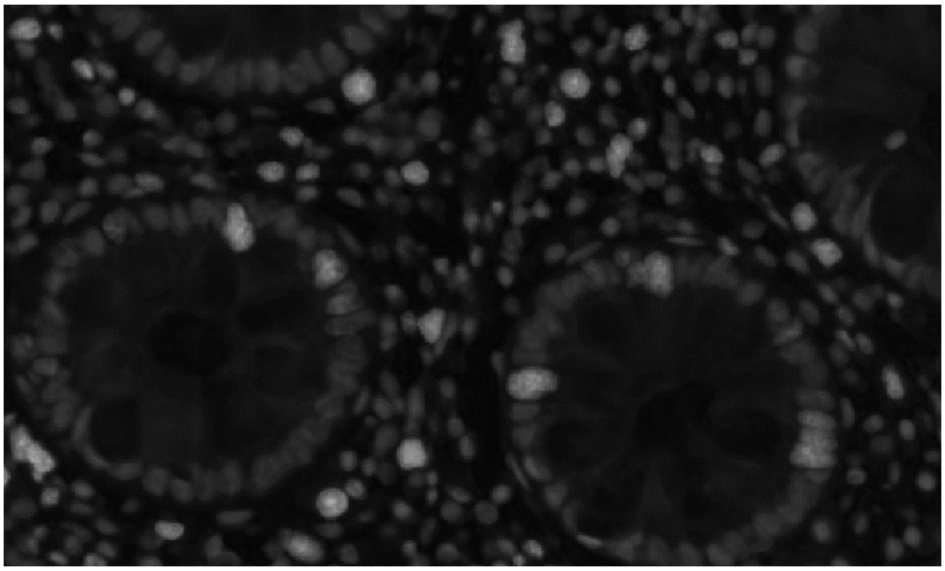


Figure 464 – Colon sample

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a grayscale image (8-bit, 1-channel), matching the input.

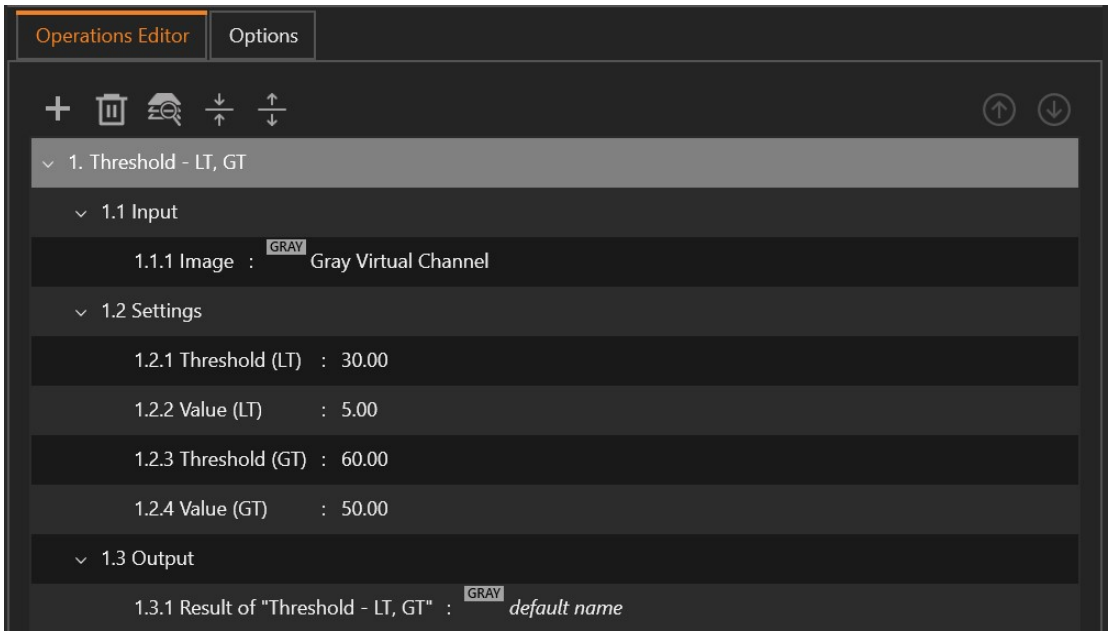


Figure 465 – Engine settings

- Output:

Figure below shows the result of a double thresholding the input image, using the formula:

$$Result\ (x,y) = \begin{cases} Image\ (x,y), & 30 < Image(x,y) < 60 \\ 5, & Image(x,y) \leq 30 \\ 50, & Image\ (x,y) \geq 60 \end{cases}$$

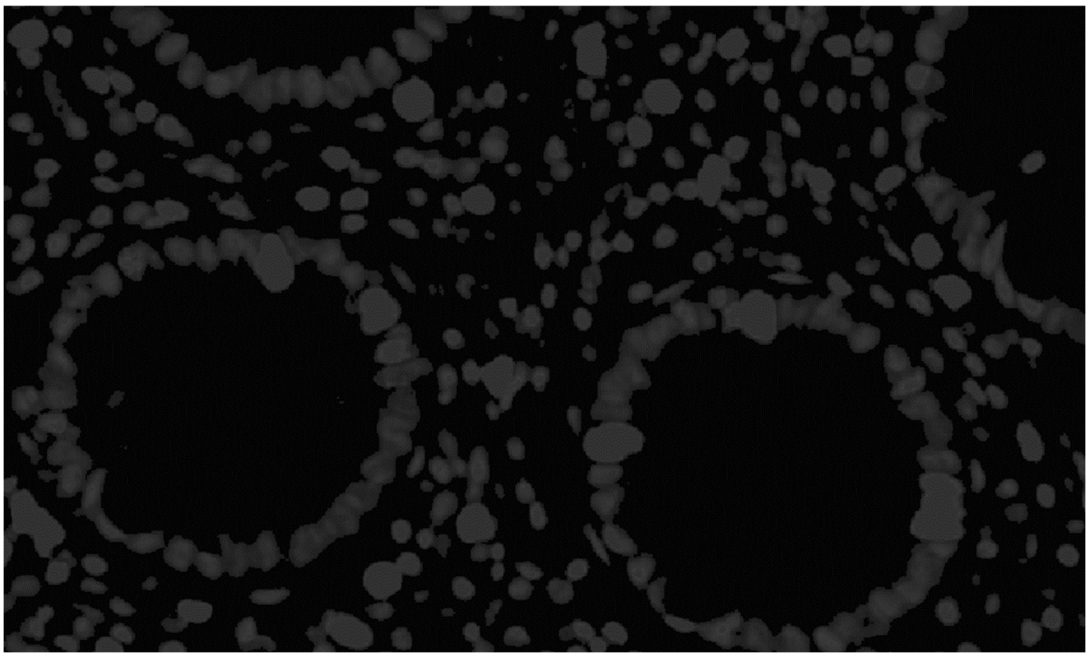


Figure 466 – Result of Threshold - LT, GT engine

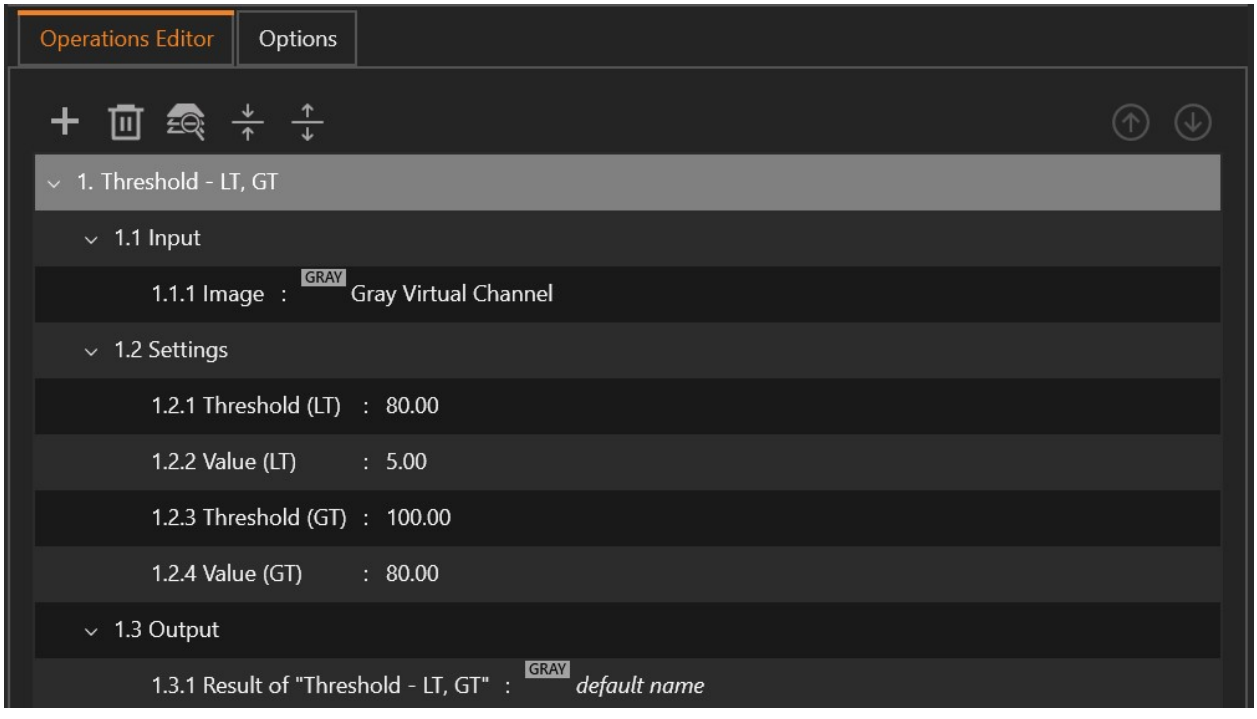


Figure 467 – Engine settings

In Figure A we have a gray image composed of light objects on a dark background. Figure B shows the results after we applied the following values for the parameters: “LowerTTresh”=80, “LowerTValue”=5, “GreaterTTresh”=100, and “GreaterTValue”=80. The pixels that are smaller than 80 will be set on 5, and the pixels that are greater than 100 will be set on 80. Others will remain with their value unchanged.

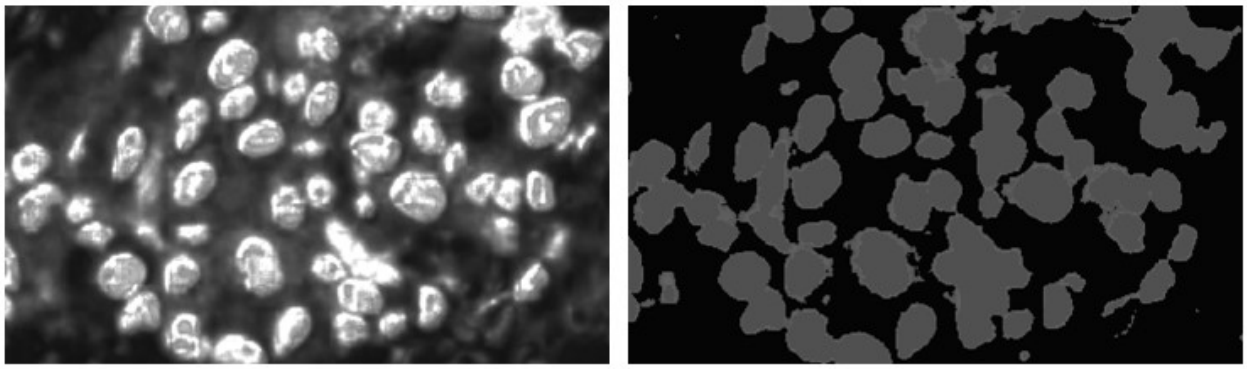


Figure A

Figure B

Figure 468 – Figure A and Figure B

Threshold - LT, GT (RGB)

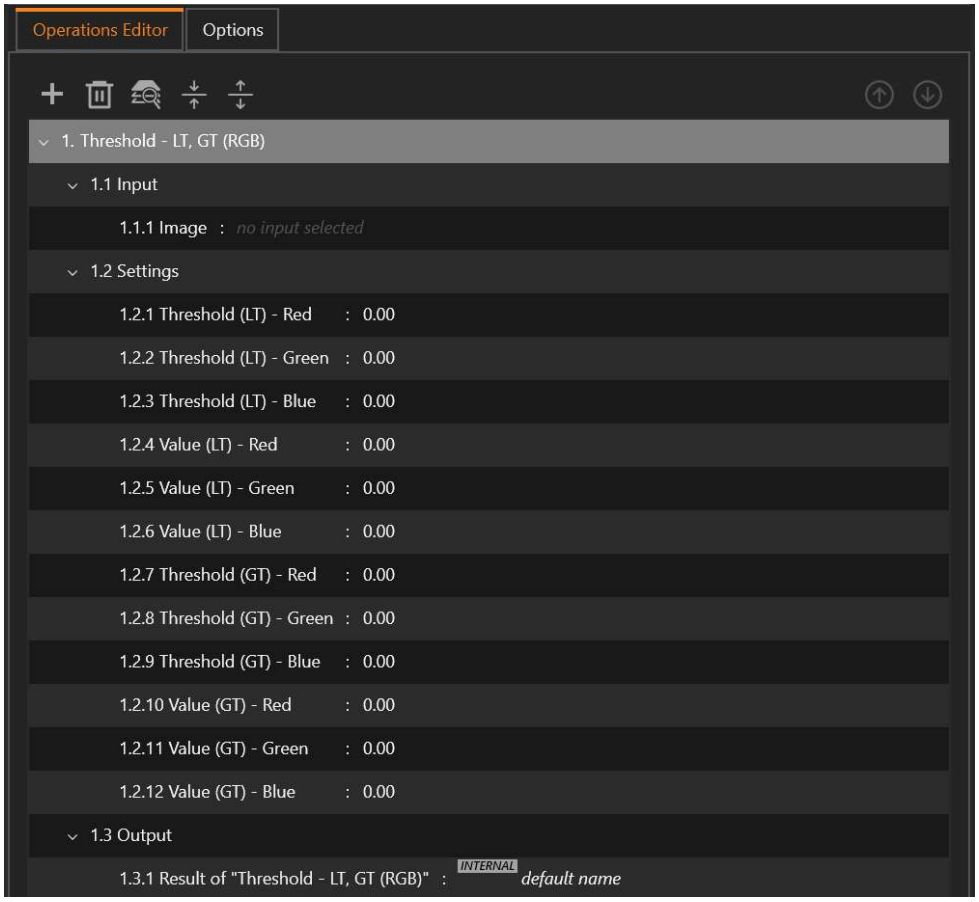


Figure 469 – Threshold LT, GT (RGB) LTValGTVal RGB

Where it can be found:

Basic Operations Module ► Threshold and Compare ► Threshold -LT, GT (RGB)

Description:

Threshold - LT, GT (RGB) is a threshold operation available in Basic Operations Module.

This operation performs double thresholding for pixels of each source image channel and places the results in the output. For each channel, if the pixel value of the source image is lower than Threshold (LT), the corresponding output pixel value is set to Value (LT). If the pixel value is higher than Threshold (GT), the corresponding output pixel is set to Value (GT). Otherwise, it is set to the source pixel value.

$$Result(x,y,c) = \begin{cases} Value\ (LT), & Image(x,y,c) \leq Threshold\ (LT) \\ Value\ (GT), & Image(x,y,c)\ \geq\ Threshold\ (GT) \\ Image(x,y,c), & otherwise \end{cases}$$

where: x, y -represents the spatial coordinates of pixels and c is the channel.

Required conditions:

- Threshold (LT) - Red ≤ Threshold (GT) -Red
- Threshold (LT) - Green ≤ Threshold (GT) - Green
- Threshold (LT) - Blue ≤ Threshold (GT) - Blue

Parameters:

- 1.1 Image - the input image
- 1.2 Threshold (LT) - Red - the lower threshold level for red channel (default = 0);
- 1.3 Threshold (LT) - Green - the lower threshold level for green channel (default = 0);
- 1.4 Threshold (LT) - Blue - the lower threshold level for blue channel (default = 0);
- 1.5 Value (LT) -Red - the new value for pixels lower than Threshold (LT) - Red
- 1.6 Value (LT) -Green - the new value for pixels lower than Threshold (LT) - Green
- 1.7 Value (LT) -Blue - the new value for pixels lower than Threshold (LT) - Blue
- 1.8 Threshold (GT) - Red - the upper threshold level for red channel (default = 0);
- 1.9 Threshold (GT) - Green - the upper threshold level for green channel (default = 0);
- 1.10 Threshold (GT) - Blue - the upper threshold level for blue channel (default = 0);

- 1.11 Value (GT) -Red - the new value for pixels greater than Threshold (GT) - Red
- 1.12 Value (GT) -Green - new value for pixels greater than Threshold (GT) - Green
- 1.13 Value (GT) -Blue - the new value for pixels greater than Threshold (GT) - Blue
- 1.14 Result of "…" - the result of the operation

Effect:

Caps the values of each image channel to a specified minimum and maximum value. Source pixels values outside the range are set to new specified values.

Example:

- Input:

The input is a brightfield color image (8-bit, 3-channel) representing a small region of an immunohistochemical colon sample.

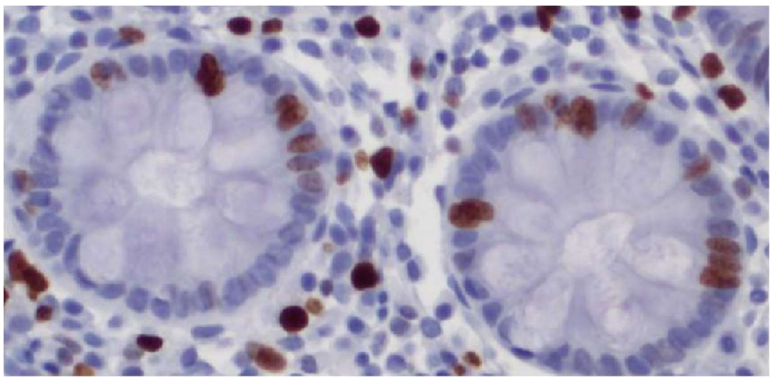


Figure 470 – Colon sample

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a color image (8-bit, 3-channel), matching the input.

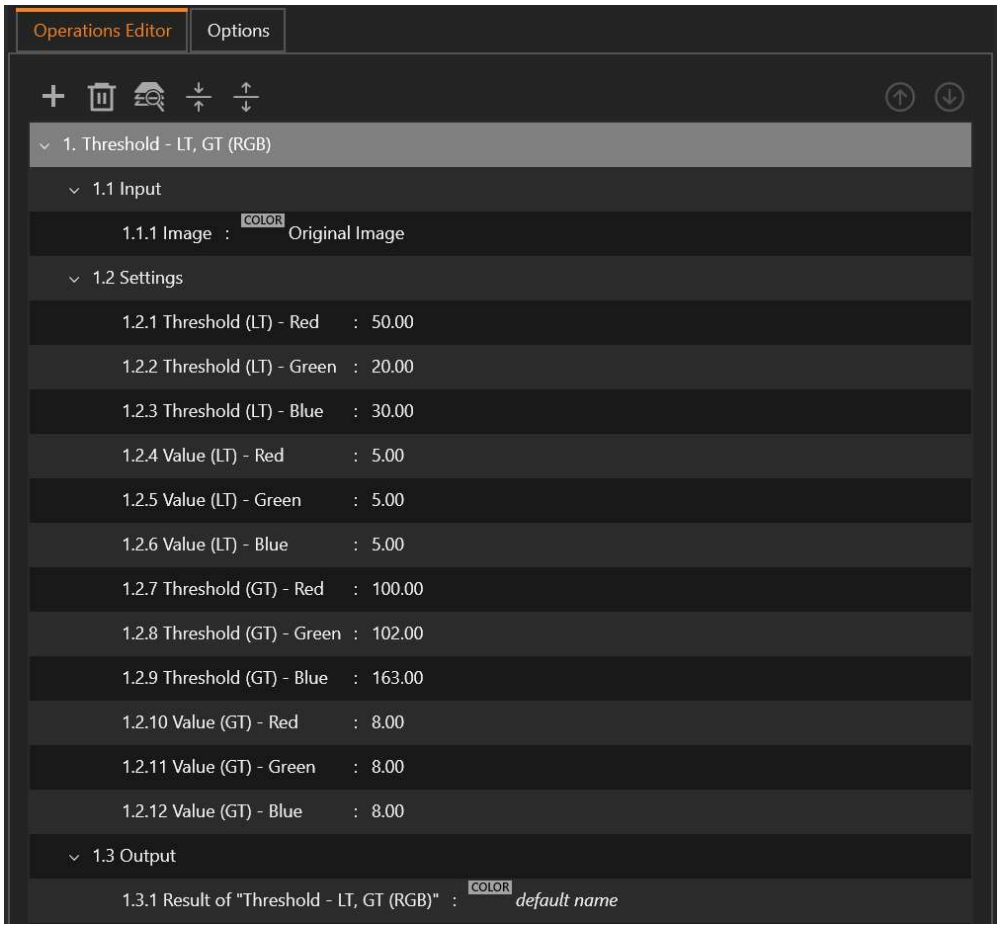


Figure 471 – Engine settings

- Output:

Figure below shows the result of double thresholding the input image, using the formula:

$$Result\ (x,y,red) = \begin{cases} Image\ (x,y,red), & 50 < Image(x,y,red) < 100 \\ 5, & Image(x,y,red) \leq 50 \\ 8, & Image\ (x,y,red) \geq 60 \end{cases}$$

$$Result\ (x,y,green) = \begin{cases} Image\ (x,y,green), & 20 < Image(x,y,green) < 102 \\ 5, & Image(x,y,green) \leq 30 \\ 8, & Image\ (x,y,green) \geq 60 \end{cases}$$

$$Result\ (x,y) = \begin{cases} Image\ (x,y,blue), & 30 < Image(x,y,blue) < 163 \\ 5, & Image(x,y,blue) \leq 30 \\ 8, & Image\ (x,y,blue) \geq 60 \end{cases}$$

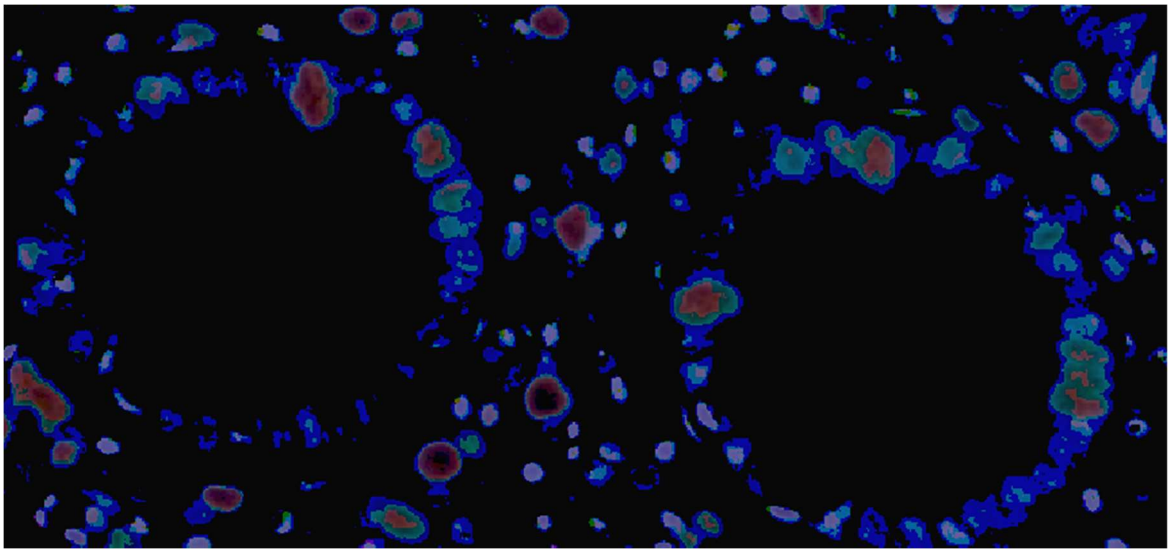


Figure 472 – Result of Threshold - LT, GT (RGB) engine

5.12. Legacy

Add weighted images (v1)

Figure 473 – Add Weighted Images

Where it can be found:

Basic Operations Module ► Legacy ► Add weighted images (v1)

Description:

Add weighted images (v1) engine is deprecated.

This engine was not completely removed for reasons of compatibility with older projects.

The new version can be found at:

Basic Operations Module ► Arithmetic ► Add weighted images

Parameters:

- 1.1 Image 1 - the first input image
- 1.2 Image2 - the second input image
- 1.3 Weight 1 - the weight value for the first image (default = 0)
- 1.4 Weight 2 - the weight value for the second image (default = 0)
- 1.5 Result of "…" - the result of the operation

Effect:

- Combines the information from two images.

Anisotropic Filter (v1)

Figure 474 – Anisotropic filter

Where it can be found:

Basic Operations Module ► Legacy ► Anisotropic filter (v1)

Description:

Anisotropic filter (v1) engine is deprecated.

This engine was not completely removed for reasons of compatibility with older projects.

The new version can be found at:

Basic Operations Module ► Filters ► Anisotropic filter

Parameters:

- 1.1 Image - the input image
- 1.2 Diffusion Scheme - specifies the diffusion scheme used (default = Rotation Invariant)
- 1.3 Diffusion time - the total diffusion time. Using a small value may leave unfiltered some noise artifacts, while using a high value may results an image that have too much blur (default = 5);
- 1.4 Diffusion time step - the diffusion time step size (default = 5);
- 1.5 Std (for image smoothing) - the standard deviation of the gaussian smoothing before calculation of the image Hessian (default = 1);
- 1.6 Std (for Hessian smoothing) - the standard deviation of the gaussian smoothing of the Hessian (default = 1);
- 1.10 Result of "…" - the result of the operation

Effect:

Reduces the noise in the image while preserving details and smooth along image edges removing gaps due to noise.

Graph from labels (v1)

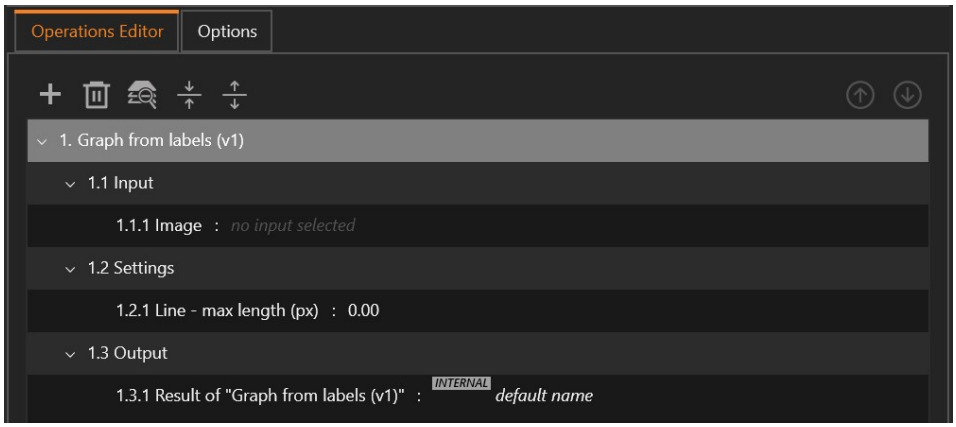


Figure 475 – Graph from Labels (v1)

Where it can be found:

Basic Operations Module ► Legacy ► Graph from labels (v1)

Description:

Graph from labels (v1) is a deprecated operation.

This operation was not completely removed for reasons of compatibility with older projects.

The new version can be found at:

Basic Operations Module ► Labels ► Graph from labels

Parameters:

- 1.1 Image - the input image
- 1.2 Line - max length (px) - the maximum distance allowed between two events to be connected by a (default = 0)

Effect:

Generates a graph structure connecting close events.

Graph from labels (v2)

Figure 476 – Graph from Labels (v2)

Where it can be found:

Basic Operations Module ► Legacy ► Graph from labels (v2)

Description:

Graph from labels (v2) is a deprecated operation.

This operation was not completely removed for reasons of compatibility with older projects.

The new version can be found at:

Basic Operations Module ► Labels ► Graph from labels

Parameters:

- 1.1 Image - the input image
- 1.2 Line - max length (px) - the maximum distance allowed between two events to be connected by a (default = 0)
- 1.3 Distance mode - specifies the reference points used to compute the distances: centroids or contours (default = 0)
- 1.5 Result of "…" - the result of the operation

Effect:

Generates a graph structure connecting close events.

Grow labels (v1)

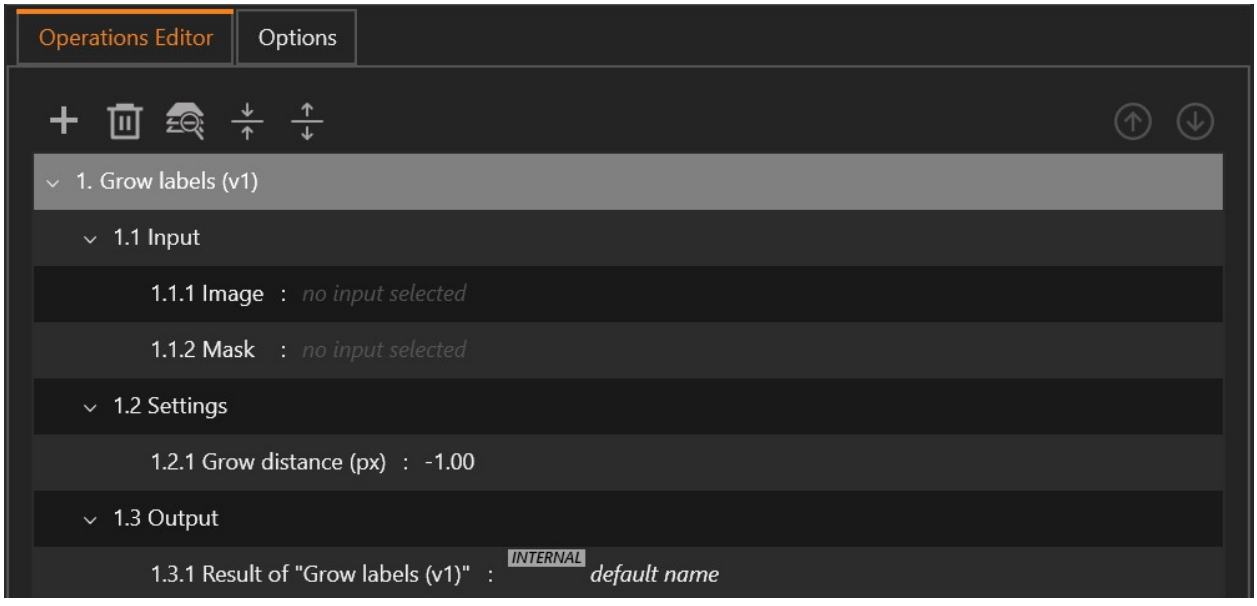


Figure 477 – Grow Labels (v1)

Where it can be found:

Basic Operations Module ► Legacy ► Grow Labels (v1)

Description:

Grow labels (v1) is a deprecated operation.

This operation was not completely removed for reasons of compatibility with older projects.

The new version can be found at:

Basic Operations Module ► Labels ► Grow labels

Parameters:

- 1.1 Image - the input image
- 1.2 Grow distance (px) - the growing distance in pixels (default = 0);
- 1.3 Seeds ignore mask - enables / disables mask constraint on seeds (default = Yes).

Effect:

Performs labels growing.

OD to absorbance

Figure 478 – OD to absorbance

Where it can be found:

Basic Operations Module ► Legacy ► Convert RGB to Grayscale

Description:

OD to absorbance is a deprecated operation.

This operation was not completely removed for reasons of compatibility with older projects.

The new version can be found at:

Basic Operations Module ► Color Conversions ► Convert Optical Density (OD) to RGB

Parameters:

- 1.1 Image - the input image
- 1.2 Value – Red - the intensity value for the Red channel
- 1.3 Value – Green - the intensity value for the Green channel
- 1.4 Value – Blue - the intensity value for the Blue channel
- 1.5 Result of "…" - the result of the operation

Effect:

- Converts an image from the Optical Density representation to the RGB color-space.

Polynomial - a + b * Image ^ c (v1)

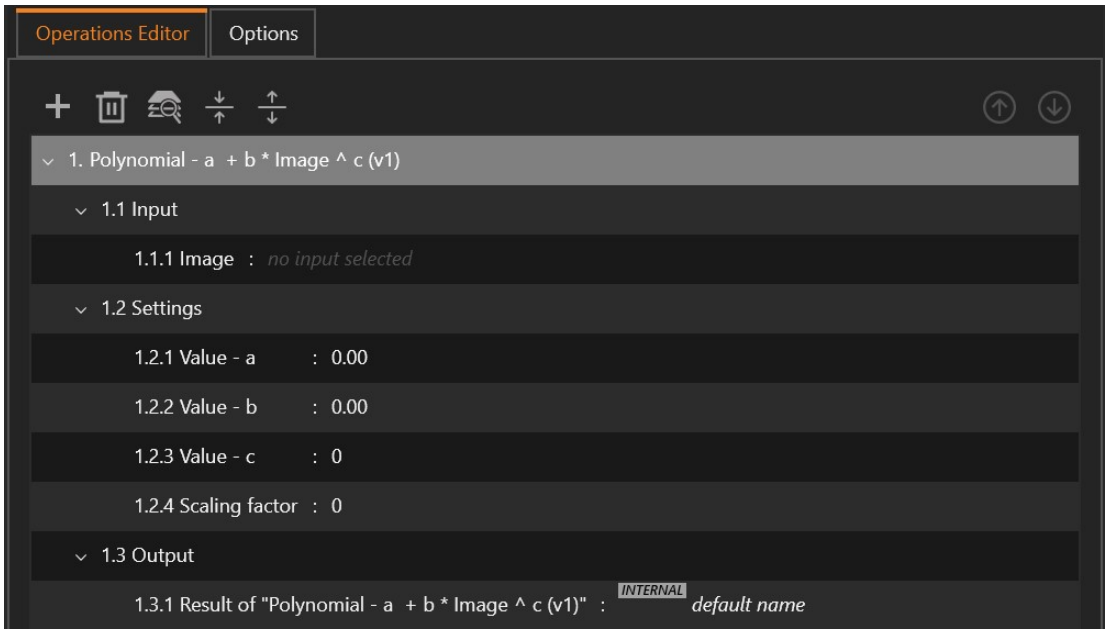


Figure 479 – Polynomial -a + b * Image ^ c (v1)

Where it can be found:

Basic Operations Module ► Legacy ► Polynomial -a + b * Image ^ c (v1)

Description:

Polynomial - a + b * Image ^ c (v1) is a deprecated operation.

This operation was not completely removed for reasons of compatibility with older projects.

The new version can be found at:

Basic Operations Module ► Arithmetic ► Polynomial - a + b * Image ^ c

Parameters:

- 1.1 Image - the input image
- 1.2 Value - “a” - the value of “a” in the polynomial function
- 1.3 Value - “b” - the value of “b” in the polynomial function
- 1.4 Value - “c” - the value of “c” in the polynomial function
- 1.5 Scaling factor - scales the result by multiplying with 2^{value} (default = 0).
- 1.6 Result of “…” - the result of the operation

Effect:

Changes the intensity of a grayscale image by applying a polynomial function.

Polynomial - Image ^ a (v1)

Figure 480 – Polynomial -Image ^ a (v1)

Where it can be found:

Basic Operations Module ► Legacy ► Polynomial - Image ^ a (v1)

Description:

Polynomial -Image ^ a (v1) is a deprecated operation.

This operation was not completely removed for reasons of compatibility with older projects.

The new version can be found at:

Basic Operations Module ► Arithmetic ► Polynomial - a + b * Image ^ c

Parameters:

- 1.1 Image - the input image
- 1.2 Value - “a” - the value of “a” in the polynomial function
- 1.3 Scaling factor - scales the result by multiplying with 2^{value} (default = 0).
- 1.4 Result of “…” - the result of the operation

Effect:

Changes the intensity of a grayscale image by applying a polynomial function.

Polynomial - Image ^ a (v2)

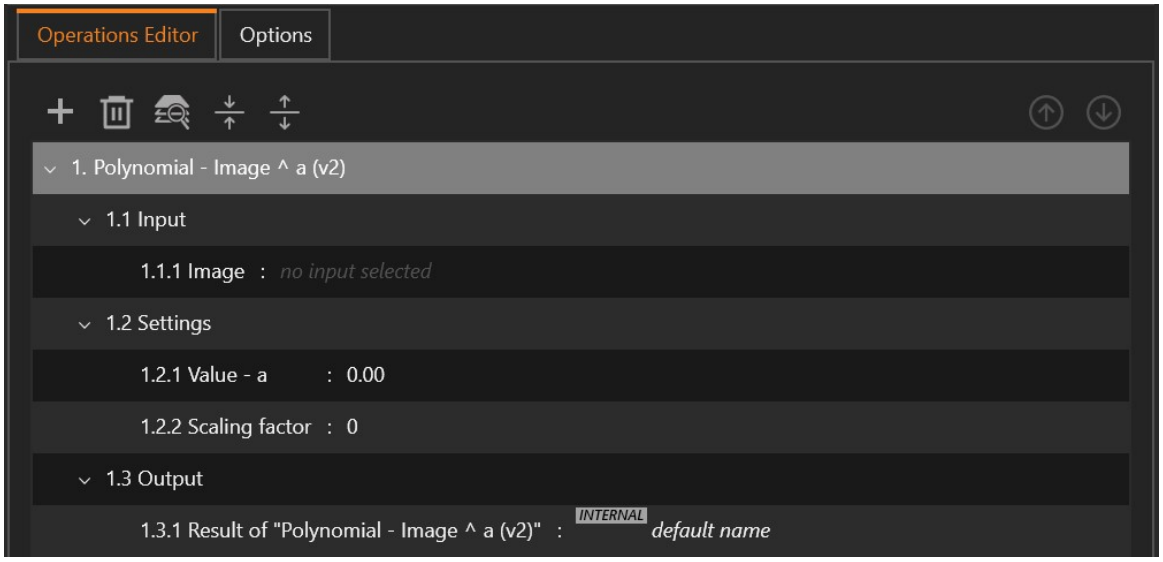


Figure 481 – Polynomial - Polynomial -Image ^ a (v2)

Where it can be found:

Basic Operations Module ► Legacy ► Polynomial - Image ^ a (v2)

Description:

Polynomial -Image ^ a (v2) is a deprecated operation.

This operation was not completely removed for reasons of compatibility with older projects.

The new version can be found at:

Basic Operations Module ► Arithmetic ► Polynomial - a + b * Image ^ c

Parameters:

- 1.1 Image - input image
- 1.2 Value - “a” - value of “a” in the polynomial function
- 1.3 Scaling factor - scales the result by multiplying with 2^{value} (default = 0)
- 1.4 Result of “…” - result of the operation

Effect:

Change the intensity of a grayscale image by applying a polynomial function.

Skeleton - extend spurs (v1)

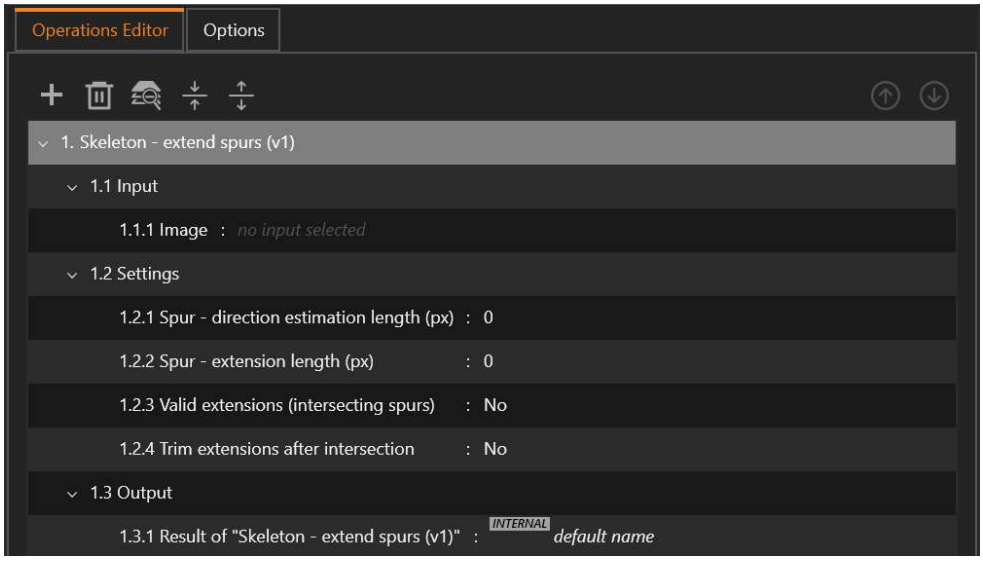


Figure 482 – Extended spurs

Where it can be found:

Basic Operations Module ► Legacy ► Skeleton - extend spurs (v1)

Description:

Skeleton - extend spurs (v1) is a deprecated operation.

This operation was not completely removed for reasons of compatibility with older projects.

The new version can be found at:

Basic Operations Module ► Morphology ► Skeleton - extend spurs

Parameters:

- 1.1 Image - the input image
- 1.2 Spur - angle estimation distance (px) - the number of pixels (from endpoint) used to estimate the angle of the spurs (default = 0)
- 1.3 Spur - extension length (px) - the pixel number representing the extension size (default = 0)
- 1.4 Valid extensions (intersecting spurs) - enables / disables validation only for extensions intersecting other extensions (default = No)
- 1.5 Trim extensions after intersection - enables / disables the removal of extensions beyond the intersection point (default = No).
- 1.6 Result of "…" - the result of the operation

Effect:

Extends skeleton spurs.

6. Cellular

Cellular

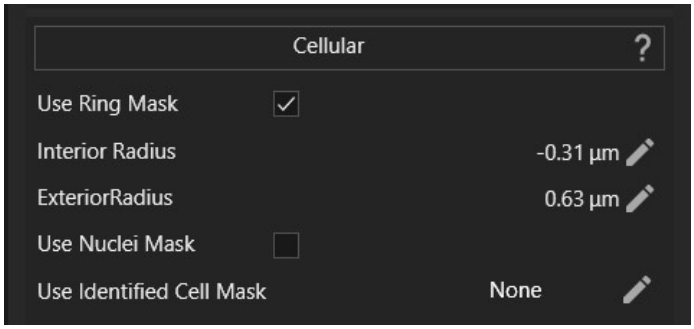


Figure 483 – Cellular

Where it can be found:

Layer's "Measurements Masks" ► Add ► Operations (Basic) ► Cellular

Description:

Cellular is an engine available in the Measurement Masks section of the layers editor.

This engine generates new measurements masks for the source set of events. A measurements mask is a set of pixels having the same label as the event they are associated with, defining an area, having a specific meaning related to the event (e.g. nuclear body, cytoplasmic body, etc.). This can be used to compute relevant measurements for the event (e.g. cytoplasmic marker expression around the nucleus, nuclear marker expression, membrane marker expression around the nuclei, etc.).

Three types of cellular masks are available and can be combined in any combination:

Nuclear mask - uses the body of the detected nucleus

Ring mask - generates a ring-shaped mask starting from the nucleus contour

Identified mask - generates a mask by identifying the presence of a marker around the nucleus, starting from contour.

Parameters:

- Use Ring Mask - enables / disables generation of ring mask for each event
- Use Nuclei Mask - enables / disables usage of events masks
- Use Identified Cell Mask - enables / disables generation of identified mask for each event

Ring Mask:

When enabled, generates a ring-shaped mask. The mask is created by growing from the event's contour inward and outward. The growing distances are limited by the values of the Interior Radius and Exterior Radius and the growing directions are controlled by the signs of the Interior Radius and Exterior Radius (a positive value for Interior Radius means a growth in the event's body interior, while a negative value means a growth in the opposite direction; same applies to Exterior Radius).

- Interior Radius - the interior growth distance. A negative value triggers an exterior growth
- Exterior Radius - the exterior growth distance. A negative value triggers an interior growth

Nuclei Mask:

When enabled, a copy of events bodies is used as a measurements mask.

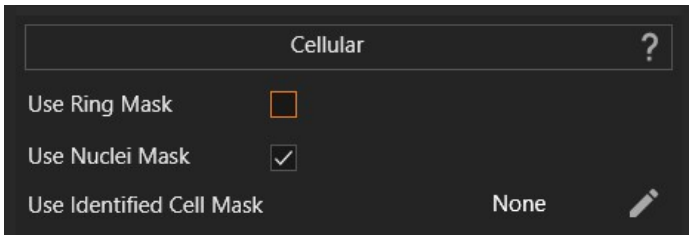


Figure 484 – Cellular: measurements mask

Identified Mask:

When enabled, it generates a mask based on the presence of specific markers in the proximity of events. The mask is created by constrained growing from the event's contour inward and / or outward. The constraints are:

- growing distance is limited by the Max Growing Steps parameter
- growing process includes only pixels with values (collected from Gray Mask) higher than threshold level (automatic or manual)
- grown area is connected with the growing starting point (events contours)

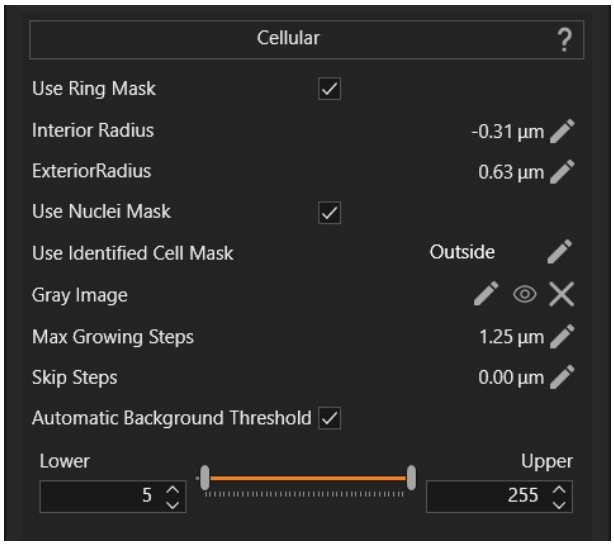


Figure 485 – Cellular: Mask constraints

- Gray Image - the grayscale image representing the marker of interest expression
- Max Growing Steps - the maximum growing distance from starting point
- Skip Steps - skips distance before growing starts. By default, the growing starts from the event contours. This parameter shifts the growing starting point outside events, at a specified distance from the contour (acts like a Dilate).
- Automatic Background Threshold - enables / disables automatic computation of threshold

Figure 486 – Cellular: Automatic Background Threshold

- Lower - the lower limit of the intensity range used to compute the automatic background
- Upper - the upper limit of the intensity range used to compute the automatic background

Figure 487 – Cellular: Manual Threshold

- Manual Threshold - the threshold value (available only when Automatic Background Threshold is disabled)

For all distance parameters, the associated measurement units can be changed between µm (default) and pixels. The options can be accessed by activating edit more (…) and right clicking the slider bar.

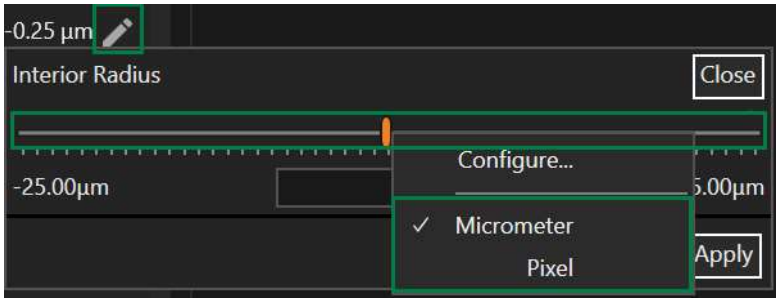


Figure 488 – Cellular: Configure Interior Radius

Effect:

Generates a custom measurements mask which can be used to measure the expression of different markers (e.g. nuclear, ring shaped, cytoplasmatic, etc.).

Example:

- Input:

For this example, we consider a nuclear detection (see Nuclei) performed previously, which generates the events displayed first figure. For each event a cellular mask is generated. The main events (nuclei) and all generated cellular masks are linked (have the same label / id). The second channel, used for cellular masks, is displayed in second figure.

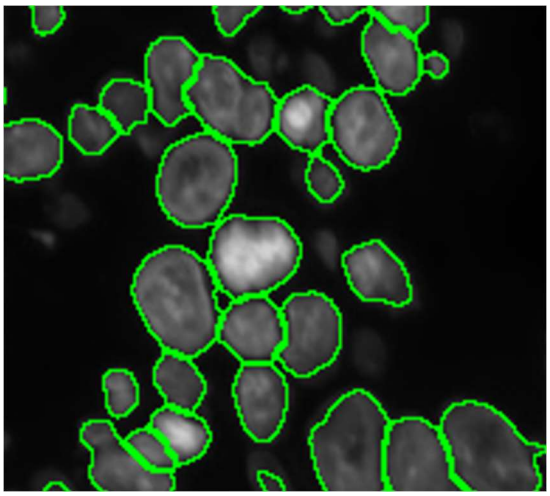


Figure 489 – Detected nuclei (on DAPI channel)

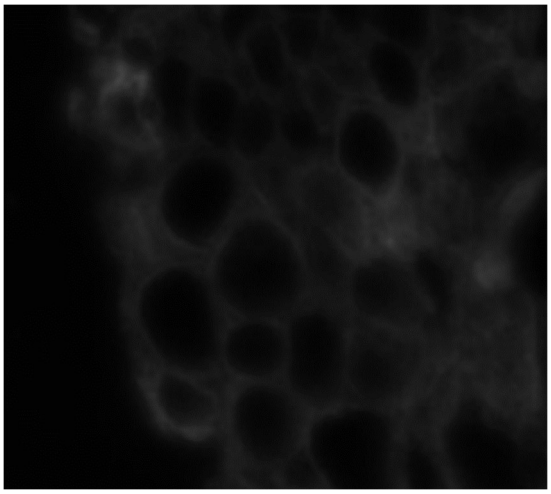


Figure 490 – Cytoplasmic marker

- Engine settings:

Figure below shows the engine settings used to generate ring-shaped cellular masks.

Figure 491 – Engine settings

Figure below shows the engine settings used to generate identified cellular masks, based on the marker presence.

Figure 492 – Engine settings

- Output:

Figure 4-11 shows the ring-shaped cellular mask (Use Ring Mask = on) combined with the nuclei (Use Nuclei Mask = on). The result looks like inflated nuclei.

Figure 4-12 shows the identified cellular mask using the information from the cytoplasmic marker (SpGold channel). The detection of the cellular mask starts from the nucleus contour and grows for Max Growing Steps = 23 um, if the threshold condition is satisfied (pixel value > Manual Threshold = 10).

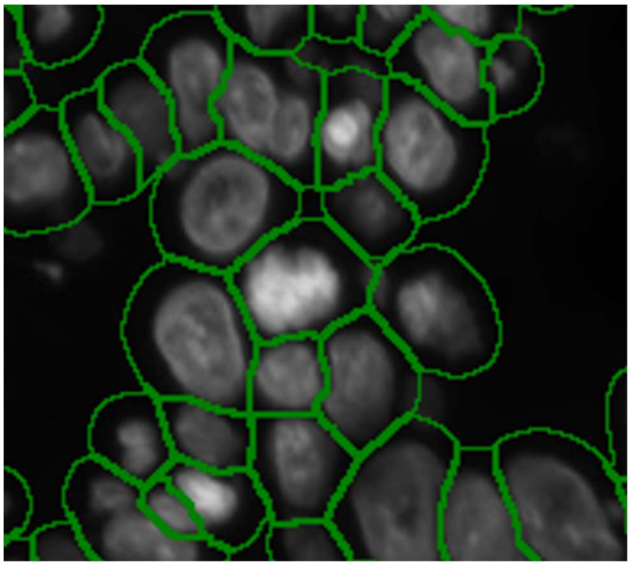


Figure 493 – Result of Cellular engine (ring mask)

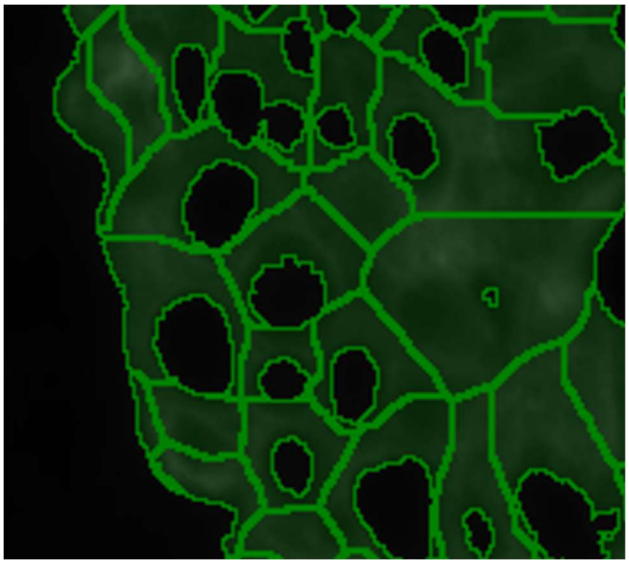


Figure 494 – Result of Cellular engine (identified mask)

7. Classifier

Classifier

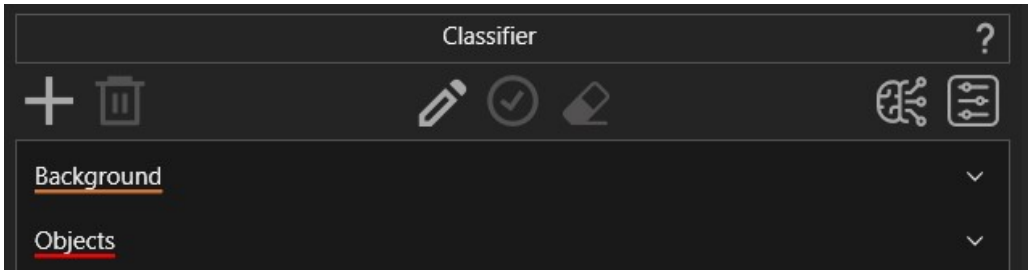


Figure 495 – Classifier

Where it can be found:

Layer's "Pre-Processing" ► Add ► Operations (Advanced) ► Classifier

Description:

Classifier is an engine available in the Pre-Processing section of the layers editor.

This engine assigns a user defined label (class) to each macro block (pixel or set of pixels), based on computed features (measurements). The macro block size is defined by the user. The classification is based on a training process, where a training data set (labelled macro blocks, set of pixels whose class appurtenance is known) is provided and the engine "learns" how different information must be classified.

Due to high variability found in the characteristics of different tissue samples (from different organs to different staining protocols and different acquisition settings), it's impossible to say a classification method is the best one, that's why this engine offers several methods for classification to select from:

- Random forcest (default) (more details https://en.wikipedia.org/wiki/Random_forcest)
- K-nearest (more details https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm)
- Support vector machines (more details https://en.wikipedia.org/wiki/Support_vector_machine)
- Artificial neural networks (more details https://en.wikipedia.org/wiki/Artificial_neural_network)
- Normal Bayes (more details https://en.wikipedia.org/wiki/Naïve Bayes classifier)

Parameters:

The Classifier main window allows:

- Class management
- Advanced settings access

- Training a new classifier

Classes

The engine's main window allows the definition and management of a list of classes which covers the data information variability:

1. Add new class - adds new class to the list
2. Remove selected class - removes selected class from the list
3. Edit a class - a set of tools that helps define a class

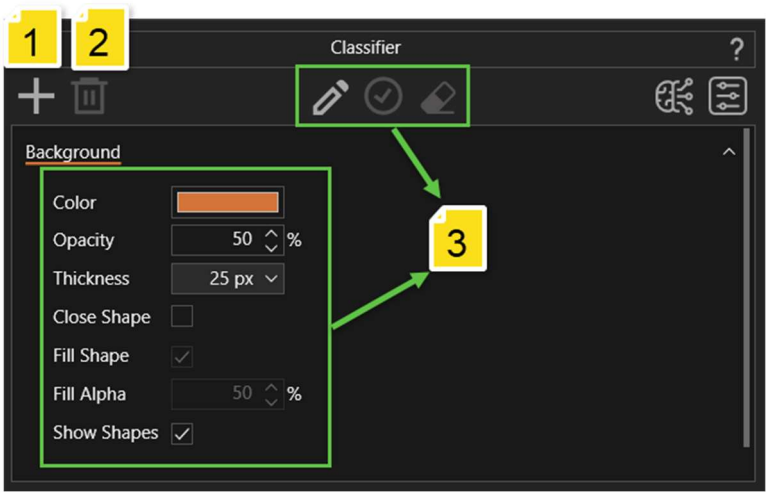


Figure 496 – Classifier: Manage classes

A new class can be managed by using the following steps:

1. Add a new class by using the Add new Class button;
2. Name the class by double clicking on the name;
3. Assign a color;
4. Settings for annotations:
    a. Opacity - sets the opacity level of the annotation. A value of 100% will make the annotation opaque making impossible to see the original image information drawn upon;
    b. Thickness - size of the brush used to annotate;
    c. Close shape - enables / disables shape's closing. When enabled, the end point of the annotation (when left mouse button is released) will be connected by a line to the starting point of the annotation (when the mouse button was pressed);
    d. Fill Shape - enables / disables shape's filling. When enabled, only for closed shapes, the interior is filled;

e. Fill Alpha - sets the opacity level of the shape's interior, only for filled shapes;

5. Manage annotations (class examples):
   a. Add new annotations:
      i. enable the drawing mode using the Start drawing button;
      ii. start drawing with the mouse on the images to highlight defining areas for the current class;
      iii. use the Apply button when the process is complete.
   b. Delete annotations:
      i. enable the drawing mode using the Start drawing button;
      ii. enable the erase mode by using Start erasing button;
      iii. start erasing previous annotations with the mouse on the images to correct annotation errors;
      iv. use the Apply button when the process is complete.
   c. Validate an annotation:
      i. enable the drawing mode using the Start drawing button;
      ii. double click an annotation from the list (available only when drawing mode is active) to center the image on the selected annotation position;
      iii. accept / correct (with help from 5.1.2 and/or 5.2.1) / delete (use Delete keyboard key) the selected annotation;
      iv. use the Apply button when the process is complete.

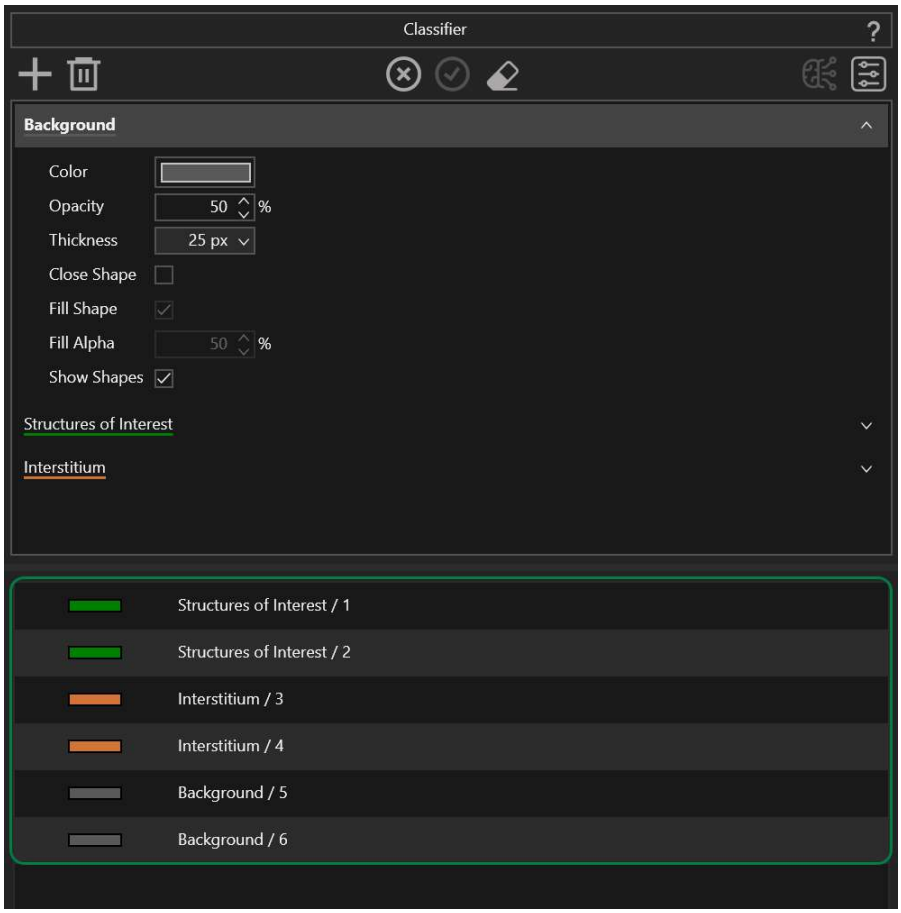
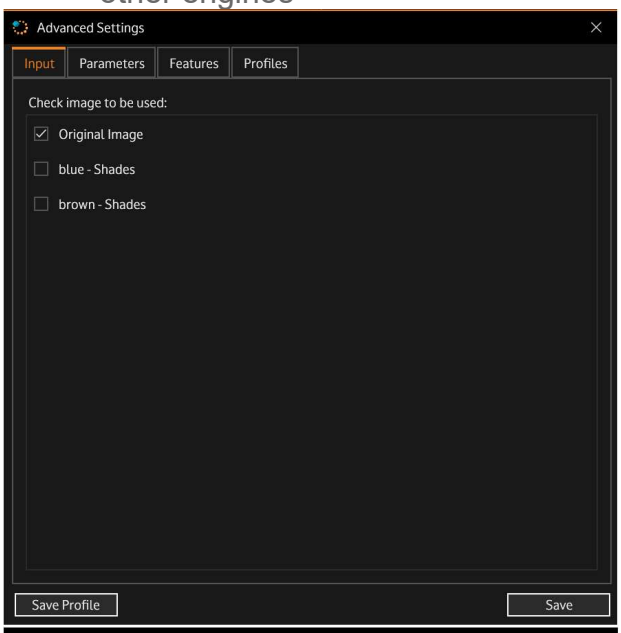


Figure 497 – Classifier: Added classes

Advanced settings

The engine's Advanced settings window enables classifier configuration. There are 4 tabs:

1. Input
- select the images used in classification: original images or images generated by other engines



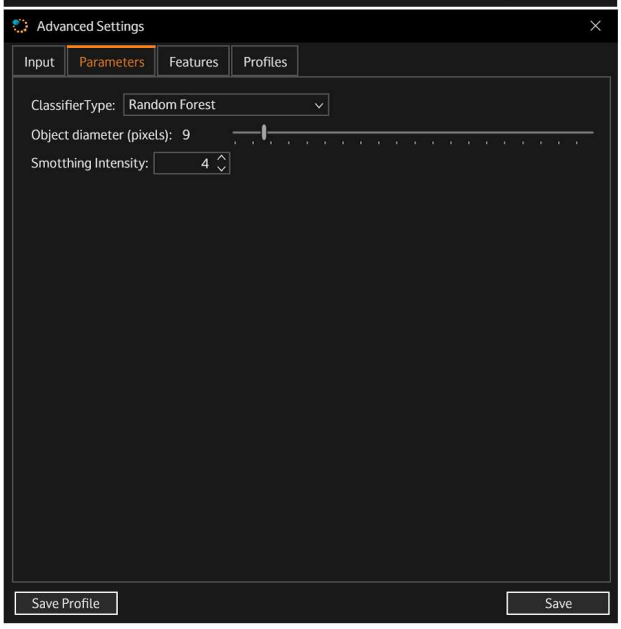


Figure 498 – Classifier: Advanced Settings (a)

2. **Parameters**
   - Classifier Type - selects the classifier model;
   - Object diameter (pixels) - defines the size of the observations (block of pixels, default = 9 x 9 pixels);

- Smoothing Intensity - the level of smoothing applied on the output to overcome the pixilation effect generated by large values used for Object diameter.

3. **Features**

Select which features to be computed for each observation:

Measurements - select set of measurements:

- Intensity - intensity-based features;
- Histogram - histogram-based features;
- Percentile - percentile-based features;
- HuMoments - features computed using central moments that are invariant to image transformations;
- Haralick - texture-based features.

Images - select images used to compute selected features. Images are affected by defined size:

- Original - original images;
- Morpho - virtual images generated from the original images using morphological operations;
- Average - virtual images generated from the original images using average filters;
- Gauss - virtual images generated from the original images using gauss filters;
- Median - virtual images generated from the original images using median filters;
- Bounding Box - use tile's bounding box to compute selected measurements on original image. Bounding box size is affected by defined size (tile size + defined sizes).

Sizes

- Selected Sizes - define proximity areas for observations; useful to generate contextual information;
- Maximum Small Size - maximum size allowed for some time-consuming measurements.

Misc

- Use Reduced Feature Selection - enables / disables the identification of relevant features in the training phase. Only the relevant features will be computed and used in the classification phase.

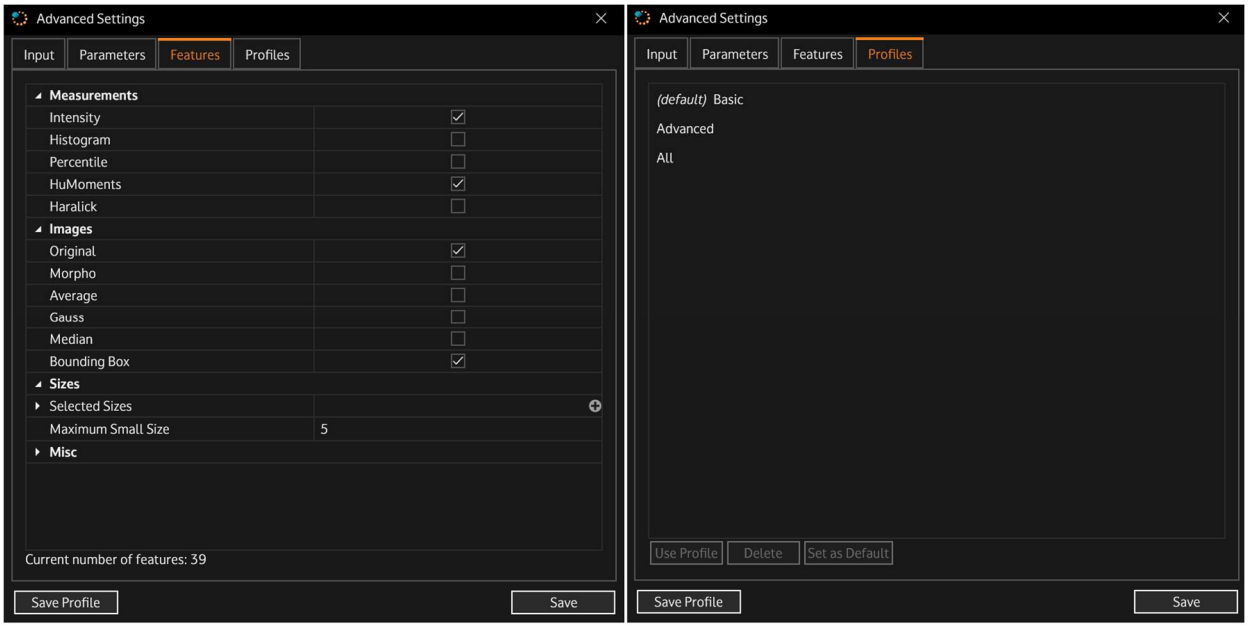


Figure 499 – Classifier: Advanced Settings (b)

4. **Profiles**

All settings made in Parameters and Features tabs can be saved for further use. This can be done by creating a new profile using the button Save Profile. The new profile is available in the list presented in 4th tab.

By default, there are 3 pre-defined profiles:

- Basic - a minimum selection of features; optimized for speed
- Advanced - a balanced selection of features
- All - all features are selected

Training

The classification is supervised learning process, where it is learned how to assign pre-defined labels (classes) to observations (pixels or set of pixels), based on internally computed features (selected measurements).

The learning phase generates a trained model, which can be used to classify new data.

Effect:

Assigns pixels or set of pixels to user defined classes.

Example:

- Input:

The input consists on an IHC skin sample. The classifier is trained using manual annotations to provide examples for each defined class.

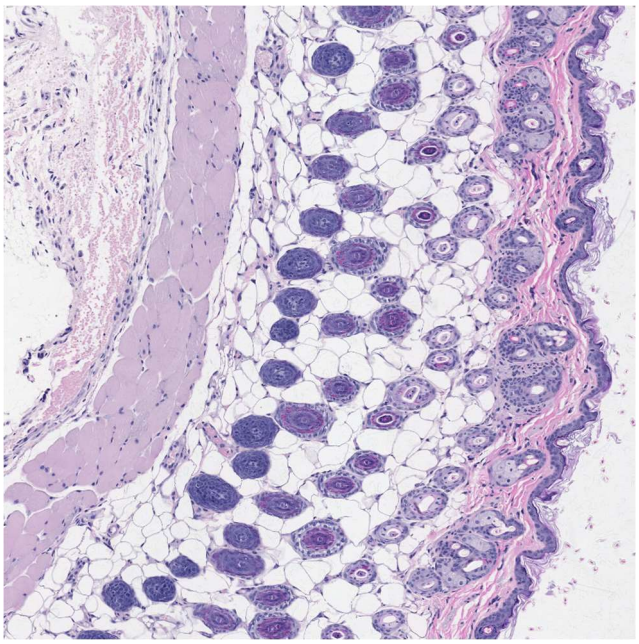


Figure 500 – Skin sample

- Engine settings:

Figure below shows the defined classes used for this example.

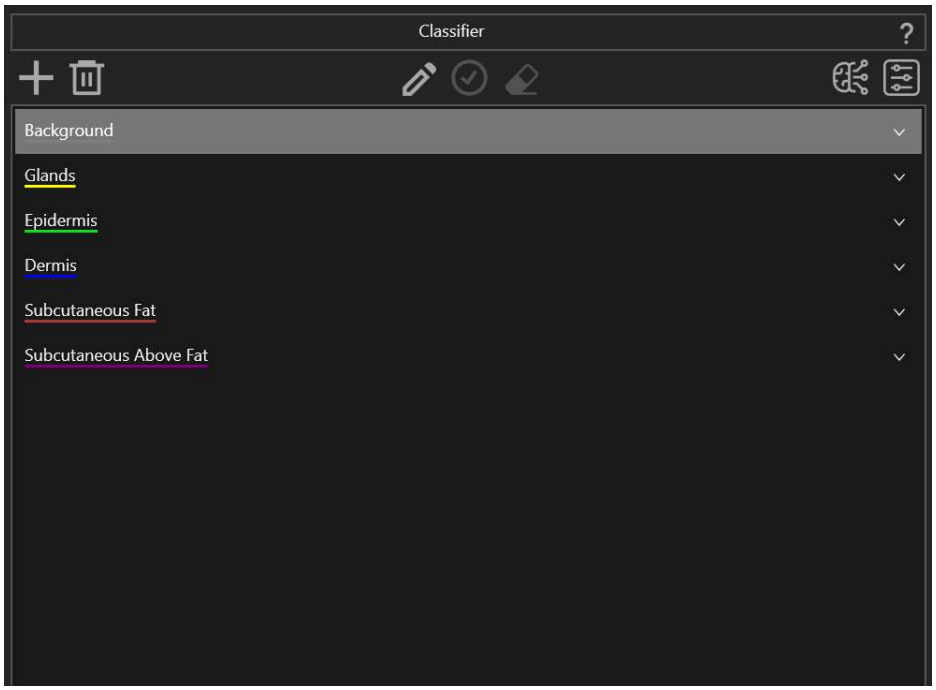


Figure 501 – Engine settings

Output:

Figure below shows the result of the classifier engine. Each pixel is overlaid with the color of its assigned class. The engine also generates individual binary masks for each class.

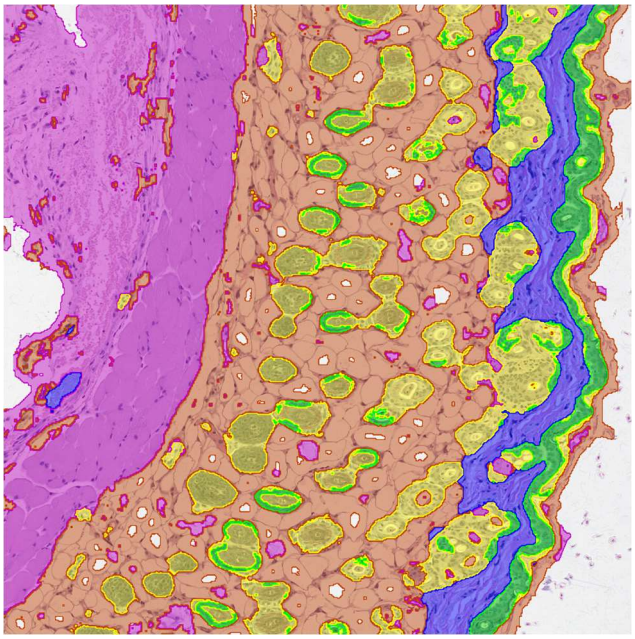


Figure 502 – Result of Classifier engine

7.1. Classifier Measurements and Visualization

Visualization

1. Click on the eye button for “visualization Coded Map”.
2. Click on Images -> find the overlay button for “Visualization Coded Map” -> Edit.
3. For the selected “visualization Coded Map” image select “Labelled Interior and Contour”, Set Transparency to 30.
4. Option to rename and set a button color for the overlay.

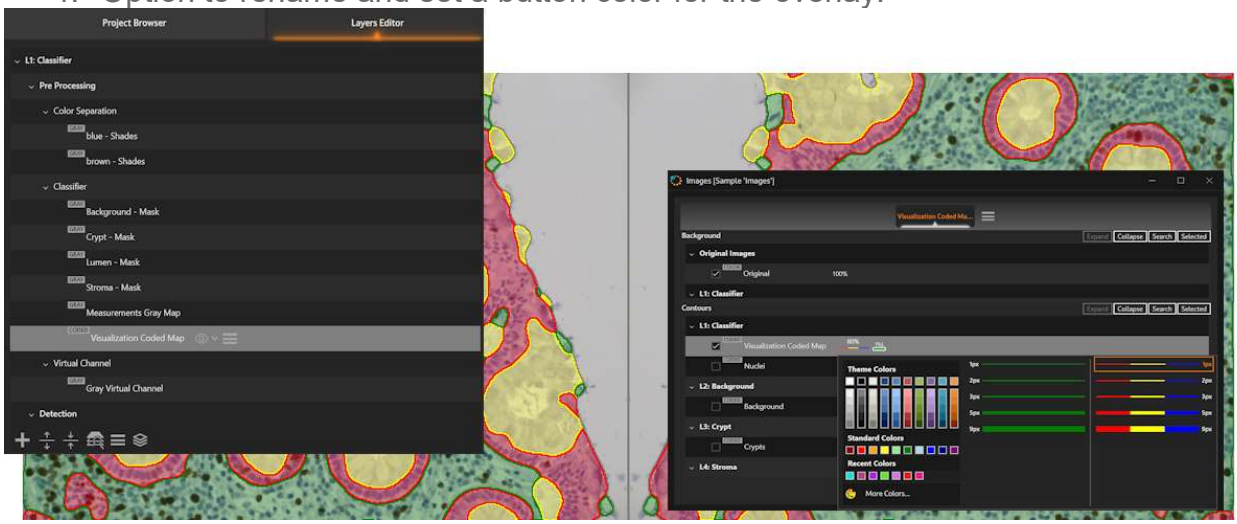


Figure 503 – Visualization

Select nuclei on a specific classifier class

1. Add “Standard Measurements (global)” in the layer where you have the classifier results and nuclei detection.
2. In “Coded”, select the Nuclei.
3. In “Shades”, select the “Measurements Gray Map”. This is an image generated by the classifier, used for measurements, where the 1st class will have value 1, the 2nd class will have value 2, and so on.
4. Select “Most Frequent Intensity”. This measure will ensure that each nucleus will have a value that corresponds to a class. If the nuclei are on top of multiple classifier classes, the result will take the class that is the most predominant.
5. To Add Diagram, set Y axis to the measurement “Most Frequent Intensity”.
6. Right-click on diagram -> Parameters.
7. Set “Max” to “5” (based on the number of classes in the classifier).
8. Set Gate to select a specific class.
9. Backwards connection for selected class/gate.

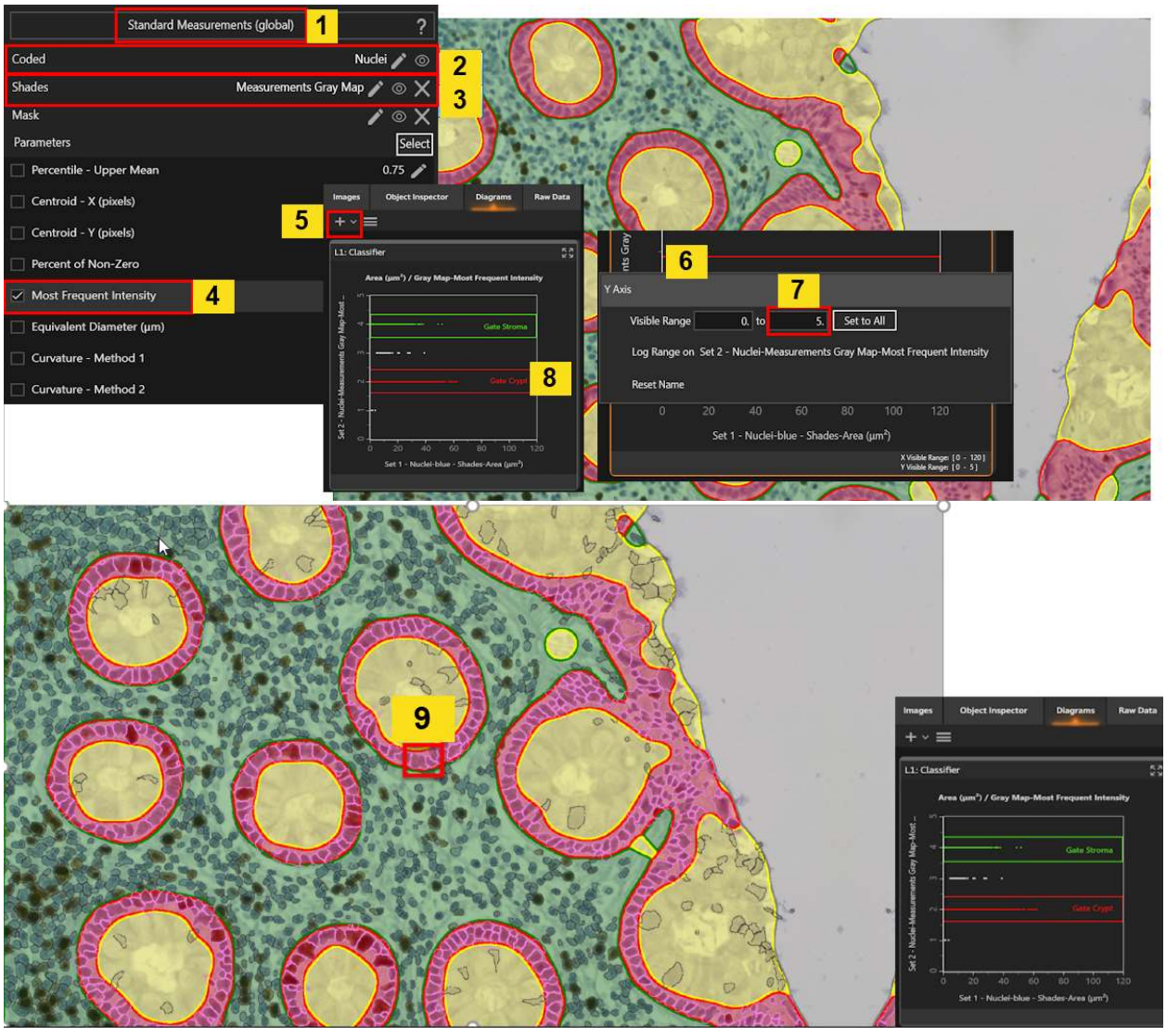


Figure 504 – Select nuclei on a specific classifier class

Area of Classifier Class (Layer)

1. For each class that will be measured:
2. Add Layer.
3. Pre Processing -> Add -> Import Gray.
4. Detection -> Set -> Total Area. Set gray classifier mask as input. Uncheck “Automatic Background Threshold”. Set “Manual Threshold”.
5. Measurements -> Add -> Standard Measurements (global). For Coded, select the image from Detection. For Measurements: select “Area”, unselect “Mean Intensity”.

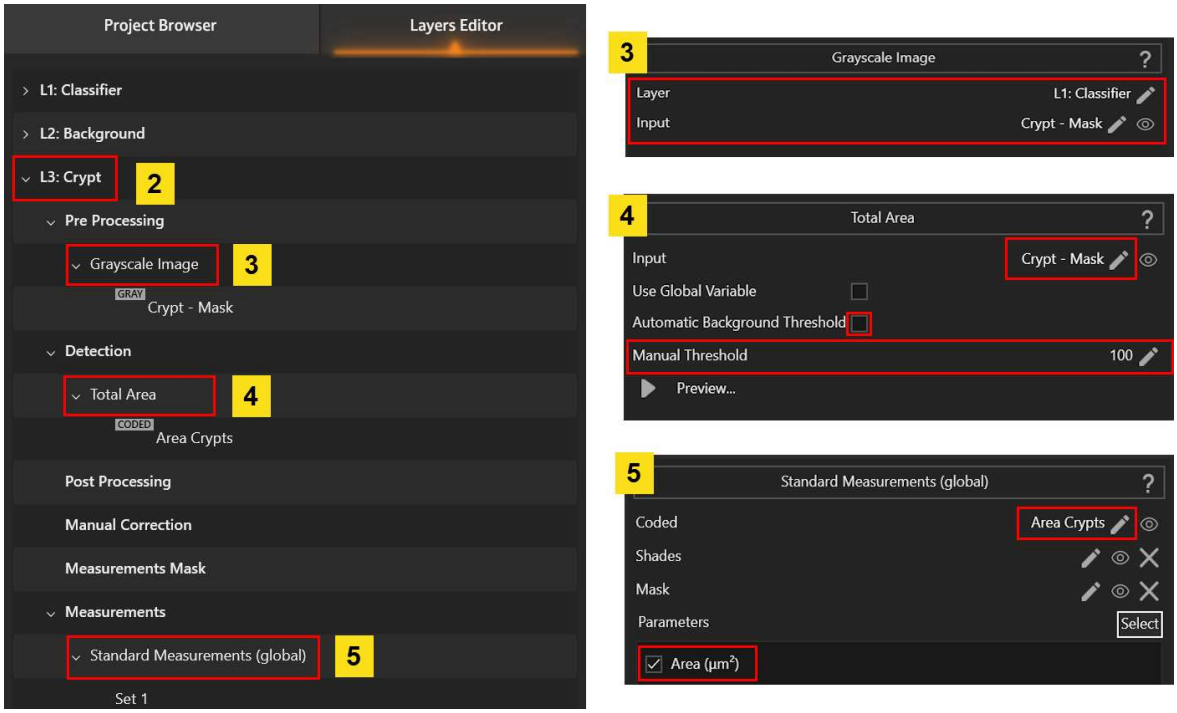


Figure 505 – Area of Classifier Class (Layer)

Area of Classifier Class (Visualization)

1. After a layer was created for each class that needs to be measured:
2. Add new overlay image.
3. Select overlay/color background image.
4. For “Contour” images, select all “Area” images generated by “Detect Objects” from each layer where the area of each Classifier class is measured. (see above)
5. For each selected contour image, select “Interior Fill”, select color, select “Transparency” color and Thickness. Select the color based on the color used in the classifier.

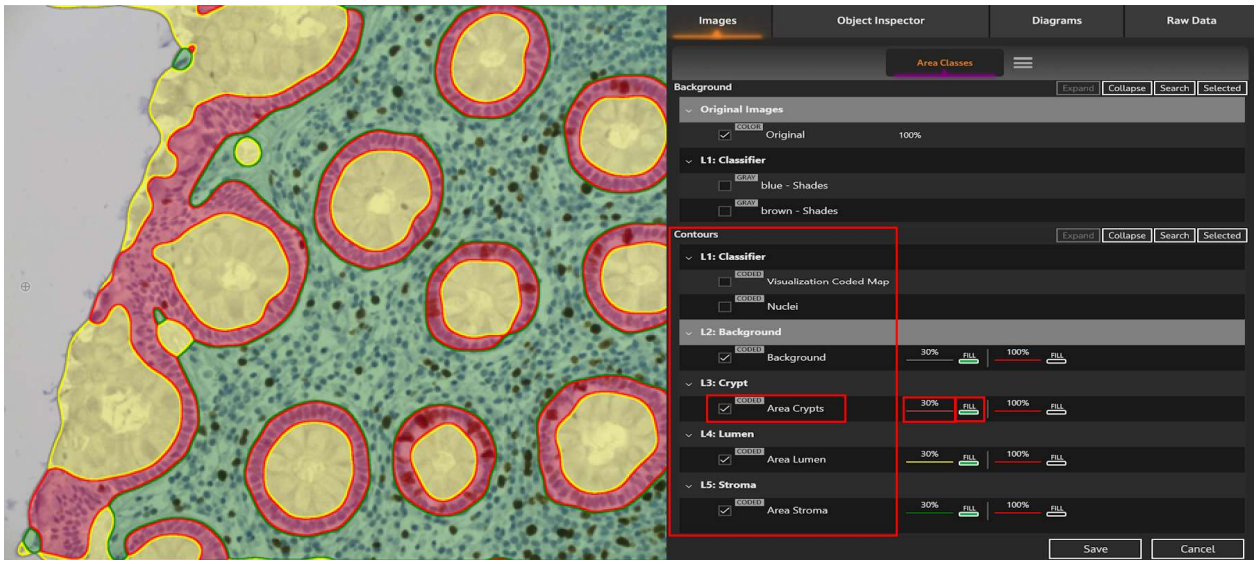


Figure 506 – Area of Classifier Class (Visualization)

Classifier Visualization in ROI (restrict to ROI)

1. If previous steps are done, then classes will be restricted to current ROI and seen in the image below.
2. This happens because “Detect Objects” removes any detection outside of the ROI.
3. The default Classifier results does not perform a restriction on the ROI.

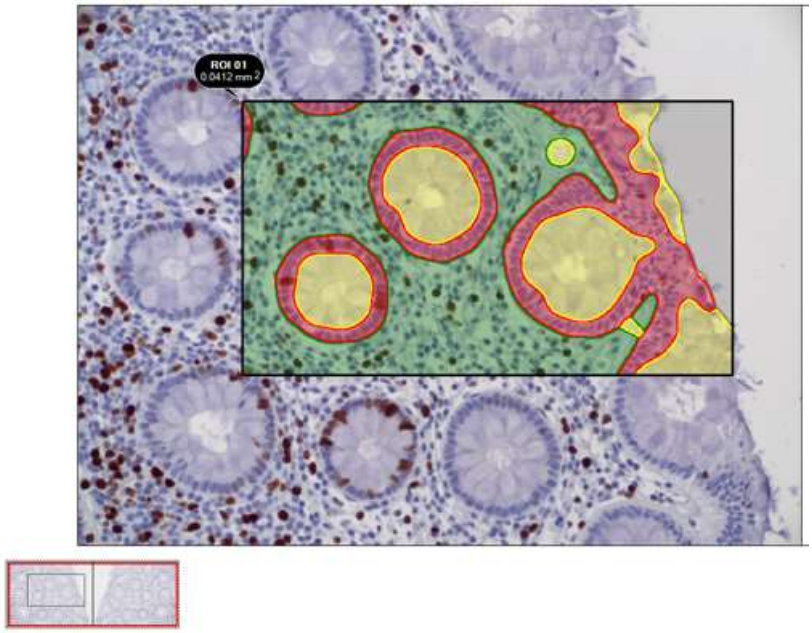


Figure 507 – Classifier Visualization in ROI (restrict to ROI)

8. Coded Image

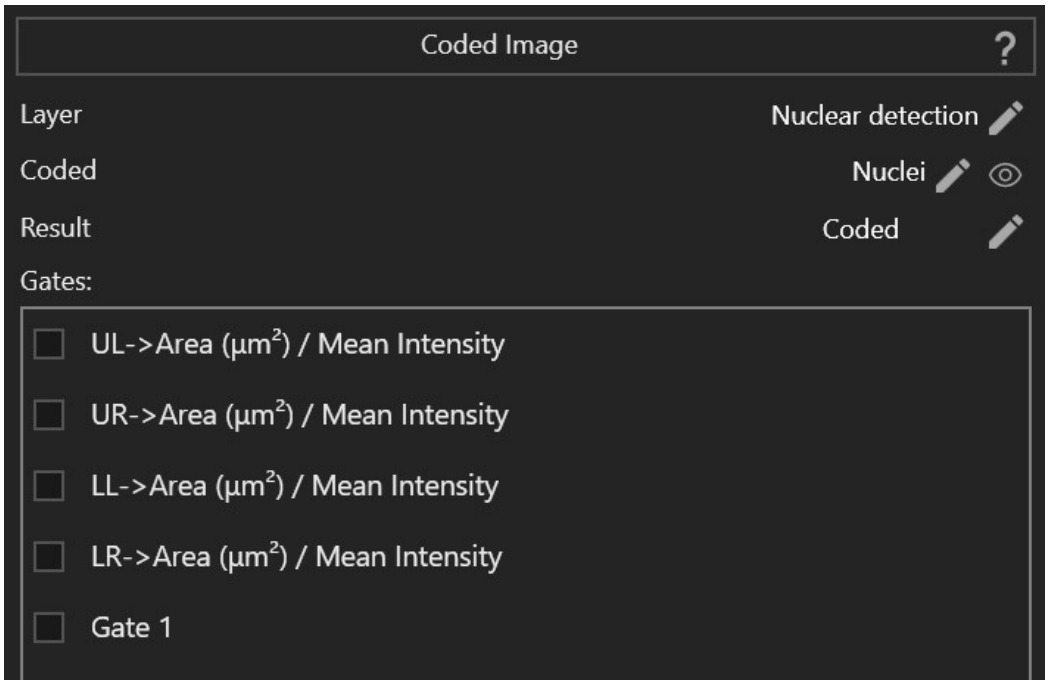


Figure 508 – Coded Image

Where it can be found:

Layer's "Pre-Processing" ► Add ► Import ► Coded image

Description:

Coded image is an engine available in the Pre-Processing section of the layers editor.

A coded image represents a set of events labeled with unique IDs.

By default, a set of detected events is "visible" only in the layer where it is generated. The (import) Coded image engine allows for importing a set of events from a different layer and making it available in the current layer, avoiding multiple detections for the same set of events.

An example is counting FISH dots (detected in Layer 1) inside nuclei (detected in Layer 2). To make the counting possible, the dots events must be made available in the layer where nuclei are detected.

Parameters:

- Layer - the layer where the coded image is available;
- Coded - the coded image representing the set of events to be imported in the current layer;

- Result - specifies the information to be imported:
  - Coded - imports the events;
  - Mask from Coded - generates a mask of all events (white = events). Unique IDs and individual contours are lost;
  - Mask Background - generates a mask for the background (white = background).
- Gates - allows import of specific sub-sets of the set of events. The option is available only for events used in scattergrams or diagrams, with defined cutoffs and/or gates.

Note	Stitch information is not available for events imported using gate(s).

Effect:

It makes a set of events from another layer available in the current layer.

Example:

- Input:

A set of events consisting of detected nuclei is considered. Measurements are performed and Ki67+ cells are identified. The Ki67+ cells are imported in a new layer for further analysis.

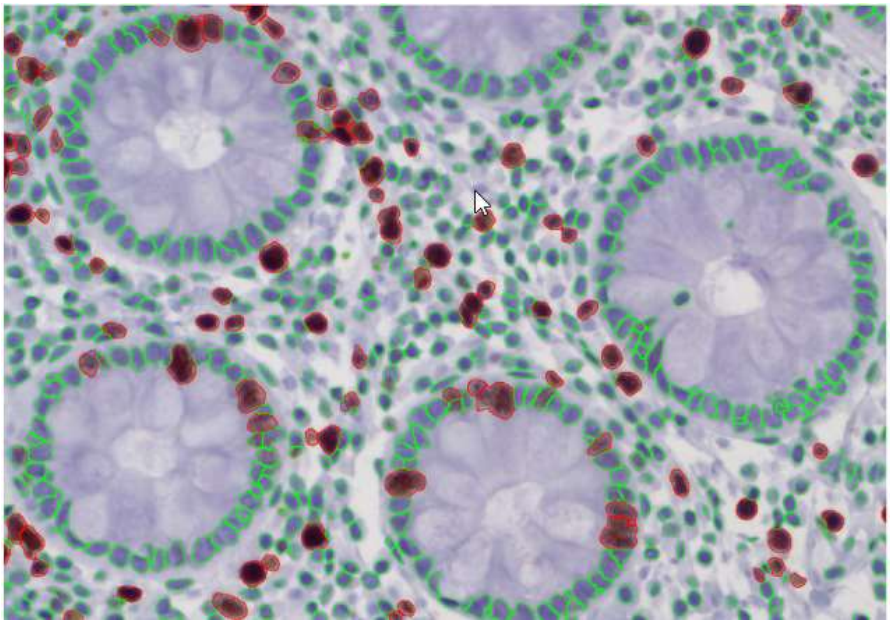


Figure 509 – Ki67+ cells on IHC colon sample (highlighted in red)

- Engine settings:

Figure below shows the engine settings used to import only the Ki67+ cells from the entire cell population.

Figure 510 – Engine settings

- Output:

Figure below shows the result of the Coded image engine. The option used, is to generate a mask of the imported events.

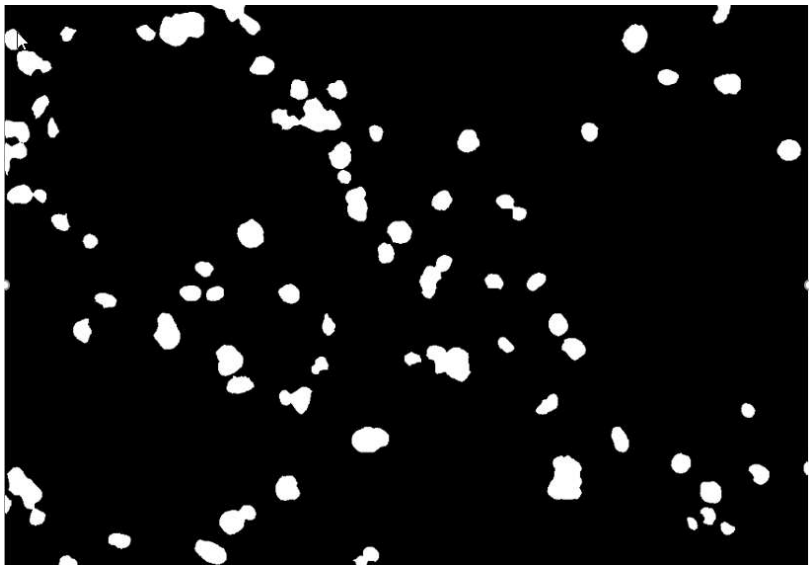


Figure 511 – Result of Coded Image engine

9. Color Separation

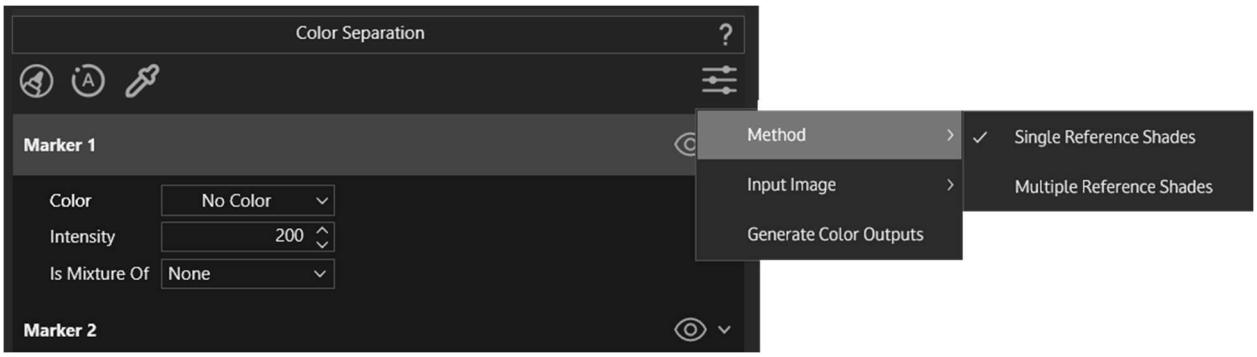


Figure 512 – Color Separation

Where it can be found:

Layer's "Pre-Processing" ► Add ► Operations (basic) ► Color Separation

Description:

Color Separation is an engine available in the Pre-Processing section of the layers editor.

Based on the selected method, the engine performs an unmixing of the color information of the pixels (Single reference shades) or a classification of the pixels based on the colorimetric properties of the pixels (Multiple reference shades). The output consists of a different image for each defined marker, showing the separated marker.

Note	The Color Separation engine is intended to be used on IHC images (color images) to separate the markers (2 or 3 markers).

Options:

- Color Separation Method – specifies the method used for color separations
  - Single Reference Shades – performs an unmixing of the color information for each pixel, generating a quantification image with respect to the reference marker value defined by the user.
  - Multiple Reference Shades – performs a classification / clusterization of the pixels based on their color information and the shades indicated as reference

- Autodetect – auto-detects the markers present in the image. Available only for Single Reference Shade method with two markers defined (does not work for more than 2 markers).
- Markers – menu for markers options
  - Add Marker – defines new marker
  - Rename – renames selected marker
  - Remove – removes selected marker
  - Remove all markers – removes all defined markers
  - Manage markers – manages all defined markers
- Use Original Image – enables / disables the usage of the original image as input for the color separation process
- Color Image – specifies the input for the color separation process, when the Use Original Image option is disabled (e.g. the original image is filtered for noise reduction and the result is used for color separation)
- Generate Color Outputs – specifies the type of images used to represent the separated markers: grayscale (disabled) or color (enabled)

9.1. Single Reference Shades

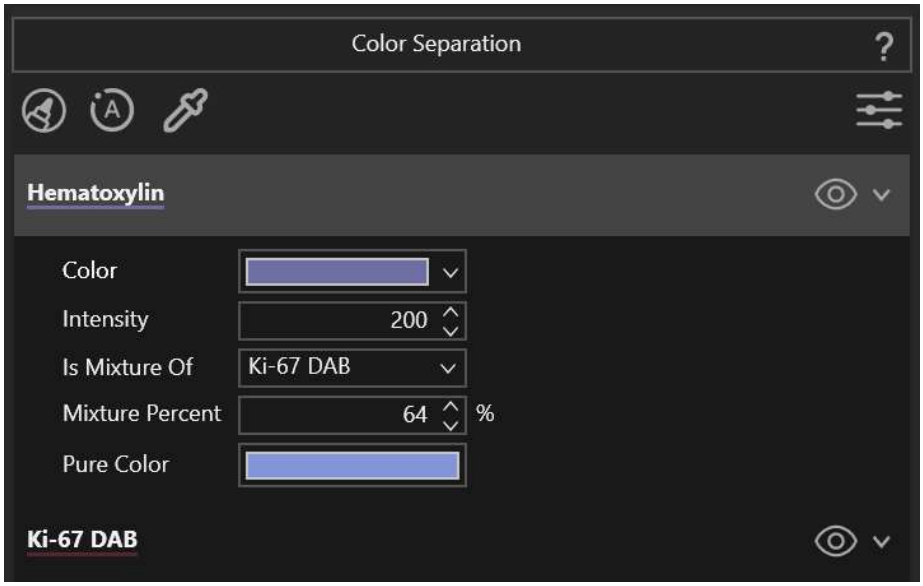


Figure 513 – Single Reference Shade (Hematoxylin marker)

This color separation method is an unmixing process that estimates the presence of markers with respect to the reference markers value(s). For example, if the reference value for the Hematoxylin marker is selected to be a blue shade similar to the one presented in Figure 7-2, the resulting image representing the separated Hematoxylin marker will have pixels with value given by the formula:

$$I_{Hematoxylin}(x,y) = \begin{cases} < 200, when\ I_{Original}(x,y) = ligther\ blue\ than\ reference \\ = 200, when\ I_{Original}(x,y) = same\ blue\ as\ reference \\ > 200, when\ I_{Original}(x,y) = darker\ blue\ than\ reference \end{cases}$$

, where:

- the value of 200 (Intensity = 200) is the used to represent the quantity of Hematoxylin maker similar with the one present in the position used to collect the blue shade for reference (Color / Pure Color)
- lighter / darker blue shade translates in fewer / more markers present at (x,y) position, in the original image, compared with the position used to collect the reference shade of blue (Color / Pure Color) for the Hematoxylin marker.

Reference markers:

The reference markers values can be assigned manually or automatically, using one of the following options:

- Autodetect – an automated method to identify the markers present in the selected region. It works only for two markers.

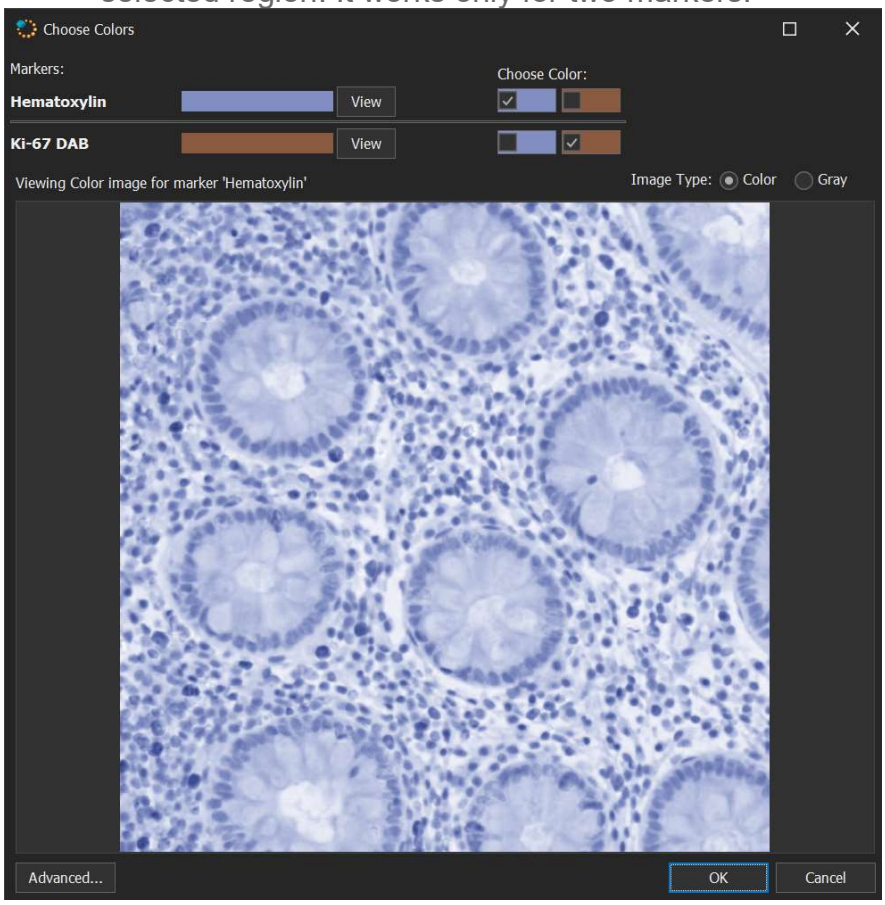


Figure 514 – Autodetect markers

If autodetected colors are incorrectly assigned, the correct assignment can be done by enabling the correct Choose Color checkboxes.

Depending on the information present in the image, unexpected color(s) may be automatically assigned to the markers. This can be prevented by changing the color outliers, accessible from the Advanced button.

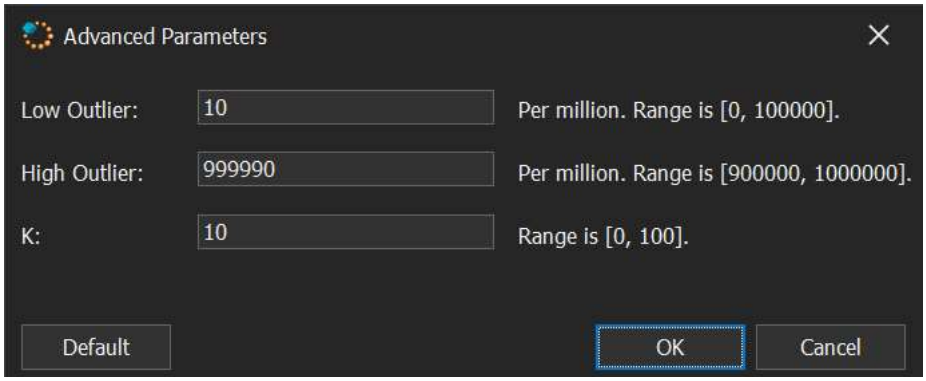


Figure 515 – Autodetect (advanced parameters)

The area used for auto-detection is 1000 x 1000 pixels (1 Million pixels). The values of the lower / upper outlier are in pixels, representing the interval of pixels considered in the auto-detection process:

- Low Outlier – the lower end of the pixel range considered in the auto-detection process
- High Outlier – the upper end of the pixel range considered in the auto-detection process
- K – by increasing this value, lighter shades will not be considered in the auto-detection process
- Color Picker – the reference values (color) can be assigned to the selected marker by using the Color Picker functionality. Once activated, a color box will appear to the right of the button, showing the current mouse position color (collected from the image). If the current mouse cursor position is not valid position (not over an image), the displayed color will be white. The shade is collected from the image and added in the selected marker's list by clicking on the identified position in the original / color reduced image.

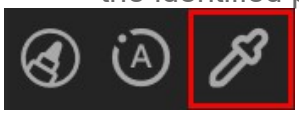


Figure 516 – Color Picker

- Color Palette – opens a window for manual selection of a reference color. It can be used to also input the reference color as RGB values

Figure 517 – Color Palette

Options:

- View Gray Image – displays the grayscale image representing the separated selected marker

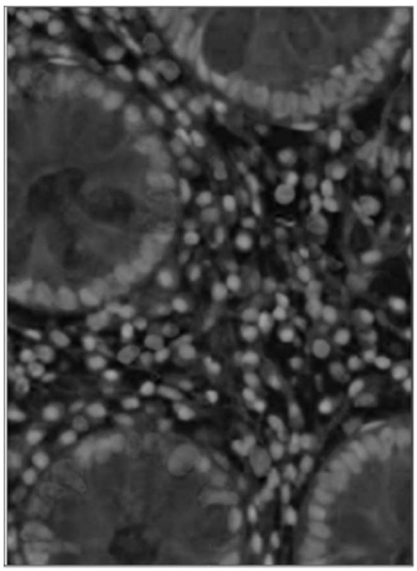


Figure 518 – View Gray Image result

- View Color Image – displays the color image representing the separated selected marker

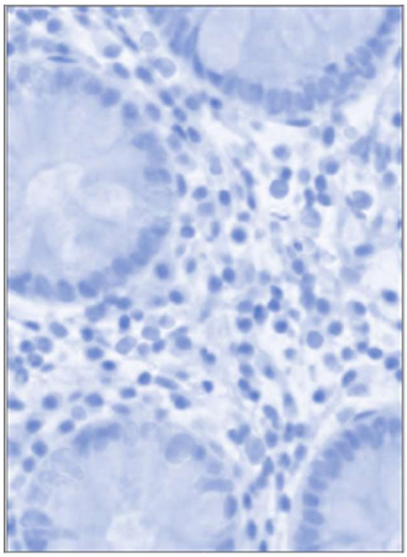


Figure 519 – View Color Image result

- Color – display the reference color associated with the selected marker
- Color Palette – opens the reference color selection window
- Intensity – specifies the output grayscale value used to represent pixels indicating the same color shade (quantity of marker present) as the reference color (marker quantity used as reference). All other pixel values are scaled with respect to the reference color (marker quantity).
- Is mixture of – when enabled, specifies that the collected reference color is in fact a mixture of the current marker with another marker which must be specified

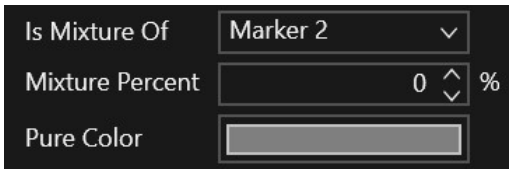


Figure 520 – Marker mixture settings

- Mixture Percent – specifies the estimated percent of the second marker present in the position used for reference color selection for the current marker.
- Pure Color – shows the resulting reference color associated with the current marker based on the Mixture Percent specified

Example:

- Input:

Image below is a brightfield color image (8-bit, 3-channel) representing an IHC small region from a colon sample.

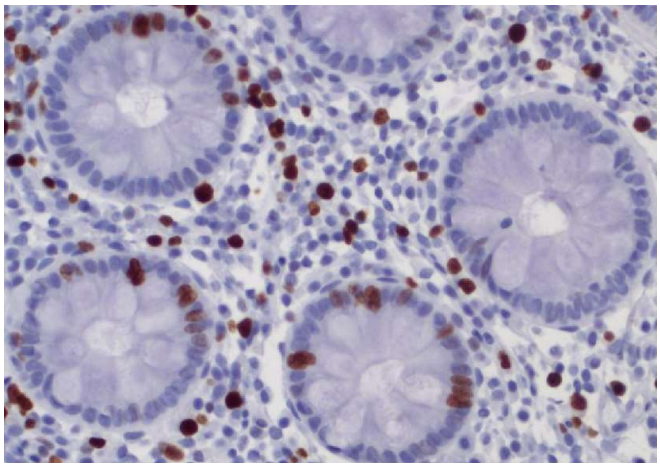


Figure 521 – IHC colon sample

- Engine settings:

Figures below show the engine settings used for this example.

The engine's results are grayscale (default) / color (if enabled) images.

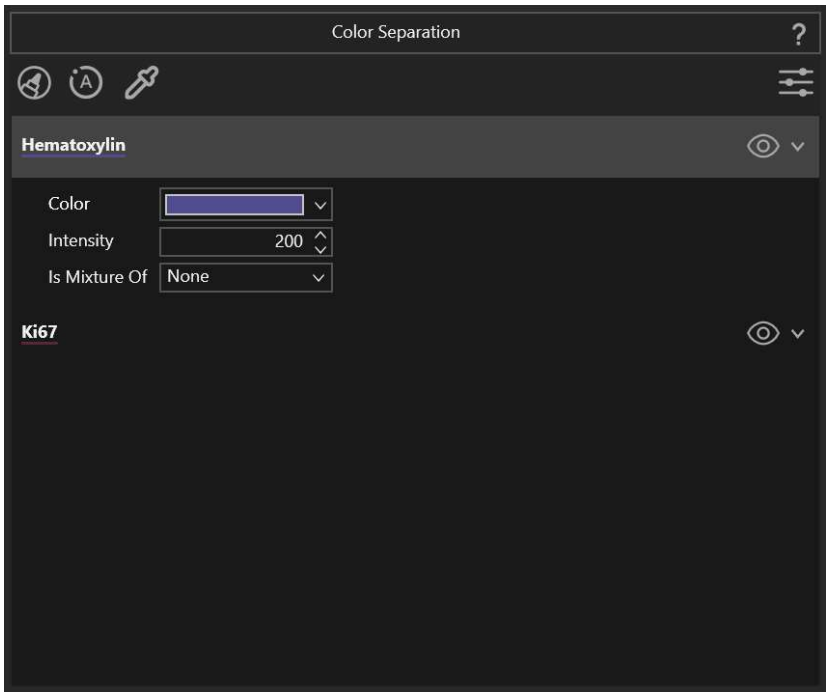


Figure 522 – Color Separation (SRS) engine settings (Hematoxylin)

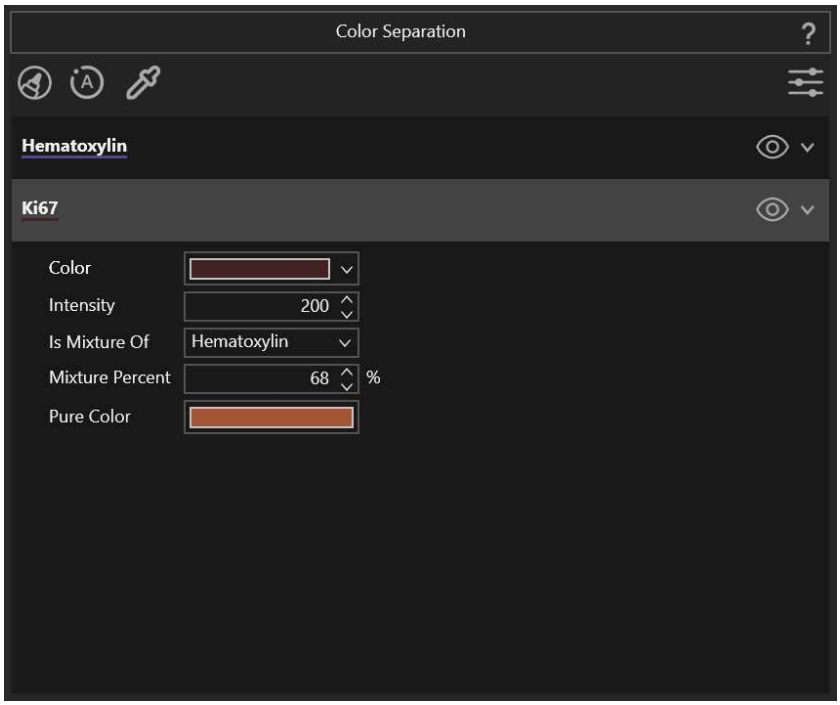


Figure 523 – Color Separation (SRS) engine settings (Ki67)

- Output:

Figures below show the grayscale images representing the 2 separated markers: Hematoxylin and Ki67.

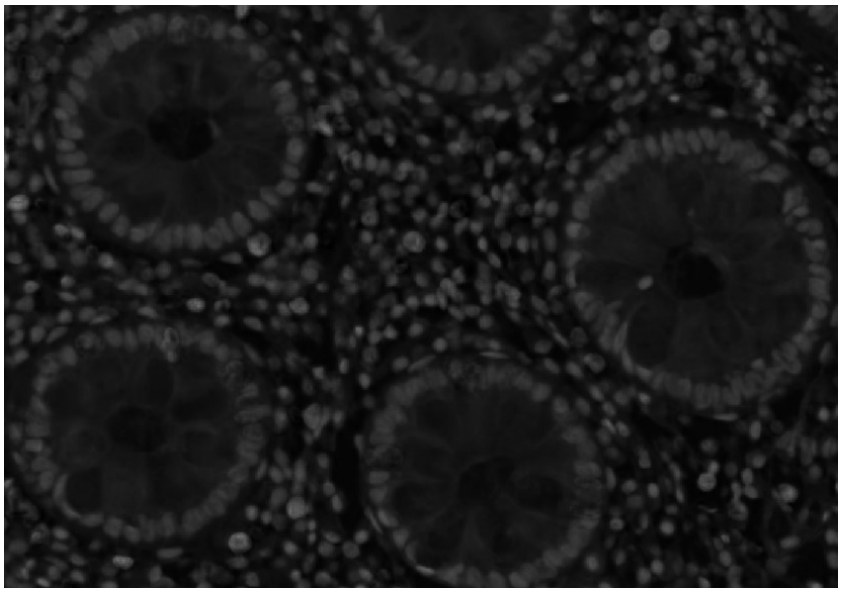


Figure 524 – Result of Color Separation (SRS) engine (Hematoxylin)

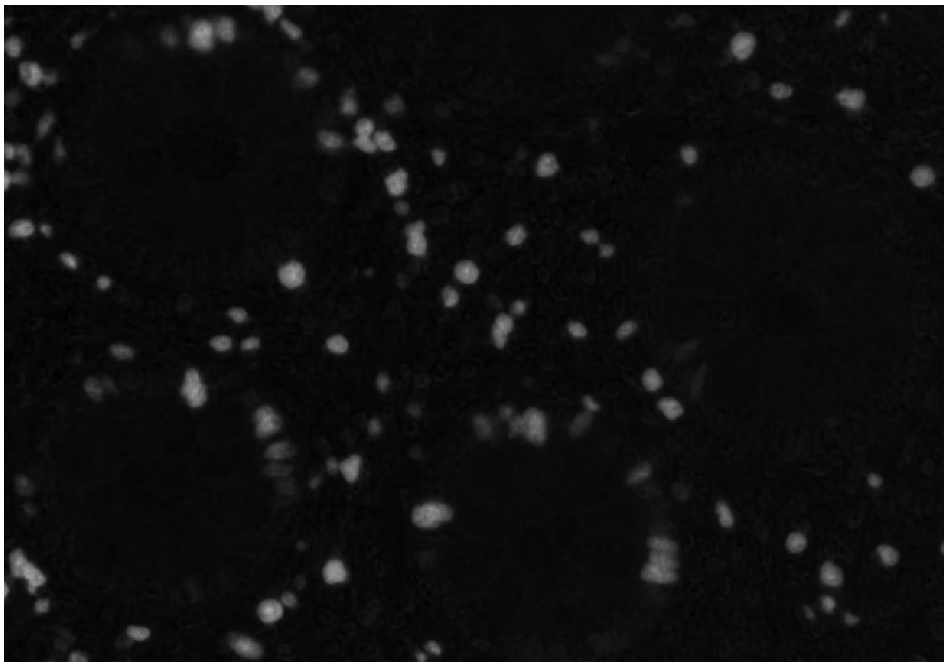


Figure 525 – Result of Color Separation (SRS) engine (Ki67)

9.2. Multiple Reference Shades

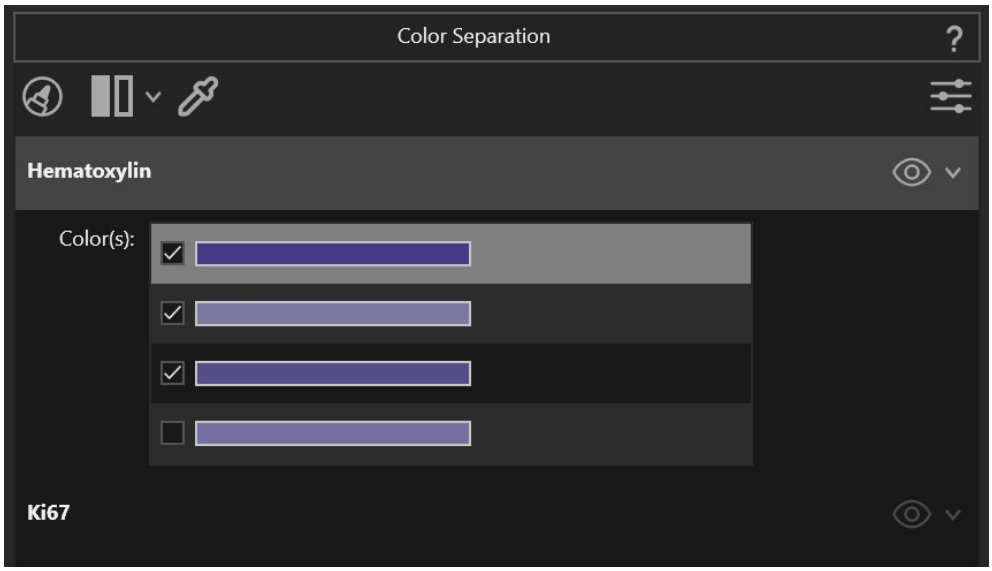


Figure 526 – Multiple Reference Shades (Hematoxylin marker)

A set of shades can be assigned for each marker. By default, each shade is also considered a measurement shade. If a shade is intended to be used only for detection, the corresponding Measure check box must be disabled.

This color separation method is a color clusterization process that estimates the probability for each pixel to belong to a given reference shade. The probability is

estimated based on the distance between the pixel color and the reference shades, in the selected color space. The smaller the distance, the brighter the output pixel value.

The output quality depends on the reference shades accuracy; selecting a reference shade representing a low / false positive marker presence will generate high grayscale values in the output image, for low / false positive marker presence. Confusing readouts can be generated if such a shade is used for measurements of marker intensity. Therefore, is important to have the Measure option enabled ONLY for shades representing a strong positive marker presence.

Adding shades to a maker:

- Identify a sample region where the maker to be defined express a good presence
- For easier identification and selection of the shades present in the image, the number of the colors

visible in the image can be reduced by activating the Separated option. A square will allow the selection of the area for which the number of colors will be reduced.

Figure 527 – Option to reduce the number of colors

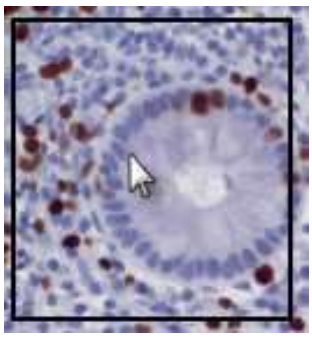


Figure 528 – Area selection for color reduction

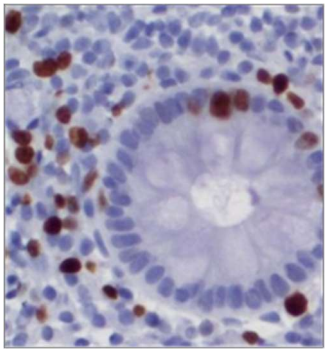


Figure 529 – Original image

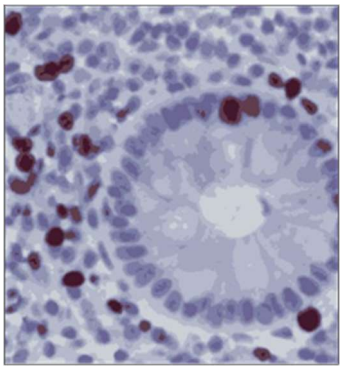


Figure 530 – Color reduce image

The process of reducing the color number can be customized by accessing Options:

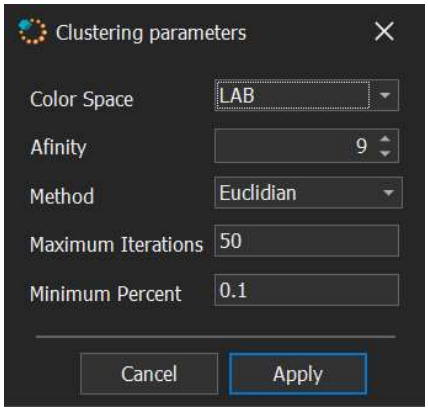


Figure 531 – Clustering parameters

- Color Space – specifies the color space used in the color separation process. The available options are:
  - RGB – the most common color representation. Not recommended because ligther shades (hues) of different colors might be grouped together
  - HSV / HSL – have the advantage that, in clustering, shades (hues) of different intensities are grouped together
  - LAB / LUV – recommended in clustering since they are closer to human eye perception
- Affinity – specifies the number of color groups present in the separated image
- Method – specifies the distance measure used between colors in the clustering process. The available options are:
  - Euclidian – the length of the straight line between two points
  - Manhattan – the distance between two points is measured along axes, at right angles (90 degrees)
- Maximum Iterations – specifies the maximum number of iterations of the color clustering process. Increasing the value may yield better results, but requires more time

- Minimum Percent – specifies the percent of pixels switching groups of colors between color clustering iterations. It acts like a threshold, if a smaller percent of pixels changed groups, it is considered an acceptable error / insignificant change compared with the last state of the color groups (previous iteration) and the clustering process is considered finished.
- New shades can be assigned to the selected marker by using the Color Picker functionality. Once activated, a color box will appear to the right of the button, showing the current mouse position color (collected from the image). If the current mouse cursor position is not valid position (not over an image), the displayed color will be white. The shade is collected from the image and added in the selected marker's list by clicking on the identified position in the original / color reduced image.

Marker options:

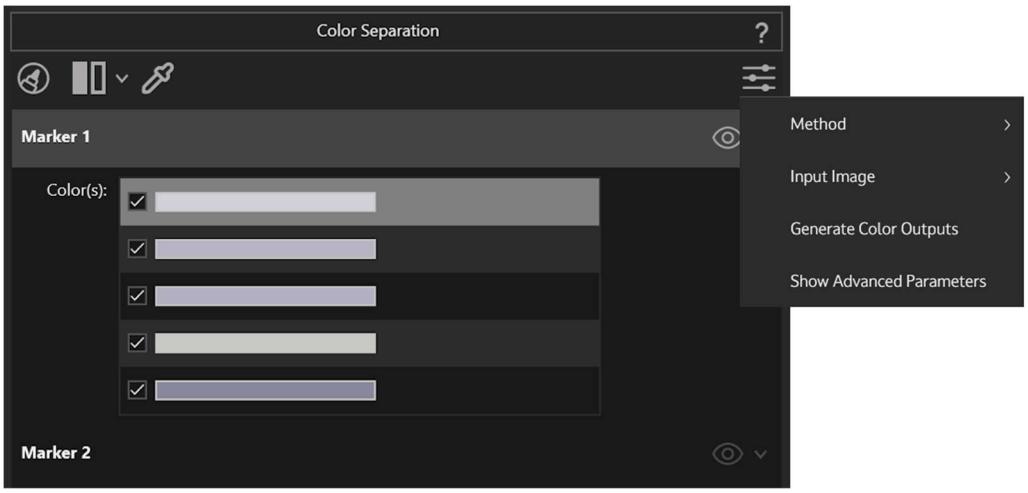


Figure 532 – Multiple Reference Shades - Settings

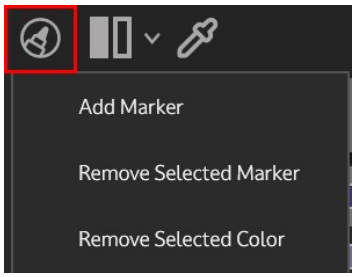


Figure 533 – Shades contextual menu

Shade options:

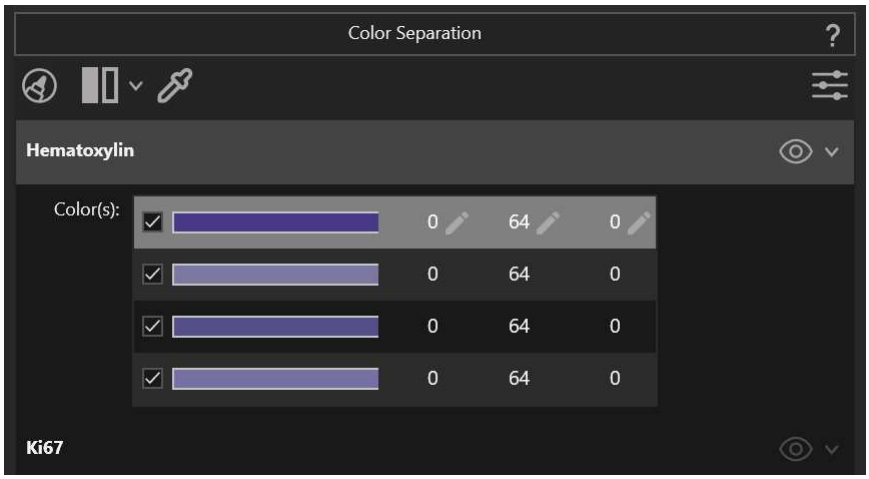


Figure 534 – Shade advanced options

1. Measure – enables the usage of the current shade's grayscale image for measurements
2. Shade – the color of the selected shade
3. Gain – increase the brightness of the grayscale image associated with the selected shade
4. Sigma – relaxes the accuracy of pixels colors to be assigned with the selected reference shade
5. Gauss – smooths / blurs the grayscale image associated with the selected shade
6. Color Space – specifies the color space used to generate the grayscale image for each shade

Example:

- Input:

The image is a brightfield color image (8-bit, 3-channel) representing an IHC small region from a colon sample.

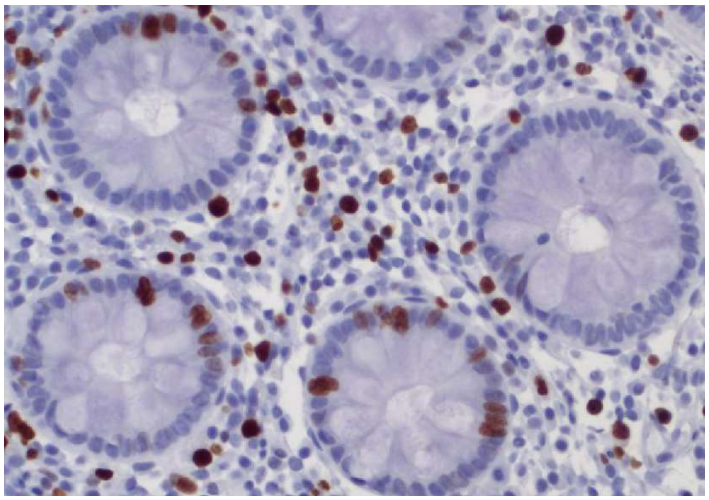


Figure 535 – IHC colon sample

- Engine settings:

Figures below show the engine settings used for this example.

The engine's results are grayscale (default) / color (if enabled) images.

Figure 536 – Color Separation (MRS) engine settings (Hematoxylin)

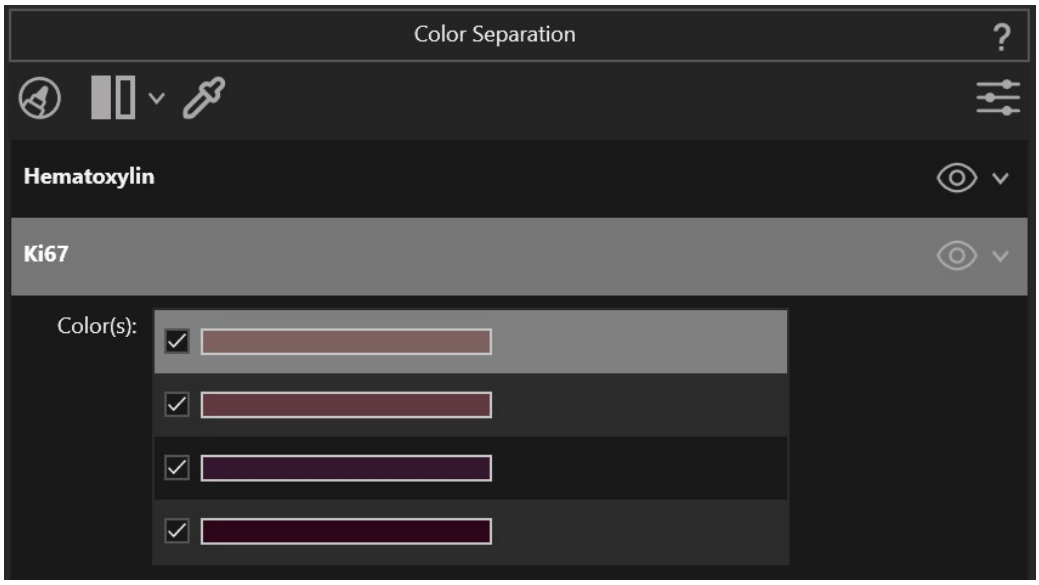


Figure 537 – Color Separation (MRS) engine settings (Ki67)

- Output:

The result has the brightness increased differently for the 3 channels (R=20 / G=20 / B=50) compared with the input image.

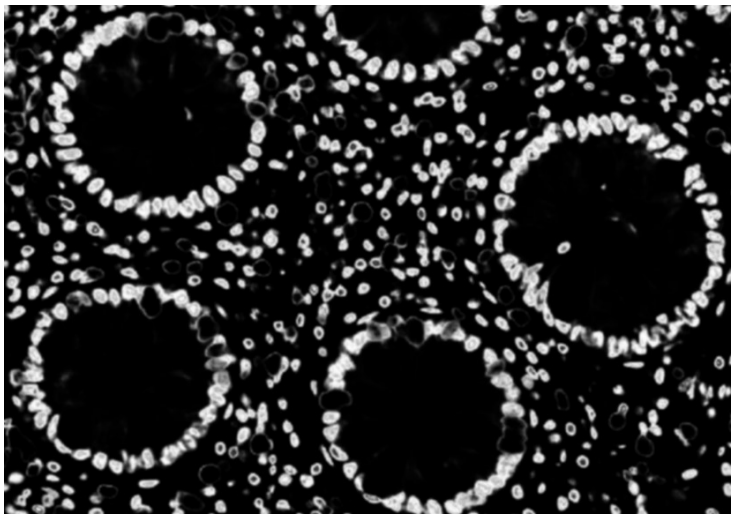


Figure 538 – Result of Color Separation (MRS) engine (Hematoxylin)

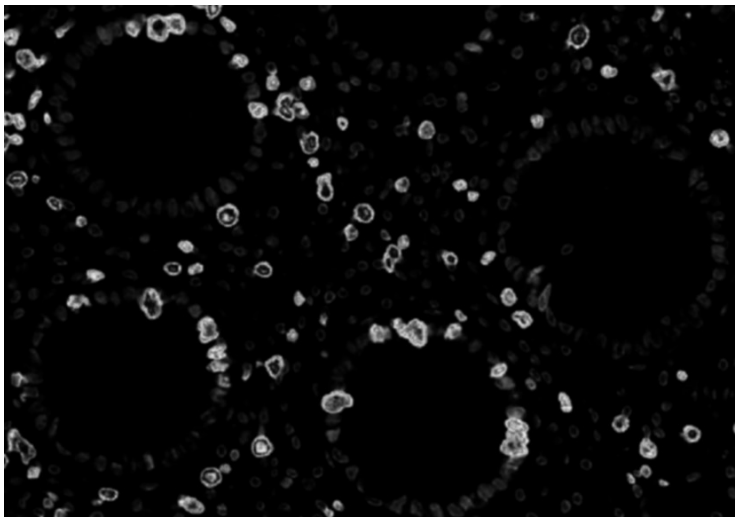
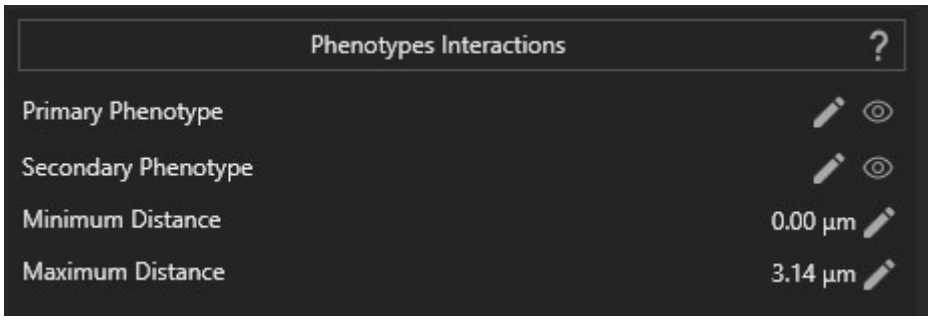


Figure 539 – Result of Color Separation (MRS) engine (Ki67)

10. Phenotype Interactions



Where it can be found:

Layer's "Pre-Processing" ► Operations (advanced) ► Phenotypes Interactions

Layer's "Measurements" ► Measurements (advanced) ► Phenotypes Interactions

Description:

We define an interaction as two cells: one from the primary phenotype and another from the secondary phenotype that are inside a specific distance interval (between "Minimum Distance" and "Maximum Distance").

Primary Phenotype:

- The primary phenotype. Measurements are generated for this phenotype.

Secondary Phenotype

- The secondary phenotype. This phenotype interacts with the primary phenotype, but measurements are not generated for this phenotype.

Minimum Distance

- minimum distance for interaction between the primary and secondary phenotype. Use value 0 to generate all interactions under "Maximum Distance"

Maximum Distance

- maximum distance for interaction between the primary and secondary phenotype.

Parameters

- selected measurements (only when the engine is used in the Measurements Category

Count in Range

- for each "Primary Phenotype" cell: the number of "Secondary Phenotype" cells within the distance interval (between "Minimum Distance" and "Maximum Distance").

Mean Range

- for each "Primary Phenotype" cell: the average distance of "Secondary Phenotype" cells within the distance interval (between "Minimum Distance" and "Maximum Distance").

Minimum Range

- for each "Primary Phenotype" cell: the distance to the closest "Secondary Phenotype" cell within the distance interval (between "Minimum Distance" and "Maximum Distance").

Maximum Range

- for each "Primary Phenotype" cell: the distance to the most distant "Secondary Phenotype" cell within the distance interval (between "Minimum Distance" and "Maximum Distance").

Closest ID in Range

- for each "Primary Phenotype" cell: the ID of the closest "Secondary Phenotype" cell within the distance interval (between "Minimum Distance" and "Maximum Distance").

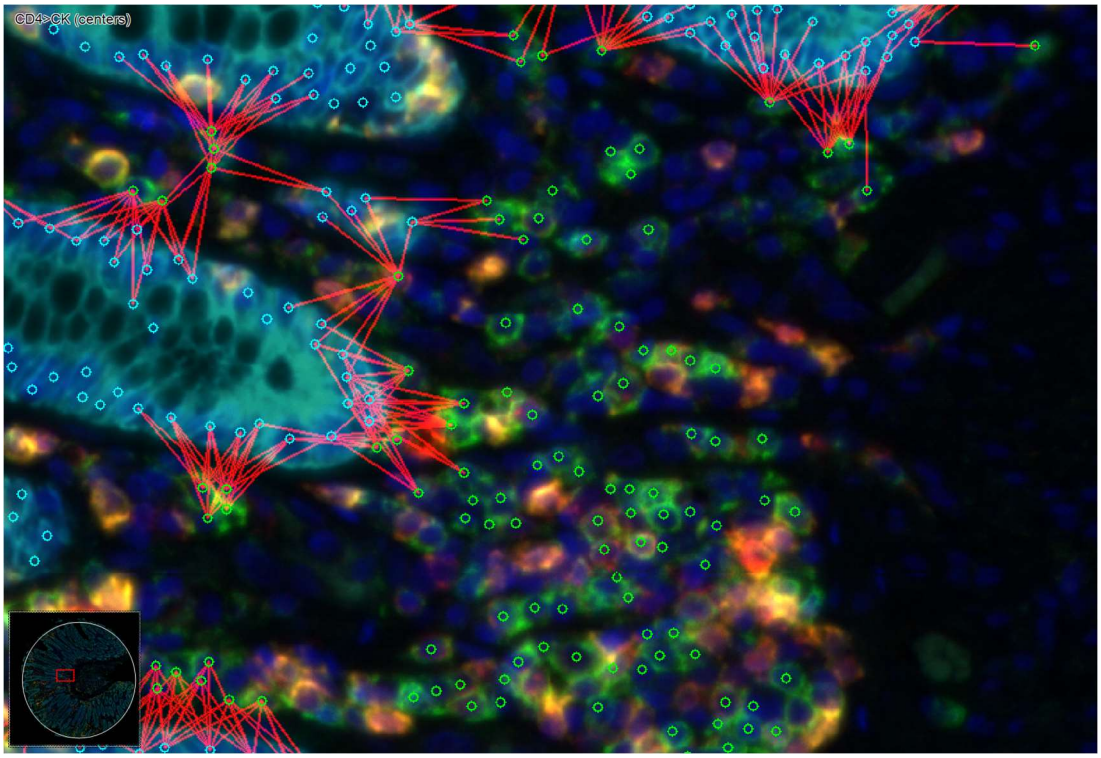
Draw Only Closest Interaction

- Draw only the closest interactions for each "Primary Phenotype"

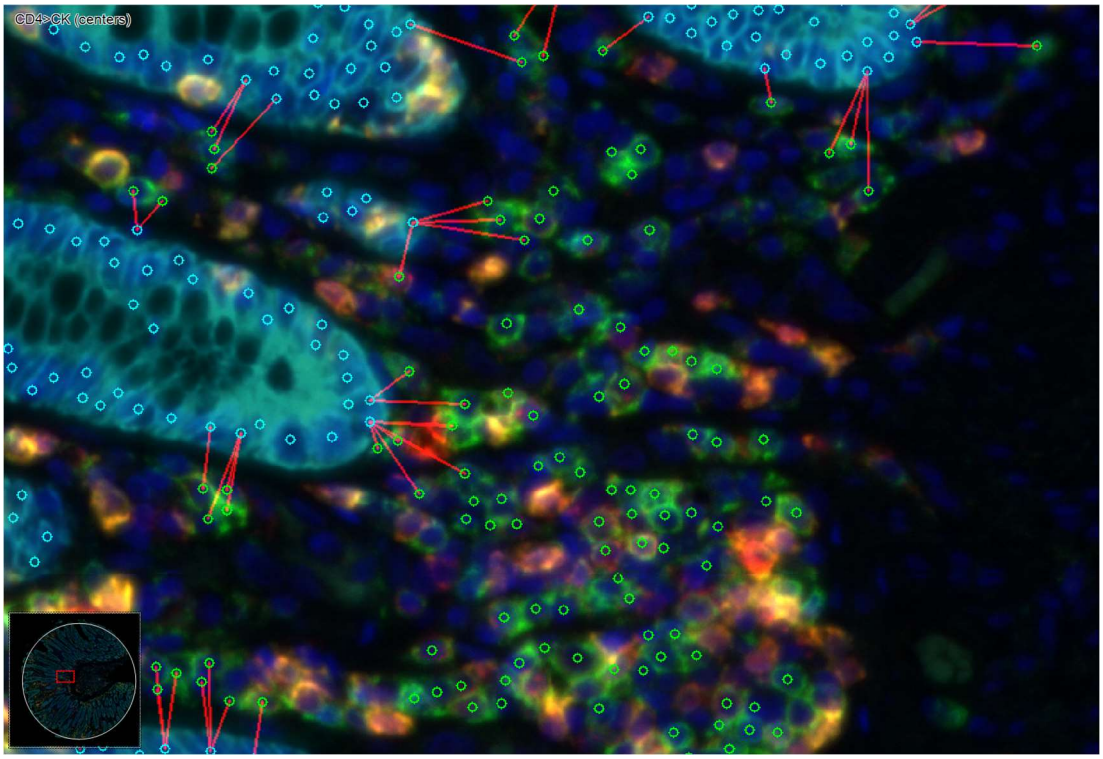
Examples

1. Draw closest conn

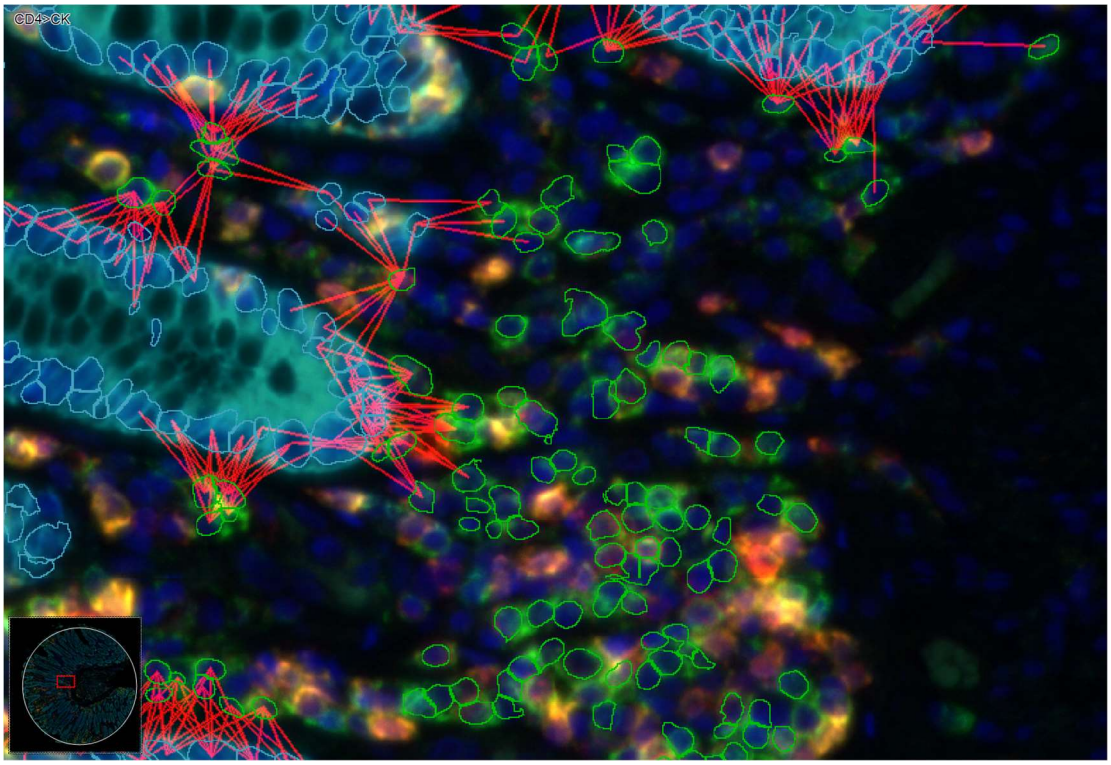
Ranges:
- min distance, max distance: (0, 30)
- show advanced parameters - draw only closest interactions: checked



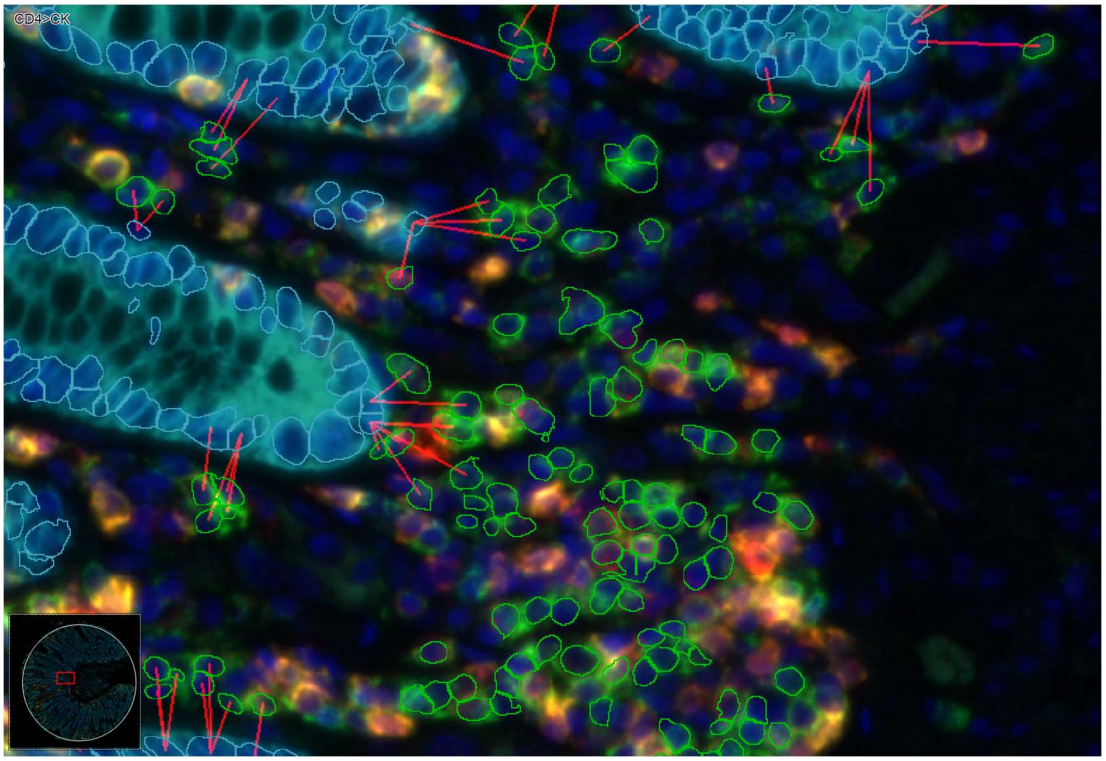
CK-to-CD4_centers_30um_all_conn



CK-to-CD4_centers_30um_closest_conn



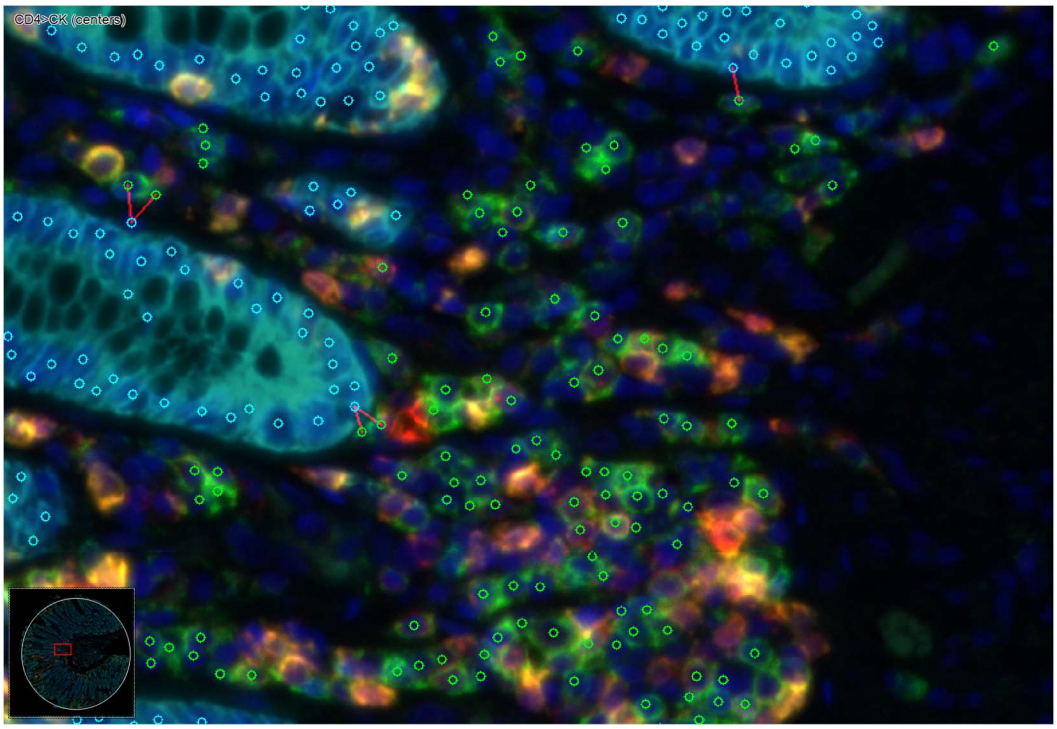
CK-to-CD4_nucl_30um_all_conn



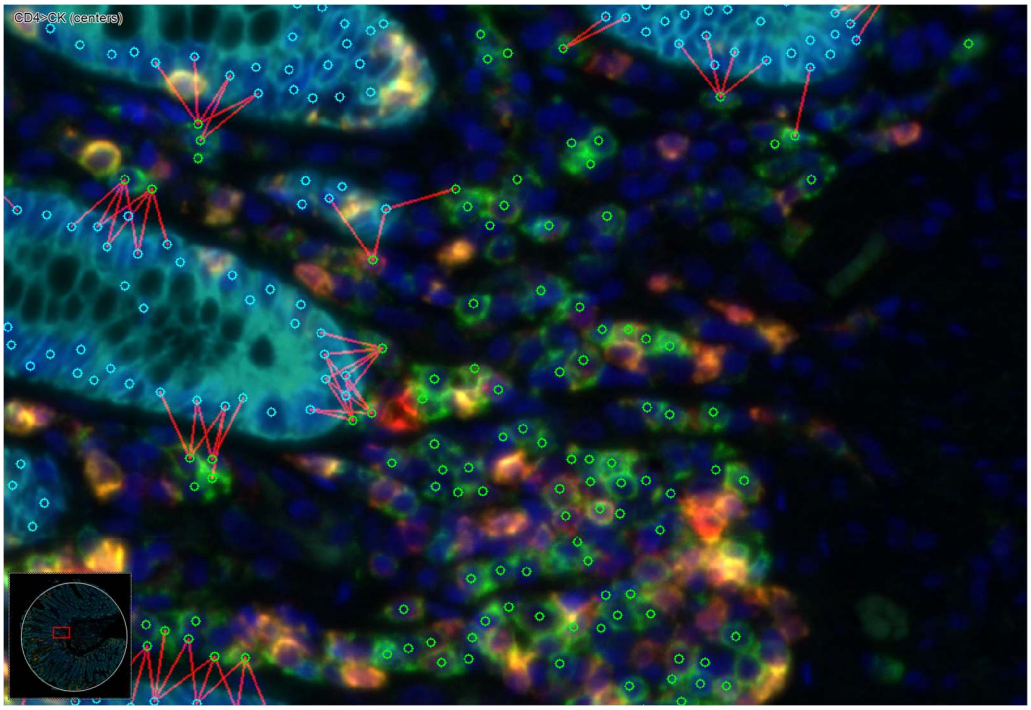
CK-to-CD4_nucl_30um_closest_conn

2. Intervals

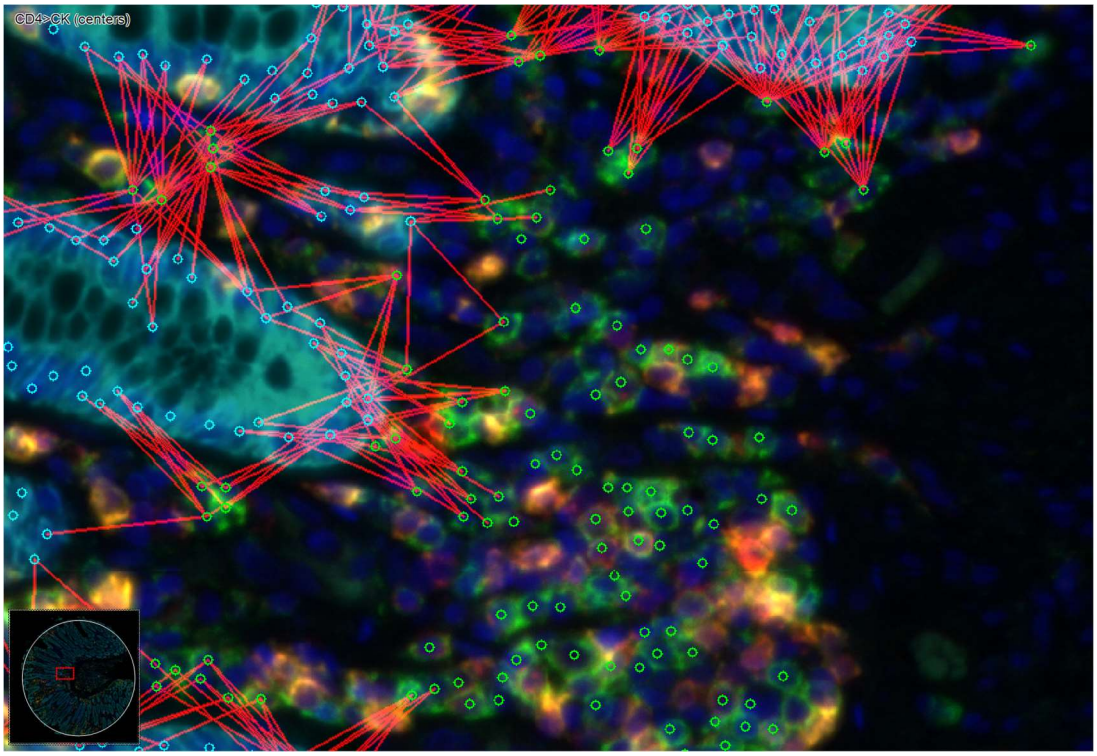
- min distance, max distance: (0, 10), (10, 20), (20, 30), (30, 40)
- show advanced parameters - draw only closest interactions: unchecked



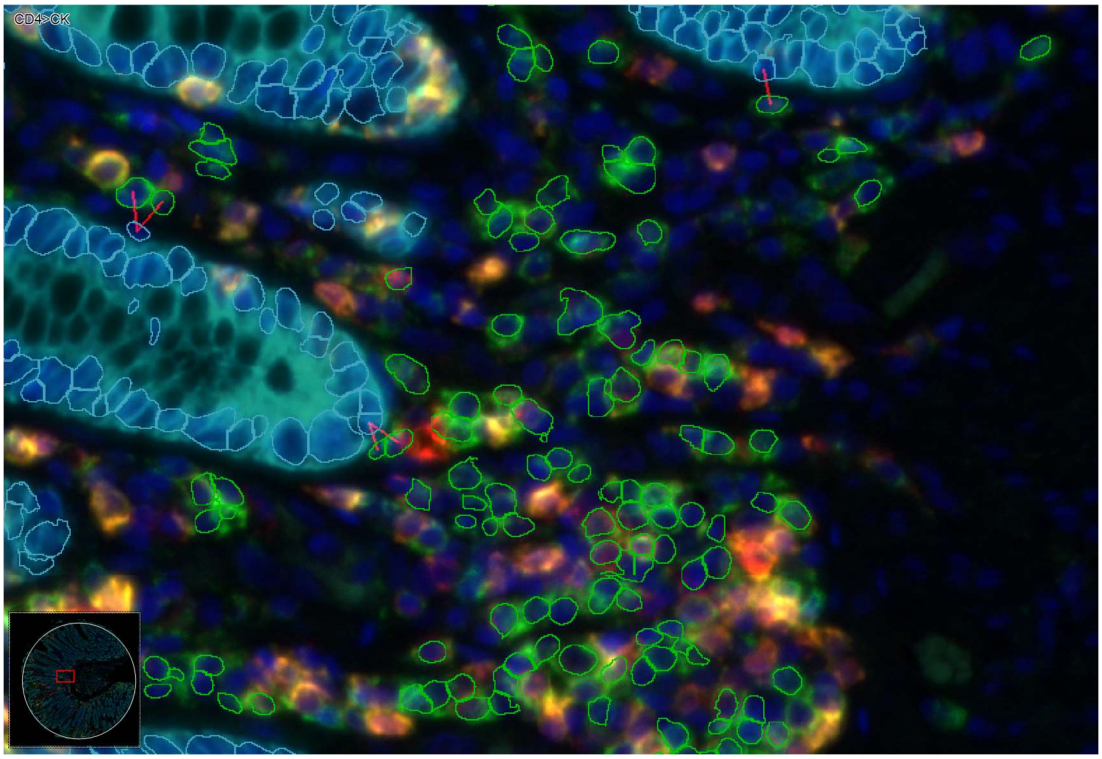
CK-to-CD4_centers_0-10um



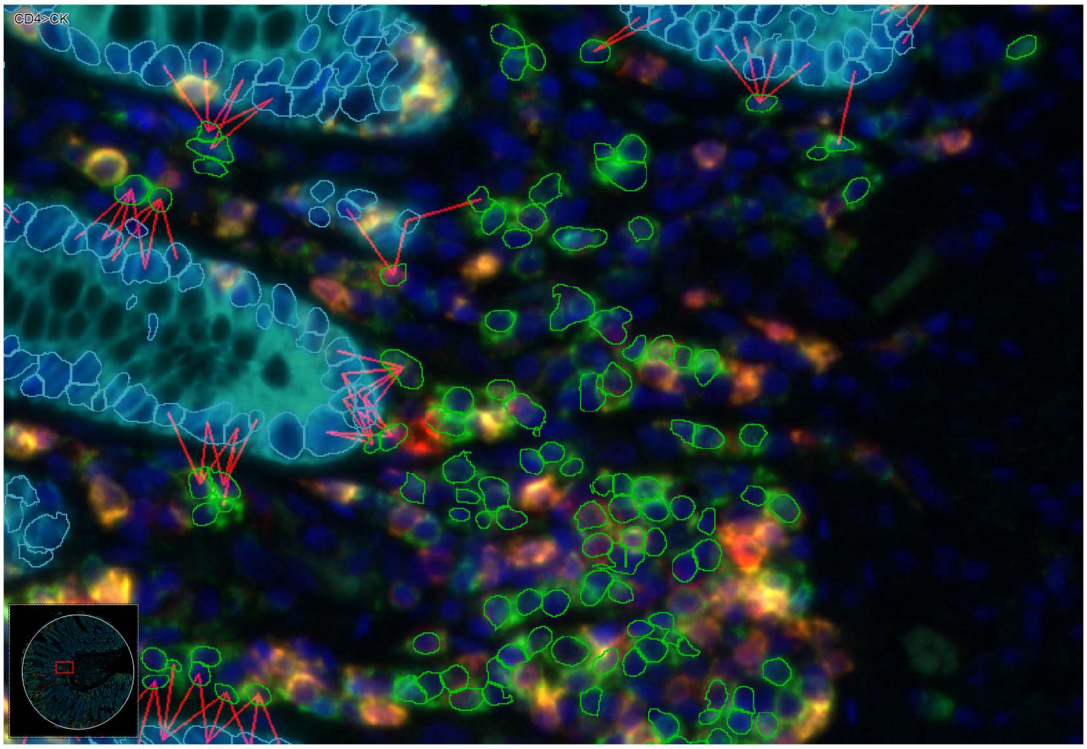
CK-to-CD4_centers_10-20um



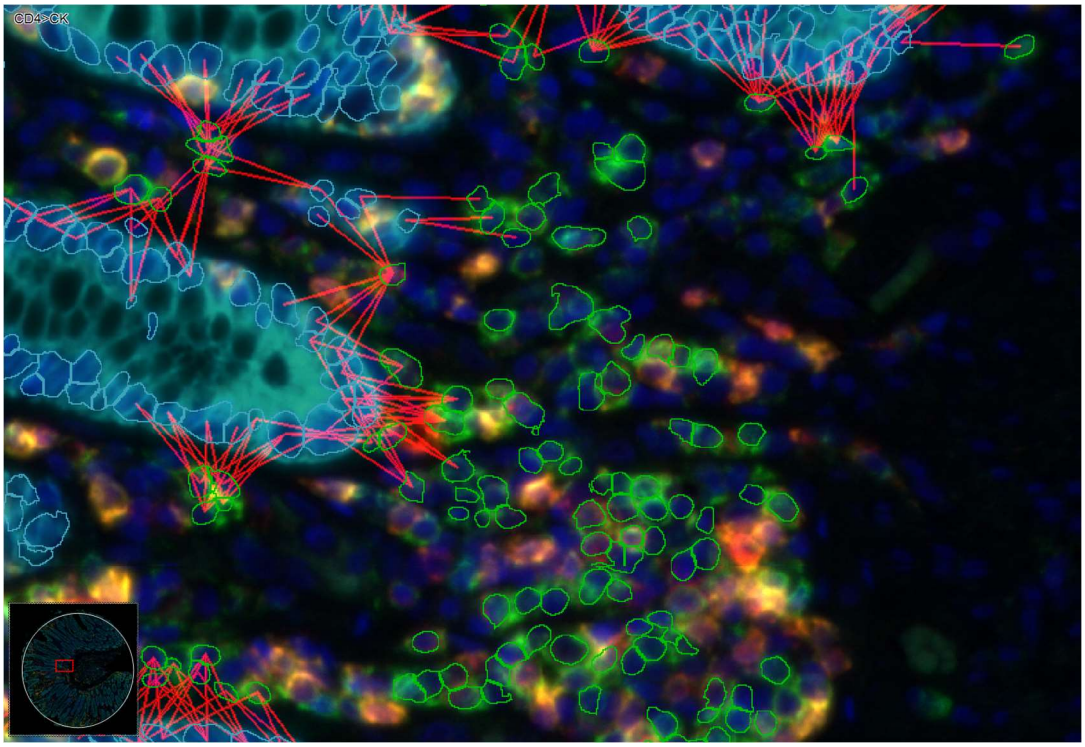
CK-to-CD4_centers_30-40um



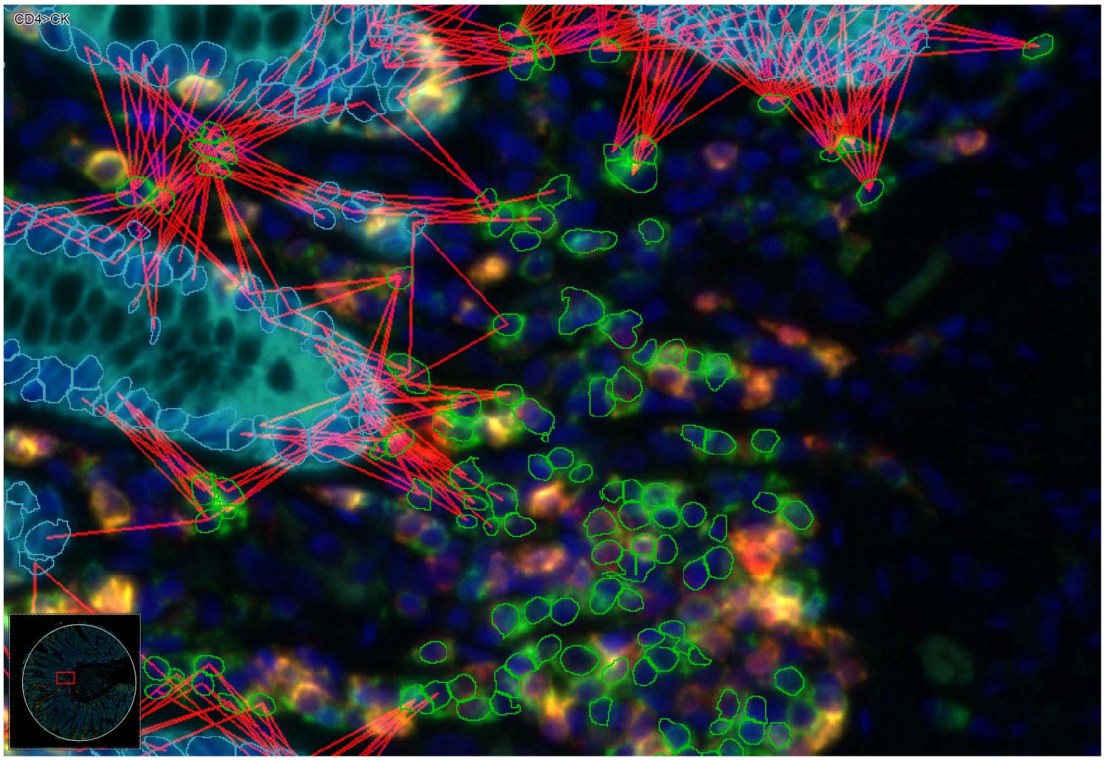
CK-to-CD4_nucl_0-10um



CK-to-CD4_nucl_10-20um



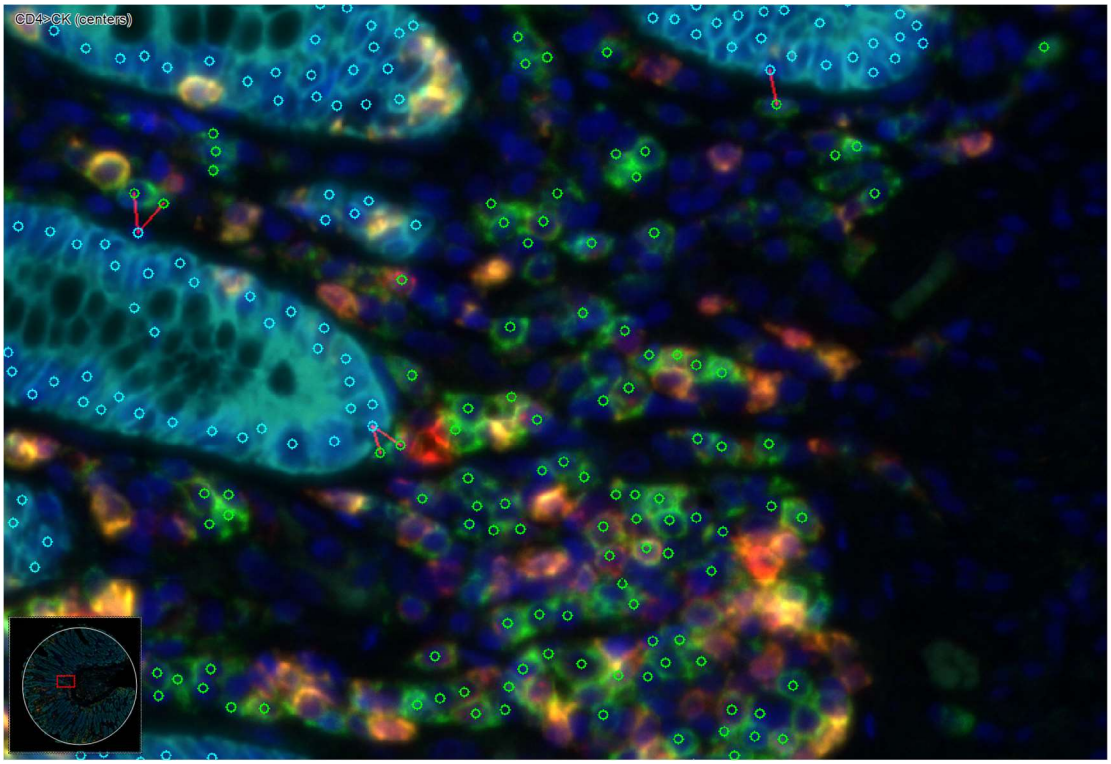
CK-to-CD4_nucl_20-30um



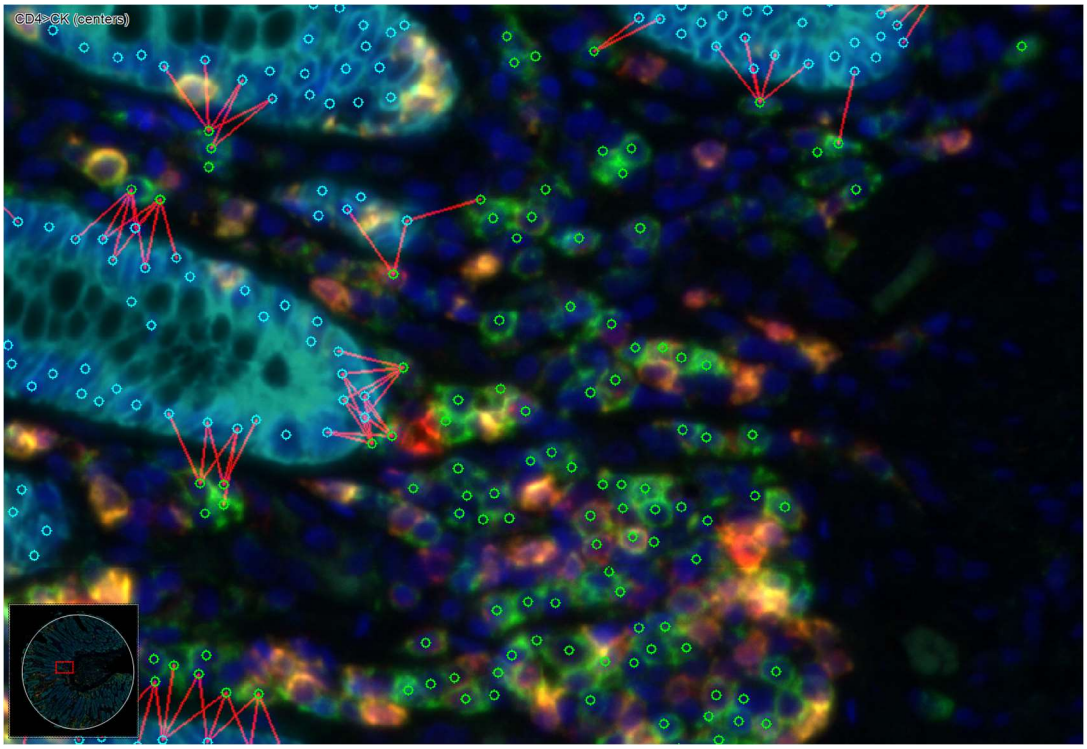
CK-to-CD4_nucl_30-40um

3. Range (0 - x)

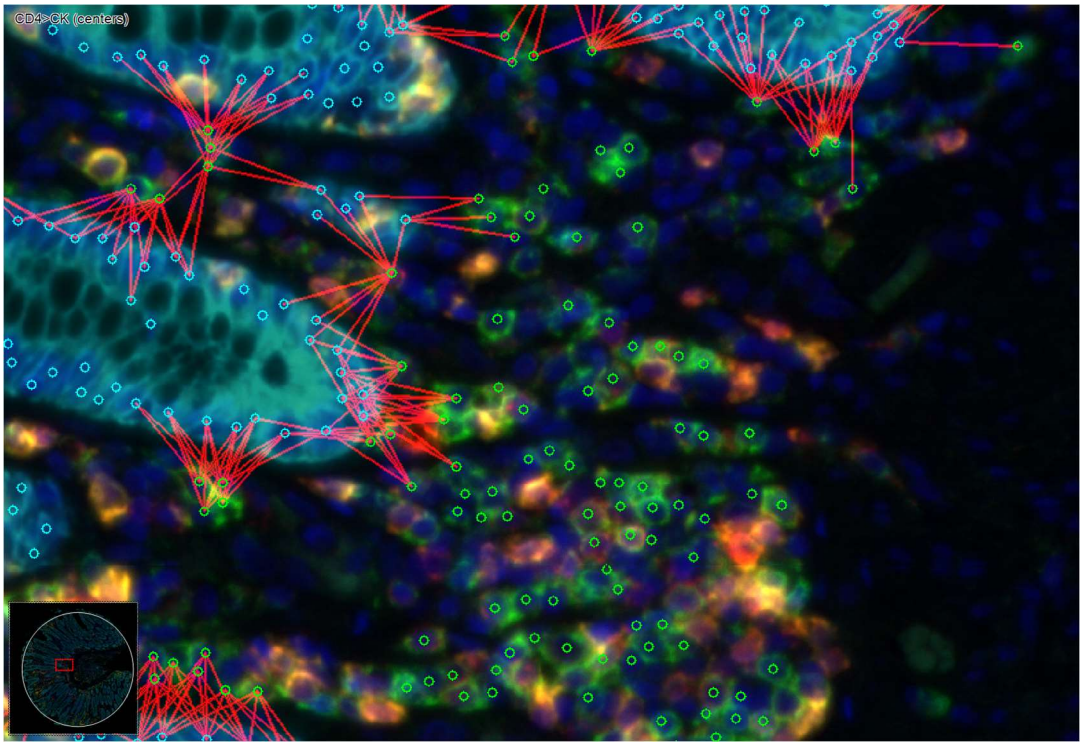
- min distance, max distance: (0, 10), (0, 20), (0, 30), (0, 40)
- show advanced parameters - draw only closest interactions: unchecked



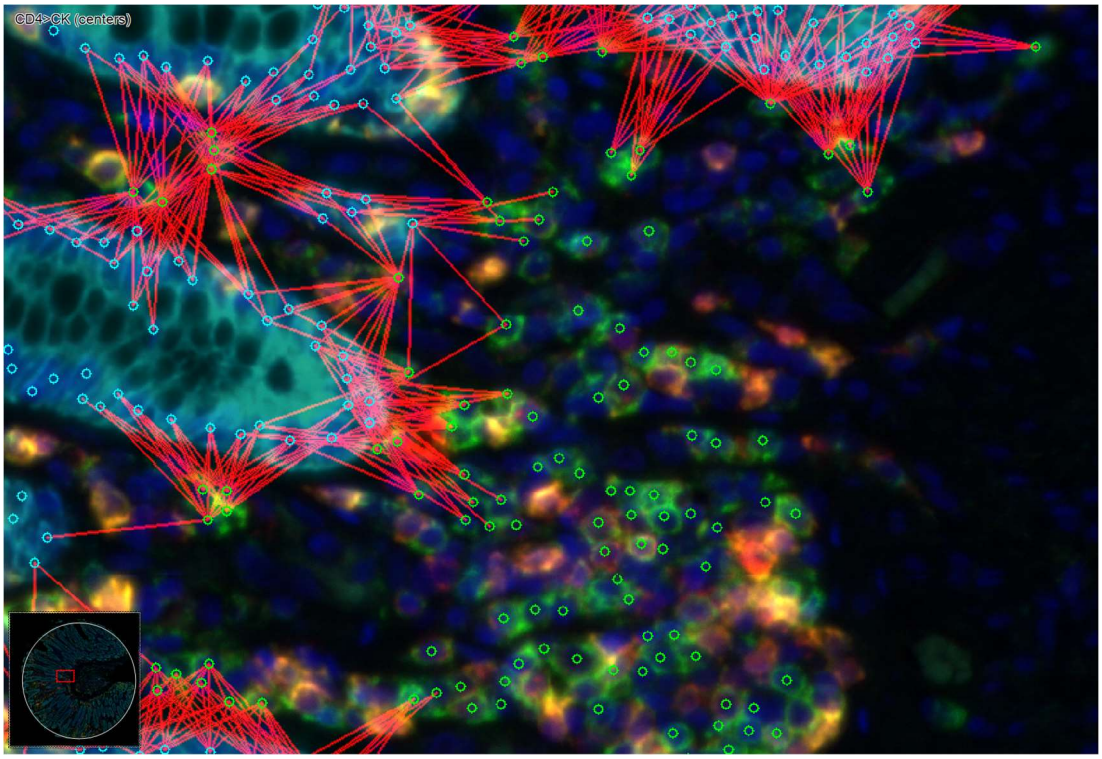
CK-to-CD4_centers_10um



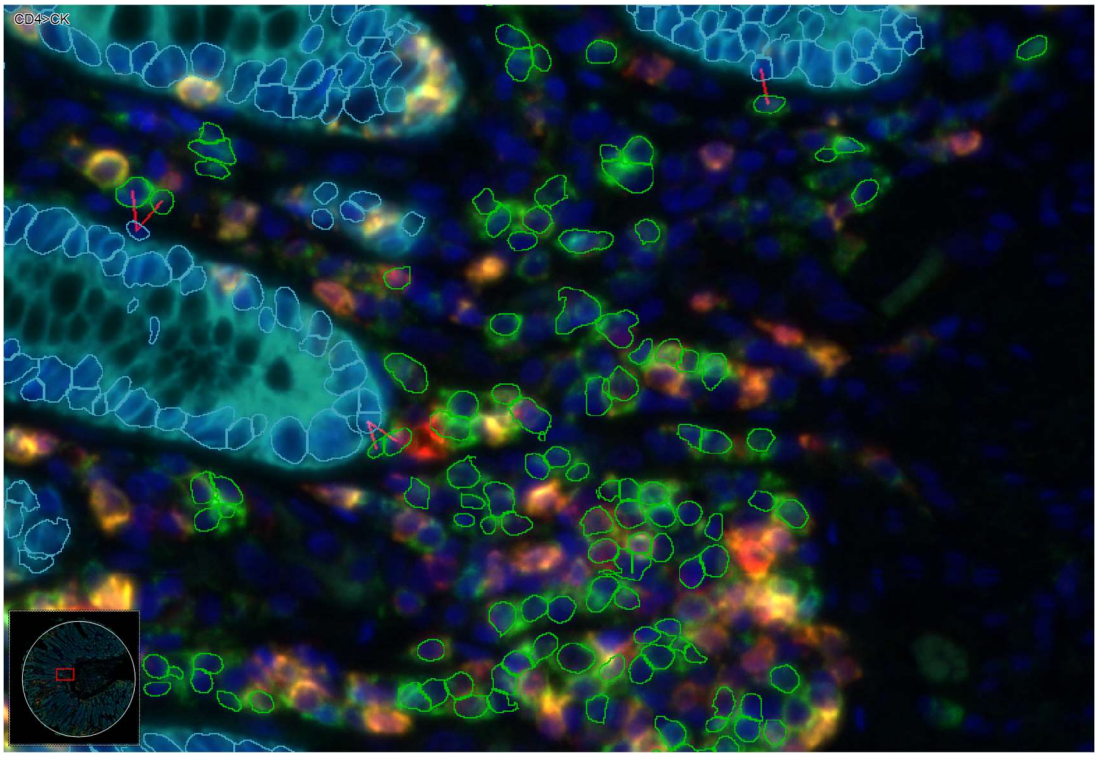
CK-to-CD4_centers_20um



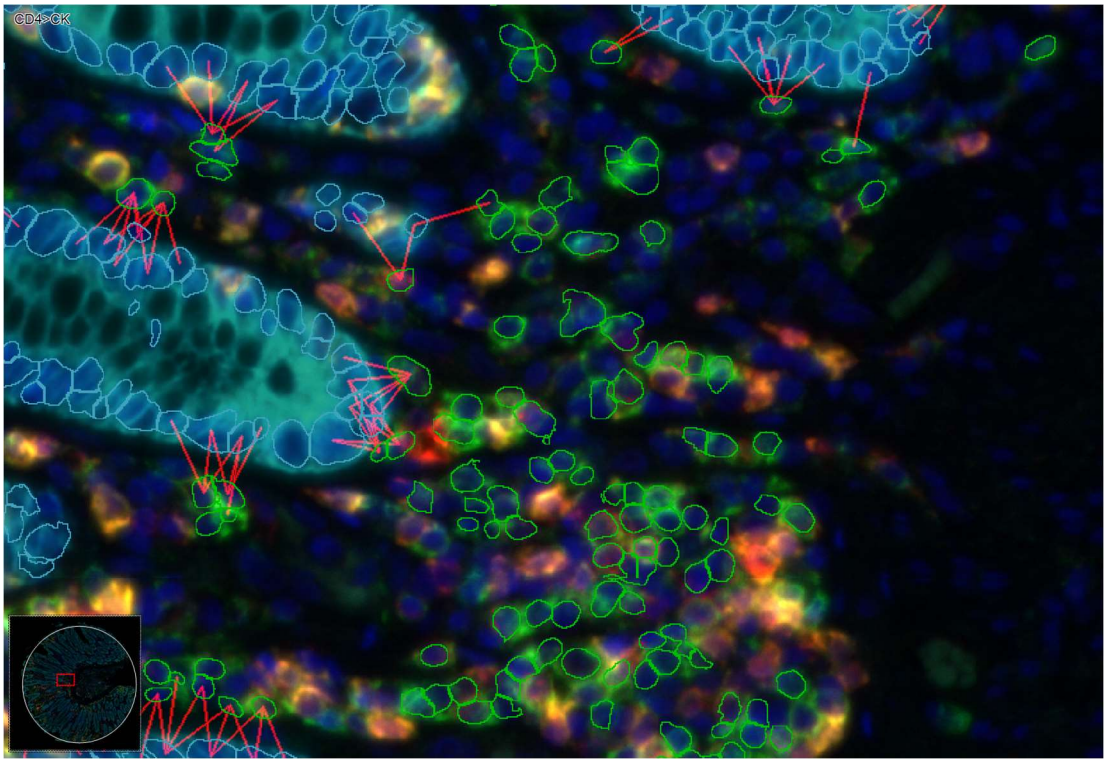
CK-to-CD4_centers_30um



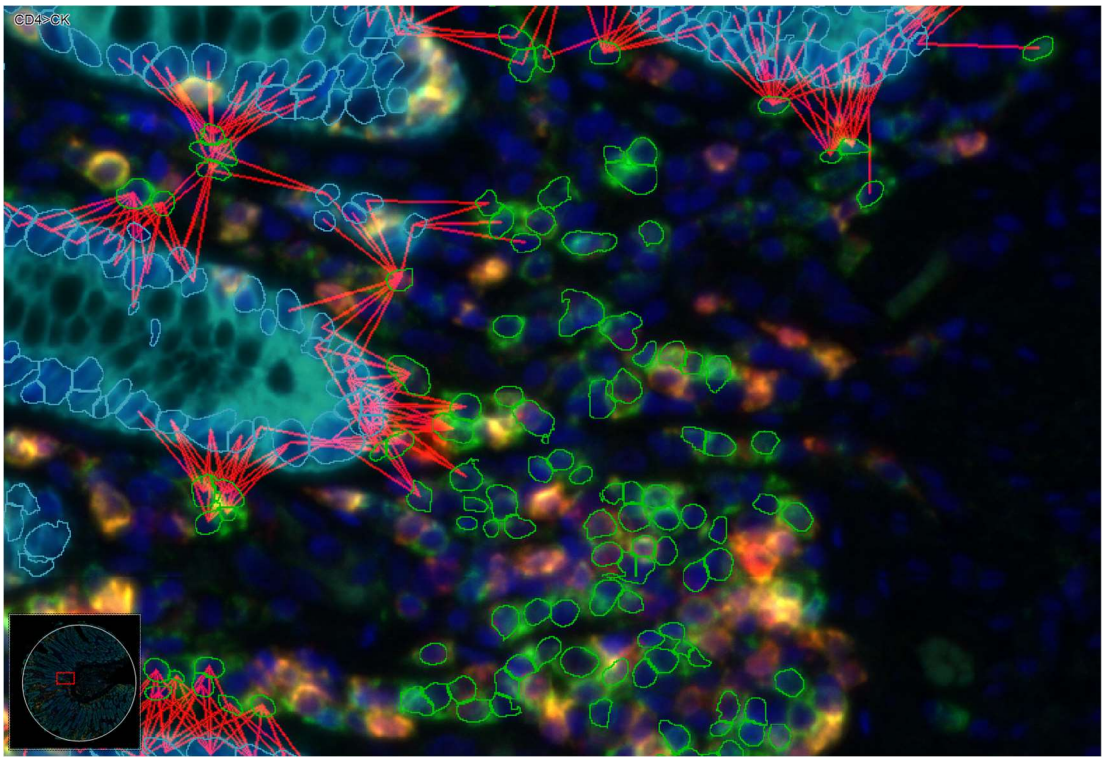
CK-to-CD4_centers_40um



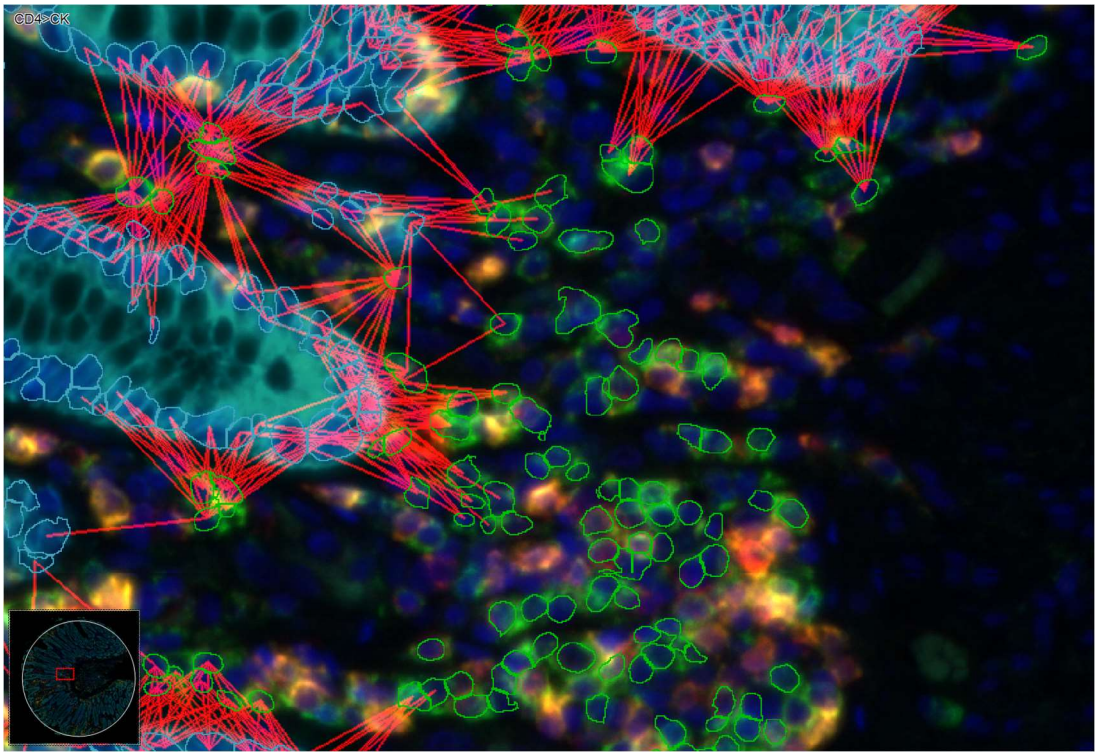
CK-to-CD4_nucl_10um



CK-to-CD4_nucl_20um



CK-to-CD4_nucl_30um



CK-to-CD4_nucl_40um

11. Density of Events

Figure 546 – Density of events

Where it can be found:

Layer's "Pre-Processing" ► Add ► Operations (advanced) ► Density of Events

Description:

Density of Events is an engine available in the Pre-Processing section of the layers editor.

This engine uses a set of events represented by a coded image to generate a grayscale image, highlighting a high number of events. In the output image, the high values indicate a high possibility of having many events in the user defined proximity, while a low value indicates the opposite.

The engine uses a Gaussian model to search for events in the defined proximity area. This means closer pixels are more relevant than distant ones.

Parameters:

- Coded - inputs a coded image representing the set of events;
- Kernel radius (px) - determines the size of processing window (default = 1) using the formulas:

$$Kernel\ width = 2 * \boldsymbol{Kernel\ radius(px)} + 1$$

$$Kernel\ height = 2 * \boldsymbol{Kernel\ radius(px)} + 1$$

- Kernel shape - specifies the shape of the kernel (default = Circle)
- Standard deviation - controls the distribution of weights in the generated Gaussian model. Indicates the amount of variation or dispersion of a data set. A value close to 0 indicates that the data points tend to be very close to the mean of the set, while a high value indicates that the data points are spread out over a wider range of values (default = 1

- Border size - the size of the border added to the current FOV. The border consists of pixels from neighboring FOVs, in order to preserve the continuity of the information (default = 100 pixels)

Effect:

Generates a grayscale image, where high values indicate the presence of a dense population of events.

Example:

- Input:

Figure below shows a set of events representing detected nuclei on an IHC colon sample. The events represent the input for this example.

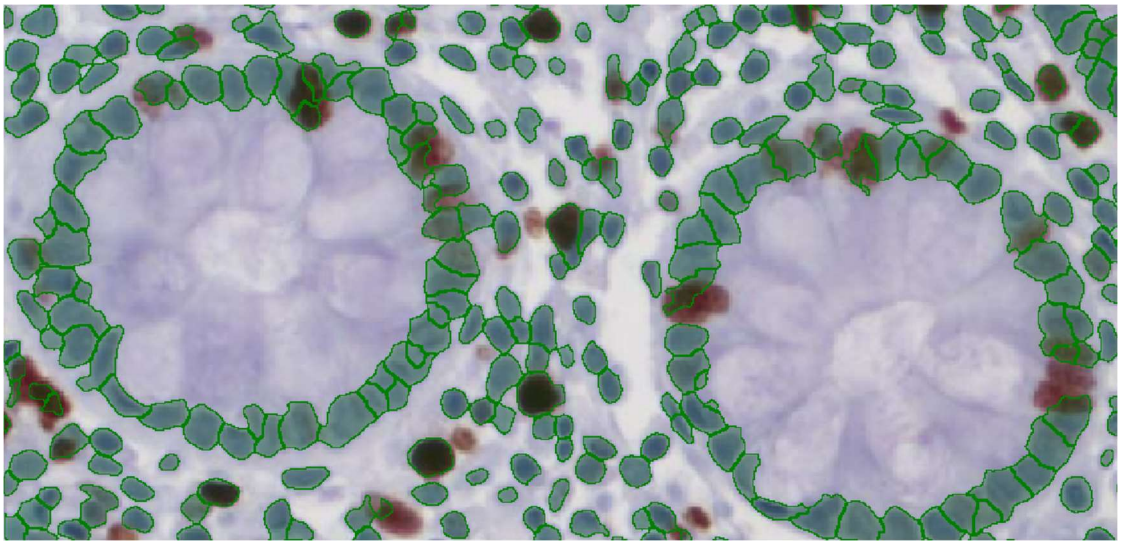


Figure 547 – Detected nuclei on IHC colon sample

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a grayscale image indicating the probability of having multiple events present in the vicinity of the area defined by the kernel radius.

Figure 548 – Engine settings

Output:

Figure below shows the engine result, where brighter areas are associated with a high probability of having multiple events in the vicinity.

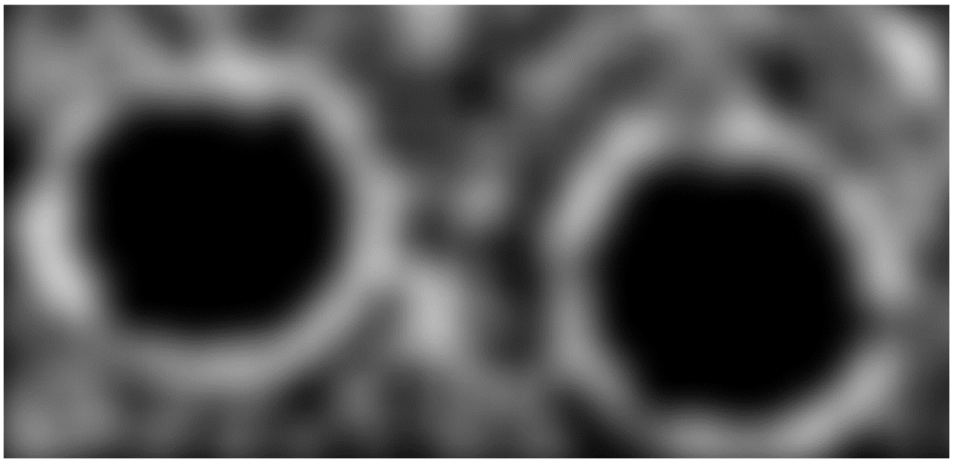


Figure 549 – Result of Density of Events engine

12. Derived Measurements

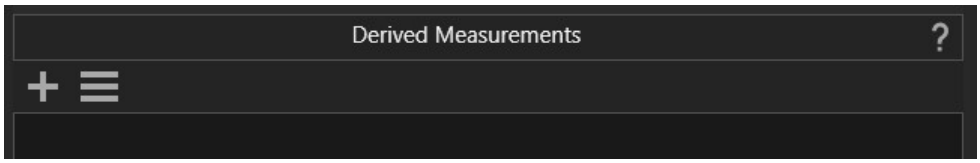


Figure 550 – Derived Measurements

Where it can be found:

Layer's "Measurements" ► Add ► Measurements (advanced) ► Derived Measurements

Description:

Derived Measurements is an engine available in the Measurements section of the layers editor.

This engine generates new measurements derived from the existing ones, using basic arithmetic operations. The possible combinations are:

- Measurement and measurement
- Measurement and user defined value

Parameters:

- New - defines a new derived measurement
- Edit - edits an existing derived measurement
- Remove - deletes an existing derived measurement
- Remove All - deletes all existing derived measurements

Effect:

Generates new measurements derived from the existing ones.

Example 1:

Having measured the area for nuclei and cytoplasm, a new derived measurement can be defined, representing the Total Cell Area, where:

$$Total\ Cell\ Area = Area\ (Nucleus) + Area\ (Cytoplasm)$$

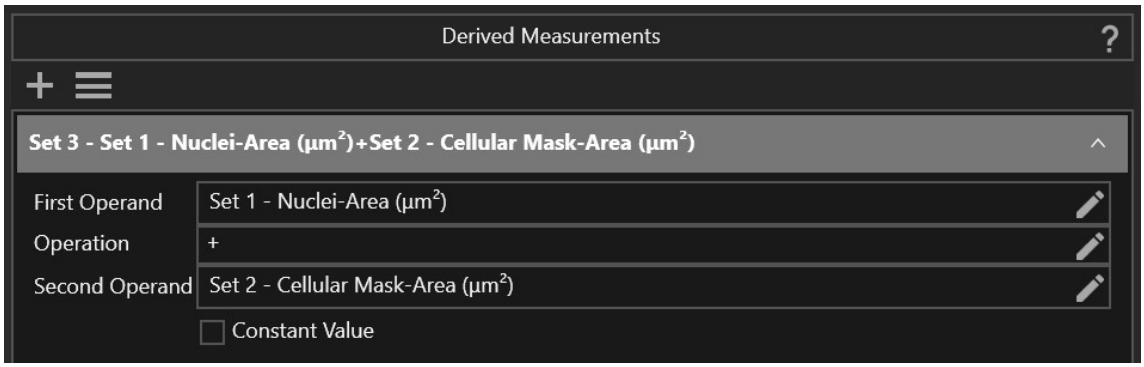


Figure 551 – Adding derived measurement

Example 2:

Conversion of a measurement unit from pixels to µm (or vice-versa). Events distances are usually computed using pixels as a measurement unit. The conversion to the metric system can be done by multiplying the distance value with the size of the pixel. The pixel size can be found (or edited) from the FOV Size window, which is found in projects Properties list from Main Window.

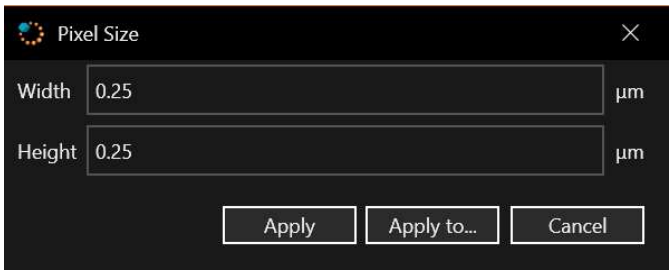


Figure 552 – Adjusting pixel size

Figure 553 – Example of conversion: pixels to µm

13. Distance Map Outside (global)

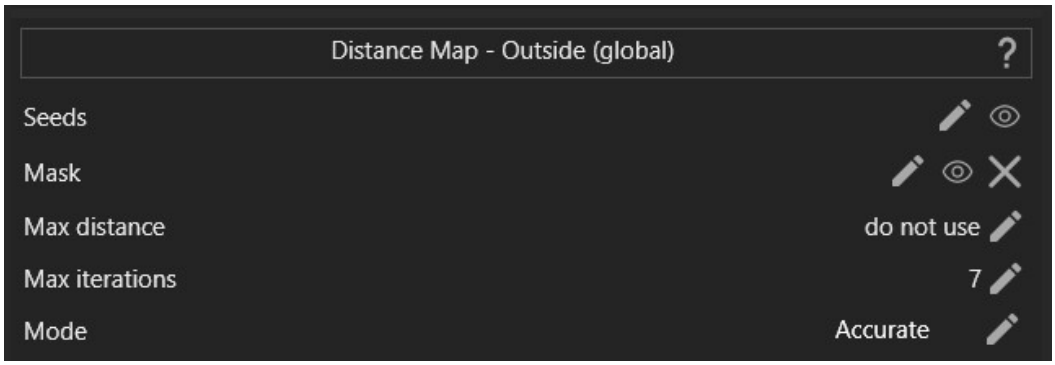


Figure 554 – Distance Map Outside (global)

Where it can be found:

Layer's "Pre-Processing" ► Add ► Operations (advanced) ► Distance Map - Outside (global)

Description:

Distance Map (outside) is an engine available in the Pre-Processing section of the layers editor.

This engine computes the distance value for all background pixels (pixels value = 0 / black) relative to the closest foreground pixel (pixels value = 255 / white). In other words, it computes the outer distances relative to events contours.

The distance map is stored using 32-bit datatype for better accuracy. The visualization is realized with the help of a color image using Parula colormap, where colors vary from blue (minimum value) to red (maximum value).

Figure 555 – Parula colormap

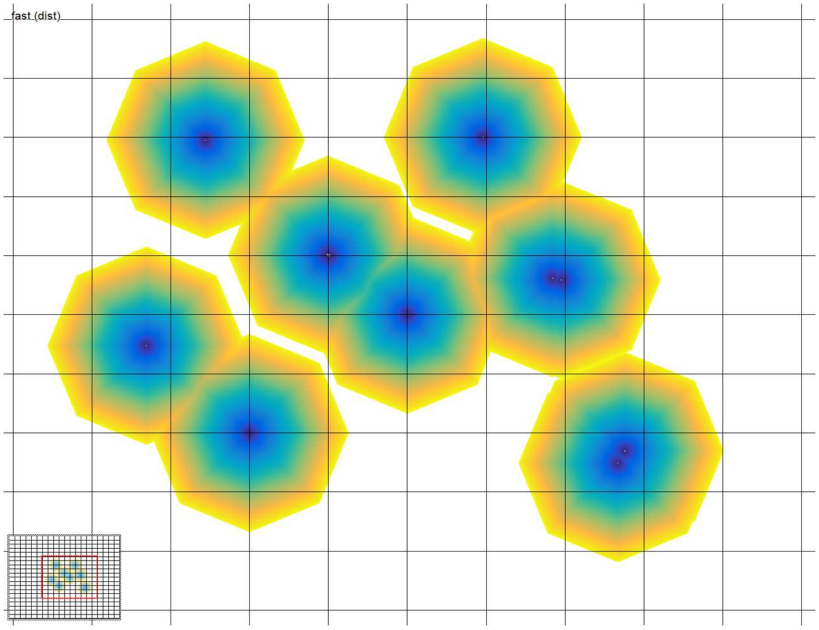
Note	Because a color map is used to display the distance map, Pixel Inspector is needed to find the correct distance value of a pixel.

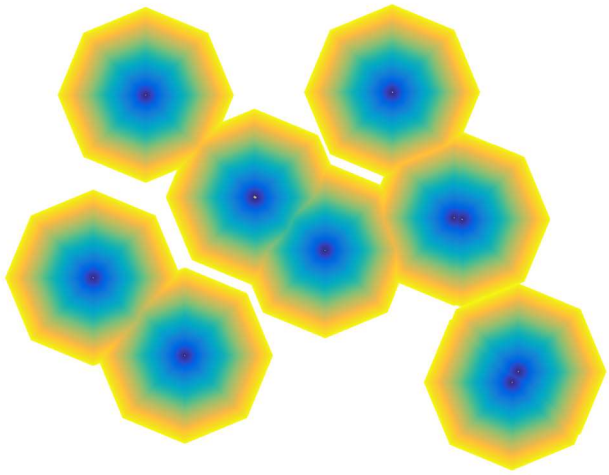
Parameters:

- Seeds - the input coded image representing the areas of interest (from which the distances will be computed)
- Mask - the input mask image. The distance values are computed using only paths inside the indicated areas (white).
- Maximum distance - the upper limit for the distances

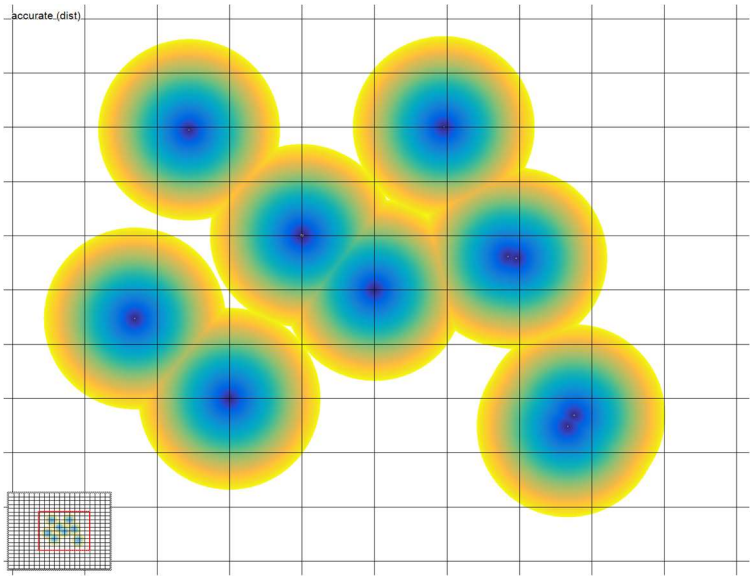
- Maximum iterations - the maximum number of iterations allowed to find all distance values (default = 7).
- Mode: Accurate and Fast:

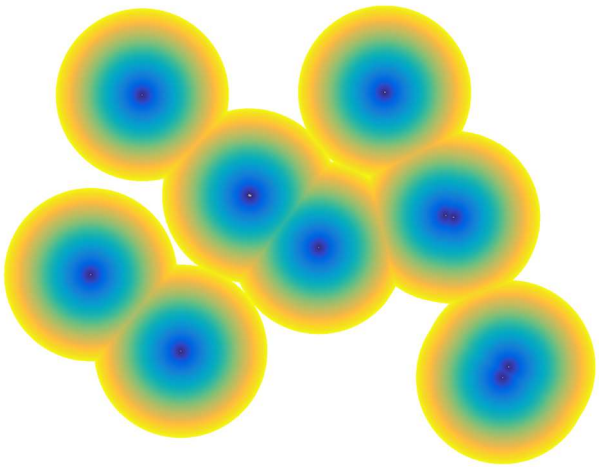
- Fast: this mode uses an approximation method, making the detection faster, but has the tendency to generate hexagonal visual patterns;





- Accurate: this mode uses the Euclidian distance, making the detection slower, but more accurate.





Effect:

Generates a distance map relative to the areas of interest.

Example:

- Input:

Figure below shows detected epithelial area in a fluorescence colorectal cancer sample. The events representing the epithelial area, are the reference for this distance map example; the distances are computed in relation to these events.

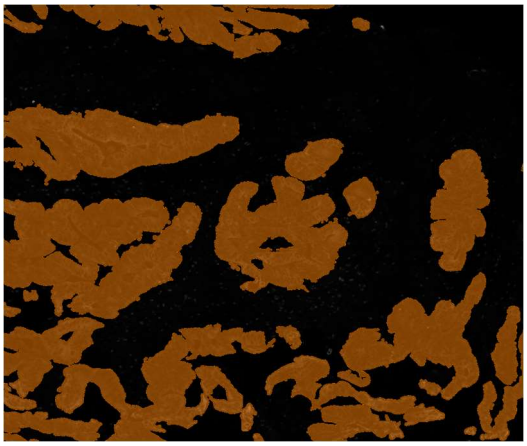


Figure 556 – Detected epithelial area

- Engine settings:

Figure below shows the engine settings used for this example.

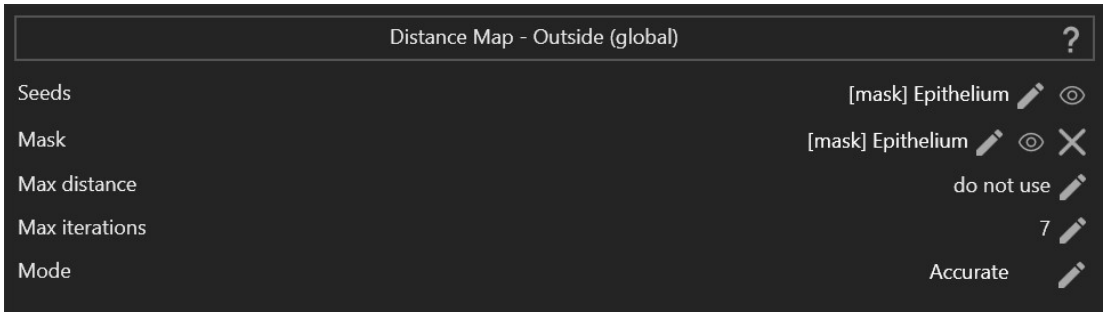


Figure 557 – Engine settings

Output:

Figure below shows the generated distance map (shades of blue) overlaid with the detected epithelial areas (highlighted in orange). The higher the distance from the epithelial area, the lighter the shade of blue.

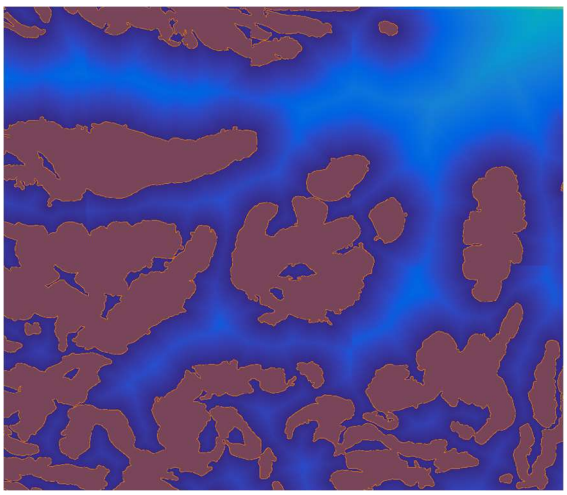


Figure 558 – Result of Distance Map (global) engine

14. Distance Map Outside (local)

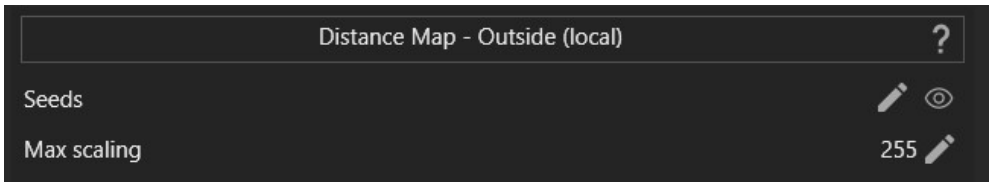


Figure 559 – Distance Map Outside (local)

Where it can be found:

Layer's "Pre-Processing" ► Add ► Operations (advanced) ► Distance Map Outside (Local)

Description:

Distance Map (local) is an engine available in the Pre-Processing section of the layers editor.

This engine computes the distance value for all background pixels (pixels value = 0 / black) relative to the closest foreground pixel (pixels value = 255 / white).

The distance map is stored using an 8-bit datatype and the visualization is realized with the help of a grayscale image, which means a lower accuracy than in the case of the global variant of the engine.

The engine is not suited to estimate distances for areas of interest being one Field of View (FOV) apart.

Figure 560 – Overlay (input mask image + grid highlighting FOVs)

Parameters:

- Seeds - input grayscale image representing the areas of interest (from which the distances will be computed)
- Max scaling - represents a scaling factor used to approximate distances higher than 255 (maximum value for 8-bit datatype), based on the scale:

$$Distance = Pixel\ value \times \frac{Max\ scaling}{255}$$

For example, the default value (Max scaling = 255) means a relation of 1:1. If the scaling factor value is modified (Max scaling = 512), the new relation will be 1:2. Therefore, a pixel value of 10 (in the distance map image) represents a distance of 20.

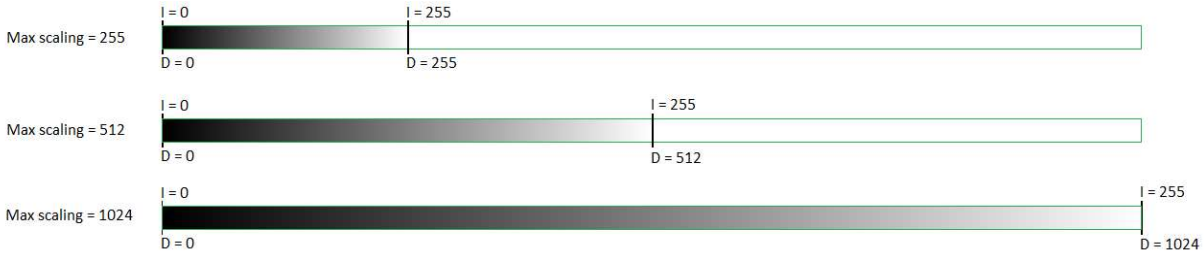


Figure 561 – Max scaling (I = image intensity, D = distance value)

Effect: Generates a distance map relative to the areas of interest.

Example:

- Input:

Figure below shows a binary mask representing the presence of a nuclear marker in an IHC colon sample. The white area represents the reference for the distance map generated in this example. The distances are computed in relation to these areas.

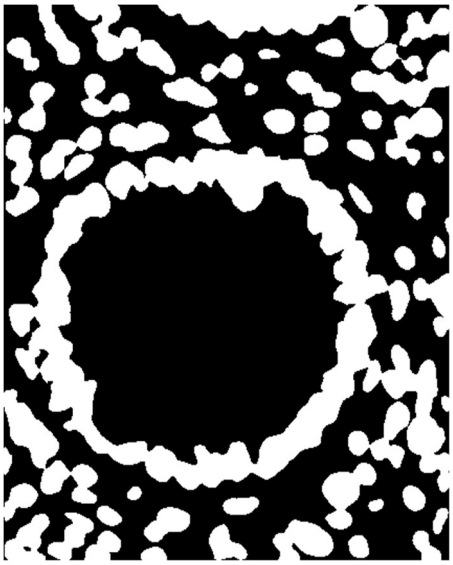


Figure 562 – Nucleus mask

- Engine settings:

Figure below shows the engine settings used for this example.

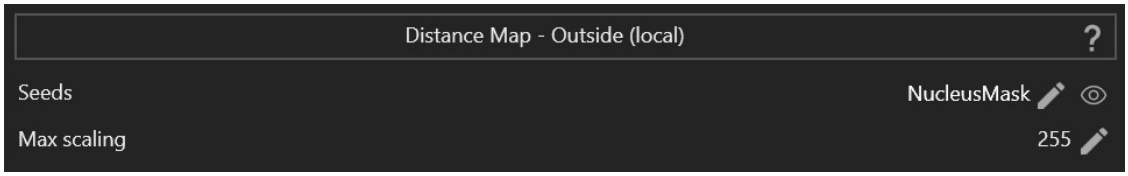


Figure 563 – Engine settings

Output:

Figure below shows the generated distance map, where brighter pixels correspond to higher distances.

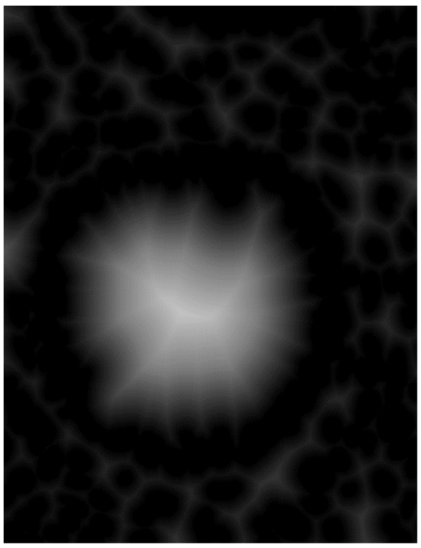


Figure 564 – Distance transform

15. Distance Map – inside (global)

This engine generates a distance transformation image on a Stitched/Global object. It can even handle large macrostructures that span on multiple FOVs.

Figure 565 – Distance map – inside (global)

Where it can be found:

Layer's "Pre-Processing" ► Add ► Operations (advanced) ► Distance map – inside (global)

Description:

Distance Map - inside (global) is an engine available in the Pre-Processing section of the layers editor.

This engine computes the distance value for all foreground pixel (pixels value = 255 / white) relative to the closest background pixels (pixels value = 0 / black). It computes the inner distances relative to events contours.

The distance map is stored using 32-bit datatype for better accuracy. The visualization is realized with the help of a color image using Parula colormap, where colors vary from blue (minimum value) to red (maximum value).

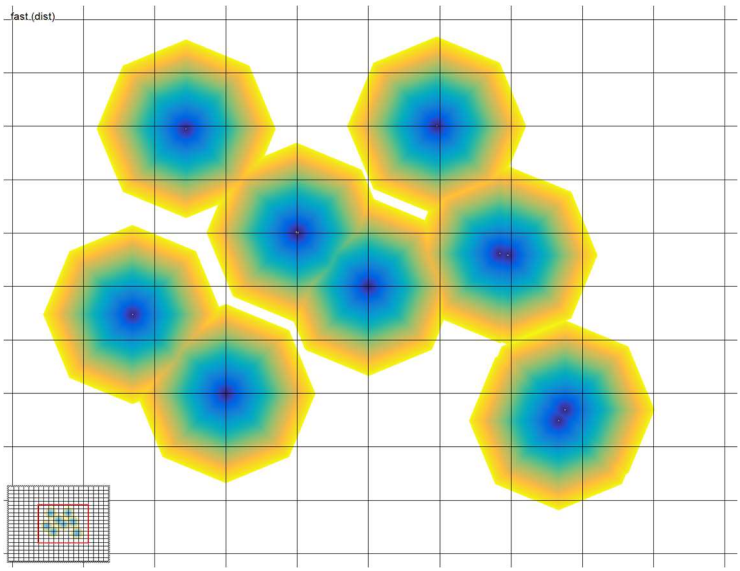
Figure 566 – Parula colormap

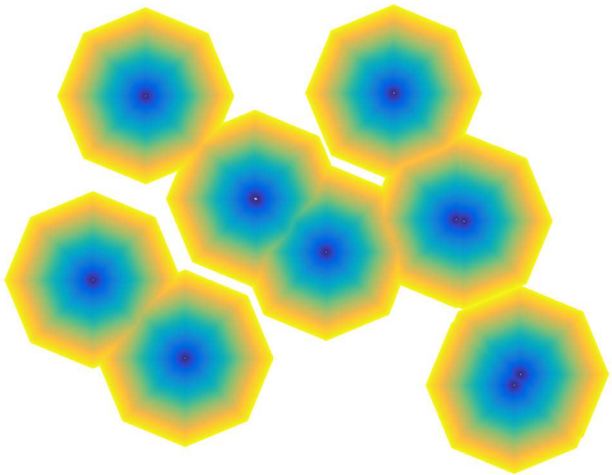
Note	Because a color map is used to display the distance map, Pixel Inspector is needed to find the correct distance value of a pixel.

Parameters:

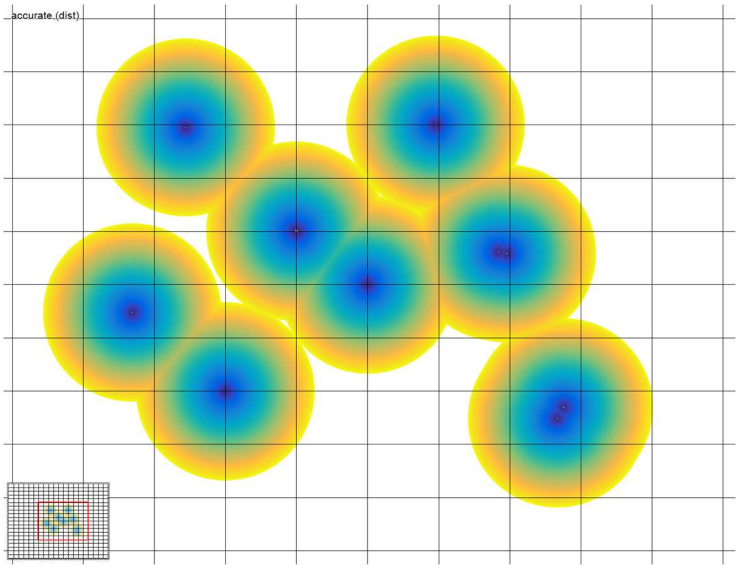
- Events - the input coded image representing the events of interest (from which the distances will be computed)
- Mask - the input mask image. The distance values are computed using only paths inside the indicated areas (white)
- Max distance - the upper limit for the distances
- Max iterations - the maximum number of iterations allowed to find all distance values (default = 7)
- Mode: Accurate and Fast:

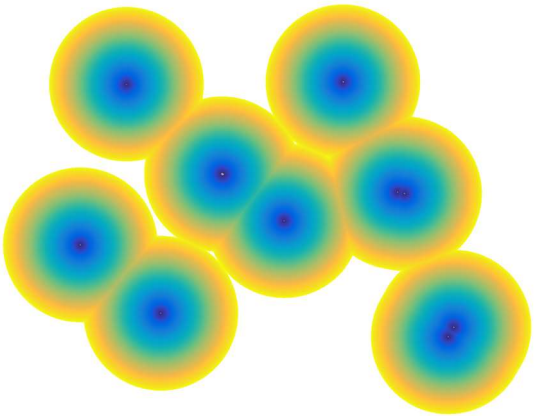
- Fast: this mode uses an approximation method, making the detection faster, but has the tendency to generate hexagonal visual patterns;





- Accurate: this mode uses the Euclidian distance, making the detection slower, but more accurate.





Effect:

Generates a distance map relative to the areas of interest.

Example:

- Input:

Figure below shows detected glomeruli in a fluorescence kidney sample. The events representing the glomeruli, are the reference for this distance map example; the distances are computed in relation to these events contours.

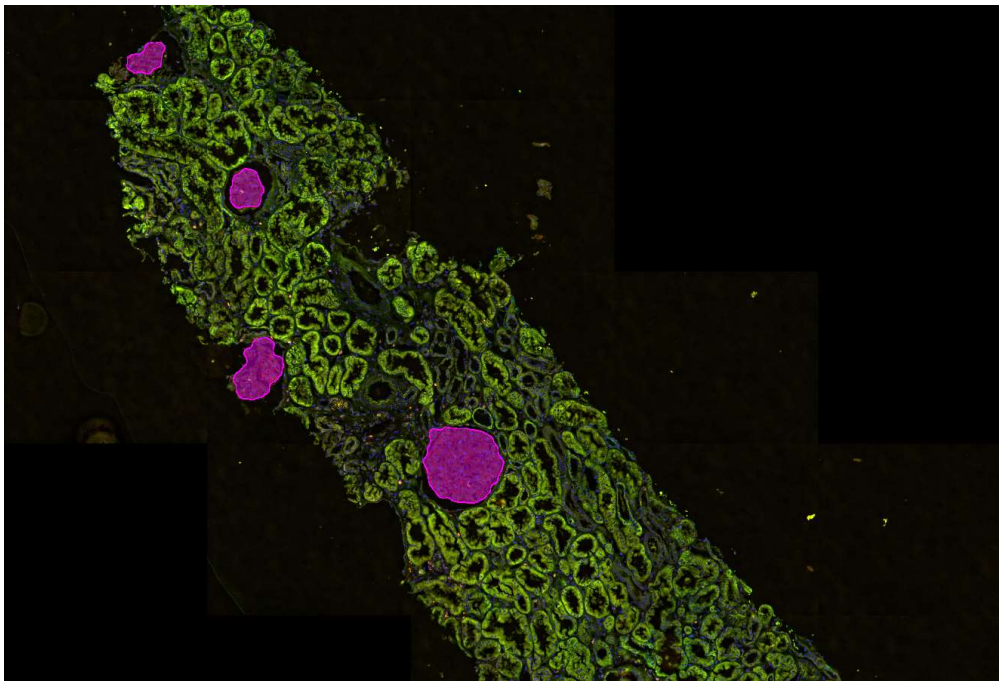


Figure 567 – Input image

- Engine settings:

Figure below shows the engine settings used for this example.

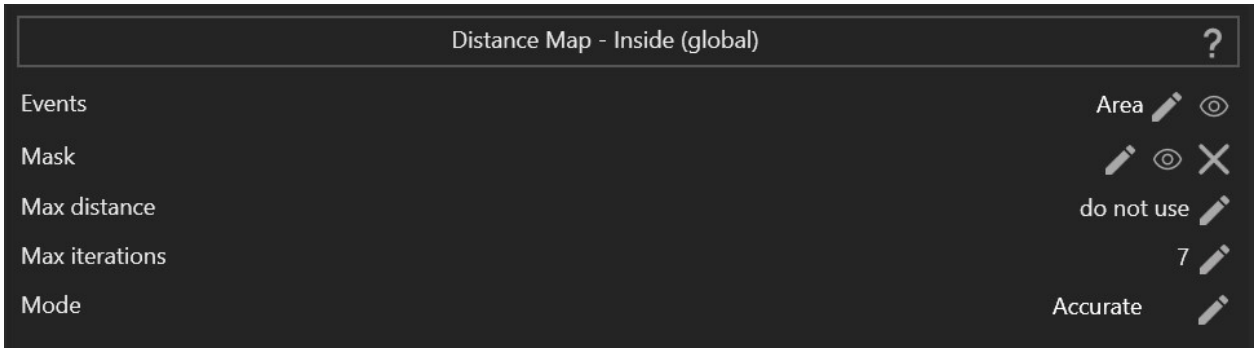


Figure 568 – Engine settings

- Output:

Figure below shows the generated inner distance map overlaid with the detected glomeruli contours (highlighted in magenta). The higher the distance from the glomeruli contours, the hotter the representation color.

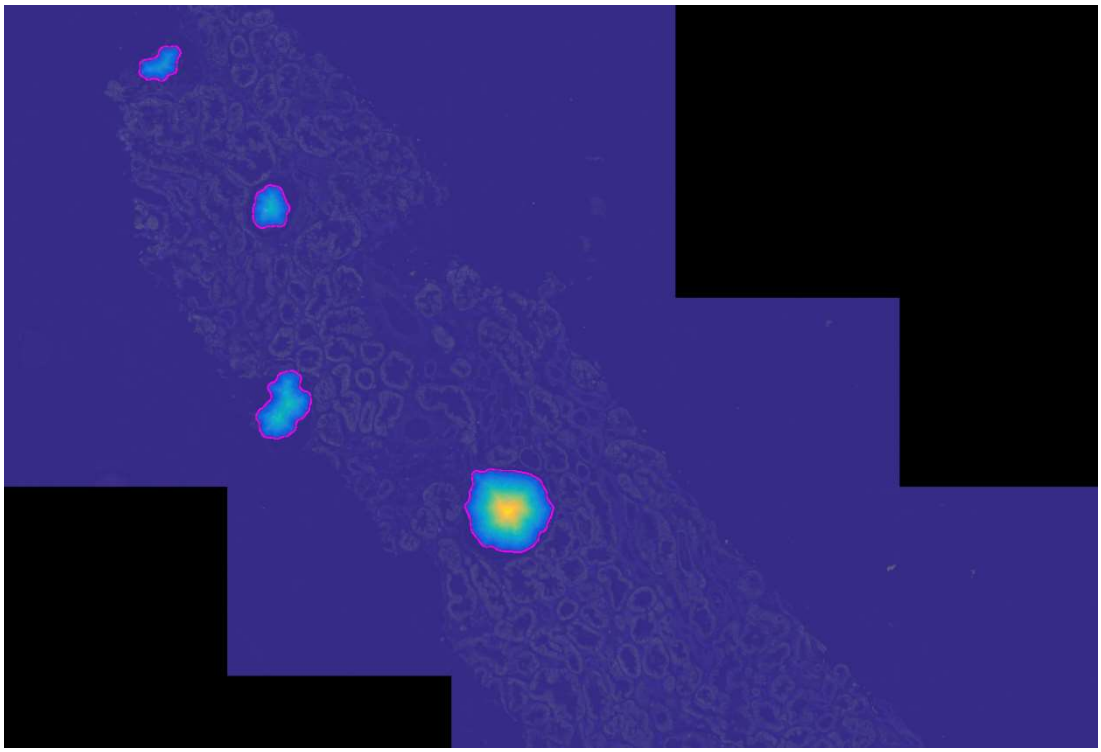


Figure 569 – Output image

16. Dots (FISH/CISH)

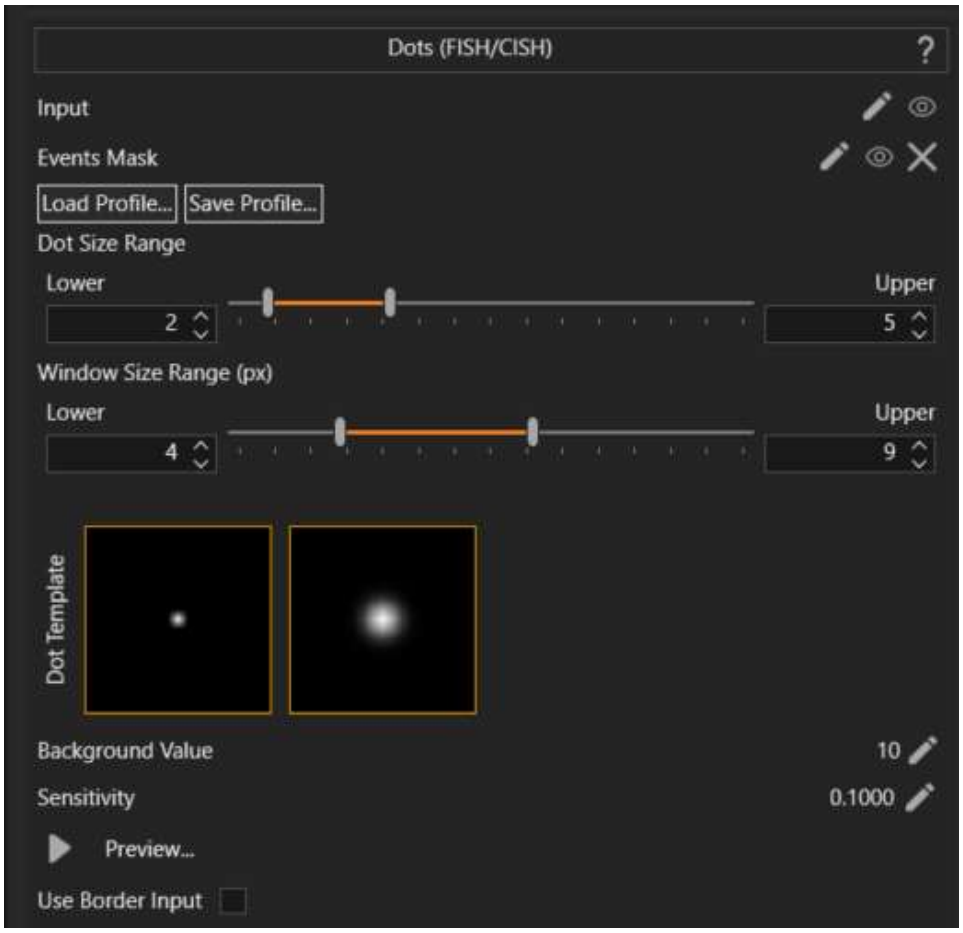


Figure 570 – DOTS

Where it can be found:

Layer's "Pre-Processing" ► Add ► Detection ► Dots (FISH/CISH)

Layer's "Detection" ► Add ► Detection ► Dots (FISH/CISH)

Description:

Dots (FISH/CISH) detection is an engine available in the Pre-Processing and Detection sections of the layers editor.

This engine performs the detection of dot-like structures on a grayscale image, generating set of events represented by a coded image. In the coded image, all pixels belonging to an event have the same unique ID, thus preserving the morphology of the event even when two or more events are touching.

Parameters:

- Input - the input grayscale image
- Events Mask - a coded image representing the set of associated dot events (e.g. nuclei). The detection is limited only inside these events
- Dot Size Range - the range of dots sizes

Figure 571 – DOTS: setting dot size range

- Lower - the lower limit for the dot size range
- Upper - the upper limit for the dot size range
- Windows Size Range (px) - range of kernel sizes

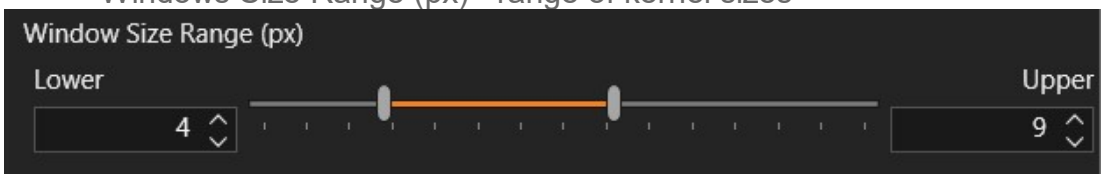


Figure 572 – DOTS: setting window size range

- Lower - the lower limit for the kernel size range
- Upper - the upper limit for the kernel size range
- Background Value - the value to be subtracted from the source image to attenuate the high background (if present).
- Sensitivity - the threshold for of generated dots patterns the matching scores.

A series of dot patterns are generated using various dot sizes and various kernel sizes, defined by the two ranges Dot Size Range and Windows Size Range. Each pattern is used to search the source image for dots, generating matching scores for each location (pixel). The scores generated by all patterns are also evaluated using the Sensitivity parameter (a lower value will return a higher number of detected dots).

All detection settings can be saved as a new profile using the Save Profile button and can be used on different projects by using the Load Profile button. By default, several predefined profiles are available for loading.

Effect:

Performs dots detection.

Example:

- Input:

Figure below presents a small Region of Interest (ROI) from a FISH sample (a). The ROI is also presented in higher magnification (b) for a better visualization of the FISH dots and in a 3D representation (c), where:

x and y axis - the pixel spatial positions

z axis - the grayscale intensity of pixels

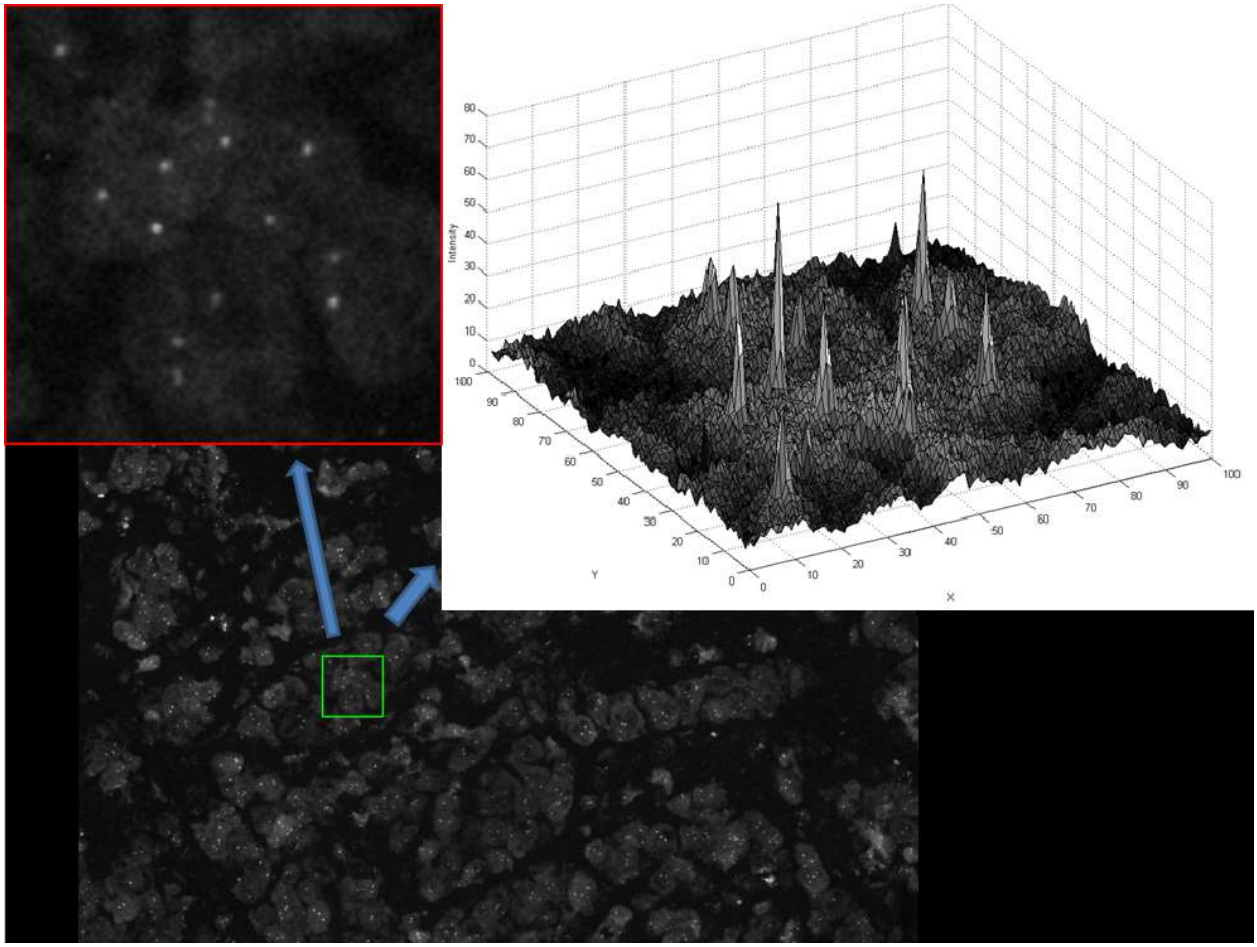


Figure 573 – (a) FISH sample; (b) Green ROI with higher magnification; (c) Green ROI in 3D view

- Engine settings:

Based on the user defined ranges for dot size (Dot Size Range = [Lower … Upper]) and kernel size (Window Size Range = [Lower … Upper])), a series of dot patterns are generated and used to search FISH / CISH dots in the source image.

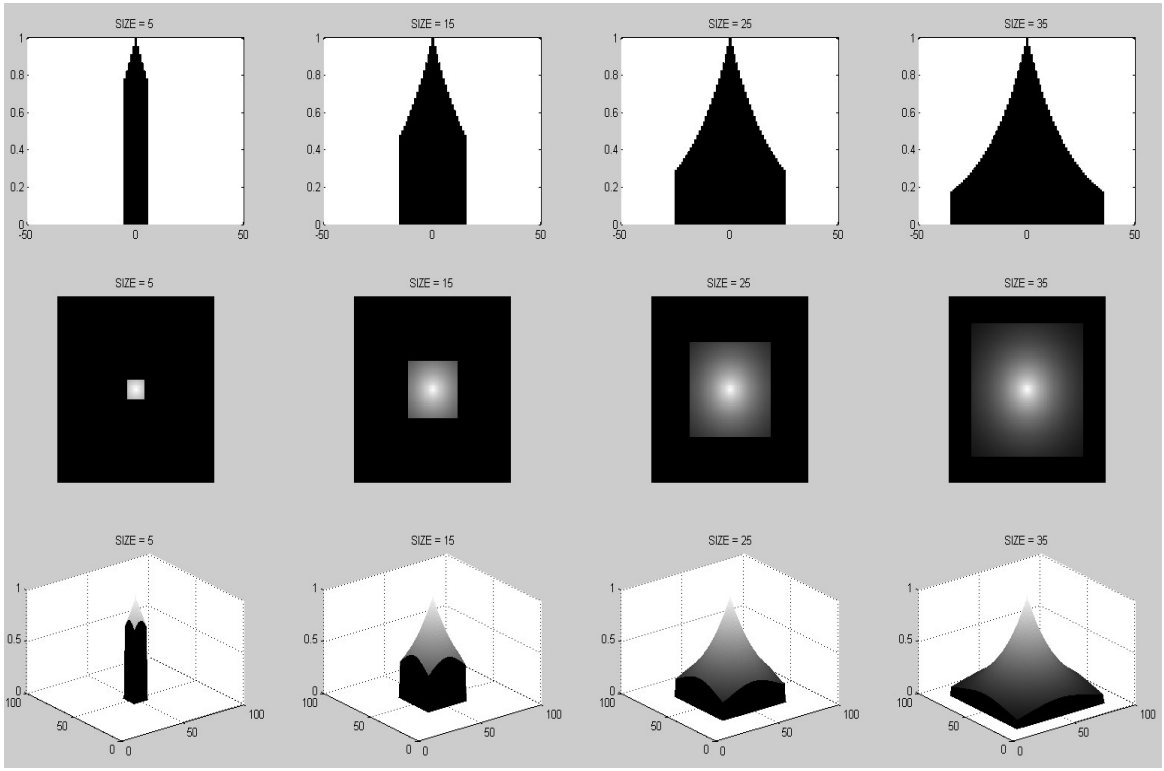


Figure 574 – Dots Patterns Examples (Window Size = variable, Dot Size = 7 (constant))

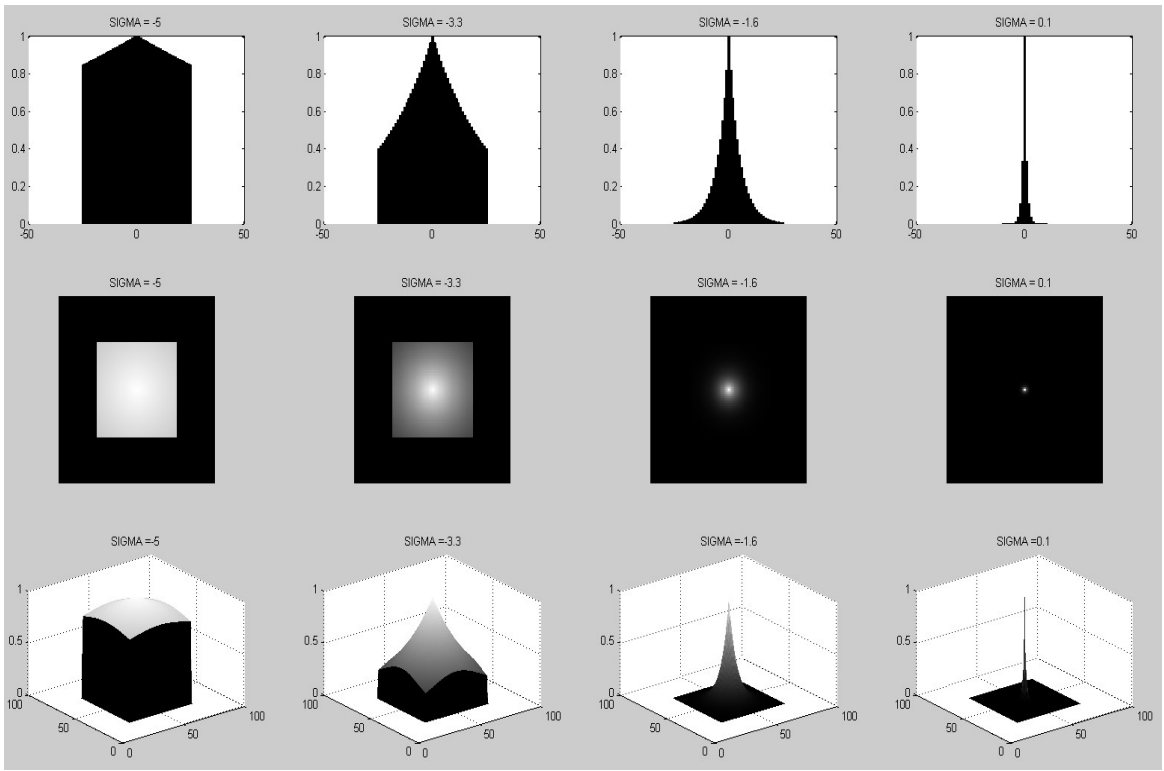


Figure 575 – Dots Patterns Examples (Window Size = 25 (constant), Dot Size = variable)

Output:

Figure below shows the detected evets overlaid on the original image and in the 3D representation.

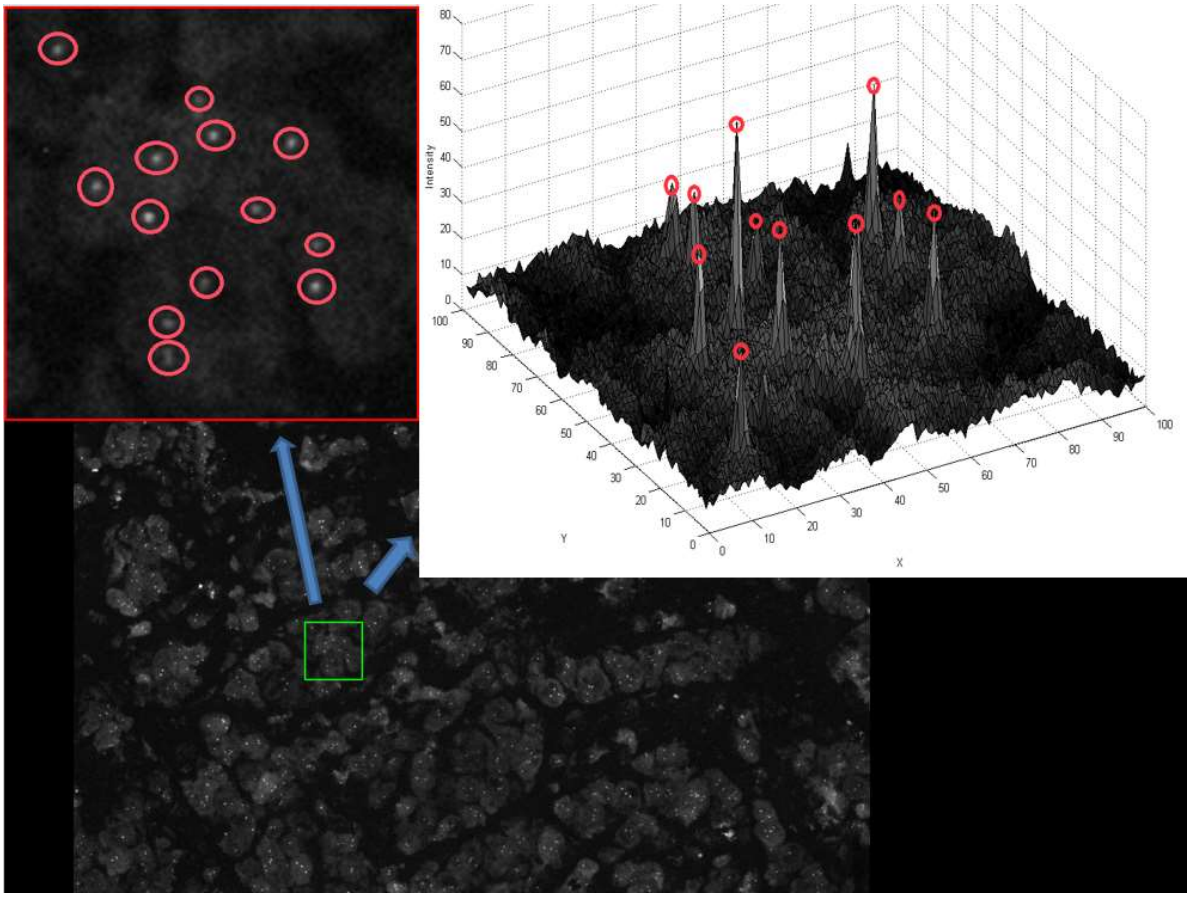


Figure 576 –Identified DOTS

17. Dot Measurements

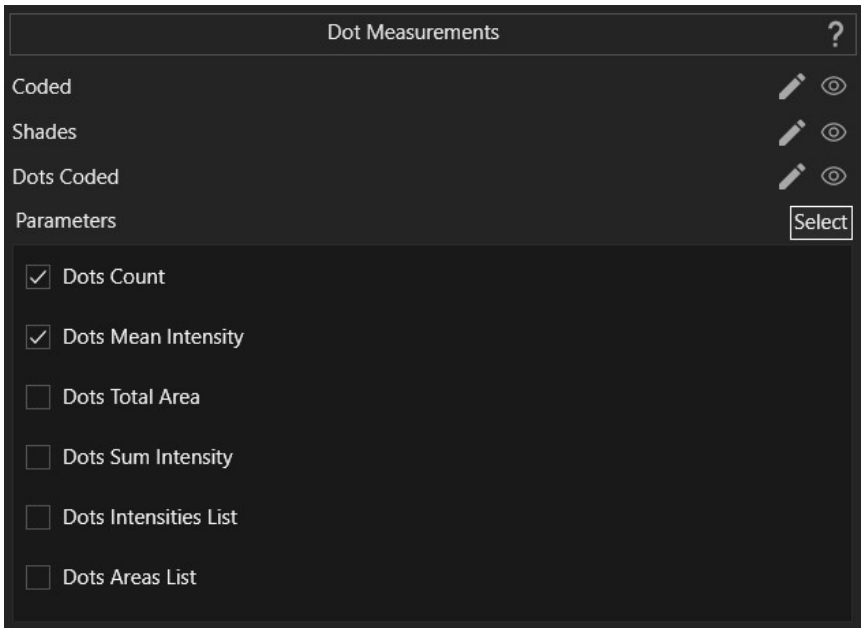


Figure 577 – Dot Measurements

Where it can be found:

Layer's "Measurements" ► Add ► Measurements (advanced) ► Dot Measurements

Description:

Dots Measurements is an engine available in the Measurements section of the layers editor.

This engine computes dot-based measurements for a set of events, which may be considered master events (e.g. nuclei). For each master event, dot-based measurements are computed using information from all dot events detected inside the master event.

Parameters:

- Coded - input coded image representing the set of master events
- Shades - input grayscale image representing dot marker expression
- Dots Coded - input coded image representing the set of dot events

Different dot-based measurements can be computed using information from the three input images and a selection of the following list options:

- Dots Count - the number of dot events inside each master event

- Dots Mean Intensity - the average gray level expressed by the pixels of all dots inside a master event
- Dots Total Area - the total area of all dots inside a master event
- Dots Sum Intensity - the sum of all gray level expressed by the pixels of the dots inside a master event
- Dots Intensities List - a list of the average intensities of all dots inside a main event
- Dots Areas List - a list of areas of all dots inside a master event.

Effect:

Computes dot-based measurements for a set of master events.

Example:

- Input:

Figure below shows 2 different sets of events detected on an IHC sample:

- nuclei (highlighted in blue) – represents the master events. Measurements are associated to each nucleus
- dots (highlighted in green) – measured events (e.g. mean intensity of the dots inside nucleus X)

The separated marker image and the 2 sets of events represent the engine input.

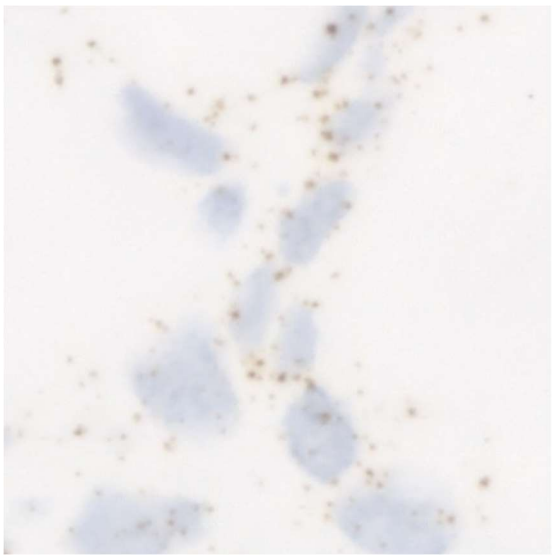


Figure 578 – IHC sample

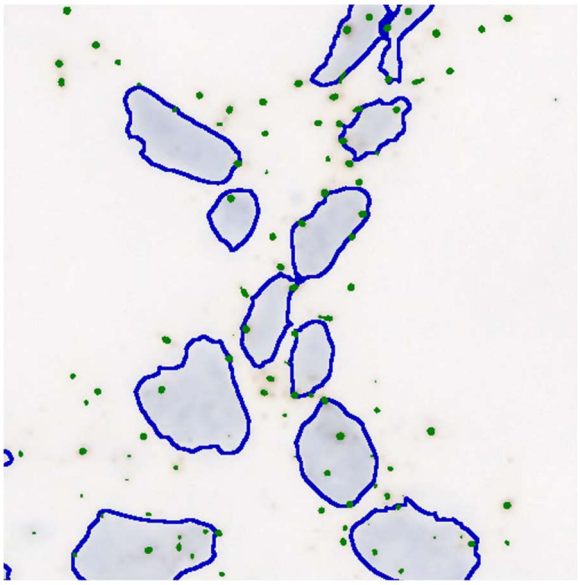


Figure 579 – Detected nuclei (blue) and dots (green)

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a list of measurements for each master event (nucleus).

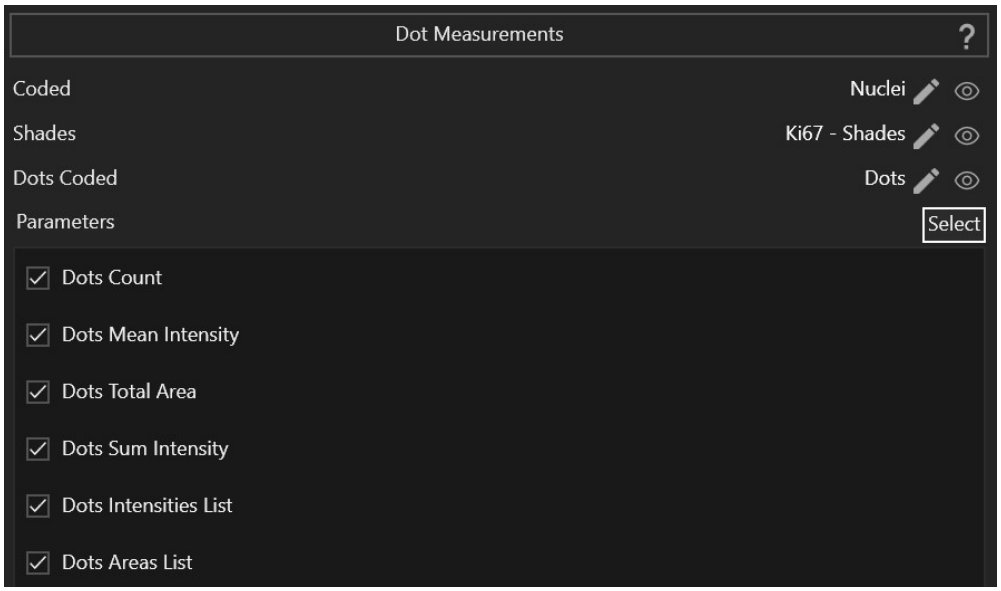


Figure 580 – Engine settings

Output:

First figure shows the Dot Measurements for a selected nucleus, accessible from Event Data (right click on an event).

Second figure shows the Dot Measurements for all events, accessible from Raw Data.

Measurement	Value
Object Label	1528
Layer	New Layer
Set 1 - Nuclei-Ki67-Dots Count	3
Set 1 - Nuclei-Ki67-Dots Mean Intensity	77.076920
Set 1 - Nuclei-Ki67-Dots Total Area	13
Set 1 - Nuclei-Ki67-Dots Sum Intensity	1002
Set 1 - Nuclei-Ki67-Dots Intensities List	125.50;123;36.29
Set 1 - Nuclei-Ki67-Dots Areas List	4;2;7

Figure 581 – Result of the Dot Measurements engine shown for a particular event

Event Label	Set 1 - Nuclei-Ki6···	Set 1 - Nuclei-Ki6···	Set 1 - Nuclei-Ki6···	Set 1 - Nuclei-Ki6···
1	0	0.000000	0.000000	0.000000
2	0	0.000000	0.000000	0.000000
3	2	8.545455	11.000000	94.000000
4	0	0.000000	0.000000	0.000000
5	2	27.750000	4.000000	111.000000
6	0	0.000000	0.000000	0.000000
7	0	0.000000	0.000000	0.000000
8	1	8.000000	7.000000	56.000000
9	1	7.000000	5.000000	35.000000
10	0	0.000000	0.000000	0.000000
11	2	8.000000	5.000000	40.000000
12	0	0.000000	0.000000	0.000000
13	0	0.000000	0.000000	0.000000
14	0	0.000000	0.000000	0.000000
15	0	0.000000	0.000000	0.000000
16	0	0.000000	0.000000	0.000000

Figure 582 – Result of Dot Measurements engine

18. Events center

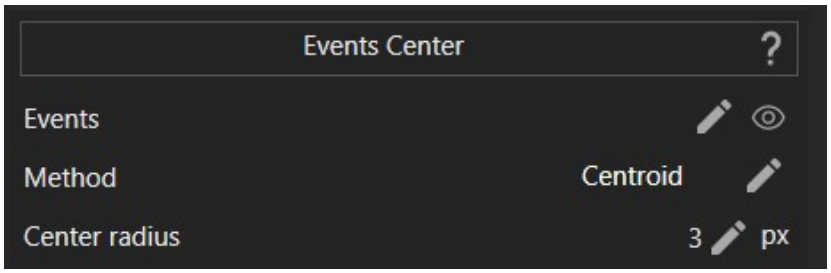


Figure 583 – Events center

Where it can be found:

Layer's "Measurements Mask" ► Add ► Operations (basic) ► Events center

Description:

Events center is an engine available in the Measurements Mask section of the layers editor.

This operation detects the event's centroids.

Parameters:

- Events - inputs coded image representing the set of events
- Method - this engine identifies the events centers using 2 methods:
  - Centroid – finds the geometric centroid based on event's shape
  - Distance – finds the centroid as the inner point most distant from event's edges
- Grow Radius – defines the centroids radius. By default, the generated centroids are represented by 1 pixel and are inflated with the specified radius, especially for better visualization.

Effect:

Detects events centroids.

Example:

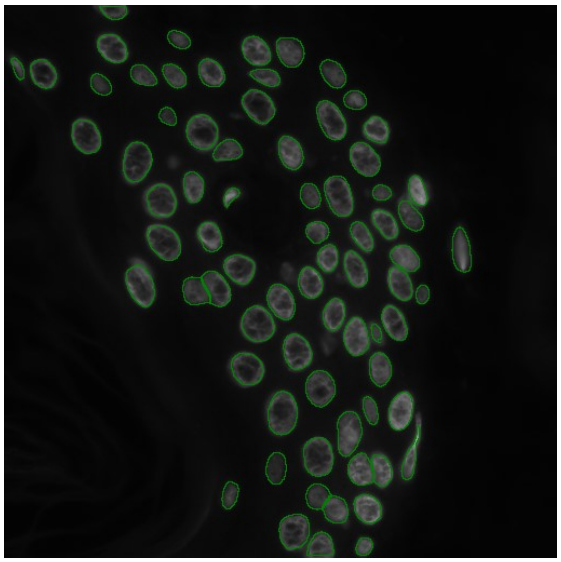


Figure 584 – Input image

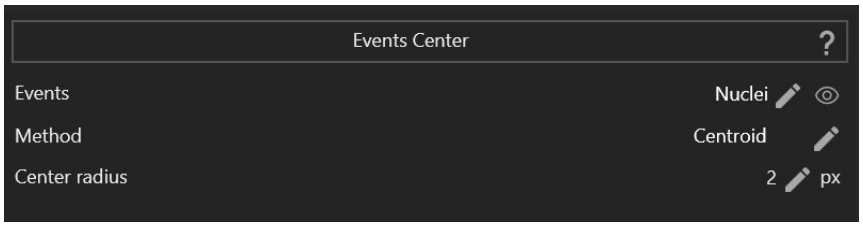


Figure 585 – Engine Settings

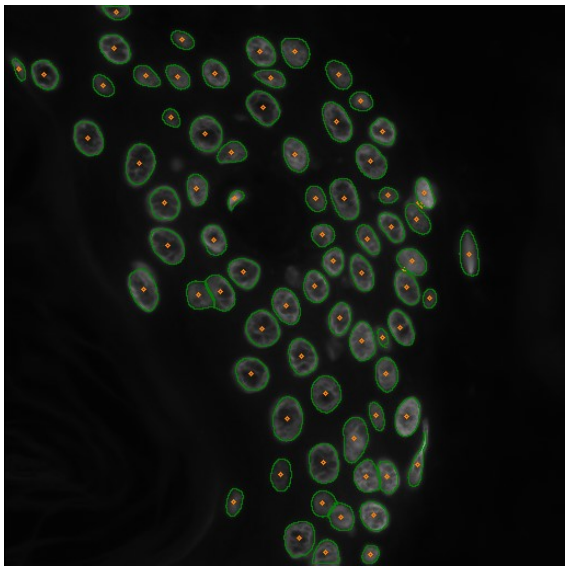


Figure 586 – Output Image

19. Events Count Inside Master Events

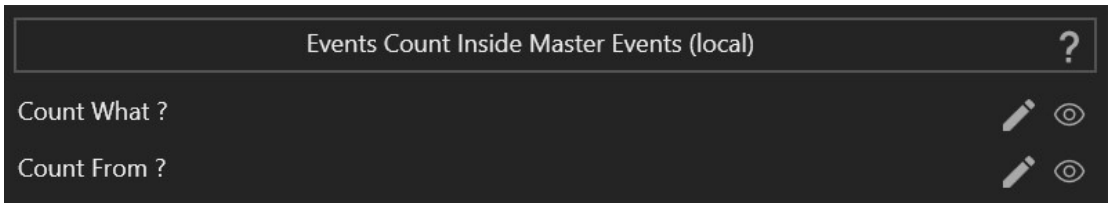


Figure 587 – Events Count Inside Master Events (local)

Where it can be found:

Layer's "Measurements" ► Add ► Measurements (basic) ► Events Count Inside Master Events (global)

Layer's "Measurements" ► Add ► Measurements (basic) ► Events Count Inside Master Events (local)

Description:

Events Count Inside Master Events is an engine available in the Measurements section of the layers editor.

This engine counts how many events are detected inside in each master event.

Two versions of the engine are available:

- Local - performs events counting with no information about events stitching for counted events (counts local events inside global master events).
- Global - performs events counting with information about events stitching for counted events (counts global events inside global master events).

A Local Event is considered an event detected in a Field of View (FOV). Multiple Local Events located at the border (touching border) of a neighboring FOV, representing a structure positioned at the intersection of two or more FOVs, are combined (stitched) into a Global Event for a better representation of the real information present in the sample or ROI.

Parameters:

- Count what - inputs a coded image representing the counted set of events
- Count from - inputs a coded image representing the set of master events, for which the measurements are computed

Effect:

Counts how many events are present inside each master event.

Example:

- Input:

The input consists of 2 set of events:

- the structures - highlighted in green
- brown nuclei (ki67+) – highlighted in yellow

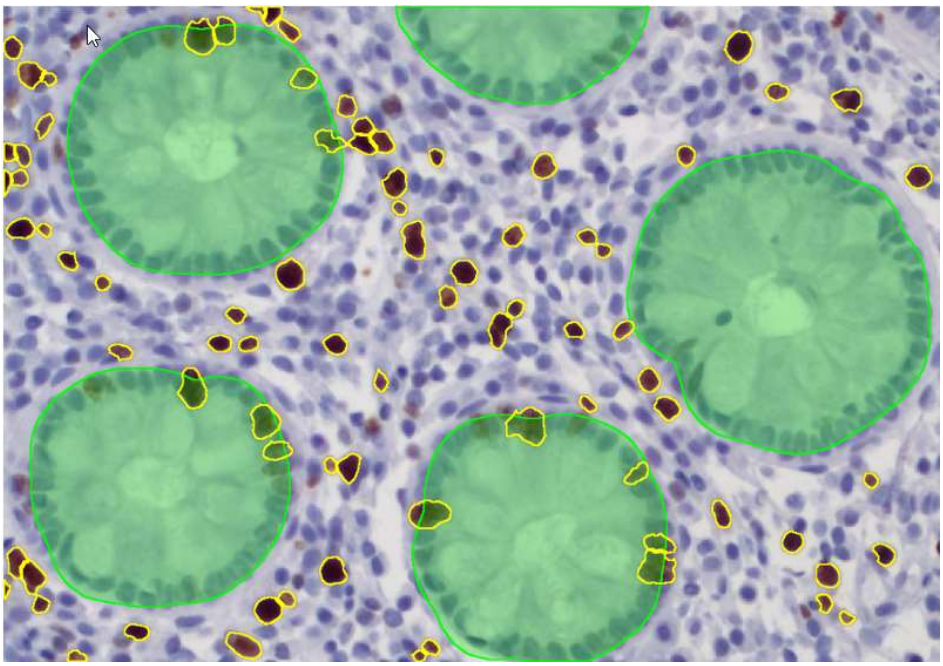


Figure 588 – Detected events on IHC colon sample

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result a set of measurements associated to the master events.

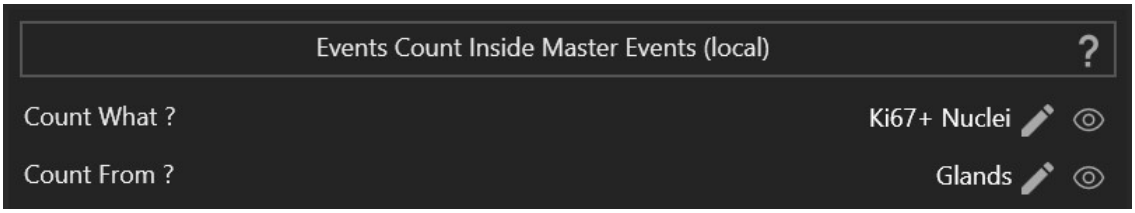


Figure 589 – Engine settings

Output:

- First figure shows the events count measurement for a selected structure, accessible from Event Data (right click on an event).
- Second figure shows the events count measurement for all events, accessible from Raw Data.

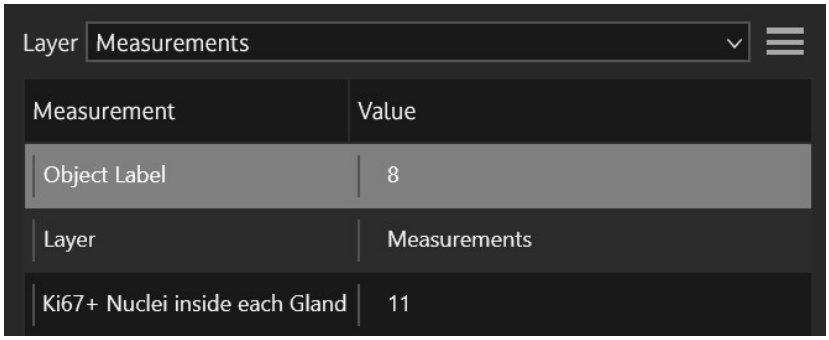


Figure 590 – Result of the Events Count Inside Master Events engine shown for a particular event

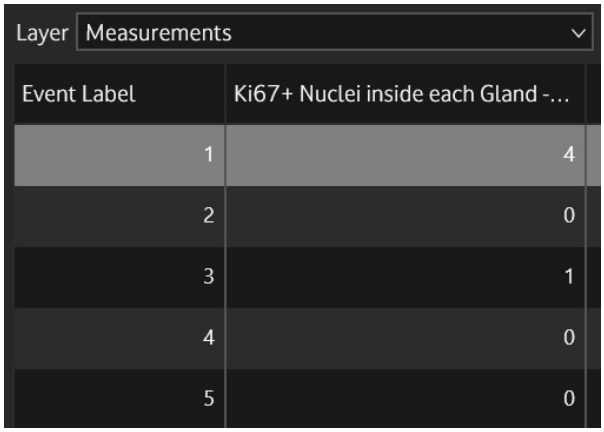


Figure 591 – Result of the Events Count Inside Master Events engine

20. Existing Coded

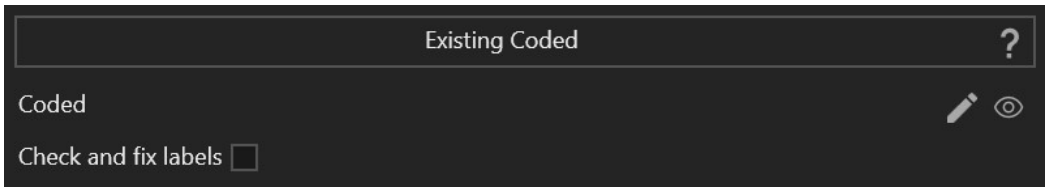


Figure 592 – Existing Coded

Where it can be found:

Layer's "Detection" ► Add ► Detection ► Existing Coded

Description:

Existing Coded is an engine available in the Detection section of the layers editor.

This engine sets an existing set of events (e.g. imported from another layer) as events detected in the current layer. This allows measurements to be computed for a set of events computed in a different layer, without re-detecting the events. Measurements can be computed only for events detected in the layer where the measurements engine is used.

Import events (coded) from another layer using gate(s) or applying different engines on the imported set of events may cause duplicate labels (IDs) or missing labels. To avoid possible errors, the engine offers the possibility of fixing the labels list by enabling the Check and fix labels option.

Parameters:

- Coded - inputs coded image representing the set of events to be set as detected events
- Check and fix labels - enables / disables the correction of labels list (duplicates, missing labels)

Effect:

Sets a set of events imported from another layer as events detected in the current layer to avoid re-running the detection.

Example:

Figure below shows an example where a set of nuclei is imported from another layer (can be a specific set of nuclei, e.g. CD3+ nuclei) and is passed to the detection section of the current layer for the computation of measurements.

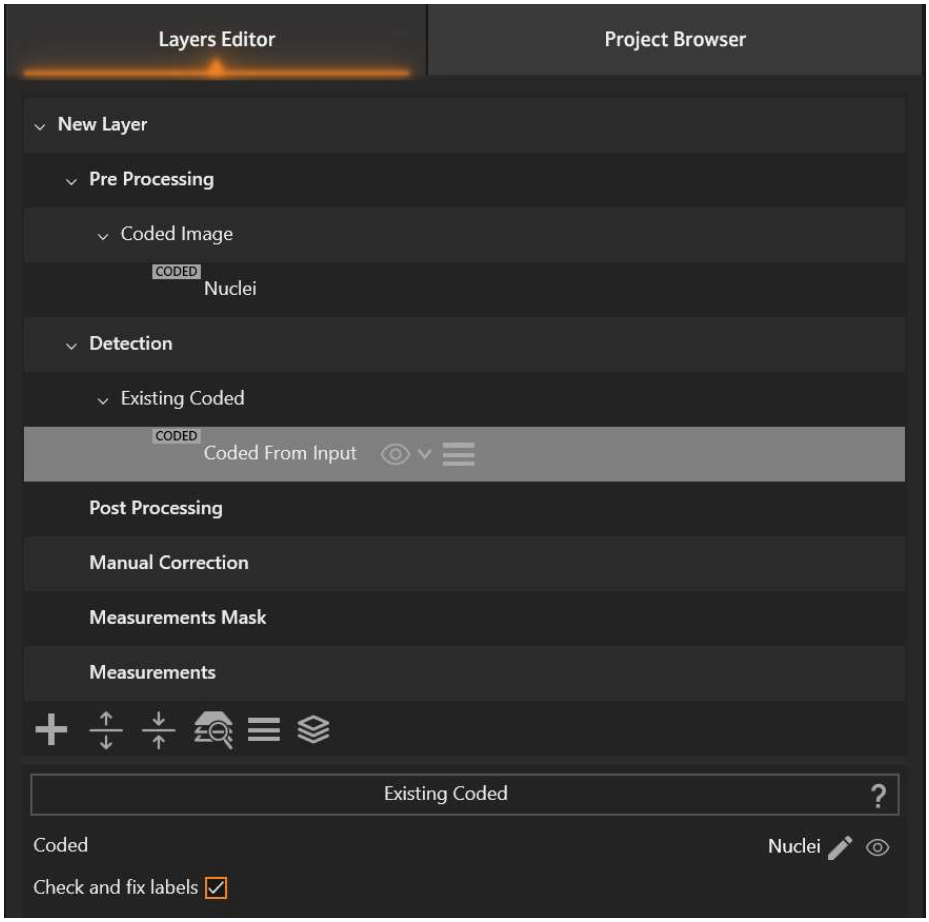


Figure 593 – Coded From Input engine example

21. Grayscale Image

Figure 594 – Grayscale Image

Where it can be found:

Layer's "Pre-Processing" ► Add ► Import ► Grayscale image

Description:

Grayscale image is an engine available in the Pre-Processing section of the layers editor.

Except original channels (FL) / original color image (BF), all the virtual channels (grayscale, color) are "visible" only in the layer where they are generated. The (import) Grayscale image engine allows importing a grayscale image from a different layer and making it available in the current layer. This avoids setting up and running again the engine or succession of engines needed to generate the same grayscale image.

Parameters:

- Layer - the layer where the grayscale image is available
- Coded - the grayscale image to be imported

Effect:

Makes a virtual channel from another selected layer available in the current layer.

22. Grow (global)

Figure 595 – Grow

Where it can be found:

Layer's "Measurements Masks" ► Add ► Operations (basic) ► Grow (global)

Description:

Grow (global) is an engine available in the Measurements Masks section of the layers editor.

This engine generates new measurements masks for a set of detected events (coded image), by using one of the three modes available:

- Grow - the events bodies are inflated with a user defined size
- Shrink - the events bodies are deflated with a user defined size
- Shrink (retain last pixel) – the events bodies are deflated with a user defined size. It will retain the last pixel of each event body if the deflations size is higher than the event size.
- Ring - can be defined by the formula:

$$Ring(event) = Grow(event) - Shrink\,(event)$$

The engine uses the events contours as starting points for Grow / Shrink / Ring.

Parameters:

- Mode - specifies the mode used to generate the new events measurements masks (default = Grow)

Grow:

Figure 596 – Grow type: Grow

- Grow radius - the size of the growing operation, starting from events contours
- Mask - inputs a mask image indicating growing allowed areas (white)

Shrink:

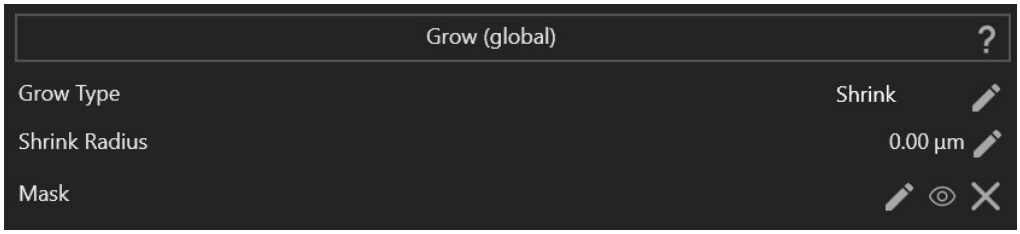


Figure 597 – Grow type: Shrink

- Shrink radius - the size of the shrinking operation, starting from the events contours
- Mask - inputs a mask image indicating shrinking allowed areas (white)

Shrink (retain last pixel):

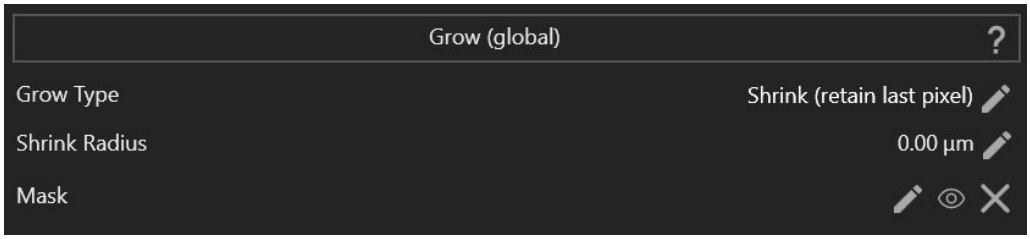


Figure 598 – Grow type: Shrink

- Shrink radius - the size of the shrinking operation, starting from the events contours
- Mask - inputs a mask image indicating shrinking allowed areas (white)

Grow:

Figure 599 – Grow type: Ring

- Skip distance - shifts the starting point (event's contour) outward, by the specified distance value.
- Interior radius - the size of inward growing, starting from the event's contour (can be shifted outward if Skip distance > 0). Acts like a Shrink radius
- Exterior radius - the size of outward growing, starting from the event's contour (can be shifted outward if Skip distance > 0). Acts like a Grow radius
- Include seeds - the original event's body is added with the resulting ring. Acts like an Or operation of the two masks
- Mask - inputs a mask image indicating growing allowed areas (white)

Effect:

Generates a new measurements mask by performing events growing.

Example:

- Input:

Figure below shows the detected epithelial area in a fluorescence colorectal cancer sample. For this example, 3 cases are considered:

- case 1: growing with 10 µm
- case 2: shrinking with 10 µm
- case 3: ring shapes

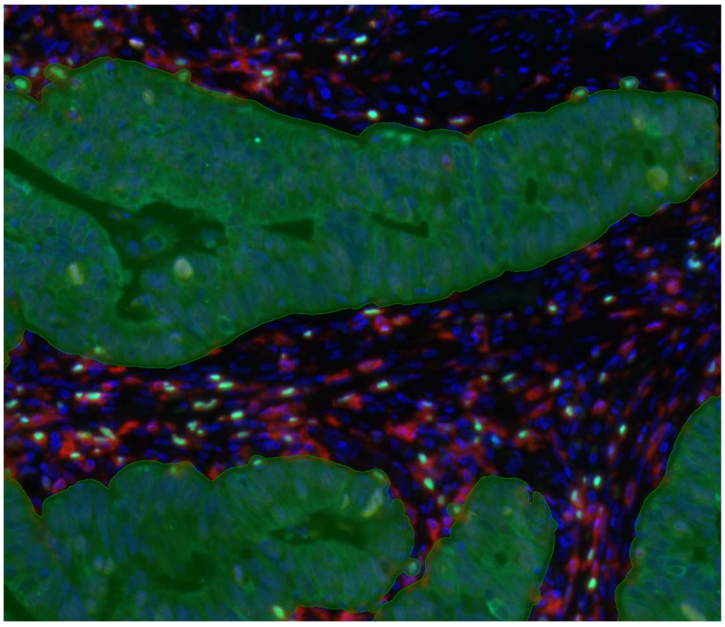


Figure 600 – Detected epithelial area

- Engine settings:

Figures below show the engine settings for the 3 cases considered.

The engine's result is a coded image representing the grown / shrank / ring-shaped events.

Figure 601 – Engine settings (case 1)

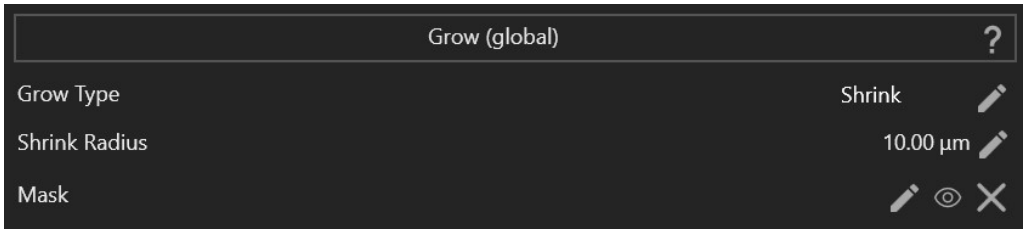


Figure 602 – Engine settings (case 2)

Figure 603 – Engine settings (case 3)

Output:

Figures below show the result of the growing process for the 3 cases considered.

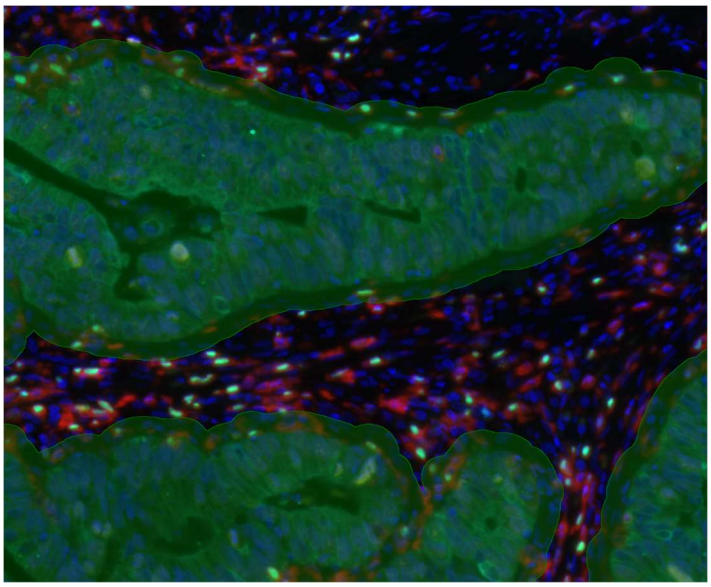


Figure 604 – Result of Grow (global) engine (case 1)

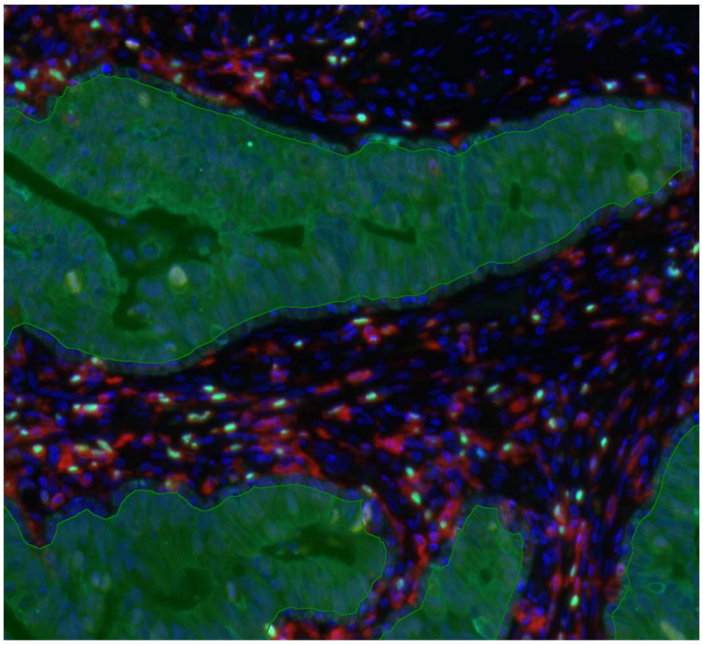


Figure 605 – Result of Grow (global) engine (case 2)

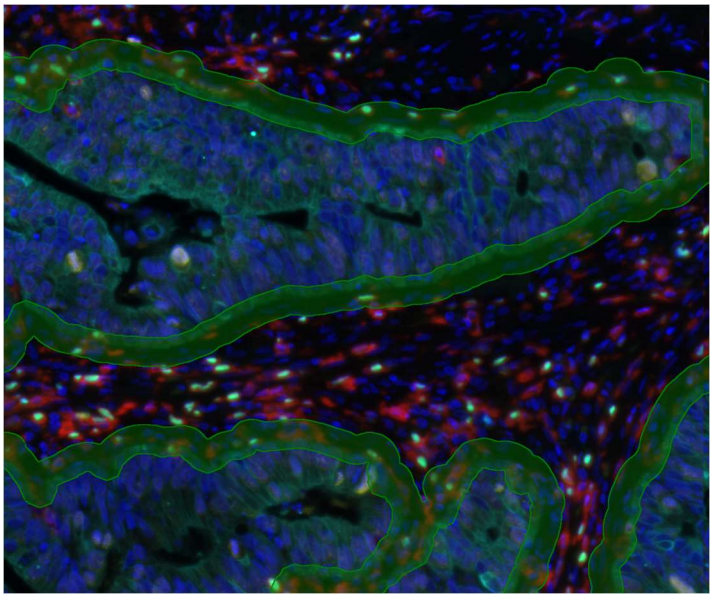


Figure 606 – Result of Grow (global) engine (case 3)

23. Nuclei (Deep Learning)

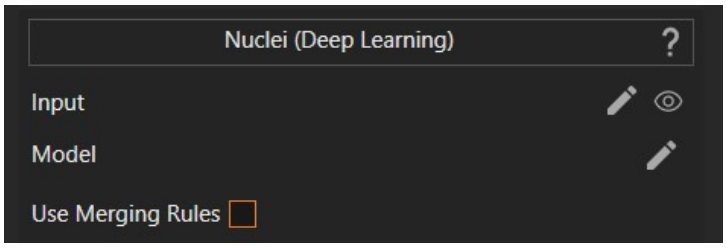


Figure 607 – Nuclei – Deep Learning engine

Where it can be found:

Layer's "Detection" ► Add ► Detection ► Nuclei (Deep Learning)

Description:

Nuclei (Deep Learning) segmentation is an engine available in the Detection section of the layers editor.

This engine uses a trained deep learning model to perform nuclear detection and segmentation on a grayscale image, generating set of events represented by a coded image. In the coded image, all pixels belonging to an event have the same unique ID. This preserves the morphology of the event even when two or more events are touching.

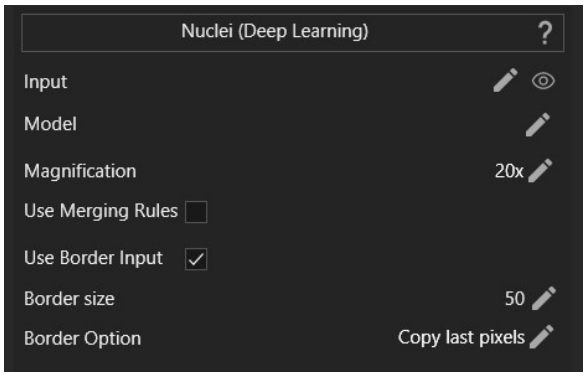
The nuclear segmentation engine works only with images where the nuclei are the bright structures on a dark background, fluorescence-like images.

Note	Nuclei (Deep Learning) requires TissueGnostics Python Environment to be installed on the computer.

Parameters:

- Input - the input image
- Model - the trained deep learning model to be used for nuclear detection

The following parameters can be accessed by enabling the Show Advanced Parameters option, which can be found by pressing the right mouse button on the engine's window.



- Magnification – represents the magnification used to acquire the images. If the information is available, this parameter is set automatically with the corresponding value.

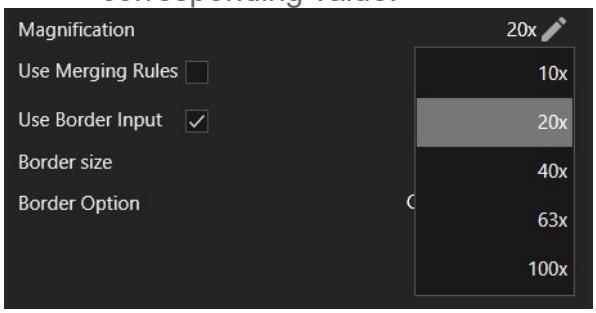
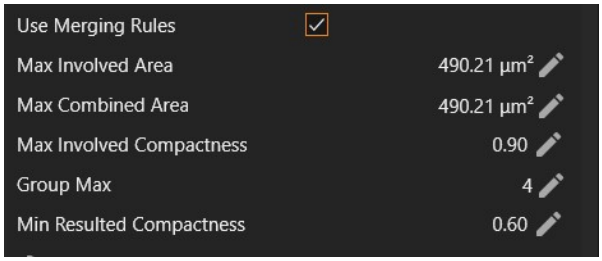


Figure 608 – Magnification parameter

- Use Merging Rules - enables / disables criteria for events merging. All touching events are candidates for merging. Merging will be performed only for the combination of touching events that meet all conditions.



Use Merging Rules parameter

-
    - Max Involved Area - the maximum area of the objects considered for merging;
    - Max Combined Area - the maximum allowed area for resulting objects (after merging);
    - Max Involved Compactness - the maximum allowed compactness of the events considered for merging;
    - Group Max - the maximum number of events that may take part in a combination considered for merging;
    - Min Resulting Compactness - the minimum allowed compactness of the resulting objects (after merging).
- Use Border Input - enables / disables border usage in the segmentation process:

Figure 609 – Border Input Parameter

- Border size – the size of the border added to the current FOV. The border consists in pixels from neighbor FOVs, to preserve to continuity of the information;
- Border option - specifies the artificial border model in case of no neighbor FOVs:
  - Copy last pixels - the last pixels of the image (ones touching borders) are replicated by Border size times
  - Fill with black - the border is entirely black (all values = 0)
  - Fill with white - the border is entirely white (all values = 255)
  - Wrap border - copies the last Border size image pixels

Effect:

Performs nuclear segmentation.

Example:

- Input:

The input image below is a fluorescence grayscale image (8-bit, 1-channel) showing the nuclear marker (DAPI channel).

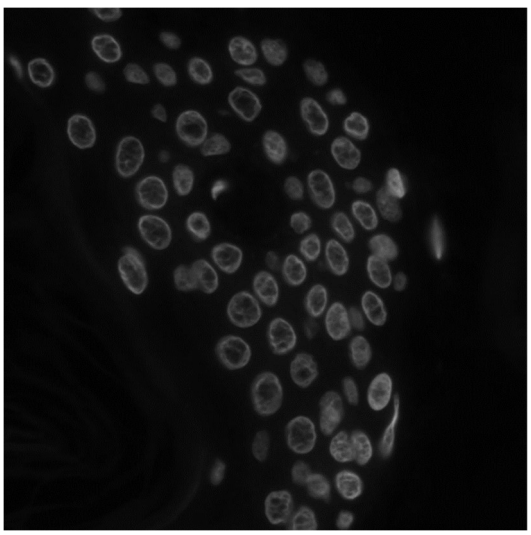


Figure 610 – Input image

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a coded image representing the detected events.

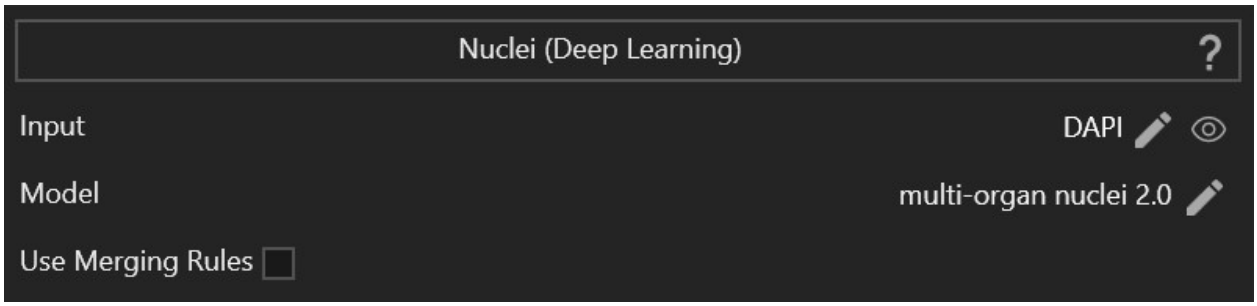


Figure 611 – Nuclei – Deep Learning parameters

- Output:

Figure below shows the detected nuclei overlaid on the original image.

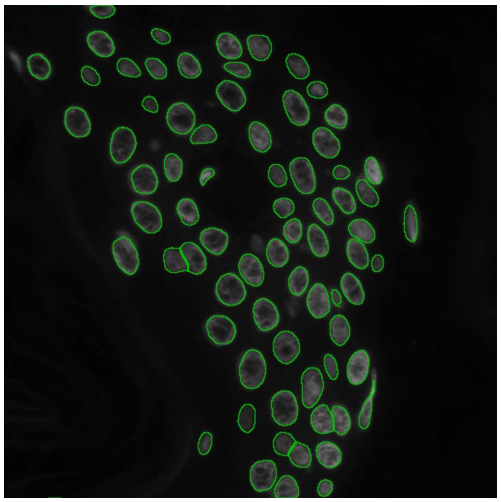


Figure 612 – Output image

24. Nuclei

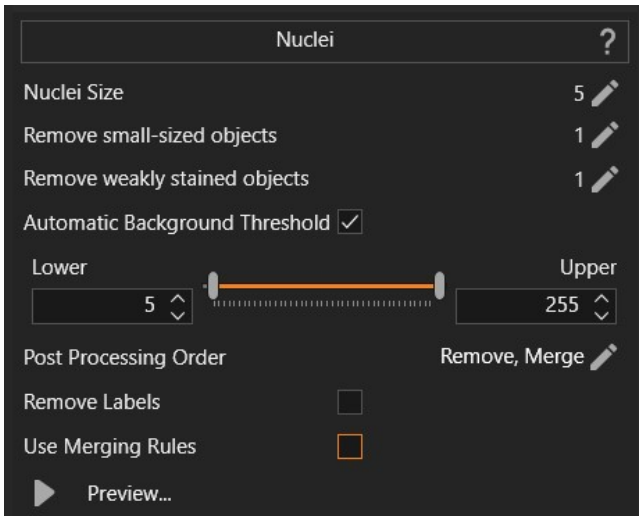


Figure 613 – Nuclei

Where it can be found:

Layer's "Detection" ► Add ► Detection ► Nuclei

Description:

Nuclei segmentation is an engine available in the Detection section of the layers editor.

This engine performs nuclear detection and segmentation on a grayscale image, generating a set of events represented by a coded image. In the coded image, all pixels belonging to an event have the same unique ID, thus preserving the morphology of the event even when two or more events are touching.

The nuclear segmentation engine works only with images where the nuclei are the bright structures on a dark background, in fluorescence-like images.

Parameters:

- Input - the input image
- Nuclei size - the average size of the nuclei present in the source image. Acts like a hint for the algorithm to look for smaller / larger nuclei and does not have a measurements unit.

Figure 614 – Nuclei Size Parameter

- Remove small-sized objects - the percentage of the smallest events to be removed. After the segmentation is complete, all events are sorted by area (ascending order). The smallest events are removed accordingly with the value of this parameter, which can be interpreted as:

$$Remove\ percent\ (\%) = (Remove\ small - sized\ objects * 10)\%$$

Figure 615 – Small-sized objects Parameter

- Remove weakly stained objects - the percentage of the dimmest events to be removed. After the segmentation is complete, all events are sorted by mean intensity (ascending order). The dimmest events are removed accordingly with the value of this parameter, which can be interpreted as:

$$Remove\ percent\ (\%) = (Remove\ weakly\ stained\ objects * 10)\%$$

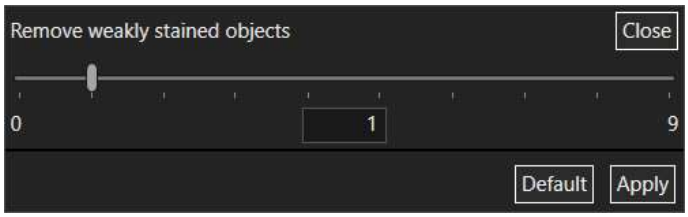


Figure 616 – Weakly stained objects Parameter

- Automatic Background Threshold - enables / disables the automatic computation of the background threshold.
  - Automatic - the automatic background threshold is computed using Otsu's method, which separate pixels into two classes, foreground and background. This threshold is determined by minimizing intra-class intensity variance, or equivalently, by maximizing inter-class variance. In some cases (e.g. uneven illumination), more than two classes can be

identified in the histogram. These kinds of issues can be overcome by changing the intensity range considered by the algorithm:

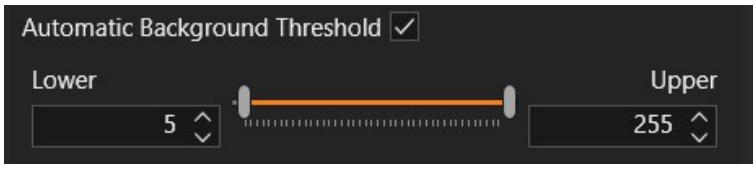


Figure 617 – Automatic Background Threshold Parameter

- Lower - the lower limit of the intensity range. It must be higher than the background values and lower than the dimmest nuclei that must be detected;
- Upper - the upper limit of the intensity range. A lower value forces a lower threshold value. Must be higher than the average nucleus mean intensity.

- Manual - a user-defined value is used as background threshold

Figure 618 – Automatic Background Threshold Parameter: setting threshold

- Post Processing Order - the order of post-processing correction operations:
  - Remove events followed by Merge events;
  - Merge events followed by Remove events.
- Remove Labels - enables / disables criteria for events removal:

Figure 619 – Remove Labels Parameter

- Smaller Than - the lower limit for accepted event's area. All events with smaller area will be removed.
- Larger Than - the upper limit for accepted event's area. All events with larger area will be removed

- Weaker Than - the lower limit for accepted event's mean intensity. All events dimmer will be removed
- Stringer Than - the upper limit for accepted event's mean intensity. All events stronger will be removed
- Use Merging Rules - enables / disables criteria for events merging. All touching events are candidates for merging. Merging will be performed only for the combination of touching events that meet all conditions.

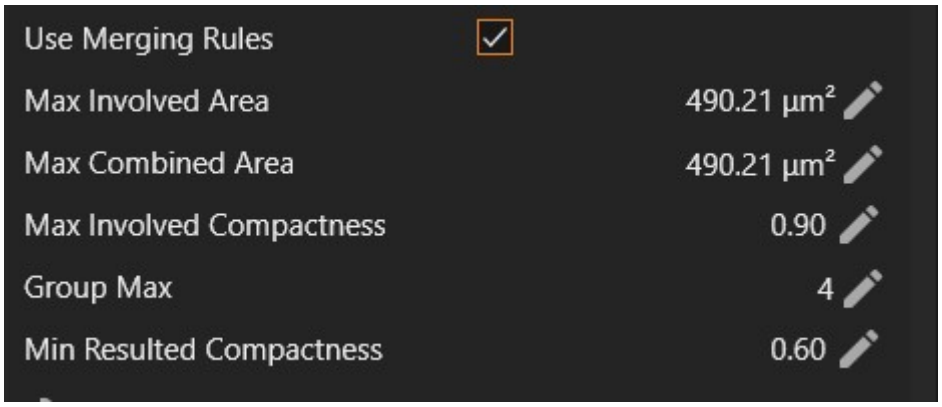


Figure 620 – Use Merging Rules Parameter

  - Max Involved Area - the maximum area of the objects considered for merging;
  - Max Combined Area - the maximum allowed area for resulting objects (after merging);
  - Max Involved Compactness - the maximum allowed compactness of the events considered for merging;
  - Group Max - the maximum number of events that may take part in a combination considered for merging;
  - Min Resulting Compactness - the minimum allowed compactness of the resulting objects (after merging).
- Use Border Input - enables / disables border usage in the segmentation process:

Figure 621 – Border Input Parameter

- Border size - size of the border added to the current FOV. The border consists in pixels from neighbor FOVs, to preserve to continuity of the information;
- Border option - specify the artificial border model in case of no neighbor FOVs:
  - Copy last pixels - the last pixels of the image (ones touching borders) are replicated by Border size times;
  - Fill with black - the border is entirely black (all values = 0);

- Fill with white - the border is entirely white (all values = 255);
- Wrap border - copies the last Border size image pixels.

Effect:

Performs nuclear segmentation.

Example:

- Input:

The input image is a fluorescence grayscale image (8-bit, 1-channel) showing the nuclear marker (DAPI channel).

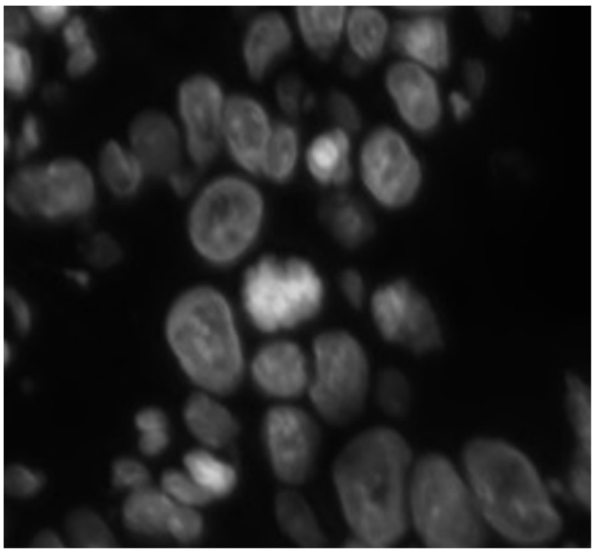


Figure 622 – Nuclear marker sample (DAPI channel)

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a coded image representing the detected events.

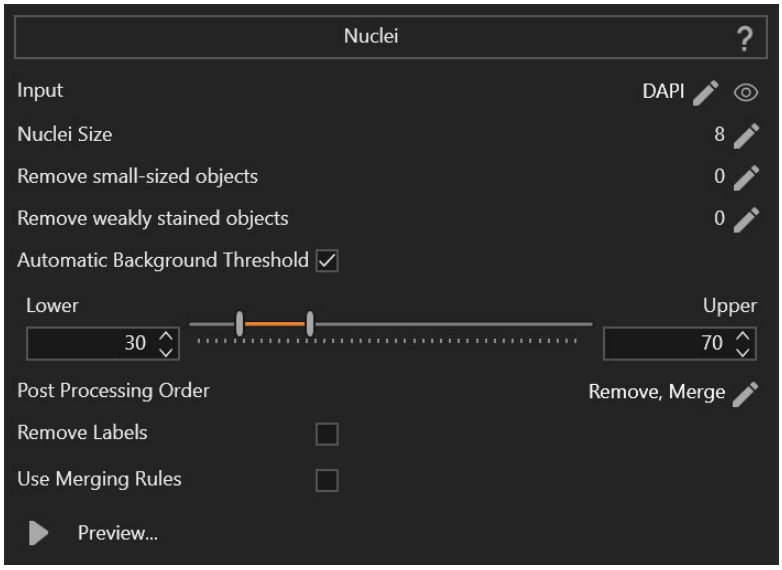


Figure 623 – Engine settings

Output:

Figure below shows the detected nuclei overlaid on the original image.

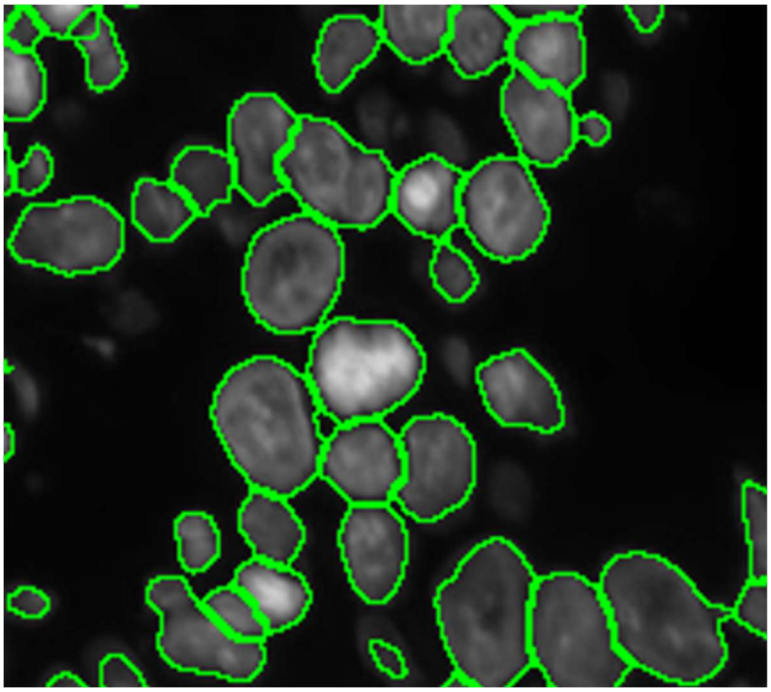


Figure 624 – Result

25. Manual Correction

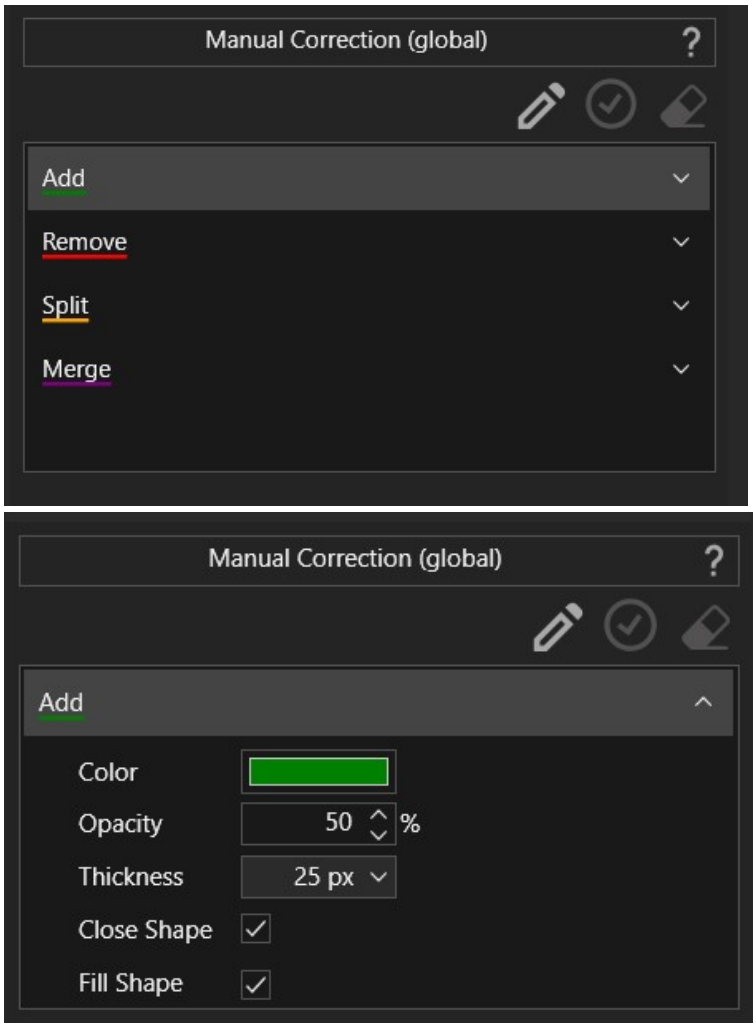


Figure 625 – Manual Correction

Where it can be found:

Layer's "Manual Correction" ► Add ► Manual Correction ► Manual Correction

Description:

Manual Correction is an engine available in the Manual Correction section of the layers editor.

This engine corrects the set of events resulting from the detection process of the current layer. The detection can generate different wrong results, depending on the sample difficulty and image(s) quality:

- False positives - detected events when there is nothing to be detected

- False negatives - undetected events when there is information to be detected
- Under-segmentation - two more structures are detected as one
- Over segmentation - a structure is split into multiple events

The Manual Correction engine offers 4 different types of corrective actions:

- Add - adds a new event to the set of detected events; useful to correct the false negative errors
- Remove - removes an event from the set of detected events; useful to correct false positive errors
- Split - splits an event in two new events; useful to correct under-segmentation errors
- Merge - merges two events into a single event; useful to correct over-segmentation errors

The detection and the correction are treated as a single process. If the correction engine is used, the set of events, available as output from the detection engine, will be the corrected one. The original set of detected events (without correction) is available as one of the Manual Correction engine outputs (Original Detection Results).

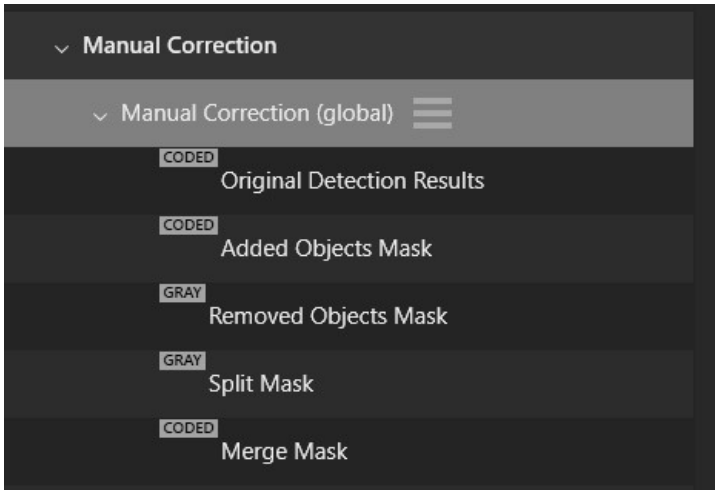


Figure 626 – Manual Correction engine outputs

Each corrective action is indicated by a drawn shape (mask) and multiple shapes can be drawn for each corrective category (mask image). All events intersecting a drawn mask will undergo the corresponding corrective action. For each category, an output mask image is generated, containing all corrective actions applied.

The order of correction operations is always:

$$Correction = Remove \rightarrow Add \rightarrow Split \rightarrow Merge$$

The corrective actions can be managed by activating the drawing mode, using the Start Drawing button:

1. Add new corrective action(s)
    a. enable the drawing mode using the Start drawing button
    b. start drawing with the mouse on the images, to select the events needing correction
    c. use the Apply button when the process is complete

2. Delete corrective action(s)
    a. enable the drawing mode using the Start drawing button
    b. enable the erase mode by using Start erasing button
    c. start erasing drawn shapes with the mouse on the images, to correct the events
    d. use the Apply button when the process is complete

3. Delete all corrective action(s)
    a. enable the drawing mode using the Start drawing button
    b. delete all drawn shapes by using Remove all Objects button
    c. use the Apply button when the process is complete

Parameters:

- What will be corrected - the set of events selected for correction (default = set of events generated in the Detection section of the current layer)
- Manual Correction Action - the correction operation type
- Color - the correction operation assigned color, which is also used in the drawing process
- Transparency - the transparency level of the mask created in the drawing process (default = 100)
- Show only current action shapes - enables / disables the display of the current correction category shapes only
- Thickness - the brush size used to draw shapes
- Automatically close the drawing shape - enables / disables shape's closing. When enabled, the end point of the annotation (when left mouse button is released) will be connected by a line to the starting point of the annotation (when the mouse button was pressed)
- Fill Shape - enables / disables shape's filling. When enabled, only for closed shapes, the interior is filled
- Fill Transparency - sets the opacity level of the shape's interior, only for filled shapes.

Effect:

- Takes defined corrective actions on the set of detected events.

Example:

- Input:

Figure below shows the detected epithelial area in a fluorescence colorectal cancer sample.

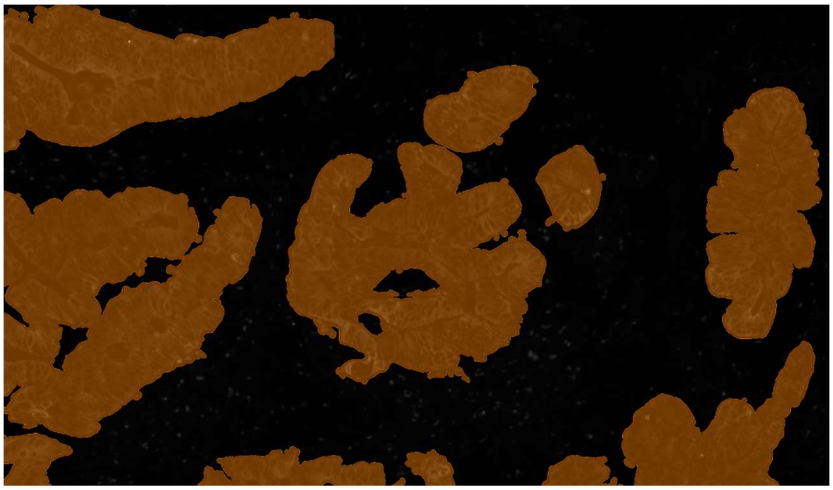


Figure 627 – Detected epithelial area

- Engine settings:

Figure below shows manual drawings for each corrective action class:

- add event (highlighted in green)
- delete event (highlighted in red)
- split events (highlighted in yellow)
- merge events (highlighted in purple)

The engine's result is a coded image representing the corrected set of events.

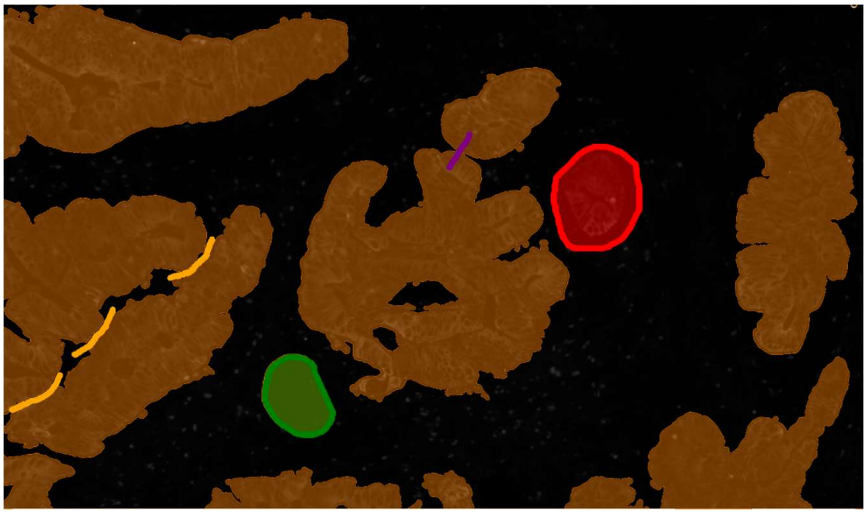


Figure 628 – Manual drawings

Output:

Figure below shows the corrected set of events.

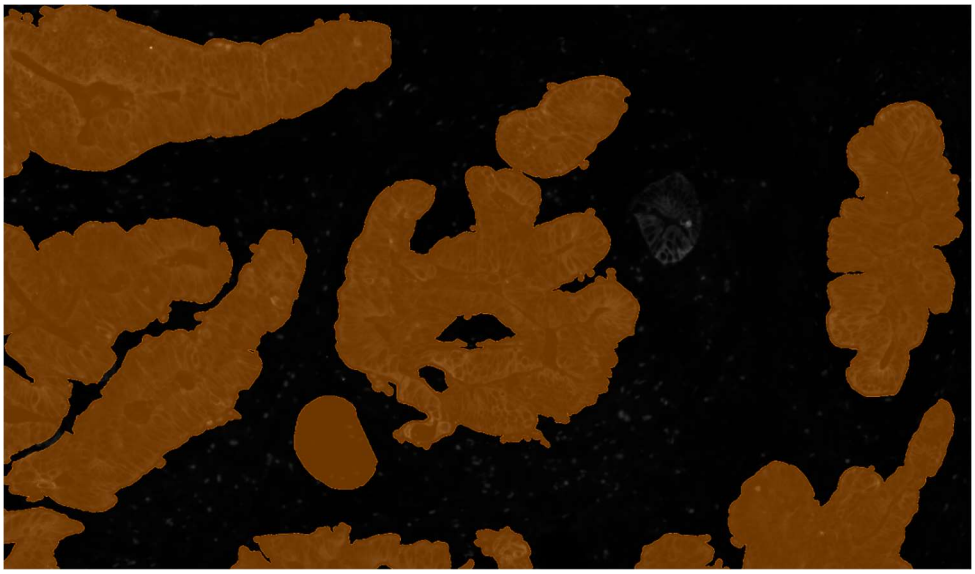


Figure 629 – Result of Manual correction engine

26. Manual Input

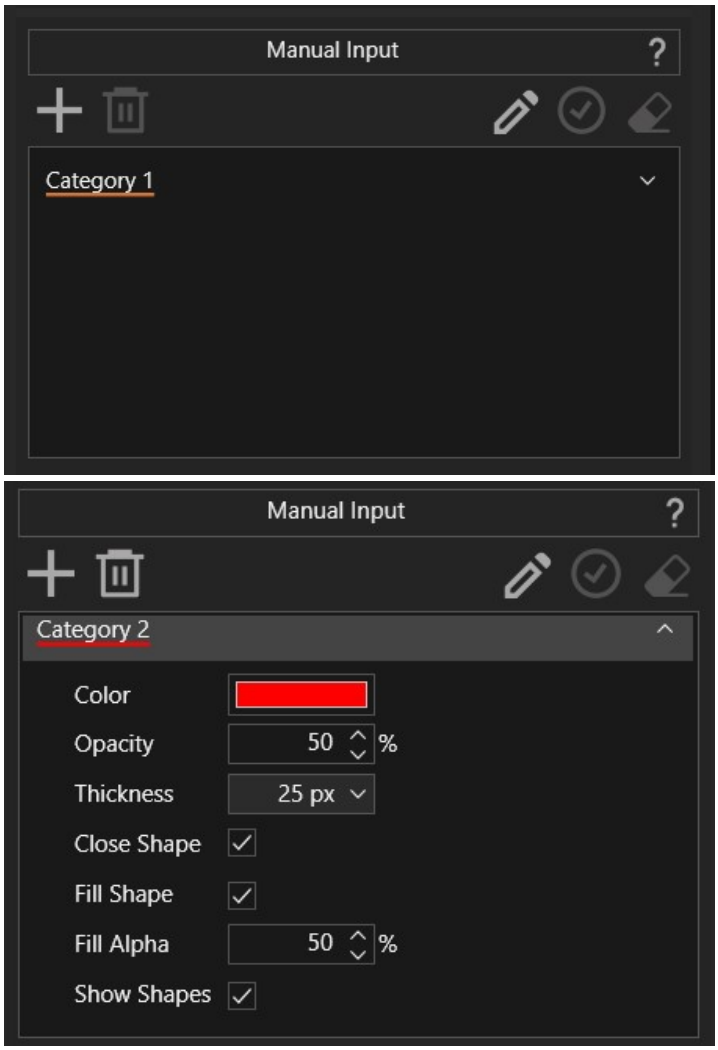


Figure 630 – Manual Input

Where it can be found:

Layer's "Pre-Processing" ► Add ► User Input ► Manual Input

Description:

Manual Input is an engine available in the Pre-Processing section of the layers editor.

This engine allows the creation of annotations part of user defined categories, with the help of a drawing tool. The result can be:

- mask image - useful to restrict detection in areas of interest, to correct detection results by deleting / splitting events, etc.
- set of events (coded image) - useful to get statistical data for all defined categories (e.g. nuclei count). All annotations that are part of a category will share the same unique ID (label).

Categories:

The engine offers the possibility to define categories of annotation or to serve different purposes (e.g. correct detection results by adding new events or by deleting events, mark areas of interest like tumoral areas, interstitial areas, etc.).

- Add new category - adds new category
- Delete selected category - deletes selected category
- Rename - renames selected category by double clicking on the name of the category

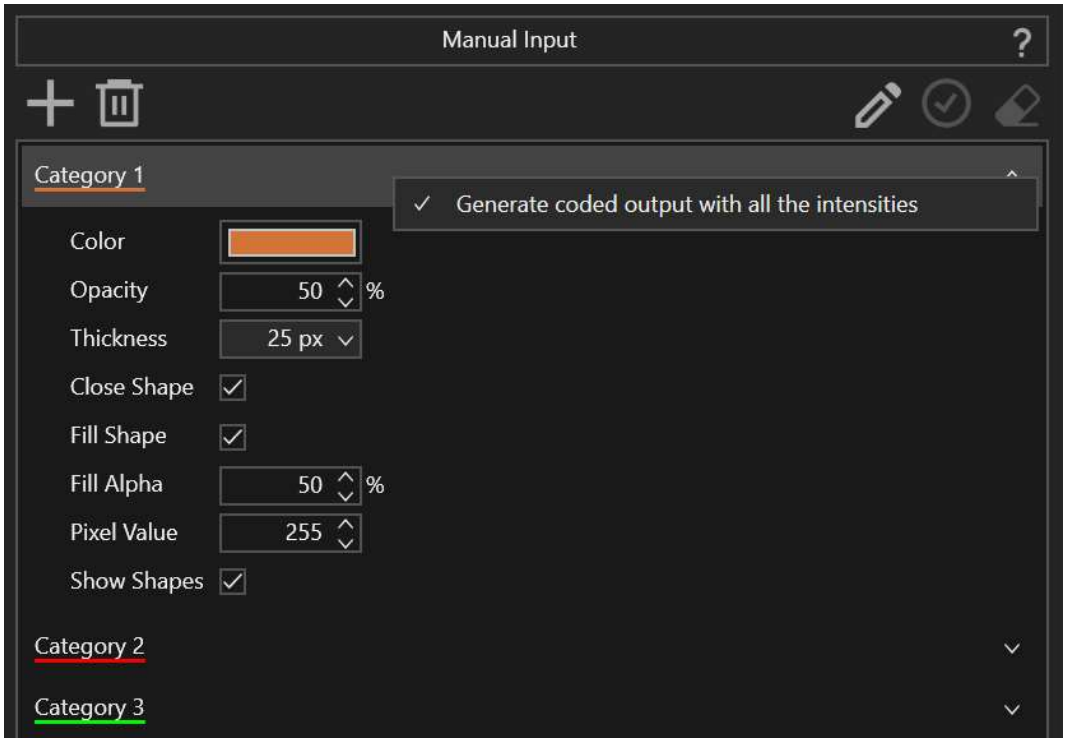


Figure 631 – Manual Input: Generate coded output with all the intensities

Drawing tool:

The drawing brush can be configured using the settings available in the left part of the figure above:

- Color - assigns a color for the selected category. Is used to draw annotations.
- Opacity - sets the opacity level of the annotation. A value of 100% will make the annotation opaque making impossible to see the original image information drawn upon.
-
- Thickness - the size of the brush used to annotate.
- Close shape - enables / disables shape's closing. When enabled, the end point of the annotation (when left mouse button is released) will be connected by a line to the starting point of the annotation (when the mouse button was pressed).
- Fill shape - enables / disables shape's filling. When enabled, only for closed shapes, the interior is filled.

- Fill opacity (alpha) - sets the opacity level of the shape's interior, only for filled shapes.
- Pixel Value - a unique ID (label) assigned to all annotations part of the selected category. Available only if Generate coded output with all the intensities is activated.
- Show shapes - enables / disables the display of the current correction category shapes only.
- Generate coded output with all the intensities (right click on the panel to see this option) - enables / disables the generation of a coded output image, representing all categories as events (all annotations drawn for a category will be considered a single event; different disjoint areas with the same label). It can be useful when of extracting statistical data for relevant areas of interest (e.g. tumoral areas vs. non-tumoral areas, etc.)

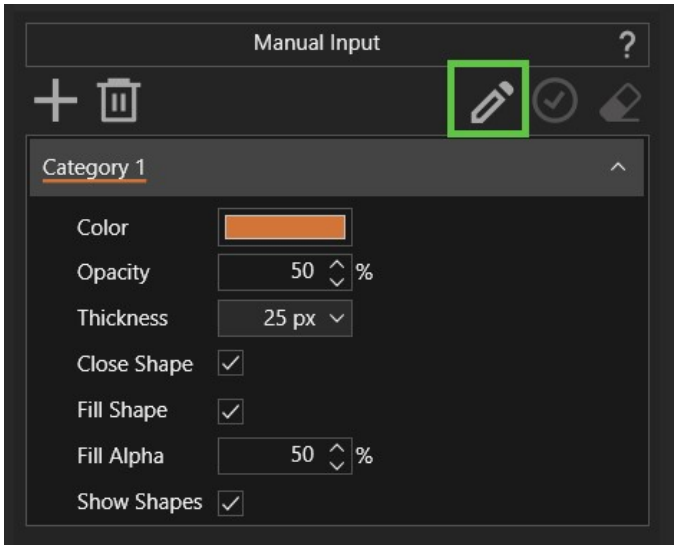


Figure 632 – Manual Input: Drawing settings

The drawing mode is activated using the Start Drawing button. Once activated, the rest of the drawing options become active:

- Start Erasing - enables erasing mode; instead of drawing new annotations, it erases existing ones
- Apply - accepts newly drawn annotations without running the analysis. If the output image is used as input by other engines, the results will not be changed unless the analysis is run again
- Cancel - deactivates drawing mode and discard any annotations made in the current session

Effect:

Generates a mask image for each user defined category, containing all drawn annotations.

Example:

- Input:

Figure below shows manual drawings for 2 defined categories. The drawings are converted to binary masks by the engine on the binary mask for each defined category.

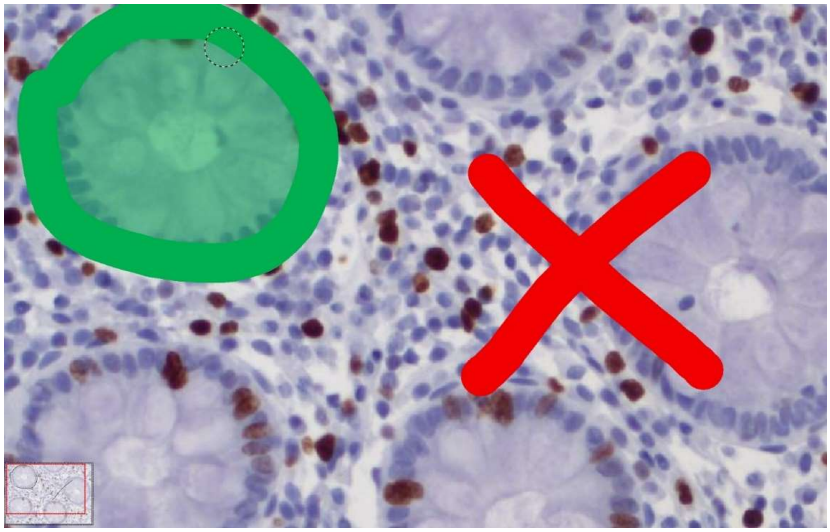


Figure 633 – Manual drawings

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's results are binary masks, one for each defined category.

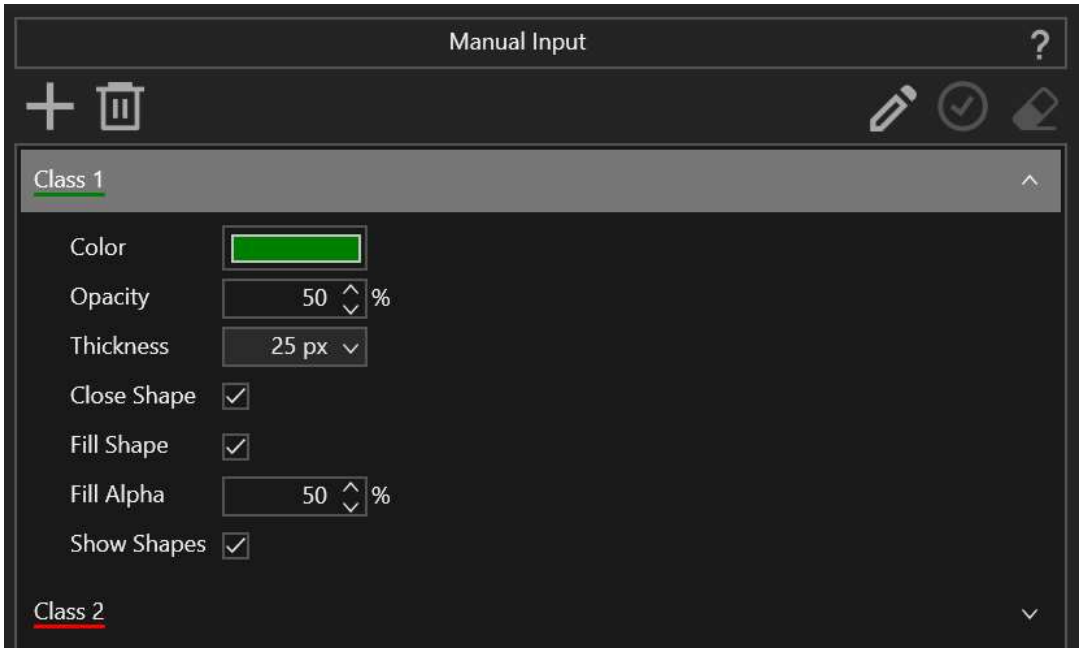


Figure 634 – Engine settings

Output:

Figures below show the generated binary masks for the 2 defined categories, using the manual drawn information.

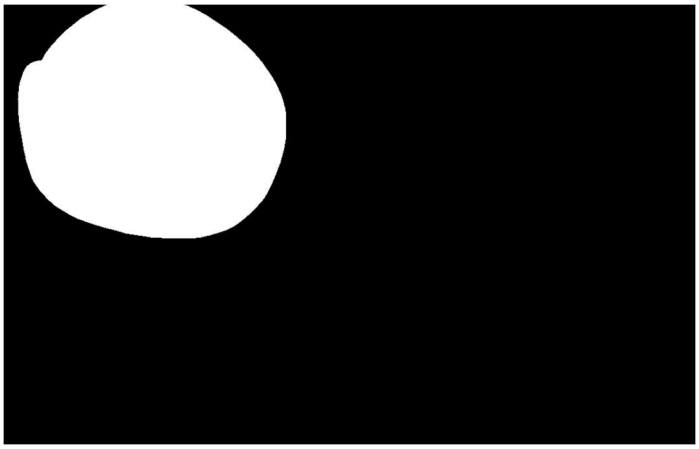


Figure 635 – Result of Manual Input engine (category 1)

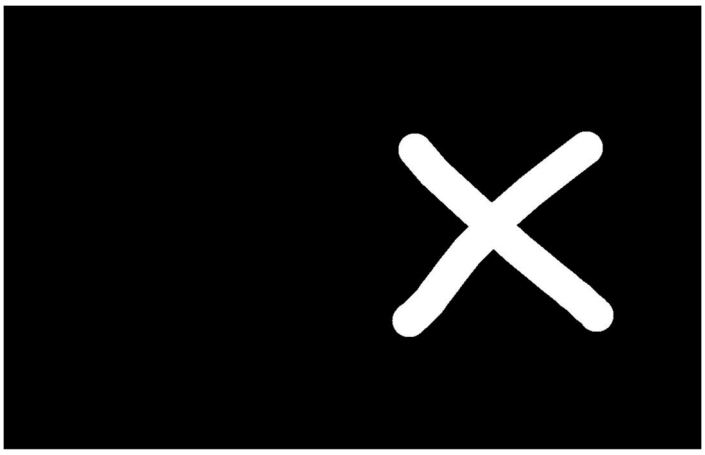


Figure 636 – Result of Manual Input engine (category 2)

27. Membrane

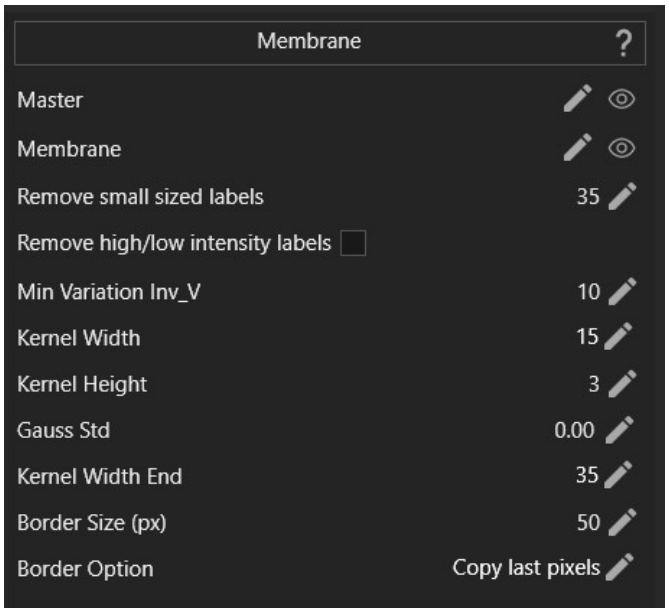


Figure 647 – Membrane

Where it can be found:

Layer's "Pre-Processing" ► Add ► Detection ► Membrane

Description:

Membrane detection is an engine available in the Pre-Processing section of the layers editor.

This engine performs the detection of membrane-like structures on a grayscale image, generating a mask image indicating the presence of membranes and a skeleton image of the detected membranes structures.

The detection is performed using a set of kernels with varying width, given by a user set range (Kernel Width … Kernel Width End).

Parameters:

- Shades (membranes) - inputs grayscale image representing the membrane marker expression
- Remove smaller than - the area threshold for detected objects.
- Remove high/low intensity labels - the intensity threshold for detected objects;

Figure 648 – Membrane: set intensity threshold

- Lower - lower limit for detected objects mean intensity;
- Upper - upper limit for detected objects mean intensity.
- Min variation (for ∧-like) - the minimum variation threshold for ∧-like membranes (default = 0)
- Kernel width (LL) - the lower limit of the kernel width range
- Kernel width (UL) - the upper limit of the kernel width range
- Kernel height - the height of the kernel
- Standard deviation - controls the distribution of weights in the generated Gaussian model used to search for membranes (default = 0)
- Border size - the size of the border added to the current FOV. The border consists in pixels from neighbor FOVs, to preserve the continuity of the information
- Border option - specifies the artificial border model in case of no neighbor FOVs
    - Copy last pixels - the last pixels of the image (touching borders) are replicated by Border size times
    - Fill with black - the border is entirely black (all values = 0)
    - Fill with white - the border is entirely white (all values = 255)
    - Wrap border - copies the last Border size image pixels

Effect:

Detects the presence of membrane-like structures.

Example:

- Input:

Figure below is a grayscale image (8-bit, 1-channel) showing the membrane marker. The image has been generated using the Color Separation engine on an IHC sample.

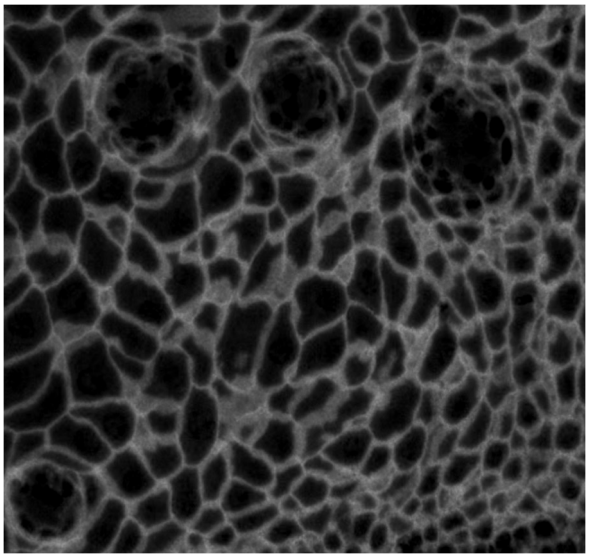


Figure 649 – Input

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's results are 2 binary masks (black & white, 1-channel, 8-bit) representing:

- Membrane thick result – membrane detection result
- Membrane thin result – skeleton of the thick membrane result

Figure 650 – Engine settings

Output:

First figure shows the membrane detected result (thick) while second figure shows the skeleton (thin).

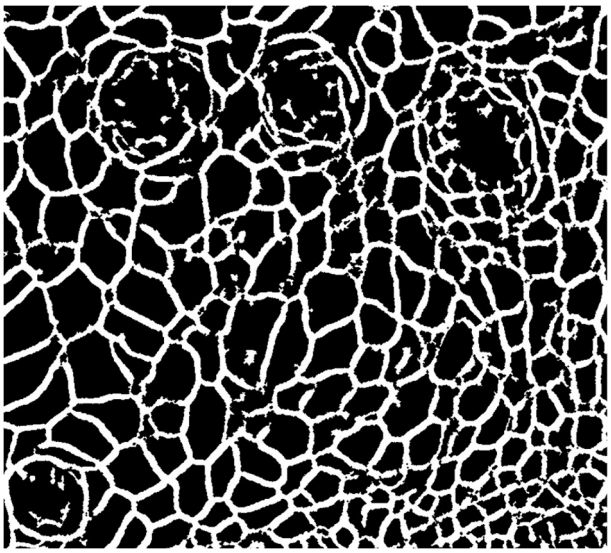


Figure 651 – Membrane thick results

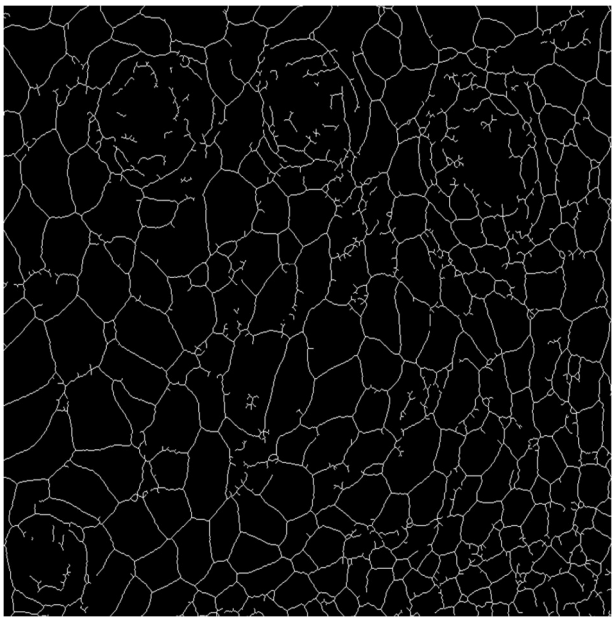


Figure 652 – Membrane thin results

28. Membrane Measurements (local)

Figure 653 – Membrane Measurements local

Where it can be found:

Layer's "Measurements" ► Add ► Measurements (advanced) ► Membrane Measurements (local)

Description:

Membrane Measurements local is an engine available in the Measurements section of the layers editor.

This engine computes membrane-based measurements for a set of events, which may be considered master events (e.g. nuclei). For each master event, membrane-based measurements are computed using information of the proximity detected membranes.

Parameters:

- Coded (master) - input coded image representing the master set of events
- Mask (membranes) - input mask image representing the presence of membrane-like structures.

Different membrane-based measurements can be computed using information from the two input images and a selection of the following list options:

- Angle of staining - an estimation of the angle made by the detected membranes in the proximity of the master event. A virtual disk is imagined starting from the master event, with a radius given by Disk Radius and split into a number of sectors given by Sectors count. The Angle of Staining is approximated by summing angles of all sectors intersecting detected membrane closer than Disk Radius
- Staining - Minimum Distance (µm) - the minimum distance from the master event to the detected membranes
- Staining - Maximum Distance (µm) - the maximum distance from the master event to the detected membranes
- Staining - Mean Distance (µm) - the average distance from the master event to the detected membranes. The value is approximated by averaging the minimum distances of all sectors intersecting membranes.

Effect:

Computes membrane-based measurements for a set of master events.

29. Membrane Measurements (global)

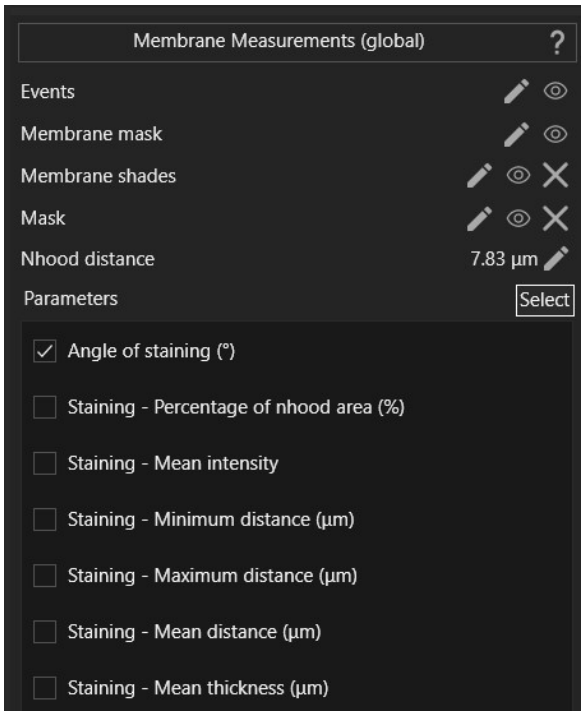


Figure 654 – Membrane measurements global

Where it can be found:

Layer's "Measurements" ► Add ► Measurements (advanced) ► Membrane Measurements (global)

Description:

Membrane Measurements (global) is an engine available in the Measurements section of the layers editor.

This engine generates membrane-based measurements for the detected events:

- the angle generated by the membrane associated / surrounding the detected event
- percentage of the detected event's proximity area covered by membrane
- intensity of the membrane associated / surrounding with the detected event
- distances (min, max, mean) of the membrane associated / surrounding with the detected event
- thickness of the membrane associated / surrounding with the detected event.

The membrane information is represented as a mask image and is provided as an input to the engine. The mask can be generated in the current or in a different layer.

Parameters:

- coded - the input coded image representing the set of events for which the measurements will be computed;
- proximity mask - the input mask image representing the membrane information;
- shades - the input grayscale image used to compute the intensity-based measurements;
- mask - the input mask image used to restrict the measurements computation only inside specified area(s);
- proximity distance - defines the size of the area around the detected event, used to generate the measurements.

The following measurements can be enabled:

- Angle of staining (°) - the angle generated by the membrane mask surrounding the event.
- Staining – Percentage of neighborhood area (%) - the percentage of the event's proximity area covered by the membrane mask;
- Staining – Mean intensity - the mean intensity of the membrane area associated with the event;
- Staining – Minimum distance (um) - minimum distance between the event and the associated membrane mask;
- Staining – Maximum distance) (um) - maximum distance between the event and the associated membrane mask;
- Staining – Mean distance (um) - average distance between the event and the associated membrane mask;
- Staining – Mean thickness (um) - average thickness of the membrane mask associated with the event.

Effect:

Generates the membrane-based measurements for the set of detected events.

Example:

For this example, we consider 2 channels representing:

- Nuclear marker - used to detect the nuclei (coded image), representing the events for which the membrane measurements will be computed. From the entire set of detected events, a single nucleus is considered for this example.
- Membrane marker - used to detect the membrane presence (mask image)

The nuclear marker is used to detect the nuclei which are considered the main events in this example. The detection is performed in the Detection section of the layer, using the

Nuclear Segmentation engine. This generates a coded image representing all detected events. From the entire set of events, we consider a single nucleus for this example (see image below).

Figure 655 – Nucleus mask

The membrane marker (see image below) is used to detect the membrane presence.

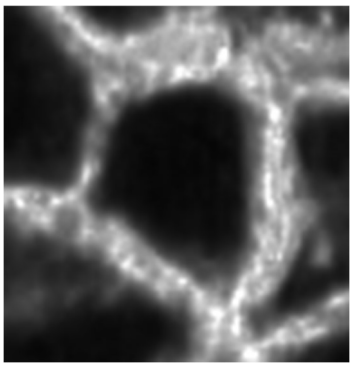


Figure 656 – Membrane marker

This can be done in the Pre-processing section of the current layer or in a different layer and the result is presented as a mask image (see image below).

Figure 657 – Membrane mask

Fig. 635 shows the surrounding area defined by a proximity distance = 30 pixels and fig. 636 shows the overlay of the 2 masks: nucleus with blue and surrounding area in red.

The membrane measurements are computed in a proximity area defined by proximity distance. The two images below show the surrounding area for the nucleus considered in this example.

Figure 658 – Event's surrounding area (defined by a proximity distance = 30 pixels)

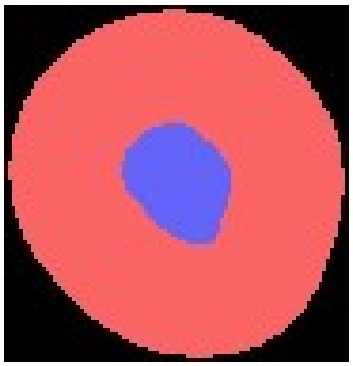


Figure 659 – Overlay of the 2 masks: nucleus with blue and surrounding area in red

The proximity area is split in roughly equal sectors representing the orientation in relation with the event (nucleus). Figure below shows a circle split into 360 sectors (1 sector = 1 degree) highlighted by different colors. In our example, the number of sectors is limited by the proximity distance and the pixel representation of the information.

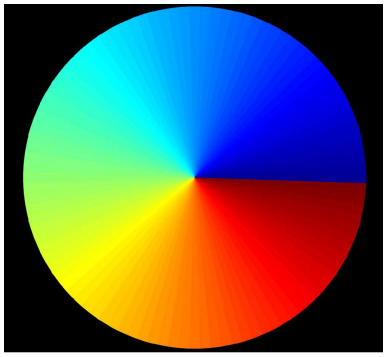


Figure 660 – Area split into sectors

The lower limit is presented in the figure below. The event is highlighted in blue and has an area = 1 pixel, the proximity distance is 1 pixel which defines the proximity area highlighted in green. The proximity area can be split into maximum 8 sector, each sector = 1 pixel and each sector representing approx. 360 degrees / 8 = 45 degrees.

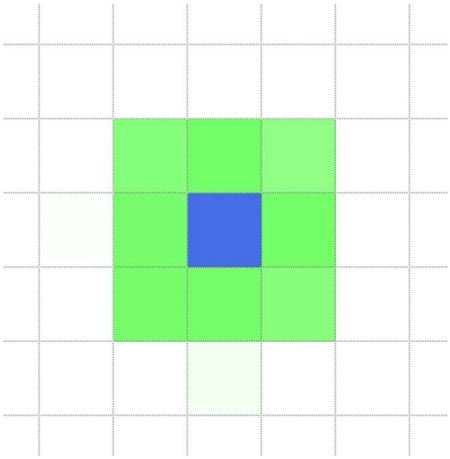


Figure 661 – Pixel representation limitation

With the information presented above, the selected membrane measurements are computed:

Angle of Staining:

Value (in degrees [0 ... 360]) representing how much of the event is surrounded by membrane. This value is generated by the sum of all sectors intersecting the membrane mask.

Figure below shows:

- Nucleus (blue)
- Proximity area (red)
- Membrane mask (yellow)
- Empty sectors (brown)

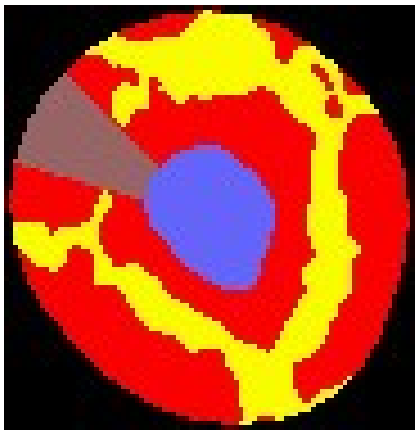


Figure 662 – Angle of staining

Staining – Percentage of neighborhood area (%):

Figure below shows:

- Nucleus (blue)
- Proximity area (red)
- Membrane mask (yellow)

The value (percent [0 … 100]) represents the percentage of the proximity area (red) covered by membrane (yellow).

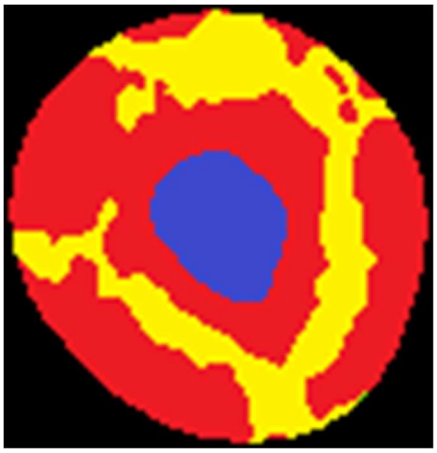


Figure 663 – Staining - Percentage of neighborhood area

Staining – Mean intensity:

Figure below shows:

- Nucleus (blue)
- Proximity area (red)
- Membrane intensities (green)

The value represents the mean intensity of the membrane area surrounding the nucleus. The value is computed using intensities of the pixels indicated by the membrane mask.

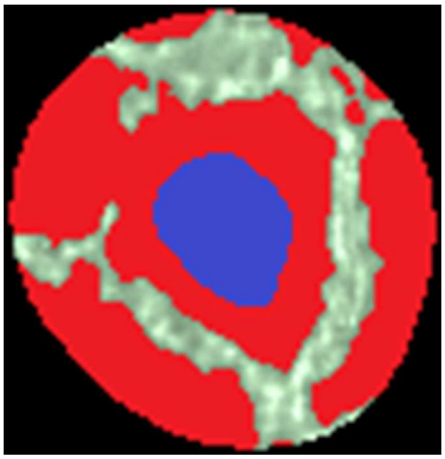


Figure 664 – Staining - mean intensity

Staining – Maximum distance (um):

Figure below shows:

- Nucleus (blue)
- Proximity area (red)
- Membrane mask (yellow)
- Distances from nucleus to membrane (green)

The value represents the minimum distance (green from image Staining - mean intensity) from nucleus to membrane mask.

Figure 665 – Staining - Distances

Staining – Mean distance (um):

The value represents the maximum distance (green from image Staining - mean intensity) from nucleus to membrane mask.

Staining – Mean thickness (um):

The value represents the average distance (green from image Staining - mean intensity) from nucleus to membrane mask.

30. No Detection

Where it can be found:

Layer's "Detection" ► Add ► Detection ► No Detection

Description:

No detection is an engine available in the Detection section of the layers editor.

No detection is a dummy engine, it does nothing. It is useful when no detection is needed and only engines in the pre-processing section of the layer need to be run.

Parameters:

- No parameters available.

Effect:

Skips the detection process.

31. Otsu Threshold (global)

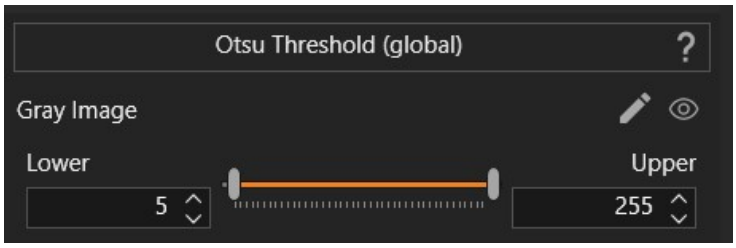


Figure 666 – Otsu Threshold

Where it can be found:

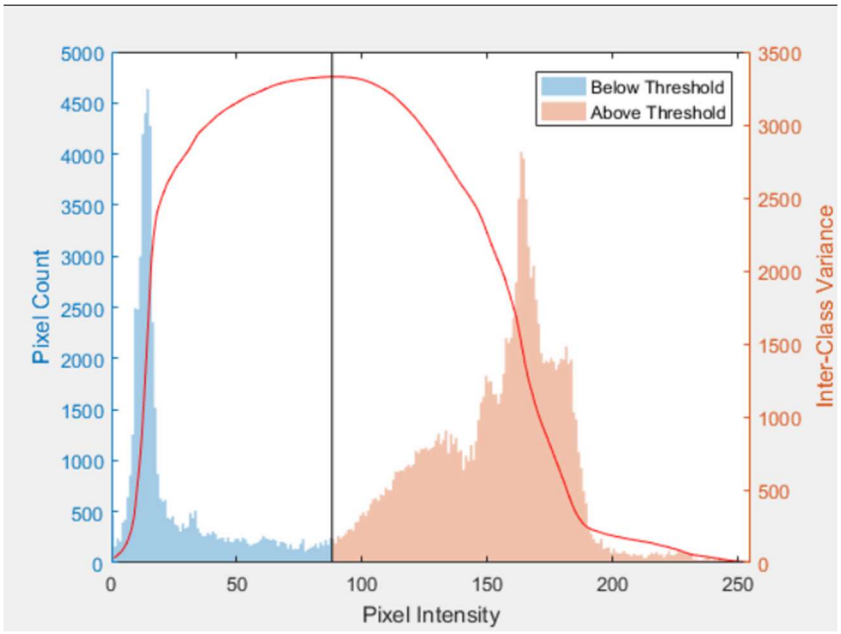
Layer's "Pre-Processing" ► Add ► Operations (advanced) ► Otsu Threshold (global)

Description:

Otsu Threshold (global) is an engine available in the Detection section of the layers editor.

This engine computes a threshold value using the information of all Fields of View (FOVs), based on Otsu's method.

The Otsu algorithm returns a single intensity threshold that separates pixels in two classes: foreground and background. This threshold is determined by minimizing intra-class intensity variance (or by maximizing inter-class variance).



Otsu's method exhibits the relatively good performance if the histogram can be assumed to have bimodal distribution and assumed to possess a deep and sharp valley

between two peaks. But if the object area is small compared with the background area, the histogram no longer exhibits bimodality. And if the variances of the object and the background intensities are large compared with the mean difference, or the image is severely corrupted by additive noise, the sharp valley of the gray level histogram is degraded. Then the possibly incorrect threshold determined by Otsu's method results in a segmentation error. In this case, the intensity range used by the algorithm can be limited by Lower Limit and Upper Limit.

Parameters:

- Gray Image - inputs grayscale image;
- Lower - the lower limit of the intensity range. Must be higher than background values and lower than the dimmest areas that must be detected;
- Upper - the upper limit of the intensity range. A lower value forces a lower threshold value. Must be higher than the average intensity of areas that must be detected.

Effect:

Computes threshold level for the entire sample or region of interest, using Otsu's method.

Example:

- Input:

The input image is a fluorescence grayscale image (8-bit, 1-channel) showing the SpGold channel of a colorectal cancer sample.

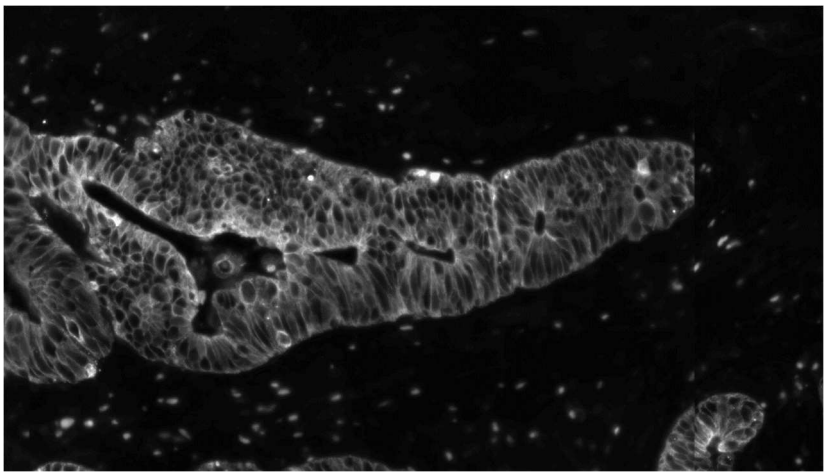


Figure 667 – Colorectal cancer sample (SpGold channel)

- Engine settings:

First figure shows the Total Area engine settings used for this example. The Total Area engine uses the threshold value generated by the Otsu Threshold (global) engine using the settings shown in second figure.

Figure 668 – Engine settings – Otsu Threshold (global)

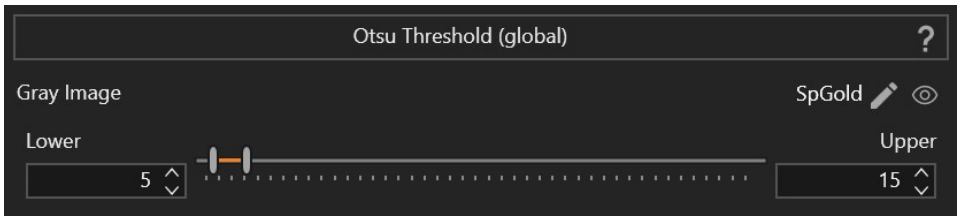


Figure 669 – Engine settings – Total Area

Output:

Figure below shows the result of Total Area engine using the threshold value generated by Otsu Threshold (global) engine.

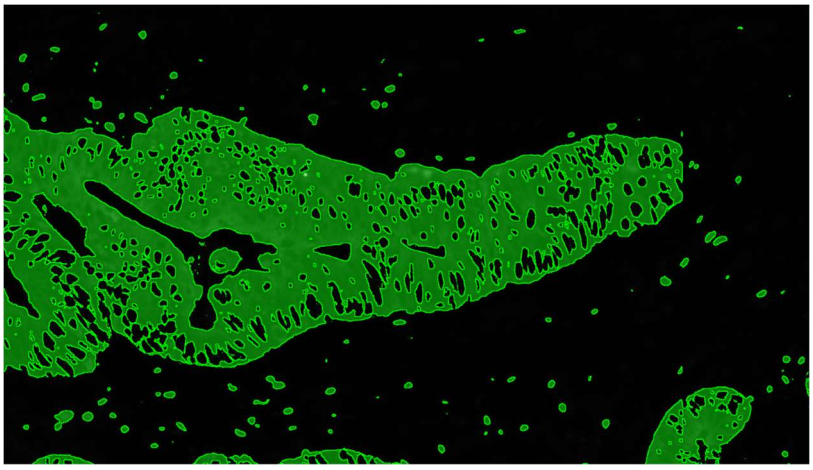


Figure 670 – Result of Total Area engine using Otsu Threshold (global) engine

32. Projection

Projection

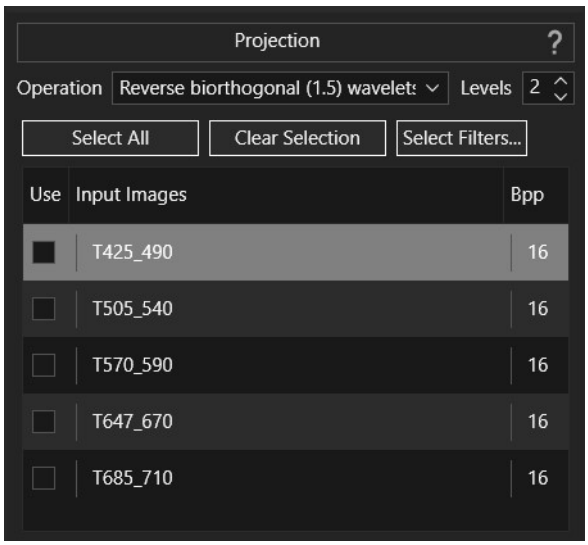


Figure 671 – Projection

Where it can be found:

Layer's "Pre-Processing" ► Add ► Operations (advanced) ► Projection

Description:

Projection is an engine available in the Pre-Processing section of the layers editor.

This engine combines the information of the selected source images into a new image. Usually, the source images are slices of a z-stack (a set of images taken at different focus distances), which, combined, generate an image with a greater depth of field than any of the individual source images. It can be used in any situation where individual images have a very shallow depth of field.

Several methods are available to fuse images:

- Sum - fuses information by summing the corresponding source images pixels
- Max - fuses information by computing the max of corresponding source images pixels
- Daubechies 1 - fuses information using Daubechies 1 (db1) wavelet
- Daubechies 2 - fuses information using Daubechies 2 (db2) wavelet
- Biorthogonal 1.5 - fuses information using Biorthogonal 1.5 (bior1.5) wavelet
- Reverse biorthogonal 1.5 - fuses information using Reverse biorthogonal 1.5 (rior1.5) wavelet
- Discrete Meyer - fuses information using Discrete Meyer (dmey) wavelet

Except Sum and Max, the rest of the methods use wavelet decomposition of source images in the fusing process.

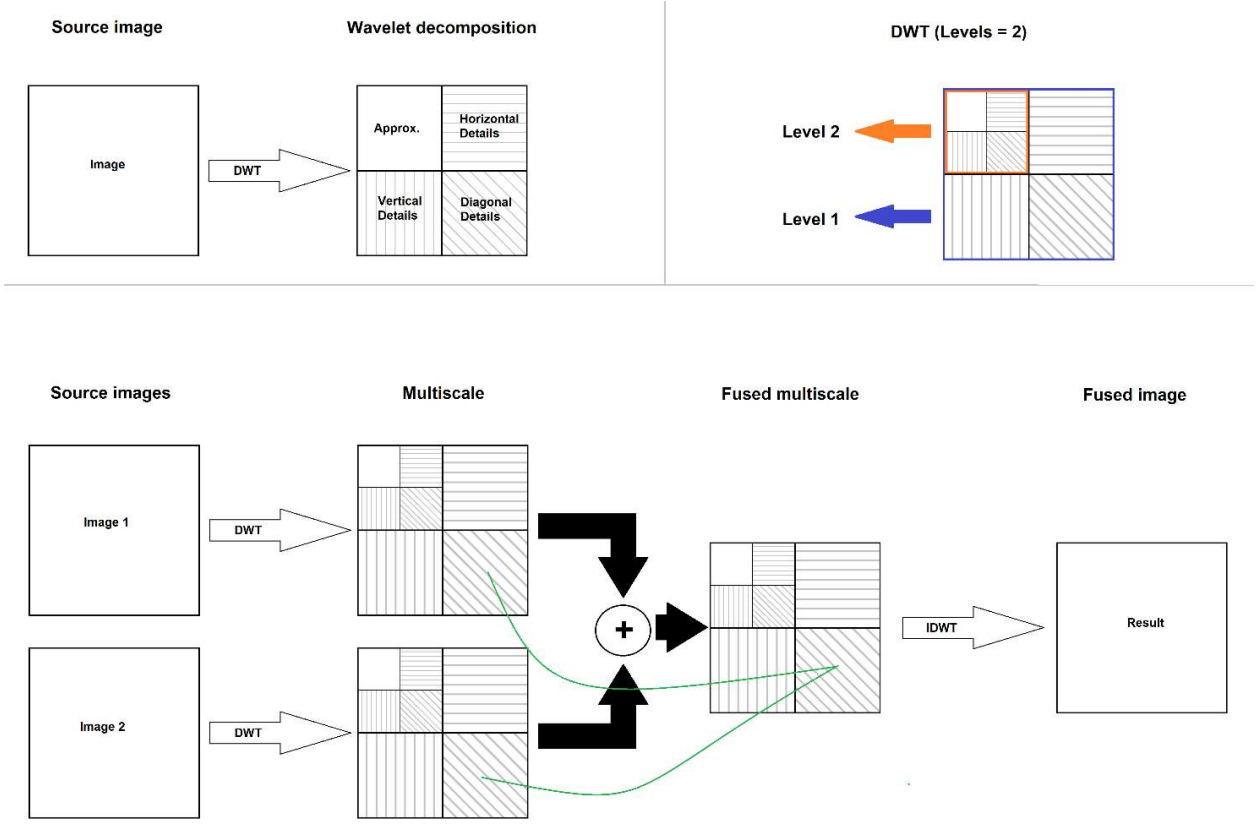


Figure 672 – Fuse images using Discrete Wavelet Transform (DWT)

Parameters:

- Operation - selected method used to fuse the images
- Levels - wavelet decomposition level (default = 2)
- List of images - all images available for fuse are displayed for selection

Effect:

Combines the information of different z-stack slices into an image with a greater depth of field.

Example:

- Input:

The input is represented by the list of all images acquired at different Z-positions (Z-stack).

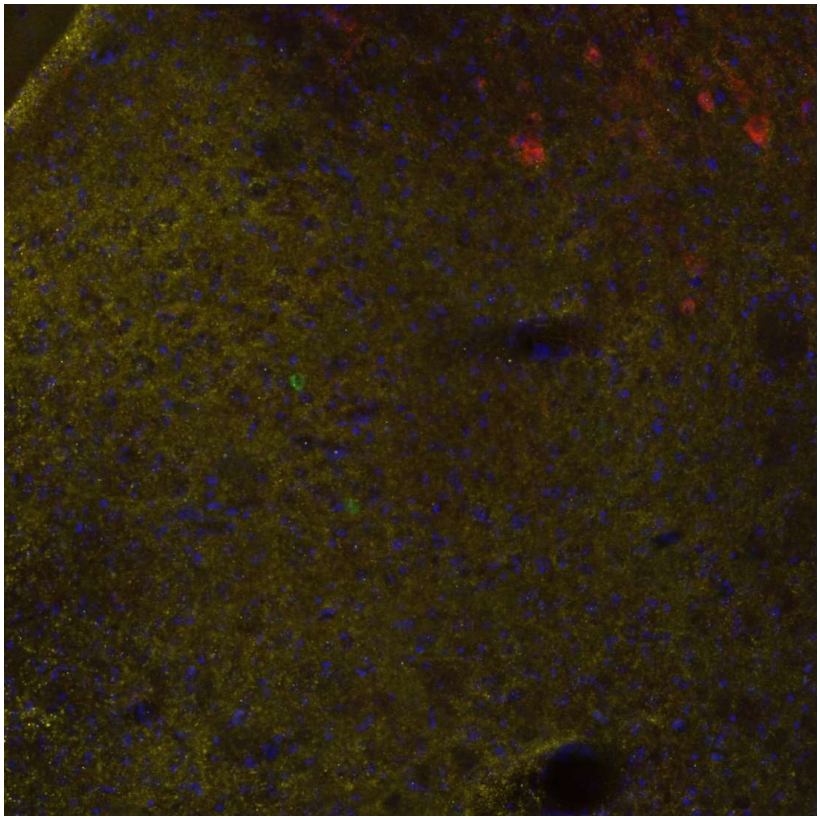


Figure 673 – Input image (two channels)

Output:

The result represents the fused information of all input images.

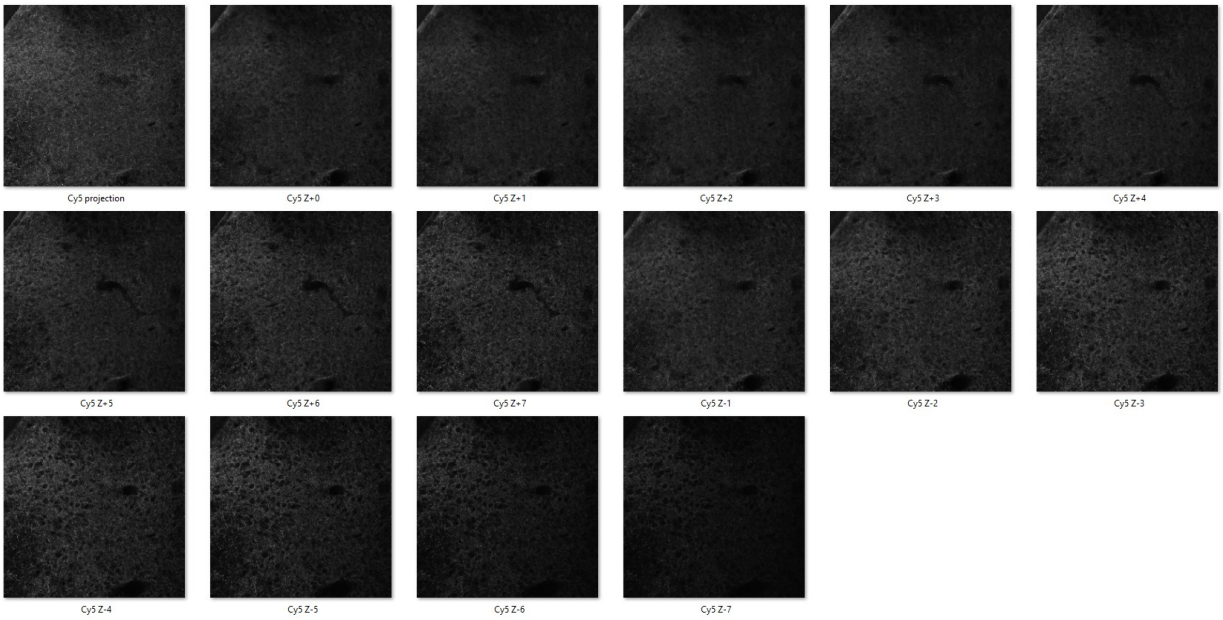


Figure 674 – 15 z-levels for Cy5

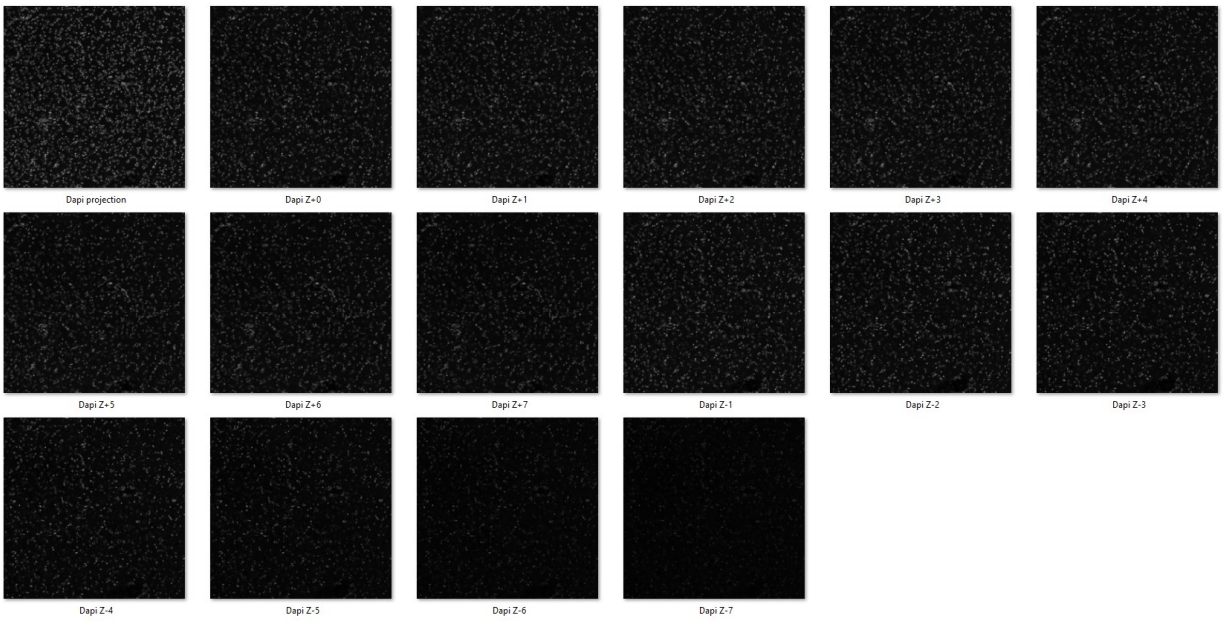


Figure 675 – 15 z-levels for DAPI

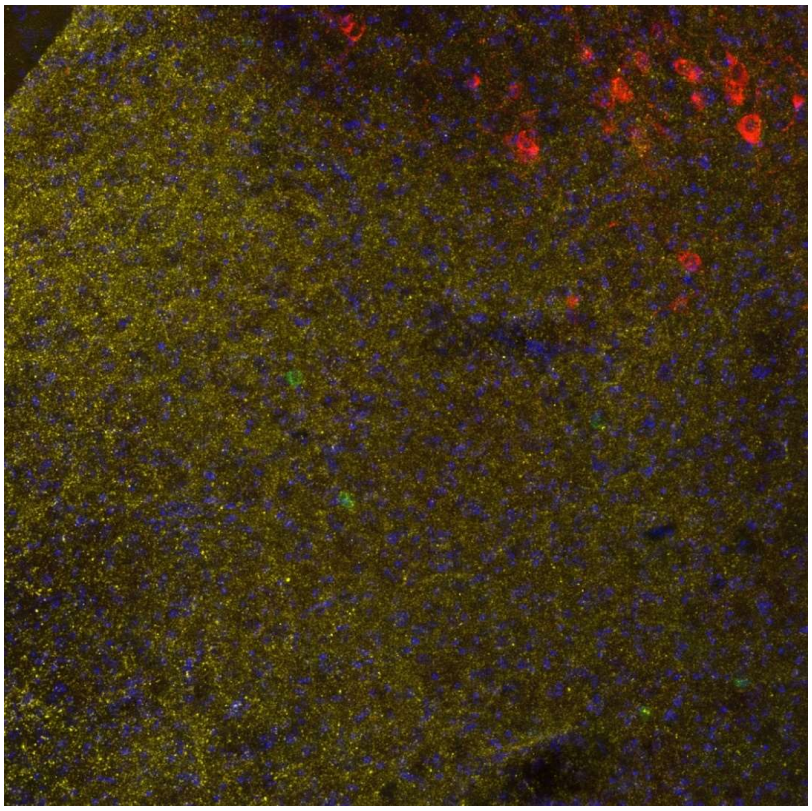


Figure 676 – Result: Overlay – Projection (4 channels)

33. Proximity Areas

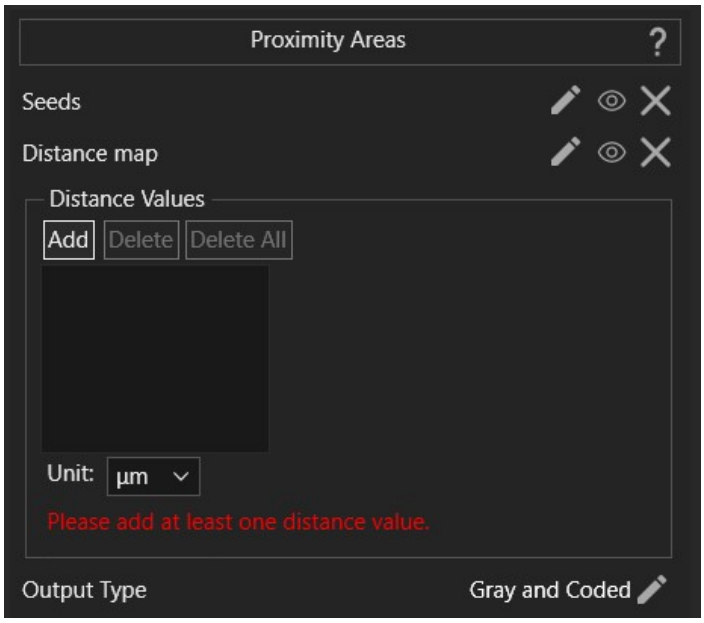


Figure 677 – Proximity Areas

Where it can be found:

Layer's "Pre-Processing" ► Add ► Operations (advanced) ► Proximity Areas

Description:

Proximity Areas is an engine available in the Pre-Processing section of the layers editor.

This engine generates proximity areas using a list of user defined distances, relative to indicated areas of interest. The areas of interest are represented by sets of pixels that have relevance in the analysis workflow, like a set of events (coded image), detected areas (mask image), etc. Each proximity area is defined as the set of pixels located between two consecutive distance values from the list (first proximity area is between 0 = area of interest contour and first distance value from the list).

Parameters:

- Seeds - inputs an image representing the reference (areas of interest) used to generate the proximity areas
- Distance map - inputs a distance map. The distances are generated having the areas of interest as reference (see Distance Map engine).

Note	Only one of the two input images is required.

	If Seeds is provided, Distance map is computed internally in accordance with the information present in the Seeds image. If Distance map is provided, it includes the location of areas of interest (0 valued distances).

- Distance Values - the list of distances used to generate the proximity areas. All values must be strict positives and are sorted in ascending order. The list management is done with the help of the available options:
  - Add - adds a new distance value which will define a new proximity area
  - Remove - deletes a distance value from the list
  - Remove all - clears list by removing all entries
- Unit - specifies the measurement unit used to generate the proximity areas. The options are: µm (default) and pixel.
- Output Type - specifies mode used to represent the output:
  - Gray - the output is represented as a grayscale image (useful for measurements)
  - Coded - the output is represented using a coded image (useful for visualization)
  - Gray and Coded - both types of output representations are generated

Effect:

Generates proximity areas relative to indicated areas of interest.

Example:

- Input:

Figure below shows the detected subcapsular sinus (highlighted in yellow) in a lymphatic tissue sample. The aim of this example is to generate proximity areas starting from the detected subcapsular sinus. Each proximity area has a distance range of 20 µm.

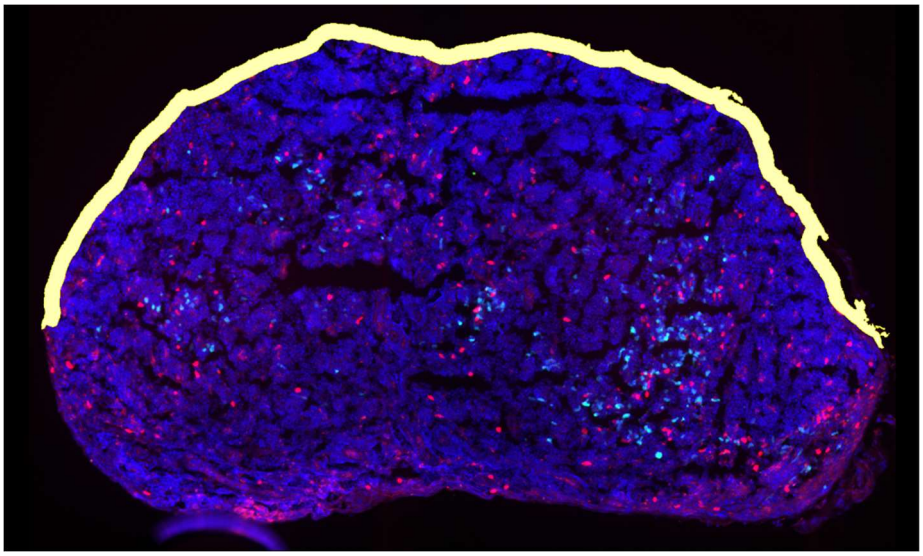


Figure 678 – Lymphatic tissue sample

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a grayscale image and / or a coded image representing the generated proximity areas (depending on user selection).

Figure 679 – Engine settings

Output:

Figure below shows the generated proximity areas highlighted in different shades of gray, in relation with the event(s) of reference highlighted in yellow.

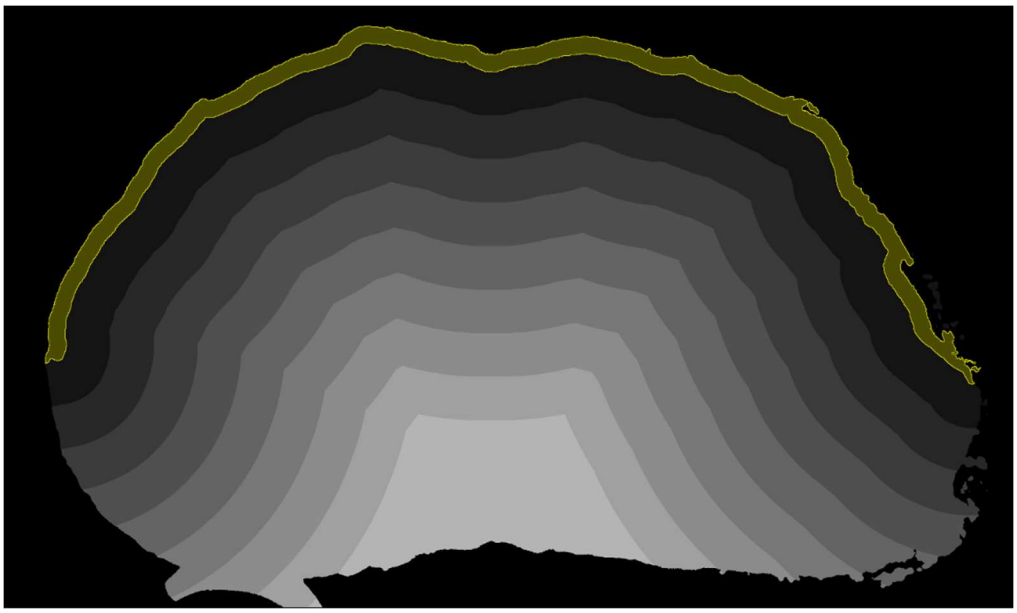


Figure 680 – Result of Proximity Areas engine

34. Remove objects v2

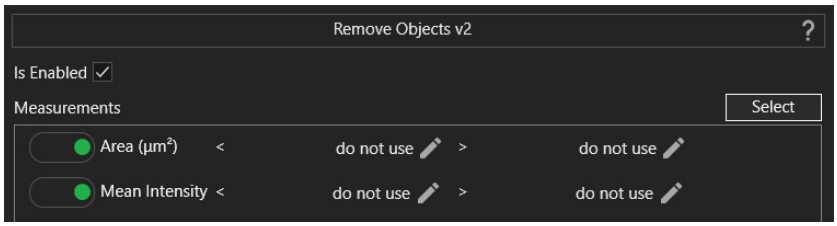


Figure 691 – Remove Objects

Where it can be found:

Layer's "Post-Processing" ► Add ► Post-Processing ► Remove objects v2

Description:

Remove objects is an engine available in the Post-Processing section of the layers editor.

This engine removes detected events based on selected criteria.

Parameters:

- Is Enabled – an option to enable / disable the engine
- Select - opens the list of parameters available to define new removal criteria (by default, only Area and Mean Intensity are selected)

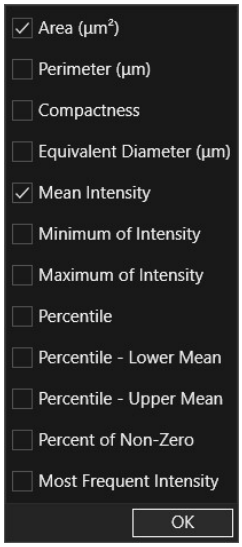


Figure 692 – Parameter list

- Measurements – shows the list of defined removal criteria. Each criterion consists of:

1. An option to enable / disable the current criterion
2. The selected measurement
3. The lower limit
4. The upper limit

All events having the selected measurement value (2) lower that the Lower Limit (3) and higher than the Upper Limit (4) will be removed. The output will consist in all events having the selected measurement value (2) in the range (Lower Limit … Upper Limit).

$$Result = \{Events, Lower\ Limit < Area(Event) < Upper\ Limit\}$$

$$Result = \{Events, Weaker\ than < Mean\ Intensity(Event) < Stronger\ than\}$$

Effect:

Cleans up the set of detected events.

35. Remove objects touching border

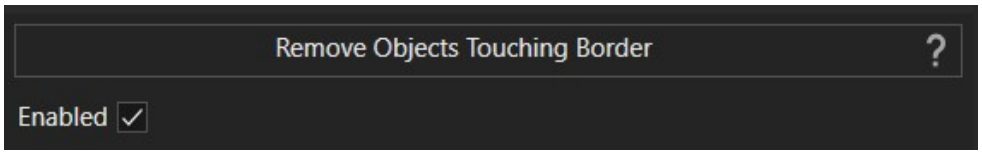


Figure 693 – Remove Objects Touching Border

Where it can be found:

Layer's "Post-Processing" ► Add ► Post-Processing ► Remove objects touching border

Description:

Remove Objects Touching Border is an engine available in the Post-Processing section of the layers editor.

This engine removes detected events touching the sample or region of interest border.

Parameters:

- Enabled - enables / disables the removal of touching border events.

Effect:

Cleans up the set of detected events by removing events representing incomplete structures.

Example:

- Input:

The input consists of a set of events representing the detected structures (highlighted in green) in an IHC colon sample.

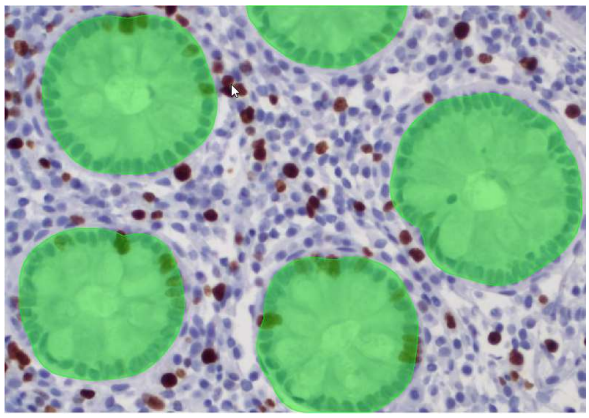


Figure 694 – Detected structures

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a coded image representing the events which are not in direct contact with the borders.

Figure 695 – Engine settings

Output:

Figure below shows the resulting events.

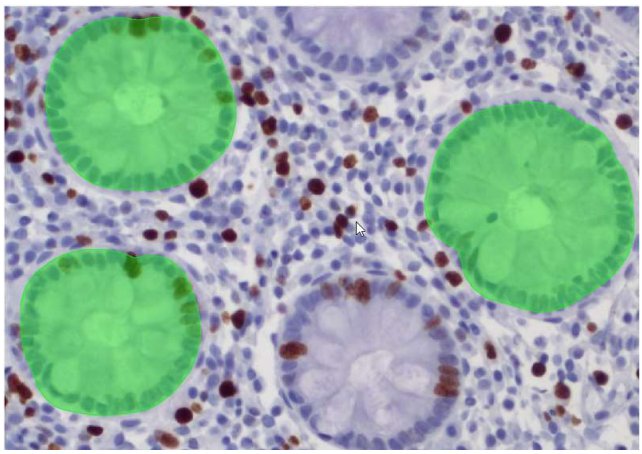


Figure 696 – Result of Remove objects touching border engine

36. Spectral Unmixing

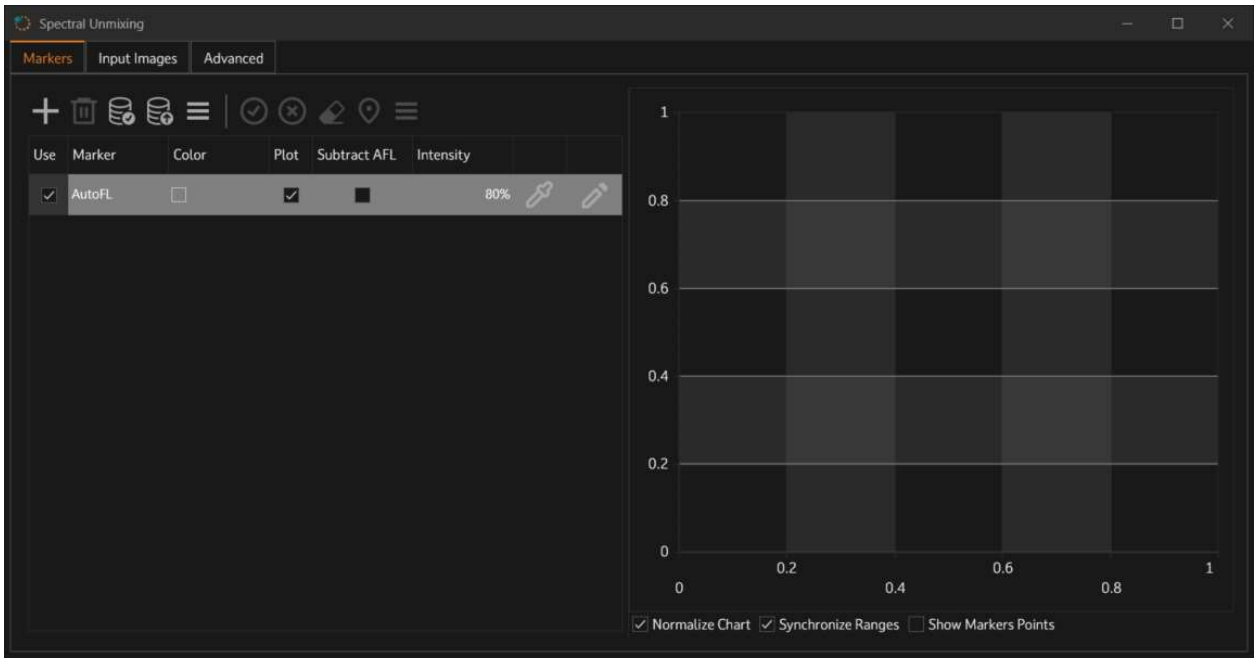


Figure 697 – Spectral Unmixing

Where it can be found:

Layer's "Pre-Processing" ► Add ► Operations (Advanced) ► Spectral Unmixing

Description:

Spectral Unmixing is an engine available in the Pre-Processing section of the layers editor.

This engine separates the measured spectrum (mixed signal pixel) into a collection of constituent spectra, or endmembers, and a set of corresponding fractions or abundances that indicate the proportion of each endmember present in the pixel.

Multispectral imaging and unmixing are useful to separate fluorophores having overlapping emission spectra and for removing unwanted autofluorescence.

Parameters:

The Spectral Unmixing engine consists of 3 tabs used for:

- Managing markers
- Managing spectral images
- Advanced settings.

Managing Markers

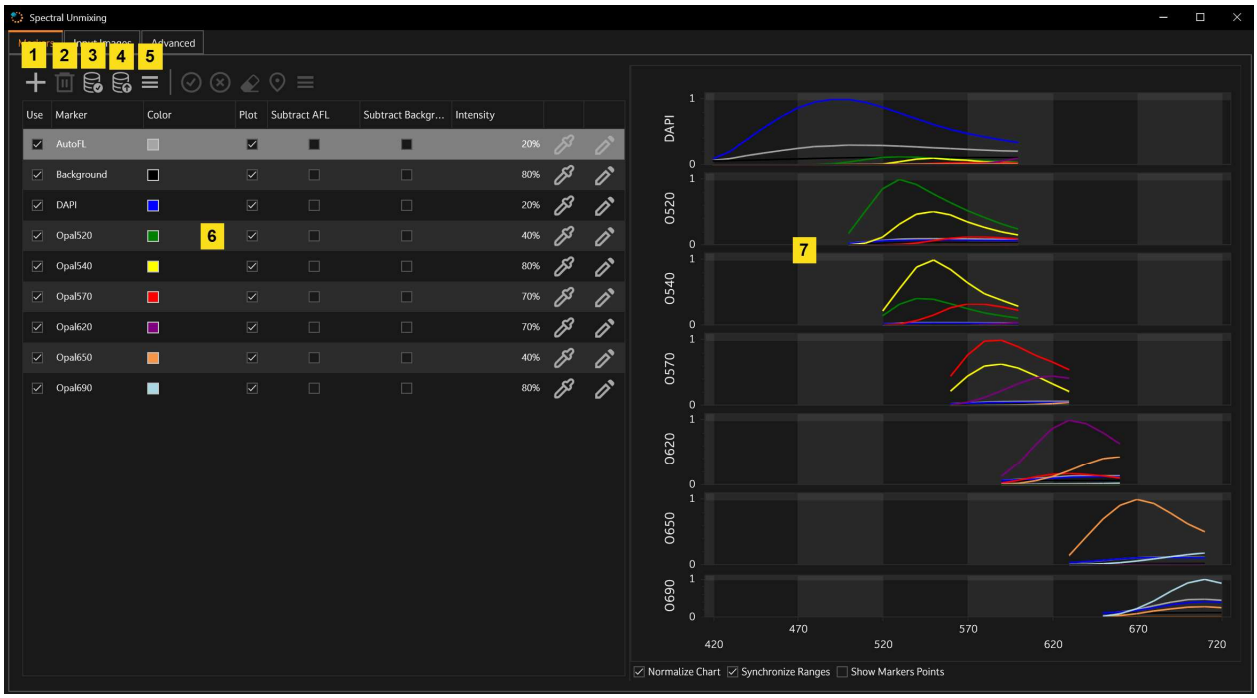


Figure 698 – Spectral Unmixing: how to configure

The first tab of the Spectral Unmixing engine, called Markers, offers a set of tools for the management of reference markers used in the unmixing process:

1. Add Marker - adds a new reference marker to the list
2. Delete Selected Marker - removes selected marker from the list
3. Save Markers to Spectral Database - saves marker(s) into the Spectral Database
4. Load Markers from Spectral Database - loads marker(s) from the Spectral Database
5. More (View Colors, Delete All markers, Help)
1. Remove All Markers - removes all markers from the list
2. View Markers Values - opens a new window showing all values of the markers present in the list
3.
4. Makers List - all reference markers used in the unmixing process
5. Charts - plots for all reference markers spectra available in Markers List

Knowing the makers present in the sample, the reference markers list can be created by defining a new reference marker or loading one from the spectral database. Each marker has associated tools for definition or setting changes:

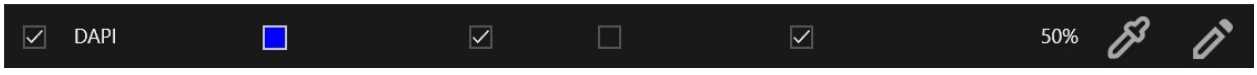
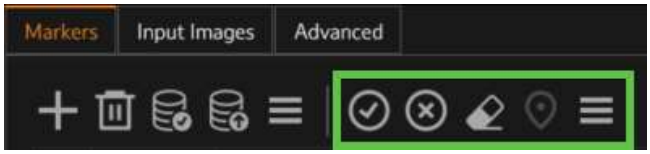


Figure 699 – Spectral Unmixing: Marker Tools

- Marker - the name of the reference marker. This can be changed by double clicking on it.
- Color - associates a color to the reference marker.
- Plot - enables / disables the plot of the current marker in the plots window.
- Subtract AFL - enables / disables the subtraction of an auto-fluorescence spectrum from the current marker's spectrum (when collecting marker's values from images, the result is a mixture of pure marker and auto-fluorescence).
- Subtract Background - enables / disables the subtraction of background spectrum from the current marker's spectrum (available only if the Remove High Background option is enabled from the Advanced tab).
- Intensity - the intensity of the unmixed results.
- Color Picker - defines marker's spectrum by single pixel selection. The position (x, y) of the selected pixel is used to collect values from all the spectral images (for all wavelengths).
- Area Selection - defines marker's spectrum by selecting a set of pixels. The positions (x, y) of all selected pixels are used to collect values from all the spectral images (for all wavelengths). The final values of the spectrum are given by averaging the collected values for each wavelength.

Color Picker and Area Selection are tools used to define the reference markers spectra. While Color Picker is a single pixel selection, Area Selection generates annotations (can have multiple annotations for a single maker), which requires additional editing tools:



- Apply Area Selection - finishes the edit mode and accept all modifications
- Cancel - finishes the edit mode and discard all modifications
-
- Start Erase Areas - enables / disables erase mode
- Locate annotation - locates annotations (locates and centres each annotation in the view window).
- Delete All Areas for Selected Marker - removes all annotations for current reference marker -
- Delete All Areas - removes all annotations for all reference markers

After a reference marker spectrum is defined, it is plotted in the Charts window. Each filter has its own plot window, where:

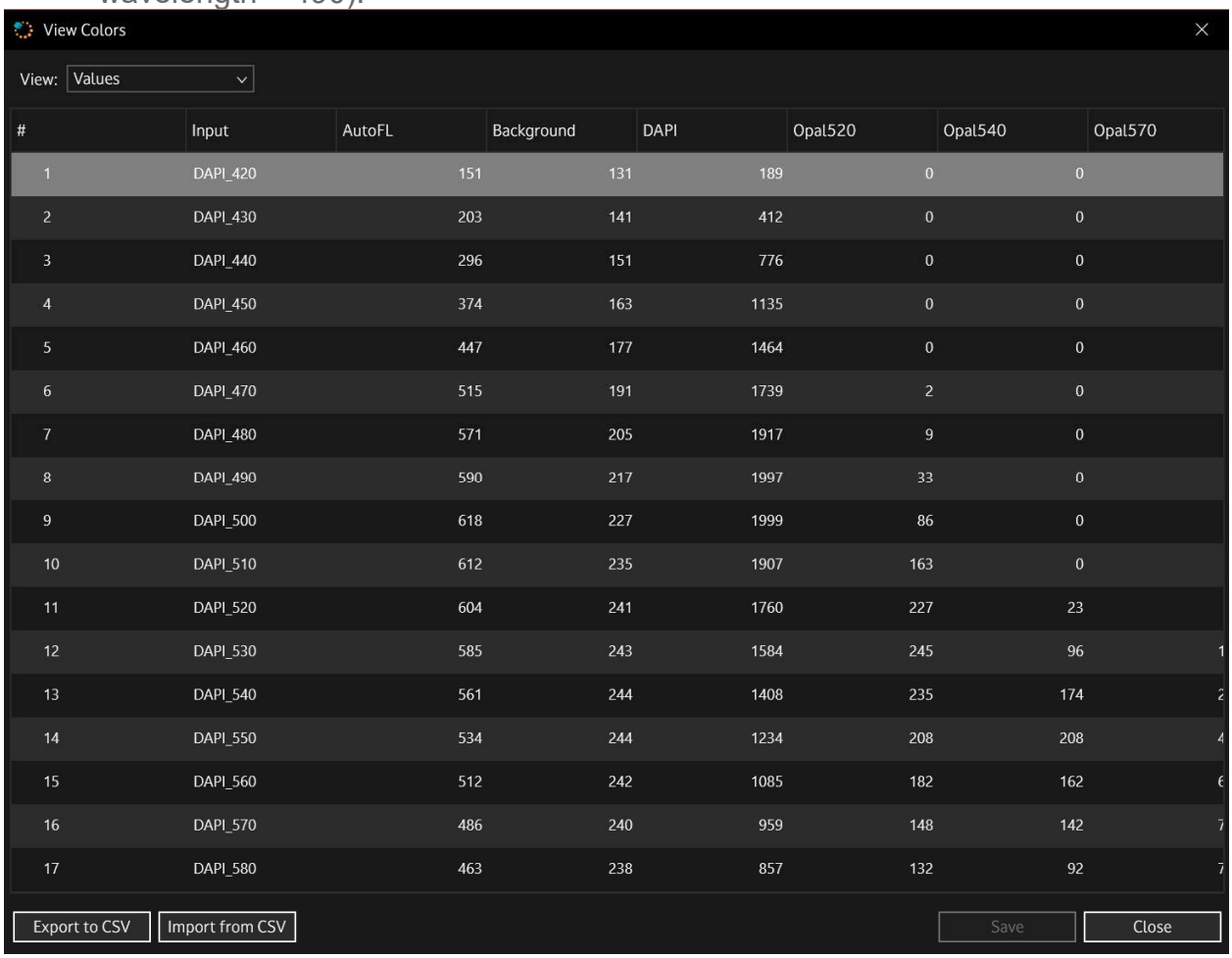
- the wavelength range is on the X axis
- the intensity of the signal on the Y axis

The plots options are:

- Normalize Chart - the spectra values are normalized using the setting from the Advanced tab
- Synchronize Ranges - all charts use the same range for the Y axis
- Show Markers Points - enables / disables bullet display for markers values (the plots are continuous curved lines -interpolation of markers values).

The values of the reference markers spectra plotted in the charts can be inspected by using the View Colors button. In the newly opened window, the reference markers values are displayed in a form of a table, where:

- Columns - reference makers;
- Rows - filter and wavelength combination (e.g. DAPI_490 → filter = DAPI, wavelength = 490).



#	Input	AutoFL	Background	DAPI	Opal520	Opal540	Opal570
1	DAPI_420	151	131	189	0	0	
2	DAPI_430	203	141	412	0	0	
3	DAPI_440	296	151	776	0	0	
4	DAPI_450	374	163	1135	0	0	
5	DAPI_460	447	177	1464	0	0	
6	DAPI_470	515	191	1739	2	0	
7	DAPI_480	571	205	1917	9	0	
8	DAPI_490	590	217	1997	33	0	
9	DAPI_500	618	227	1999	86	0	
10	DAPI_510	612	235	1907	163	0	
11	DAPI_520	604	241	1760	227	23	
12	DAPI_530	585	243	1584	245	96	1
13	DAPI_540	561	244	1408	235	174	2
14	DAPI_550	534	244	1234	208	208	4
15	DAPI_560	512	242	1085	182	162	6
16	DAPI_570	486	240	959	148	142	7
17	DAPI_580	463	238	857	132	92	7

Figure 700 – Spectral Unmixing: View Colors

All values can be edited by clicking on the corresponding cell and inserting the new value.

By default, the intensity Values are displayed. It is possible to visualize the Exposure Times or Lamp Intensities values by selecting it from the View drop down list.

All values displayed in the window can be exported to or imported from a CSV file using the buttons Export to CSV and Import from CSV.

Save to Spectral Database

Once a marker is defined, it can be saved into the spectral database for further use, by using the Save Markers to Spectral Database button. A new window opens with the following options:

- Marker set name - names the new database entry
- Description - adds a description to the new database entry
- Selection - selects marker(s) defining the new database entry
- Save cleaned marker(s) - enables / disables marker(s) cleaning by subtracting auto-fluorescence and background (if used)

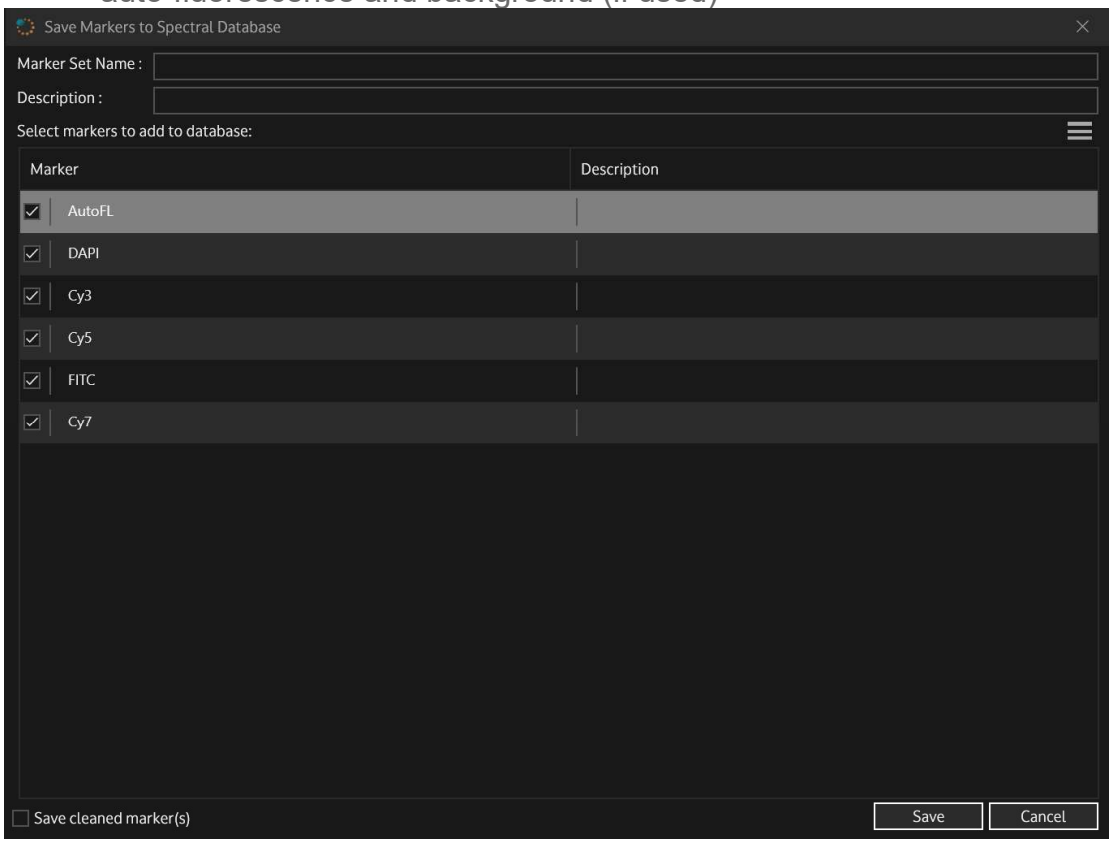


Figure 701 – Spectral Unmixing: Save markers to spectral database

Load from Spectral Database

Previously saved markers or predefined standard markers are available for loading from the spectral database for usage. Hoovering the mouse on the marker's name will open a small window showing the marker's spectrum.

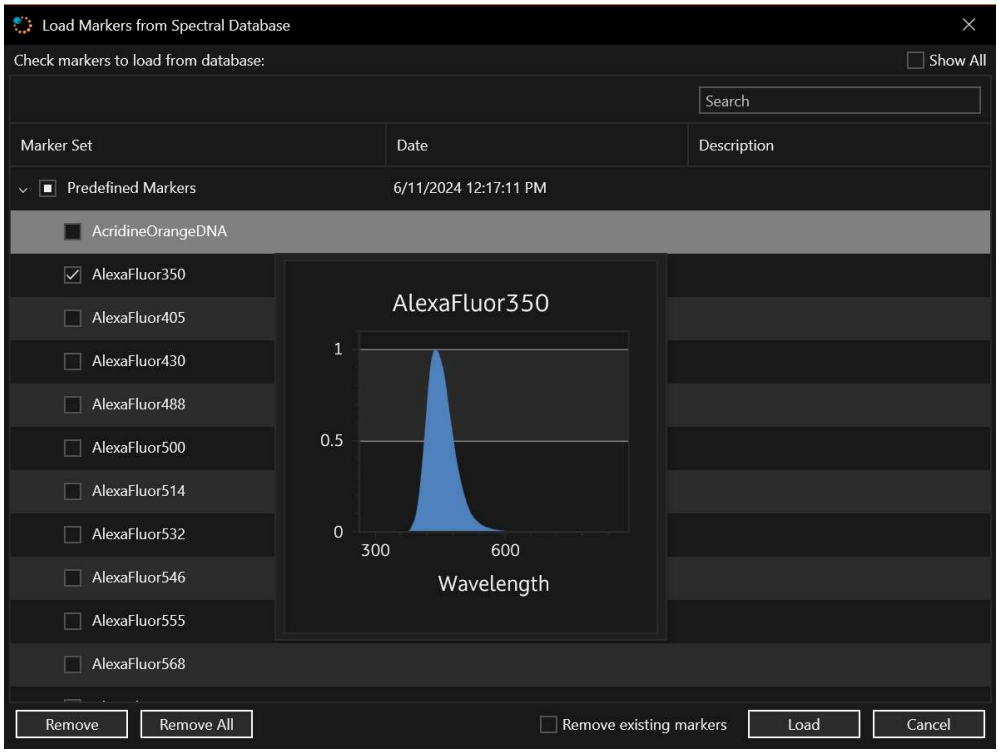


Figure 702 – Choosing markers to load from database

StrataQuest offers a comprehensive list of predefined standard markers covering the most used ones. The search option on the top right side of the window makes the selection easier.

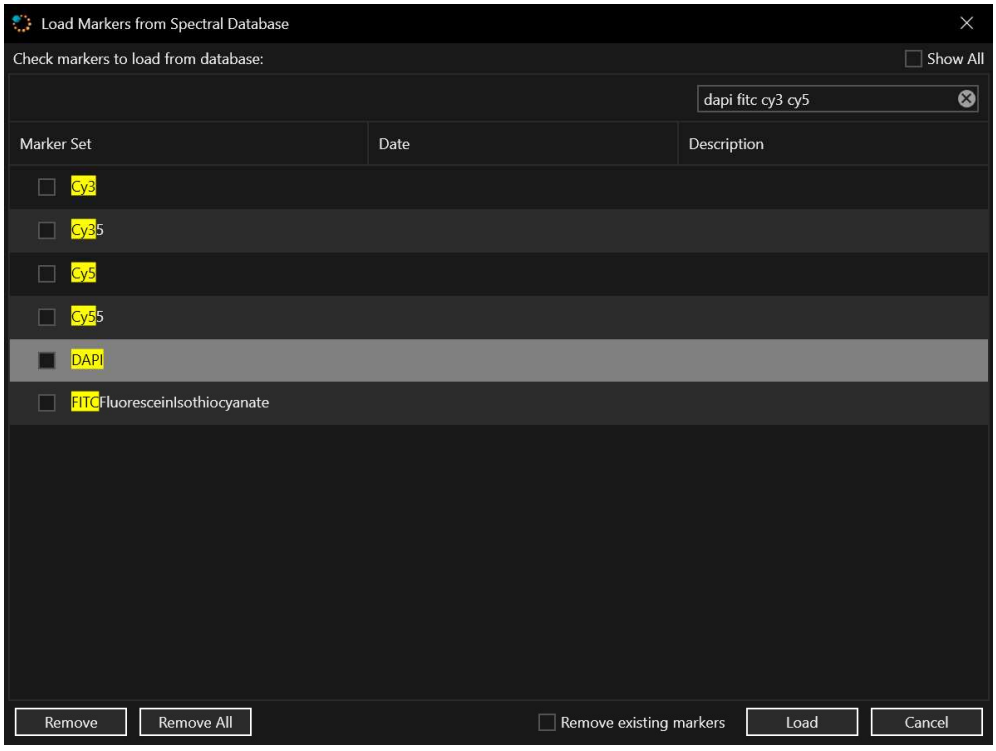


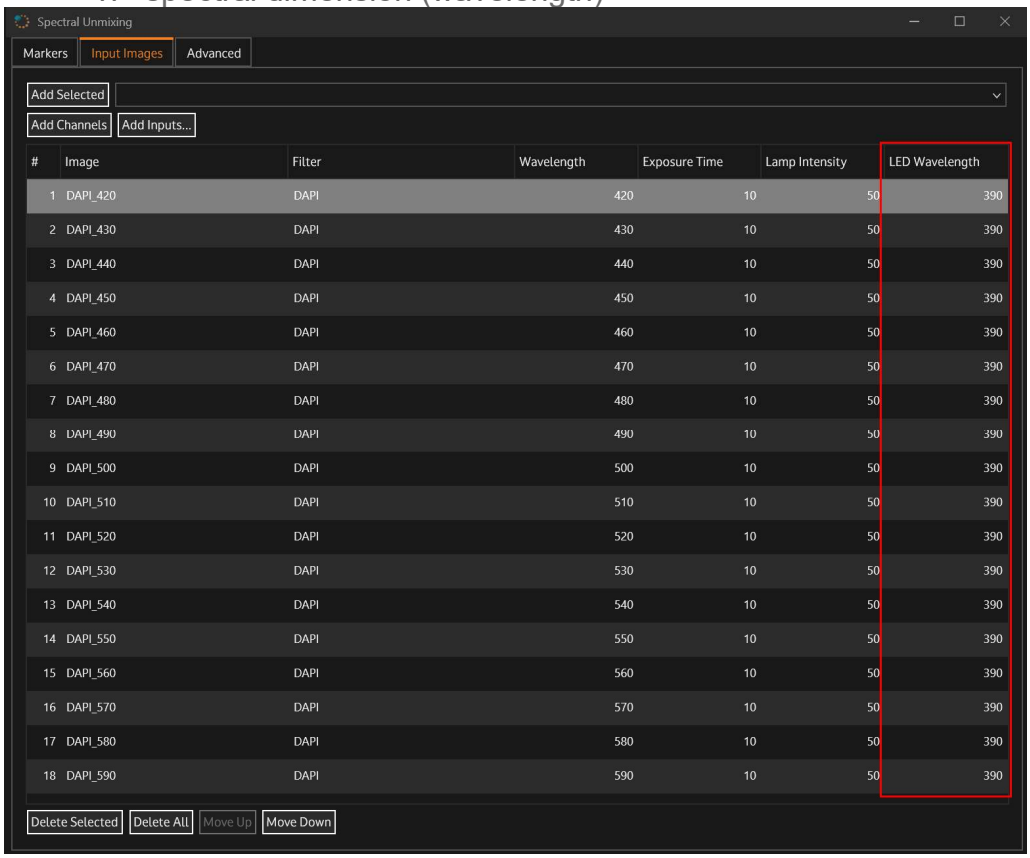
Figure 703 – Predefined markers

Once the selection is made, the loading process is finalized by pressing the Load button.

Manage spectral images

The second tab of the Spectral Unmixing engine, called Input Images, allows the selection of images used as input in the unmixing process. Each image represents a narrow wavelength range of the visible range (of the electromagnetic spectrum), also known as a spectral band. These images are combined to form a three-dimensional (x, y, λ) multispectral data cube for processing and analysis, where:

- x and y - spatial dimensions of the scene
- λ - spectral dimension (wavelength)



#	Image	Filter	Wavelength	Exposure Time	Lamp Intensity	LED Wavelength
1	DAPI_420	DAPI	420	10	50	390
2	DAPI_430	DAPI	430	10	50	390
3	DAPI_440	DAPI	440	10	50	390
4	DAPI_450	DAPI	450	10	50	390
5	DAPI_460	DAPI	460	10	50	390
6	DAPI_470	DAPI	470	10	50	390
7	DAPI_480	DAPI	480	10	50	390
8	DAPI_490	DAPI	490	10	50	390
9	DAPI_500	DAPI	500	10	50	390
10	DAPI_510	DAPI	510	10	50	390
11	DAPI_520	DAPI	520	10	50	390
12	DAPI_530	DAPI	530	10	50	390
13	DAPI_540	DAPI	540	10	50	390
14	DAPI_550	DAPI	550	10	50	390
15	DAPI_560	DAPI	560	10	50	390
16	DAPI_570	DAPI	570	10	50	390
17	DAPI_580	DAPI	580	10	50	390
18	DAPI_590	DAPI	590	10	50	390

Figure 704 – Spectral Unmixing: input images

By default, the cube is comprised of all the spectral bands acquired, but can be modified with the help of the following tools:

- Add Selected - selects an image (from the available images) to the cube;
- Add Channels - adds all original images to the cube;
- Add Inputs - adds virtual channels (images generated by other engines, e.g. the outputs of a median filter applied to the original spectral bands) to the cube;
- Cube items - displays all images defining the cube in a list, with acquisition information:
    - Index - the image index in the list;

  - Image - the name of the image;
  - Filter - the filter used to acquire the image;
  - Wavelength - the wavelength used to acquire the image;
  - Exposure Time - the camera exposure time used to acquire the image;
  - Lamp Intensity - the light intensity used to acquire the image.
  - LED Wavelength - the light wavelength used to acquire the image.
- Remove Selected - removes the selected image from the cube;
- Remove All - removes all images from the cube;
- Move Up - moves the selected image up in the list;
- Move Down - moves the selected image down in the list;

Note: If the project has been acquired using TissueFAXS, the values for Exposure Time, Lamp Intensity and LED Wavelength are automatically set with the corresponding values used in the acquisition process.

Advanced settings

The third tab of the Spectral Unmixing engine, called Advanced, offers additional settings for the unmixing process:

- Use AutoFL Marker - enables / disables the auto-fluorescence marker as part of the reference markers list. E.g. in brightfield microscopy there is no auto-fluorescence.
- Remove High Background - enables / disables the background subtraction from the multispectral cube and reference markers. In case of 16-bit FL images, the background is almost never pure black; it may be considered a random noise (values in the range of 100...300). Because any value higher than 0 is interpreted as a mixed signal, the background will be unmixed into reference components. This can be avoided by defining the background spectrum and subtracting it from the images, thus lowering the background as close as possible to 0.

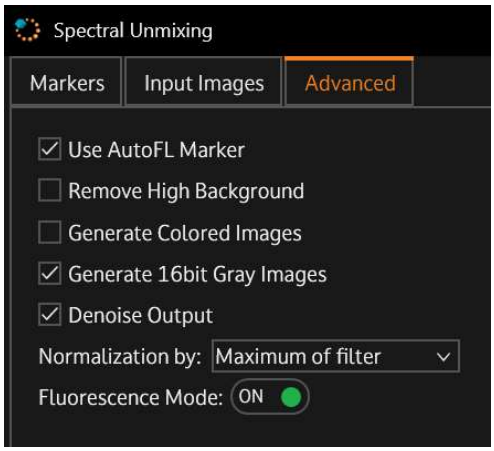


Figure 705 – Spectral Unmixing: Advanced Tab

- Generate Colored Images - enables / disables the colored unmixed results. By default, only grayscale image is generated for each reference marker. When

enabled, a color image will also be generated, using the color associated with the marker.

- Generate 16-bit gray images - enables / disables the 16-bit mode for the output grayscale images. By default, the unmixed markers results are generated as 8-bit images.
- Denoise output - enables / disables the usage of a 3x3 median filter on the unmixed results.
- Normalization by - changes the way normalization is performed for reference spectra (only for display purposes - plots):
    - Maximum of filter - all reference markers components associated to a filter are normalized using the maximum of all those components of all markers (e.g. the maximum of all values of all markers associated with DAPI filter).
    - Maximum of marker - all reference markers components are normalized using their maximum.

Effect:

Unmixes the signal present in the multi-spectral images.

Example:

The Multi-Spectral Unmixing Engine will generate a gray image for each defined marker representing the quantification of that marker. Everything is scaled around the color picked by the user. Those values define the marker quantity associated with the gray coefficient selected by the user. For example, assume the user selected as blue marker, a pixel with a spectrum (25, 25, 225, 200, 175, 25, 25 …) and associated a gray coefficient of 200. This means that the quantity of marker visible as (25, 25, 225, 200, 175, 25, 25, …) is displayed on the unmixed gray image as 200 and all other positions are scaled around this definition: (25, 25, 225, 200, 175, 25, 25, …) = 200.

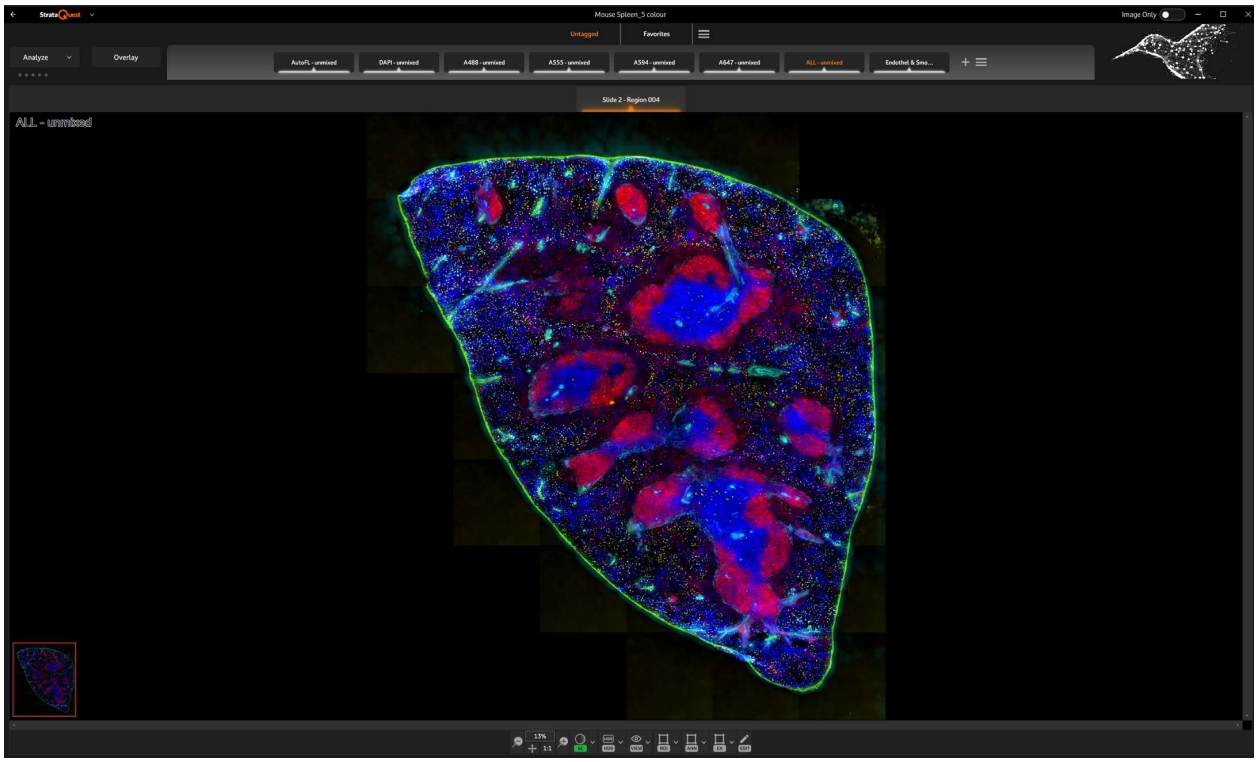


Figure 706 – Spectral Unmixing output

37. Standard Measurements

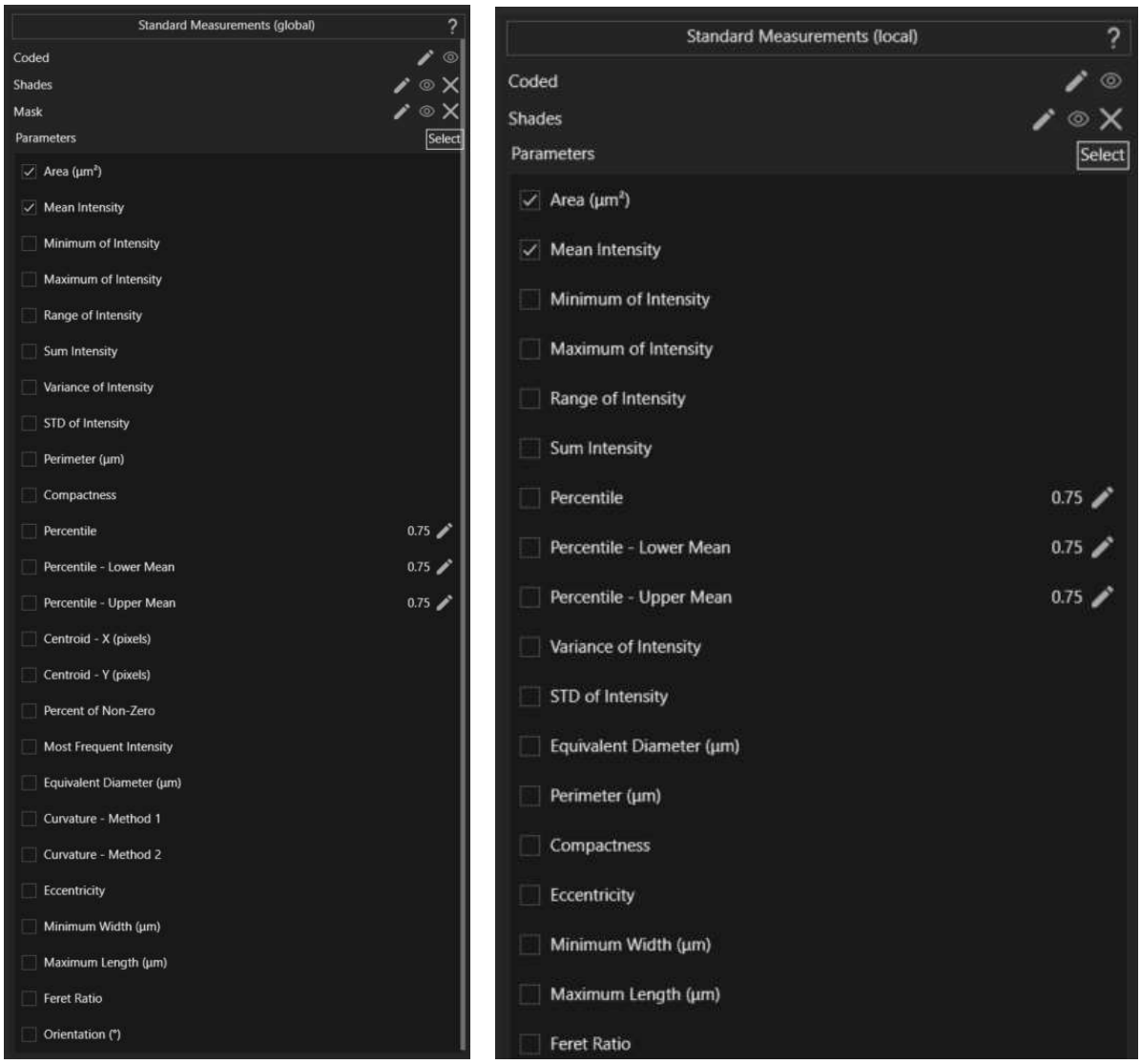


Figure 707 – Standard Measurements

Where it can be found:

Layer's "Measurements" ► Add ► Measurements (basic) ► Standard Measurements (global)

Layer's "Measurements" ► Add ► Measurements (advanced) ► Standard Measurements (local)

Description:

Standard measurements is an engine available in the Measurements section of the layers editor.

For each detected event this engine computes the measurements selected from the list.

Two versions of the engine are available:

- Local - computes measurements for local events (without any events stitching information). To use the local version of the engine, Stitch objects must be disabled in the layer's option window
- Global - computes measurements for global events (using events stitching information). To use the global version of the engine, Stitch objects must be enabled (it is by default) in the layer's option window

A Local Event is considered an event detected in a Field of View (FOV). Multiple Local Events located at neighboring FOVs border (touching border), representing a structure positioned at the intersection of two or more FOVs, are combined (stitched) into a Global Event, for a better representation of the real information present in the sample or ROI.

Parameters:

- Coded - inputs a coded image representing the set of events for which the measurements will be computed. This image is used to compute all morphological (shape-based) measurements.
- Shades - inputs a grayscale image. This image is used to compute all intensity-based measurements. Can represent a marker expression or can be a virtual channel generated by another engine which has a specific meaning in the analysis workflow (e.g. a distance transform image used to find the closest specific structure).
- Mask (global only) – inputs a mask image used to compute measurements only in the indicated areas (white)

List of measurements available for selection in the global version:

- Morphological measurements:
  - Area - the area of an event
  - Perimeter - the perimeter of an event
  - Compactness - measures how much the segmented object resembles a disk. The more likely it resembles a disk, the expressed compactness is closer to 1. If not, its value will be closer to 0
  - Centroid -X - the X position of event's centroid
  - Centroid -Y - the Y position of event's centroid
  - Equivalent diameter - the diameter of a circle having the same area as the event
  - Curvature (v1) - the curvature of the event using method 1 (faster, lesser accuracy)
  - Curvature (v2) - the curvature of the event using method 2 (slower, greater accuracy)

  - Eccentricity - measures how much the object shape resembles a circle or a line segment. When it is closer to 0, the object resembles a circle, when it is closer to 1, the object looks more elongated
  - Minimum Width - minimum value a micrometer would express, considering all possible measurements positions (orientations)
  - Maximum Length - maximum value a micrometer would express, considering all possible measurements positions (orientations)
  - Feret Ratio - the ration between minimum width and maximum width
  - Orientation - the angle between maximum length of the object and horizontal axis, in anti clock-wise direction
- Intensity-based measurements:
  - Mean - the average intensity of event's pixels
  - Minimum - the minimum intensity of event's pixels
  - Maximum - the maximum intensity of event's pixels
  - Range - the intensity range size of event's pixels (max - min)
  - Sum - the intensity sum of all event's pixels
  - Variance - the variance of event's pixels
  - Standard deviation - the standard deviation of event's pixels
  - Percentile - the intensity value of the user defined percentile
  - Lower mean - the average intensity of all event's pixels having a value lower than percentile value
  - Upper mean - the average intensity of all event's pixels having a value higher than percentile value

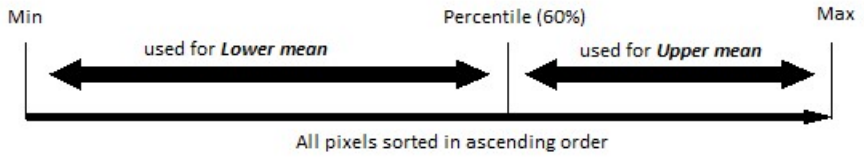


Figure 708 – Illustration of pixels used to compute Lower mean and Upper mean

  - Percent of non-zero - the percent of event's pixels having value higher than 0
  - Most frequent value - the intensity value having the highest appearance in event's pixel

List of measurements available for selection in the local version:

- Morphological measurements:
  - Area - the area of an event
  - Perimeter - the perimeter of an event
  - Compactness - measures how much the segmented object resembles a disk. The more likely it resembles a disk, the expressed compactness is closer to 1. If not, its value will be closer to 0
  - Equivalent diameter - the diameter of a circle having the same area as the event

- Eccentricity - measures how much the object shape resembles a circle or a line segment. When it is closer to 0, the object resembles a circle, when it is closer to 1, the object looks more elongated
- Minimum caliper diameter - the minimum event size as it would be measured with a caliper
- Maximum caliper diameter - the maximum event size as it would be measured with a caliper
- Feret ratio - the ratio between the minimum and maximum caliper diameters

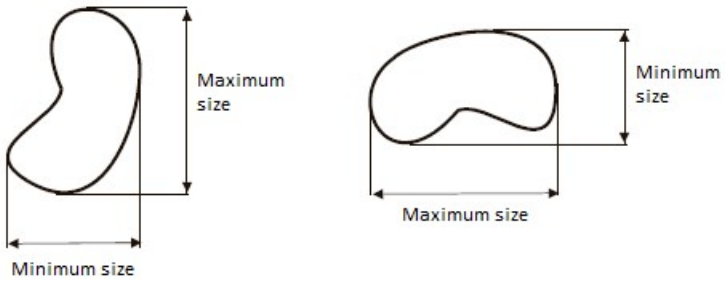


Figure 709 – Illustration of caliper sizes

- Intensity-based measurements:
  - Mean - the average intensity of an event's pixels
  - Minimum - the minimum intensity of an event's pixels
  - Maximum - the maximum intensity of an event's pixels
  - Range - the intensity range size of an event's pixels (max - min)
  - Sum - the intensity sum of all event's pixels
  - Variance - the variance of an event's pixels
  - Standard deviation - the standard deviation of an event's pixels
  - Percentile - the intensity value of the user defined percentile
  - Lower mean - the average intensity of all event's pixels having a value lower than percentile value
  - Upper mean - the average intensity of all event's pixels having a value higher than percentile value

The measurements are computed based on the list selection, which can be done manually or by using a pre-defined selection from the Select button:

- Default - the default selection: Area, Mean Intensity

Morphological - all morphological measurements selection:

- for global version: Area, Perimeter, Compactness, Centroid -X, Centroid -Y, Equivalent diameter, Curvature (v1), Curvature (v2);

- for local version: Area, Perimeter, Compactness, Equivalent diameter, Eccentricity, Minimum caliper diameter, Maximum caliper diameter, and Feret ratio.

Intensity - all intensity-based measurements selection:

- for global version: Mean, Minimum, Maximum, Range, Sum, Variance, Standard deviation, Percentile, Mean (lower than percentile), Mean (higher than percentile), Percent of non-zero, Most frequent value;
- for local version: Mean, Minimum, Maximum, Range, Sum, Variance, Standard deviation, Percentile, Mean (lower than percentile), Mean (higher than percentile).

Keep current selection - enables / disables the possibility to add a selection of parameters to the one already existing, without canceling the first one

Select All - selects all measurements

Clear selection - discards the current selection

- **Effect**:

Computes selected measurements for a set of detected events.

Example:

- Input:

The inputs are represented by:

- A set of events – detected structures on an IHC colon sample (highlighted in green)
- A grayscale image representing the ki67 marker generated by the Color separation engine

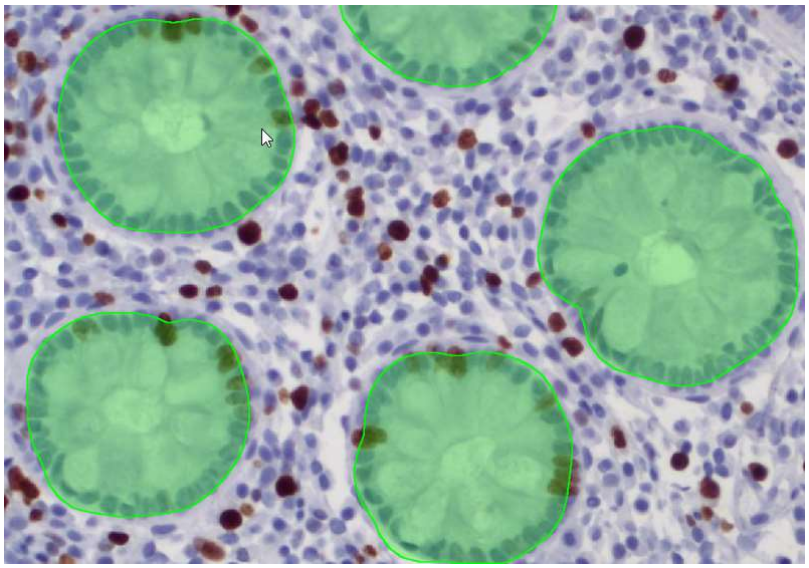


Figure 710 – Detected structures on an IHC colon sample

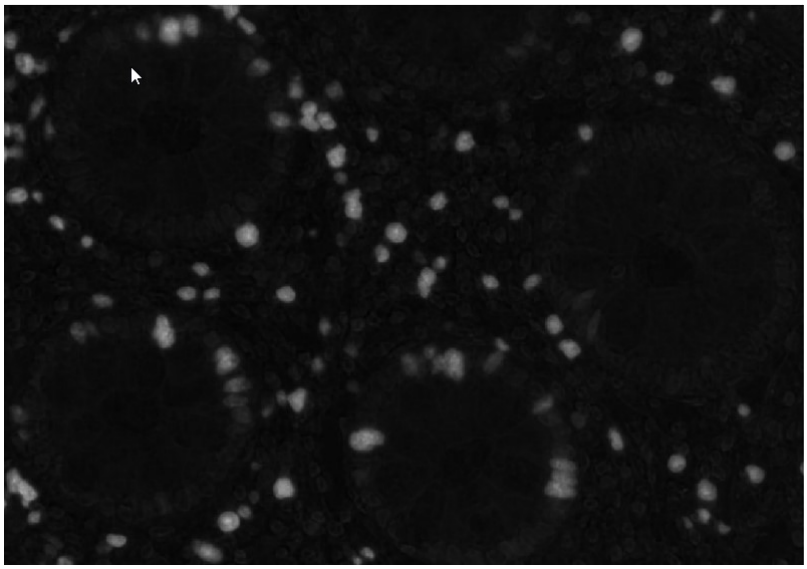


Figure 711 – Ki67 virtual channel

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a set of measurements associated with the input events.

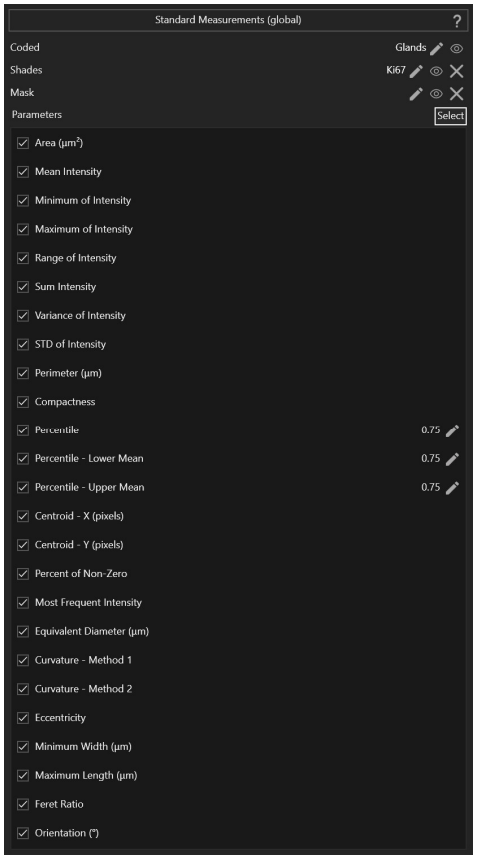


Figure 712 – Engine settings

Output:

Figure below shows the set of measurements computed for the selected event.

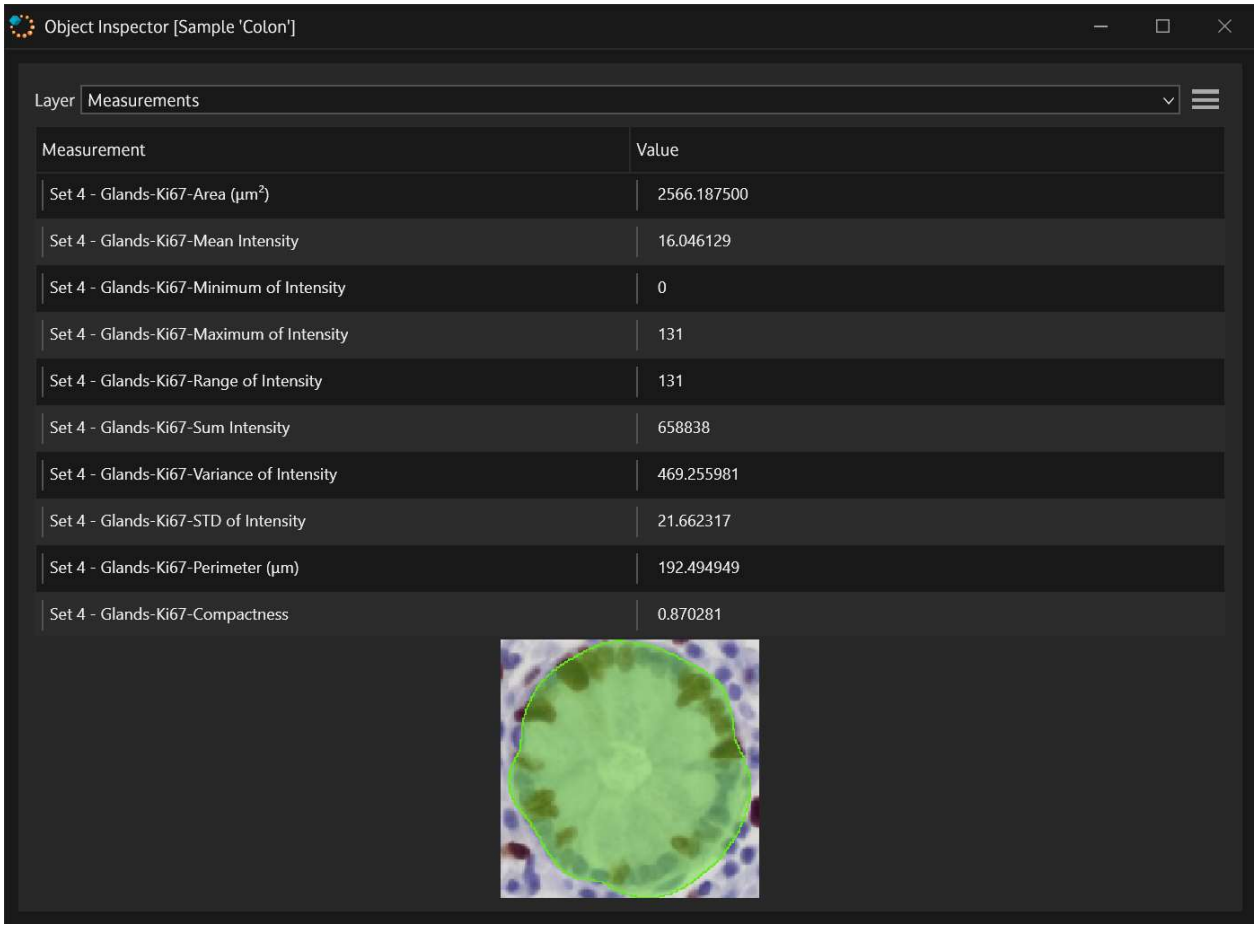


Figure 713 – Result of the Standard Measurements engine shown for a particular event

38. Virtual Channel

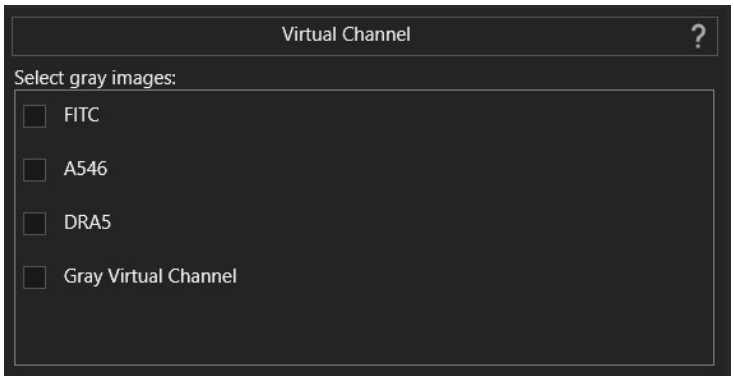


Figure 714 – Virtual Channel

Where it can be found:

Layer's "Pre-Processing" ► Add ► Operations (basic) ► Virtual Channel

Description:

Virtual Channel is an engine available in the Pre-Processing section of the layers editor.

This engine generates a virtual channel by combining the information from the selected grayscale images. The input images can be the original channels (FL) or outputs from other engines and must be available in the current layer (grayscale images generated in other layers must be imported in the current layer before they can be used).

Selected images must have the same type.

You can select only one image and it will output that image without any processing.

Parameters:

- Select gray images - list of grayscale images available in the current layer

Effect:

Generates a virtual channel by combining the information of selected grayscale images.

Example:

- Input:

The input consists of 2 grayscale images representing:

- the nuclear marker channel
- the membrane marker channel

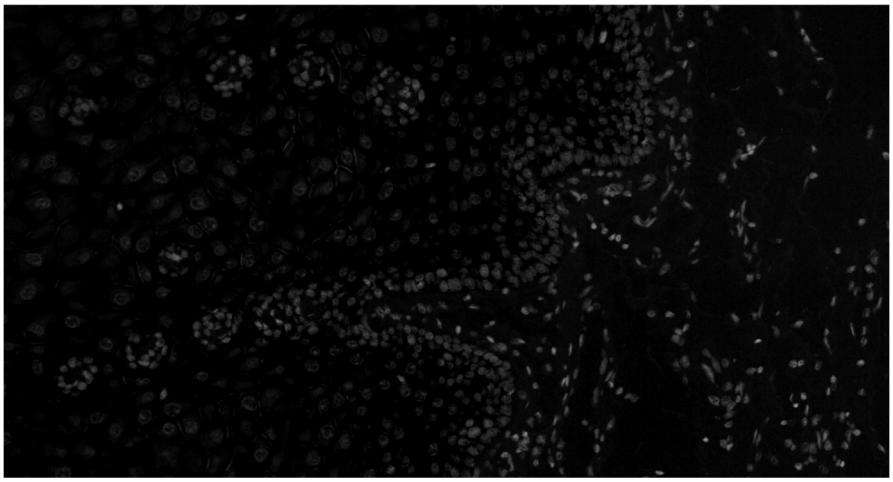


Figure 715 – Nuclear marker channel

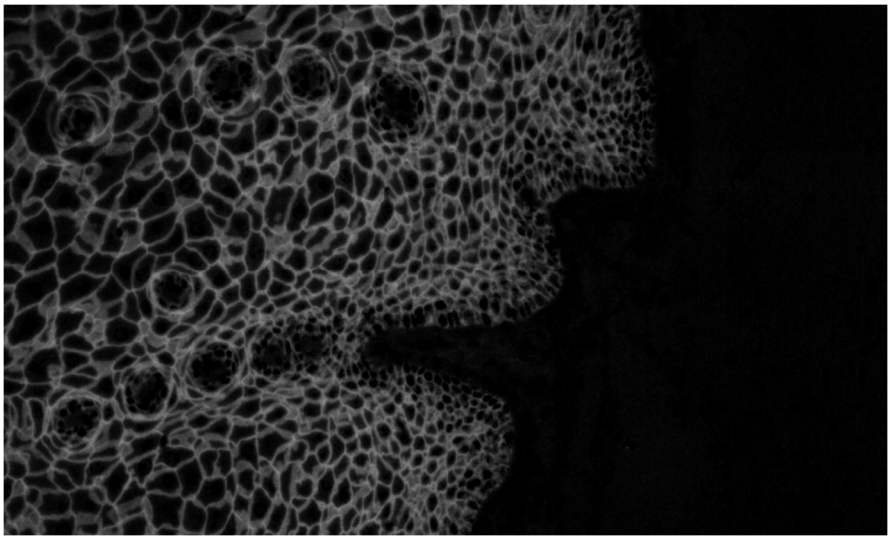


Figure 716 – Membrane marker channel

- Engine settings:

Figure below shows the engine settings used for this example.

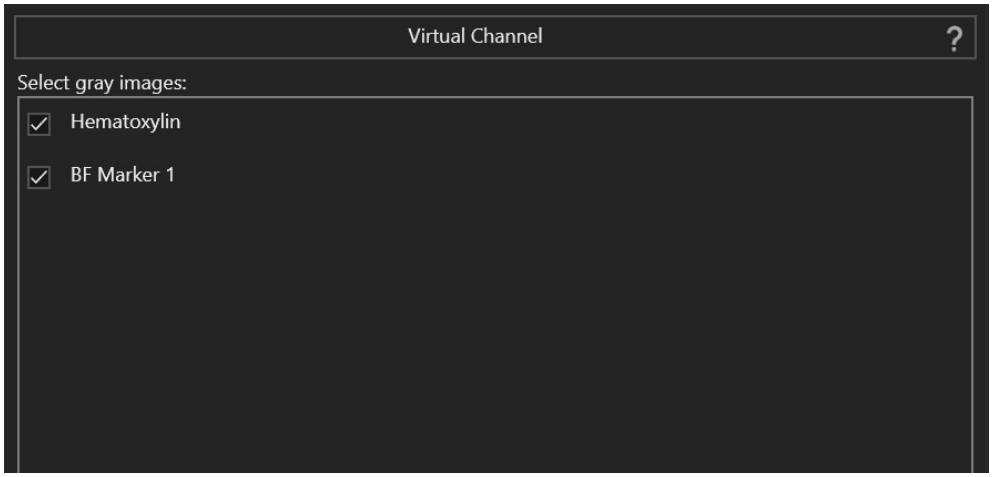


Figure 717 – Engine settings

Output:

Figure below shows the result of the Virtual Chanel engine, representing the combined information from the 2 input images.

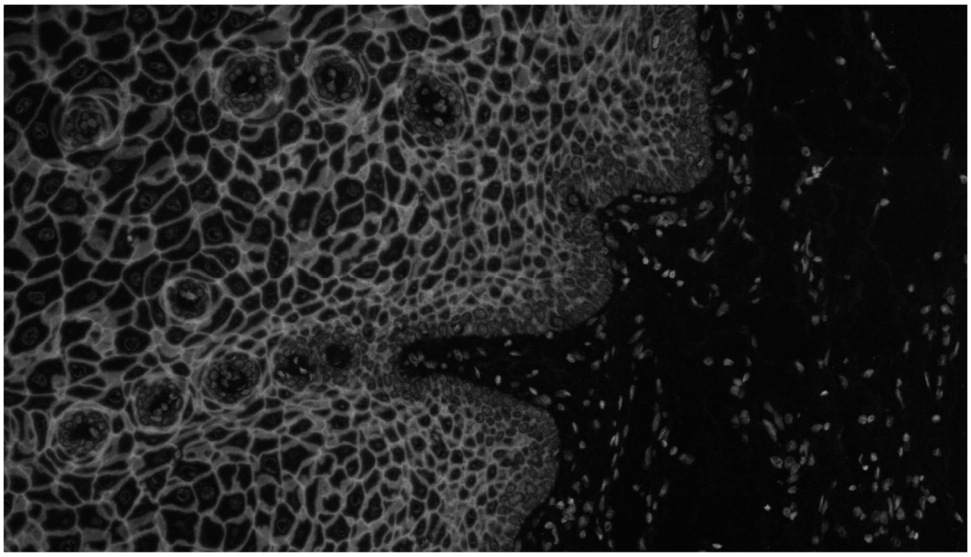


Figure 718 – Result of Virtual Channel engine

39. Watershed (global)

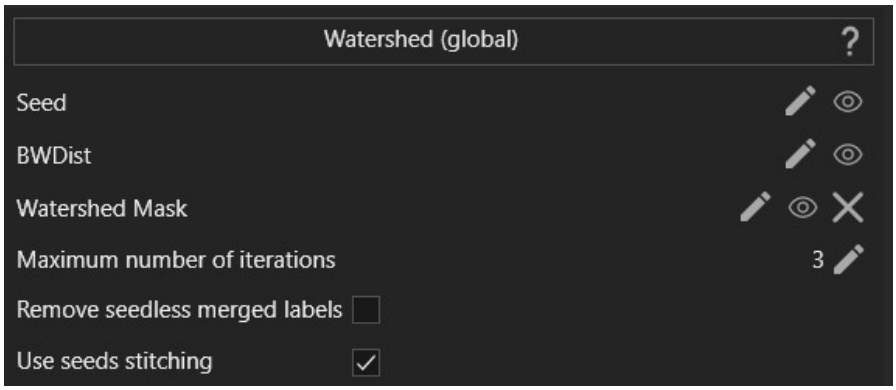


Figure 719 – Watershed

Where it can be found:

Layer's "Detection" ► Add ► Operations (advanced) ► Watershed (global)

Description:

Watershed (global) is an engine available in the Detection section of the layers editor.

This engine performs image segmentation using the watershed algorithm. The algorithm grows the seeds with priorities defined by the information present in the image it operates upon (the image is interpreted like a topographic map, where the intensity of each pixel can be associated with its height).

Non-zero pixels from the Seeds image represent the water sources, which flood the entire relief, represented by the BWDist, and build barriers when different water sources meet. The resulting set of barriers constitutes the watershed segmentation result.

The engine performs a global watershed segmentation, which means the continuity of information from FOV to FOV is considered. That's why the engine requires a number of iterations to assure an accurate segmentation.

Parameters:

- Seeds - inputs an image representing the starting points of the flooding
- Topographic Map - inputs an image representing the topographic map used to prioritize flooding
- Mask - applies the watershed transformation only for indicated areas (white)
- Maximum iterations - the maximum number of iterations
- Remove seedless merged labels - enables / disables the removal of resulting events (labels) which are not connected to a seed (water source)

- Stitch seeds - enables / disables stitching for seeds (only for seeds with continuity from one FOV to another)

Effect:

Segments contiguous regions of interest into distinct events (objects), using the watershed transformation.

Example:

- Input:

Figure below shows a brightfield fat cells sample, which is used to extract the information needed for watershed segmentation:

- seeds image – generated by identifying the fat cells bodies (white areas)
- topographic image – generated by Distance Map engine having the membranes as reference
- growing mask – for this example, the growing is allowed everywhere (entire image), meaning a full white binary mask is used.

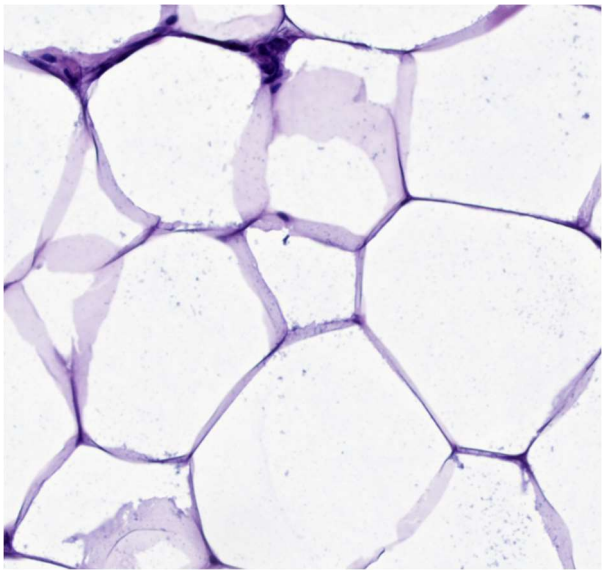


Figure 720 – Fat cells

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a coded image representing the segmented events using the watershed algorithm.

Figure 721 – Engine settings

Output:

Figure below shows the resulting events using the watershed engine.

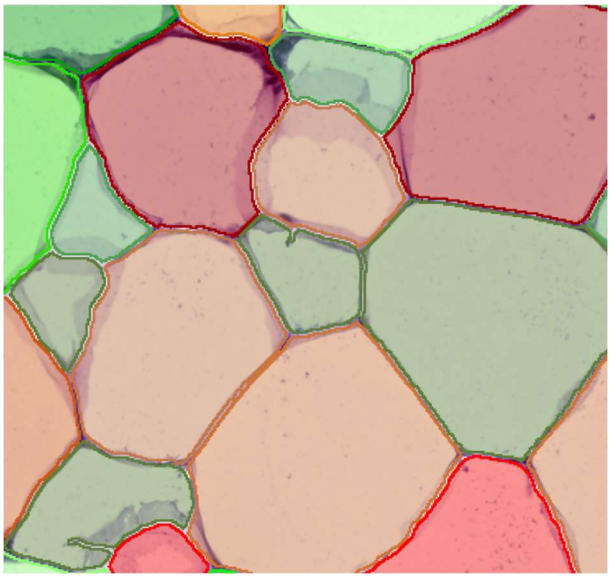


Figure 722 – Result of Watershed (global) engine

40. Tissue Detection

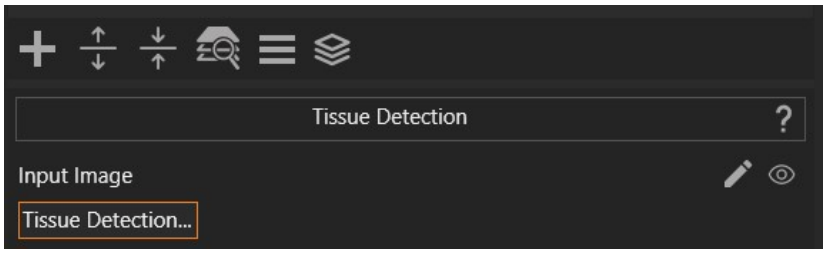


Figure 723 – Tissue Detection

Where it can be found:

Layer's "Pre-Processing" ► Add ► Detection ► Tissue Detection

Description:

Tissue detection is an engine available in the Pre-Processing section of the layers editor.

This engine performs tissue detection on the preview image of the input sample or region of interest. The preview image is the top level of the Cache (a pyramid structure of the sample's images, each pyramid level at a different zoom level). The cache must be built prior to tissue detection.

The Tissue Detection result is a mask image representing the tissue (white).

The general usage of the Tissue Detection engine can be summarized in the next steps:

1. select the input image and open the Tissue Detection window (build cache if necessary);
2. adjust settings if the default ones are not optimal for a good detection:
    a. make sure the detection is as permissive as possible:

- Minimum Tissue Area - the smallest value possible to visualize all small detected tissue areas
- Tissue Closing Radius - the smallest value possible to visualize the fragmentation of the detection. A high fragmentation means the threshold value is too high
- Threshold - disables the Automatic mode and changes to a low value.
    b. Increase the threshold value until the detection of tissue area(s) is satisfactory.
    c. Increase the Tissue Closing Radius (in small steps) to overcome possible fragmentation of detected tissue area(s)
    d. Increase the Minimum Tissue Area to remove small and irrelevant detected areas

Parameters:

- Input image - inputs an image for tissue detection

Pressing the Tissue Detection button once the input image is set will open the Tissue Detection window.

Multiple settings tabs are available to optimize tissue detection:

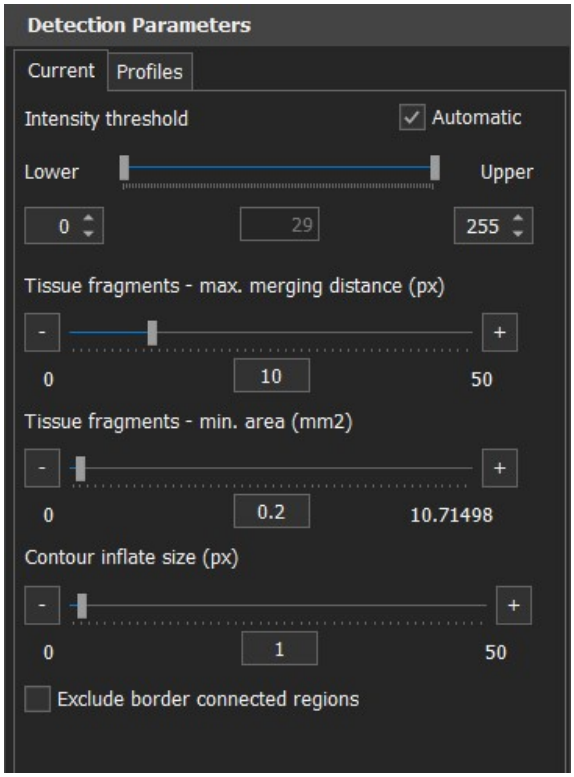


Figure 724 – Detection parameters: Tissue

- Automatic: If Automatic is checked, the Intensity Threshold will be automatically computed on the gray image in the range selected by the user. The determined value will be shown in the edit box in the middle. A value can be entered by disabling Automatic and inputting the desired value in the edit box. Use the Gray indicator on the gray image to determine the pixel intensity in the image.
- Intensity Threshold: sets the threshold for intensity;
- Tissue Fragments – max. merging distance: try to increase this parameter if the region is split into multiple small subregions. The result will be a single region containing smaller subregions;
- Tissue Fragments – min. area: is the smallest area a tissue region can have. All regions with a smaller area will not be considered;
- Contour Inflate Size: the initially detected region will be inflated with a specified number of pixels, in order to include the edges of the tissue and a small surrounding area;

- Exclude border connected regions: regions touching the border are excluded from detection

Detection Profiles

For an easier workflow, you can create detection profiles that work like detection templates.

Profile options:

- Default: allows the current set of parameter values to be stored for future detection.
- Restore Defaults: restores all the parameters to the last saved default parameters.
- Profile: saves the current parameter settings for future use.
- Load Profile: loads a previously saved profile from the existing list (by double clicking on it).

Example:

- Input:

Figure below shows a fluorescence lymphatic tissue sample.

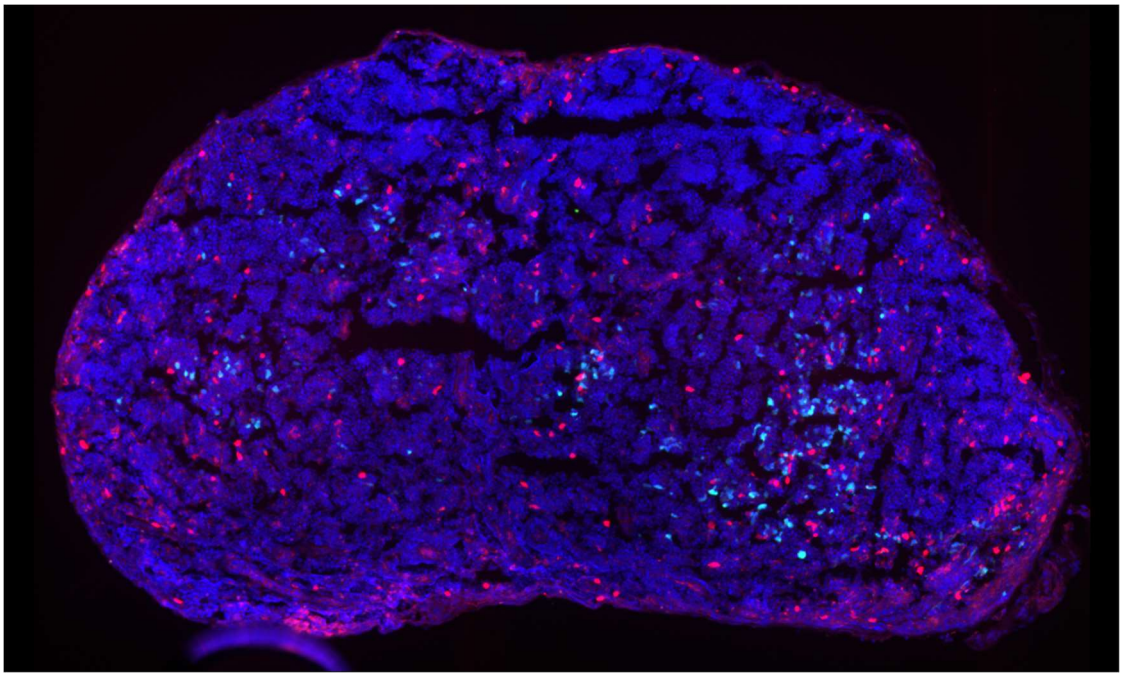


Figure 727 – Lymphatic tissue sample

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a binary mask representing the presence (white) / absence (black) of tissue.

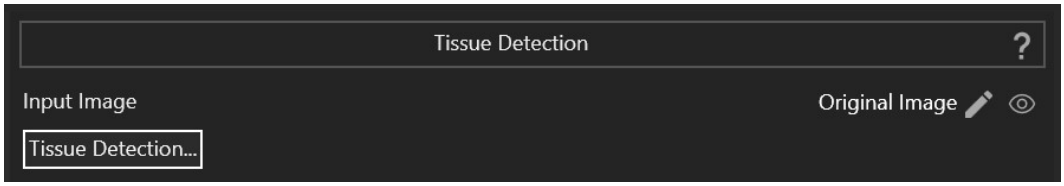


Figure 728 – Engine settings

Output:

Figure below shows the generated mask by the Tissue engine.

Figure 729 – Result of Tissue engine

41. Total Area

Figure 730 – Total Area

Where it can be found:

Layer's "Detection" ► Add ► Detection ► Total Area

Description:

Total Area is an engine available in the Detection section of the layers editor.

This engine identifies areas, defined by pixels with intensity values higher than the threshold level, and labels them with a unique ID.

The engine can be seen as a succession of the following operations:

1. Otsu threshold - if automatic background threshold is enabled, the threshold value is computed automatically;
2. Binarization - the pixels of the source grayscale image are compared with a threshold level;
3. Labelling - each set of connected pixels is assigned with a unique ID (label).

Parameters:

- Input - inputs a grayscale image
- Use global variable - enables / disables the usage of a global threshold instead of a threshold computed for each Field of View (FOV)
- Automatic Background Threshold - enables / disables the automatic computation of the background threshold:
    - Automatic - the automatic background threshold is computed using Otsu's method, which separates pixels into two classes, foreground and background. This threshold is determined by minimizing intra-class intensity variance, or equivalently, by maximizing inter-class variance. In some cases (e.g. uneven illumination), more than two classes can be

identified in the histogram. These kinds of issues can be overcome by changing the intensity range considered by the algorithm:

Figure 731 – Automatic Background Threshold Parameter

- Lower - the lower limit of the intensity range. Must be higher than background values and lower than the dimmest areas that must be detected;
- Upper - the upper limit of the intensity range. A lower value forces a lower threshold value. It must be higher than the average intensity of areas that must be detected.

- Manual - a user defined value is used as background threshold.

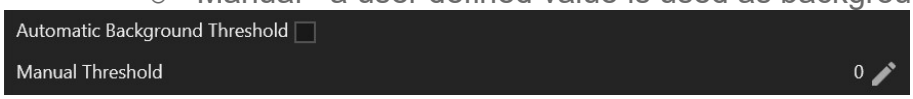


Figure 732 – Automatic Background Threshold Parameter: Setting manual threshold

Effect:

Identifies areas expressing a signal stronger than the threshold value.

Example:

- Input:

The input image is a fluorescence grayscale image (8-bit, 1-channel) showing the SpGold channel of a colorectal cancer sample.

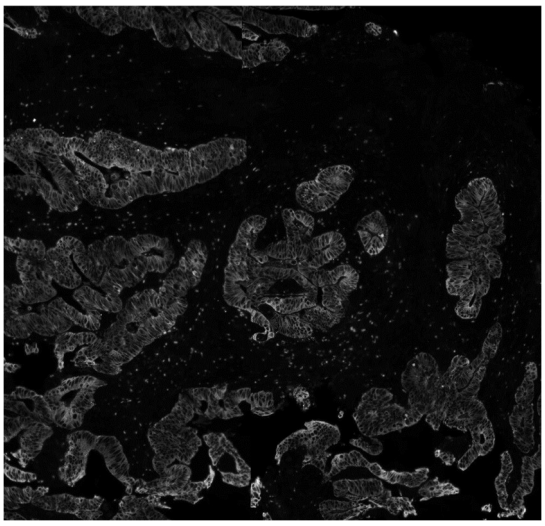


Figure 733 – Colorectal cancer sample (SpGold channel)

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a coded image representing the detected events.

Figure 734 – Engine settings

Output:

Figure below shows the detected events overlaid on the original image. For a better visualization, each event is highlighted with a different color.

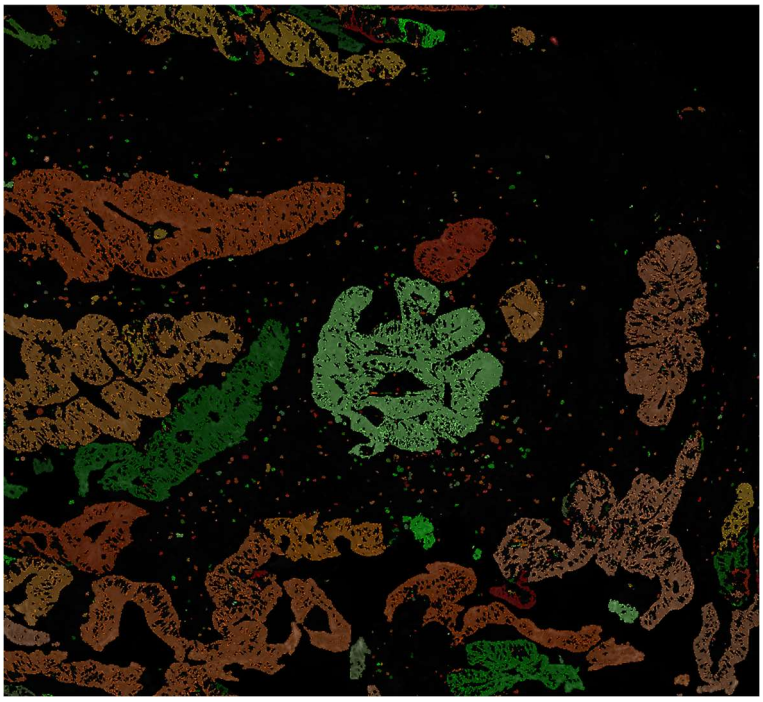


Figure 735 – Result of Total Area engine

42. Experimental

An experimental engine or measurement means it is out of IVD and requires special care from the users.

The results generated by experimental engines and measurements must be validated by the user, who is responsible for their accuracy.

42.1. Active Contours

Note	• An experimental engine or measurement means it is out of IVD (In Vitro Diagnostics) and requires special care from the users. • The results generated by experimental engines and measurements must be validated by the user, who is responsible with their accuracy.

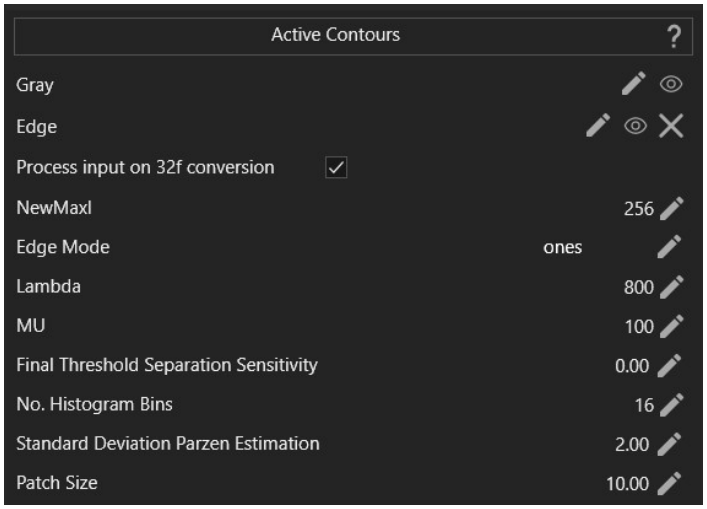


Figure 736 – Active contours

Where it can be found:

Layer's "Detection" ► Add ► Experimental ► Active Contours

Description:

Active contours is an engine available in the Detection section of the layers editor.

This engine is used to delineating object(s) from a noisy image, by finding active contours (or snakes), which are constrained by an energy minimization cost function.

E = Einternal + Eexternal

where Einternal and Eexternal represent the internal / external energy of the contour, which can be interpreted as forces that act to deform the contour.

Parameters:

- Gray - inputs a grayscale image
- Edge - only if Edge Mode - "from input" is selected
- Edge Mode
  - ones - default
  - from input - instead of an internal computation for edge method, use a user-provided image;
- Lambda and MU:

- these are proportional parameters; increasing and lowering the proportion will change the types of events that are selected.

- Final Threshold Separation Sensitivity: increasing this value will generate larger/stronger events.

Effect:

Detects a structure of interest (events) in a grayscale image.

Example:

- Input:

Figure below shows a fluorescence cell sample.

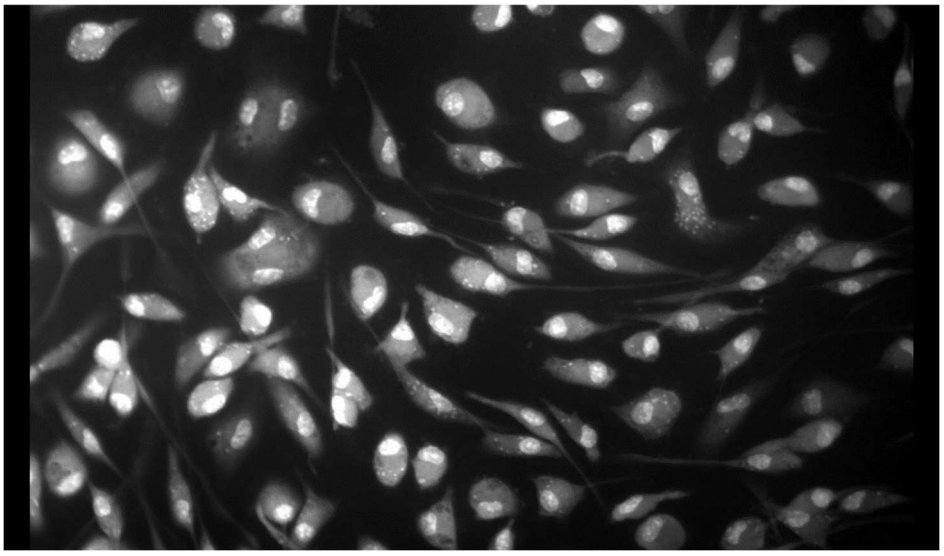


Figure 737 – Cells

- Engine settings:

Figure below shows the engine settings used for this example.

The engine's result is a coded image representing the detected events.

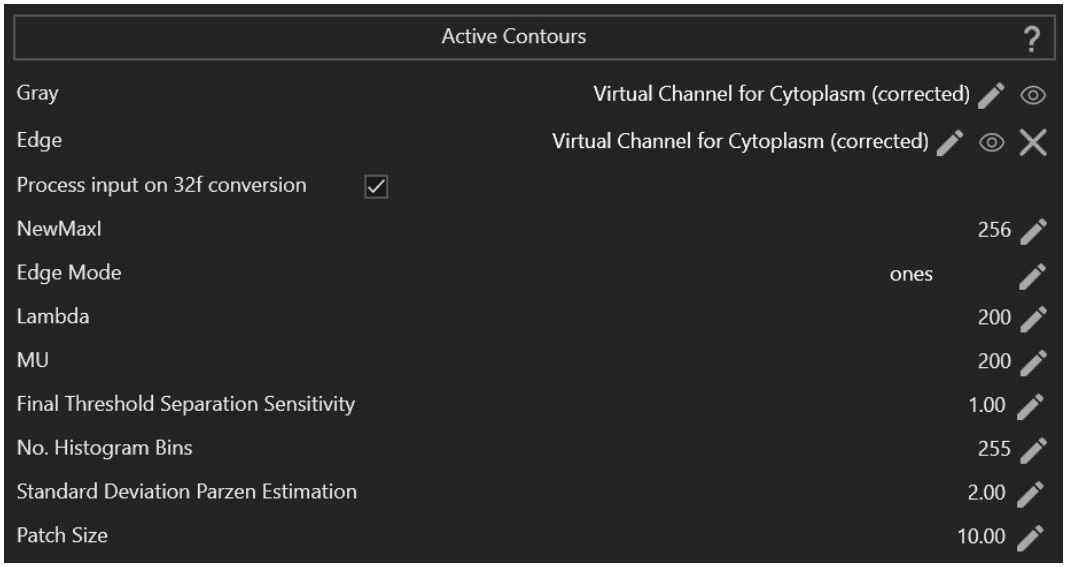


Figure 738 – Engine settings

Output:

Figure below shows the detection result of the Active Contours engine.

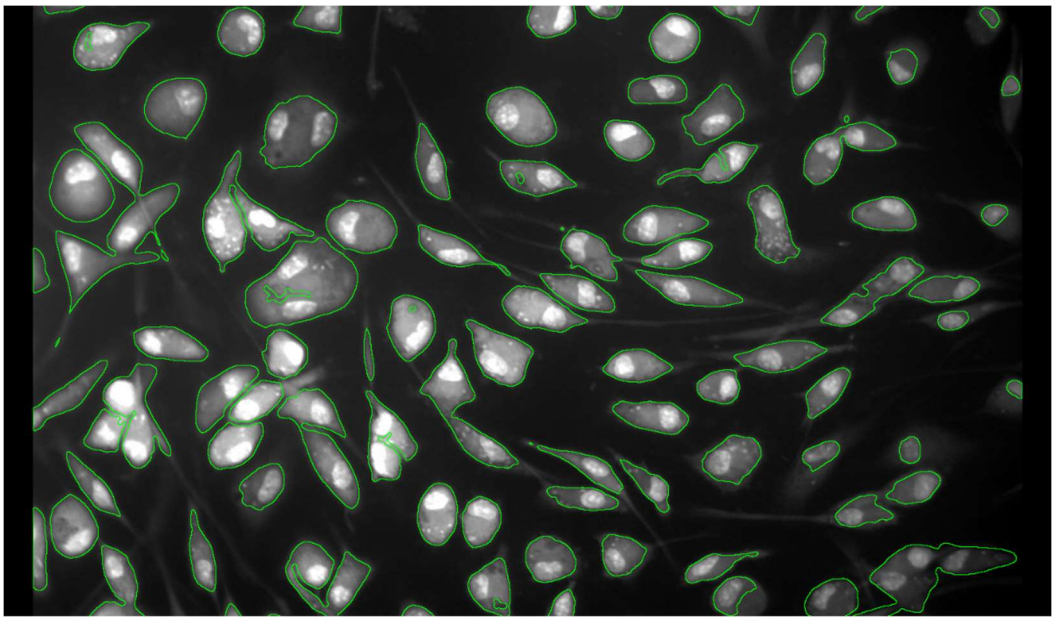


Figure 739 – Result of Active Contours engine

42.2. Cross Correlation

Note	• An experimental engine or measurement means it is out of IVD and requires special care from the users. • The results generated by experimental engines and measurements must be validated by the user, who is responsible with their accuracy.

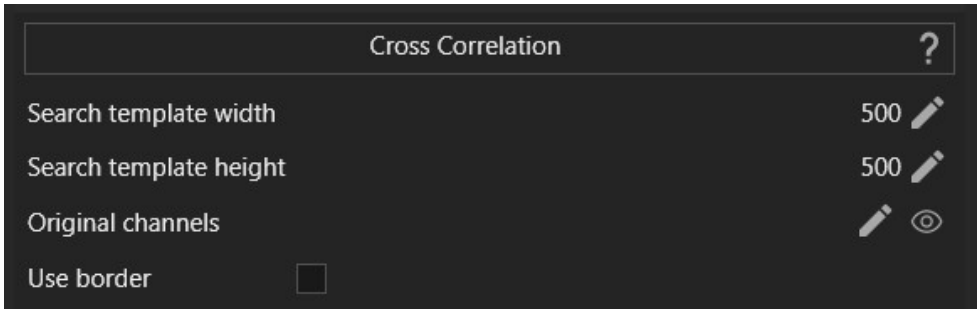


Figure 740 – Cross Correlation

Where it can be found:

Layer's "Pre-Processing" ► Add ► Experimental ► Cross Correlation

Description:

Cross Correlation is an engine available in the Pre-Processing section of the layers editor.

This engine registers multiple fluorescence samples, by correcting horizontal and vertical shifts (rotation is not corrected). To make this possible, all samples must have a common channel, usually the nuclear channel (e.g. DAPI).

To use the Cross-Correlation engine, the information from all samples subjected to alignment must be organized into a single sample. All samples are added as additional channels to one of the samples. This can be done using the option: Toolbar ► Project ► Advanced Tools ► Multiplexing - Add Channels.

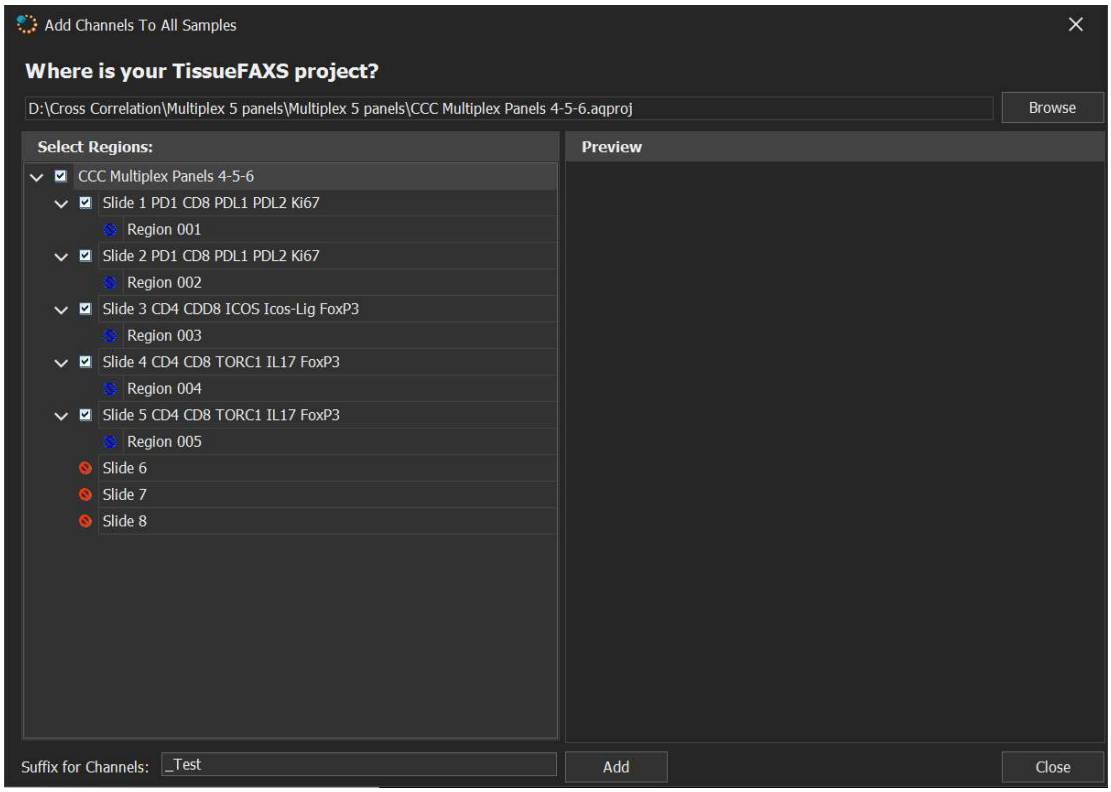


Figure 741 – Add Channels to All samples dialog

The Add Channels to All Samples dialog is straightforward:

1. Locate the project containing the sample to be registered using the Browse button
2. Select the sample or region of interest from the project's tree list
3. Define the suffix added to all new channels
4. Finalize the process using the Add button.

The engine outputs the aligned channels for each set of source channels (each source sample).

Parameters:

- Search Template Width - the window width used to align the samples
- Search Template Height - the window height used to align the samples
- Original Channels - the reference sample channel used for alignment
- Sample 2 - the Sample 2 channel used for alignment
- Sample 3 - the Sample 3 channel used for alignment

The nuclear channel is the most used channel for alignment because it contains the best suited spatial distributed information for cross-correlation.

Effect:

Performs tissue detection on the entire sample.

42.3. Events Length

Note	• An experimental engine or measurement means it is out of IVD and requires special care from the users. • The results generated by experimental engines and measurements must be validated by the user, who is responsible with their accuracy.

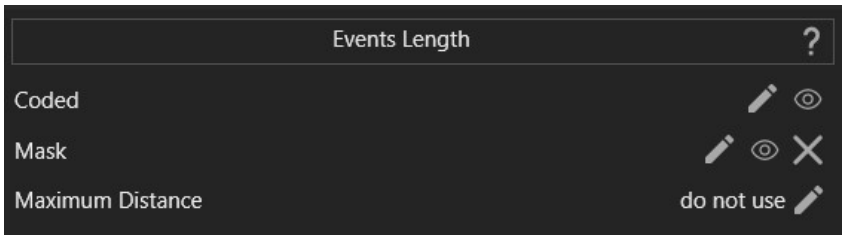


Figure 742 – Events Length

Where it can be found:

Layer's "Measurements" ► Add ► Experimental ► Events Length

Description:

Events Length is an engine available in the Measurements section of the layers editor.

This engine estimates the length of each event from a set of detected events (coded image). The result is a new measurement associated with each event.

This is a morphological measurement, meaning only the mask of the event is considered. The event's length is approximated with the shortest path (inside the event's mask only) from the most distant contour points of the event.

Figure 743 – Events length

Parameters:

- Coded - inputs a coded image representing the set of events;
- Mask - inputs a mask image. Restricts the engine in the indicated area only (white);
- Maximum Distance - the upper limit for distance value (very large events may take a long time to compute).

Effect:

Generates new events measurements: event's length.

42.4. Good Working Area

Note	• An experimental engine or measurement means it is out of IVD and requires special care from the users. • The results generated by experimental engines and measurements must be validated by the user, who is responsible with their accuracy.

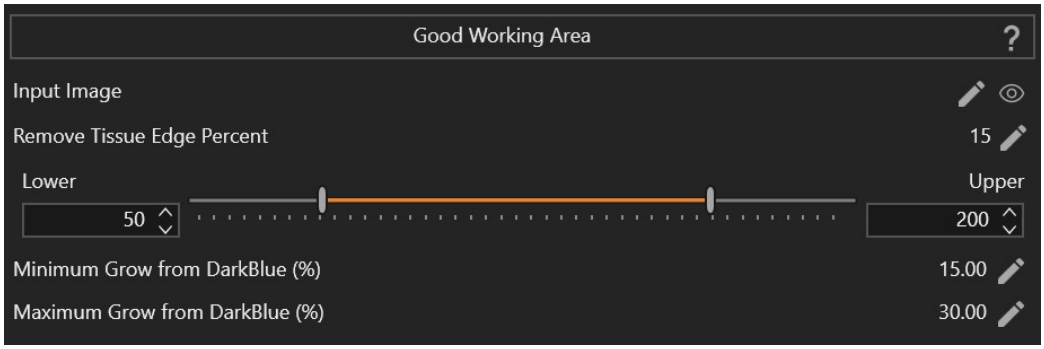


Figure 744 – Good Working Area

Where it can be found:

Layer's "Pre-Processing" ► Add ► Experimental ► Good Working Area

Description:

Good Working Area is an engine available in the Pre-Processing section of the layers editor.

This is an experimental engine, specially developed for Bone Marrow sample analysis, which identifies Good Working Areas (GWA). The GWA is used to detect leukocytes and get relevant statistical data.

The engine detects GWA using the information from the preview image (cache top-level) and tissue mask (Tissue Detection must be run prior to GWA detection).

Parameters:

- Image - the input image
- Peripheral safety area (%) - defines a safety inward area, starting from tissue edge, which may not be part of the identified GWA. The distance from tissue edge is defined by the formula:

$$Safety\ area\ (size)\ =\ min(sample\ width, sample\ height)\ \times\ \textbf{Safety area (\%)}$$

- BM automatic threshold range - the intensity range used to automatically compute threshold value using Otsu's method:
  - Lower - the lower limit of the intensity range
  - Upper - the upper limit of the intensity range
- GWA min distance from BM (%) - the minimum GWA distance from bone marrow. It can be interpreted as a safety distance from bone marrow. The distance value is computed in relation to the preview image size:

$$\boldsymbol{GWA\ min.distance}\ =\ \frac{\mathbf{1}}{\mathbf{2}}(\boldsymbol{sample\ width}\ +\ \boldsymbol{sample\ height})\ \times\ \textbf{GWA min distacen from BM (\%)}$$

- GWA max distance from BM (%) - the maximum GWA distance from bone marrow. The distance value is computed in relation to tissue area:

$$GWA\ max.distance = \sqrt{tissue\ area} \times \textbf{GWA max distance from BM (\%)}$$

Maximum size of the GWA can be defined by the formula:

$$GWA\ size = GWA\ max.distance - GWA\ min.distance$$

Effect:

Identifies the good working area in a bone marrow sample.

42.5. Leukocytes

Note	• An experimental engine or measurement means it is out of IVD and requires special care from the users. • The results generated by experimental engines and measurements must be validated by the user, who is responsible with their accuracy.

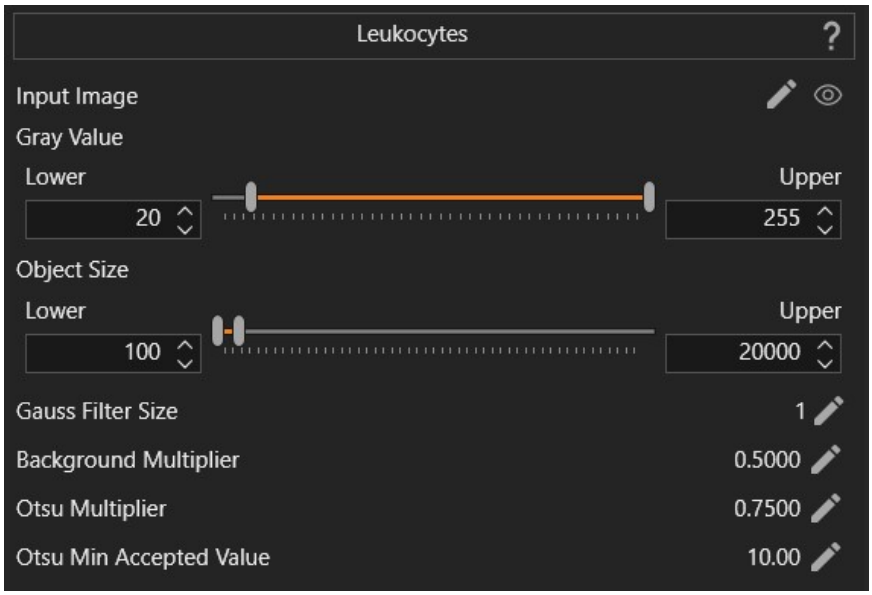


Figure 748 – Leukocytes engine

Where it can be found:

Layer's "Pre-Processing" ► Add ► Experimental ► Leukocytes

Description:

Leukocytes is an engine available in the Pre-Processing section of the layers editor.

This is an experimental engine, specially developed for blood smears and bone marrow sample analysis, which detect leukocytes. The result is a mask image highlighting the leukocytes presence (white).

Parameters:

- Image - the input image
- Leukocytes automatic threshold range - the intensity range used to automatically compute the threshold value using Otsu's method
- Leukocytes size range - the detected leukocytes size must verify the condition:

$$\textbf{Lower}\ area\ limit\ <\ Leukocyte\ area\ <\ \textbf{Upper}\ area\ limit$$

- Gauss filter radius (px) - defines the kernel size used to apply Gaussian filter (default = 1), using the formulas:

$$Kernel\ width = 2 * \textbf{Kernel radius(px)} + 1$$

$$Kernel\ height = 2 * \textbf{Kernel radius(px)} + 1$$

- Coefficient (background) - the coefficient used to relax / increase background correction
- Coefficient (leukocyte threshold) - the coefficient used to relax / increase the leukocytes threshold level (automatically computed)
- Leukocytes threshold min. accepted value - the minimum value accepted for leukocytes automatically computed threshold level. If the threshold level is lower than this parameter value, it may be possible no leukocytes are present in the image

Effect:

Detects the presence of leukocytes.

Example:

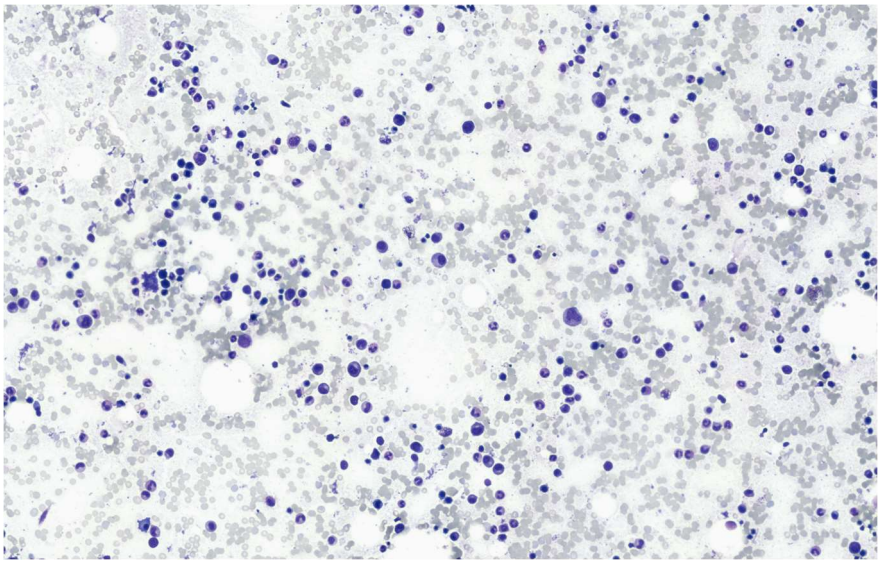


Figure 749 – Input image: bone marrow sample

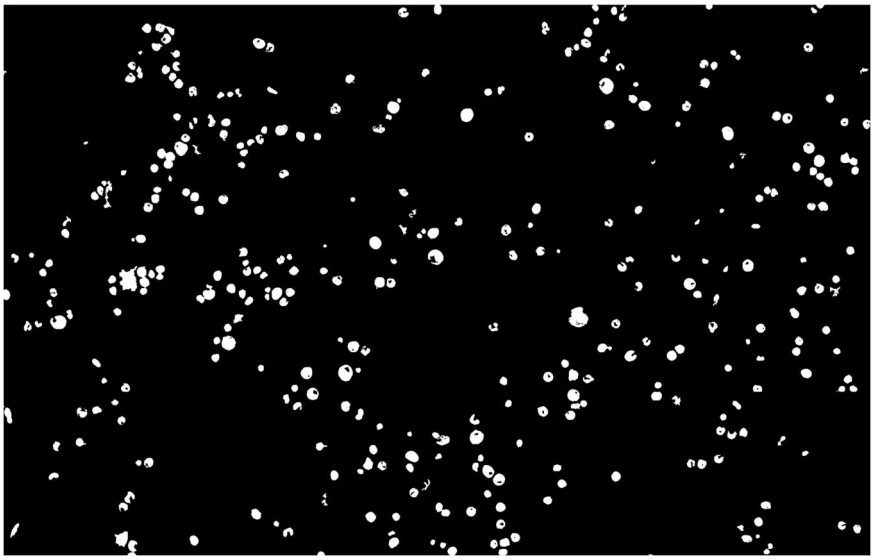


Figure 750 – Resulting image: bone marrow sample

42.6. Object Selector

Note	• An experimental engine or measurement means it is out of IVD and requires special care from the users. • The results generated by experimental engines and measurements must be validated by the user, who is responsible with their accuracy.

Figure 751 – Objects Selector

Where it can be found:

Layer's "Measurements" ► Add ► Experimental ► Object Selector

Description:

Objects Selector is an engine available in the Measurements section of the layers editor.

This is an experimental engine, specially developed for bone marrow sample analysis, which randomly selects a user defined number of events (leukocytes) from the entire set of events. The selected events positions are transmitted to TissueFAXS for higher magnification acquisition (e.g. 100x).

Parameters:

- Max. number of events - the number of randomly selected events from the entire set of events
- Select all events - selects the entire set of events

Effect:

Randomly selects a user defined number of events from the entire set of events available.

42.7. Proximity Background Percent

Note	• An experimental engine or measurement means it is out of IVD and requires special care from the users. • The results generated by experimental engines and measurements must be validated by the user, who is responsible with their accuracy.

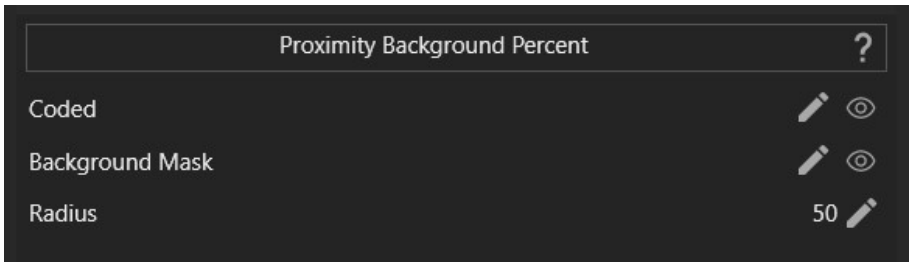


Figure 752 – Proximity Background Percent

Where it can be found:

Layer's "Measurements" ► Add ► Experimental ► Proximity Background Percent

Description:

Proximity Background Percent is an engine available in the Measurements section of the layers editor.

This is an experimental engine, specially developed for bone marrow sample analysis, which estimates the amount of background surrounding an event. The result is a new measurement associated with each event, which may be useful identifying isolated leukocytes (not part of a leukocyte cluster).

Parameters:

- Coded - inputs a coded image representing the set of events
- Background mask - inputs a mask image highlighting the background (white)
- Radius - the area distance from the event, used to compute the percentage of background

Effect:

Generates new events measurements: proximity background percent

42.8. Scanner

Note	• An experimental engine or measurement means it is out of IVD and requires special care from the users.

	• The results generated by experimental engines and measurements must be validated by the user, who is responsible with their accuracy.

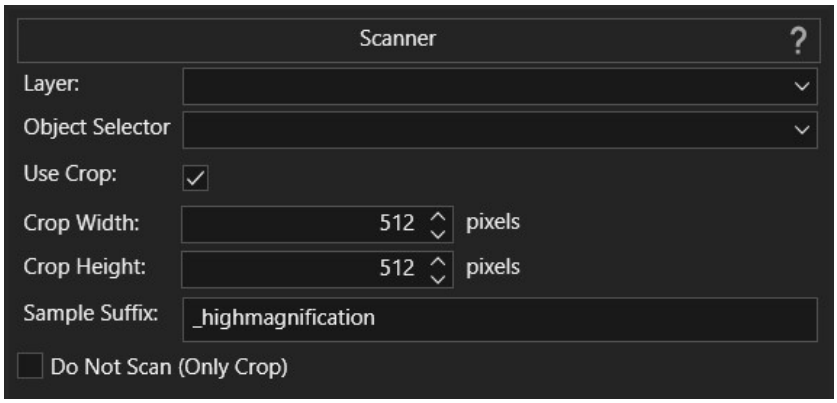


Figure 753 – Scanner

Where it can be found:

Layer's "Pre-Processing" ► Add ► Experimental ► Scanner

Description:

Scanner is an engine available in the Pre-Processing section of the layers editor.

This engine represents a communication tool between StrataQuest (analysis) and TissueFAXS (scanner). A set of events detected in the analysis workflow, can be the subject of re-acquisition using a higher magnification.

An example is the blood smears or bone marrow analysis workflow, where:

- samples are acquired using low magnification objective (20x / 40x)
- leukocytes positions are identified on the low magnification images
- leukocytes are re-acquired using high magnification objective (66x / 100x)
- leukocytes are classified based on the information extracted from high magnification images.

The events positions (centroids coordinates) are sent to TissueFAXS for re-acquisition, usually using a higher magnification. The events are grouped in Fields of View (FOVs) to optimize the acquisition time. The newly acquired FOVs are available in a new sample added to the project.

Parameters:

- Layer - specifies the layer where the set of events (coded image) is available
- Events - a selected set of events (coded image) for re-acquisition
- Use crop - crops images
- Crop width - the width of the cropped image
- Crop height - the height of the cropped image
- Sample suffix - specifies the suffix of the new sample added to the current project, generated by the newly acquired data (with high magnification)
- Do not scan (crop only) - enables / disables the crop generator for all events from available images (without sending the information to TissueFAXS for additional scan). Available only when Use crop is enabled.

Effect:

Ensures the communication between StrataQuest and TissueFAXS for complex analysis workflows, which require re-acquisition of events of interest.

42.9. Tiles

Note	• An experimental engine or measurement means it is out of IVD and requires special care from the users. • The results generated by experimental engines and measurements must be validated by the user, who is responsible with their accuracy.

Figure 754 – Tiles

Where it can be found:

Layer's "Detection" ► Add ► Experimental ► Tiles

Description:

Tiles is an engine available in the Detection section of the layers editor.

This engine splits the Field of View (FOV) into user defined sized tiles. The result is a set of events (coded image), each event representing a tile.

The split is made based on image size and tile size. Additional tiles validation can be imposed by providing a grayscale image and additional tissue presence constraints:

- Minimum tile average intensity value
- Minimum tile area percent represented by tissue

Parameters:

- Gray image - inputs a grayscale image
- Tissue threshold (gray image) - the threshold level used to identify tissue mask from Gray image
- Tile size (px) - the size of a square used to define each tile
- Tile min. avg. intensity - the minimum tile mean intensity (computed on Gray image)
- Tile -min. accepted area percent (%) - the minimum tile area percent represented by tissue mask area (inside tile)

Effect:

Splits the source image into tiles, generating a new set of events (coded image).

Example:

Figure 755 – Engine Settings

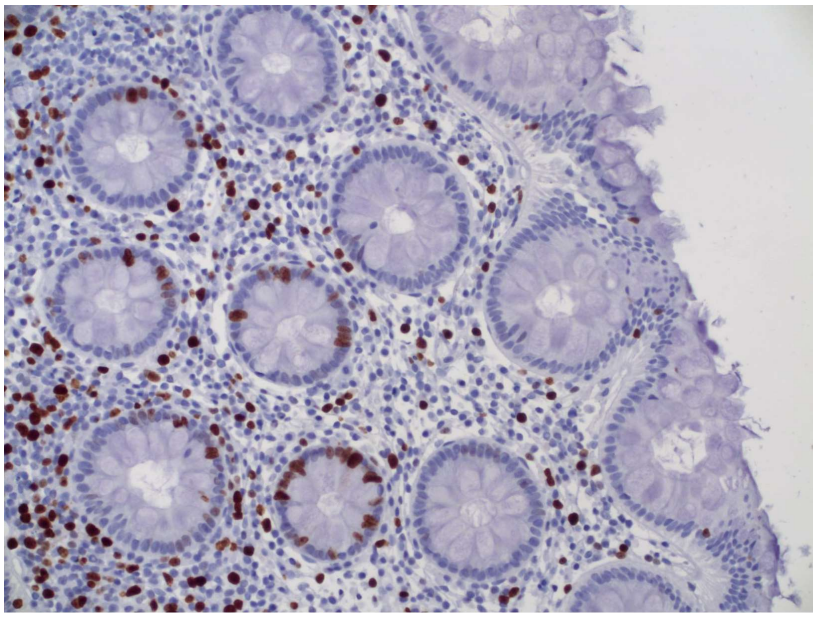


Figure 756 – Original Image

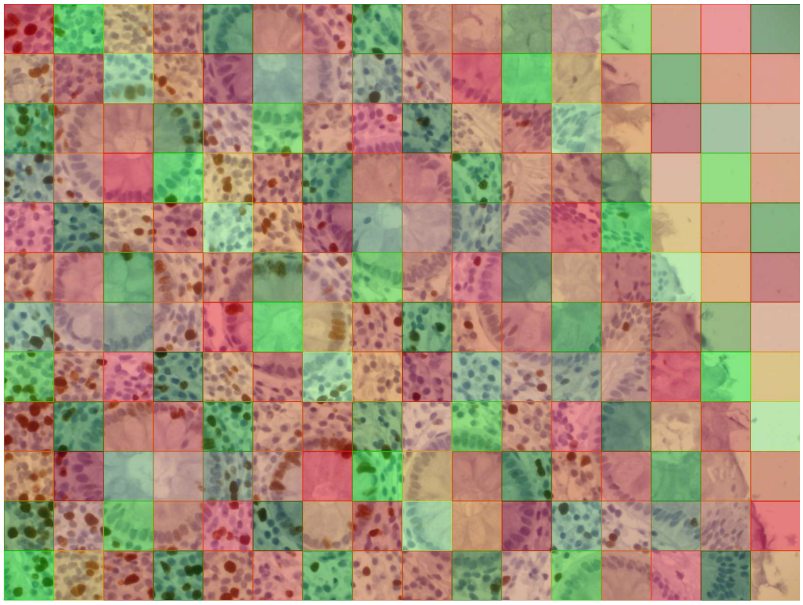


Figure 757 – Tiles result