

Image segmentation

Robert Haase

Using materials from

Alba Villaronga Luque and Jesse Veenvliet, MPI-CBG Dresden

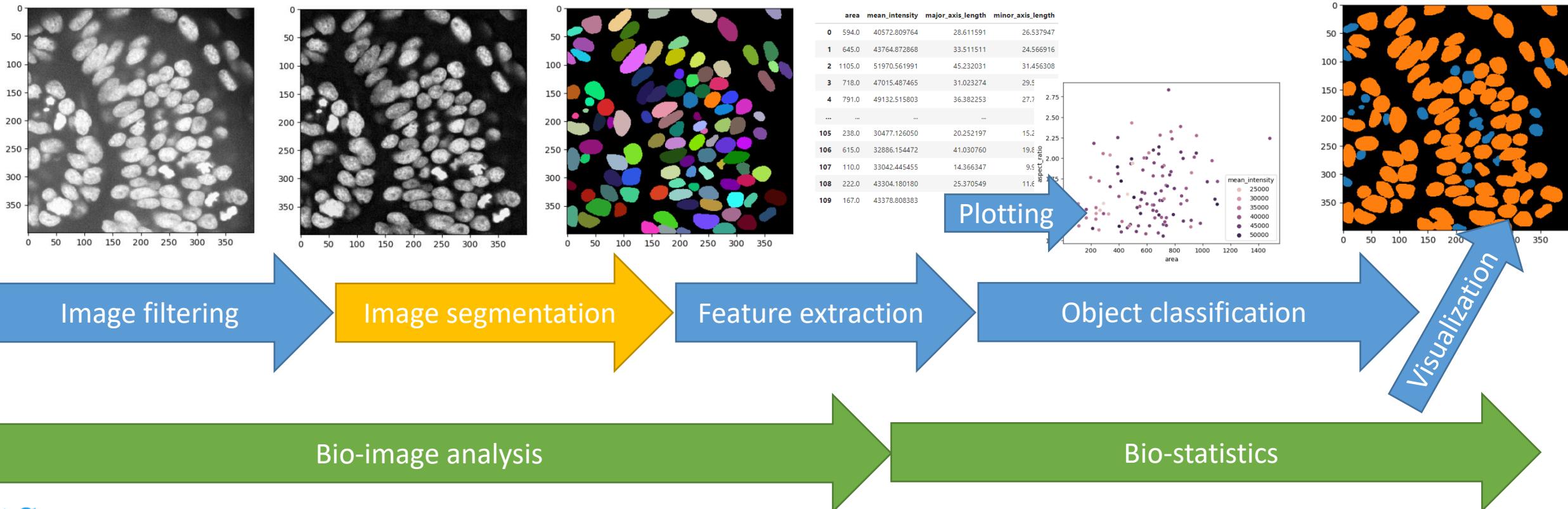
Ryan Savill George, MPI CBG Dresden

Johannes Soltwedel, PoL, TU Dresden

April 2023

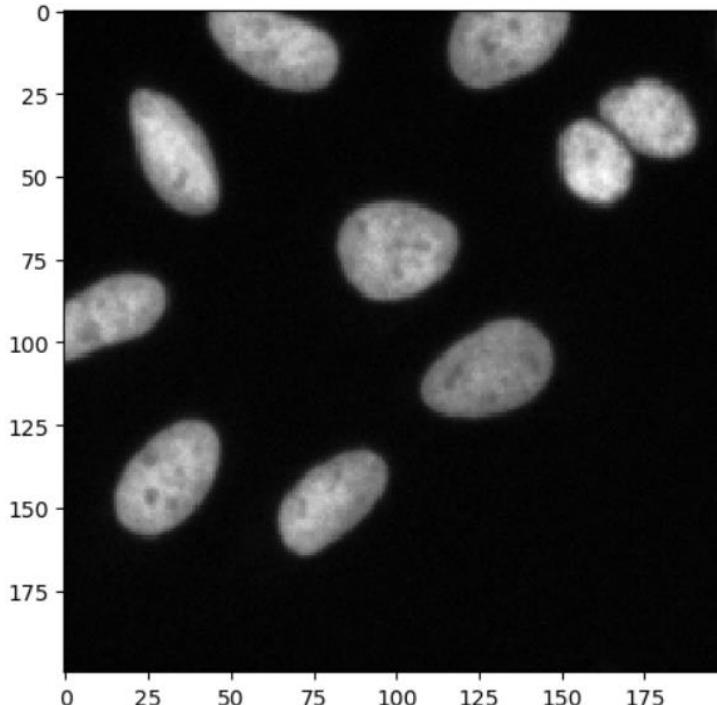
Lecture overview: Bio-image Analysis

- Image Data Analysis workflows
- Goal: **Quantify observations, substantiate conclusions with numbers**

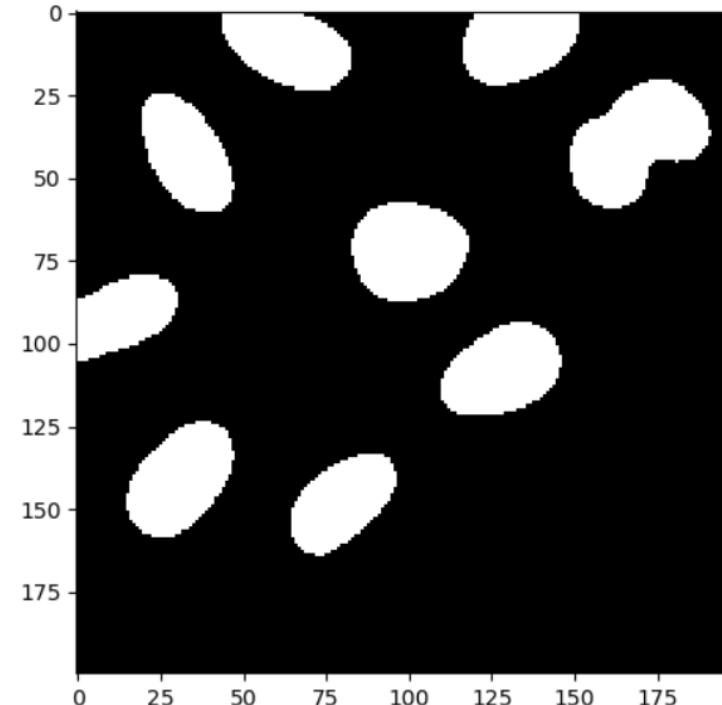


Terminology

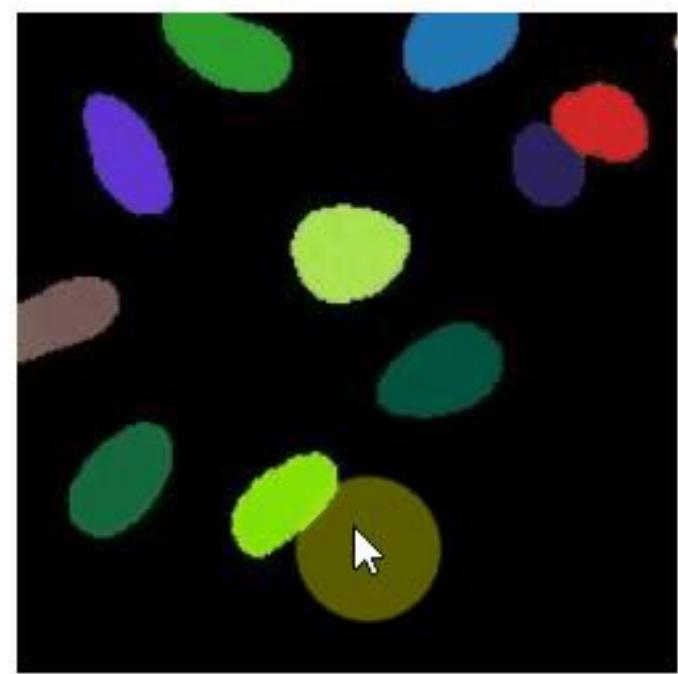
Intensity image



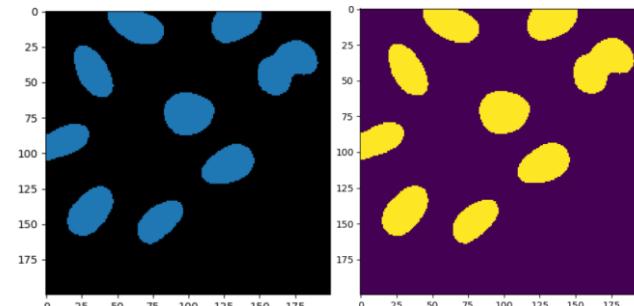
Binary image



Label image



No matter how they are displayed



[y=152, x=92] = 0

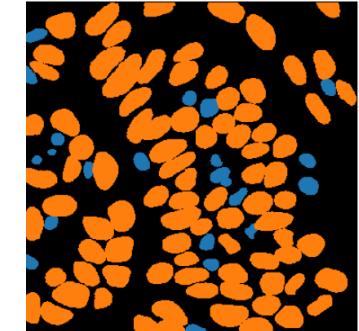
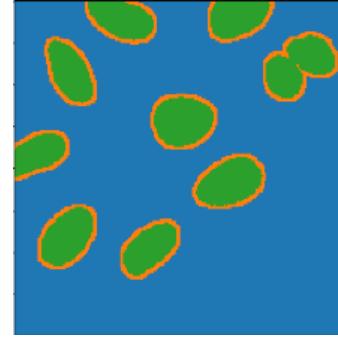
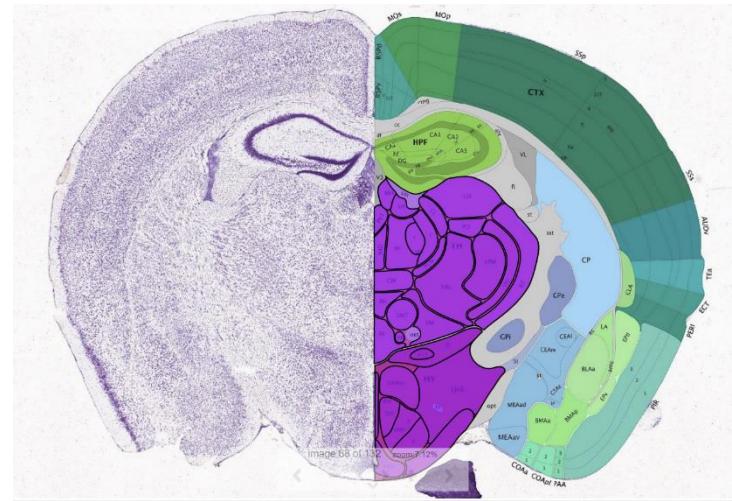
Instance segmentation



Instances:

- Cells, nuclei, cats, dogs, cars, trees

Semantic segmentation

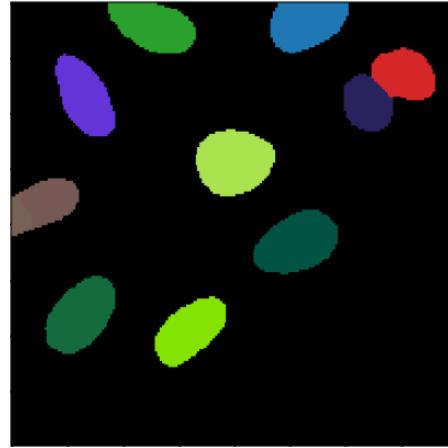


Regions:

- Anatomical, geographical

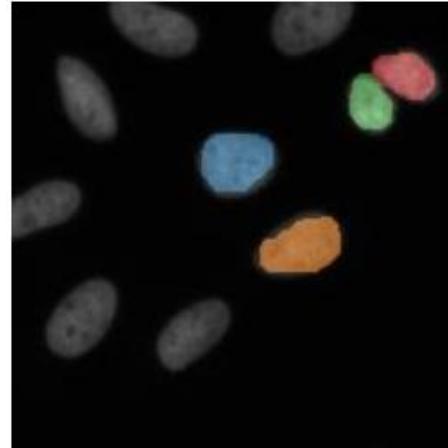
- Annotations are typically drawn by humans (e.g. to train machine learning models)

Instance segmentation



Semantic segmentation

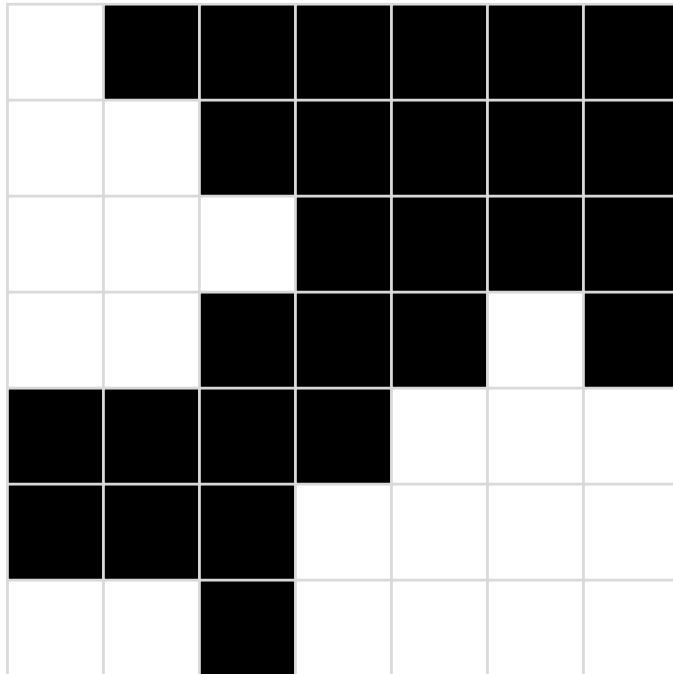
Sparse instance annotation



Sparse semantic annotation

Connected component labelling

- In order to allow the computer differentiating objects, connected component analysis (CCA) is used to mark pixels belonging to different objects with different numbers
- Background pixels are marked with 0.
- The maximum intensity of a labelled map corresponds to the number of objects.



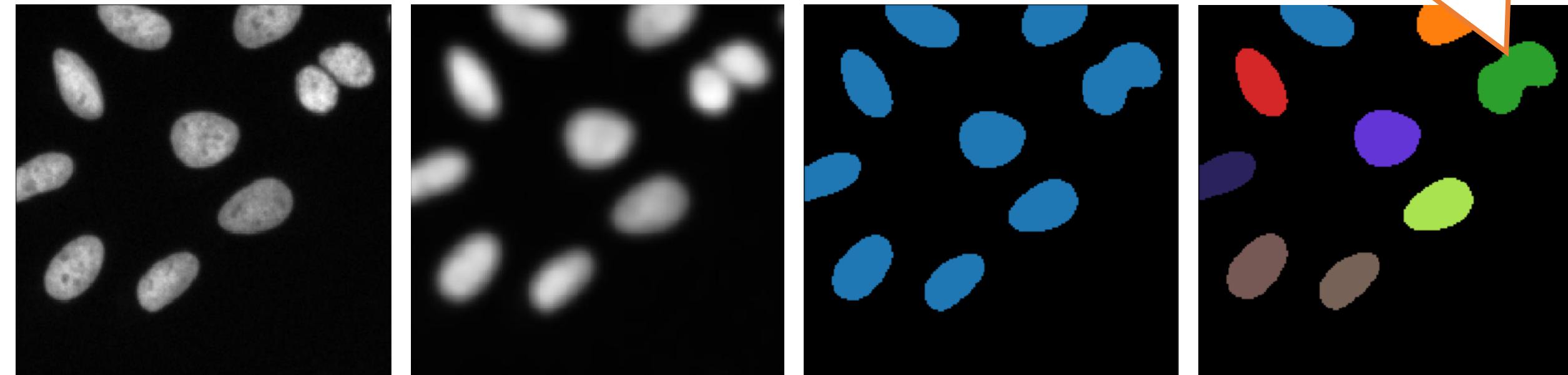
CCA

1	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0
1	1	1	0	0	0	0	0
1	1	0	0	0	3	0	0
0	0	0	0	3	3	3	3
0	0	0	3	3	3	3	3
2	2	0	3	3	3	3	3

Common image segmentation workflows

- Presumably the most common segmentation algorithm used for fluorescent microscopy images:
 - Gaussian blur, Otsu's Threshold, Connected Component Labeling

Limitation: Dense objects



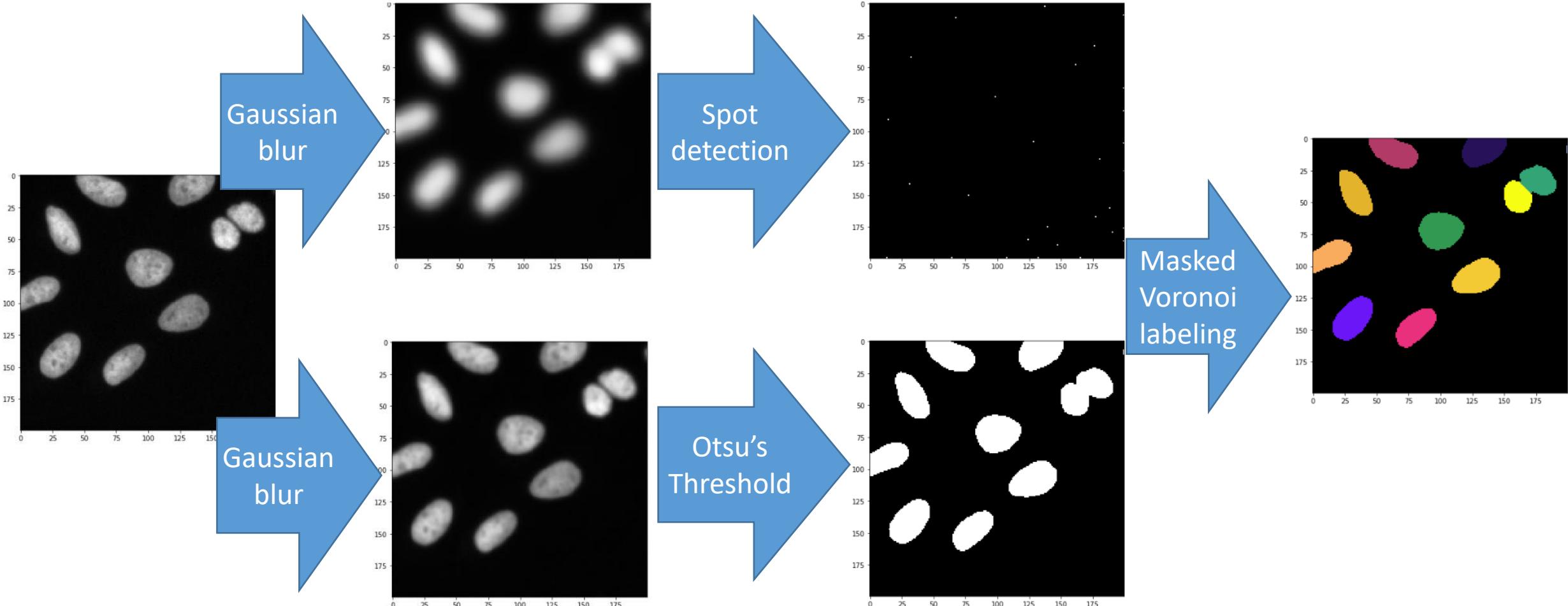
Denoising

Binarization

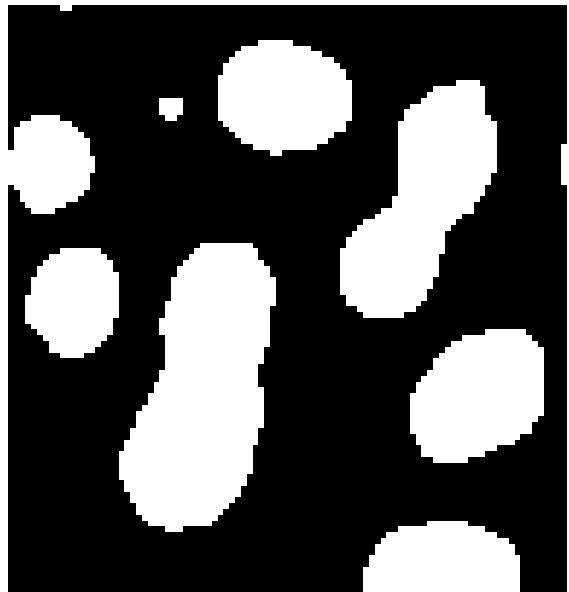
Labeling

Common image segmentation workflows

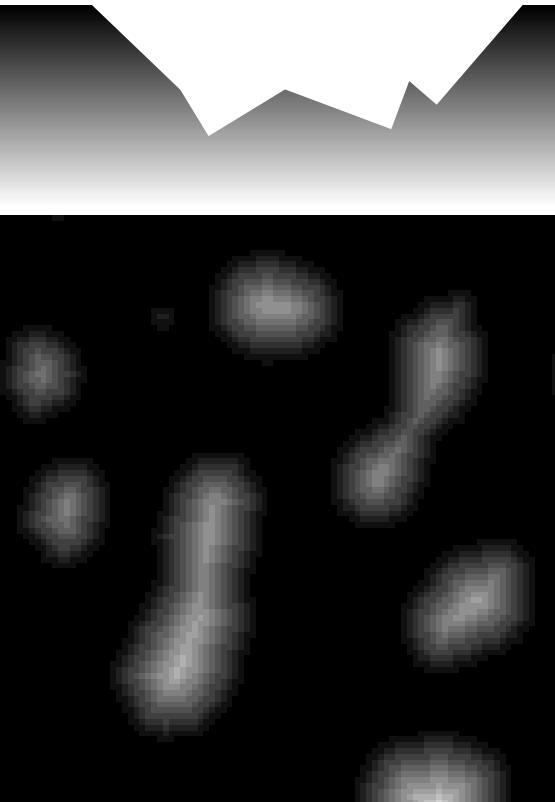
- Combination of Gaussian blur, Otsu's Threshold and Voronoi-labeling



- The watershed algorithm for binary images allows cutting one object into two where it's reasonable.

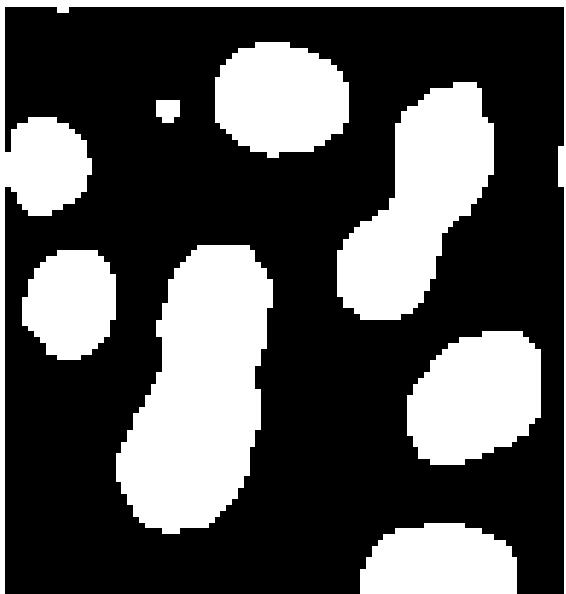


Binary segmentation

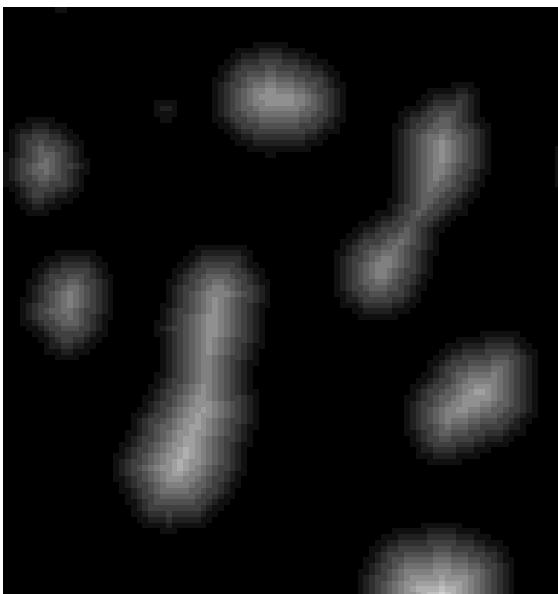
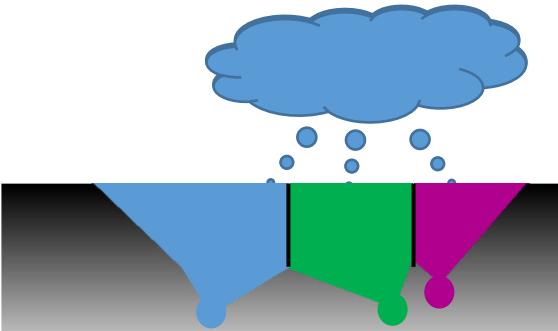


Distance map

- The watershed algorithm for binary images allows cutting one object into two where it's reasonable.



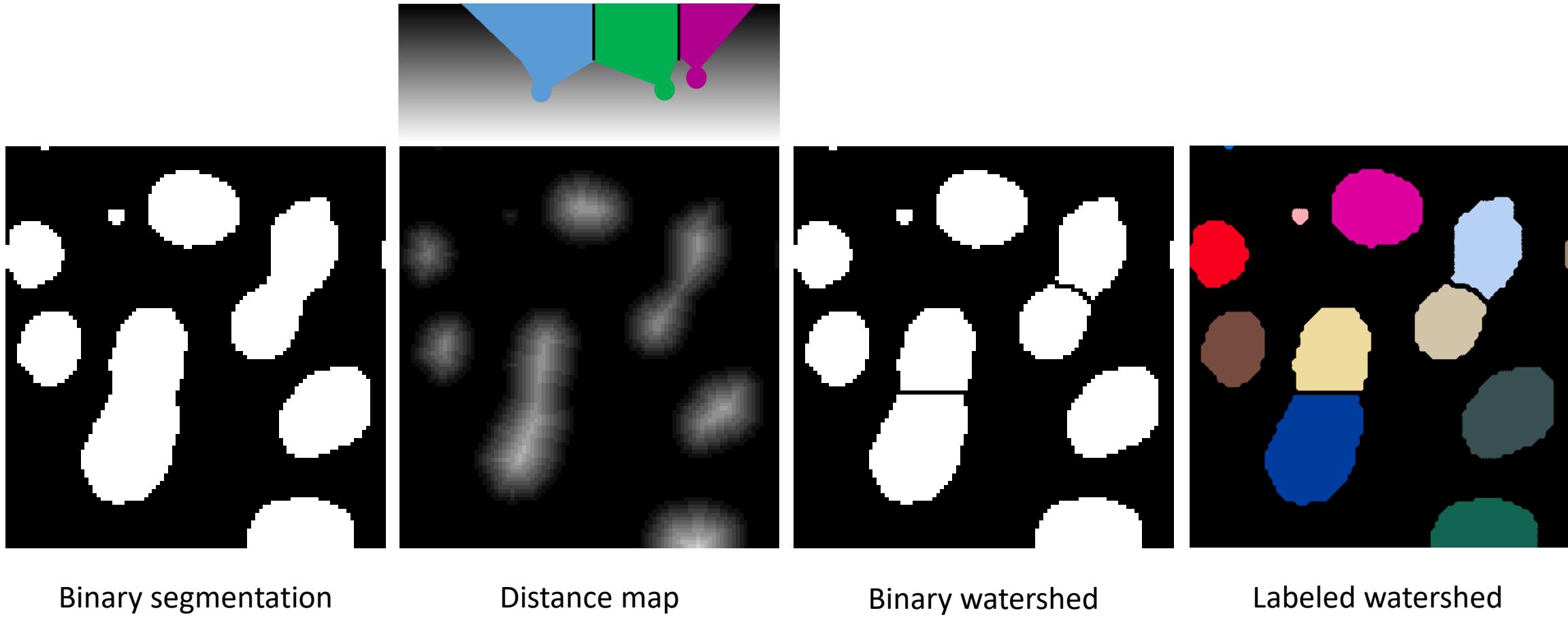
Binary segmentation



Distance map

Watershed

- The watershed algorithm for binary images allows cutting one object into two where it's reasonable.
- The watersheds are made from binary images. The algorithm does not take the original image into account!



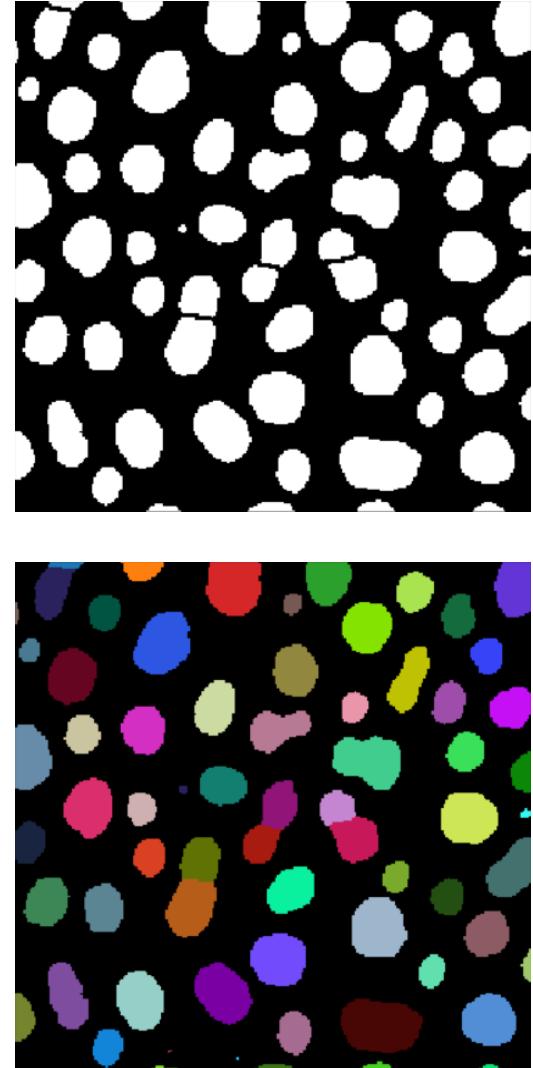
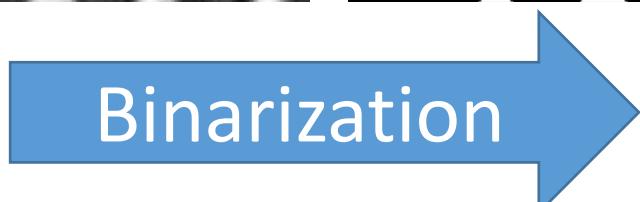
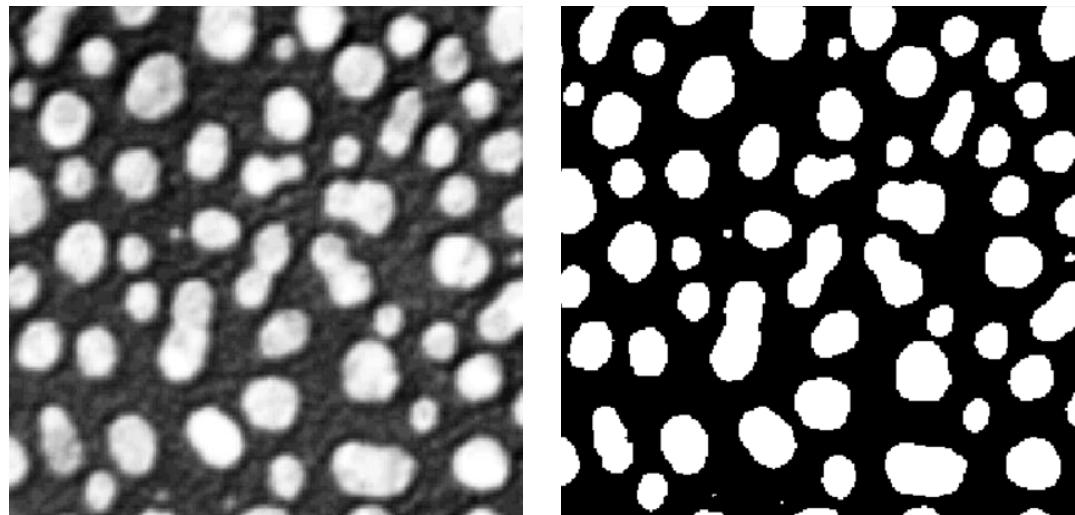
Binary segmentation

Distance map

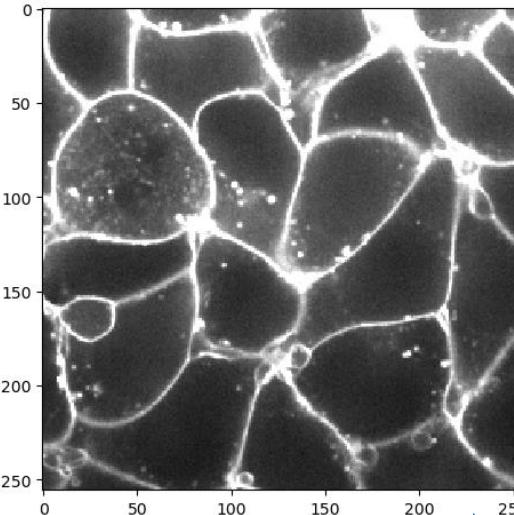
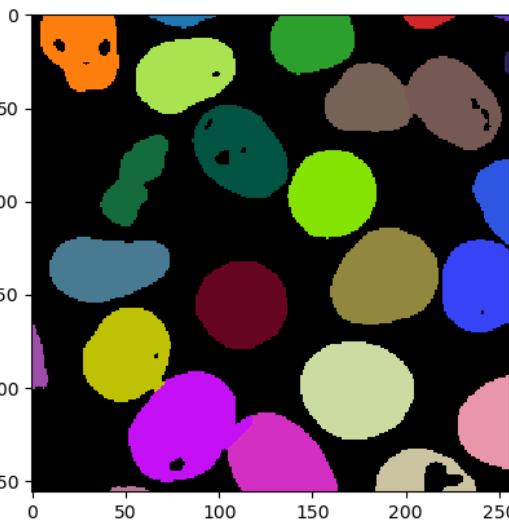
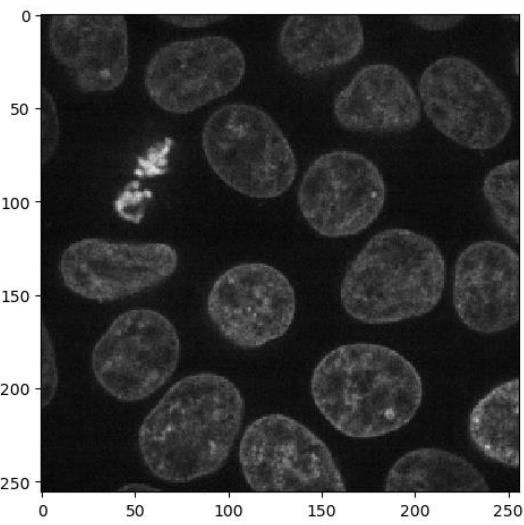
Binary watershed

Labeled watershed

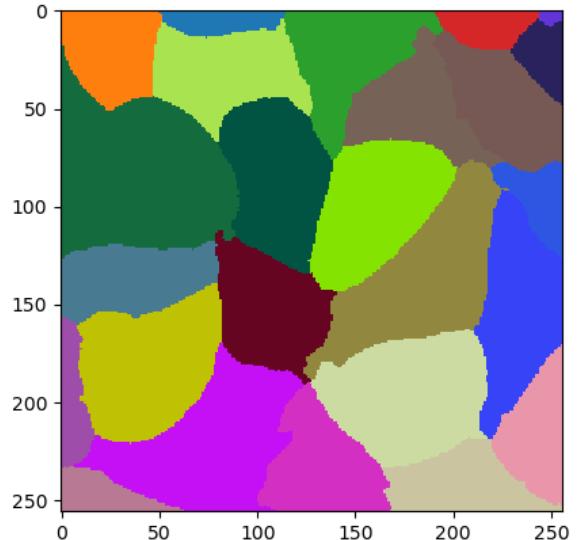
- Split dense objects



- Seeded watershed: Flood regions from pre-defined seeds
- Example: Flood cells from nuclei positions

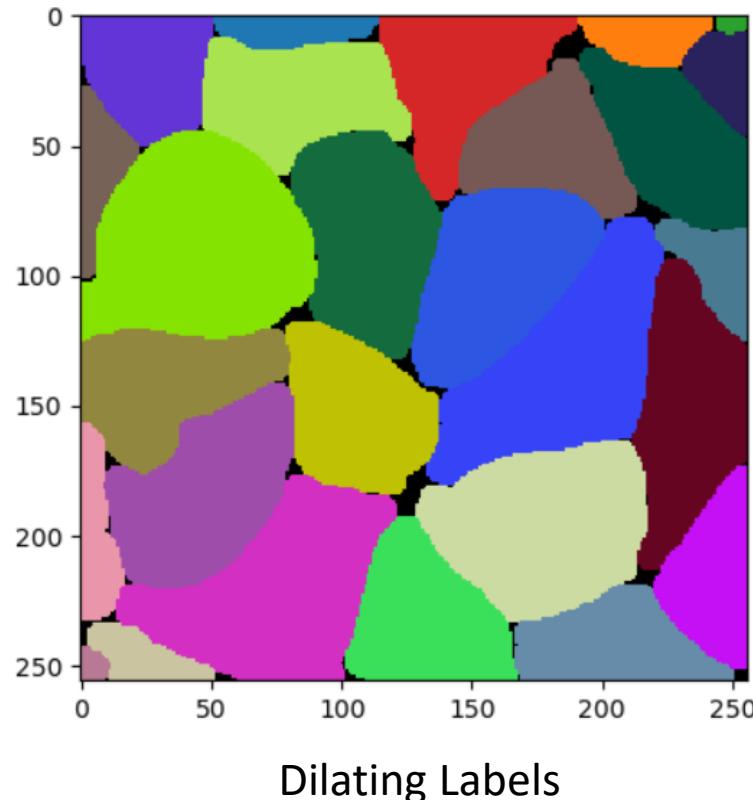
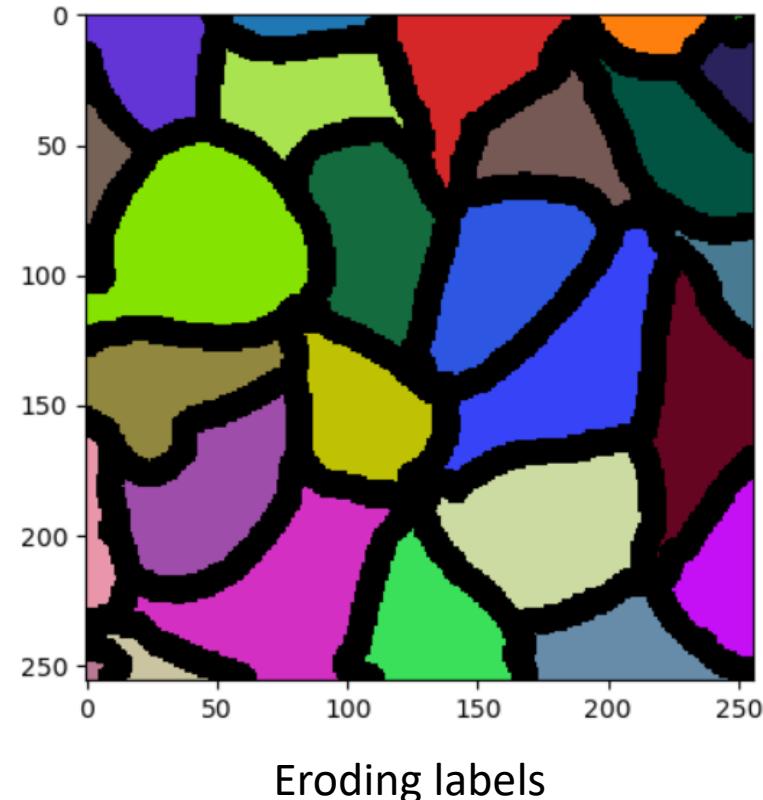
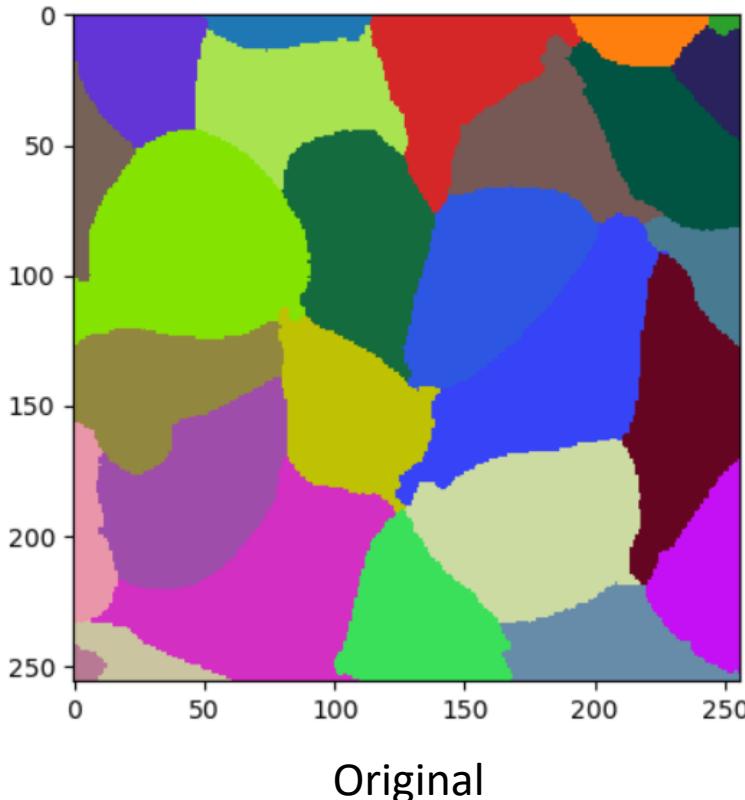


“Seeded” watershed



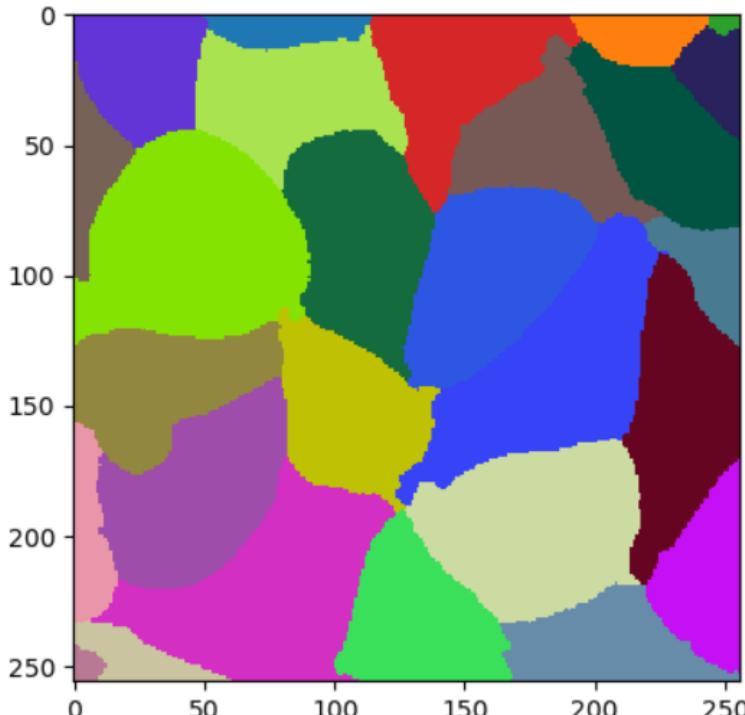
Label post-processing / morphological operations

- ... similar to morphological operations on binary images

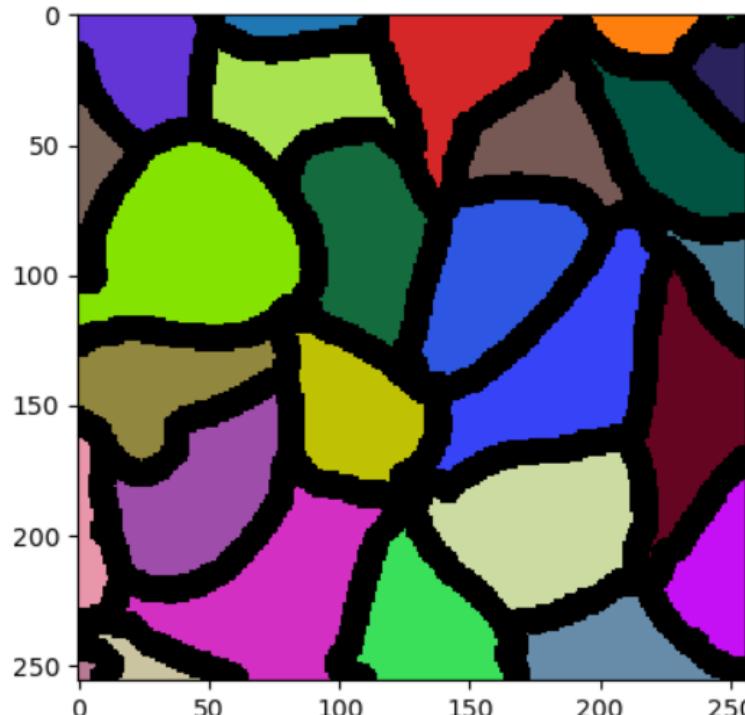


Label post-processing / morphological operations

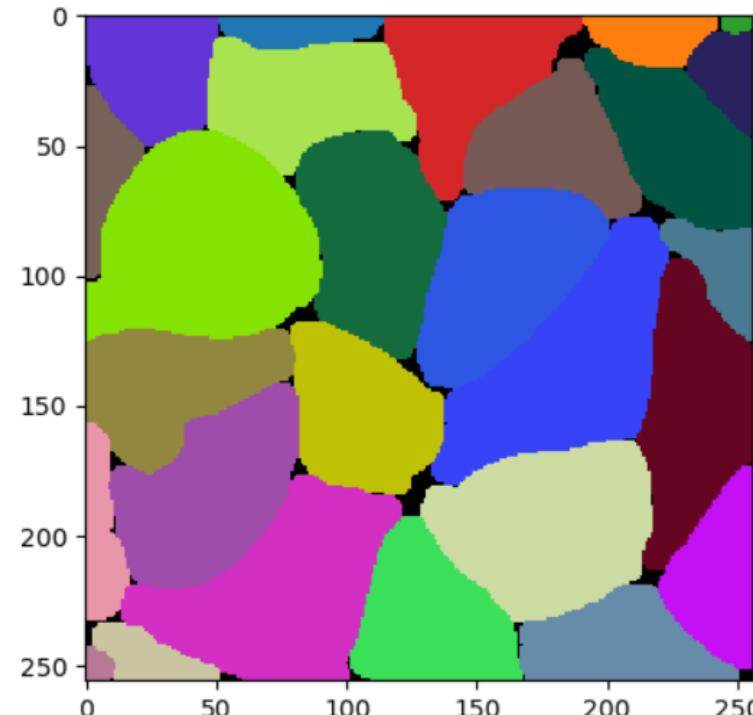
- ... similar to morphological operations on binary images



Original



Eroding labels



Dilating Labels

This combination is called

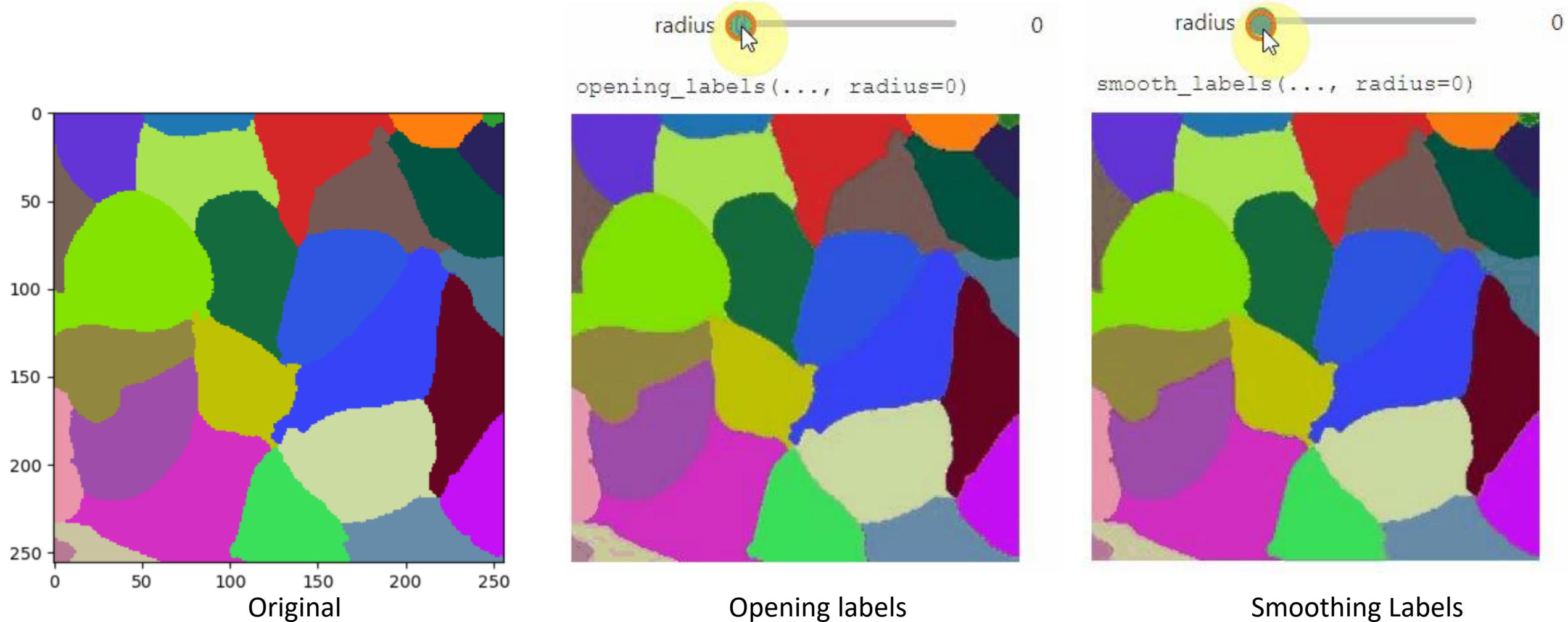
Opening

Closing

Epilepsy warning

Label post-processing / morphological operations

- ... similar to morphological operations on binary images



Surface reconstruction

Robert Haase

Using materials from

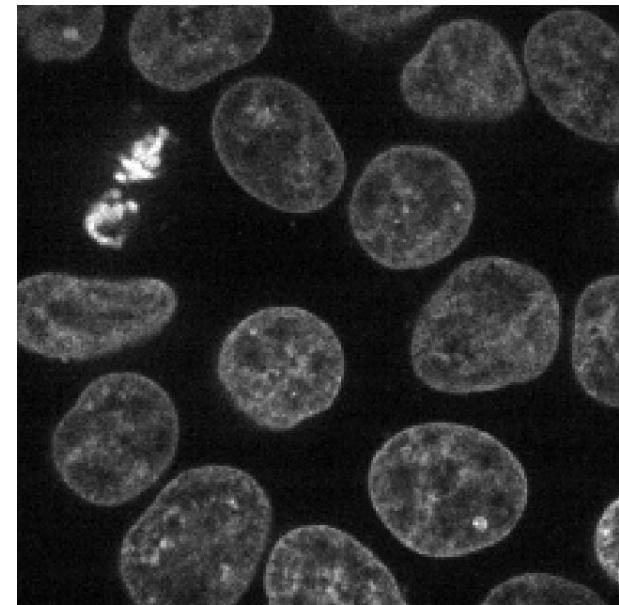
Alba Villaronga Luque and Jesse Veenvliet, MPI-CBG Dresden

Johannes Soltwedel, PoL, TU Dresden

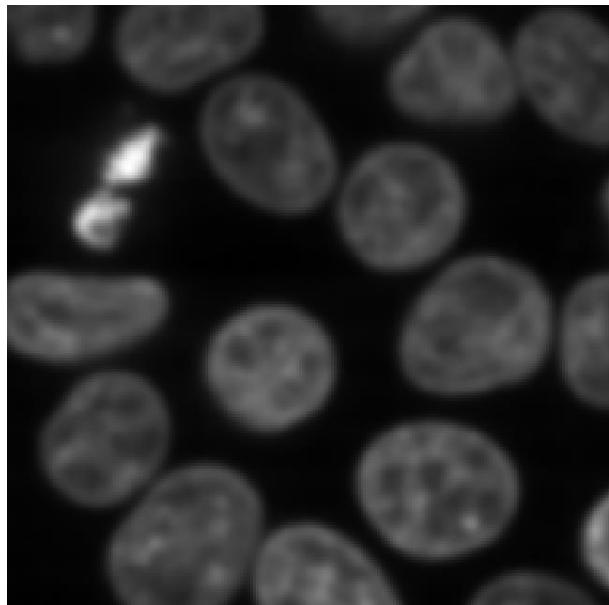
April 2023

Motivation: Surface reconstruction

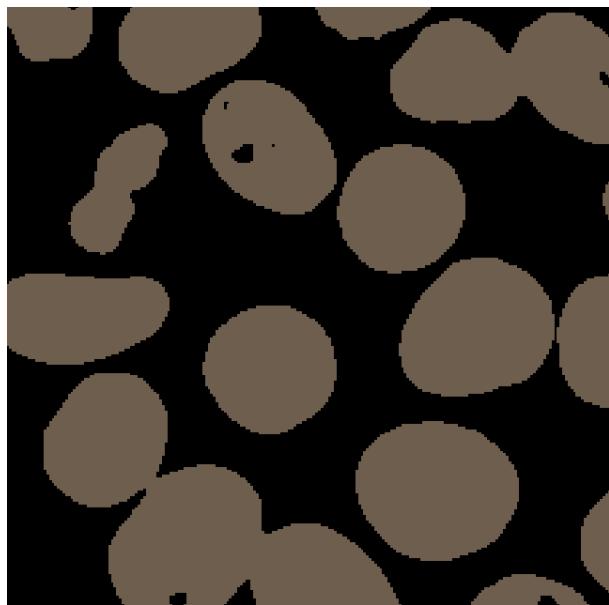
- Pixel and voxel arrays can be huge in memory
- Processing 3D arrays is time-consuming



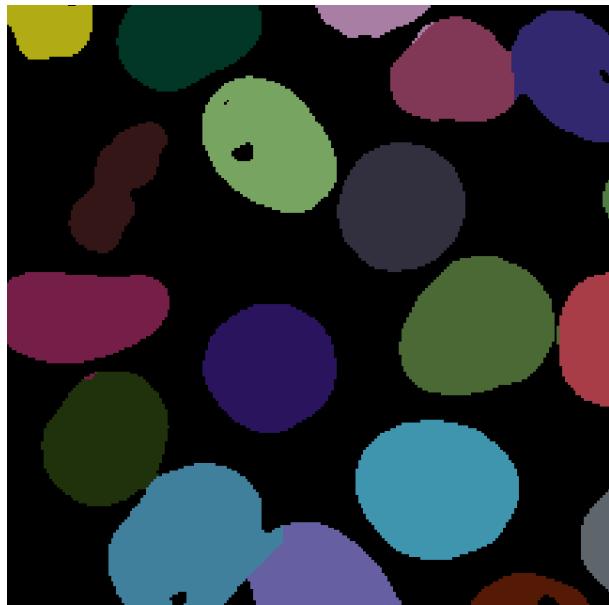
1024 x1024 x 100
16-bit image



1024 x1024 x 100
16-bit image



1024 x1024 x 100
8-bit image



1024 x1024 x 100
16-bit image

How much memory does
this workflow cost?

700 MB

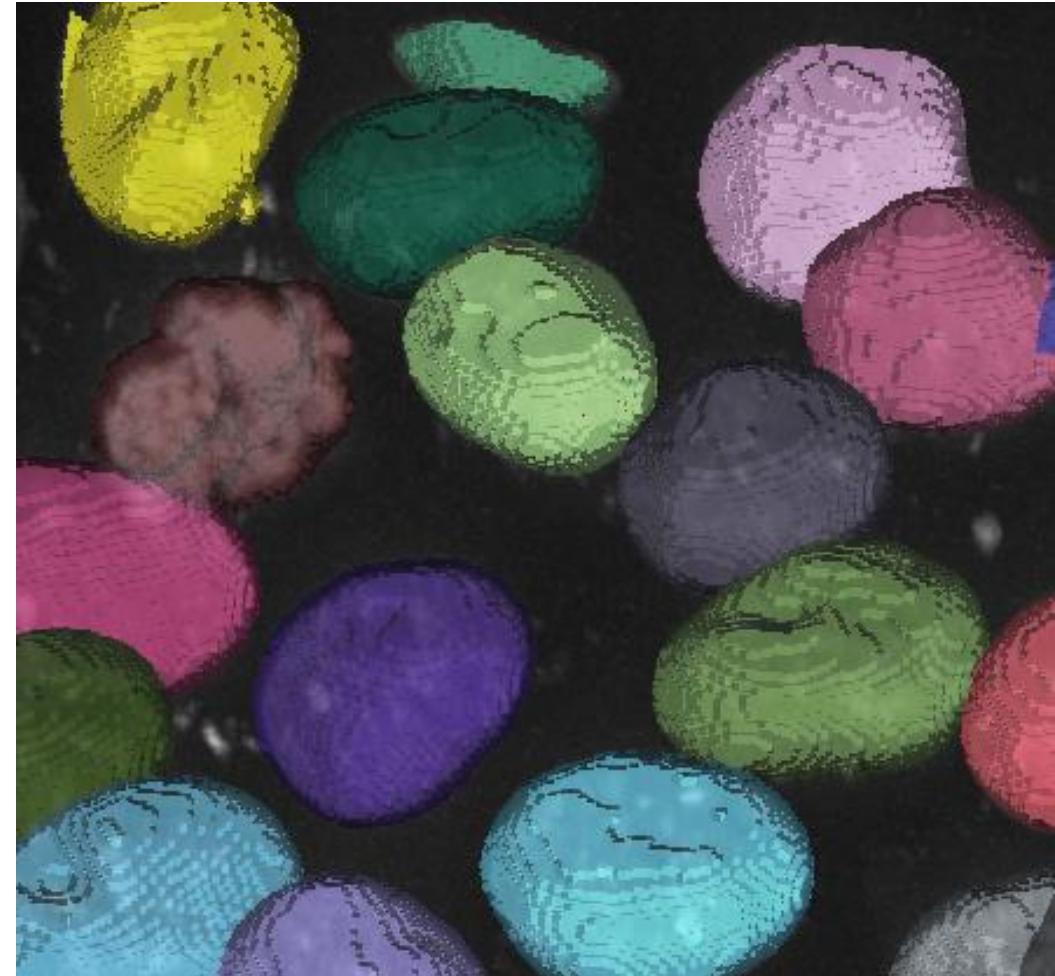
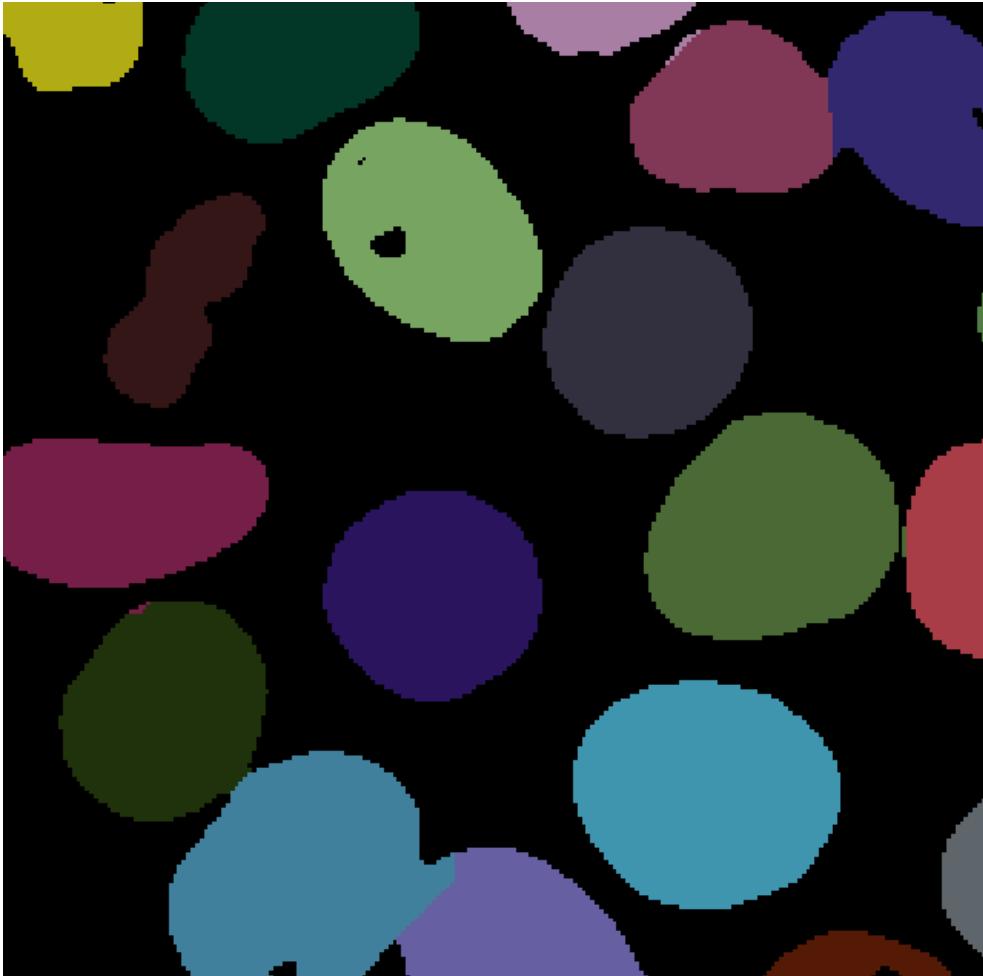
400 MB

4 GB

7 GB

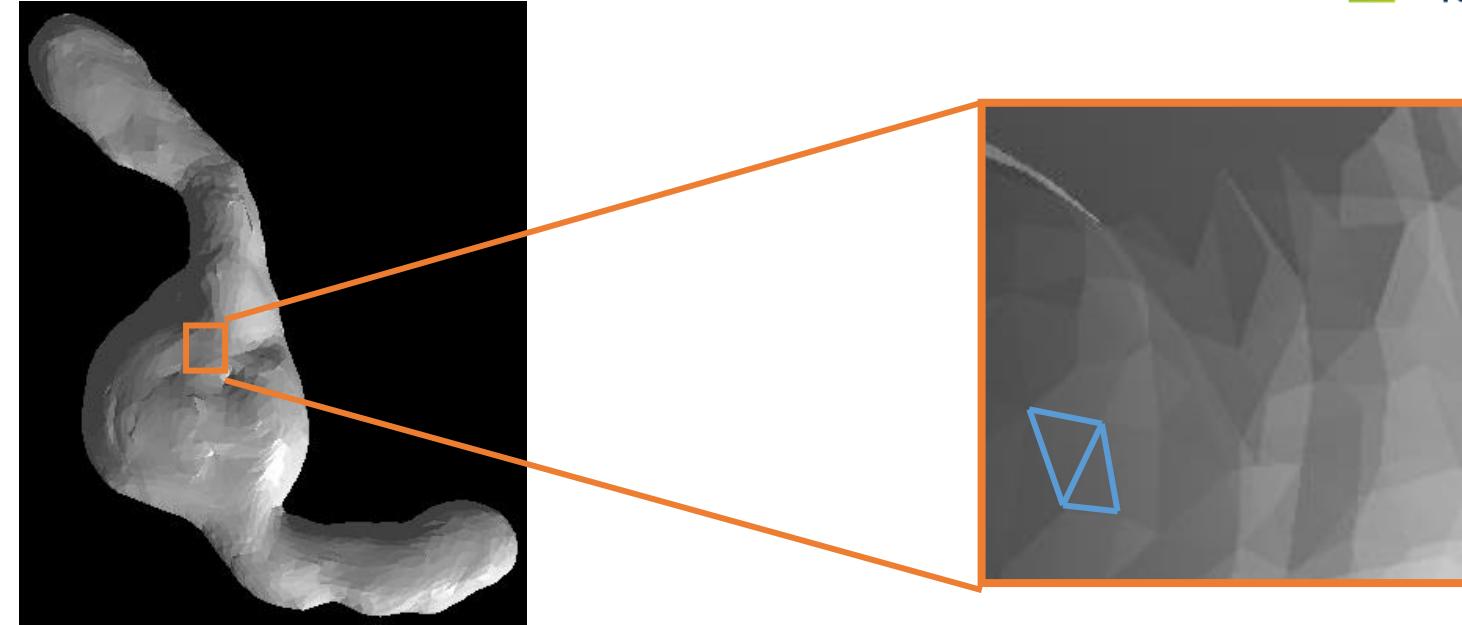
Motivation: Surface reconstruction

- Pixel and voxel borders introduce artifacts, potentially problematic for measurements, e.g. surface area



Surface meshes

- Points on a surfaces connected by triangles form a surface mesh



“Vertices” / points

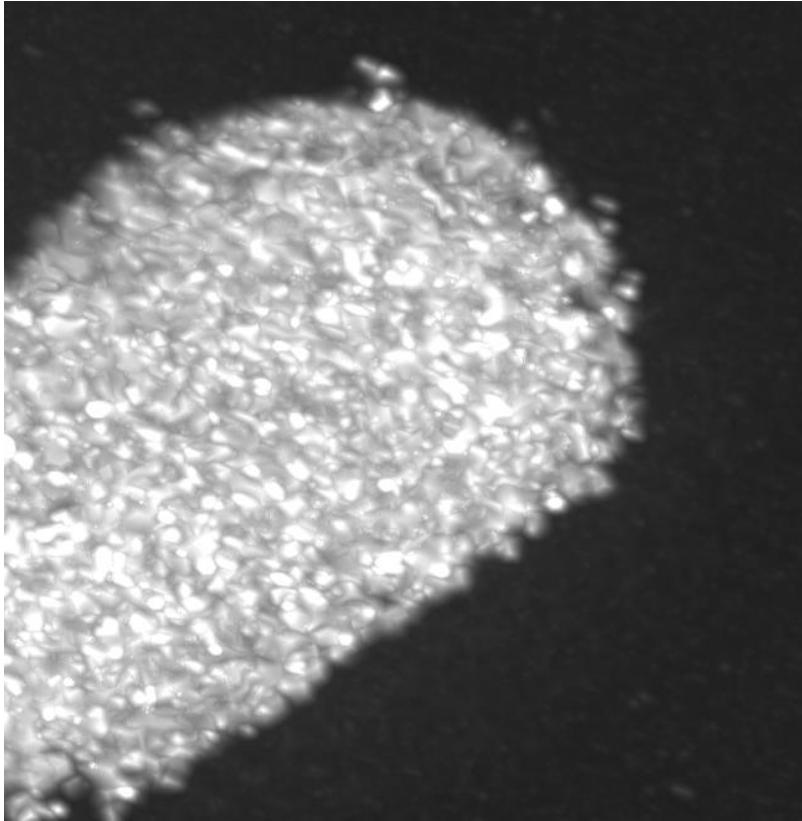
Point x	Point y	Point z
x_1	y_1	z_1
X_2	Y_2	Z_2
X_3	Y_3	Z_3
X_4	Y_4	Z_4
...

+

“Faces” / Triangles

Point 1	Point 2	Point 3
1	2	3
1	2	4
2	3	4
1	3	4

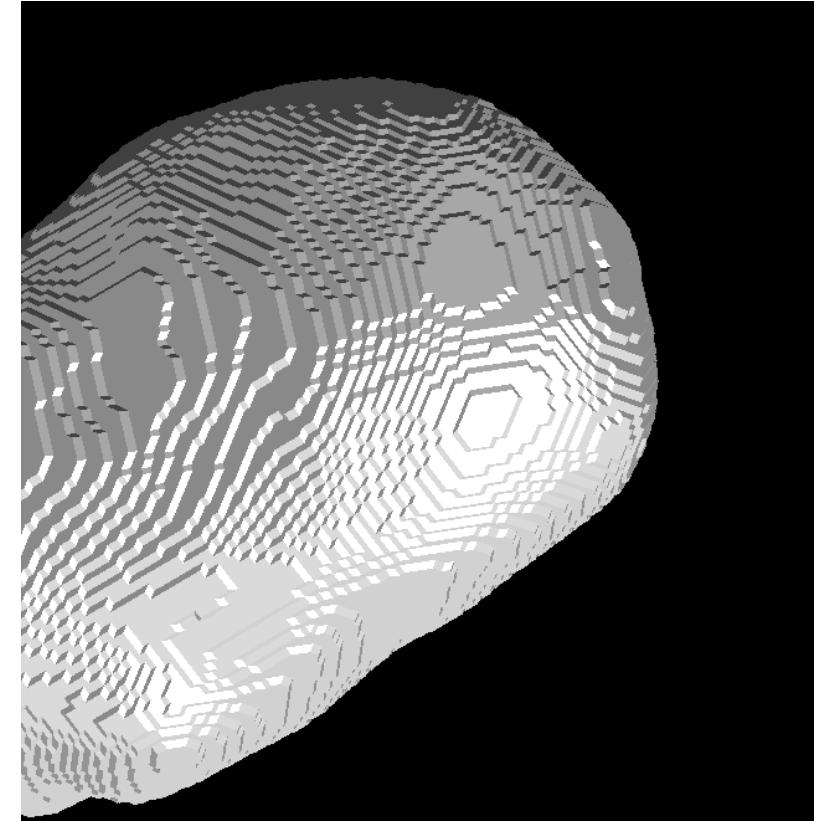
Surface reconstruction



3D image of nuclei



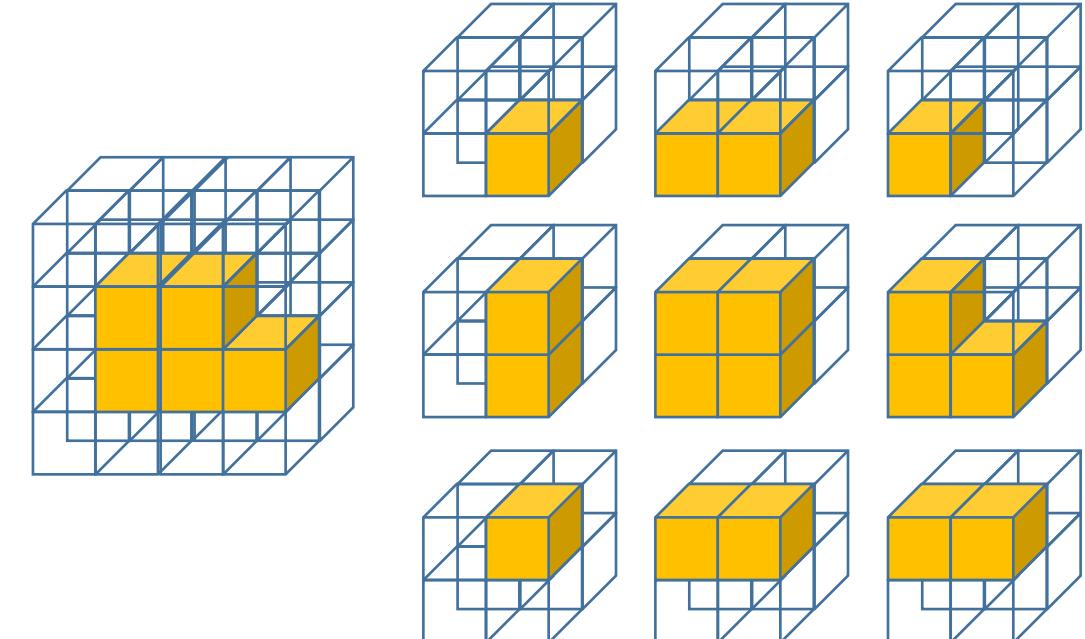
Gaussian filtered



Binary 3D image (visualized
as surface mesh)

Marching cubes algorithm

- Starting point: 3D binary image
- Cuts the image in small cubes and iterates over them

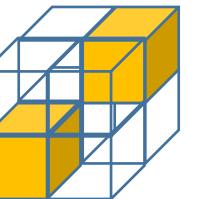
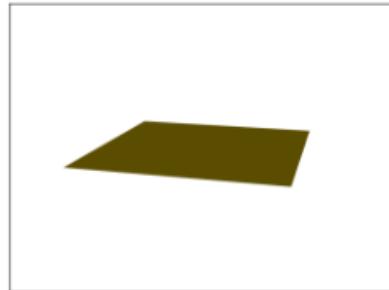
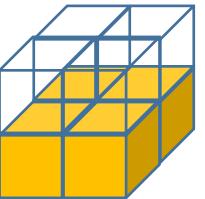
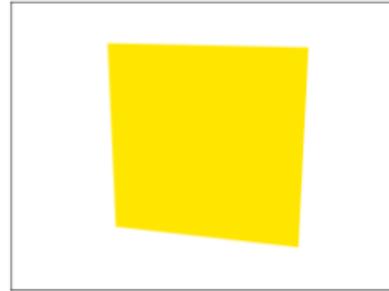
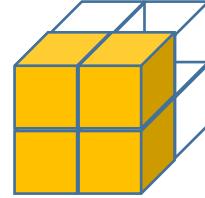
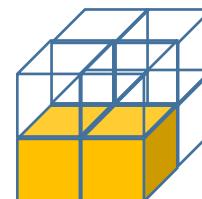
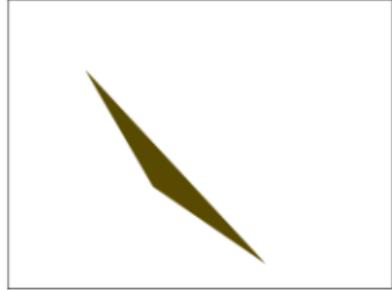
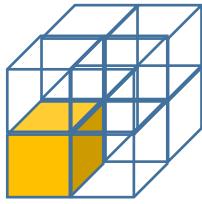
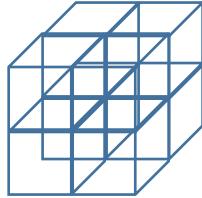


Split into cubes

Lorensen, William E.; Cline, Harvey E. (1 August 1987). "Marching cubes: A high resolution 3D surface construction algorithm". *ACM SIGGRAPH Computer Graphics*. **21** (4): 163-169. [CiteSeerX 10.1.1.545.613](https://doi.org/10.1145/37402.37422). doi:[10.1145/37402.37422](https://doi.org/10.1145/37402.37422).

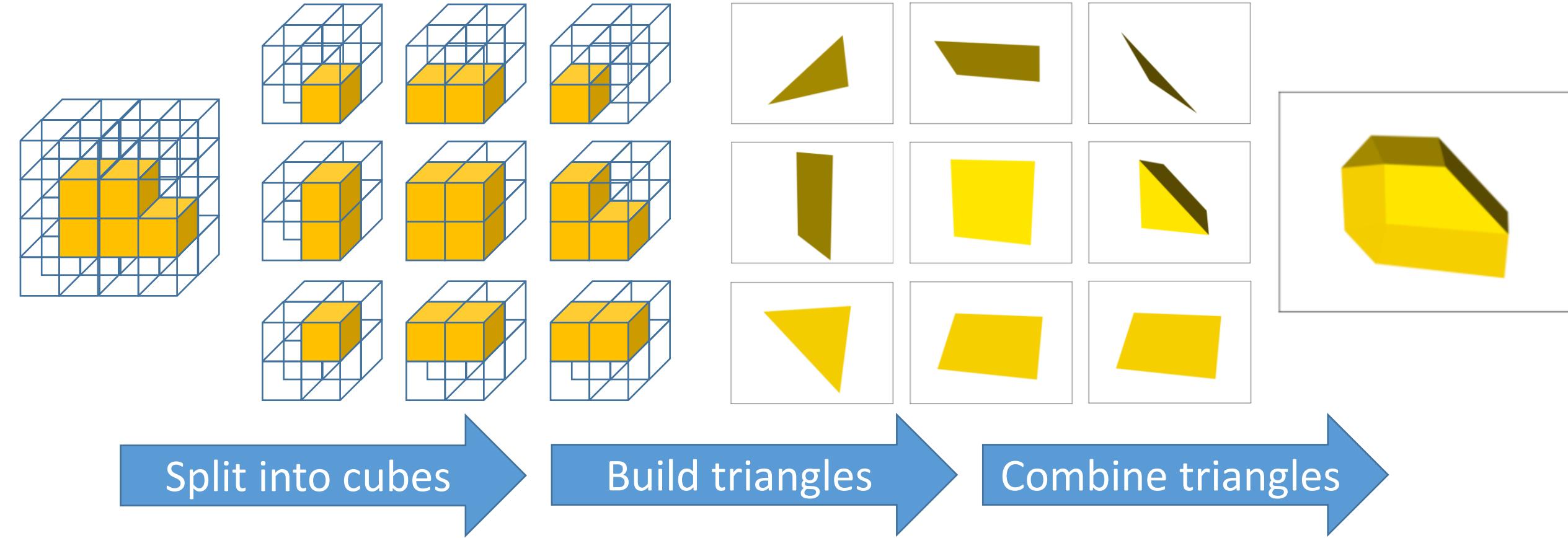
Marching cubes algorithm

- Starting point: 3D binary image
- Cuts the image in small cubes and iterates over them



Marching cubes algorithm

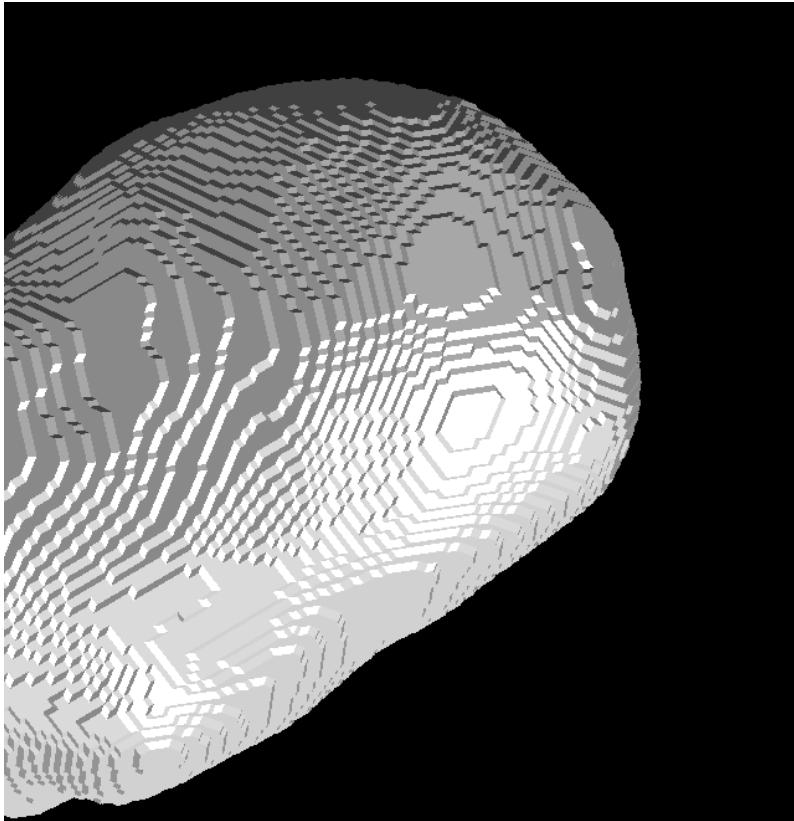
- Starting point: 3D binary image
- Cuts the image in small cubes and iterates over them



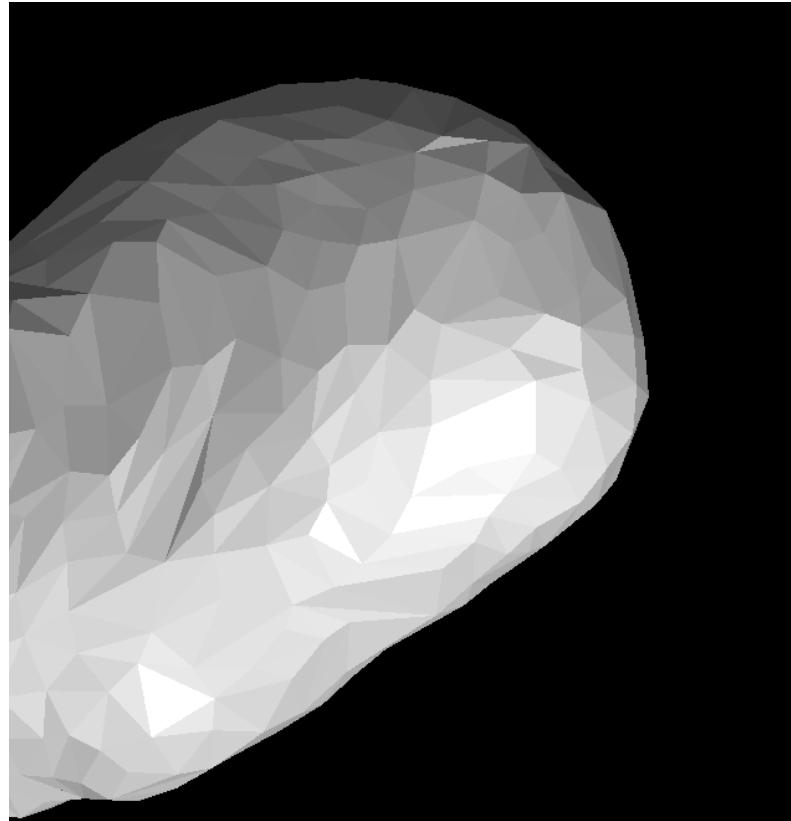
Lorensen, William E.; Cline, Harvey E. (1 August 1987). "Marching cubes: A high resolution 3D surface construction algorithm". *ACM SIGGRAPH Computer Graphics*. **21** (4): 163-169. [CiteSeerX 10.1.1.545.613](https://doi.org/10.1145/37402.37422). doi:[10.1145/37402.37422](https://doi.org/10.1145/37402.37422).

Surface post-processing

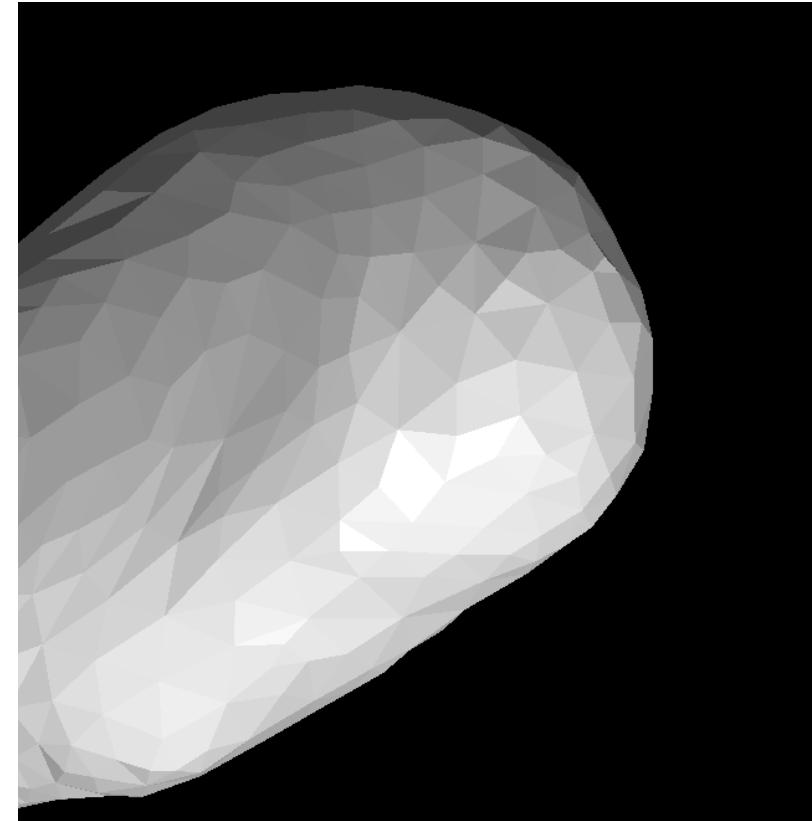
- Necessary to better match biological reality.



Marching cubes result



Simplified mesh
(less points, locally averaged)

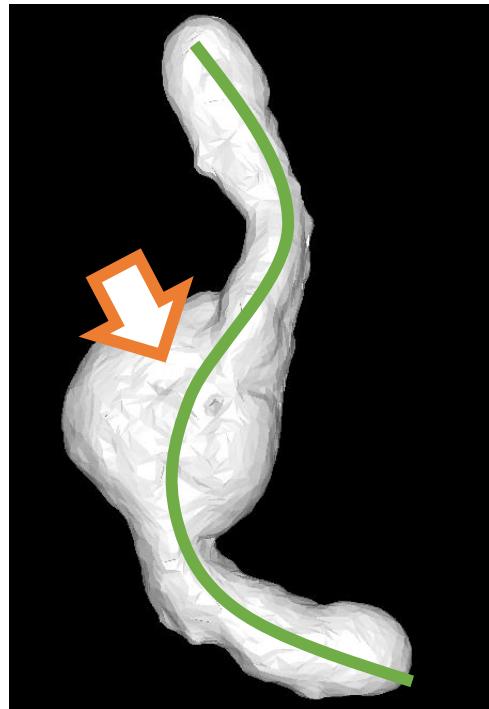


Smoothed mesh
(position locally planarized)

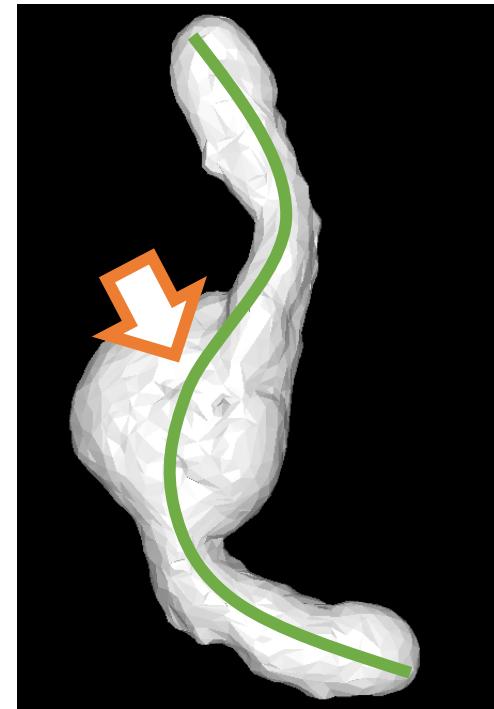
Data derived from [AV Luque and JV Veenvliet \(2023\)](#)
licensed [CC-BY](#) : <https://zenodo.org/record/7603081>

Surface post-processing

- Every processing step has consequences errors of later measurements
- Depends on desired measurement



Surface mesh



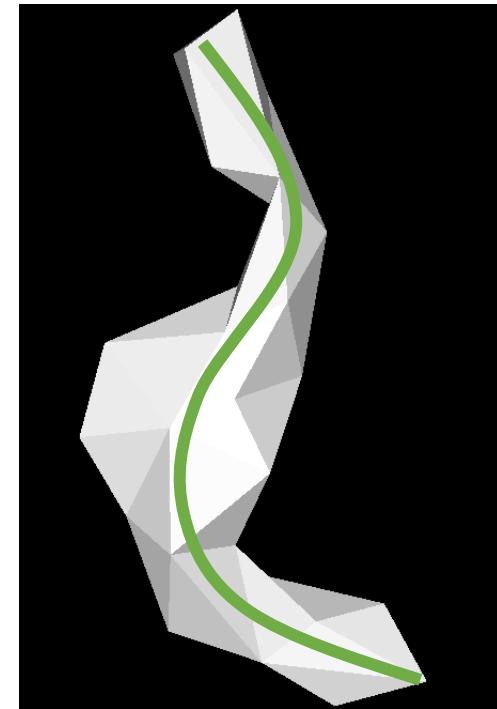
Simplified by factor 0.5

Number of small
concave regions



Simplified by factor 0.05

Total length

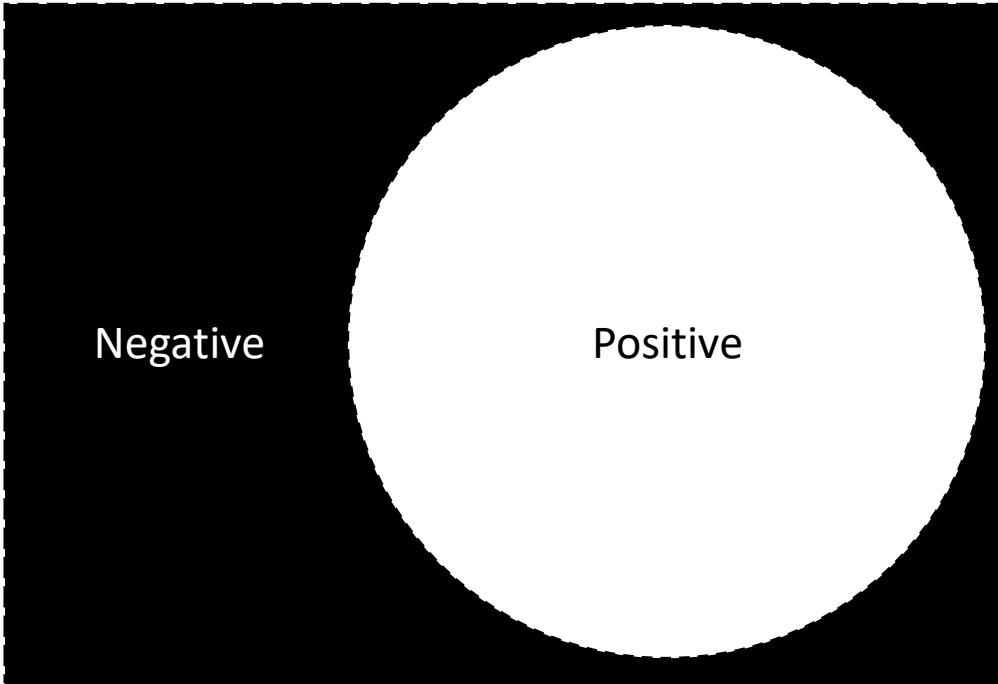


Simplified by factor 0.01

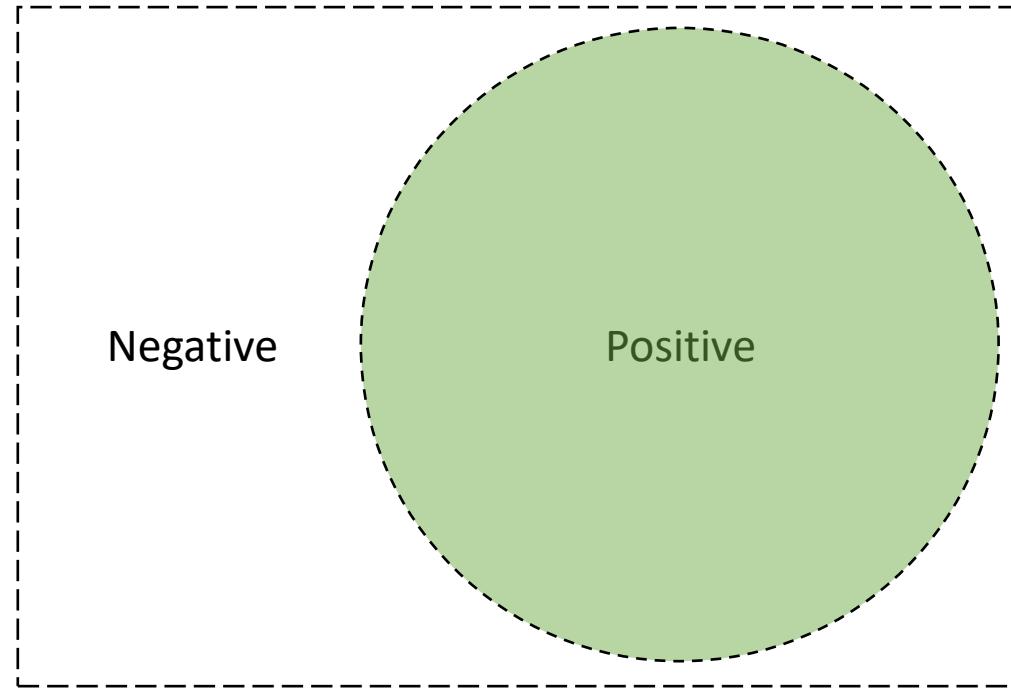
Segmentation quality estimation

Robert Haase

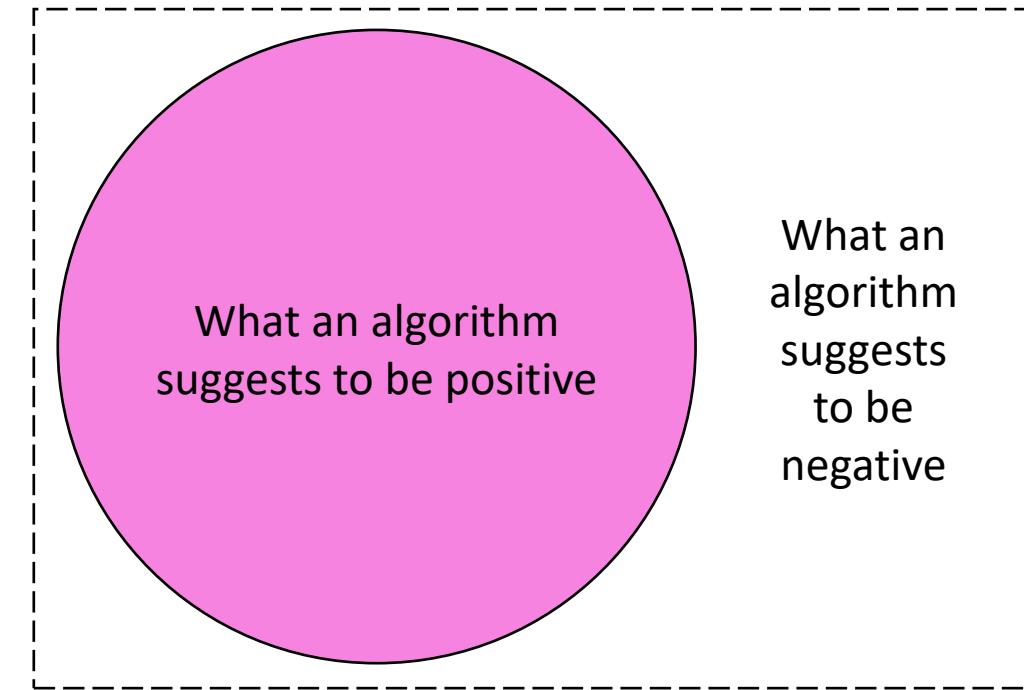
- In general
 - Define what's positive and what's negative.



- In general
 - Define what's positive and what's negative.

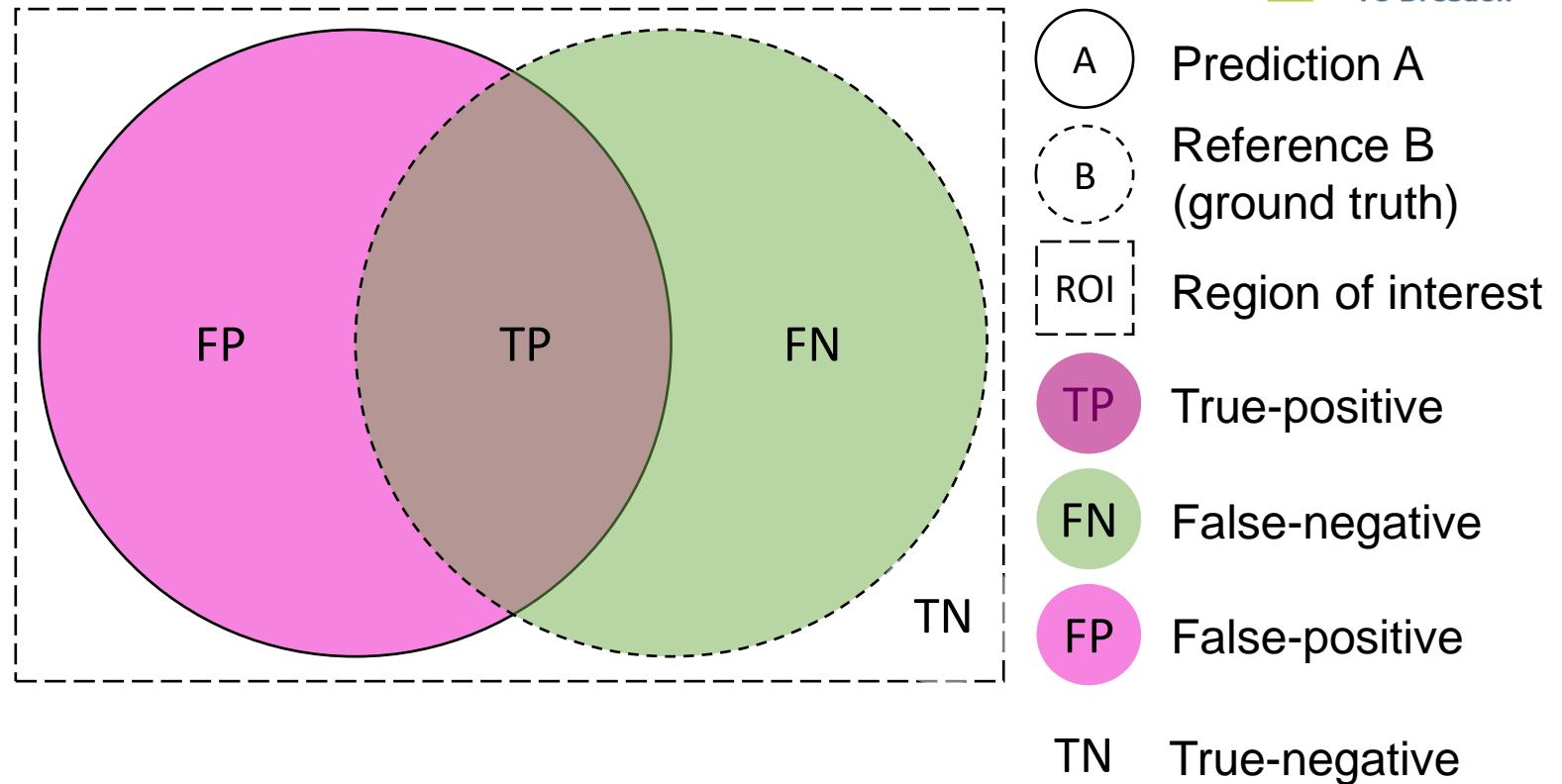


- In general
 - Define what's positive and what's negative.



Algorithm evaluation

- In general
 - Define what's positive and what's negative.
 - Compare with a reference to figure out what was true and false
- Welcome to the Theory of Sets



Precision

$$\frac{TP}{TP + FP}$$

What fraction of points that were predicted as positives were really positive?

Recall
(a.k.a. sensitivity)

$$\frac{TP}{TP + FN}$$

What fraction of positives points were predicted as positives?

Pixel-wise versus Object-wise evaluation

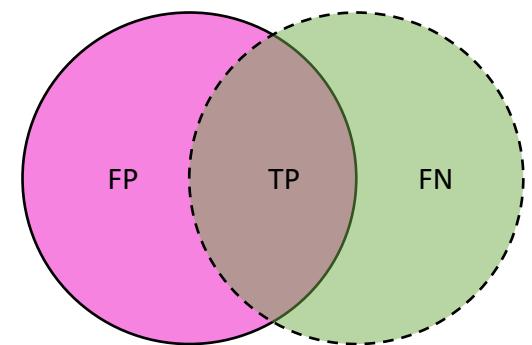
- Pixel wise: Segmentation quality
- Object wise: Detection quality



FP

TP

FN

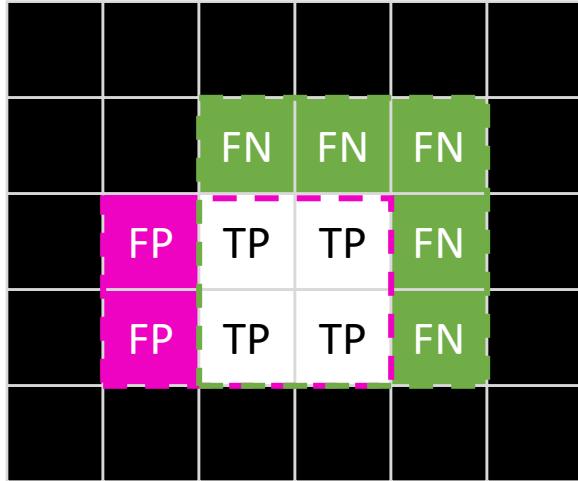


Precision

$$\frac{TP}{TP + FP}$$

Recall
(a.k.a. sensitivity)

$$\frac{TP}{TP + FN}$$



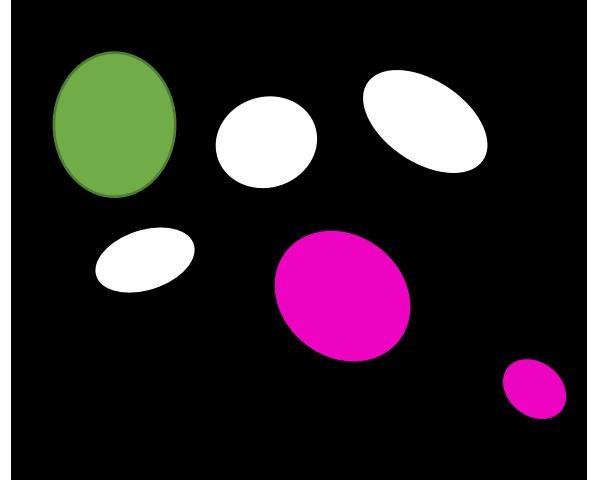
True-positive: 4

False-negative: 5

False-positive: 2

Precision: 4/6 = 66%

Recall: 4/9 = 44%



True-positive: 3

False-negative: 1

False-positive: 2

Precision: 3/4 = 75%

Recall: 3/5 = 60%

Pixel-wise versus Object-wise evaluation

- Average Overlap for all ground-truth objects
- <https://github.com/haesleinhuepf/the-segmentation-game>



- Voxel-wise Youden-Index

$$YI = p_{TP} + p_{TN} - 1$$

- Volume error

$$\Delta_V = V_A - V_B$$

$$\delta_V = \frac{\Delta_V}{V_B}$$

- Dice Index

$$DI(A, B) = \frac{2|A \cap B|}{|A| + |B|}$$

- Jaccard Index

$$JI(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{DI}{2 - DI}$$

- Contour distance

$$d_{e,min}(a, B) = \min(d_e(a, b) | b \in B)$$

$$\bar{d}_c(A, B) = \frac{\sum_{\forall a \in C(A)} d_{e,min}(a, C(B))}{|C(A)|}$$

$$\bar{d}_{bil,c}(A, B) = \frac{\bar{d}_c(A, B) + \bar{d}_c(B, A)}{2}$$

- Hausdorff distance

$$d_H(A, B) = \max(d_{e,min}(a, B) | a \in A)$$

$$d_{bil,H}(A, B) = \max(d_H(A, B), d_H(B, A))$$

- Simplified Hausdorff distance

$$d_H(A, B) = \max(d_{e,min}(a, C(B)) | a \in C(A))$$

- Volume standard deviation

$$\delta_{\bar{V}} = 2 \frac{|V_A - V_B|}{|V_A + V_B|}$$

- Classification error

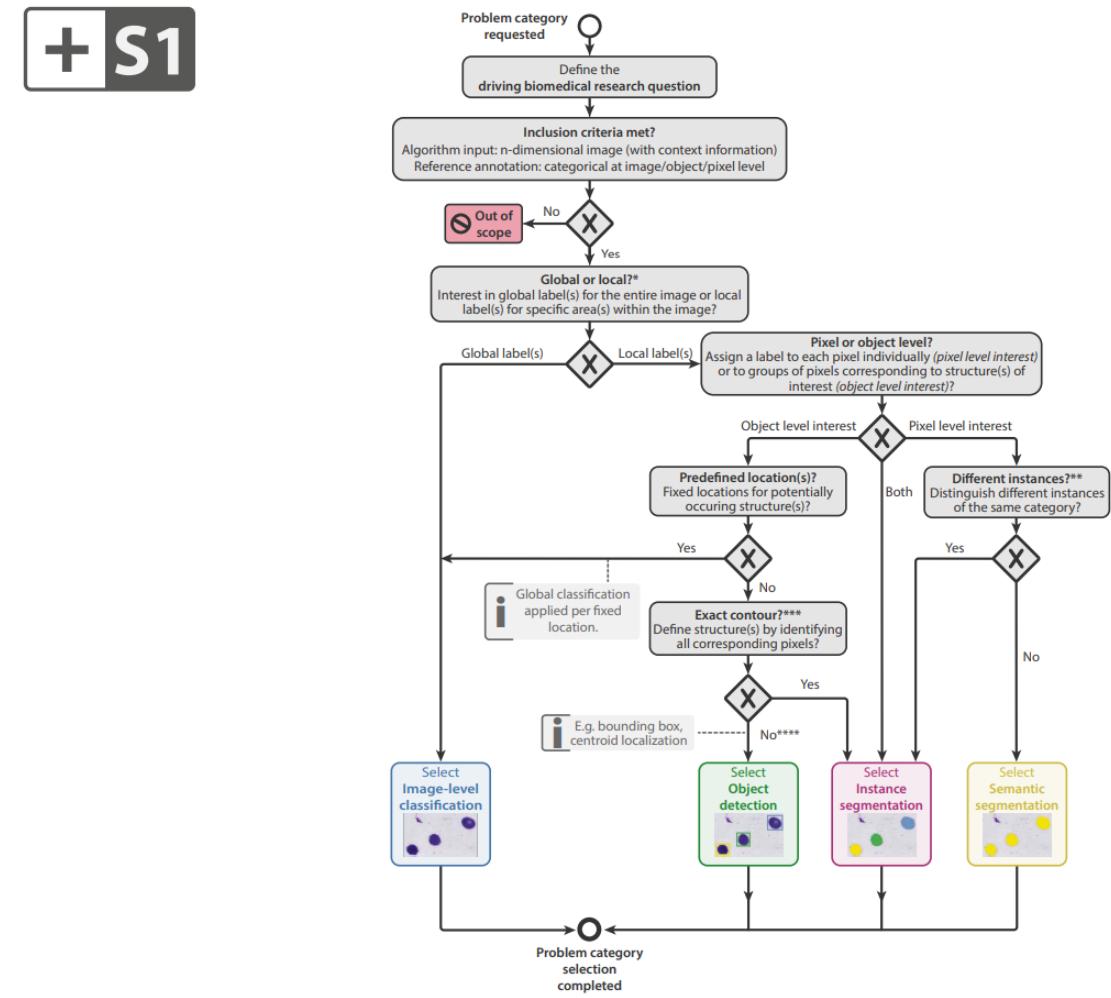
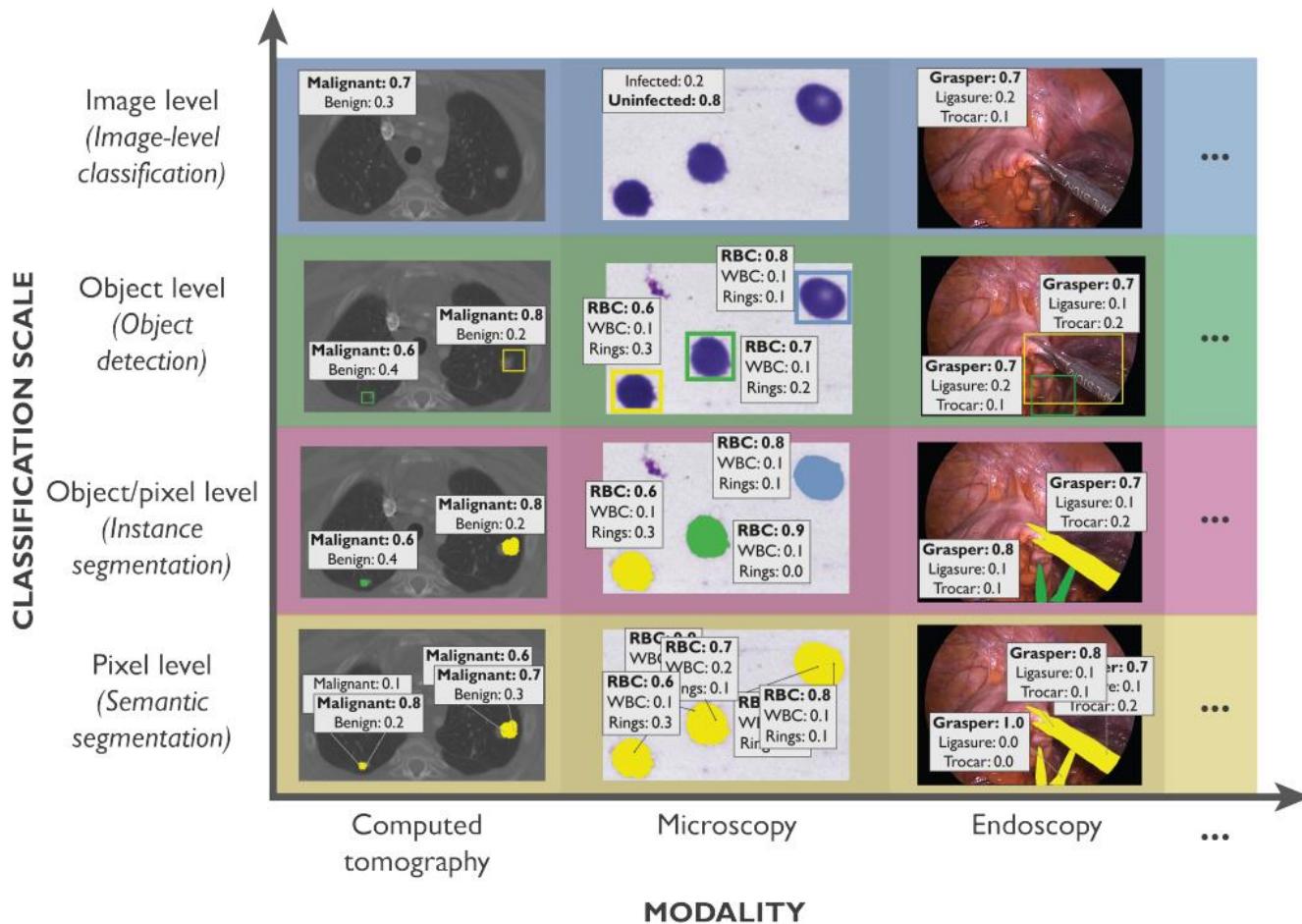
$$e_{Class} = \frac{H}{|TP| + |FN|}$$

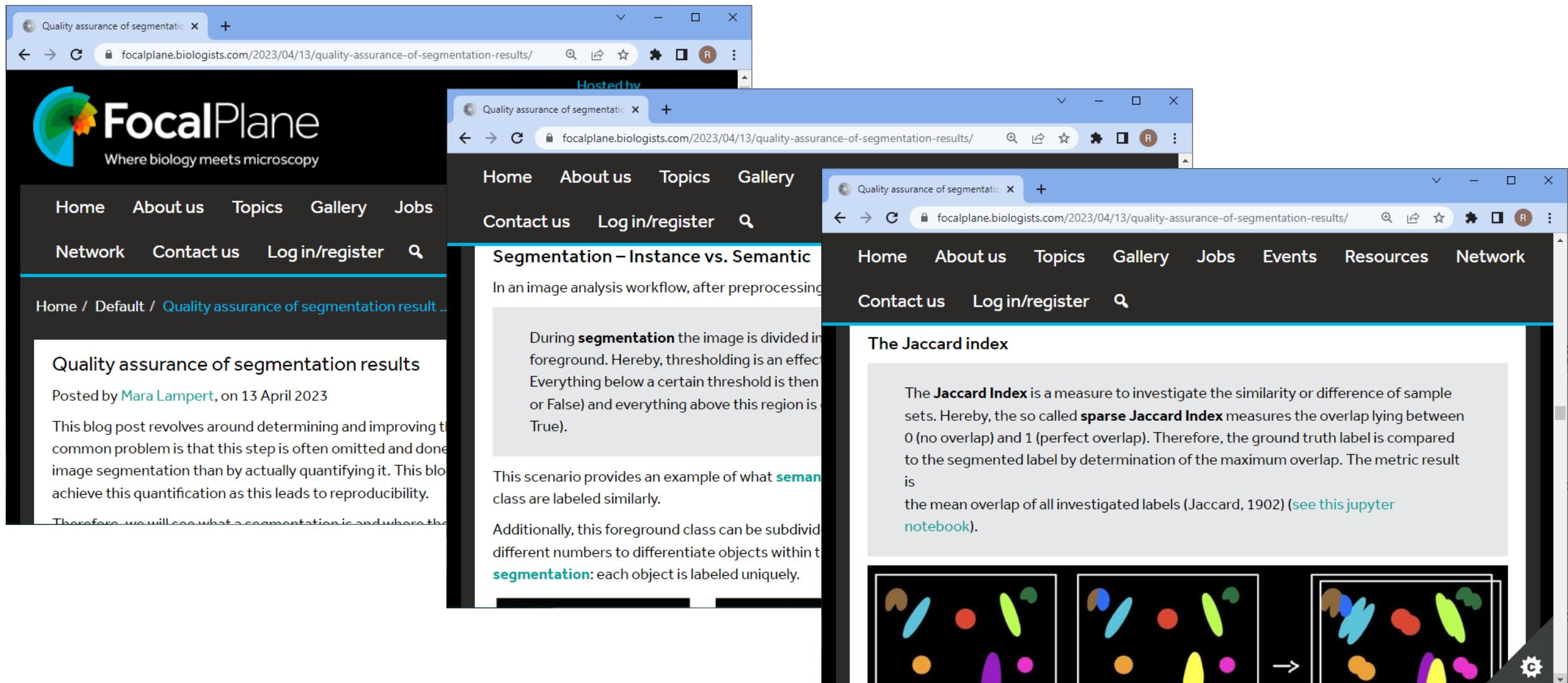
- Hamming distance

$$d_h = |A \cup B| - |A \cap B| \\ = |FP| + |FN|$$

What metric to use when?

- “Metrics reloaded: Pitfalls and recommendations for image analysis validation”
Maier-Hein, Reinke et al. <https://arxiv.org/abs/2206.01653>





The image shows three side-by-side screenshots of a blog post titled "Quality assurance of segmentation results" from the FocalPlane website. The post discusses the difference between instance and semantic segmentation and introduces the Jaccard index.

Screenshot 1: Introduction and Instance Segmentation

Header: FocalPlane - Where biology meets microscopy

Content: "Quality assurance of segmentation results" by Mara Lampert, posted on 13 April 2023. The text explains that segmentation divides an image into foreground and background. Thresholding is used to identify regions above a certain threshold as True (or False) and everything below as False. It also mentions that foreground classes can be subdivided into different numbers to differentiate objects within a scene.

Screenshot 2: Semantic Segmentation and the Jaccard Index

Header: FocalPlane - Where biology meets microscopy

Content: "Segmentation – Instance vs. Semantic". The text continues from the previous screenshot, explaining that semantic segmentation ensures each object is labeled uniquely. It then introduces the Jaccard index as a measure of similarity or difference between sample sets.

Screenshot 3: Visualizing Segmentation Results

Header: FocalPlane - Where biology meets microscopy

Content: "The Jaccard index". The text provides a detailed explanation of the Jaccard index, stating it measures overlap between 0 (no overlap) and 1 (perfect overlap). It compares ground truth labels with segmented labels and calculates the mean overlap of all investigated labels. Below the text are three panels showing grayscale images with colored overlays representing different segmentation results.

<https://focalplane.biologists.com/2023/04/13/quality-assurance-of-segmentation-results/>

Image segmentation in Napari

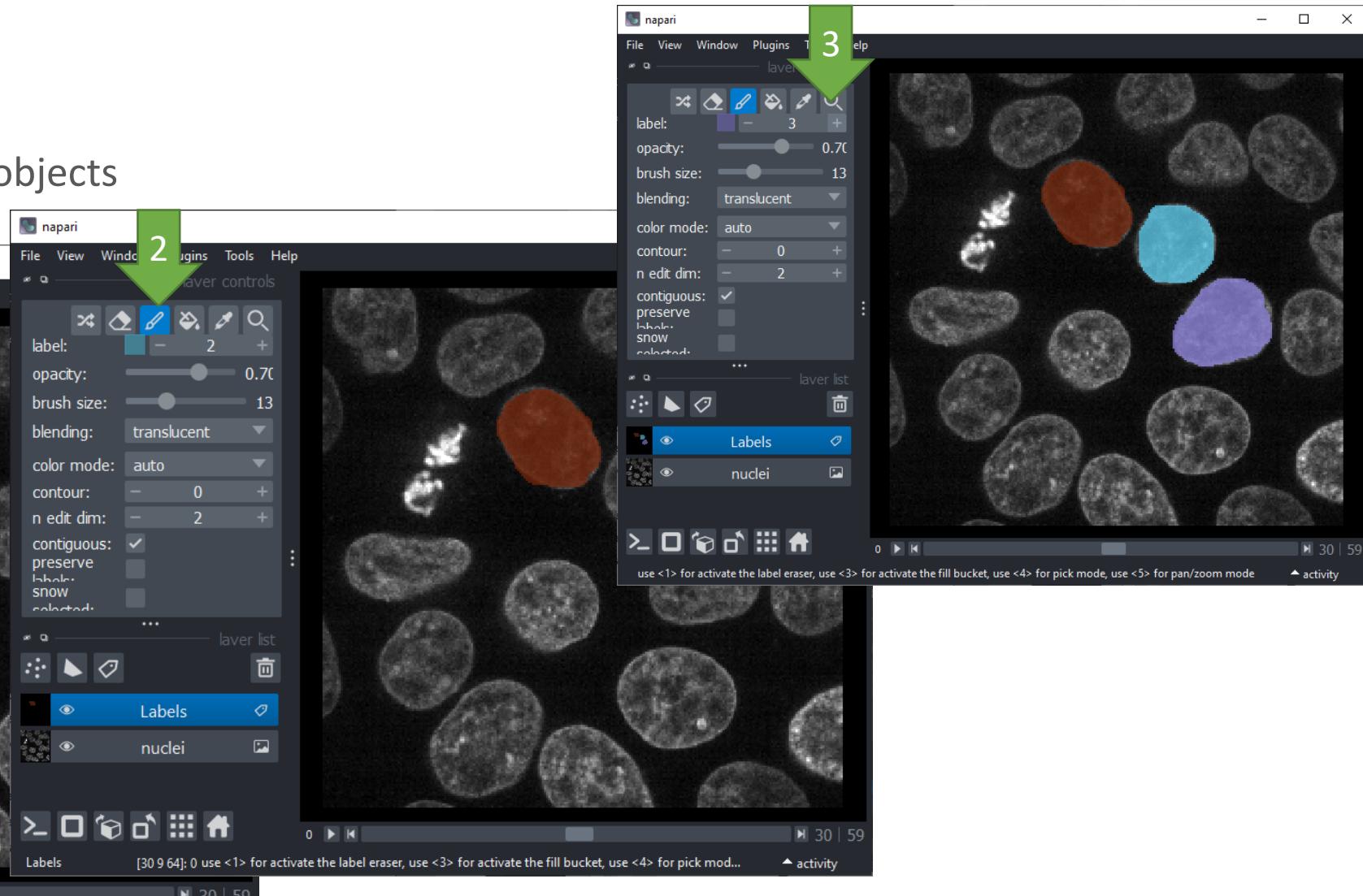
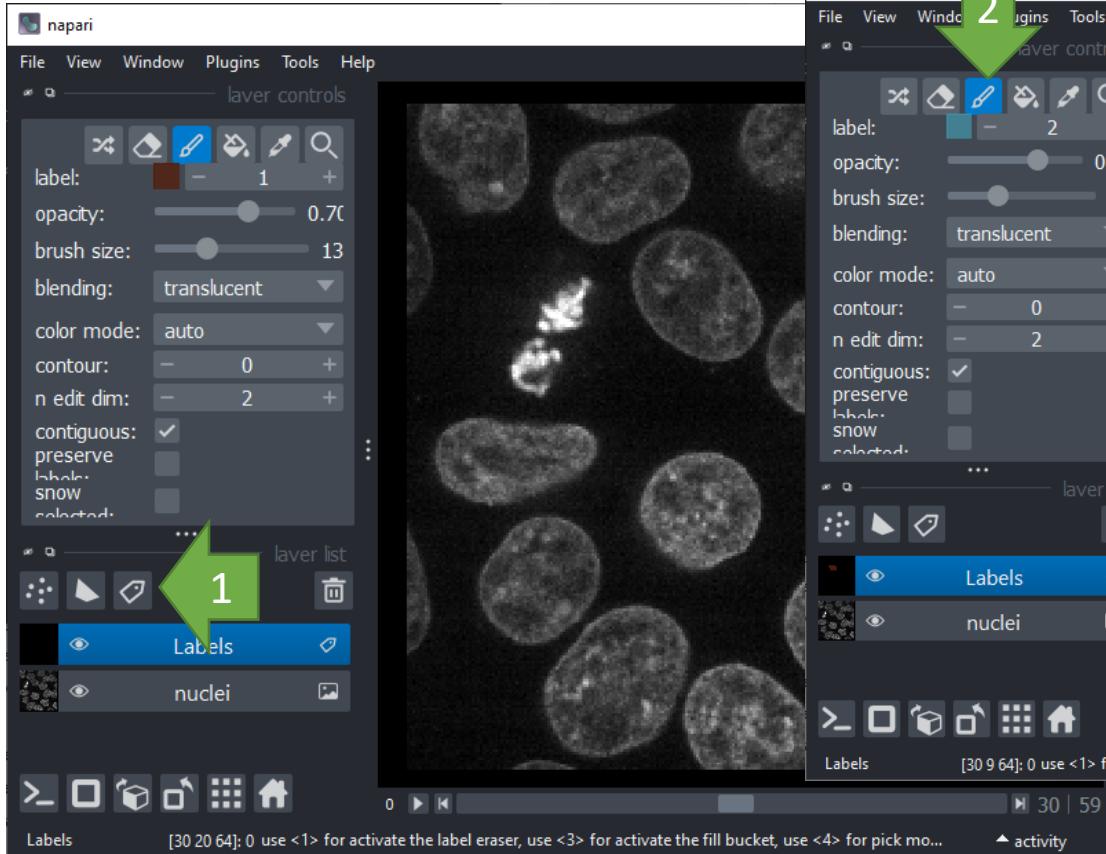
Robert Haase

Using materials from
Ryan Savill George, MPI CBG Dresden

April 2023

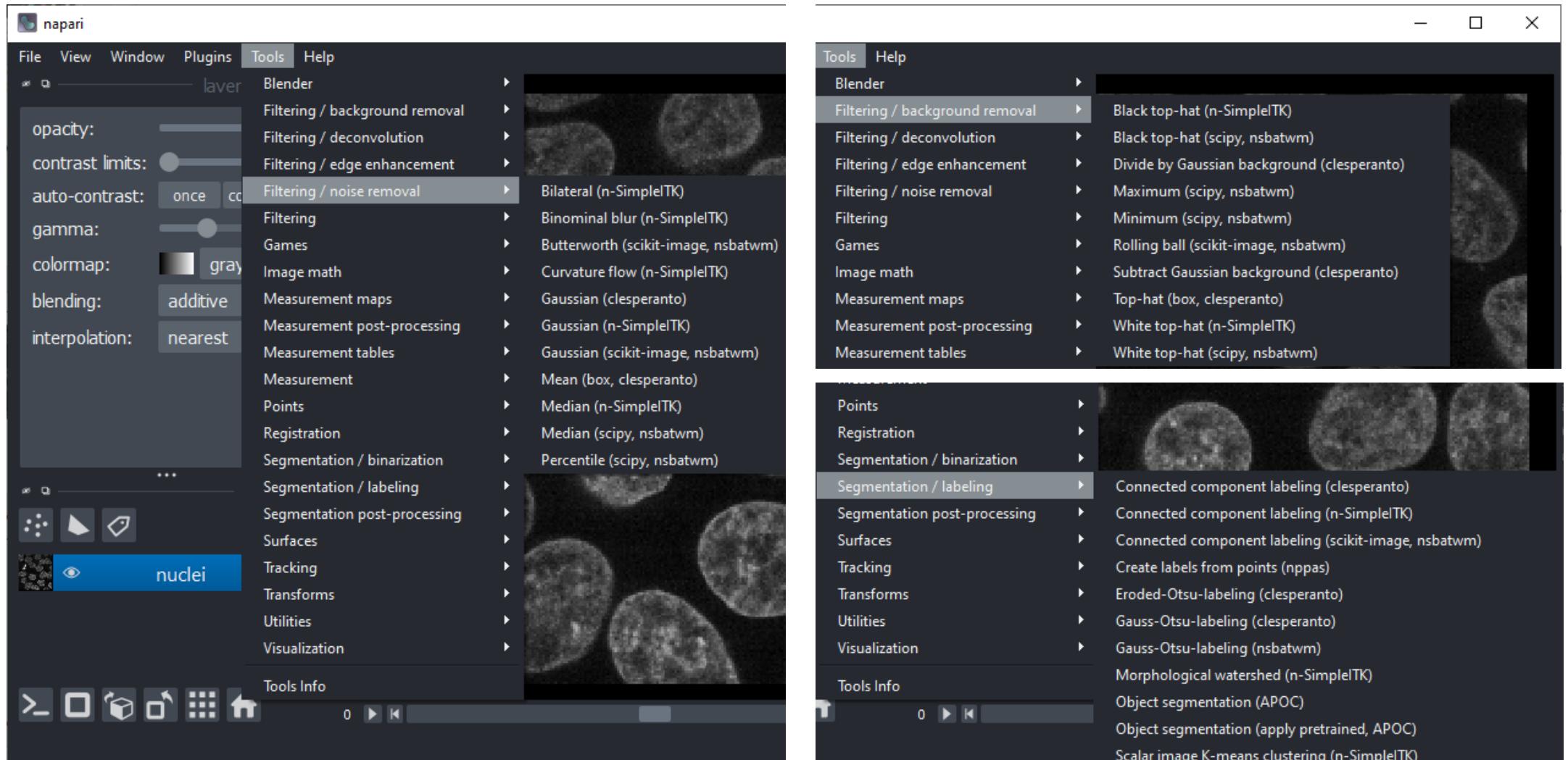
Manual annotation

1. Create a labels layer
2. Annotate first object
3. Increase label, annotate more objects



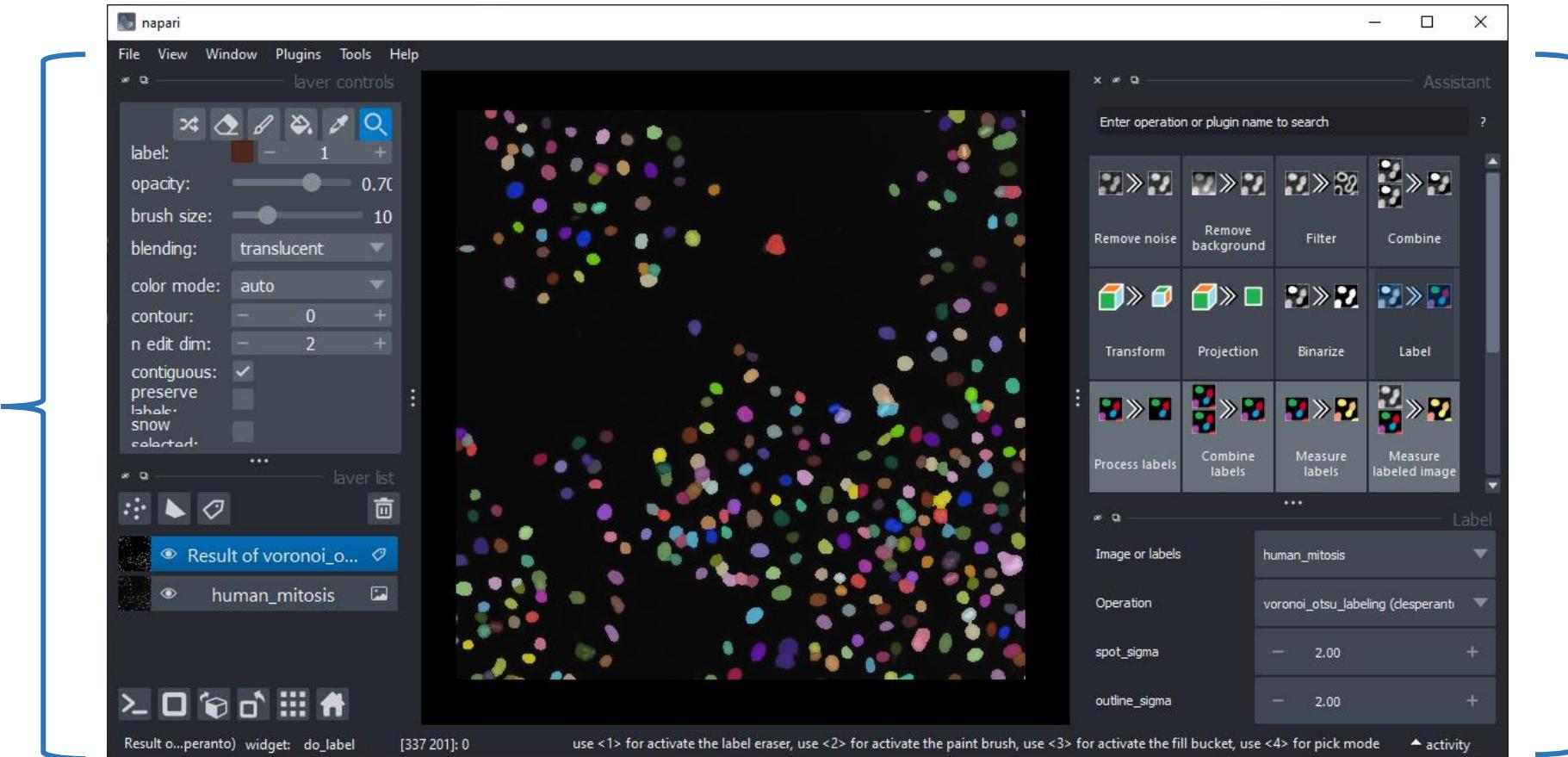
The Tools menu

- Organized in categories similar to what you learned last week



The Napari Assistant

- Tools > Utilities > Assistant (na)

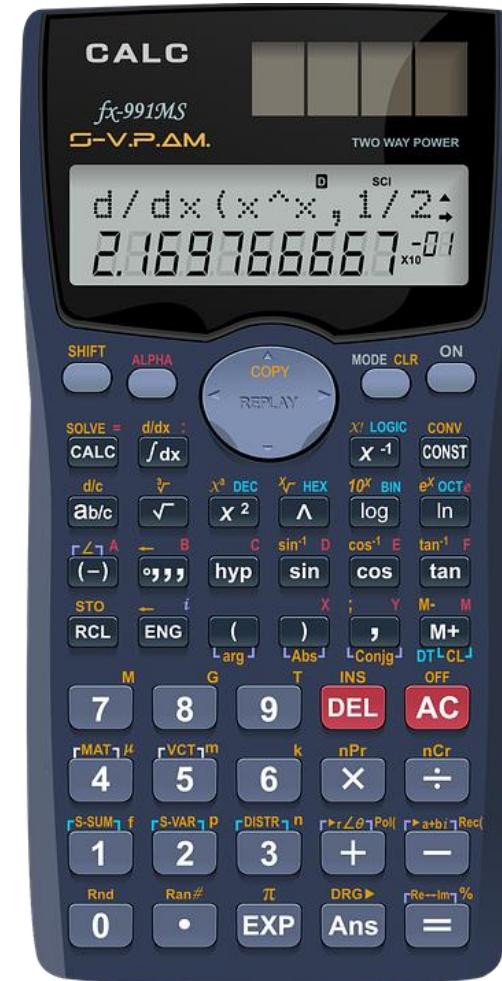
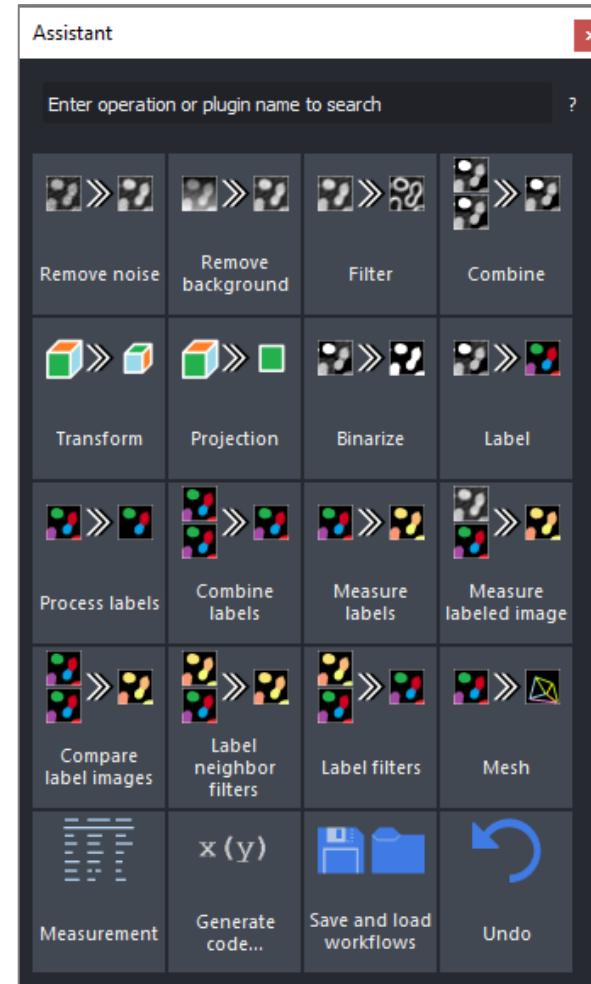


Viewer
controls

Image
Processing

The Napari Assistant

- A pocket-calculator-like interface to build image analysis workflows



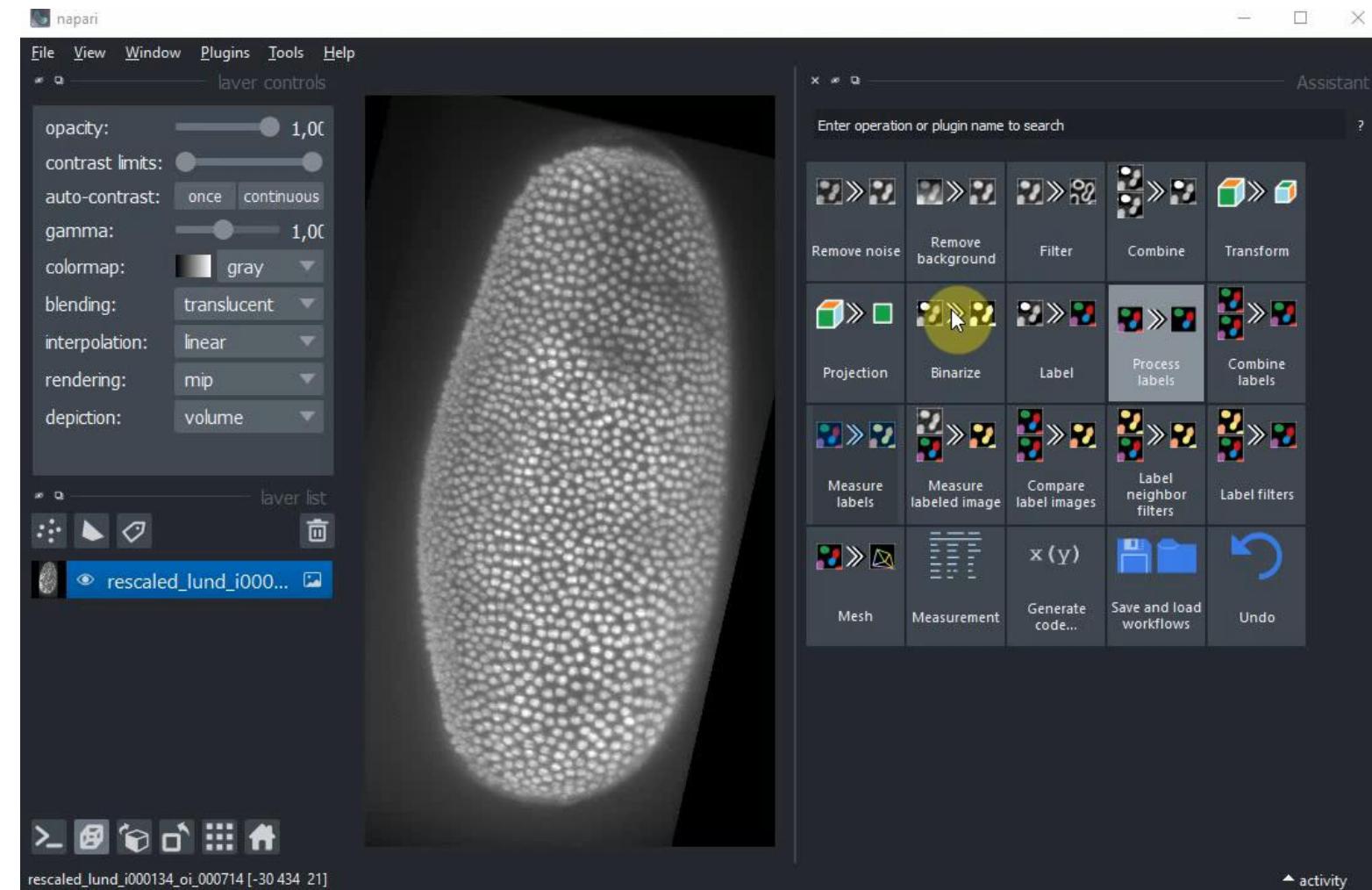
The Napari Assistant

- Classical image processing operations + advanced tools
- Saving&loading supported
- Undo [redo]
- Hints for next steps
- ...

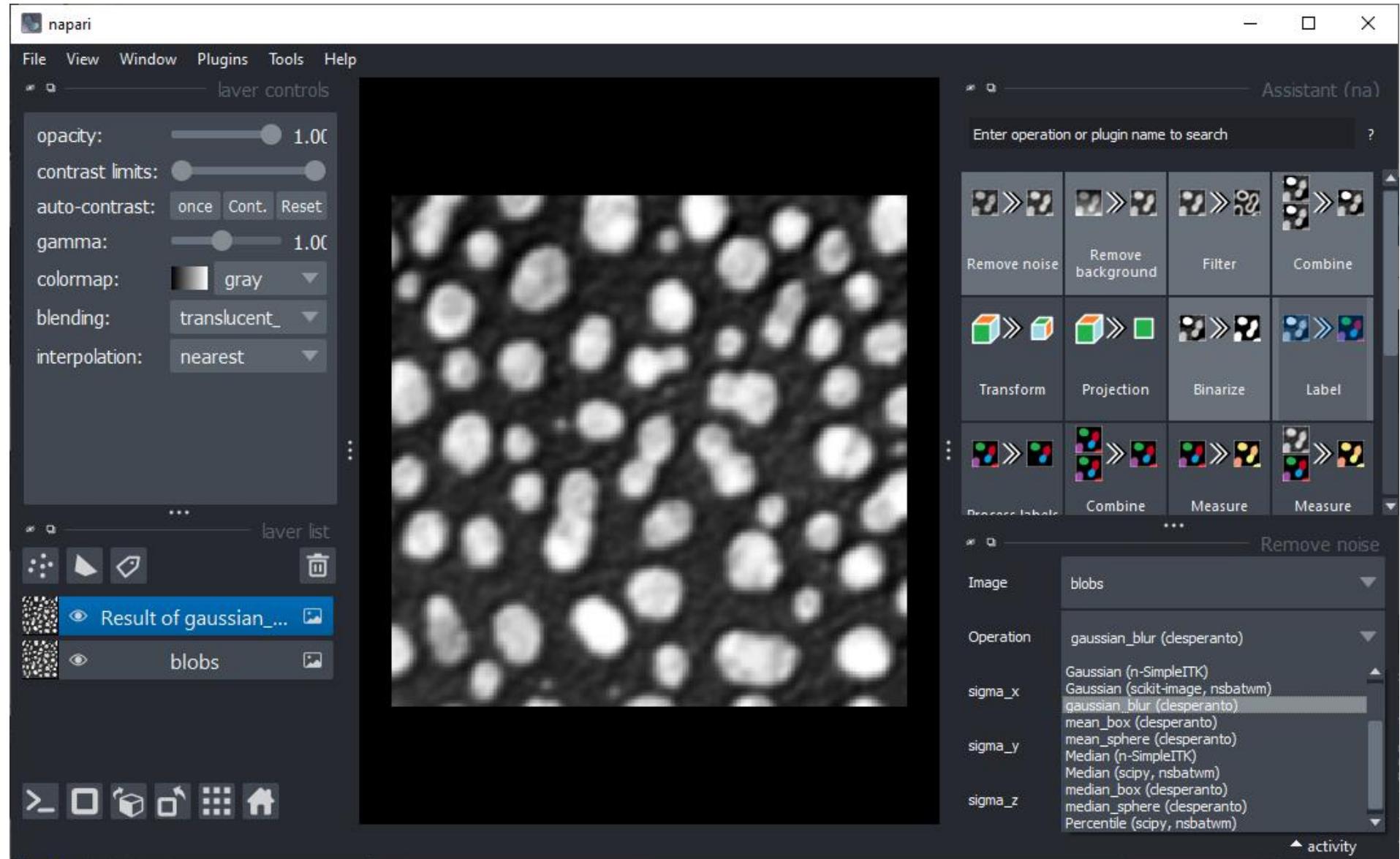
Big thanks to:



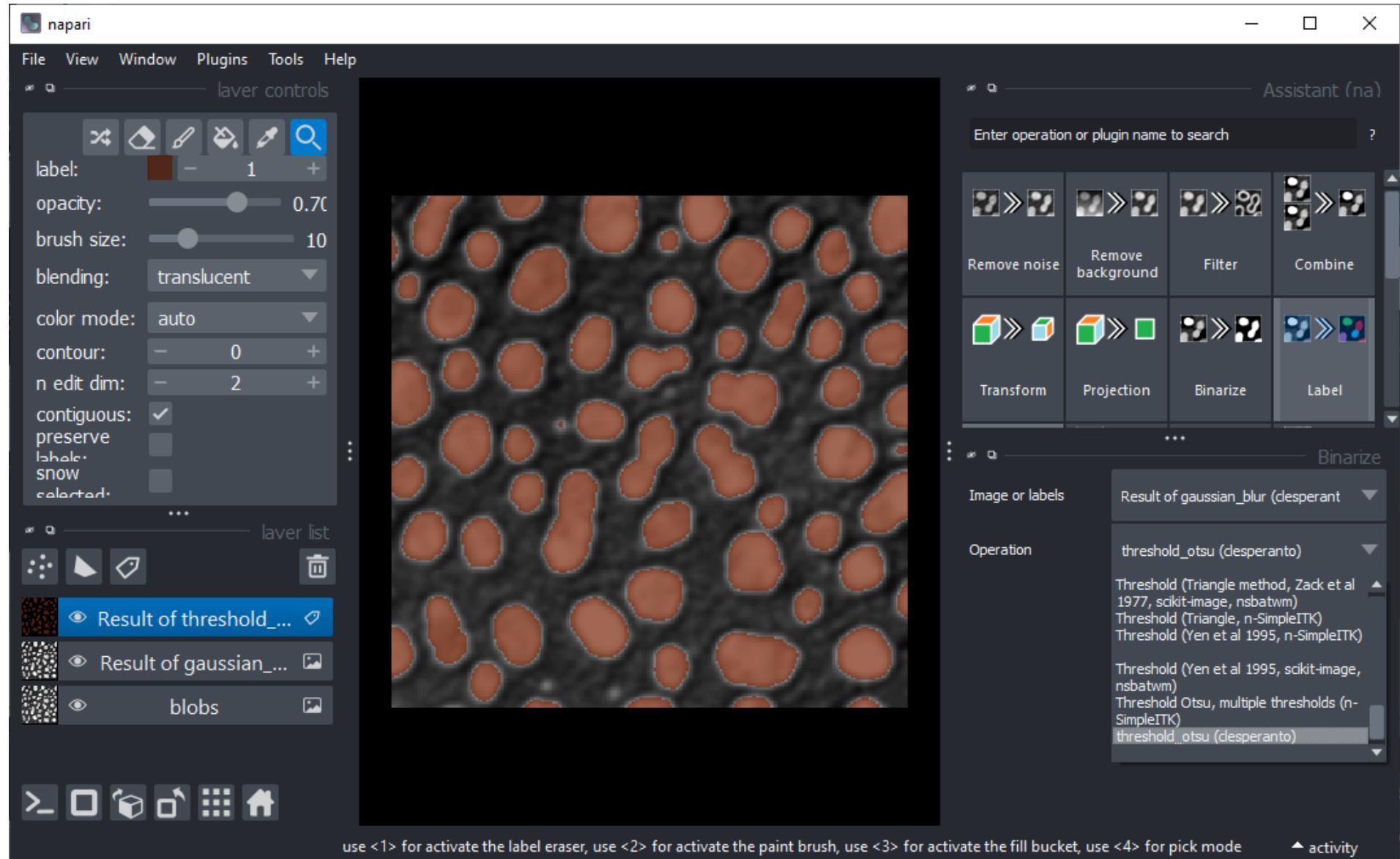
Ryan Savill
@RyanSavill4



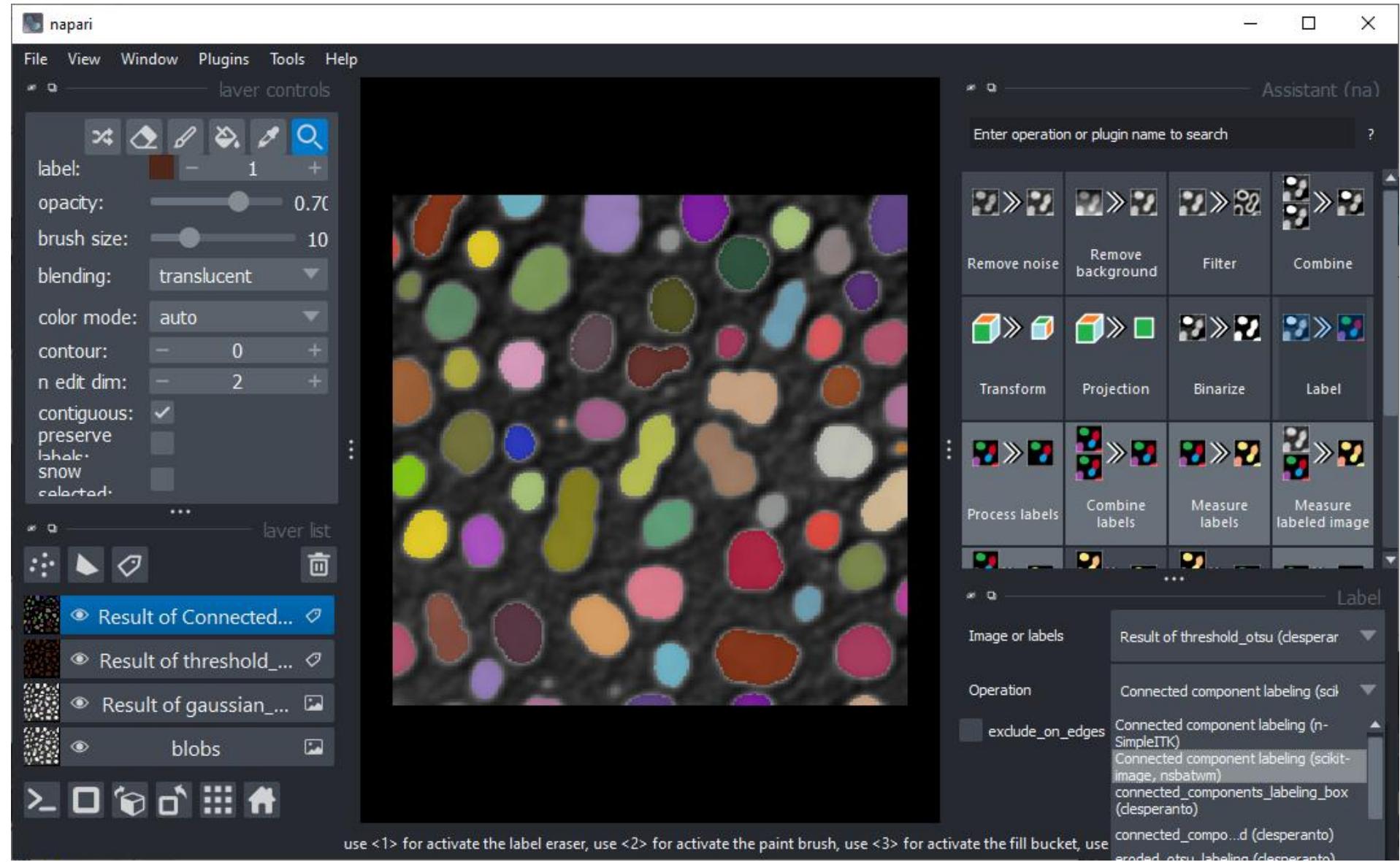
Workflow building



Workflow building

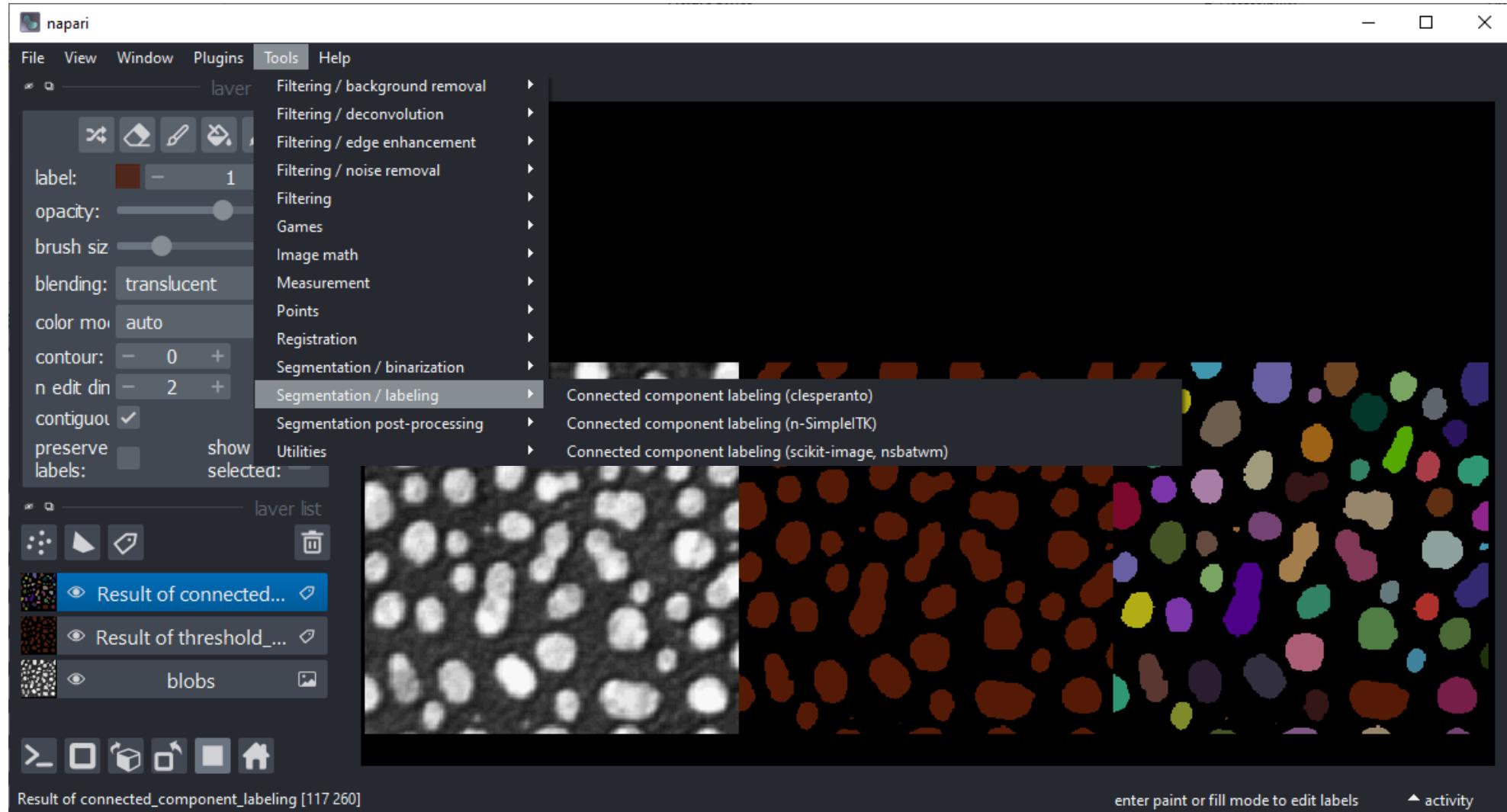


Workflow building



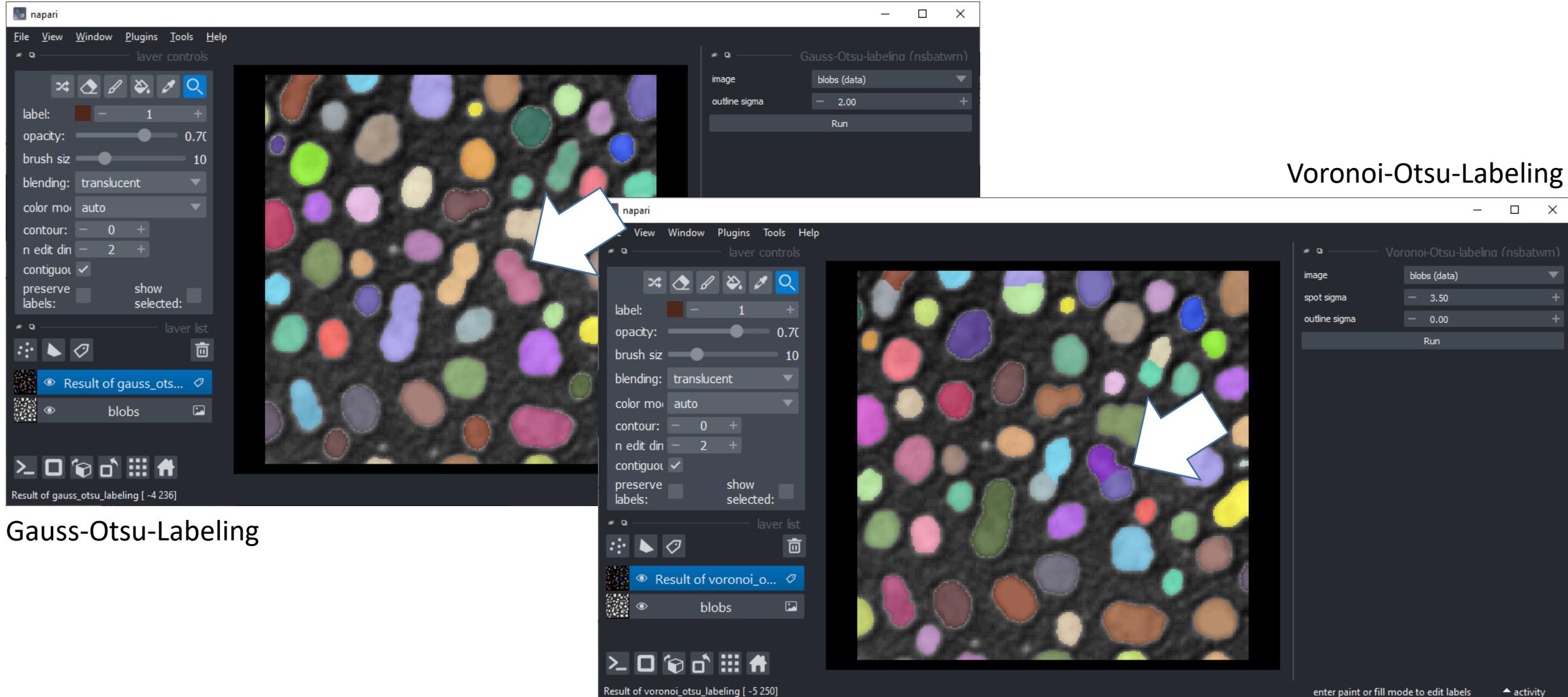
Workflow building

- In napari: Tools > Segmentation / labeling menu



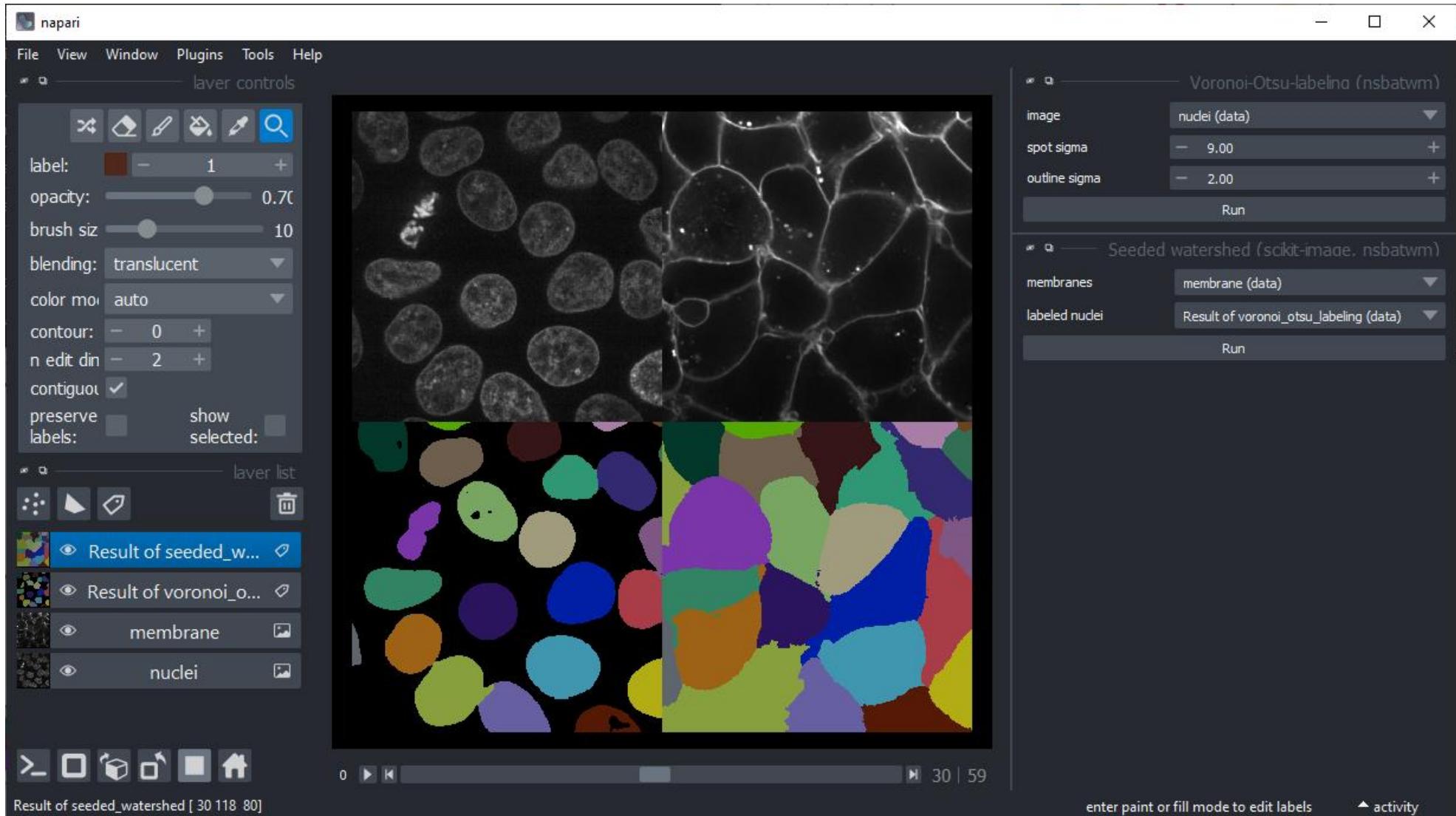
Short-cuts: Voronoi-Otsu-Labeling

- In Napari: Tools > Segmentation / labeling menu



Watershed

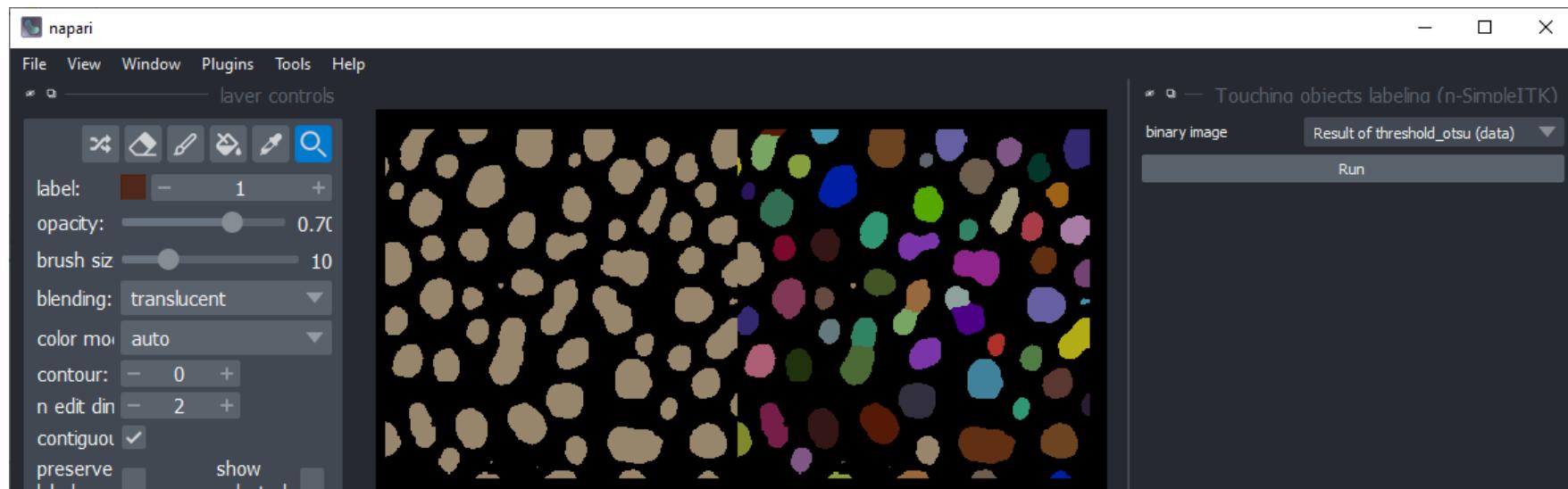
- ... in Napari practice: Tools > Segmentation / Labeling menu



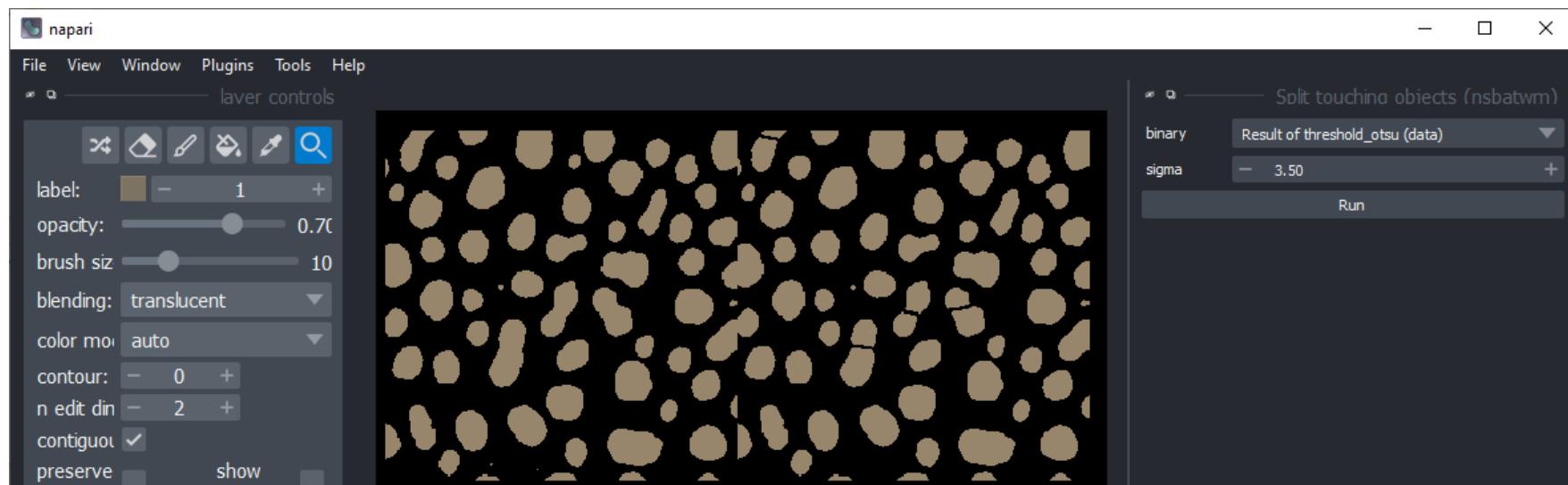
Watershed

- From binary images

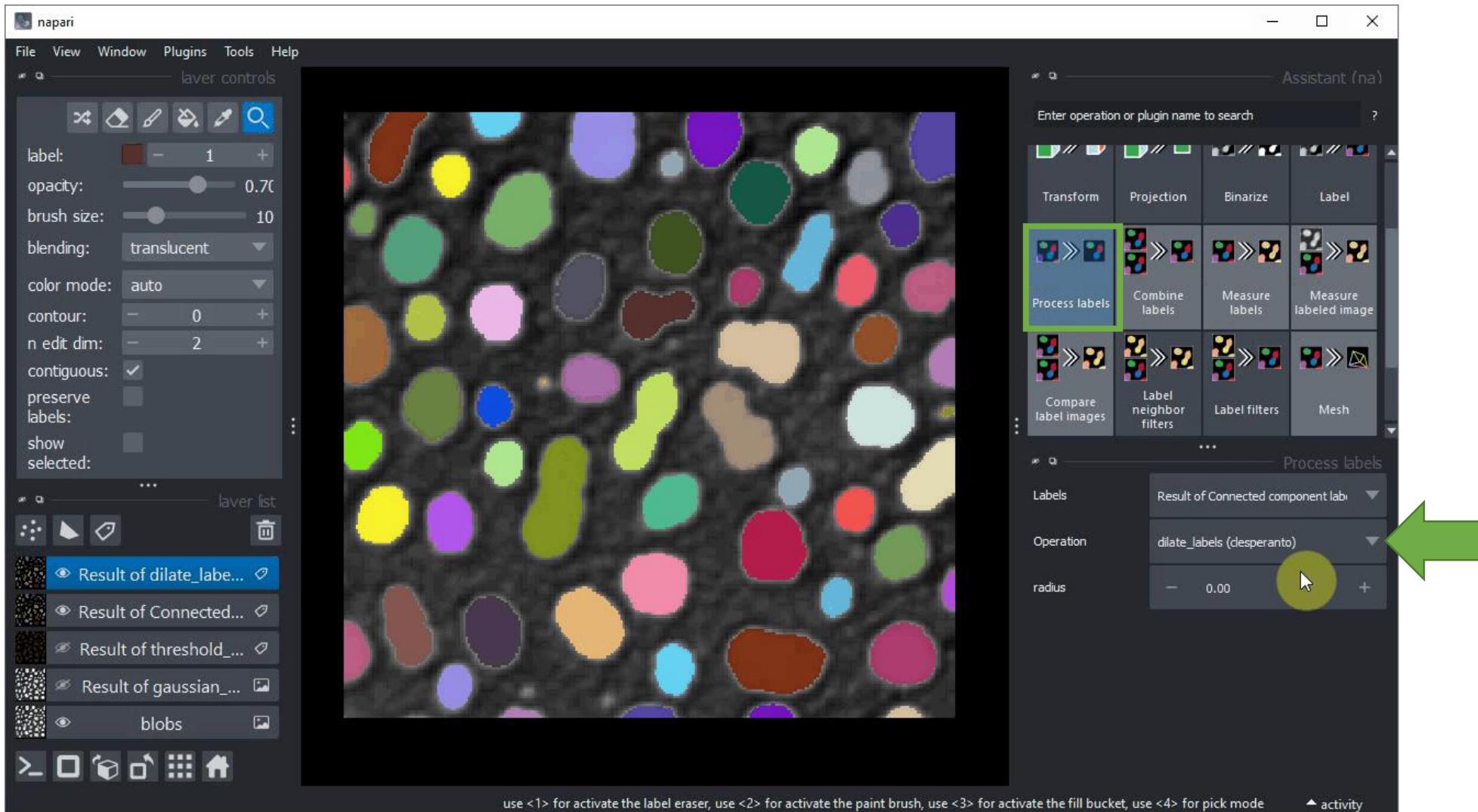
Tools > Segmentation / labeling >
Label touching objects



Tools > Segmentation post-
processing >
Split touching objects
(Similar to ImageJ's Watershed)

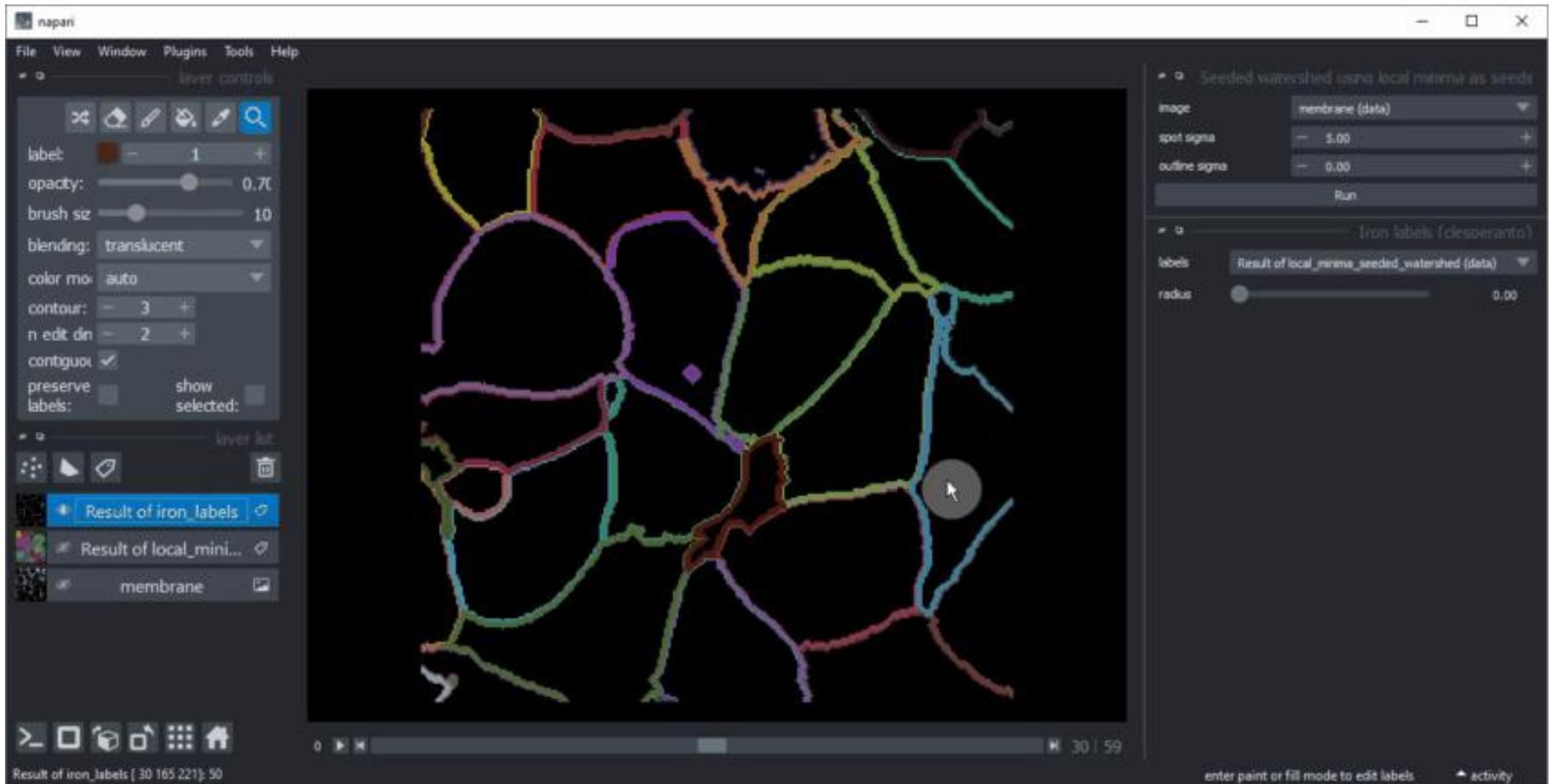


- In Napari Assistant: Process labels



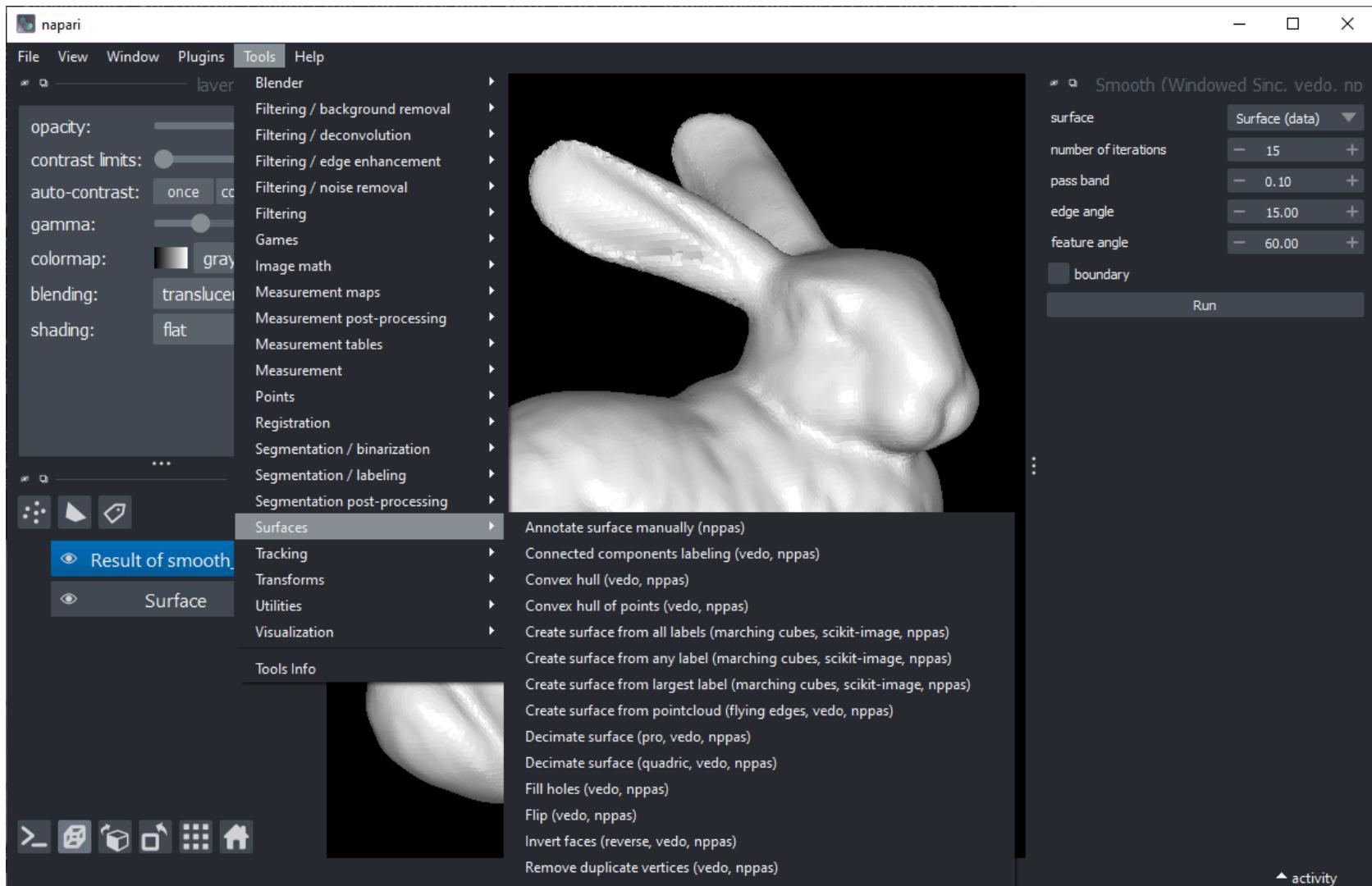
Label post-processing / morphological operations

- In Napari menu Tools > Segmentation post-processing > Smooth labels (clEsperanto)



Surface reconstruction / Processing

- Tools > Surfaces > Create surface ...

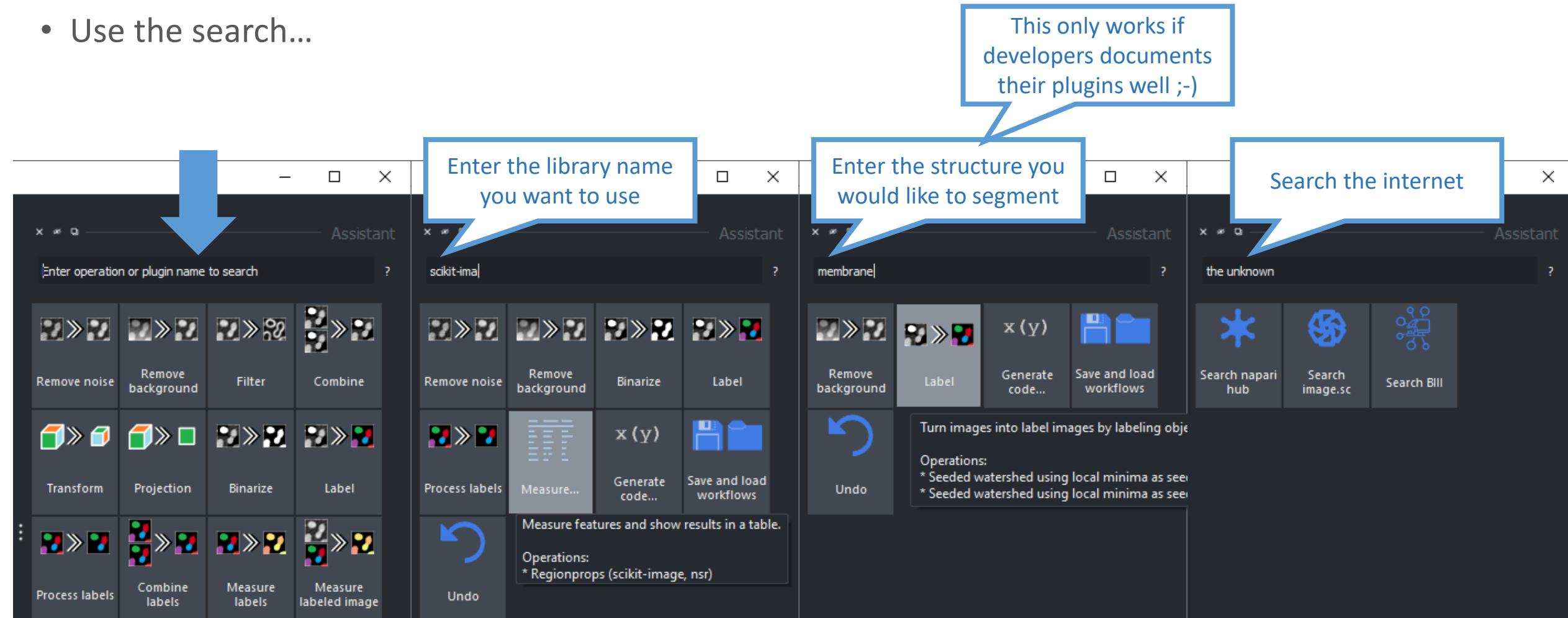


You need to install an extra napari-plugin:

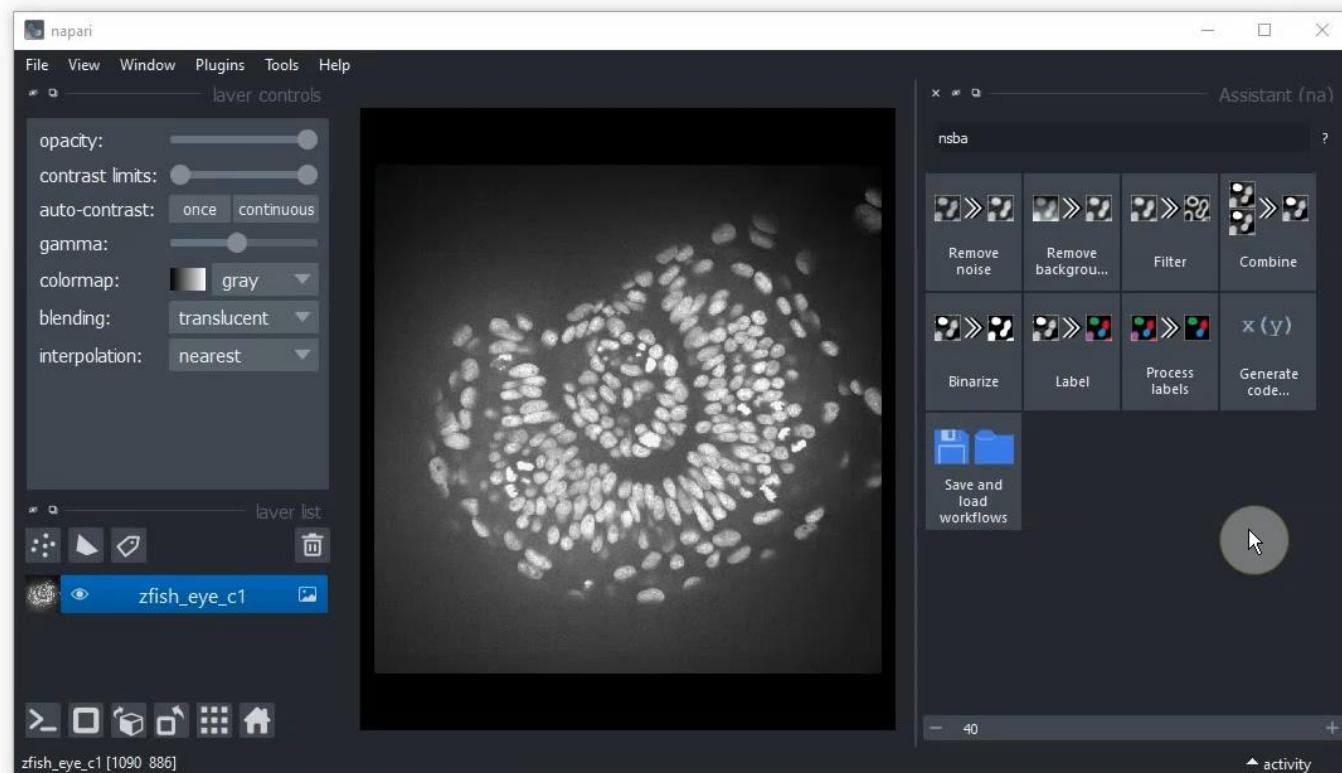
<https://github.com/haesleinhuepf/napari-process-points-and-surfaces>

Browse operations

- Use the search...



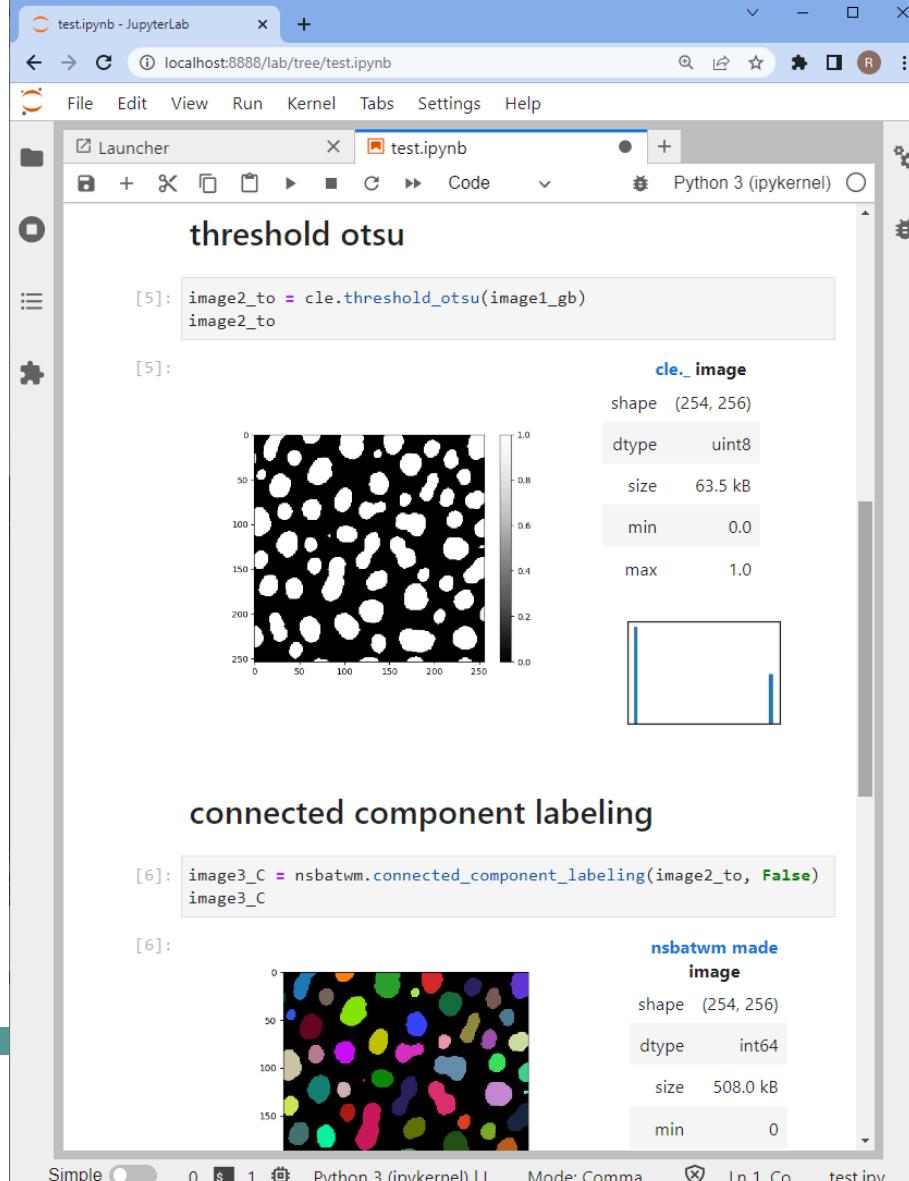
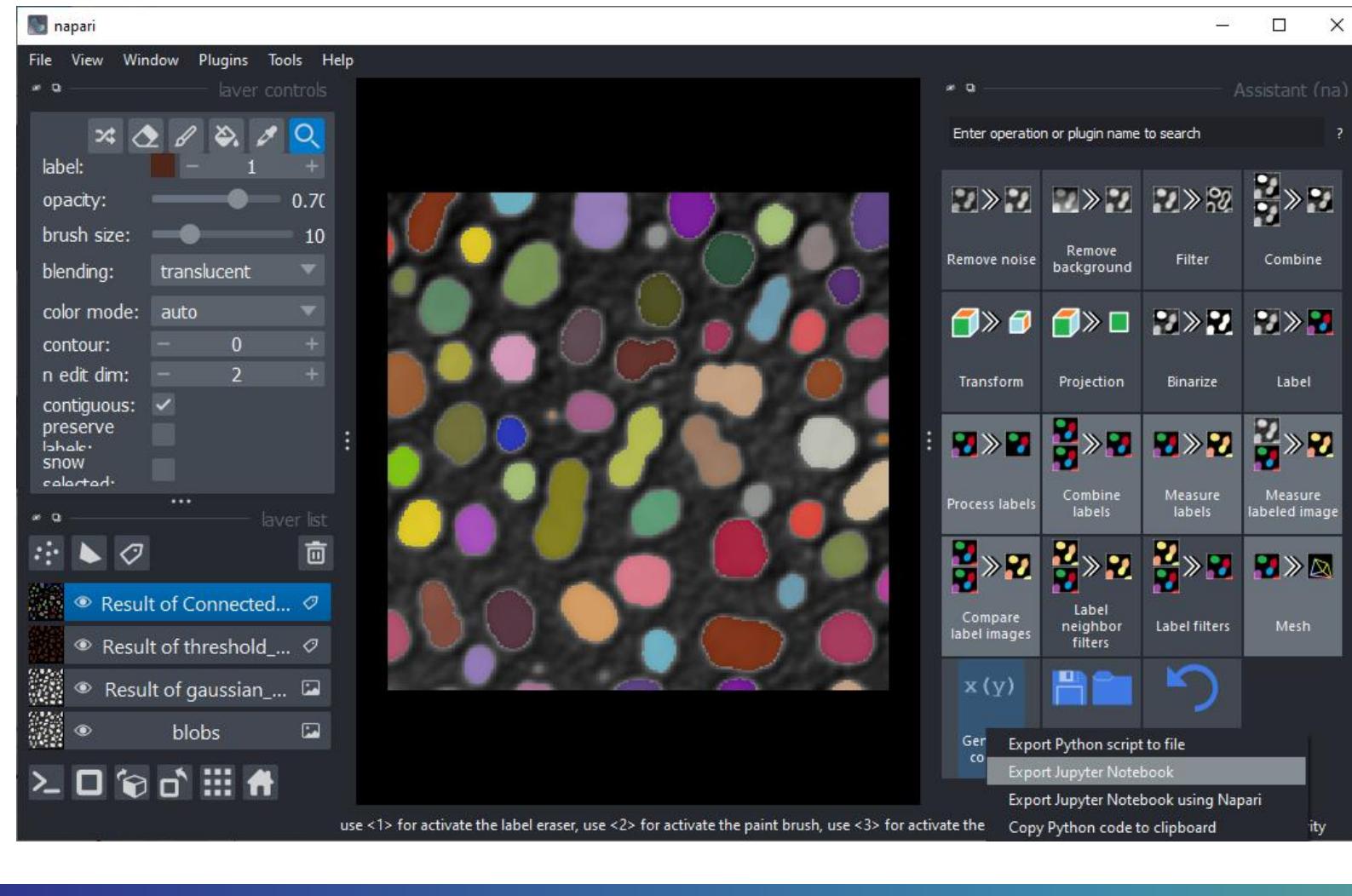
Export code to Jupyter Notebooks



<https://github.com/haesleinhuepf/napari-assistant>

Image data source: Mauricio Rocha Martins, Norden lab, MPI CBG (now at IGC Oeiras)

Export code to Jupyter Notebooks



The screenshot shows a Jupyter Notebook titled 'test.ipynb - JupyterLab'. The code cell [5] contains the command `image2_to = cle.threshold_otsu(image1_gb)`. Below it, the variable `cle._image` is displayed with its properties: shape (254, 256), dtype uint8, size 63.5 kB, min 0.0, and max 1.0. The output cell [5] shows a binary mask image where white blobs are on a black background. The code cell [6] contains `image3_C = nsbatwm.connected_component_labeling(image2_to, False)`. The output cell [6] shows the original segmented image from the napari screenshot. Below the notebook, the variable `nsbatwm.made_image` is shown with its properties: shape (254, 256), dtype int64, size 508.0 kB, and min 0.

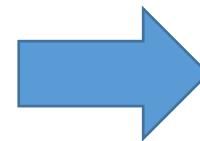
Image segmentation in Python

Robert Haase

April 2023

Voronoi-Otsu-Labeling

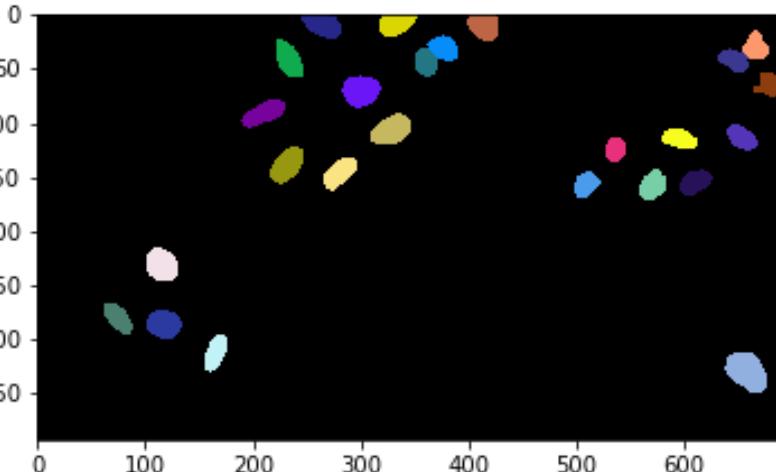
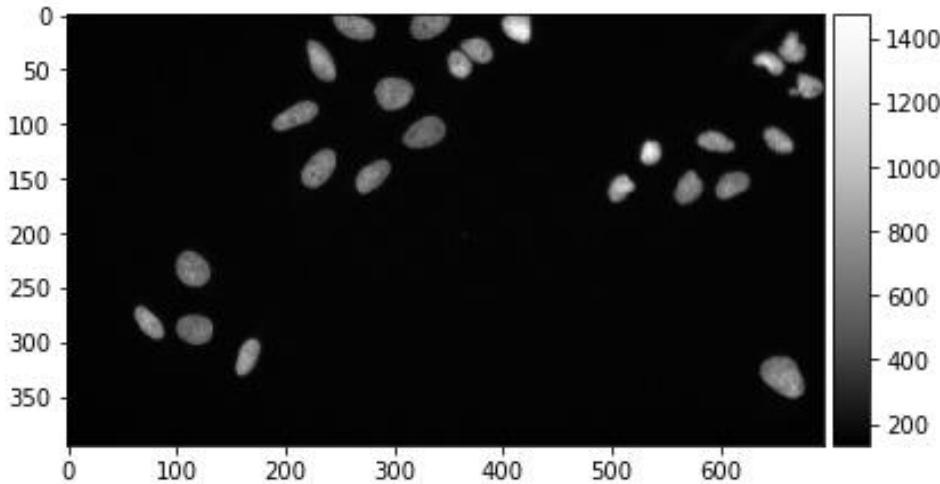
- Gaussian-Blur
- Otsu-Thresholding
- Spot-detection
- Watershed on the binary image



... in a single line of code:

```
segmented = nsbatwm.voronoi_otsu_labeling(input_image,  
                                             spot_sigma=5,  
                                             outline_sigma=1  
)
```

segmented



nsbatwm made image

shape	(395, 695)
dtype	int32
size	1.0 MB
min	0
max	25

- Some [segmentation] algorithms have prerequisites...

```
[1]: import pyclesperanto_prototype as cle
```

```
[ ]: cle.voronoi_otsu_labeling(
```

Docstring:

Labels objects directly from grey-value images.

The two sigma parameters allow tuning the segmentation result. Under the hood, this filter applies two Gaussian blurs, spot detection, Otsu-thresholding [2] and Voronoi-labeling [3]. The thresholded binary image is flooded using the Voronoi tessellation approach starting from the found local maxima.

Notes

* This operation assumes input images are isotropic.

Parameters

source : Image

 Input grey-value image

label_image_destination : Image, optional

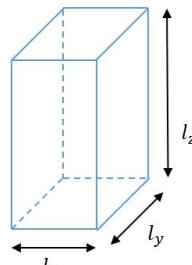
 Output image

spot_sigma : float, optional

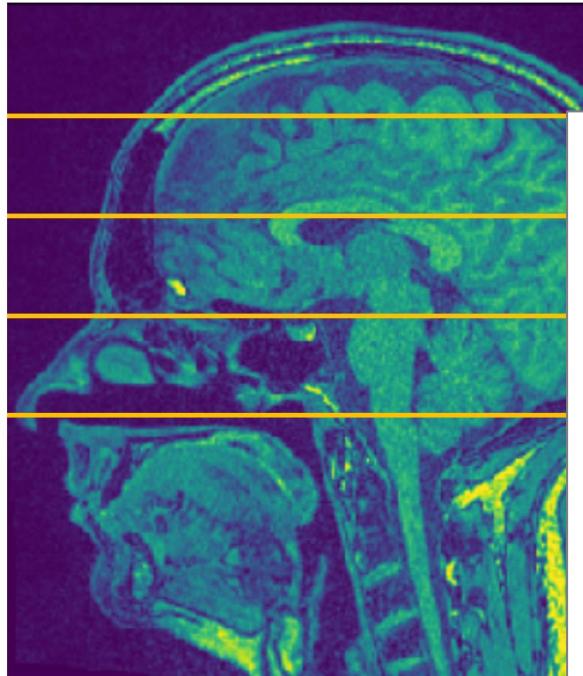
- Reminder: Anisotropic images might be tricky to process properly

Image stacks and voxels

- 3-dimensional images consisting of voxels
- "Image stack"
- Often anisotropic (not equally large in all directions)



$$l_x = l_y \neq l_z$$

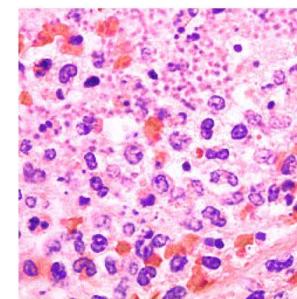


 @haesleinhuepf 



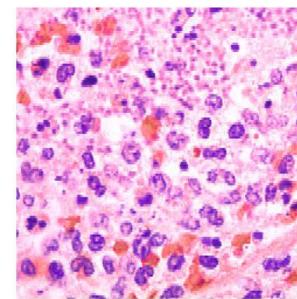
Anisotropy

- Voxel size has immediate impact on image quality and thus, on processing / analysis results.



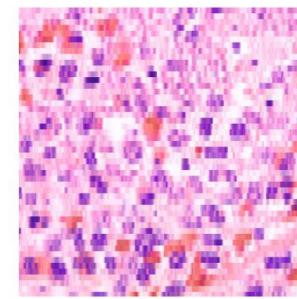
1:1

250 x 250 px



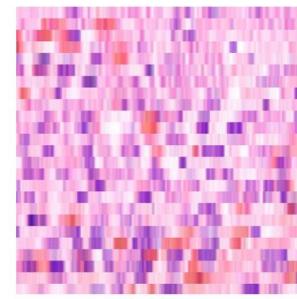
1:2

250 x 125 px



1:5

250 x 50 px



1:10

250 x 25 px

 @haesleinhuepf

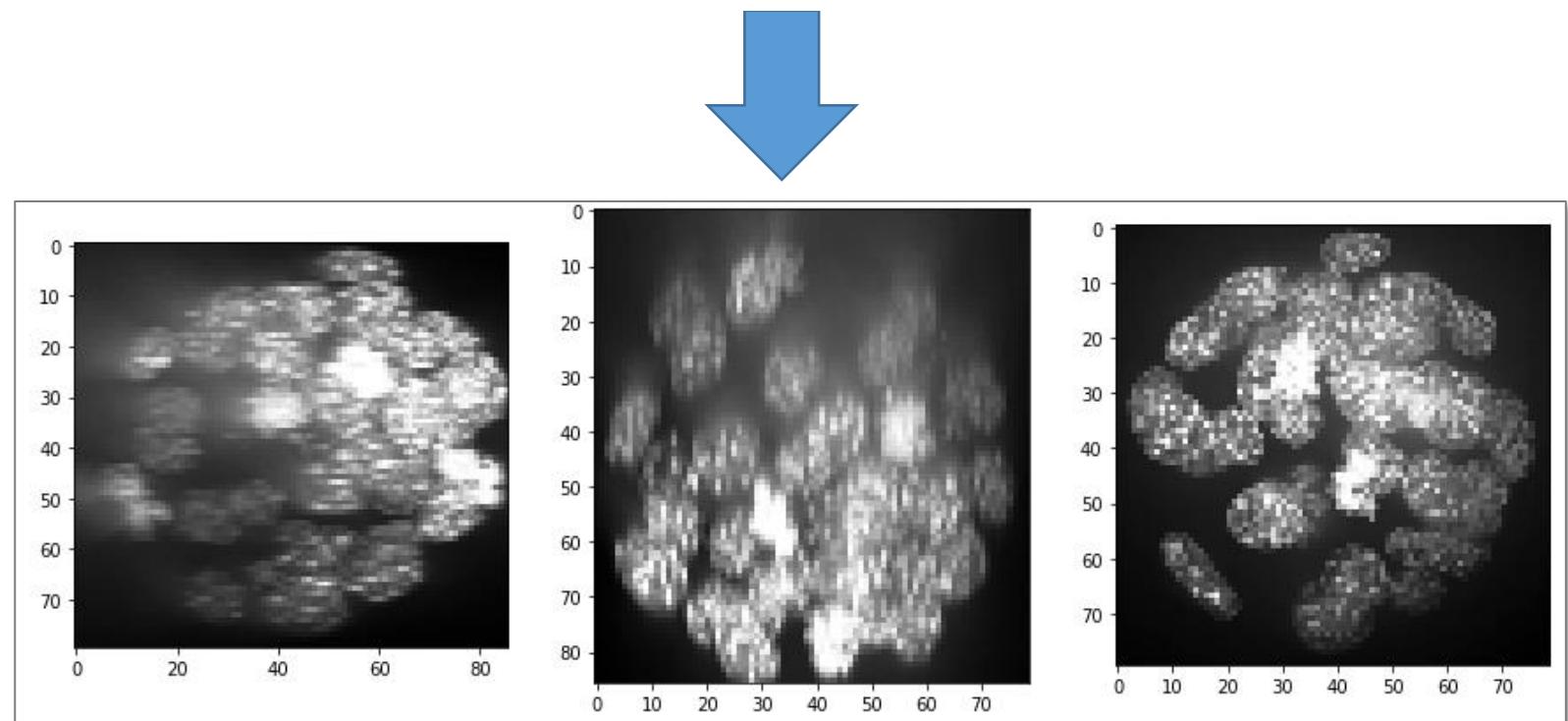
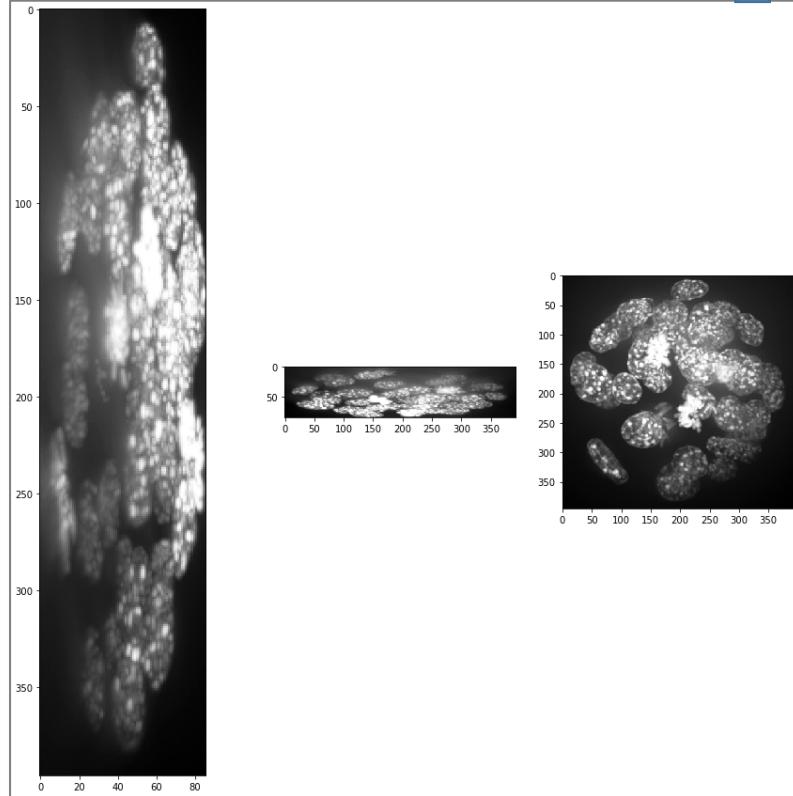
April 2023

Image source: cropped from
https://de.m.wikipedia.org/wiki/Datei:Histo_Lungenpest.jpg

Reslicing / scaling / sampling

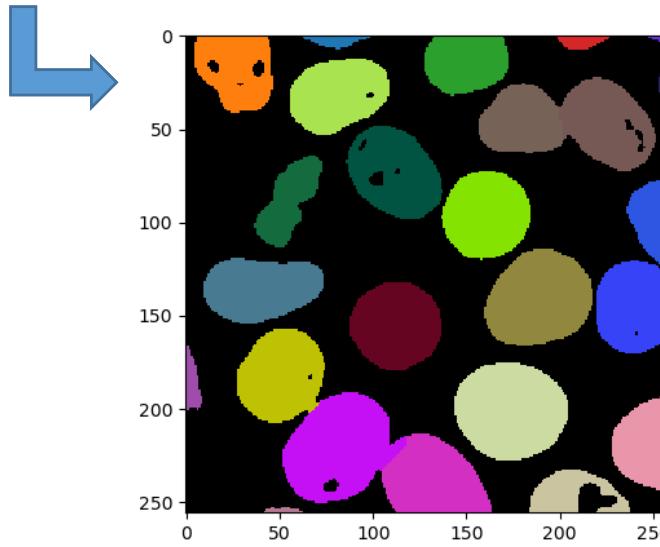
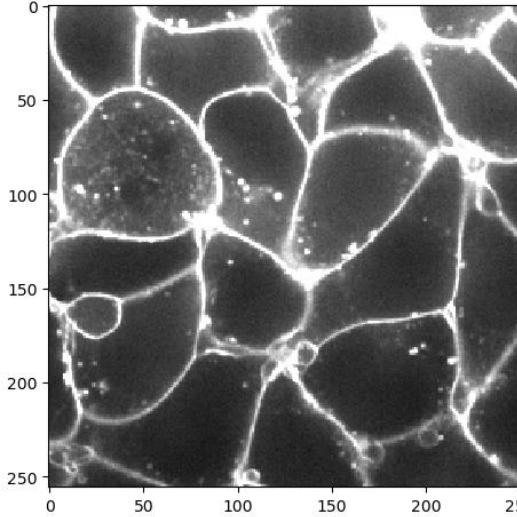
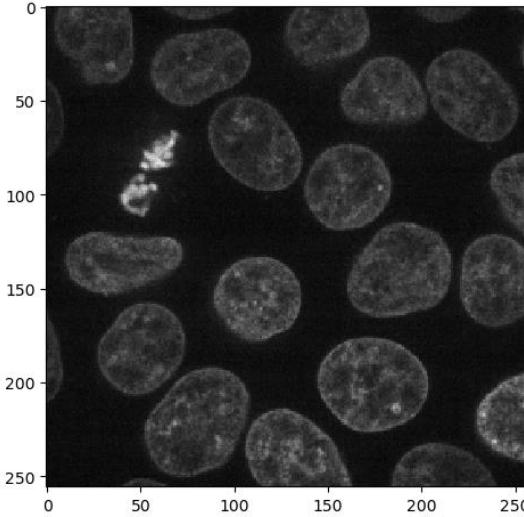
- Resample image data to a specific voxel size

```
resampled = cle.scale(input_image, factor_x=voxel_size_x, factor_y=voxel_size_y, factor_z=voxel_size_z, auto_size=True)  
  
show(resampled)
```

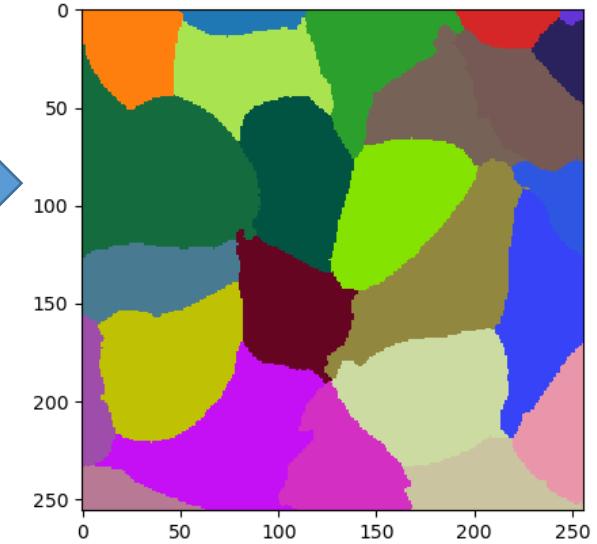


Watershed

- ... in Python practice



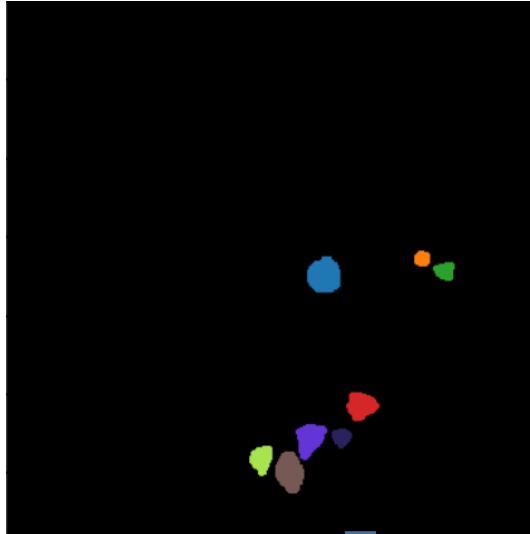
```
labeled_cells = seeded_watershed(membrane_channel, labeled_nuclei)  
labeled_cells
```



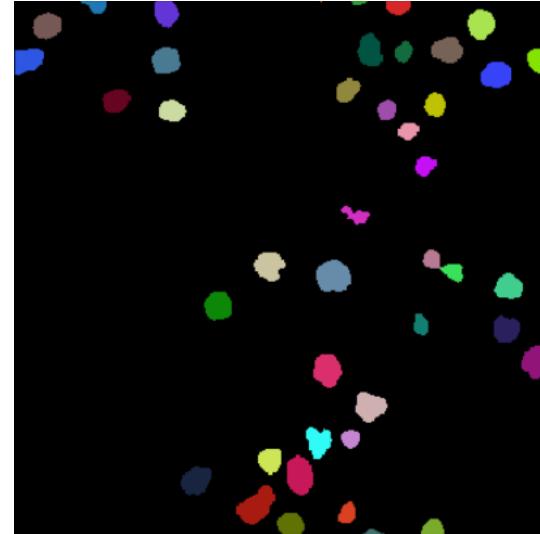
Segmentation quality estimation

- Compare annotations with algorithm results

Sparse instance annotation



Instance segmentation



```
from the_segmentation_game import metrics
```

```
[10]: metrics.jaccard_index_sparse(annotation, labels)
```

```
[10]: 0.8357392602053431
```

<https://github.com/haesleinhuepf/the-segmentation-game#segmentation-algorithm-comparison>

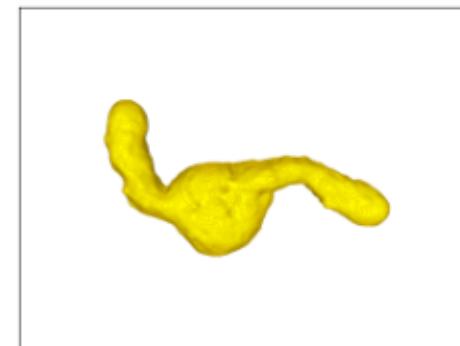
Surface reconstruction

- Turn binary and/or label images into surface meshes



```
surface = nppas.all_labels_to_surface(binary_filled)
```

```
surface
```



nppas.SurfaceTuple

origin (z/y/x) [0. 0. 0.]

center of mass(z/y/x) 57.710,309.963,440.042

scale(z/y/x) 1.000,1.000,1.000

bounds (z/y/x) 12.500...113.500
111.500...461.500
169.500...807.500

average size 170.769

number of vertices 330776

number of faces 661548

Surface mesh processing

- Surface mesh simplification
- To prevent the computer freezing

```
simplified_surface = nppas.decimate_quadric(surface, fraction=0.01)  
simplified_surface
```

nppas.SurfaceTuple

origin (z/y/x) [0. 0. 0.]

center of mass(z/y/x) 57.710,309.963,440.042

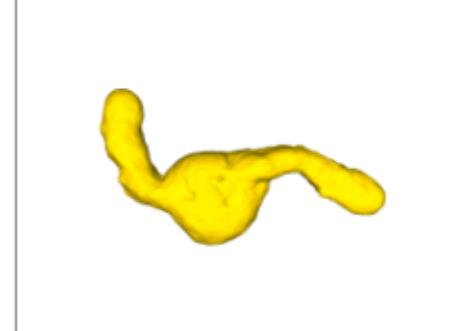
scale(z/y/x) 1.000,1.000,1.000

bounds (z/y/x)
12.500...113.500
111.500...461.500
169.500...807.500

average size 170.769

number of vertices 330776

number of faces 661548



nppas.SurfaceTuple

origin (z/y/x) [0. 0. 0.]

center of mass(z/y/x) 57.928,308.938,440.985

scale(z/y/x) 1.000,1.000,1.000

bounds (z/y/x)
13.231...113.510
111.642...461.602
170.022...806.468

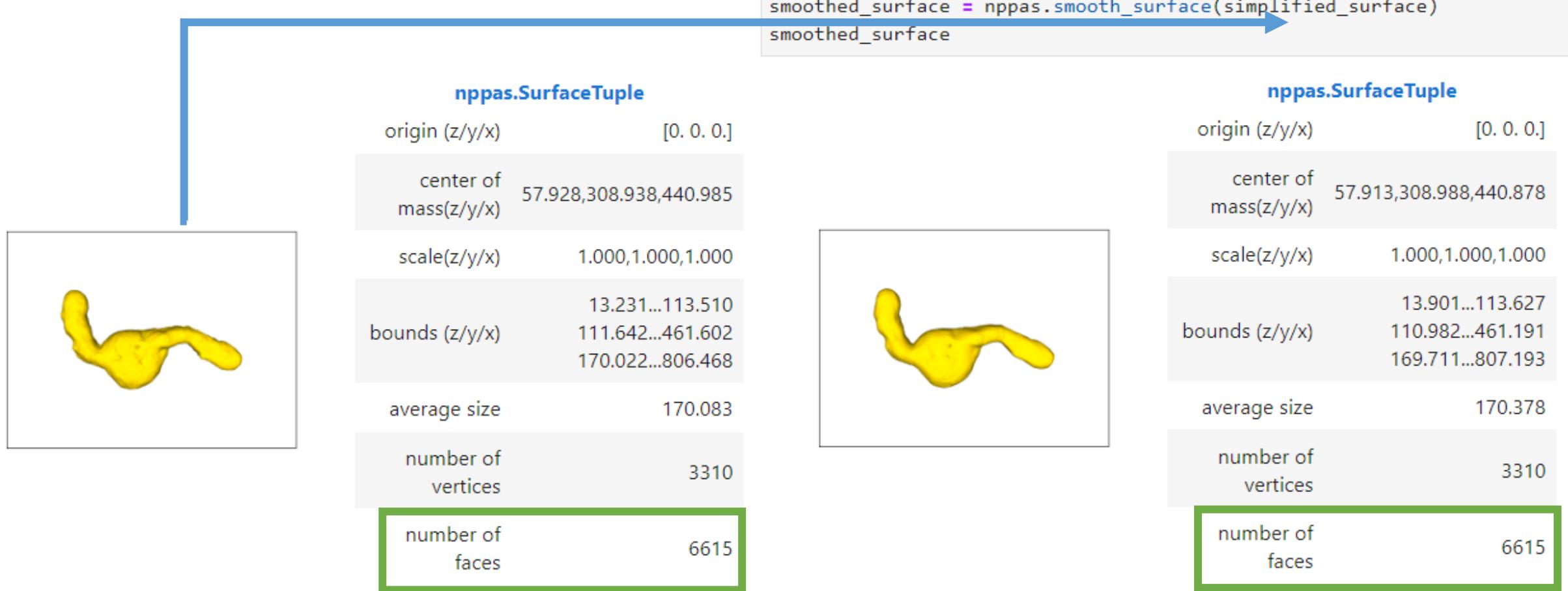
average size 170.083

number of vertices 3310

number of faces 6615

Surface mesh processing

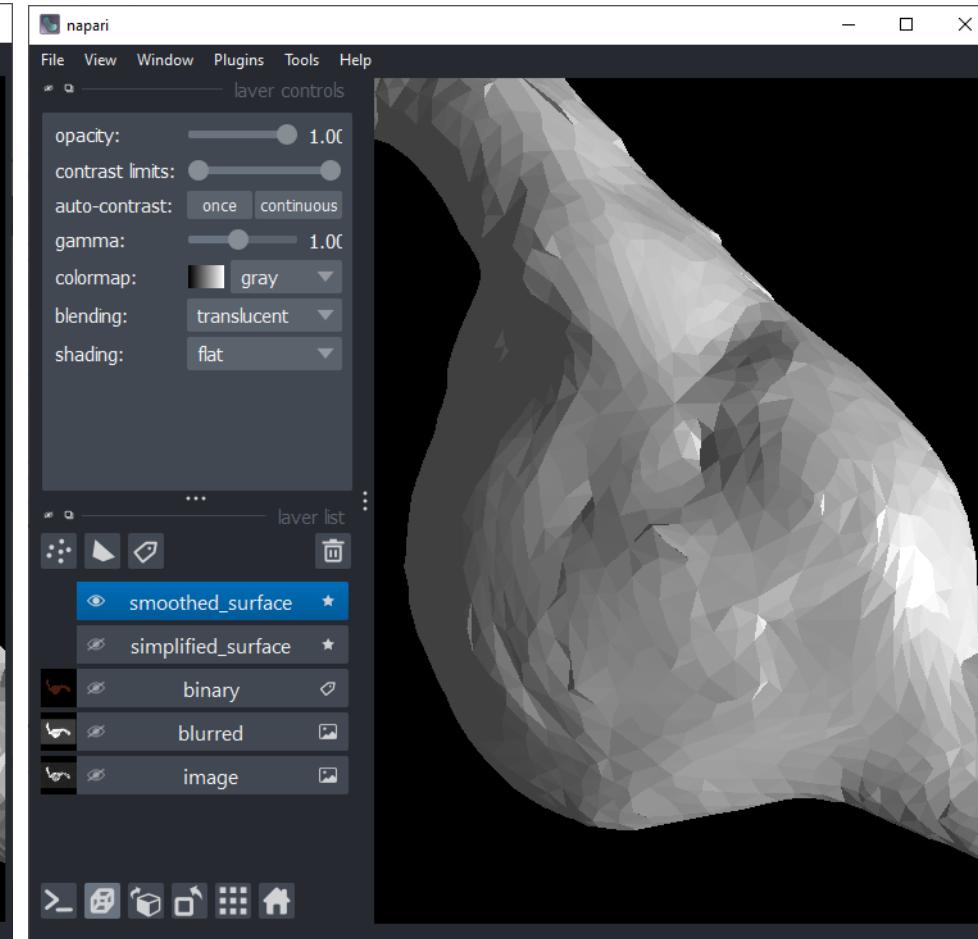
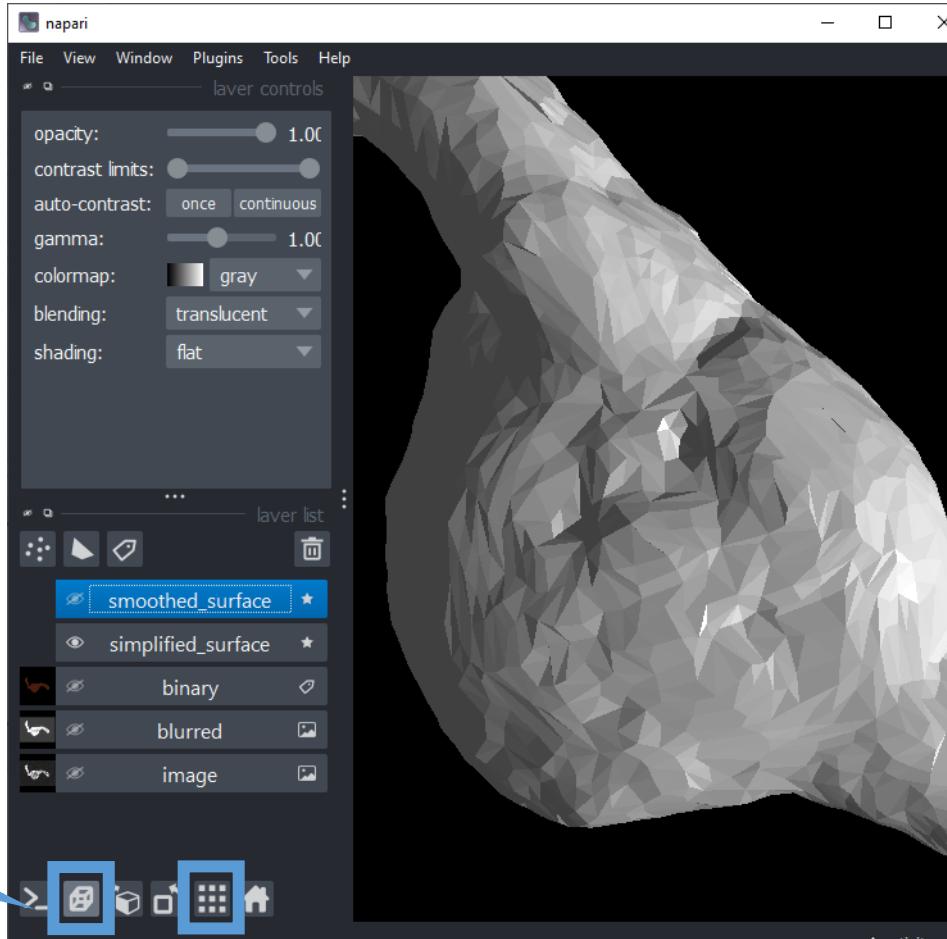
- Surface mesh smoothing



View surface meshes in Napari

```
viewer.add_surface(surface, scale=[zoom, zoom, zoom])  
viewer.add_surface(simplified_surface, scale=[zoom, zoom, zoom])  
viewer.add_surface(smoothed_surface, scale=[zoom, zoom, zoom])
```

In case your computer
freezes, comment out
this line



Exercises

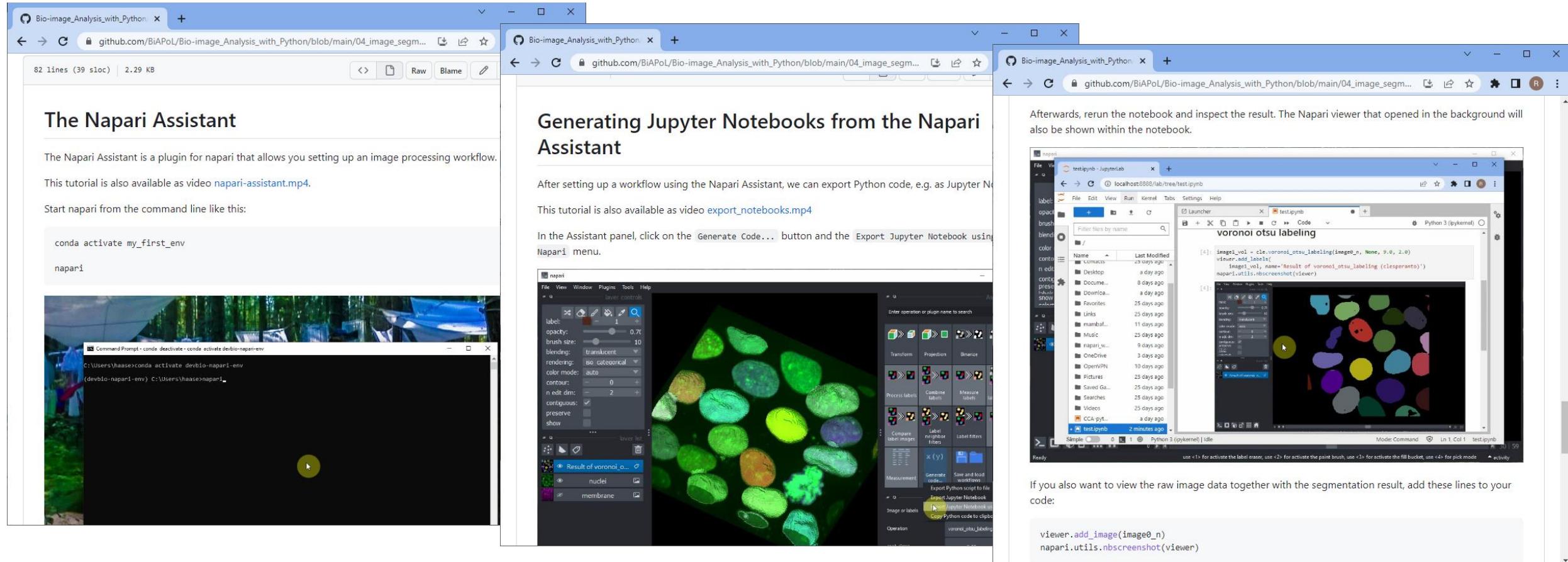
Robert Haase



April 2023

Exercise

- Follow the online instructions
- https://github.com/BiAPoL/Bio-image_Analysis_with_Python/blob/main/04_image_segmentation/06_napari-assistant.md
- https://github.com/BiAPoL/Bio-image_Analysis_with_Python/blob/main/04_image_segmentation/07_notebook_export.md



The Napari Assistant is a plugin for napari that allows you setting up an image processing workflow.

This tutorial is also available as video [napari-assistant.mp4](#).

Start napari from the command line like this:

```
conda activate my_first_env
napari
```

After setting up a workflow using the Napari Assistant, we can export Python code, e.g. as Jupyter Notebook.

This tutorial is also available as video [export_notebooks.mp4](#)

In the Assistant panel, click on the `Generate Code...` button and the `Export Jupyter Notebook using Napari` menu.

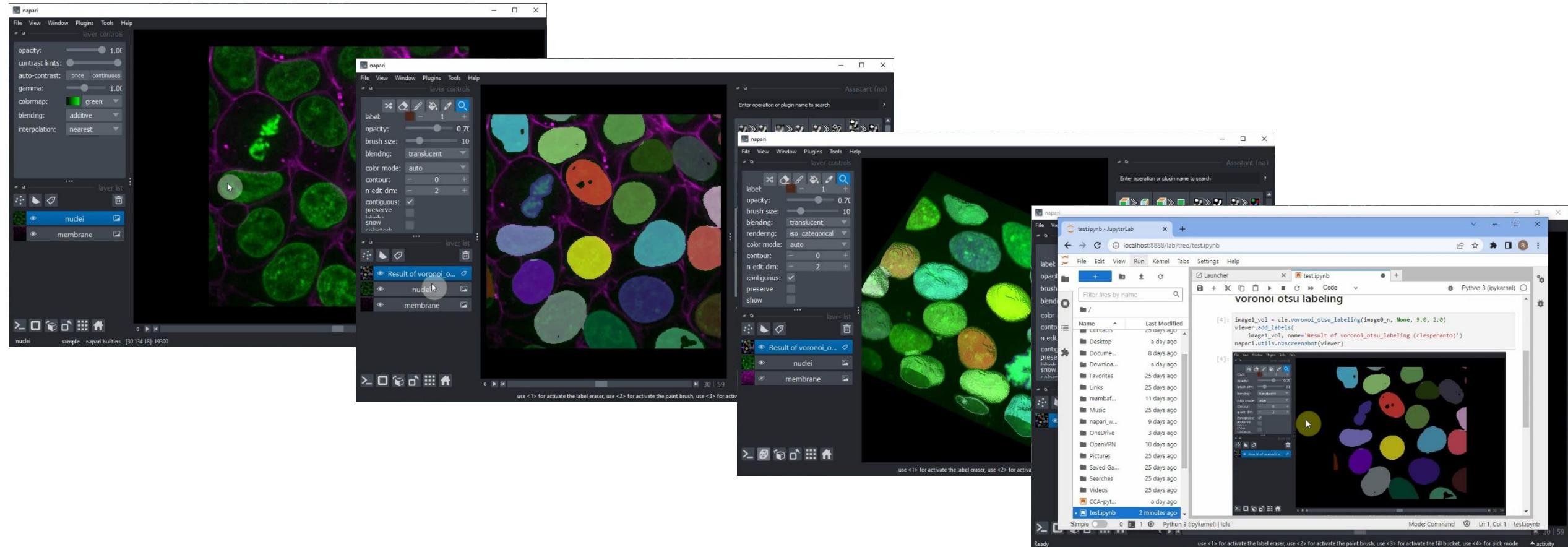
```
viewer.add_image(image0_n)
napari.utils.nbscreenshot(viewer)
```

Afterwards, rerun the notebook and inspect the result. The Napari viewer that opened in the background will also be shown within the notebook.

```
image0_n = cse.voronoi_otsu_labeling(image0_n, None, 9.0, 2.0)
viewer.add_labels(image0_n, name='Result of voronoi_otsu_labeling')
napari.utils.nbscreenshot(viewer)
```

Exercise

- Use the Napari Assistant to generate a Jupyter Notebook



- Measure the quality of a segmentation algorithm applied to a folder of images.
- <https://github.com/BiAPoL/Bio-image Analysis with Python/blob/main/04 image segmentation/17 segmentation quality estimation.ipynb>

```
metrics.jaccard_index_sparse(sparse_labels, labels)
```

```
0.8357392602053431
```

```
for image_filename in os.listdir(image_folder):
    print(image_folder + image_filename)
```

```
.../.../data/BBBC007_batch/17P1_POS0013_D_1UL.tif
.../.../data/BBBC007_batch/20P1_POS0005_D_1UL.tif
.../.../data/BBBC007_batch/20P1_POS0007_D_1UL.tif
.../.../data/BBBC007_batch/20P1_POS0010_D_1UL.tif
.../.../data/BBBC007_batch/A9_p7d.tif
.../.../data/BBBC007_batch/AS_09125_040701150004_A02f00d0.tif
```