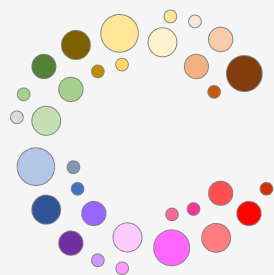


ImageC Documentation

ImageC version stable\nnewline Generated on 2025-06-03 at 18:06 UTC



Contents

| | |
|--|-----------|
| Contents | i |
| Summary | 1 |
| First steps | 5 |
| Operation | 7 |
| Project {#project-tab} | 7 |
| Image Grouping {#image-grouping-tab} | 8 |
| Working directory | 8 |
| Images {#images-tab} | 8 |
| Classification {#classification-tab} | 8 |
| Pipelines {#pipelines-tab} | 9 |
| Starting the analysis | 9 |
| Results | 11 |
| Plate view | 11 |
| Image view | 11 |
| Interactive mode {#interactive-mode} | 12 |
| Data export {#data-export} | 12 |
| Fundamentals | 13 |
| Image format | 15 |
| Image planes {#image-planes} | 15 |
| OME xml {#formats-ome} | 16 |
| Big images {#big-images} | 16 |
| Classification | 17 |
| Classes | 17 |
| Classification presets | 17 |
| Objects | 19 |
| Confidence | 19 |
| Area size | 19 |
| Perimeter | 19 |
| Circularity | 19 |
| Centroid | 19 |
| Bounding box | 20 |
| Object ID | 20 |
| Parent Object ID {#parent-object-id} | 20 |
| Origin Object ID | 20 |
| Tracking ID | 20 |
| Position | 20 |

| | |
|--|-----------|
| Nr. of intersecting objects | 20 |
| Pipelines | 21 |
| Pipeline settings | 21 |
| Pipeline input settings | 21 |
| Pipeline steps | 21 |
| Pipeline history | 22 |
| Metrics | 23 |
| Metrics and Statistics {#metrics-and-statistics} | 23 |
| Plate view | 23 |
| Well view | 25 |
| Image view | 26 |
| Commands | 29 |
| Overview | 31 |
| Image processing | 31 |
| Object segmentation | 31 |
| Image processing | 33 |
| Color filter | 33 |
| Blur | 34 |
| Intensity | 34 |
| Rolling ball | 35 |
| Median subtract | 38 |
| Edge detection | 38 |
| Image math | 39 |
| Image Cache | 41 |
| Storage scope | 41 |
| Enhance contrast | 42 |
| Rank filter | 42 |
| Binary image processing | 45 |
| Threshold | 45 |
| Maximum threshold | 46 |
| Threshold classes {#threshold-classes} | 46 |
| Auto threshold | 47 |
| Minimum threshold in combination with auto threshold | 47 |
| Watershed | 47 |
| Morphological transform | 48 |
| Erosion | 49 |
| Dilation | 49 |
| Opening | 49 |
| Closing | 49 |
| Fill holes | 49 |
| Classification | 51 |
| Classifier | 51 |
| AI classifier | 54 |
| Deep learning models | 55 |
| Deep learning engines | 55 |
| Thresholds | 55 |
| Mask threshold | 56 |

| | |
|--|-----------|
| Class threshold | 56 |
| Hough transformation | 56 |
| Object processing | 57 |
| Voronoi | 57 |
| Voronoi centers | 61 |
| Max radius | 61 |
| Masking class | 61 |
| Object filter | 61 |
| Exclude areas at the edges | 61 |
| Exclude areas without center | 61 |
| Reclassify | 62 |
| Filters | 63 |
| Intersection filter | 63 |
| Intensity filter | 63 |
| Match handling | 64 |
| Objects to binary | 64 |
| Object transform | 64 |
| Save control image | 64 |
| Measurement | 65 |
| Colocalization | 65 |
| Colocalizing handling | 66 |
| Measure intensity | 67 |
| Measure distance | 67 |
| Filtering | 69 |
| Threshold filter | 69 |
| Noise filter | 70 |
| Tutorials | 71 |
| Tutorials | 73 |
| EVAnalyzer | 75 |
| EVAnalyzer | 77 |
| Spot count | 77 |
| Spot count | 77 |
| Spot count | 77 |
| Advanced | 79 |
| Yest cell detection | 79 |
| Technical insights | 83 |
| Overview | 85 |
| Resources and Limits | 87 |
| Limits | 87 |
| Files Created by ImageC | 89 |
| File types | 89 |
| Global Files and Directories | 89 |

| | |
|---|------------|
| Local Files and Directories | 89 |
| Files created during analysis | 90 |
| Development | 91 |
| Debugging | 93 |
| Linux | 93 |
| Window | 93 |
| macOS | 93 |
| Building | 95 |
| Testing | 97 |
| ImageC Blog | 99 |
| Cell Segmentation | 101 |
| Acknowledgments | 103 |

Summary

This document contains [ImageC's official documentation and guides](#) in a single-file easy-to-search form. If you find any issues, please report them [as a GitHub issue](#). Contributions are very welcome in the form of [pull requests](#). If you are considering submitting a contribution to the documentation, please consult our [contributor guide](#).

Code repositories:

- ImageC source code: github.com/joda01/imagec
- ImageC documentation source code: github.com/joda01/imagec-doc

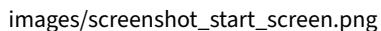
First steps

Operation

This guide takes you through your first steps with ImageC, helping you create your first ImageC project and start analyzing images.

Tip. Please read [Installation](#) first and run ImageC on your computer before starting with this tutorial.

After download the correct ImageC bundle for your computer, unzip the downloaded data and use the `imagec` for Linux and macOS resp. `imagec.exe` for Windows to start ImageC. The start page will be shown once ImageC is successfully launched.



The tabs in the navigation pane on the left hand side are used to enter basic settings. Navigate from the left **Project** tab over the **Image** to the **Classification** tab. Once all settings in these tabs are done, you can start creating image processing pipelines in the **Pipeline** tab.

Project {#project-tab}

Starting with the project settings, basic information about the experiment and the used image setting must be done.

| Title | Description | Mandatory |
|-------------------|--|-----------|
| Working directory | Storage Directory of the images to be analyzed. | x |
| Experiment name | Title of the experiment stored together with the results. | |
| Scientist | Name of the person who is responsible for this analysis. | |
| Organization | Organization responsible for the analysis. | |
| Job name | Name of the job to identify the run (auto generated if empty). | x |
| Group by | Images may be left ungrouped, or can be grouped by Filename regex or Directory. | x |
| Filename regex | If Images are grouped by filename, the regex should indicate the order of the images. | x |
| Regex test | Used to test the regex settings. Enter your Image Name and see if the wells are recognized. in the regex test result | |
| Z-Stack | Define how to handle Z-stacks in the images | x |
| T-Stack | Define how to handle T-stacks in the images | x |
| Well order | If images are taken from in a (6, 12, 24, 96, 384) well format, the order of the images position in the well can be determined here. | x |
| Plate size | Size of the uses microscopy plate. | x |
| Notes | Some free text notes on the experiment. | |

Image Grouping {#image-grouping-tab}

When grouping by Foldername or Filename is selected ImageC will group the images based on these settings, displays the images grouped by Well in the results and calculates the statistics based on the determined group.

Filename grouping uses regex (regular expressions) to extract the position on the plate and the position of the image in the well from the image filename. Extracted are: plate row and column position and the image index in the well.

A change of the grouping settings after analysis is currently not supported by ImageC. If the grouping settings are changed the analysis has to be repeated.

Warning. Make sure that the grouping options and regex settings are correct, as they are needed for valid image sorting and mean well infos. However, if the grouping settings are wrong, these statistics will also be calculated in a wrong way.

To extract the well position using regex from the filename correct, it is expected that the row position in the filename is a character in the range of [A-Z] and the column and index a decimal number. A typical series of file names for the regex {regex}_ ((.) ([0-9] +)) _ ([0-9] +) might look something like the following:

```
name_A1_1.tif  name_A1_2.tif  name_A1_3.tif  name_A1_4.tif  name_A2_1.tif  name_A2_2.tif
name_A2_3.tif name_A2_4.tif
```

To experiment with regular expressions, have a look at [regex101](#).

Working directory

The Working Directory should be set to the folder where the images to be analyzed are stored. ImageC will perform a recursive folder search using the selected Working Directory as the base folder to find all supported image files. All found files are listed in the Images panel.

Tip. See section [Images](#) for a full list of all supported image formats.

Images {#images-tab}

Once a working directory has been selected and the folder scan is complete, all the images found will be listed in the table located in the Images tab.

By clicking on an image the image meta information of the selected image is loaded and displayed in the properties table below. The image selected in this tab is also the image used in the pipeline preview.

To the top a search field allows to filter the images in the list against its filename.

images/screenshot_images.png

Classification {#classification-tab}

First before creating pipelines and starting the analysis the object classes must be defined.

images/screenshot_classification.png

A separate object class must be defined for each different object type and population to be extracted from the images. Example object classes might be: dapi@nucleus, cy7@spot, cy5@spot, col@cy7cy5. ImageC allows to either define your own classes, load a preset of classes from a template or try to automatically populate classes, using the magic stick button, based on the image meta data and channels.

Tip. Class names can consist of two parts, separated by an @. The first part is used to sort the classes in the drop down boxes to help you keep track of them.

Double click on a class opens the Class editor. The Class Editor is used to define the class name and the color used for all detected objects of this class. In addition the Metrics section allows to define the object metrics to be displayed per default in the results view. However, the measurement settings per object class can be changed at any time without having to rerun the analyses.

Double click on a class opens the pipeline editor. Selected metrics are displayed per default in the results view after the analyses has been finished but can be changed at any time without having to rerun the analyses.

Pipelines {#pipelines-tab}

The pipelines tab is used to create image processing pipelines which are used to extract objects of interest from the image channels. ImageC has no limit in the number of pipelines which can be added to a project. During a analyzes ImageC, processes each active pipeline. The optimal processing order and parallel processing of the different pipelines is determined by ImageC automatically based on available CPU cores and possible pipeline dependencies.

images/screenshot_pipelines.png

By clicking on the arrow beside the Plus button a drop down with predefined analyzing pipelines is opened. All past EVAnalyzer pipelines are included in this version, marked with the small EV icon to the left. Select EV channel for loading a pipeline (preprocessing, object filtering, segmentation) optimized for EV quantification from single vesicle imaging images with low background. Select Cell brightfield for loading a pipeline (preprocessing, object filtering, segmentation) optimized for cell segmentation on brightfield images. Select Nucleus for loading a pipeline (preprocessing, object filtering, segmentation) optimized for nucleus segmentation after fluorescent labelling of the nuclei (e.g. Hoechst, DAPI). Select EV in cell for loading a pipeline (preprocessing, object filtering, segmentation) optimized for EV quantification in complex material like cells. Just press the New pipeline button to start with an empty pipeline.

Best practice. It is a good practice to add one pipeline for each image channel to extract objects from and one pipeline for each object processing step like Coloc or in cell counting.

By click on a pipeline, the pipeline editor is opened. On the left hand side the input and output options can be defined. The input of a pipeline can either be an image channel or an empty image.

The Pipeline steps box contains all commands which are applied on the input image. All steps are performed from top to bottom. Each step can take either an image or a set of objects as input and either an image or a new set of objects as output. Based on the used command either the image is processed, objects are extracted or objects are processed. The color bar next to the command indicates the type of command.

Tip. See the section [Pipeline steps](#) for detailed information about the available pipeline steps and their behavior.

Gray commands manipulate images, white command work on binary images and green commands work on objects. Translation commands translate the output from one input type to an other output type. For example, a threshold command translates an image (grey) into a binary image (white) and an object classification command translates a binary image (white) into objects (green).

A live preview is displayed on the right. It shows the resulting object segmentation after all applied pipelines steps. Changing a parameter will directly change the preview, enabling a fast and easy adjustment and fine-tuning of the settings. A live object count is displayed in the legend of the preview image.

Based on image size and the complexity of the selected preprocessing algorithms it could take a couple of seconds for refreshing the preview. The preview can additionally be zoomed in and out and a second window with the original image and the processed image side by side further enables smooth segmentation setting.

Starting the analysis

After all pipelines are created, the analysis can be started by pressing the Play button on the top.

A dialog box informs you about the progress of the analysis. At the bottom left of the dialog a `Open results folder` button is placed. Press this button to open the file explorer showing the folder with results of the actual analysis run.

With `Stop` button a running analysis can be interrupted. It may take a couple of minutes to stop a running analysis since all still in progress tasks have to be finished.

Press the `Close` button to close the dialog after a successful finished analysis run.

Results

ImageC stores the results of a analysis run within a database file named `results.icdb`. Results from a previous run can be opened by clicking the arrow beside the Open button on the toolbar.

Once a results file is opened the first time, the measurement values selected in the **class editor** are displayed in the table plate view.

To add additional metrics press the blue Add column button. The opened Add Column dialog displays all available metrics which can be added to the table.

ImageC saves the actual table settings with the database file so that they are restored the next time the results are opened.

See section **Metrics** for a complete description of the available metrics.

Plate view

ImageC opens the PLate view panel per default.

images/screenshot_results_start.png

The plate shows the results grouped by well as average value of the selected statistics from the images of the group.

Image grouping must be setup before the analyze is started in the **Project tab**. If Ungrouped was selected as Group by method, the plate view is not shown and ImageC directly jumps to the image list view.

Using the heatmap button the view can be switched from a table view to a heatmap view.

images/screenshot_plate_view_heatmap.png

Wells are coloured using a heatmap calculated from all data displayed, with the mean value of all wells as the centre of the spectrum and the minimum and maximum as the outer left and outer right limits of the spectrum. In heatmap view a dropdown in the toolbar allows to select which column of the table should be displayed as heatmap.

Image view

A double click on a well in the heatmap view or a column in the table view will prompt the opening of a detailed view of the selected well.

images/screenshot_well_view.png

The Image view displays each image of the well ordered by the image index specified in the Well order matrix. The image index was extracted from the filename during the analysis using the specified regex. To reorder the image position displayed in the well view matrix use the Well order settings field.

A string formatted order matrix can be used to customize the order of images displayed in the well order matrix. Per default the matrix is set to: `[[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]]`. The numbers in the square brackets are the image indexes, where each comma-separated square block represents a row and the comma-separated numbers represent columns. This results in the following screen sequence for the example above.

Interactive mode {#interactive-mode}

In image detail view mode ImageC displays the metrics for each individual detected object. When an object in the table is selected, ImageC opens the image that the object was extracted from and highlights the position in the image where the object was found.


The position of each object in the image can be identified using this feature. However, ImageC needs access to the original images for a correct working interactive mode. If the `results.icdb` file or the images are moved, ImageC will prompt you to specify the new storage location.

Best practice. For ImageC to automatically find the images for interactive mode, it is best practice to keep the `results.icdb` file in its original folder, as this was where it was generated.

```
<images_folder>
|- <image_01>
|- <image_02>
|- ....
|- imagec
    |- <job_name>
        |- results.icdb
```

images/screenshot_image_view_interactive_mode.png

Data export {#data-export}

The Download button  allows the current settings to be exported as either XLSX or R.

Fundamentals

Image format

ImageC can be used with a wide range of image formats, thanks to the open source library **Bio-formats**: <https://docs.openmicroscopy.org/bio-formats> which is shipped together with this application.

Following 8-bit or 16-bit grayscale and 8-bit RGB color images are supported:

```
tif,tiff,btif,btiff,btf,jpg,jpeg,vsi,ics,czi,
nd2,lif,lei,fli,scn,sxm,lim,oir,top,stk,nd,
bip,fli,msr,dm3,dm4,img,cr2,ch5,dib,ims,pic,
raw,lsc,std,spc,avi,cif,sif,aim,svs,arf,sld
```

A full list of all supported image formats can be found on the [Bio-formats](#) homepage.

Warning. There are some special Photoshop tiff formats which are not supported. When using tiff images take care about using either RAW tiff or the [OME-TIFF](#) format for multi channel support.

Best practice. Use RAW images directly from your microscope without any pre-processing or compression for the best detection results.

ImageC can handle multi channel images as well as big histological images without the need of any preprocessing. Use the images directly as they were captured by the microscope camera, without any compression or pre-processing to get the best results.

Especially for multi-channel images, avoid merging the channels into an RGB color image for processing, as most of the image information will be lost. Use the original multichannel image and use ImageC pipelines to process each of the channels individually to extract the objects of interest.

Image planes {#image-planes}

Microscope images usually consist of several individual images that are summarized in a common image file. These individual images are named image planes. Each plane is identified by its channel, z-stack and time stack.

ImageC is able to access each image plane of an image and process these planes individually based on the taken project and pipeline settings.

An image may comprise one or more C (channels), with each channel in turn consisting of a series of T (time) stacks, and these in turn consisting of a series of Z stacks.

The combination channel, Z stack (z) and time stack (t) is called image plane. ImageC is able to process each image plane of an image, based on the taken project and pipeline settings.

Within a pipeline, select the image channel to be loaded as the starting point for this pipeline. The channel is a number from 0 to x. A separate pipeline must be created for each image channel required for processing.

In a second step it is necessary to define how z-stacks and time stacks should be processed. These settings can be taken in the **Project** tab of ImageC. It is possible either to process each Z-stack image individually, using a projection algorithm to combine all the images of a Z-stack into a single stack image, or to analyze exactly one image of the Z-stack. If a projection algorithm is to be used, the algorithm to be used must be defined within the individual pipeline. For the time stack it is possible either to analyze one image of a time stack or to analyze the whole time series.

OME xml {#formats-ome}

ImageC supports reading OME XML meta data stored beside image files.

OME specifies a structure how image meta can be shared in a standardized way. ImageC tries to parse this XML data if it is found in an image and displays the meta data in a sidebar on the start screen.

The OME metadata contains not only image metadata for the end user, but also information about the number of channels and the channel order. Therefore, OME metadata is mandatory if multi-channel images are to be processed.

ImageC assumes that only one channel is available if no OME metadata is found.

Tip. For a full specification OME see <https://ome-model.readthedocs.io/en/stable/ome-xml/index.html>

Big images {#big-images}

Because computing resources are limited, image size can be a limiting factor when working with histological images, particularly during analysis.

One of the hardest limits is an image resolution of 46340x46340 pixels, which exceeds the 32-bit signed data type limits of many common image processing algorithms. In addition to this data type limitation, most personal computers have limited RAM, which restricts the maximum image size that can be analyzed.

ImageC solves both issues by automatically breaking large images down into smaller pieces called image tiles. The maximum tile size can be specified in the [project settings tab](#).

If an images is loaded which is bigger than this specified tile size, ImageC analyzes tile by tile instead of the whole image at once. After the analyses are finished, ImageC will automatically combine the results from each tile to create a full image result again. From a user's perspective, the large image can be analyzed with ImageC as is to get results for the entire image.

The big tiff file format breaks the the 4 gigabyte (32-bit) size limit in comparison to the normal tiff format.

BigTIFF images are usually split into tiles whereby a typical tile size is 512x512 px. When analyzing, ImageC opens in this example 64 tiles at once and analyses one such composite tile after another. This is necessary because when working with such large images, the entire image cannot be loaded into RAM at once.

Warning. ImageC can only generate a navigation map if the large image is stored as a pyramid image. When working with large images, ensure that only supported image formats are used and that the image container contains a pyramid representation of the image.

For large images, ImageC generates a preview of only one tile to avoid exceeding the computer's RAM limits. The image map navigator within the preview window allows you to navigate through the different image areas.

Following image formats support big images including pyramid images:

.afi, .svs, .ims, .vsi, .ndpi, .ndpis, .jp2, .tiff
.tif, .tf2, .tf8, .btf, .tif, .sld, .jpg, .czi

Classification

The concept behind ImageC is to run pipelines containing image pre-processing and object segmentation steps with the goal of extracting regions of interest from the input images. For each extracted **object** the origin information: image, image channel, z-stack and t-stack and some metrics are stored. In advanced, each **object** is classified for object statistics calculation and later quantification. For classification ImageC provides the annotations `Object Class`. Every object is annotated with exact one object class.

Classes

The first step before creating pipelines or starting the analysis is to define which classes are needed for object classification in your application using the **Classification tab**.

Classes represents the populations which should be distinguished.

../first_steps/images/screenshot_classification.png

Use the **Plus** button to add a new class. In addition to the color, the name and the default displayed metrics for the class can be specified. Using the `@` symbol allows to group classes for a better reading in the selection tab.

Instead of manually specifying all required classes, ImageC provides the ability to populate the classification settings from the image channel information using the **Magic Stick** button.

Best practice. It is a best practice to use the fluorophor as prefix followed by the object type: `dapi@nucleus`, `cy5@spots`, `cy@spots-in-cell`. For processed objects use the process operator: `coloc@cy5cy7-spots`.

Classification presets

ImageC also allows to create classification presets by selecting a preset from the drop down of the **Plus** button. A classification preset is a set of predefined classes which can be loaded and shared with others. The idea behind a preset is making results easier comparable by using the same nomenclator for each analysis.

Using the **Save as template** option allows to create a new preset from an existing settings.

Objects

Objects are the result of an image **classification step** and represents the quantified data of an formally extracted region of interest. Objects are the final part of an ImageC pipeline and are those elements which are finally stored to the results database.

Each object is assigned to exact one object class to scope it. Together with the classification labels some object metrics are calculated. ImageC distinguishes between **image plane** independent and **image plane** dependent metrics. Image plane independent metrics are globally valid for the object whereby image plane dependent metrics are calculated based on the image pixel data.

Tip. See section **Metrics** to get an overview of the image plane dependent object metrics.

Confidence

The confidence interpretation depends on the used segmentation mode. For **threshold** segmentation the confidence value is the minimum threshold which was used to finally extract the object from the rest of the image. The number range is from zero to 65535.

If **AI classifier** is used the confidence value represents the output prediction probability of the used AI model. The number range is from zero to one.

Area size

The area size is defined by the number of not black pixels within the shape of the extracted region of interest. It's unit is px².

Perimeter

The perimeter calculation has been ported from ImageJ to ImageC.

The algorithm counts pixels in straight edges as 1 and pixels in corners as $\sqrt{2}$. It does this by calculating the total length of the ROI boundary and subtracting $\sqrt{2}$ for each non-adjacent corner. For example, a 1x1 pixel ROI has a boundary length of 4 and 2 non-adjacent edges so the perimeter is $(4-2\sqrt{2})$. A 2x2 pixel ROI has a boundary length of 8 and 4 non-adjacent edges so the perimeter is $(8-4\sqrt{2})$.

Circularity

Circularity defines the "roundness" of an object. In other words, how similar the object is to a circle. The value is in range of zero to one, whereby one stands for a perfect circle! The circularity of an object is calculated as follows:

$$c = \frac{4 \cdot \pi \cdot AreaSize}{perimeter^2}$$

Centroid

The centroid is the geometrical center of an object. This is the average of the x and y coordinates of all of the pixels in an object. It's coordinates are calculated by the first order spatial moments.

$$c_x = \frac{m_{10}}{m_{00}} c_y = \frac{m_{01}}{m_{00}}$$

Bounding box

The bounding box is the smallest square which can be drawn to include all pixels of the object within the box.

Object ID

During the object detection of a run, ImageC assigns an ID to each detected object, starting with 1 for the first detected object. This object ID is unique throughout the entire run, i.e. an object can be uniquely identified by this object ID.

Using the `With object ID` option of the **Image save** allows to plot the ID beside the detected ROI in the image. Together with the results table, which also allows the ID to be displayed, each region of interest within the image can be matched to its metrics.

Parent Object ID {#parent-object-id}

ImageC allows to build up a hierarchy of objects during a run using the **Reclassify** command. Once an object has been discovered and assigned to an object class, the command can be used to change this class based on some criteria.

One of these criteria is the intersection of the object with an other one. When this option is used, ImageC stores the object ID of the intersecting object (the parent) as the parent object ID together with the object with which the intersection is to be calculated. An object can have exact zero or one parent.

Origin Object ID

Once an object is duplicated using the `ReClassify copy` option of the **Reclassify** command, the ID of the origin object is stored together with the duplicated object. The origin object ID keeps the same even if a duplicated object is again duplicated.

Tracking ID

The Object ID identifies an object uniquely within a run and the parent object ID gives information about the hierarchy of the objects. The tracking ID, on the other hand, is used to link recognized objects that represent the same physical instance.

Example for such "same physical instances" are colocalizing objects from different image channels or moving objects in two different time frames. In the actual version of ImageC, the colocalizing tracking is supported by using the **Colocalization** command.

When using the colocalizing command each object which colocalizes gets the same tracking ID which allows later on to match those objects.

Position

Nr. of intersecting objects

Pipelines

Clicking on an pipeline in the `Pipelines` tab opens the pipeline settings panel.

`../first_steps/images/screenshot_pipelines.png`

The final goal of a pipeline in ImageC is to get objects and classify them. Objects can either be extracted from an image plane based on a region of interest using image processing and classification algorithms, or constructed from an existing set of objects using object math.

A typical way to extract objects from an image is to first define the image plane to be processed, set the classification information for the extracted objects, and add image preprocessing and classification steps. Then, perform some object math and filtering.

Finally, the extracted objects are stored in the resulting database along with their object metrics.

Pipeline settings

The `Pipeline` `name` input allows to give the pipeline a meaningful name. It is recommended to not use the same name for two pipelines in the same project to make failure analyzes much more easier.

Some of the processing steps are used to extract objects from the input images. Each detected object is classified by assigning to exact on class. The `Object` `class` setting allows to specify the default class that will be used for each detected object, unless specified otherwise within a pipeline step.

Pipeline input settings

The `Pipeline` `input` defines the image plane which should be used as a starting point for this pipeline. Mandatory field is the `Image` `channel`.

If the `Z-Stack` setting in the `Project` tab is set to `Intensity` `Projection`, it is necessary to select which intensity projection mode is to be used for the input channel of the images by selecting the `Z-Projection` mode. If the `Z-Stack` setting in the `Project` tab is set to `Each` `one`, nt additional settings have to be taken for the z-stack since each z-stack is processed individually. If the `Z-Stack` setting in the `Project` tab is set to `Exact` `one`, it is necessary to specify which z-stack index to use for analysis, the default being zero (0).

Same settings have to be taken for the time stack depending on taken `T-Stack` settings in the `Project` tab.

In addition to start with an image plane as input it is also possible to start with an `Empty` image. Using a blank image is useful for use cases where you are working with objects from a previous pipeline without having to extract objects again.

Pipeline steps

Commands are used to manipulate the input image, extract objects or to work on exiting objects. A command takes either an image plane or objects as input and returns a manipulated image or objects as output. In other words, based on the command category, the command either works on images or objects, or transforms an image into an object or vice versa.

This brings us to four principal categories of commands: image processing commands (gray), object segmentation commands (white), object classification commands and object post-processing commands (green).

As not every sequence of commands is possible in a meaningful way, ImageC indicates which command can be connected to the previous one.

The goal of each pipeline is to extract regions of interest from an image input plane and store these regions of interest as objects for further object processing. ImageC provides a wide range of commands that can be used for this purpose.

By clicking on the --- + --- button within the pipeline step section, the command selection dialog is opened. Commands are classified into four principal categories: image processing commands (gray), object segmentation commands (white), object classification commands and object post-processing commands (green).

When opening the command selection dialogue, this dialogue shows all the available commands that can be used at this pipeline position. Double-click to insert the command.

A typical pipeline flow might look like this:

1. First of all image processing commands are used to reduce the image noise as much as possible.
2. Object segmentation commands are used to separate foreground from background by using threshold algorithms and convert the grey scale image to a binary image.
3. In the next step an object classification command is used to extract region of interests from the grey scale image and classify them to objects.
4. Object postprocessing and filter commands can be used to do further processing or filtering on the detected objects.

Pipeline history

Using the History button to open the pipeline history. The history list shows the last 64 changes taken in this pipeline. The History tab allows you to go back in time and restore a setting by double-clicking on an entry in the history.

The Tag button can be used to mark the actual settings within the history, which allows an easy restore of these settings.

Metrics

During an analysis ImageC measures a couple of metrics of each detected **object**. All the measured data are stored in a SQL database on disk (internally we are using [DuckDB](#)). Once the analysis is finished the collected data can be displayed within the ImageC results viewer and either be exported to **XLSX** or **R** afterwards.

However, as the number of data generated can be very large, ImageC allows to choose which data to display in the result view. One way to specify the columns to show is using the **class edit** dialog in the classification tab.

When the result file is opened after a run, ImageC displays the result columns as specified there. Even if a column is not specified, the metric is still measured and can be added at any time, even after the analysis has been completed.

Metrics and Statistics {#metrics-and-statistics}

A measurement in the context of ImageC is a metrics calculated during the analysis for an object. The statistics is applied to the selected measurement. The following table gives an overview of the valid combinations of metics and statistics which are available.

Plate view

| Measurement | Statistics | Description |
|--------------|------------|--|
| Count | CNT | The average of the object number per image in the well. |
| Intersection | CNT | The average of the number of objects A intersecting with object class B per image in the well. |
| Area size | AVG | The average of the average object area sizes per image in the well. |
| Area size | MEDIAN | The average of the median of the object area sizes per image in the well. |
| Area size | MIN | The average of the minimum of the object area sizes per image in the well. |
| Area size | MAX | The average of the maximum of the object area sizes per image in the well. |
| Area size | STDEV | The average of the standard deviations of the object area sizes per image in the well. |
| Area size | SUM | The average of the sum of the object area sizes per image in the well. |
| Area size | CNT | The average number of objects per image in the well used to calculate the statistics. |
| Perimeter | AVG | The average of the average object perimeters per image in the well. |
| Perimeter | MEDIAN | The average of the median of the object perimeters per image in the well. |

| Measurement | Statistics | Description |
|------------------|------------|---|
| Perimeter | MIN | The average of the minimum of the object perimeters per image in the well. |
| Perimeter | MAX | The average of the maximum of the object perimeters per image in the well. |
| Perimeter | STDEV | The average of the standard deviations of the object perimeters per image in the well. |
| Perimeter | SUM | The average of the sum of the object perimeters per image in the well. |
| Perimeter | CNT | The average number of objects per image in the well used to calculate the statistics. |
| Circularity | AVG | The average of the average object circularities per image in the well. |
| Circularity | MEDIAN | The average of the median of the object circularities per image in the well. |
| Circularity | MIN | The average of the minimum of the object circularities per image in the well. |
| Circularity | MAX | The average of the maximum of the object circularities per image in the well. |
| Circularity | STDEV | The average of the standard deviations of the object circularities per image in the well. |
| Circularity | SUM | The average of the sum of the object circularities per image in the well. |
| Circularity | CNT | The average number of objects per image in the well used to calculate the statistics. |
| Intensity | AVG | The average of the average object intensities per image in the well. |
| Intensity | MEDIAN | The average of the median of the object intensities per image in the well. |
| Intensity | MIN | The average of the minimum of the object intensities per image in the well. |
| Intensity | MAX | The average of the maximum of the object intensities per image in the well. |
| Intensity | STDEV | The average of the standard deviations of the object intensities per image in the well. |
| Intensity | SUM | The average of the sum of the object intensities per image in the well. |
| Intensity | CNT | The average number of objects per image in the well used to calculate the statistics. |
| Center of mass X | | Not practicable in this view. |
| Center of mass Y | | Not practicable in this view. |
| Object ID | | Not practicable in this view. |
| Origin object ID | | Not practicable in this view. |
| Parent object ID | | Not practicable in this view. |
| Tracking ID | | Not practicable in this view. |

Well view

| Measurement | Statistics | Description |
|--------------|------------|--|
| Count | CNT | The number of detected objects in the image. |
| Intersection | CNT | The number of objects A intersecting with object class B in the image. |
| Area size | AVG | The average object area sizes in the image. |
| Area size | MEDIAN | The median of the object area sizes in the image. |
| Area size | MIN | The minimum of the object area sizes in the image. |
| Area size | MAX | The maximum of the object area sizes in the image. |
| Area size | STDEV | The standard deviations of the object area sizes in the image. |
| Area size | SUM | The sum of the object area sizes in the image. |
| Area size | CNT | The number of objects in the image used to calculate the statistics. |
| Perimeter | AVG | The average object perimeters in the image. |
| Perimeter | MEDIAN | The median of the object perimeters in the image. |
| Perimeter | MIN | The minimum of the object perimeters in the image. |
| Perimeter | MAX | The maximum of the object perimeters in the image. |
| Perimeter | STDEV | The standard deviations of the object perimeters in the image. |
| Perimeter | SUM | The sum of the object perimeters in the image. |
| Perimeter | CNT | The number of objects in the image used to calculate the statistics. |
| Circularity | AVG | The average object circularities in the image. |
| Circularity | MEDIAN | The median of the object circularities in the image. |
| Circularity | MIN | The minimum of the object circularities in the image. |
| Circularity | MAX | The maximum of the object circularities in the image. |
| Circularity | STDEV | The standard deviations of the object circularities in the image. |
| Circularity | SUM | The sum of the object circularities in the image. |
| Circularity | CNT | The number of objects in the image used to calculate the statistics. |
| Intensity | AVG | The average object intensities in the image. |
| Intensity | MEDIAN | The median of the object intensities in the image. |
| Intensity | MIN | The minimum of the object intensities in the image. |
| Intensity | MAX | The maximum of the object intensities in the image. |
| Intensity | STDEV | The standard deviations of the object intensities in the image. |

| Measurement | Statistics | Description |
|------------------|------------|--|
| Intensity | SUM | The sum of the object intensities in the image. |
| Intensity | CNT | The number of objects in the image used to calculate the statistics. |
| Center of mass X | | Not practicable in this view. |
| Center of mass Y | | Not practicable in this view. |
| Object ID | | Not practicable in this view. |
| Origin object ID | | Not practicable in this view. |
| Parent object ID | | Not practicable in this view. |
| Tracking ID | | Not practicable in this view. |

Image view

| Measurement | Statistics | Description |
|--------------|------------|--|
| Count | CNT | Not practicable in this view (always 1). |
| Intersection | CNT | The number of objects intersecting with this object. |
| Area size | AVG | The object area size. |
| Area size | MEDIAN | The object area size. |
| Area size | MIN | The object area size. |
| Area size | MAX | The object area size. |
| Area size | STDEV | The object area size. |
| Area size | SUM | The object area size. |
| Area size | CNT | Not practicable in this view (always 1). |
| Perimeter | AVG | The object perimeters. |
| Perimeter | MEDIAN | The object perimeters. |
| Perimeter | MIN | The object perimeters. |
| Perimeter | MAX | The object perimeters. |
| Perimeter | STDEV | The object perimeters. |
| Perimeter | SUM | The object perimeters. |
| Perimeter | CNT | Not practicable in this view (always 1). |
| Circularity | AVG | The object circularity. |
| Circularity | MEDIAN | The object circularity. |
| Circularity | MIN | The object circularity. |
| Circularity | MAX | The object circularity. |
| Circularity | STDEV | The object circularity. |
| Circularity | SUM | The object circularity. |
| Circularity | CNT | Not practicable in this view (always 1). |
| Intensity | AVG | The object (min/max/avg/sum) intensity. |
| Intensity | MEDIAN | The object (min/max/avg/sum) intensity. |

| Measurement | Statistics | Description |
|------------------|------------|---|
| Intensity | MIN | The object (min/max/avg/sum) intensity. |
| Intensity | MAX | The object (min/max/avg/sum) intensity. |
| Intensity | STDEV | The object (min/max/avg/sum) intensity. |
| Intensity | SUM | The object (min/max/avg/sum) intensity. |
| Intensity | CNT | Not practicable in this view (always 1). |
| Center of mass X | | X coordinates of the center of mass of the object in the image. |
| Center of mass Y | | Y coordinates of the center of mass of the object in the image. |
| Object ID | | Unique object ID. |
| Origin object ID | | If it was copied by reclassify this is the Object ID of the object this object was copied from. If not copied the value is 0. |
| Parent object ID | | If the object was reclassified using intersection, this is the Object ID of the object this object was intersecting with. If not reclassified the value is 0. |
| Tracking ID | | If the object has been tracked e.g. by coloc or timeframe, all related objects will have the same tracking ID. The table is sorted by Tracking ID. |

Commands

Overview

Image processing

Image processing commands work on **image planes** manipulating the input image plane and forwarding the newly image to the command output. ImageC provides a set of image processing algorithm for different use cases.

In general, image preprocessing serves to minimize the image noise as far as possible (maximizing the signal-to-noise ratio) in order to enable separation between the background and the signal. An **object segmentation command** is issued after the image preprocessing.

Tip. Image processing commands get an image as input and have a manipulated image as output.

Object segmentation

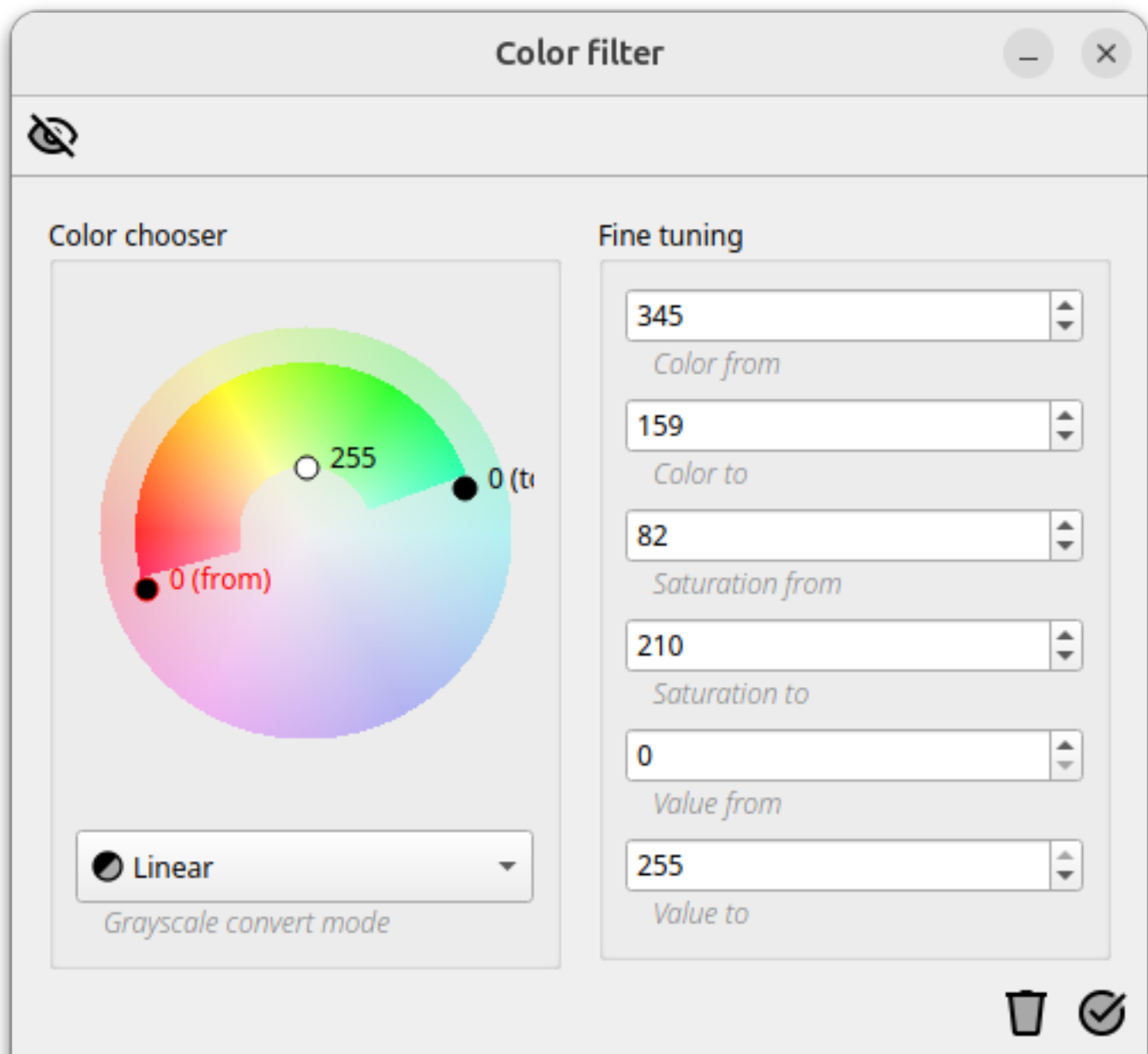
Once objects are segmented, the resulting region of interests have to be classified. Classification in ImageC is the process of converting regions of interest into objects, assigning each object to a class and calculating object metrics.

ImageC provides an object classifier based on segmented images (e.g. after applying a threshold) and an AI classifier which can directly be applied to an image without the need of segmentation before.

Tip. See chapter **classification** for understanding the fundamentals.

Image processing

Color filter



If the input image is an RGB 8-bit color image, the color image must first be converted to a grey scale image. This conversion is done by the color filter command. With this filter it is possible to define the color range containing the region of interests. For each region of interest in a different color to extract a separate pipeline with a separate color filter must be created.

The resulting image is a gray scale image showing only the image regions filtered by the selected color range. In a next step further image processing steps can be applied.

For color images, three values are stored per pixel, which contain the color information. The RGB-format is one widely used format used to store these color information. RGB stores information about the red, green and blue parts of an image and is based on how the human eye perceives color. The superimposition of these color components leads to our perception of color.

In the RGB color space, it is easy to create a color tone that is correct for human perception, but working with color ranges is difficult. A better suited format for working with color ranges is the HSV format. Instead of storing the color information using red, green, blue, in HSV format a color is defined by its color tone (hue), its saturation and its brightness (value).

Using HSV allows more easy to filter e.g. all blue tones from an image. ImageC is using HSV color format for that reason when working with colored images.

Wikipedia: https://de.wikipedia.org/wiki/HSV-Farbraum#/media/Datei:HSV_cone.png

Blur

Blur algorithms are used to reduce image noise and details.

ImageC provides two different blur algorithms, "normal" blur and Gaussian blur. For both the filter kernel size can be set. The larger the kernel, the more details are removed from the image.

Gaussian blur uses a Gaussian function to blur the image. Compared to the normal blur gaussian blur tends to preserve edges slightly better and avoids sharp transitions between blurred regions.

Tip. Bigger filter kernels removes more noise and details from the image. Use gaussian blur to preserve edges in a better way than normal blur.

In image processing, a kernel, convolution matrix, or mask is a small matrix used for blurring, sharpening, embossing, edge detection, and more. This is accomplished by doing a convolution between the kernel and an image. Or more simply, when each pixel in the output image is a function of the nearby pixels (including itself) in the input image, the kernel is that function.

““ [https://en.wikipedia.org/wiki/Kernel_\(image_processing\)](https://en.wikipedia.org/wiki/Kernel_(image_processing))

Intensity

The intensity command allows to change the image brightness, contrast and gamma. The contrast value is a factor between one and three. One means no change in contrast, three means a contrast increase by a factor of three.

For brightness a range of -32768 and +32768 can be specified. By adding this value to each pixel value in the image, the brightness is increased by the specified value.

Warning. Gamma correction is not yet supported by ImageC.

In addition to setting manual image correction factors, an automatic correction mode can be selected. Automatic contrast and brightness correction is done by calculating a histogram equalizing.

Histogram equalizing is done in three steps: Step 1: Calculate the histogram (65536 bins for 16-bit range) Step 2: Calculate the cumulative distribution function (CDF)

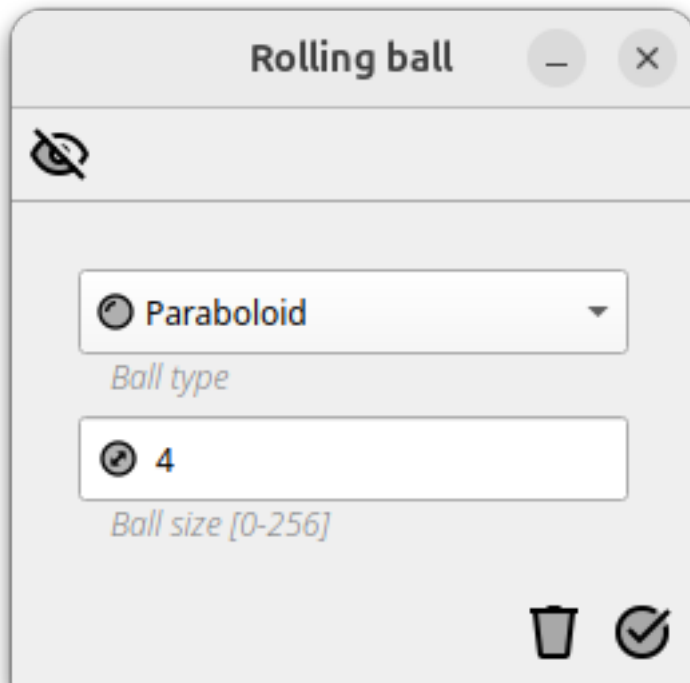
$$cdf[i] = cdf[i - 1] + hist[i];$$

Step 3: Normalize the CDF to the range [0, 65535]

$$equalization_{map}[i] = \frac{65535.0 \cdot cdf[i]}{total_{pixels}}$$

Step 4: Map the pixel values from the equalization map

Rolling ball



The rolling ball algorithm is a background subtraction algorithm, with the goal to remove most of the background noise.

This is achieved by creating a local background around a virtual ball. Within the radius of the ball, the average intensity is calculated. This is done for the entire image. The result is subtracted from the original image.

This port from ImageJ contains two ball types, a spherical ball and a paraboloid. A paraboloid handles edges more gently, reducing the appearance of artifacts along boundaries. This is beneficial in applications such as microscopy, where accurate boundary representation is essential.

Best practice. The radius of the ball should be set to at least the size of the largest object that does not belong to the background.

Note. The implementation in ImageC was taken from the original ImageJ and ported to C++ based on the NIH Image Pascal version by Michael Castle and Janice Keller of the University of Michigan Mental Health Research Institute.

The rolling-ball algorithm was inspired by Stanley Sternberg's article, "Biomedical Image Processing", IEEE Computer, January 1983.

Cytocomputer applications

Gray-scale morphological processing. Two closely spaced spots can often appear in the child's gel when each parent's gel displays only one. Here, the integrated density of each spot in the pair is half that of the parents. Thus, we place a genetic constraint on the processing of the gels by using a lower threshold on genetically screened proteins so that protein concentrations reduced by half are clearly discernible above background noise. The problem in implementing this step directly on the gel images is that the background intensity level, on which the spots appear, is not uniform over a gel and varies between gels. Thus, we must remove the background from the gel image before taking the threshold.

Programs have been developed that efficiently estimate the background level across a gel image. Subtracting the estimated background image from the original gel image gives the background-normalized gel image. The process of background estimation is an extension of the binary image opening process previously described. The only difference is that the digital spot images in the first example are arrangements of pixels whose values are either 1 or 0, and a gray-scale digital image has pixel intensity values of anywhere between 0 and 255. In defining gray-scale neighborhood transformations, we need to view the image as a set of "boxels," or cubical pixels in a 3-D volume. This representation, called the umbra of a gray-scale image, consists of rows and columns of vertical piles of boxels; the height of the pile (number of boxels) at position x,y in the umbra is equal to the gray level of the pixel at position x,y in the gray-scale image.

The umbra representation of a 2-D electrophoretic gel can be visualized as an extraterrestrial landscape of tall peaks and narrow ridges. The composited images in Figure 8 illustrate the umbra representation through appropriate shading and shadowing processing that makes the height interpretation of gel gray levels more visually apparent. Shading and shadowing, which are also implemented morphologically by neighborhood transforms, are detailed elsewhere.¹²

The gray-scale opening is a gray-scale erosion followed by a gray-scale dilation, expressed in terms of a gray-scale structuring element. This opening process is illustrated in Figure 9 for the previously shown gel section. The gray-scale gel image in Figure 9a is opened by the structuring element shown in Figure 9b to produce the image in Figure 9c, referred to as the background image. Figure 9d is the background-normalized image that results from subtracting the background image from the original.

The process of opening a gray-scale image by a gray-scale structuring element to produce the background image is understood in terms of the gray-scale image umbra. The opening of an umbra of a gray-scale gel image by a gray-scale structuring element is the union of all

that can be covered by at least one position of the translating sphere. The sphere diameter is selected so that it is considerably wider than any of the peaks formed by the gel spots and thus cannot enter the interior of the peaks. However, the diameter of the spherical structuring element is small enough to follow the smooth contours of the changing background intensity. The background is smooth with respect to the sphere, but the peaks are not.

A large digital sphere appropriately shaded is shown in Figure 11. The sphere is produced as the dilation of the umbra of 26 3-D structuring elements, each structuring element being a subset of a $3 \times 3 \times 3$ window. The sphere erodes or dilates a gray-scale image by a sequence of eroding or dilating neighborhood transformations, the neighborhood transformation being determined by the 3-D structuring elements composing the sphere.

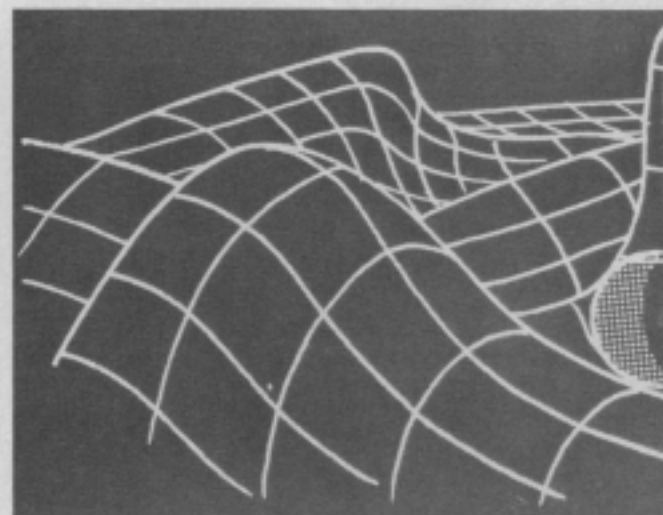
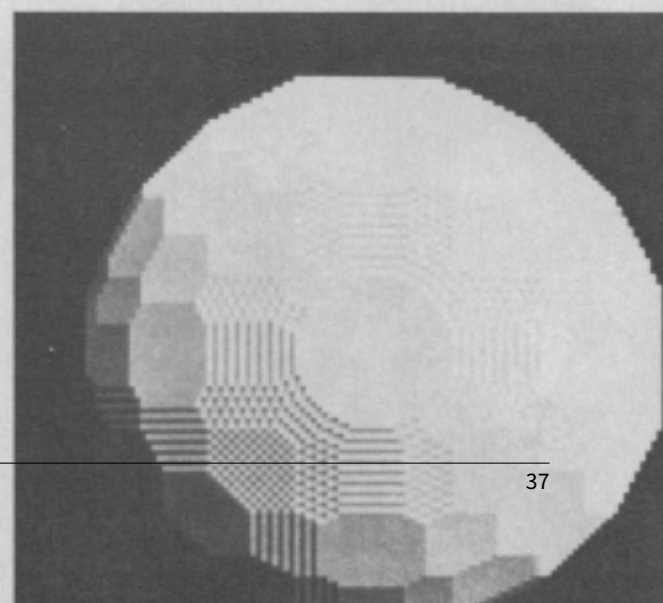


Figure 10. Schematic representation of the rolling ball method for background normalization. The ball follows the background contours but does not penetrate the spot peaks. This is equivalent to eroding and dilating by a spherical structuring element.



Median subtract

Deprecated. Use [Rank filter](#) in combination with [Image math](#) instead for future projects!

Median subtraction is a sort of noise filtering which can be used to reduce background noise similar to the rolling ball algorithm. Behind the scenes a rank filter is used to calculate the median of the image intensity, which is subtracted afterwards from the original image.

The kernel size defines the window size used for the rank filter to calculate the median. Bigger kernels reduces more details from the image than smaller ones.

Edge detection

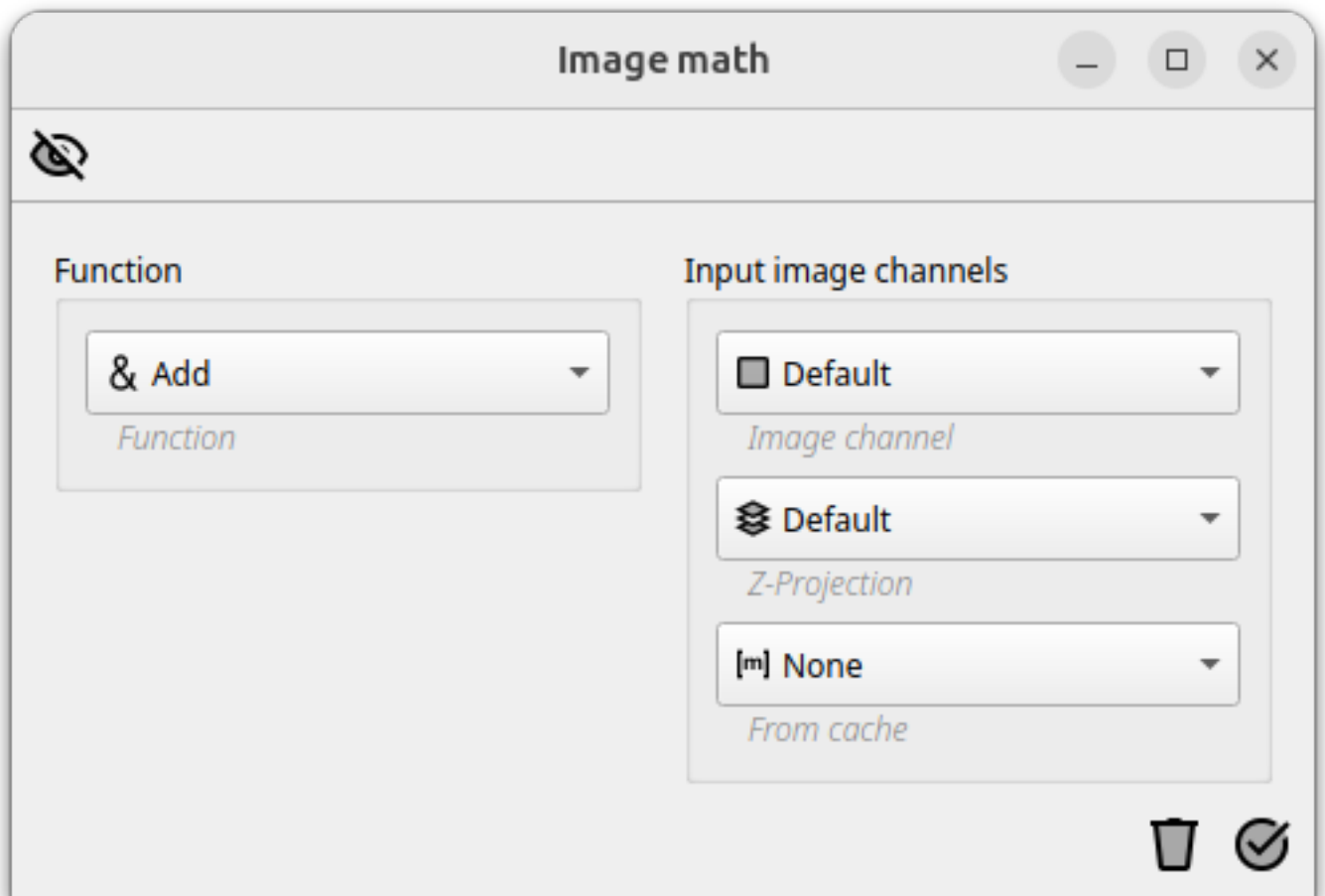
Edge detection is a function used to highlight the edges within an image. An edge is defined as a change in grey level. Larger rates of change in the values lead to a larger output value in the algorithm.

ImageC provides Sobel and Canny edge detection.

The main advantages of the Sobel operator are that it is simple and time efficient but produces rough edges. On the other hand, the Canny technique produces smoother edges due to the implementation of non-maxima suppression and thresholding. The disadvantage of the Canny algorithm is that it is more complex and less time efficient than Sobel.

Edge detection is the process of finding edges in an image and converting them to a gradient representation. Sharper changes in the intensity values lead to a higher gradient.

Image math



The image math command can be used to combine two images using one of the provided mathematical operations. Input image one is the image from the last pipeline step, the second image is the image configured in the Input image channel settings. For the second image either an image channel can be used or an image stored in the cache from a previous pipeline. If From cache is not None the channel settings are ignored and the image from the selected cache slot is loaded.

The Operating order setting allows to define if the first operand of the selected function is the image from the last pipeline step or the selected input image.

An image is represented as a matrix of pixel intensity values. When calculating with images a matrix operation is performed which uses the pixel value at (n, m) from both matrixes and applying the operator on that. The resulting matrix after applying the operator on all elements of the matrix is the resulting image.

Image 1

| | | | |
|---|---|---|---|
| 7 | 8 | 1 | 3 |
| 0 | 0 | 9 | 3 |
| 5 | 6 | 5 | 8 |
| 4 | 4 | 8 | 6 |

+

Image 2

| | | | |
|---|---|---|---|
| 8 | 4 | 3 | 9 |
| 6 | 0 | 0 | 2 |
| 1 | 2 | 2 | 3 |
| 7 | 8 | 5 | 4 |

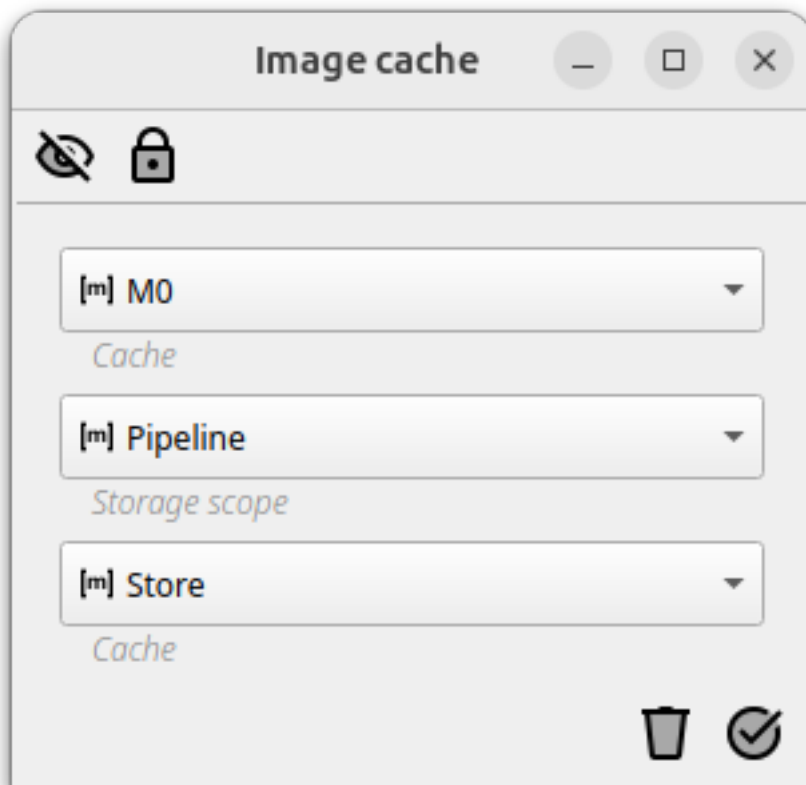


| | | | |
|----|----|----|----|
| 15 | 12 | 4 | 12 |
| 6 | 0 | 9 | 5 |
| 6 | 8 | 7 | 11 |
| 11 | 12 | 13 | 10 |

Result

Text is not SVG - cannot display

Image Cache



The store image to cache command stores an actual image at any time to the cache. Image cache allows using a preprocessed image as base for further preprocessing steps in an other pipeline or in the same pipeline in a future step. Possible applications might be to preprocess a background for subtraction and use this preprocessed background together with the image math subtract command in other pipelines to remove it.

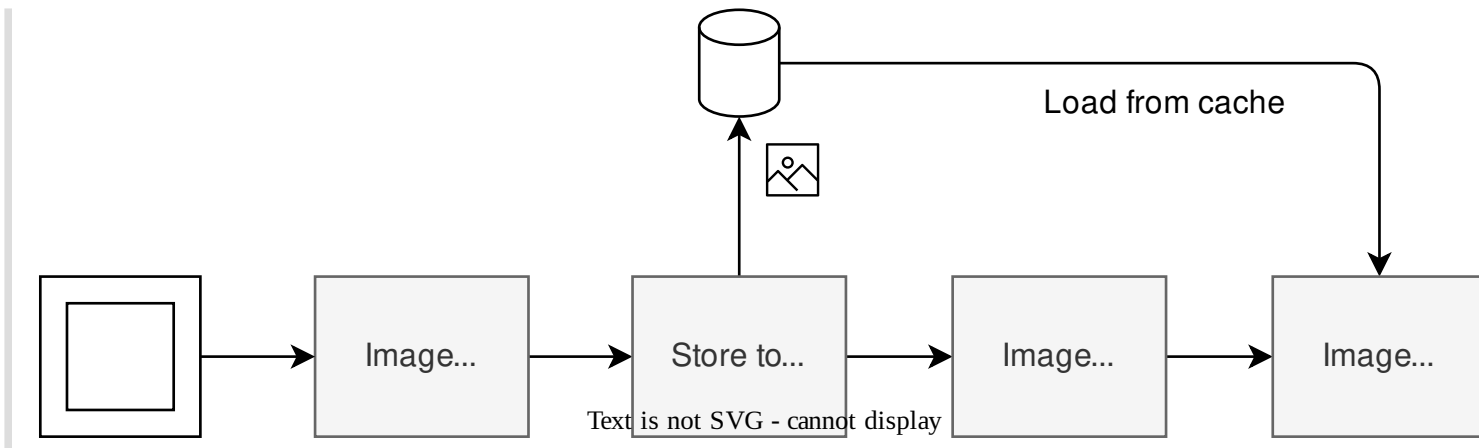
Storage scope

The storage scope defines either if the cached image is only available in the pipeline the command is used `Iteration` or if the stored image can also be used across different pipelines `Pipeline`.

The memory slots `M0` to `M10` can be used twice once for `Iteration` and once for `Pipeline` cache.

Note. Once an image is stored to the cache it is not edited any more. Loading from the cache creates a new local copy of the image and the pipeline is working on this local copy but not on the cached image.

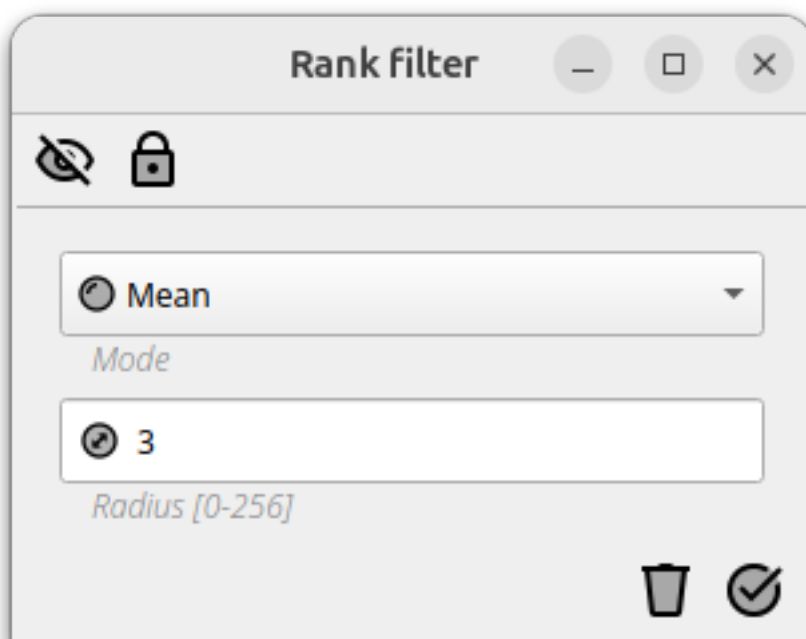
ImageC cache is reserved place in the RAM of the computer. Storing an image to the cache stores the actual image to this place in RAM. Other commands can load these images from the RAM in an efficient way and apply further processing steps on it.



Enhance contrast

Under construction. Coming soon!

Rank filter



With ranking filters, the grey values of the pixels in a defined area around a pixel are collected, sorted by size and ranked. A grey value is then selected from this sorted list to replace the grey value of the current pixel.

The **Radius** setting defines the size of the area to collect the pixels gray values in.

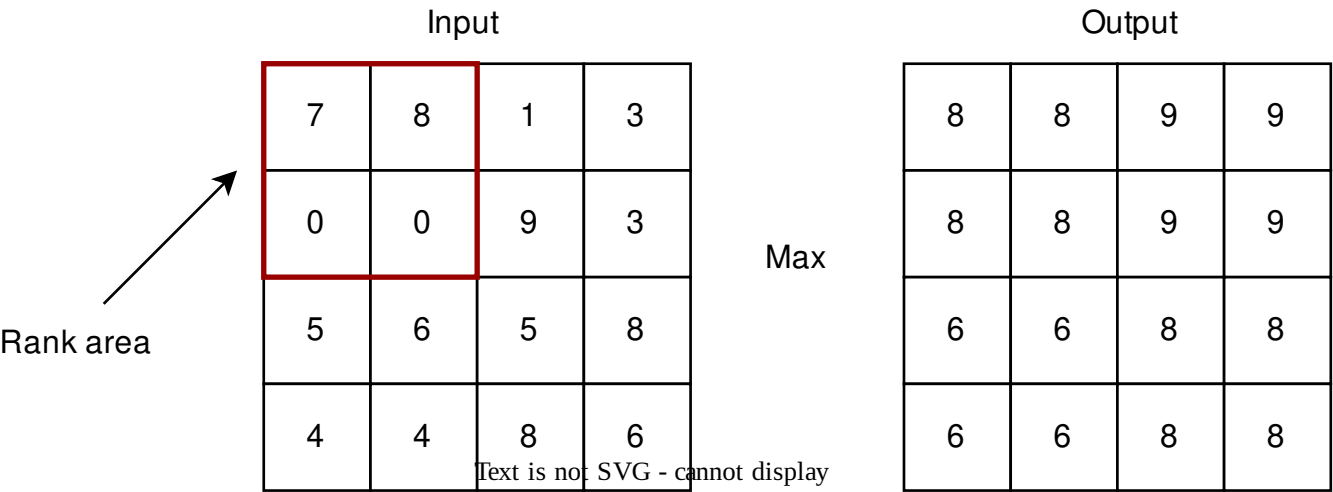
Tip. Bigger filter kernels removes more noise and details from the image.

Rank-order filters belong to the class of non-linear filters in digital image processing. These are filters that cannot be described by a convolution.

With ranking filters, the gray values of the pixels in a defined area around a pixel are collected, sorted by size and ranked. A gray value is then selected from this sorted list to replace the gray value of the current pixel.

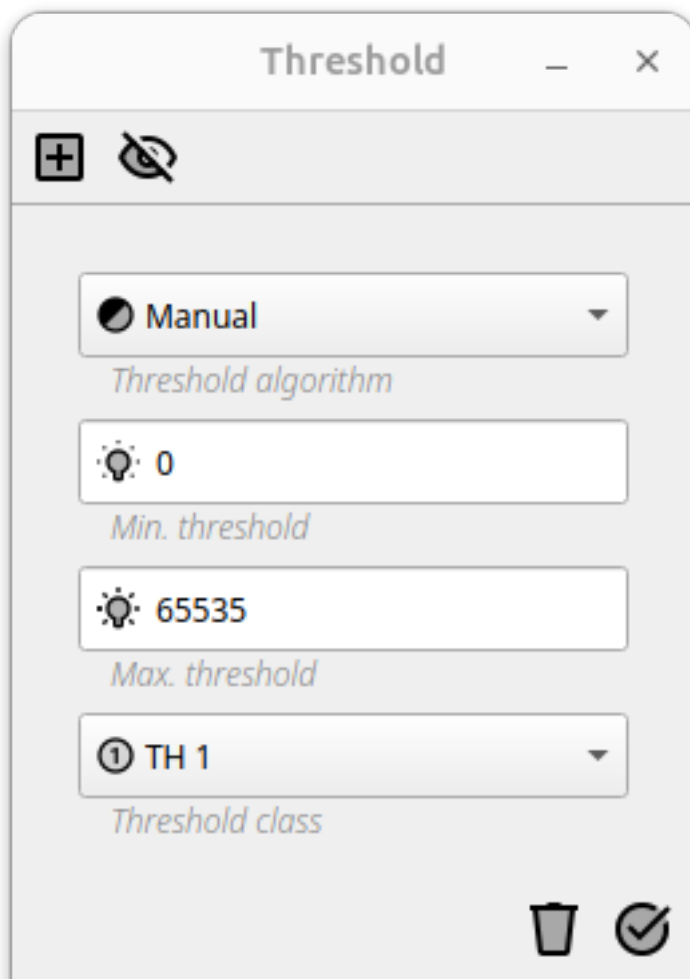
The choice of position determines the type of ranking filter. With ascending sorting, you get the:

- Minimum filter, for the minimum gray value, first position in the list
- Median filter, for the gray value in the middle of the list
- Maximum filter, for the maximum gray value, last position in the list.



Binary image processing

Threshold



Threshold is an object segmentation method with the goal to separate foreground objects from background objects. Applying a threshold to an image converts the grey scale image to a binary image. All pixels with an intensity value below the selected threshold value are converted to zero (black), the others are to 65535 (white).

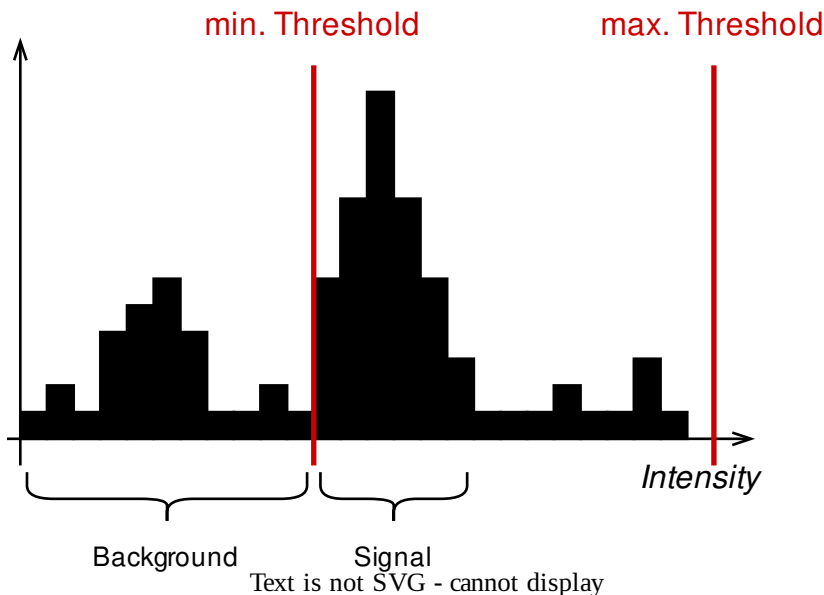
Using the correct threshold will have a significant effect on the quality of the result. The challenge is to find a threshold value that is high enough to suppress the background noise but not too high as to not remove objects of interest. Looking on the histogram of an image can help to find a good starting point when using Manual threshold value.

Set the minimum threshold value to the smallest intensity value of the signal (most left value after the signal in the histogram).

Tip. Using the [icon_external_link](#) button to open the advanced preview. The histogram of the image is displayed in the left image pane.

Best practice. Set the minimum threshold value to the smallest intensity value of the signal (most left value after the signal in the histogram) as a good starting point.

Thresholding is a technique that aims to achieve an intensity value that is higher than the highest intensity value of the background noise and low enough to achieve the lowest intensity value of the regions of interest (ROIs). Image preprocessing helps in advanced to reduce the background noise to get a sharp intensity border between background and foreground.



See also section [threshold filter](#)

Maximum threshold

Using a minimum thresholds sets the minimum signal intensity to segment an object.

However there may be use cases having different objects in an image with different signal intensity values. Using a max threshold allows to limit the maximum signal intensity for segmenting an object too. Together with the minimum threshold, the maximum threshold can be used either for more detailed object differentiation or, for example, to extract the background instead of the objects when the minimum threshold is set to zero.

Tip. A object is only segmented if the signal value is between minimum threshold value and maximum threshold value.

Threshold classes {#threshold-classes}

In addition to only separating background and foreground using a minimum and maximum threshold, ImageC also allows more than one threshold to be applied to an image.

Use the `{{icon_add}}` button in the Threshold dialogue box to define additional thresholds for the image. A threshold class must be selected for each defined threshold in order to distinguish between objects segmented with different threshold values.

Under the hood, ImageC assigns each threshold class to a well-defined pixel value in the binary image, starting from 65535 (full white) down to one. Based on this value the later on classifier is able to distinguish the segmented objects.

Tip. Multiple thresholds are assigned to threshold classes, allowing different objects to be detected depending on their signal strength.

Auto threshold

Using a manual threshold applies the same threshold for all images of your set.

If the exposure ratios of the images are very different, it is possible that no uniform threshold value can be found for all images. You can choose one of ImageC's provided auto-threshold algorithms for this. The selected algorithm attempts to identify a minimum threshold value through the analysis of statistical properties inherent to the image in question. The precise methodology employed is contingent upon the specific algorithm utilized.

Minimum threshold in combination with auto threshold

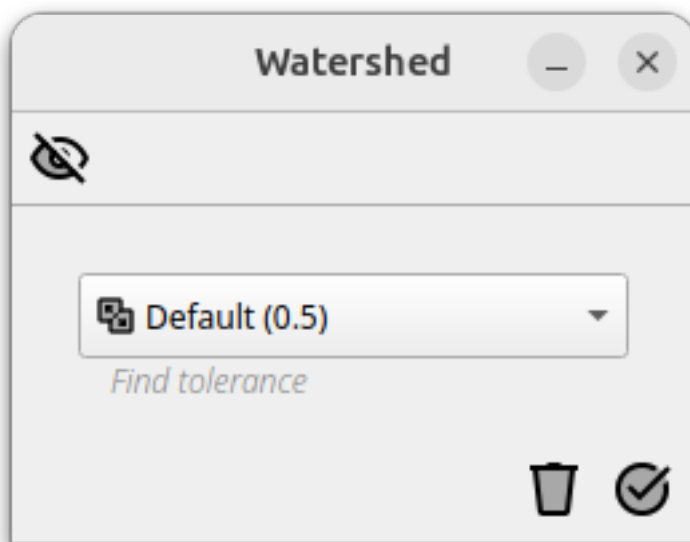
Auto thresholds can be effective if the preprocessing steps are sufficiently robust in removing background noise. However it is a common practice in research to include control images devoid of any objects as samples. Since the auto threshold algorithms are designed to find an upper bound that separates background from signal, these algorithms will also find such a bound in an empty image due to noise. Applying an auto threshold algorithm to an empty noise image will lead to the segmentation of noise.

It is therefore strongly advised that a minimum threshold value exceeding zero is set, even when an auto threshold method is employed. ImageC will use the chosen minimum threshold as a lower bound. If the auto threshold algorithm calculates a threshold value below the lower bound, ImageC will set it to the minimum value.

It is advisable to select an empty control image (an image without an object) from the list. By using the live preview and the histogram, it is possible to determine the intensity value of the background noise and set a suitable minimum value for the threshold.

Best practice. In the event that an auto threshold algorithm is employed, it is strongly recommended to establish a minimum threshold. This is to prevent the occurrence of false positive detection in images that lack objects, given that such images would otherwise be included in the analysis.

Watershed



The watershed algorithm employs a process of object segmentation/separation based on the intensity values of the objects in question. In this context, the intensity values are interpreted as altitude, with the objective of identifying the valleys between the peaks.

In applications with a high object density, it is no exception that two objects come very close or even touch each other. When using threshold techniques for object segmentation, it is not possible to distinguish between two such near objects anymore.

Some algorithms have been developed with the specific aim of addressing this issue. One of them is the Watershed algorithm which was ported from [ImageJ](#) to ImageC for that reason.

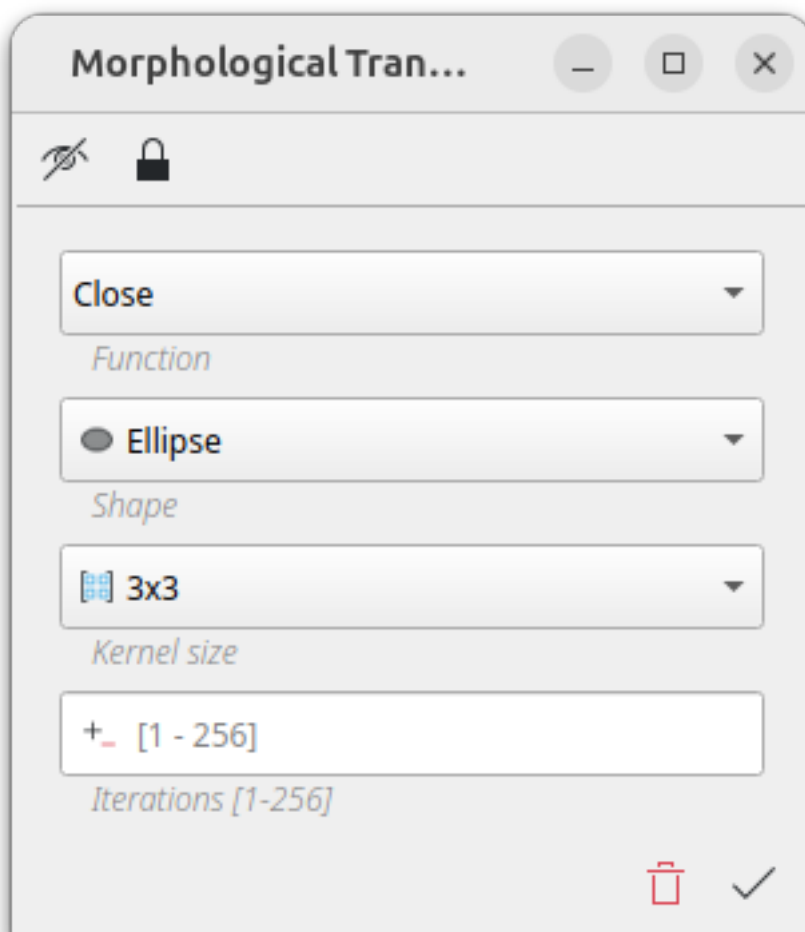
Tip. Use the tolerance value to define on which percentage of the extracted intensity level an object border is detected.

Warning. It is recommended that the watershed be activated only when necessary, as it is a highly complex algorithm that significantly reduces analysis time when activated.

Based on peaks extracted from the intensity values the valleys between the peaks are the borders for splitting objects.



Morphological transform



Morphological transformation is an operation applied to the image shape. Usually, binary images are used as input for this operation but can also be applied to gray scale images. ImageC supports the basic operations: erosion, dilation, opening and closing beside a couple of experimental operations.

A structuring element is applied to the input image performing the selected operation getting the output image. The output image pixel values are calculated by a comparison of the corresponding input image pixels with its neighbors.

A "structural element" is a shape which is used to identify the pixel to be processed and the neighboring pixels used during the process. The structuring element form is chosen according to the shape of the object which should be processed in the input image.

The centre of the structuring element is called the origin. It identifies the pixel being processed. The origin need not always be the centre but might be even outside of the structuring element.

ImageC provides following options which allows to define the structural element and the algorithm settings:

The Shape defines the form of the structural element to use. Theoretically any form can be used. ImageC actually supports: Circle, Square and Cross as predefined elements to select.

The Kernel size setting is used to define the size of the area looking for neighbor pixels.

With the iteration option the number of runs of the algorithm can be defined. Default is one run.

Erosion

Erosion removes the boundaries of an object. The core of the object remains as the result.

- In a binary image, a pixel is set to 1 if all of the neighboring pixels have the value 1.
- In a gray scale image, the value of the output pixel is the minimum value of all pixels in the neighborhood.

Erosion can be used to separate connected objects or remove noise from the image.

Dilation

Under construction. Coming soon!

Opening

Under construction. Coming soon!

Closing

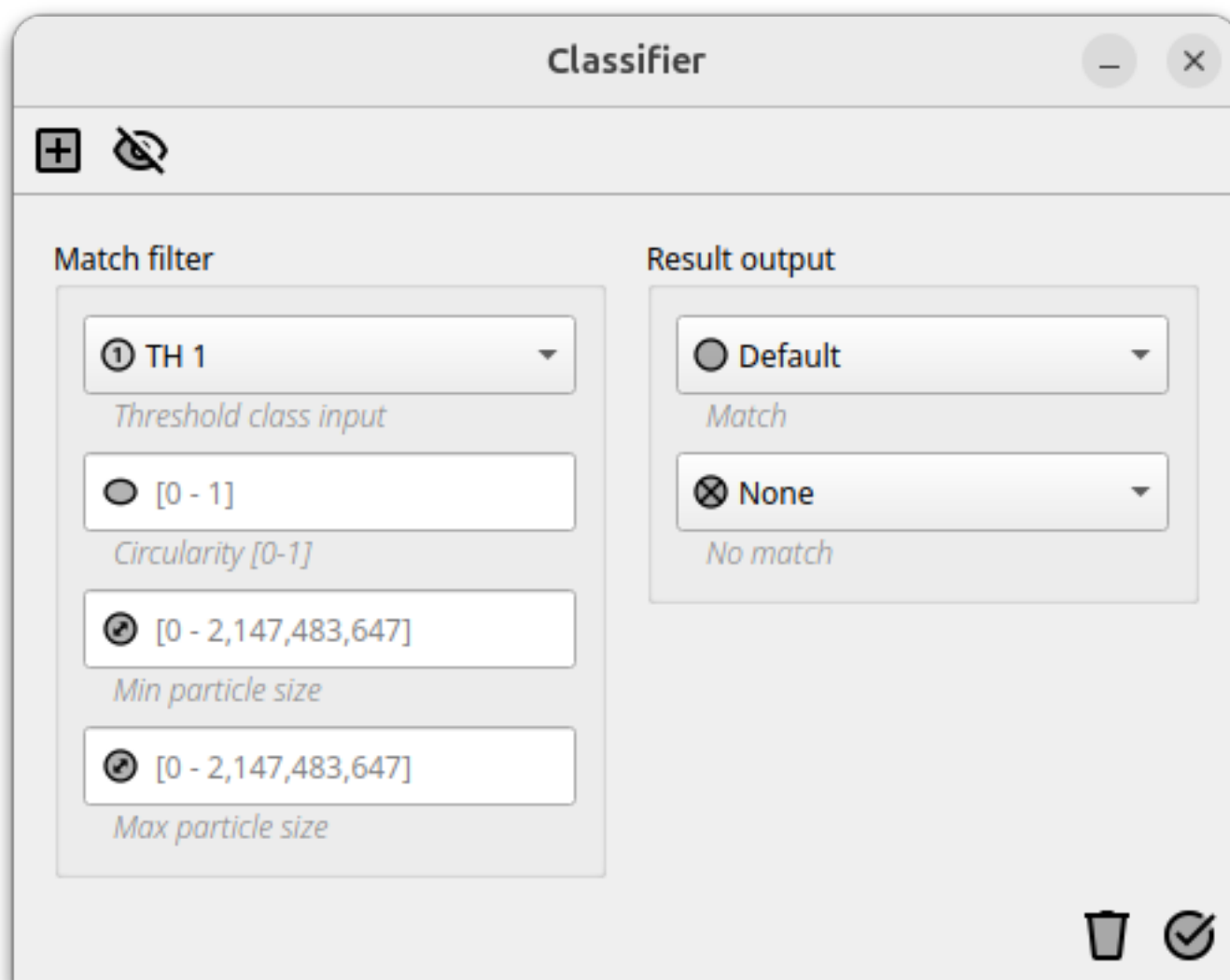
Under construction. Coming soon!

Fill holes

Under construction. Coming soon!

Classification

Classifier

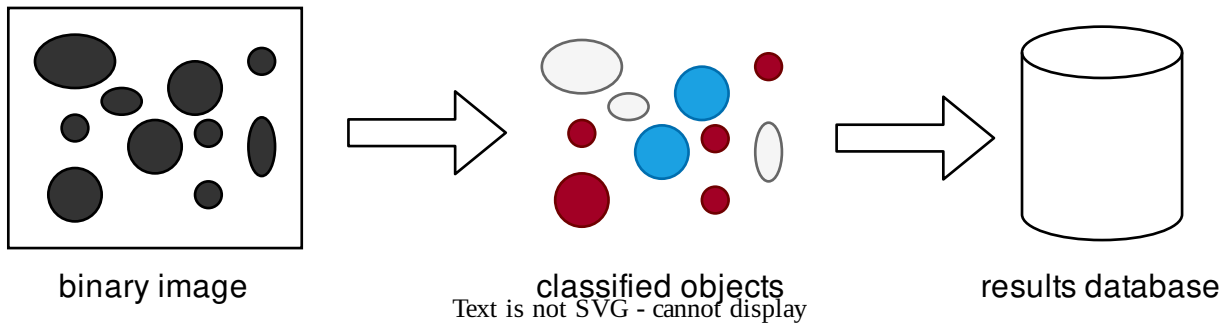


The Classifier command is used to convert regions of interest, formally extracted from the image using a segmentation algorithm such as **threshold**, into objects, where each object is assigned to an object class. For **classified** objects metrics are calculated, they are stored to the resulting database and can be used in further object-processing steps.

ImageC's classifier provides several filter methods which can be used to filter out artefact. All objects which match the filter criteria are assigned to the match class, objects not matching the filter criteria are assigned to the no match. The filter can be set for each **threshold classes** individually. This allows the classifier to assign each ROI of a **threshold classes** to a specific match and no match object class. Use the `{{icon_add}}` button to add as many filters as needed. As a rule, one filter for each threshold class is used.





Tip. See section **Objects** for more detailed information about object metrics.

Objects are extracted from a binary image representing the regions of interest of the image. For each extracted object a class is assigned and metrics are measured. Objects are stored to the resulting database and can be used in further pipeline steps.



AI classifier

AI Classifier

Model settings


Class 0

Class 1

Model settings

Cell Detector
▼

Model path


Onnx
▼

Model format

U
U-Net
▼

Model architecture

2
▲▼

Nr. of model classes


Input parameters

640
▼

Input width of the model

640
▼

Input height of the model


Color
▼

Input channels of the model

Thresholds

☒
0.8
▼

Mask threshold (0.8)

☒
0.5
▼

Class threshold (0.5)

Model details

Cell Detector v0.3.0

IN:

input: [b c y x] [1 3 640 640]

OUT:

diams: [b] [1]

flows: [b c y x] [1 8 160 160]

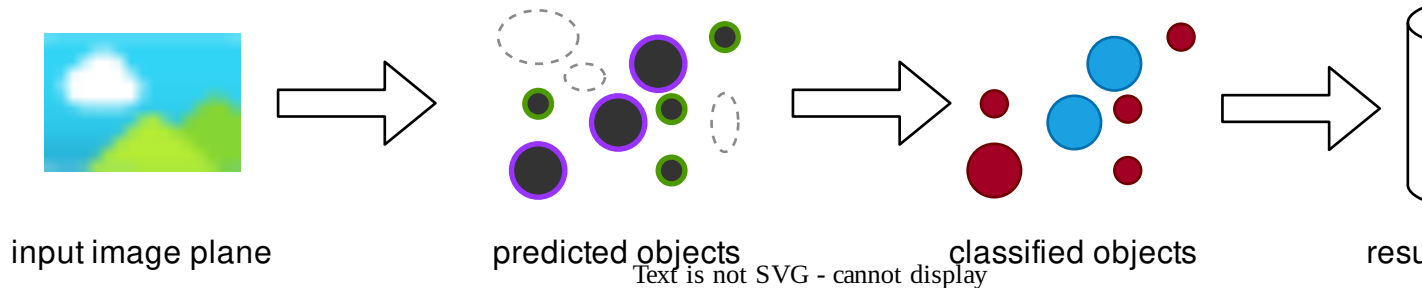
masks: [b y x] [1 160 160]

styles: [b y] [1 256]

"University of Salzburg/Joachim Danmayr

In some cases, it may not be feasible to differentiate between the background and the signal based solely on intensity values. This makes the use of threshold as a segmentation technique impractical in such situations. To overcome this limitation ImageC allows the use of artificial intelligence models for object segmentation and classification as an alternative to threshold techniques.

The image plane is forwarded to a pre-trained AI model. The output is a prediction whereby each prediction is assigned to an output class and a confidence. Using the ImageC AI classifier the predicted output is transformed to classified objects.



Deep learning models

On startup ImageC searches the `./models` folder for compatible AI models and presents them in the drop-down menu for selection. In the actual implementation of ImageC [Yolov5](#) and [U-Net](#) model architecture are supported.

ImageC expects a `rdf.yaml` file beside the weight file. This resource definition file according to the bio image zoo schema [rdf definition](#) describes the model input and output parameters required by ImageC for correct interpretation of the prediction output.

After downloading a new model, either from [ImageC AI models](#) or [BioImage Model Zoo](#), unzip the content and copy the unpacked folder to the `./models` directory of the ImageC installation directory. Press the refresh button and select the model from the `model path` dropdown.

Most of the settings are taken automatically based on the parsed information of the `rdf.yaml` file. For non onnx models the number of model classes must be set manually.

Best practice. Select the number of output classes and select an ImageC object class to associate with each output class.

ImageC is compatible with a wide range of models of [BioImage Model Zoo](#). ImageC compatible models for object segmentation can be downloaded from [imagec.org](#) below the Download section. Copy the downloaded model to the `{file}./models` folder. ImageC will load all models from this folder automatically on startup and provides them in the AI `model` dropdown for selection.

Deep learning engines

ImageC actually includes the engines PyTorch and OnnxRuntime.

Using PyTorch, [TorchScript](#) models are supported, with OnnxRuntime all models saved on [onnx](#) format can be used.

Tip. ImageC supports PyTorch TorchScript format and OnnxRuntime's onnx format.

The provided AI classifier combines object segmentation and classification in one command since both is done by the trained AI model in one step. When an image is forwarded to the AI model, the result is a prediction of objects, with each predicted object having a confidence and being assigned to an AI output class. ImageC AI classifier provides a filter tab for each possible output class of the AI model. These filters can be used to assign the predicted output classes of the AI model to an ImageC object class.

Thresholds

Thresholds in the context of AI are probability thresholds which define the minimum probability of a detection output to mark the prediction as valid.

Mask threshold

A typical output of an AI model for image segmentation is a matrix containing a probability value for each pixel, where each of these probability values represents the probability that the pixel belongs to the background or to the foreground.

The mask threshold defines the minimum probability required for a pixel to be defined as a foreground pixel.

Class threshold

The class threshold is a probability value ranging from zero to one, where one represents 100%. Once the image has been processed by the AI network, a matrix of probabilities is created. One of these probabilities is the probability that a detected object can be assigned to one of the defined classes.

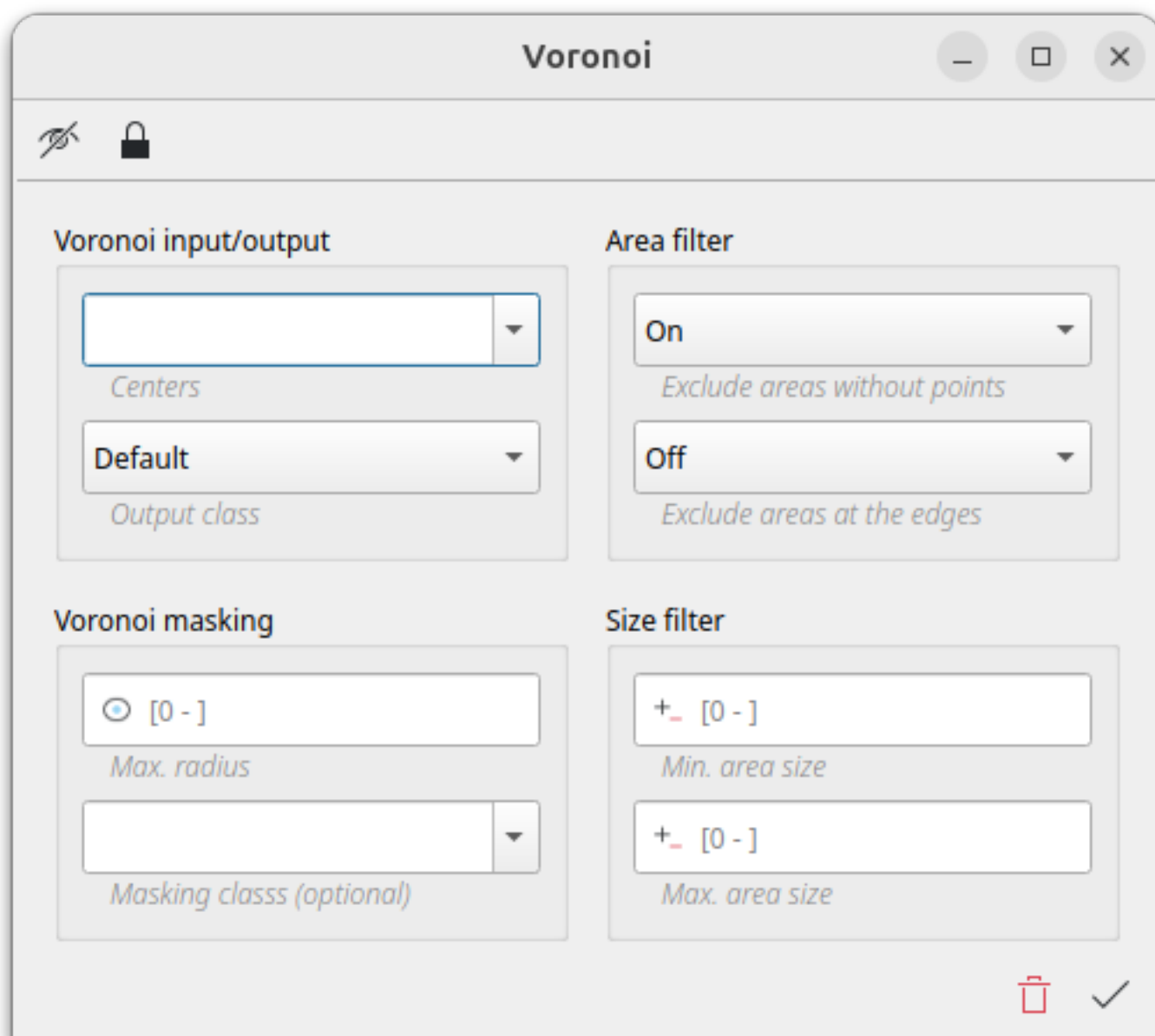
With the class threshold, it is possible to define the minimum probability that a detected object must have in order to be identified as an object of a class.

Hough transformation



Under construction. Coming soon!

Object processing


Voronoi




Voronoi


Visibility icon:  Lock icon: 


Voronoi input/output


Centers




Output class


Area filter


Exclude areas without points




Exclude areas at the edges



Voronoi masking



 
Max. radius



Size filter

 
Min. area size

 
Max. area size

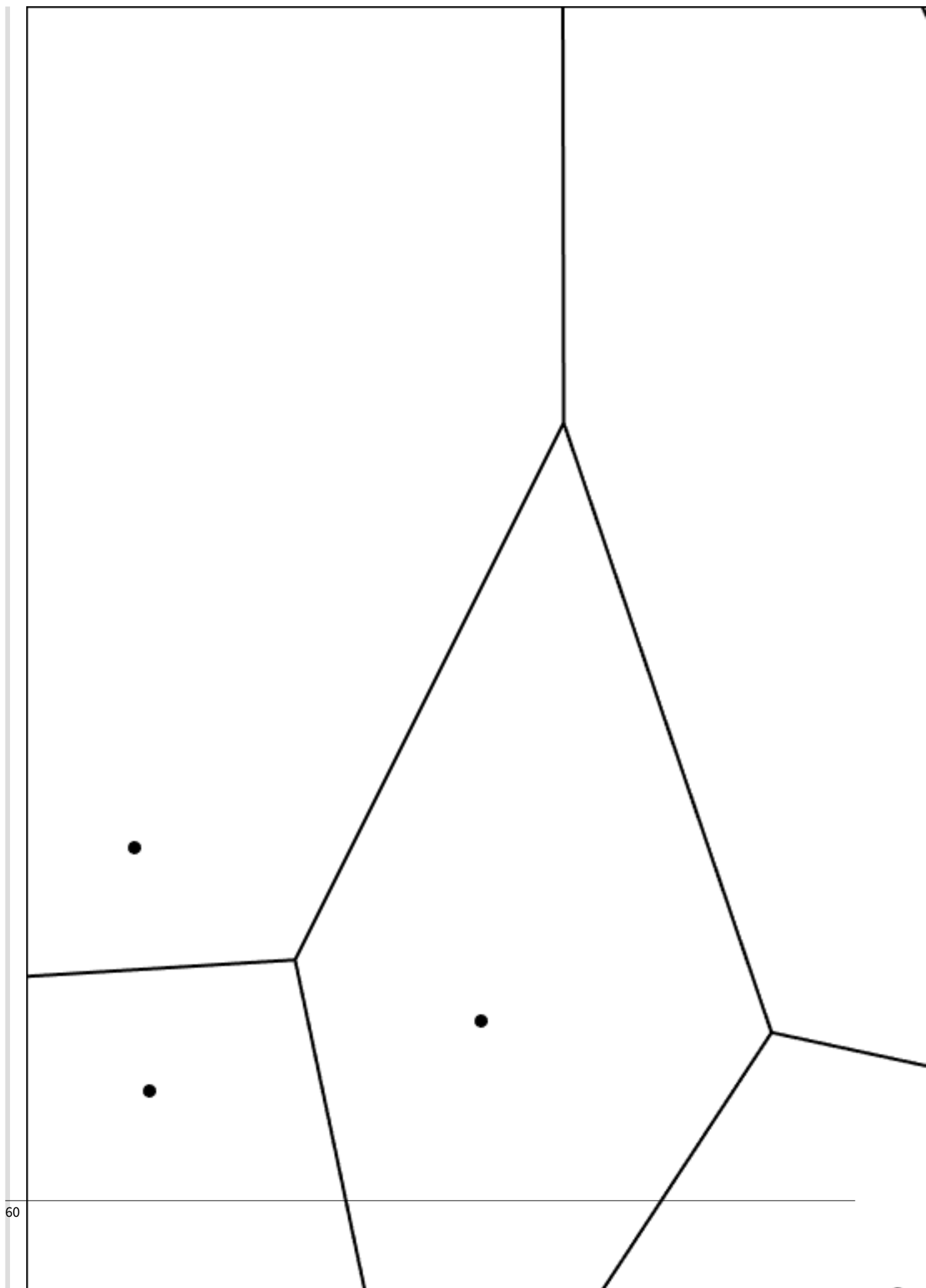
 

In order to construct a Voronoi diagram, it is necessary to gather certain fundamental data. This information can be obtained from the Voronoi settings tab, which is used for the Voronoi diagram construction process.

A Voronoi diagram, also known as a Thiessen polygon or Dirichlet decomposition, represents a decomposition of space into regions that are determined by a given set of points in space, referred to here as centres. The diagram areas are generated so that the enveloping area of each point is the closest to that point. Voronoi diagrams can be useful for approximating cell surfaces based on



known cell nuclei.



Voronoi centers

A Voronoi diagram is constructed using points in space as the fundamental building blocks. The `Voronoi centers` defines the object classes which ImageC should use to take the points in space from. The centre of mass is used as the initial point for the Voronoi construction process, from the valid regions of interest identified within the selected channel.

Best practice. Typical `Voronoi centers` are nuclei.

Max radius

With setting `Max radius` the area of a Voronoi diagram can be limited by its radius. This option is particularly beneficial in scenarios where cells must be approximated based on provided nuclei, yet the cell density is exceedingly low. Remove the value if no size limitation should be done.

Masking class

ImageC offers the option to overlay and intersect a calculated Voronoi diagram with another surface. This option is particularly beneficial in scenarios where a cell area channel exists. The intersection of the calculated Voronoi grid with the cell area channel results in a more accurate cell approximation, as any areas that are not part of the actual terrain are removed.

Object filter

Object filter in voronoi slots are applied to the calculated Voronoi diagram areas. Beside the option to exclude voronoi diagram areas based on its size two more advanced options, `Exclude areas at the edges` and `Exclude areas without center` are available.

Exclude areas at the edges

When this filter is active, all areas that touch the edge of the image are marked as invalid.

Exclude areas without center

The Voronoi diagram is computed based on a set of points which are the centres of the computed Voronoi diagram areas. When using the Voronoi diagram in combination with a `Masking class`, it can happen that after applying the mask, areas are left that don't have a centre point anymore. Enabling this filter will invalidate all areas that do not contain a point from the source point set.

Best practice. It is recommended to enable this option if nuclei are used as a set of points and cell areas as a clipping mask. Applying this filter option will filter out all left areas that do not contain nuclei.

Reclassify

Reclassify

Input

cy3@spot
Input (e.g. Spot)

Reclassify Move
Mode

cy3@tetraspeck
Reclassify to

Intersect with

tetraspeck@spot
Intersect with (e.g. Tetraspeck)

% 0.1
Min. intersection

Move/Copy if intersect
Filter logic

Create hierarchy tree
Hierarchy mode

Intensity filter

Default
Image channel

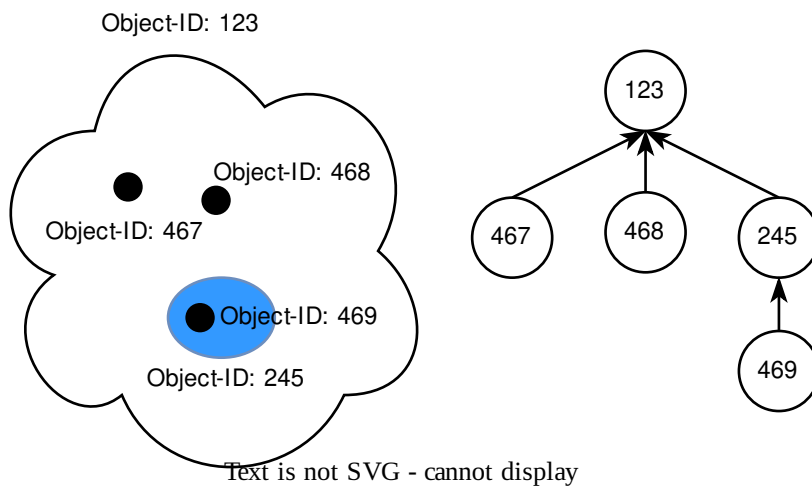
≡ Default
Z-Projection

The reclassify command can be used to change the class of an object that has been formally classified using the **classifier** or **ai-classifier** command, based on some filter criteria. This allows to create further fine granular object populations.

Typical use case for instance are the separation of spots in cell and outside a cell, which can be managed using the intersection filter. Or the separation of bright and less bright spots.

For that reason the ImageC reclassifier provides two filter options: Intersection and Intensity.

An object hierarchy is built up by storing a parent object ID together with each object. This information can be used to draw a hierarchy graph of the objects, showing which objects are part of another. Use the intersection filter of the Reclassify command to build up such a hierarchy.



Filters

Intersection filter

The intersection filter takes an intersection object class as input and is applied to those objects of the input input class that intersect with any of these objects. ImageC can create a object hierarchy (see section [parent object id](#)) when using the intersection filter. The object hierarchy allows to answer the questions: Which objects are contained within another object and how many objects intersect another object!?

The parent information is stored in the intersecting object from the input object input class whereby the parent object is the intersecting object from the intersecting object input class.

ImageC allows you to specify whether or not to create a hierarchy for intersecting objects by specifying the hierarchy mode.

Create hierarchy

This option stores the parent information for intersecting objects. Existing hierarchy information is retained which allows to build up a full hierarchy tree. E.g. parent of nucleus is cell, parent of spot is nucleus, which allows to answer the questions: Give me the spots within a Cell and give me the spots within a cell in the nucleus of this cell.

Keep existing

This option does not change the hierarchy information of the intersecting object.

Remove hierarchy information

This option removes the parent object ID from the intersecting objects.

Intensity filter

This filter allows to specify a min and max intensity measured in the defined image channel. If the measured object intensity is within this intensity range the filter is applied.

Intensity filter and intersection filter can both be activated. Only if both filters match the filter for the object is applied.

Match handling

Once the filter matches, one of the Move or Copy operations will be applied to the objects that match the filter.

The move operator applies a new object class to the object. Object ID keeps the same as before.

The copy operator creates a new object and assigns the new object class only to the new object. A new object ID is generated and the origin object ID of the newly created object is set to the object ID of the origin object. The origin object keeps untouched.

Objects to binary

Under construction. Coming soon!

Object transform

Under construction. Coming soon!

Save control image

Under construction. Coming soon!

Measurement

Colocalization

Colocalization

Input 1

Off

Class to coloc.

Off

Reclassify to if coloc.

None

Reclassify to if not coloc.

Input 2

Off

Class to coloc.

Off

Reclassify to if coloc.

None

Reclassify to if not coloc.

Output

Default

Coloc output class

% 0.1

Min. intersection

Reclassify Copy

Mode

Override existing tracking info

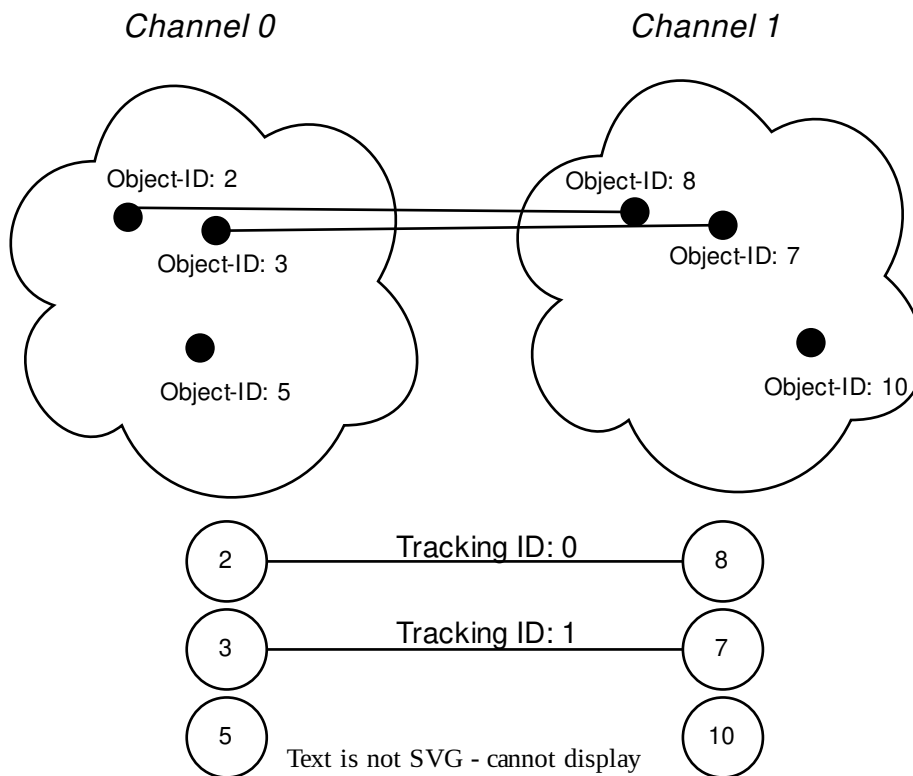
Tracking mode

65

The colocalization command is used to determine objects which intersect with each other. Up to four object classes can be given as input, for those the colocalization is calculated. A spot is defined as colocalizing if for each given input class a overlapping object is found.

The ImageC colocalization command allows to reclassify those objects which colocalizes for a further processing of the objects.

Object tracking is the process of linking two objects, either from different time frames or channels, using the same tracking ID for all objects. This makes it possible to display objects from different sources that physically represent one object.



Colocalizing handling

Once an object colocalizes, one of the Move or Copy operations will be applied to that objects.

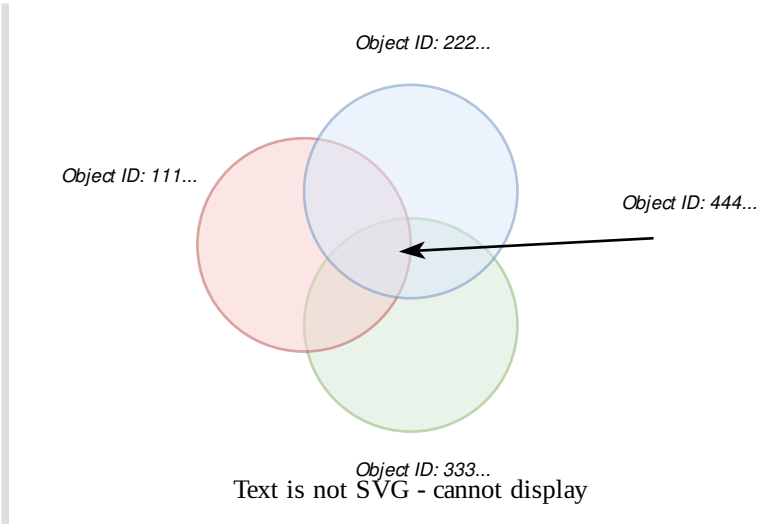
The move operator applies a new object class to the object. Object ID keeps the same as before.

The copy operator creates a new object and assigns the new object class only to the new object. A new object ID is generated and the origin object ID of the newly created object is set to the object ID of the origin object. The origin object keeps untouched.

In addition a new object is generated which the overlapping area (colocalization area) of the overlapping objects.

All colocating objects, including the newly created Coloc Area object, are given a common tracking ID. In the results, objects with the same Tracking ID are placed next to each other, allowing matching objects to be compared.

If from each input class an object overlaps with an other object from the other input classes, the object is defined as colocalizing. The overlapping area is called the coloc area and a new object for further processing of this area is generated. In addition each colocalizing object gets the same tracking ID assigned.



Measure intensity

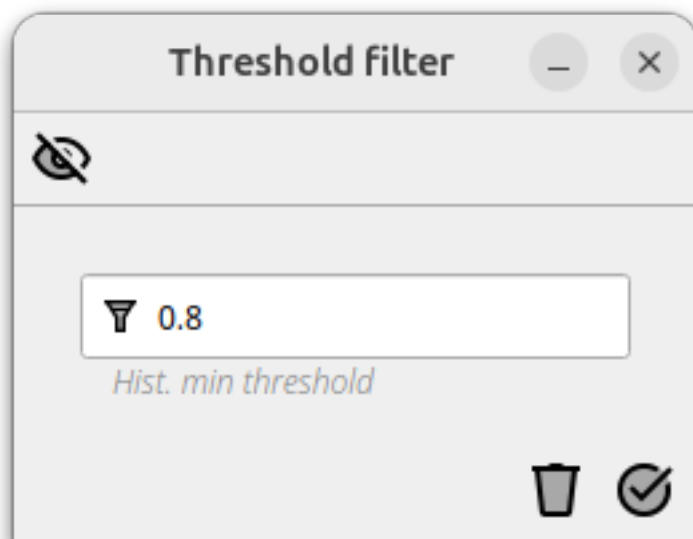
Under construction. Coming soon!

Measure distance

Under construction. Coming soon!

Filtering

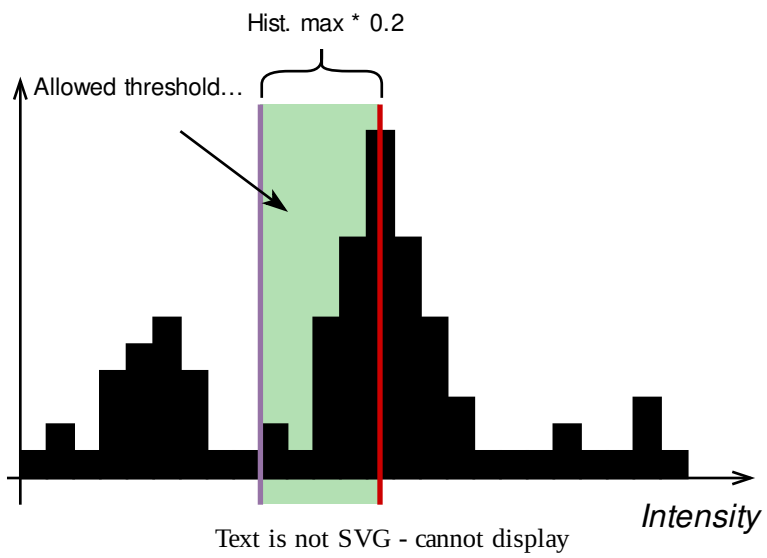
Threshold filter



To get comparable results in one experiment, it is common to use the same manual threshold value for all images. Nevertheless, given the considerable range of exposure times for images, it is possible that the selected threshold value may prove insufficient for some images. This leads to a kind of overexposed image after the threshold, in which the background is erroneously recognized as a signal.

To filter out images that are affected by this problem, ImageC offers a `Threshold filter` filter. In the event that the selected threshold value is less than the value observed at the maximum of the image histogram, multiplied by the aforementioned factor, the filter is applied.

The intensity value at the maximum of the histogram of the image multiplied by the `Threshold filter` defines the area of allowed threshold values for the image. If the min. threshold value is lower than the lower bound of this area, the image is filtered out.



Noise filter

Under construction. Coming soon!

Tutorials

Tutorials

The tutorial section contains compact examples that are centered a single goal which should help to learn working with ImageC's pipelines.

EVAnalyzer

ImageC is the direct successor of [EVAnalyzer](#) an ImageJ/Fiji plugin designed for some standard use cases in image processing especially for the EV field.

These pipelines are also available in ImageC/EVAnalyzer2, as described in this section.

EVAnalyzer citation: Schürz, M., Danmayr, J., Jaritsch, M., Klinglmayr, E., Benirschke, H. M., Matea, C. - T., Zimmerebner, P., Rauter, J., Wolf, M., Gomes, F. G., Kratochvil, Z., Heger, Z., Miller, A., Heuser, T., Stanojlovic, V., Kiefer, J., Plank, T., Johnson, L., Himly, M., ... Meisner-Kober, N. (2022). EVAnalyzer: High content imaging for rigorous characterisation of single extracellular vesicles using standard laboratory equipment and a new open-source ImageJ/Fiji plugin. *Journal of Extracellular Vesicles*, 11, e12282. <https://doi.org/10.1002/jev2.12282>

EVAnalyzer

Spot count

Under construction. Coming soon!

Spot count

Under construction. Coming soon!

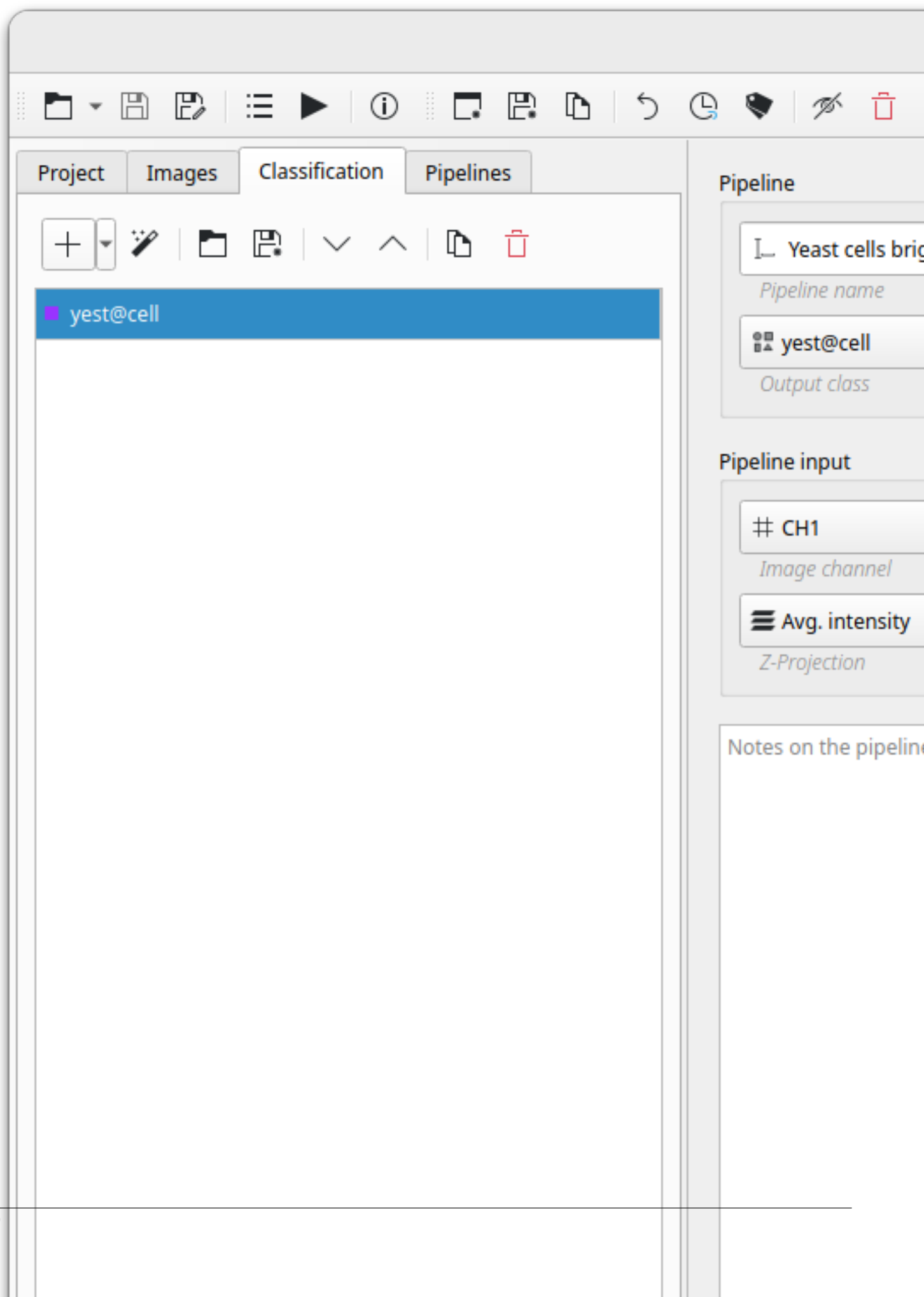
Spot count

Under construction. Coming soon!

Advanced

Yest cell detection

ImageC is shipped with a predefined pipelines for Yest cell detection in brightfield images. In this tutorial we will go step by step through this pipeline template.



Before you start creating the pipeline, look at your image and identify the metrics that distinguish your objects of interest from the rest of the image. When we examine the yeast cells in bright field, a very strong characteristic is the cell membrane, which is shown as a bold, dark border.

So let's find a pipeline which finds this border:

1. **Intensity** First, we use the intensity command to adapt the brightness of the image based on the image exposure.
2. **Blur** To find the border we will use an edge detection algorithm. However such algorithms react on intensity changes in the image. Noise per definition generates a lot of small such sharp intensity changes in an image. Therefore we use a Gaussian Blur as our second step in the pipeline. The gaussian blur protects "real" edges in the image from being blurred away but removed noise efficiently.
3. **Canny edge detection** The canny edge detection algorithm is now applied, converting the grayscale image into a binary image containing only the detected edges.
4. **Blur** the edges again to fill the are inside the circle with more contrast.
5. **Threshold** The background is removed when we reach the final threshold, which results in the cell body becoming more visible.
6. **Hough transformation** is used to find circles, which represents our cells in the image.
7. **Voronoi** The voronoi algorithm is used to separate the cells from each other.

Technical insights

Overview

The goal in developing ImageC was to use as few computing resources as possible. This means that ImageC can be executed on standard home computers and laptops, as well as on high-performance computing clusters.

Nevertheless, even the most efficient image processing application requires at least a couple of hardware components to run smoothly. This chapter gives an overview of the ImageC footprint and hardware resources.

Resources and Limits

Minimum hardware requirements. More is always better ;) ...

| Component | Minimum |
|--------------|---|
| CPU | 64 bit, 1 GHz dual core (preferred > 2 GHz) |
| RAM | 2 GB of free RAM (+0.5 GB for each additional CPU core) |
| Storage | 500 MB free storage for installation |
| Screen | Minimum recommended resolution 1528x980 |
| Graphic card | (optional) Nvidia graphical card with CUDA support. |

Limits

See section [Image formats](#) for a list of supported image formats.

| Limit | Default value |
|-------------------------|---------------|
| Max classes | 32 |
| Max image series | 10 |
| Max image channels | 10 |
| Max z-stack images | 65535 |
| Max t-stack images | 2 billion |
| Max number of objects | 1 billion |
| Max number of pipelines | 65535 |
| Maximum tile size | 46340x46340 |

Files Created by ImageC

ImageC creates several files and directories on your disk. Some global files are created and retained even after an update to ImageC, while some local files are generated directly within the ImageC installation folder. Additionally, some analysis artifacts are generated when an analysis is started.

File types

Following file types can be generated by ImageC.

| Endian | Description |
|--------------|--|
| .icdb | SQL database file containing results data. Under the hood a DuckDB storage format. |
| .ictbl | ImageC results table settings file. Used to reload table settings in the results view. |
| .icproj | ImageC project file, containing all project settings. |
| .ictemplproj | ImageC project template file for reusing and sharing project settings. |
| .ictempl | ImageC pipeline template file for reusing and sharing pipeline settings. |
| .ictemplcc | ImageC classification template file for reusing and sharing class settings. |

Global Files and Directories

ImageC creates the following global files in the user's home directory on startup. For Unix based systems (Linux, macOS) a folder `~/imagec` is generated. For Windows based systems a folder `C:\Users\<USERNAME>\imagec` is generated.

The folder contains following files and directories

| Name | Description |
|--|--|
| <code>~/imagec/user_settings.json</code> | Stores some global user settings, like the list of last opened files |
| <code>~/imagec/templates</code> | Project, pipeline and classification templates which the user has been created. Copy the new templates into this folder for easy access within ImageC. |

Local Files and Directories

ImageC creates the following files and directories in the installation folder of ImageC.

| Name | Description |
|---|--|
| <code>./models</code> | Copy AI models within this folder to use them within ImageC. |
| <code>./hs_err_pid<NR>.log</code> | Error/Trace log of the BioFormats Java integration. |

Files created during analysis

When an analysis is started, ImageC automatically creates some job artefacts in the selected image working directory. If no job name was given in the project settings, a unique job name is generated for each run.

| Name | Description |
|---|---|
| <code><working_directory>/imagec_data_base_files/analysis_results/analysis_results.sql.highlight</code> | Database file containing its results of the run. The file can be opened with the ImageC results viewer or with SQL knowledge using one of the duckDB clients. |
| <code><working_directory>/imagec_statistics/analysis_results/analysis_results.sql.highlight</code> | Statistics about the execution time of each individual pipeline step. |
| <code><working_directory>/imagec/analysis_results/analysis_results.sql.highlight</code> | Copy of the project settings when the job has been started. |
| <code><working_directory>/imagec/analysis_results/analysis_results.sql.highlight</code> | Folder containing all generated images generated during the run. |

Development

Debugging

ImageC is written in C++ and compiled into a binary that is executed directly on your CPU. While such programmes offer the highest possible performance, they have the disadvantage that, in the event of a programme error, the programme simply crashes and closes.

Even though we test ImageC, it is still possible that we have overlooked an error, causing the programme to crash in certain situations. To help us identify the cause of the crash, ImageC can be started in debug mode. Once started in debug mode, ImageC generates a crash report that can be made available to us.

Once the program crashes please send the `debug.txt` to support@imagec.org.

Linux

1. Open a terminal window
2. Change directory to the installation folder of ImageC `cd <imagec_install_directory>{:.language-sql.highlight}`
3. Start ImageC with following command `./imagec > debug.txt`

Window

1. Open Windows Powershell as Administrator.
2. Change directory to the installation folder of ImageC `cd <imagec_install_directory>{:.language-sql.highlight}`
3. Start ImageC with following command `Start-Process -RedirectStandardOutput debug.txt imagec.exe`

macOS

1. Open a terminal window
2. Change directory to the installation folder of ImageC `cd <imagec_install_directory>{:.language-sql.highlight}`
3. Start ImageC with following command `./imagec > debug.txt`

Building

ImageC is written in the C++ programming language and uses the Linux operating system. During the build process, it is compiled for Linux, Windows and macOS.

For developing [VSCode](#) and a Docker devcontainer is used. The devcontainer is actually only tested for Linux operating system.

Warning. If you want to try developing under Windows, you can try Docker for Windows together with the Windows Subsystem for Linux (WSL). However, we do not provide any support or guarantee that it works.

Preparation

1. Download [VSCode](#)
2. Clone ImageC repo <https://github.com/joda01/imagec.git>
3. Open the cloned directory using VSCode.
4. Install Dev Containers extension and Reopen open the project in container.

First time to build

ImageC uses [Conan](#) build system. The artifacts needed for building are stored in our ImageC artifactory. First time you want to build, please contact us and request access to the artifactory.

1. Request artifactory request support@imagec.org
2. Execute `./build_linux.sh --init` and enter your received token to login at the artifactory.
3. Execute `./build_linux.sh --make`
4. Execute `./build_linux.sh --build`

The build artifacts are placed into `./build/build/output`

Testing

Still under construction

ImageC executes testing pipeline before a new releases is published. These testing pipelines are placed in the <https://github.com/joda01/imagec-test> repository.

ImageC Blog

Cell Segmentation

Publication date: 2025-05-31

Author: Joachim Danmayr

TL;DR: YOLOv5 model for brightfield cell segmentation

This pre-trained AI model can be used for brightfield cell segmentation.

After download unzip and copy the whole folder into: `<imagec_installation_directory>/models{:.language-sql.highlight}`

[Download](#)

Acknowledgments

This document is built with [Pandoc](#) using the [Eisvogel template](#). The scripts to build the document are available in the [Imagec-Doc repository](#).

The emojis used in this document are provided by [Twemoji](#) under the [CC-BY 4.0 license](#).

The syntax highlighter uses the [Bluloco Light theme](#) by Umut Topuzoğlu.

