Image Visualization

Marcelo Leomil Zoccoler

With material from

Robert Haase, PoL; Mauricio Rocha Martins, Norden lab, MPI CBG; Dominic Waithe, Oxford University; Nuno P Martins, IGC Lisbon; Paulo Aguiar, INEB, Porto; Sebastian Tosi, IRB Barcelona; Benoit Lombardot, Scientific Computing Facility, MPI CBG; Alex Bird, Dan White, MPI CBG

October 2022

@zoccolermarcelo

Image Visualization

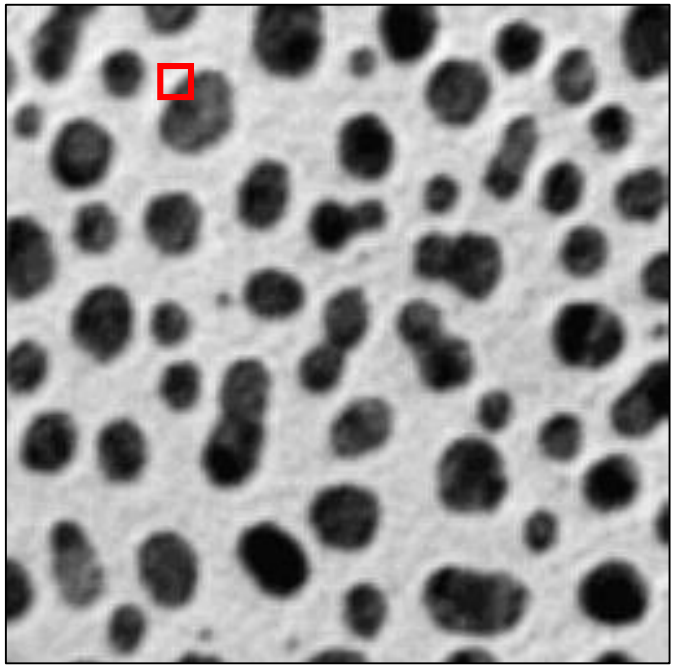
Marcelo L. Zoccoler

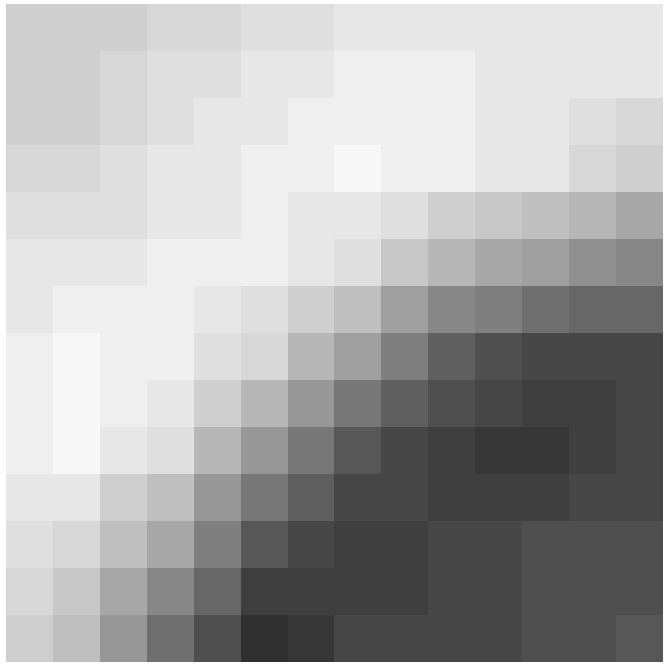
With material from:

Robert Haase, PoL

Images and pixels

- An image is just a matrix of numbers
- Pixel: “picture element”
- The edges between pixels are an artefact. In reality, they don’t exist!





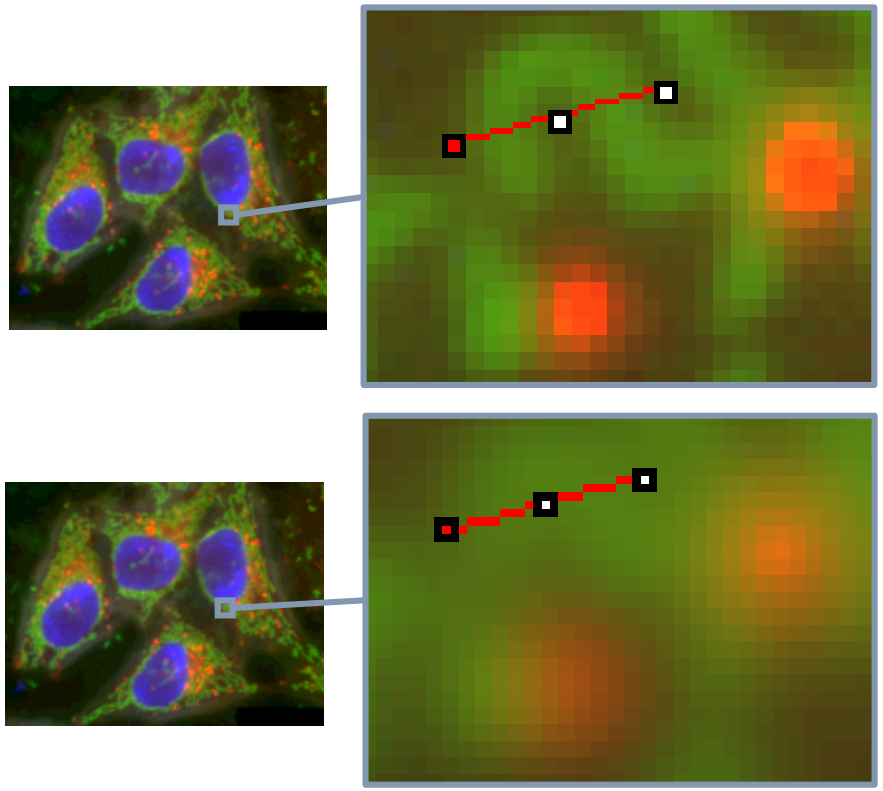
0

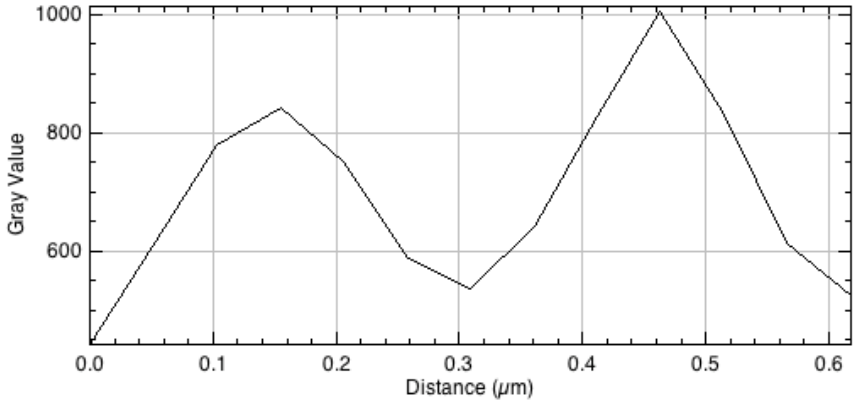
255

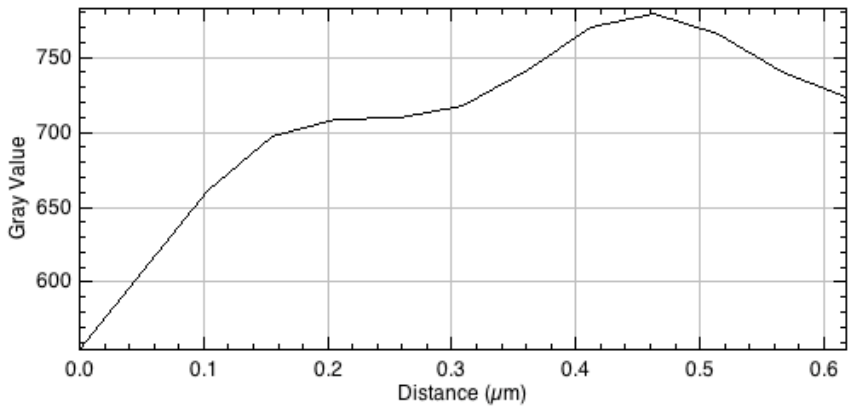
48	48	48	40	40	32	32	24	24	24	24	24	24	24
48	48	40	32	32	24	24	16	16	16	24	24	24	24
48	48	40	32	24	24	16	16	16	16	24	24	32	40
40	40	32	24	24	16	16	8	16	16	24	24	40	48
32	32	32	24	24	16	24	24	32	48	56	64	72	88
24	24	24	16	16	16	24	32	56	72	88	96	112	120
24	16	16	16	24	32	48	64	96	120	128	144	152	152
16	8	16	16	32	40	72	96	128	160	176	184	184	184
16	8	16	24	48	72	104	136	160	176	184	192	192	184
16	8	24	32	72	104	136	168	184	192	200	200	192	184
24	24	48	64	104	136	160	184	184	192	192	192	184	184
32	40	64	88	128	168	184	192	192	184	184	176	176	176
40	56	88	120	152	192	192	192	192	184	184	176	176	176
48	64	104	144	176	208	200	184	184	184	184	176	176	168

Pixel size versus resolution

- Resolution is a property of your imaging system.
- The measure of how close object can be in an image while still being differentiable, is called spatial resolution.





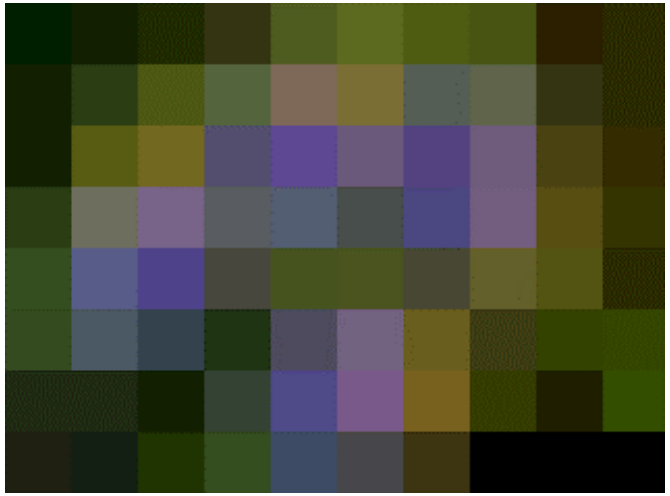


In epi-fluorescence microscopy:

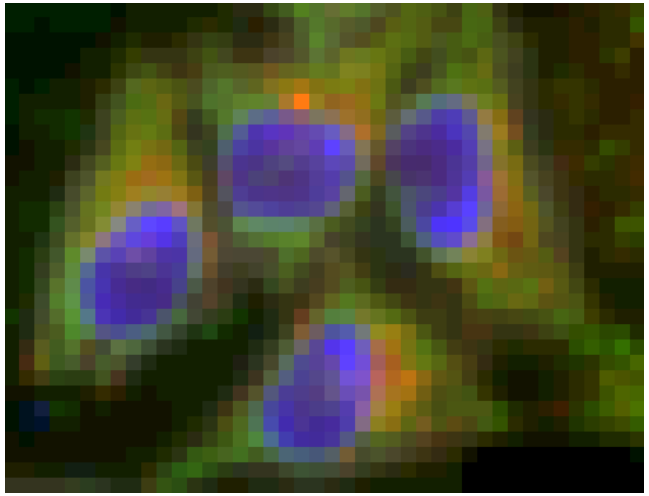
$$d = \frac{0.61\lambda}{NA}$$

Pixel size versus resolution

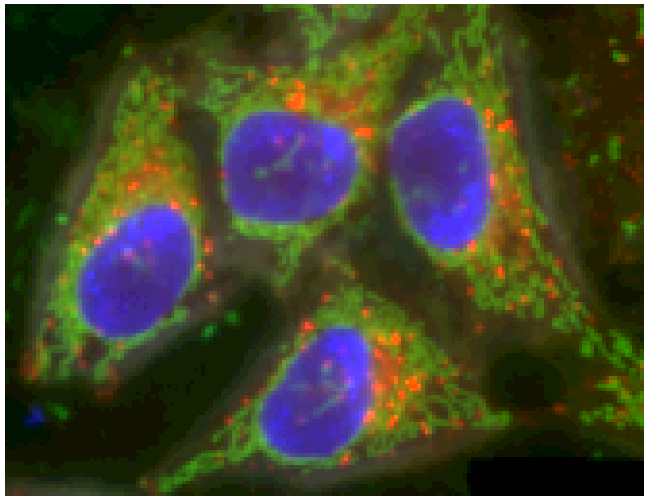
- Pixel size is a digital property of an image.
- You configure it during the imaging session at the microscope.



Pixel size: 3.3 µm



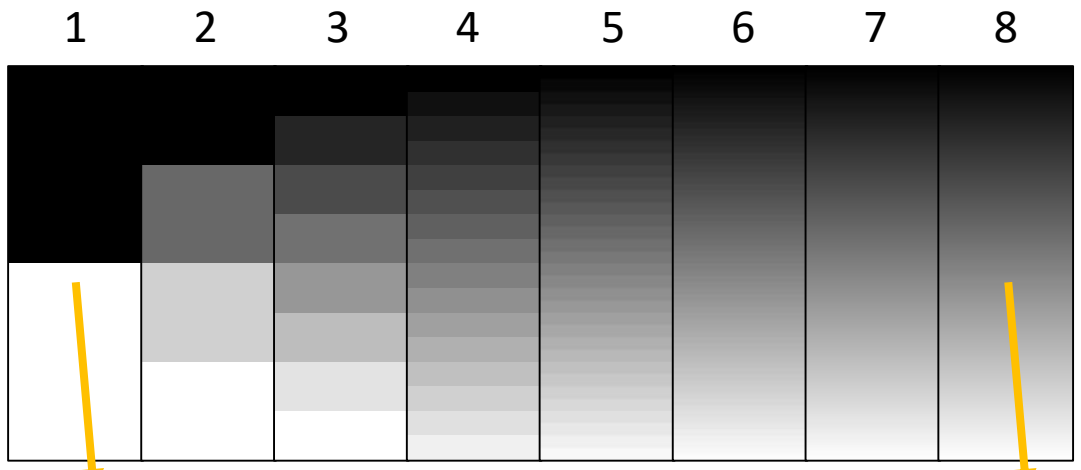
Pixel size: 0.8 µm

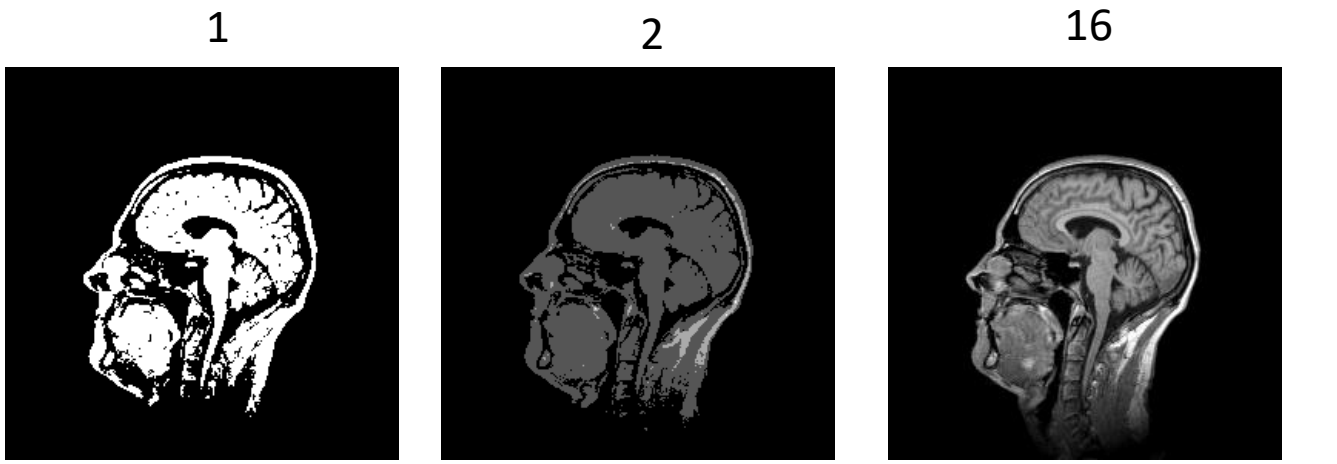


Pixel size: 0.05 µm

Bit-depth

- A bits is the smallest memory unit in computers, *atomic data*.
- The bit-depth *n* enumerates how many different intensity values are present in an image:
  - 2^n grey values
- In microscopy, images are usually stored as 8, 12 or 16-bit images.






```
image.dtype
```

```
dtype('bool')
```

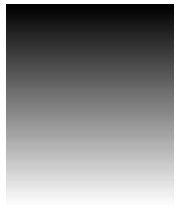
```
image.dtype
```

```
dtype('uint8')
```

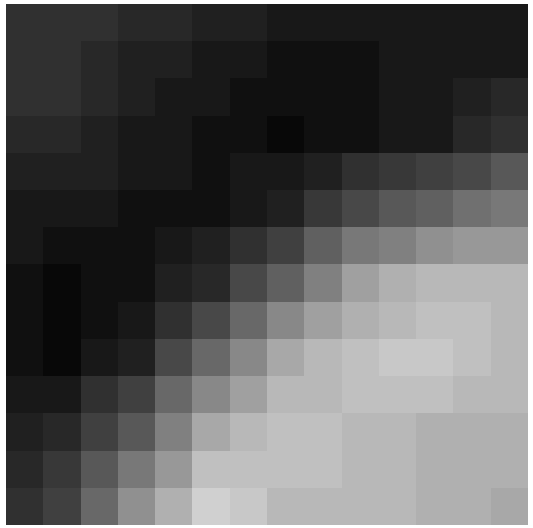
```
image_16bit = image.astype('uint16')
```

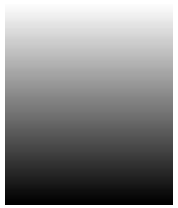
Lookup tables / Colormaps

- The lookup table decides how the image is displayed on screen.
- Applying a different lookup table doesn't change the image. All pixel values stay the same, they just appear differently

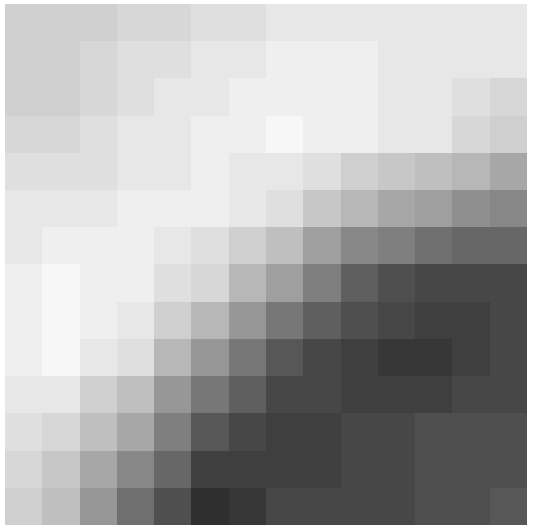
Pixel value	Display color
0 1 2 ... 255	

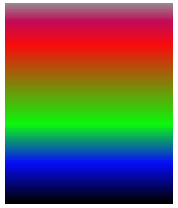
```
plt.imshow(image, cmap='gray')
```



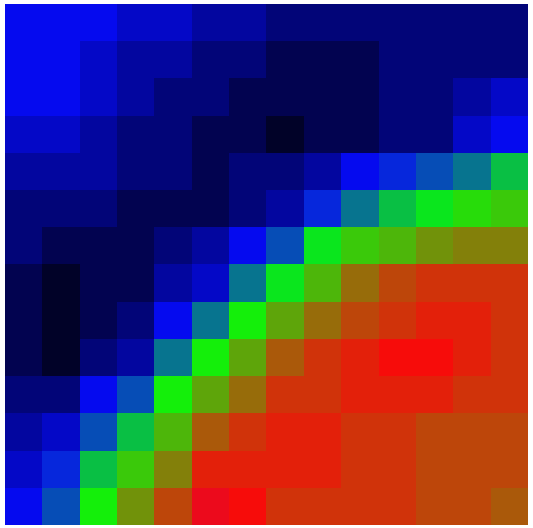
Pixel value	Display color
0 1 2 ... 255	

```
plt.imshow(image, cmap='gray_r')
```



Pixel value	Display color
0 1 2 ... 255	

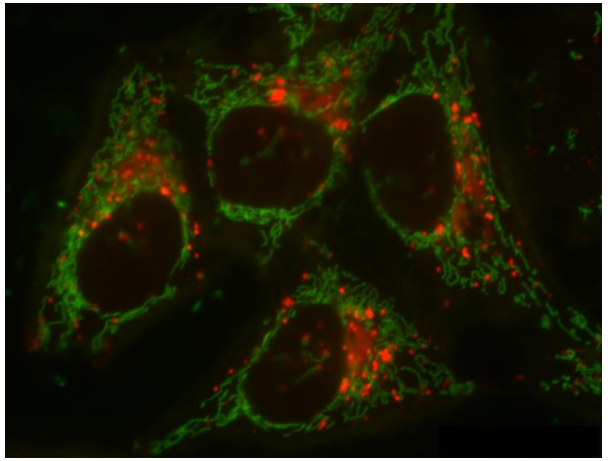
```
plt.imshow(image, cmap='jet')
```



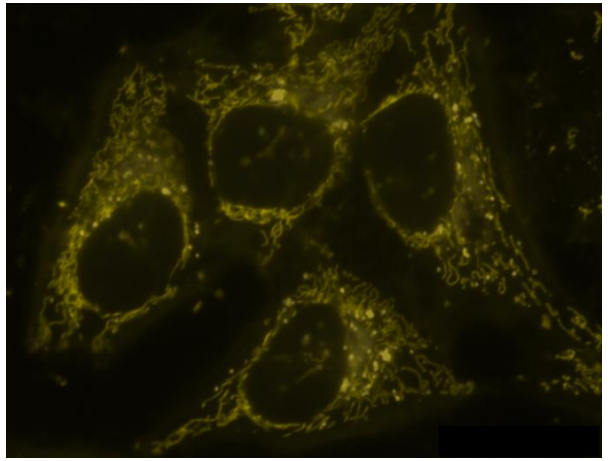
Lookup tables

- Choose visualization of your color tables wisely!
- Think of people with red/green blindness!

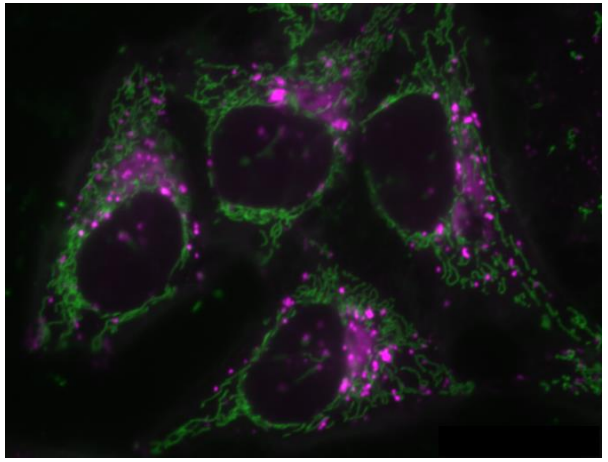
Default view

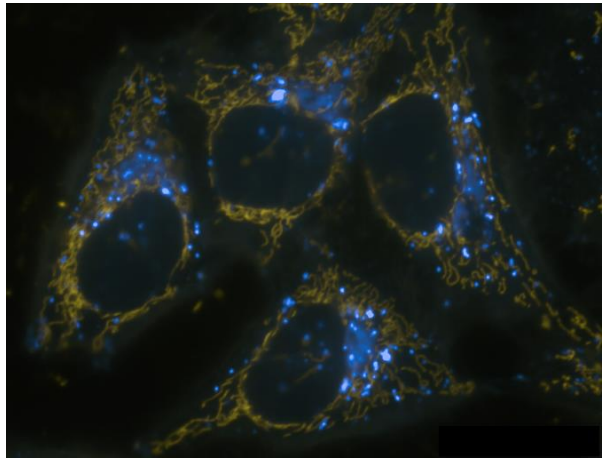


Red/green blind people see it like this



Replace red with magenta!

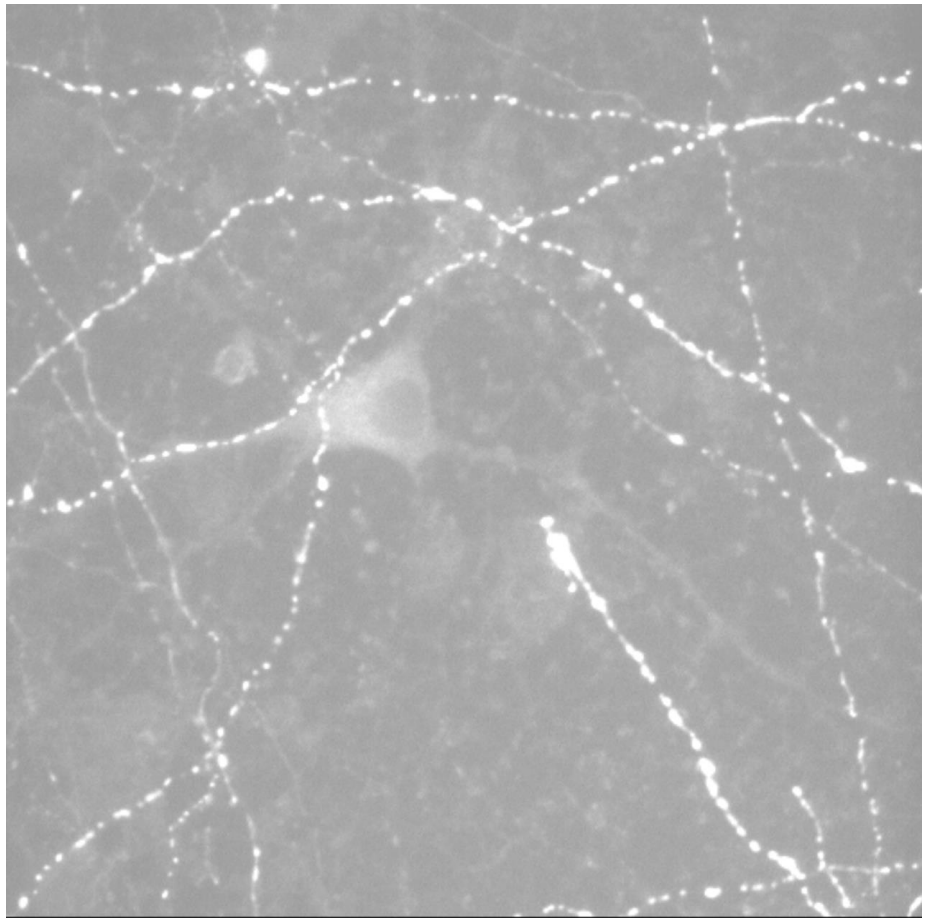




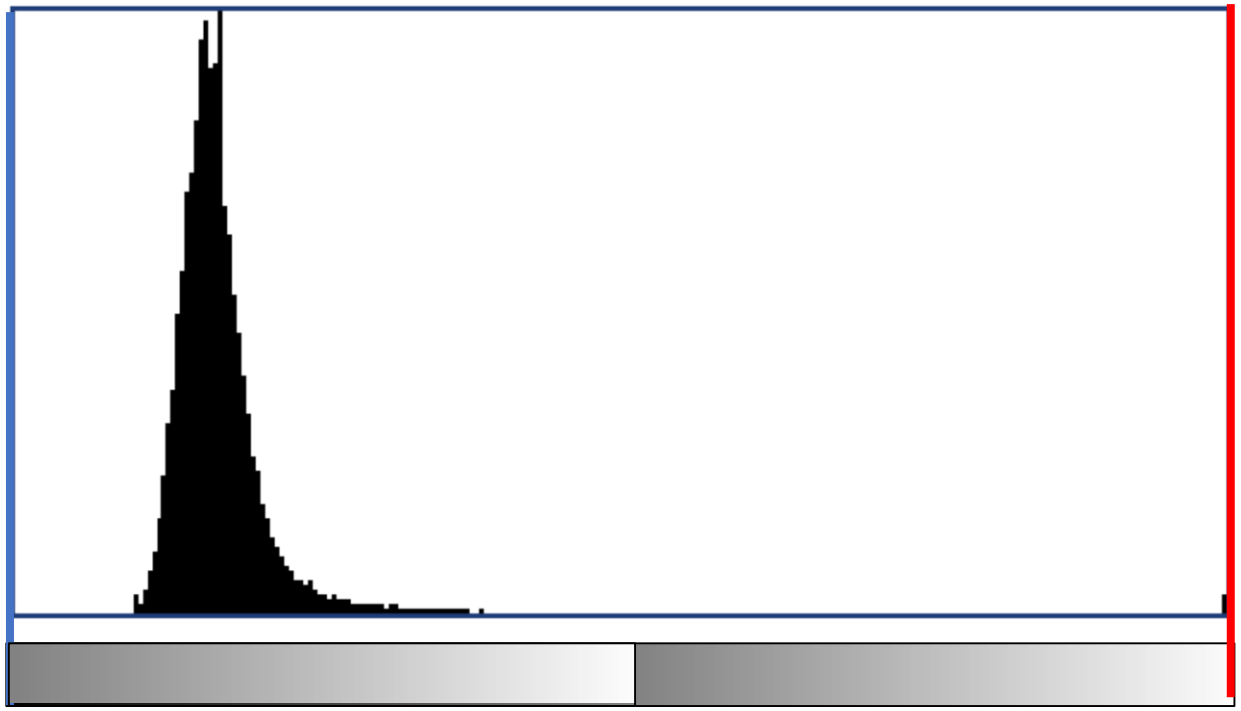
Visualisation: brightness / contrast

Brightness: a relative attribute that represents how much light a source is emitting.
Contrast: refers to the amount of color or grayscale differentiation that exists between various image features.

- Is this image bright?
- Does it have good contrast?



Decrease vmin and vmax:
Brightness Increased

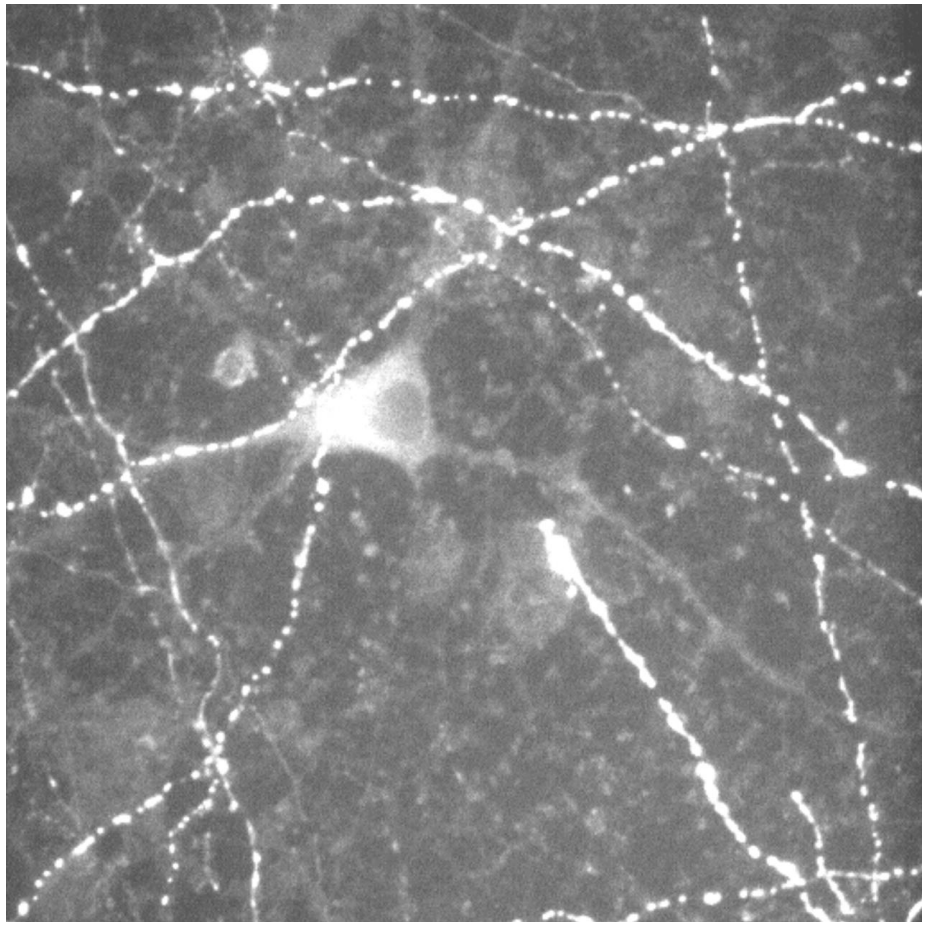


```
plt.imshow(image, cmap='gray', vmin=0, vmax=255)
```

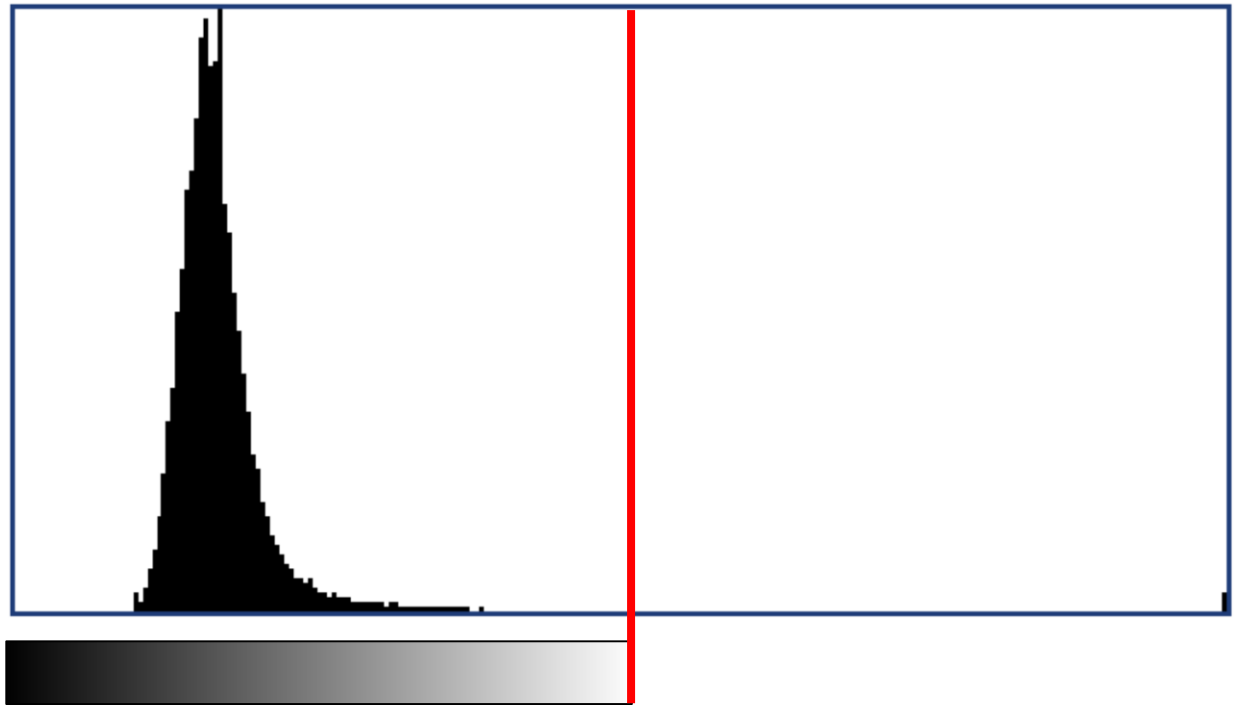
Visualisation: brightness / contrast

Brightness: a relative attribute that represents how much light a source is emitting.
Contrast: refers to the amount of color or grayscale differentiation that exists between various image features.

- Is this image bright?
- Does it have god contrast?



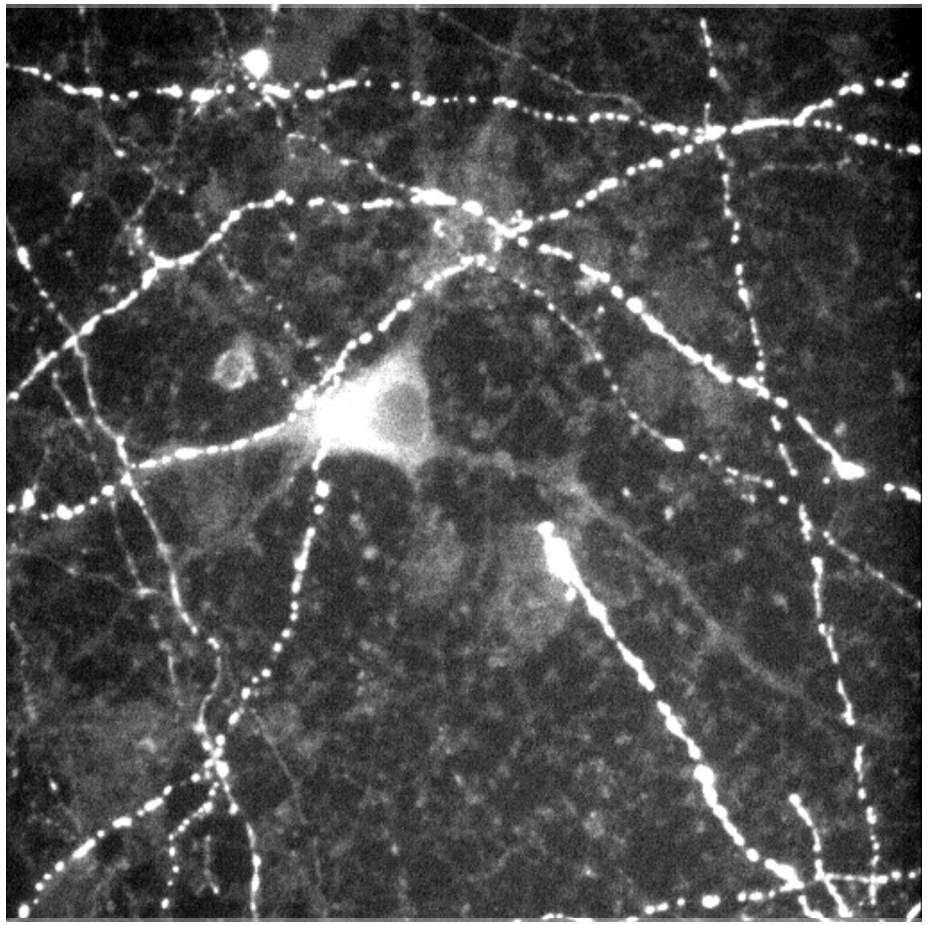
Decrease only vmax:
Contrast Increased



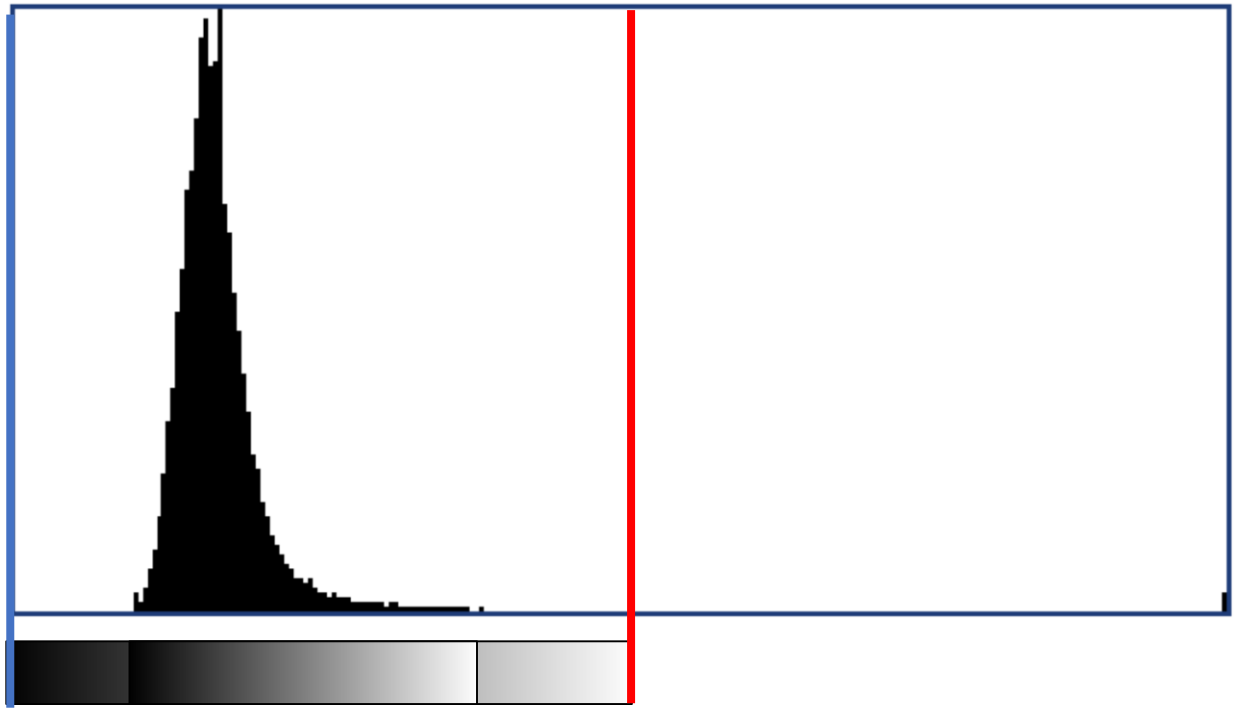
Visualisation: brightness / contrast

Brightness: a relative attribute that represents how much light a source is emitting.
Contrast: refers to the amount of color or grayscale differentiation that exists between various image features.

- Is this image bright?
- Does it have god contrast?



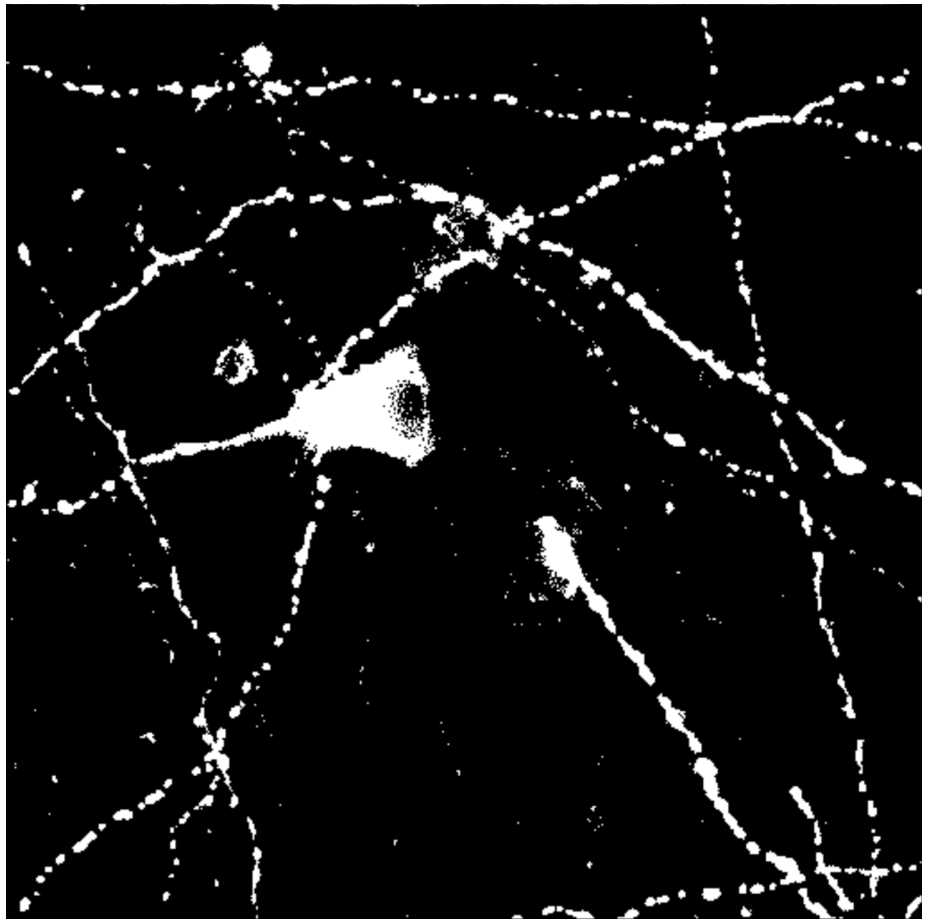
Window vmin and vmax around histogram:
Contrast Increased

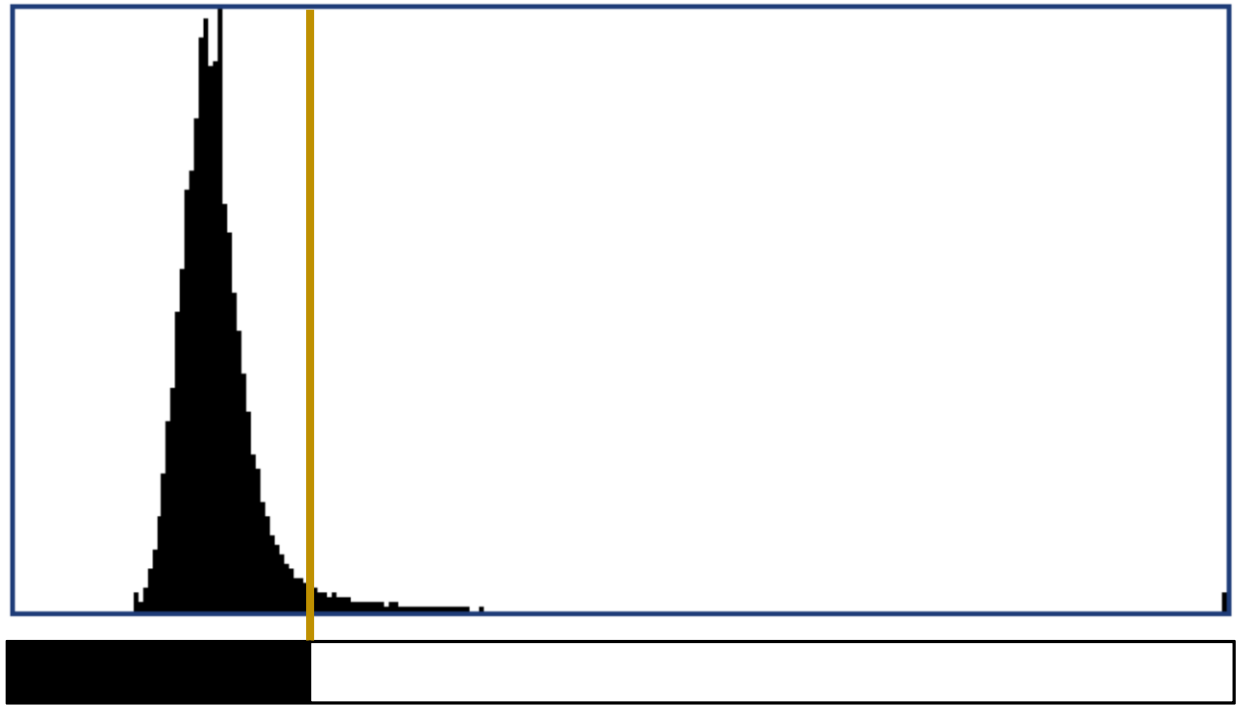


Don’t do such a narrow window right before acquisition!
If your signal changes over time/sample, you will lose information.

Binarization: a quick peek

Values above a threshold value are replaced by 1 (or True).
Values below or equal threshold are replaced by 0 (or False)





```
image_binary = image > value
```

Histogram

Consider the following image:

60	60	60	60	64	192	192
60	60	60	60	64	192	192
64	64	64	64	64	192	192

How many elements does it have?

Which pixel value occurs more often?

If we may be mistaken in such a small image, imagine for big images!

The histogram answers these questions at a glance!

Histogram: a graph that shows how frequent values are in an array/image

```
from skimage.exposure import histogram

piexl_counts, pixel_values = histogram(array)

plt.bar(pixel_values, piexl_counts)
```

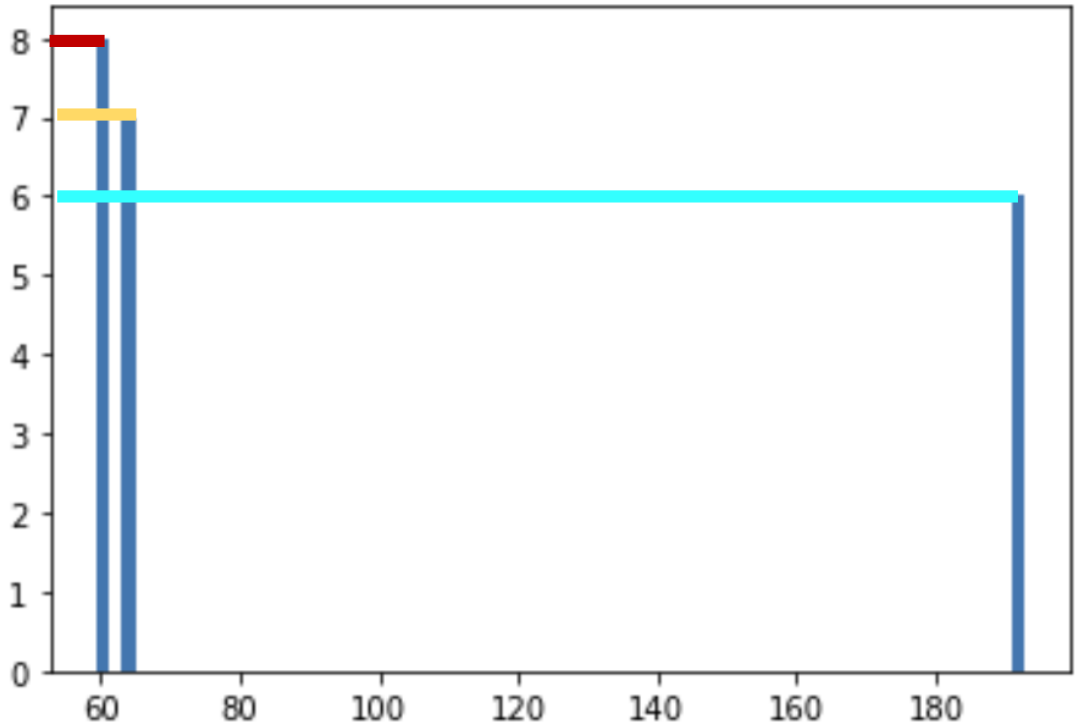
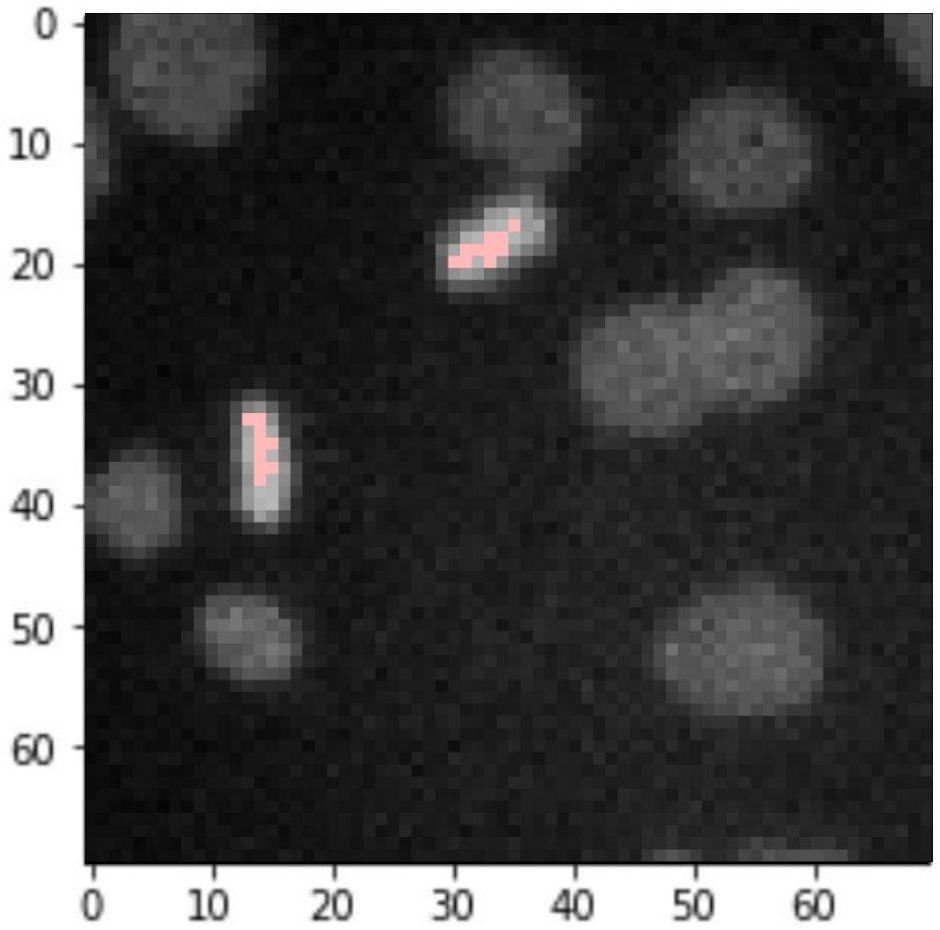
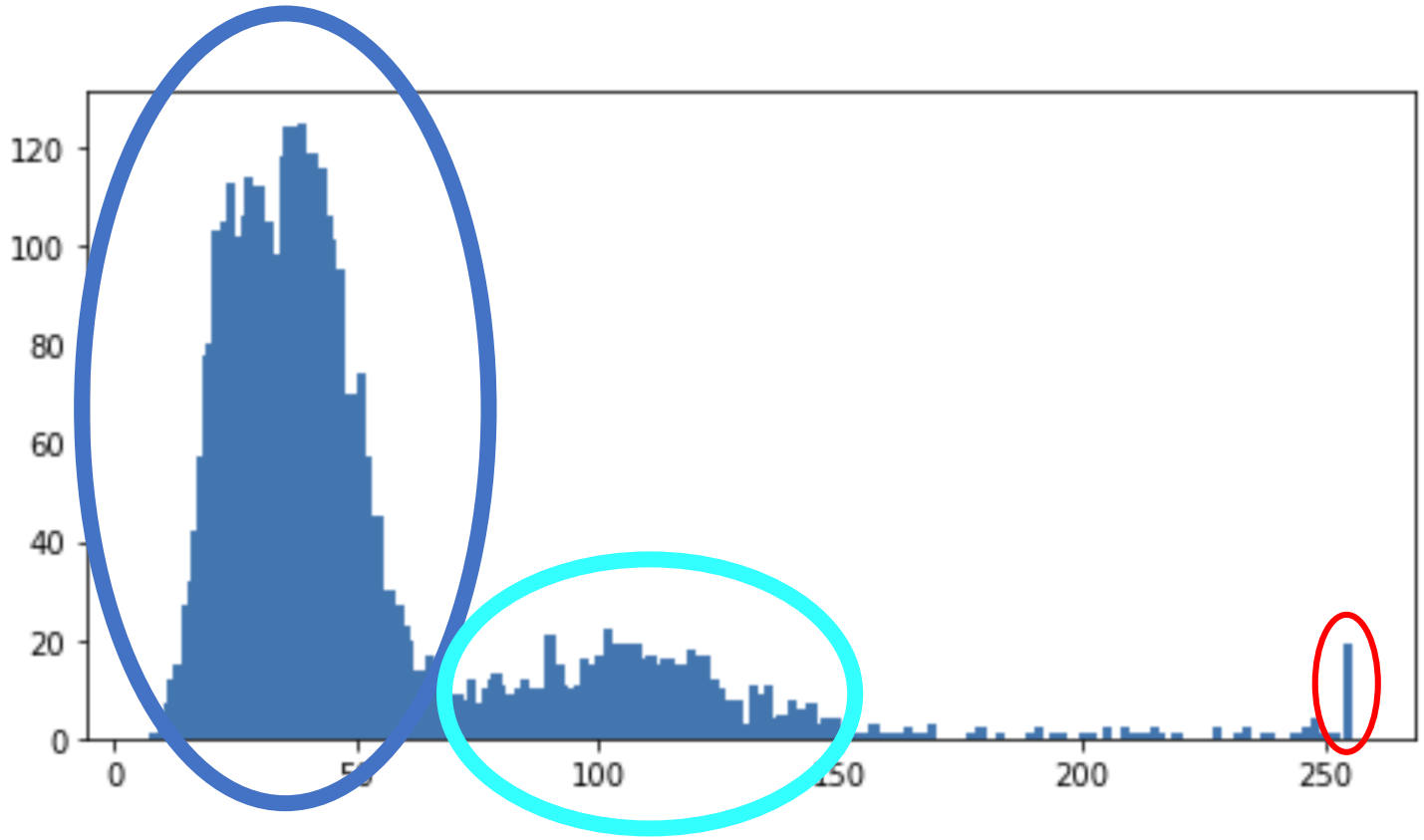


Image Histogram

Example with a slightly larger image:





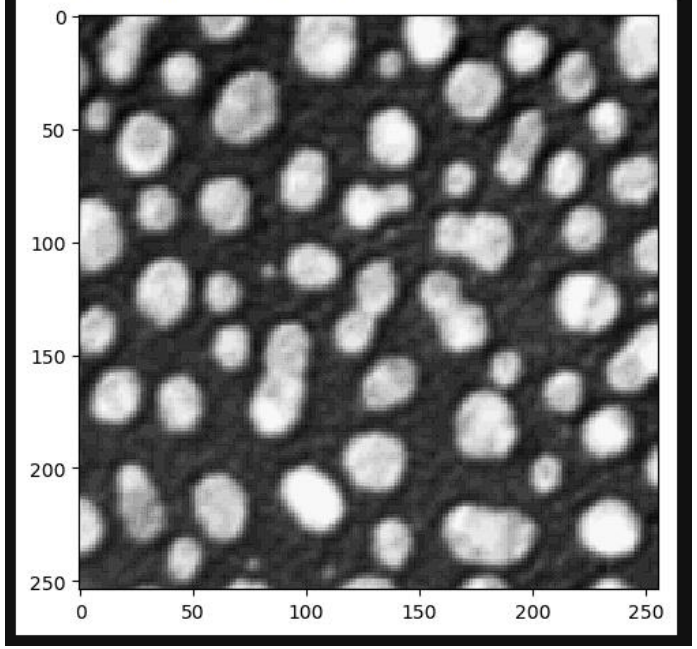
Plotting images in Jupyter notebooks

Many Python libraries usually have a “imshow” function to display images.

scikit-image

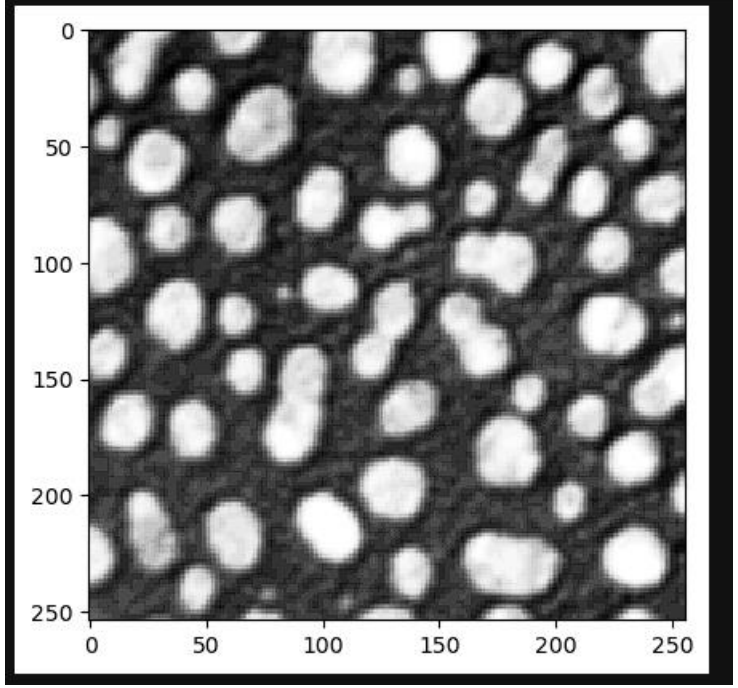
```
from skimage.io import imshow

imshow(image)
```

```
<matplotlib.image.AxesImage at 0x2d0adef0e50>
```



pyclesperanto

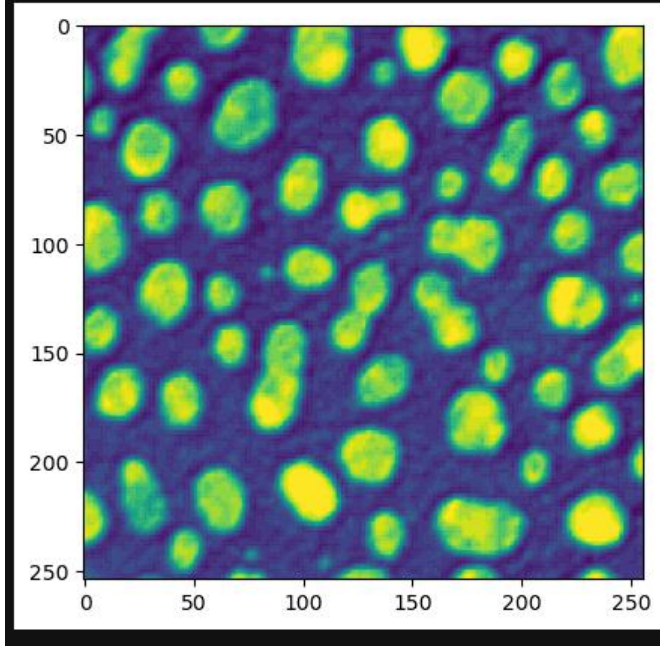
```
import pyclesperanto_prototype as cle

cle.imshow(image)
```



matplotlib

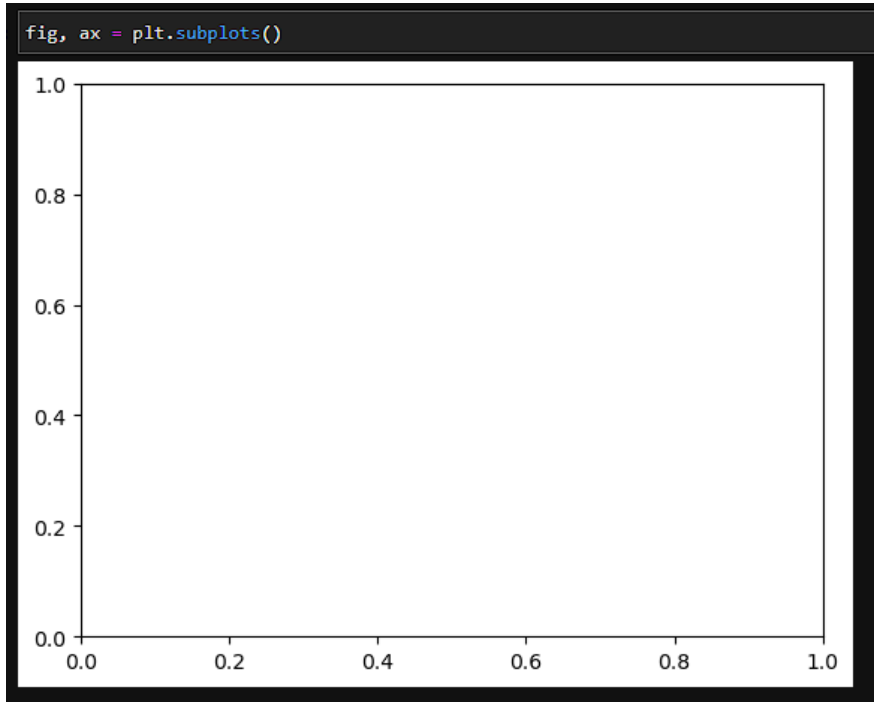
```
import matplotlib.pyplot as plt

plt.imshow(image)
```

```
<matplotlib.image.AxesImage at 0x2d0adf3e6a0>
```



Plotting with matplotlib

```
fig, ax = plt.subplots()
```



The ‘subplots’ method creates a Figure and 1 or more Axes Artists

```
fig
```

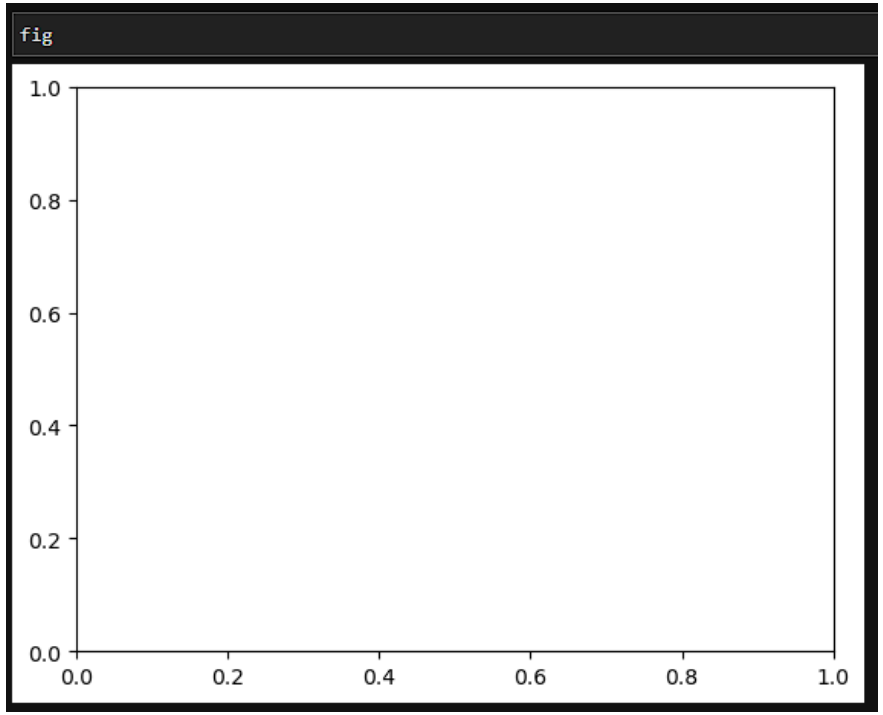


Figure: the whole figure, where many ‘Artist’ objects go inside

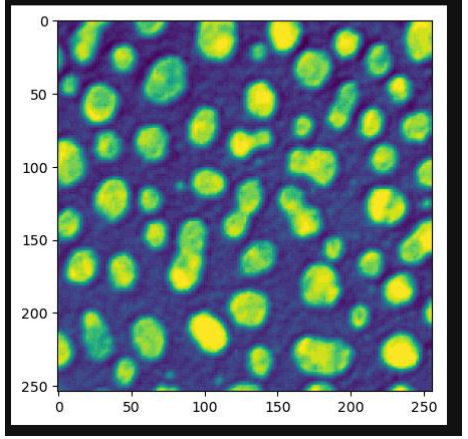
```
ax
```

```
<AxesSubplot: >
```

An Axes is an Artist attached to a Figure that contains a region for plotting data

```
import matplotlib.pyplot as plt
plt.imshow(image)
```

```
<matplotlib.image.AxesImage at 0x2d0adf3e6a0>
```



Calling imshow automatically creates a Figure and an Axes Artists under the hood.

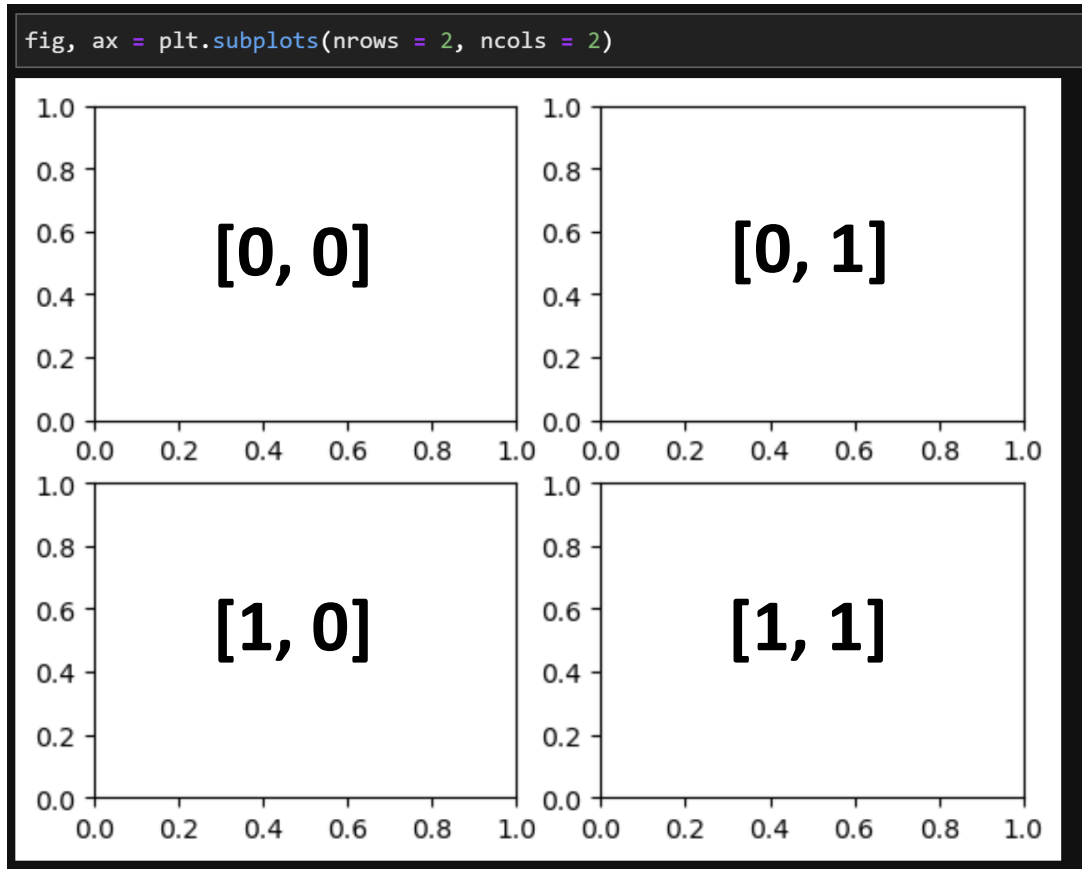
Plotting with matplotlib

- Use matplotlib to put images side-by-side

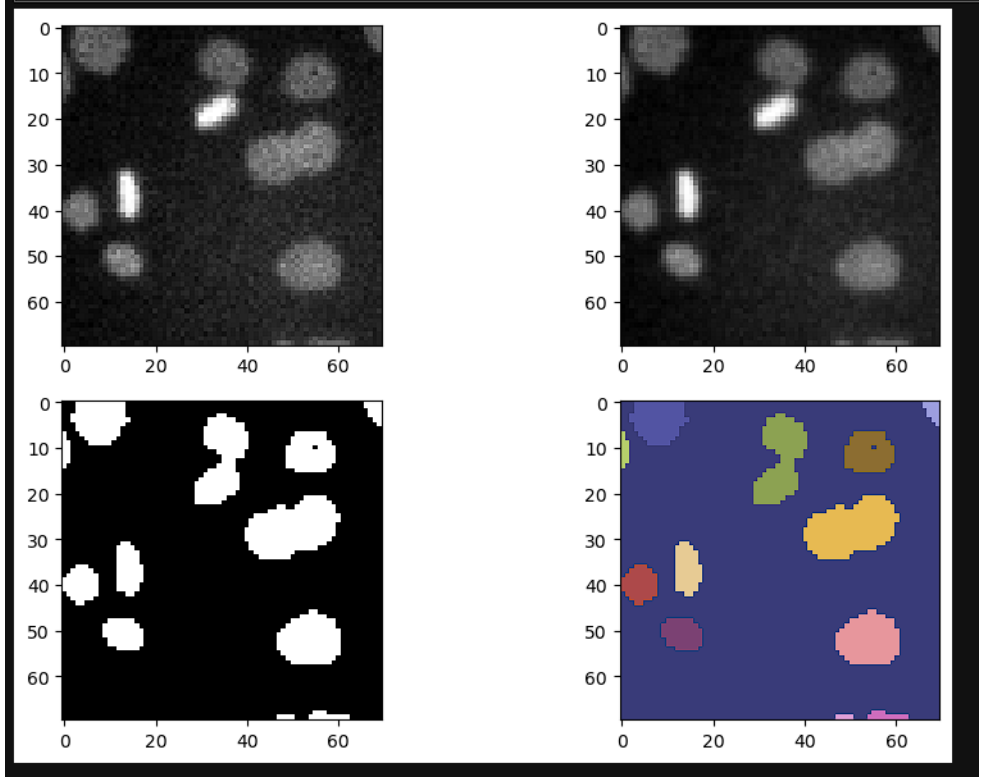
Columns 0 1

Rows 0 1

```
fig, ax = plt.subplots(nrows = 2, ncols = 2)
```




```
fig, ax = plt.subplots(nrows = 2, ncols = 2, figsize = (10,6))

ax[0, 0].imshow(image_cells, cmap = 'gray')
ax[0, 1].imshow(image_filtered, cmap = 'gray')
ax[1, 0].imshow(binary_image, cmap = 'gray')
ax[1, 1].imshow(label_image, cmap = 'tab20b')

# Handy function to avoid overlap of axes labels
plt.tight_layout()
```



https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.subplots.html

Reading Images and Folder Structures

Marcelo L. Zoccoler

Reading Images in Python

In Python, images can often be read by providing a path to a “imread” function. The path can be a relative path or an absolute path.

```python
from skimage.io import imread

image = imread('../../data/blobs.tif')
```

Relative path to image

```python
from skimage.io import imread

image = imread("C:\Users\mazo260d\Quantitative_Bio_Image_Analysis_with_Python_2022\data\blobs.tif")
```

Absolute path to image

Backslash (‘\’) is a special character. In Windows, this may lead to errors.

```
OSError: [Errno 22] Invalid argument:
```

Add a lowercase ‘r’ before path to fix that.

```python
from skimage.io import imread

image = imread(r"C:\Users\mazo260d\Quantitative_Bio_Image_Analysis_with_Python_2022\data\blobs.tif")
```

Reading Multidimentional Images in Python

PoL Physics of Life TU Dresden

- 2D images

```
image.shape
(254, 256)
```

- 3D images

```
image.shape
(36, 254, 256)
```

- Multichannel images

```
image.shape
(512, 672, 3)
```

```
channel1 = image[:,:,0]
channel2 = image[:,:,1]
channel3 = image[:,:,2]
```

- Some libraries return an array in a standardized fashion

```
image.shape
(1, 1, 1, 80, 520, 692, 1)
```

Check image metadata to know which dimension corresponds to what.

Check library documentation.

Reading Images in Python

The path can also be a variable.

```
from skimage.io import imread

image = imread(path_to_image)
```

Two popular libraries to handle paths are ‘os’ and ‘pathlib’.

pathlib

```
from pathlib import Path

image_path = Path('../../data/blobs.tif')
image_path
```

```
WindowsPath('../../data/blobs.tif')
```

name with extension

```
image_path.name
```

```
'blobs.tif'
```

name

```
image_path.stem
```

```
'blobs'
```

extension

```
image_path.suffix
```

```
'.tif'
```

parent directory

```
image_path.parent
```

```
WindowsPath('../../data')
```

os

```
import os

image_path = r'../../data/blobs.tif'
image_path
```

```
'../../data/blobs.tif'
```

parent directory + name with extension

```
os.path.split(image_path)
```

```
('../../data', 'blobs.tif')
```

parent directory with name + extension

```
os.path.splitext(image_path)
```

```
('../../data/blobs', '.tif')
```

https://docs.python.org/3/library/os.html
https://docs.python.org/3/library/pathlib.html

Reading Images from Folders in Python

PoL Physics of Life TU Dresden

pathlib

```
folder_path = Path('../../data')
folder_path
```

```
WindowsPath('../../data')
```

Iterating over a folder:

```
image_list = []
for path in folder_path.iterdir():
    image = imread(path)
    image_list.append(image)
```

```
UL.csv
UL.csv
..\..\data\BBBC007_analysis.csv
..\..\data\BBBC007_batch
```

```
image_path_list = []
for path in folder_path.iterdir():
    image_path_list.append(path)
```

```
for image_path in image_path_list:
    image = imread(image_path)
    process_image(image)
```

os

```
folder_path = r'../../data'
folder_path
```

```
'../../data'
```

Iterating over a folder:

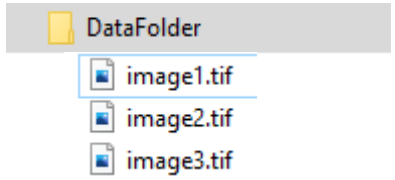
```
file_names = os.listdir(folder_path)
file_names
```

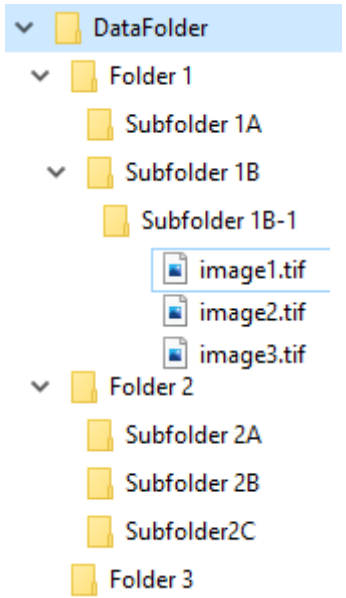
```
['BBBC007_20P1_POS0007_D_1UL.csv',
 'BBBC007_20P1_POS0010_D_1UL.csv',
 'BBBC007_analysis.csv',
 'BBBC007_batch',
```

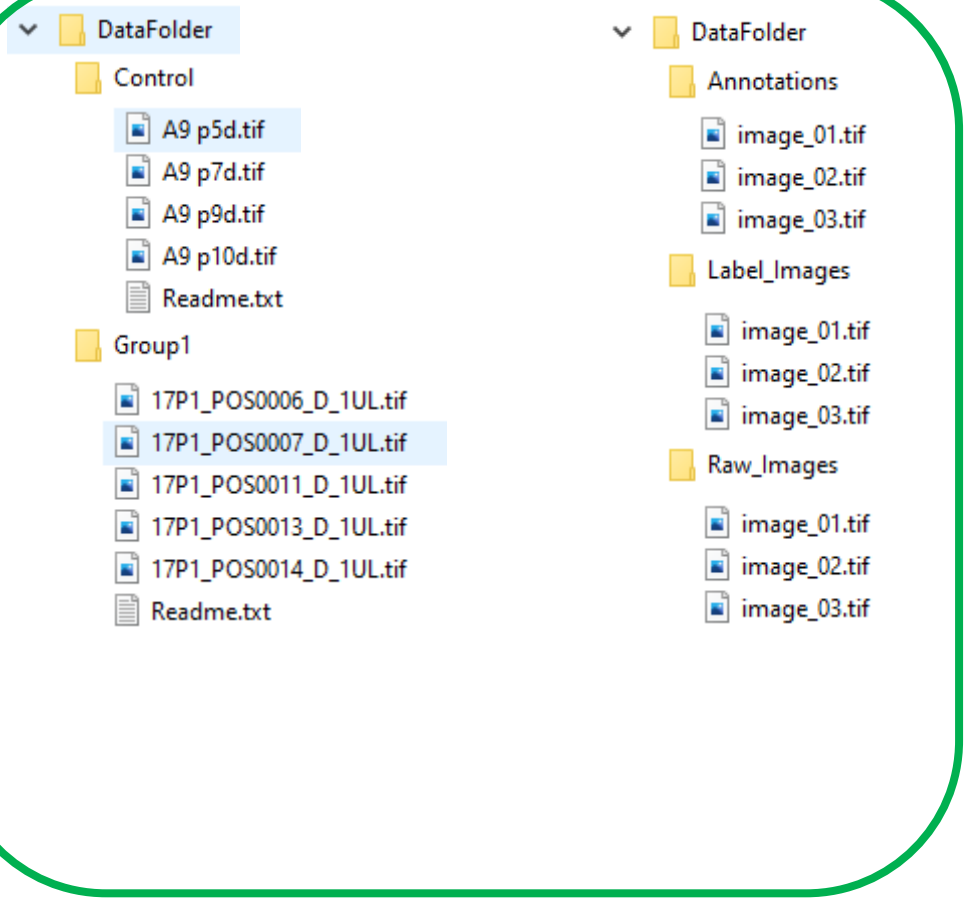
```
for file_name in file_names:
    full_path = os.path.join(folder_path, file_name)
    print(full_path)
```

```
../../data\BBBC007_20P1_POS0007_D_1UL.csv
../../data\BBBC007_20P1_POS0010_D_1UL.csv
../../data\BBBC007_analysis.csv
../../data\BBBC007_batch
```

Folder Structures







A few general advice:

- Avoid too many levels
- Add “Readme” files as soon as you create a folder (you will forget later)
- Consider using a data management platform
- Talk to a data management experts to find the best structure to your needs

Image by eikira from Pixabay (https://pixabay.com/photos/trunk-automobile-vehicle-luggage-1478832/)

Napari

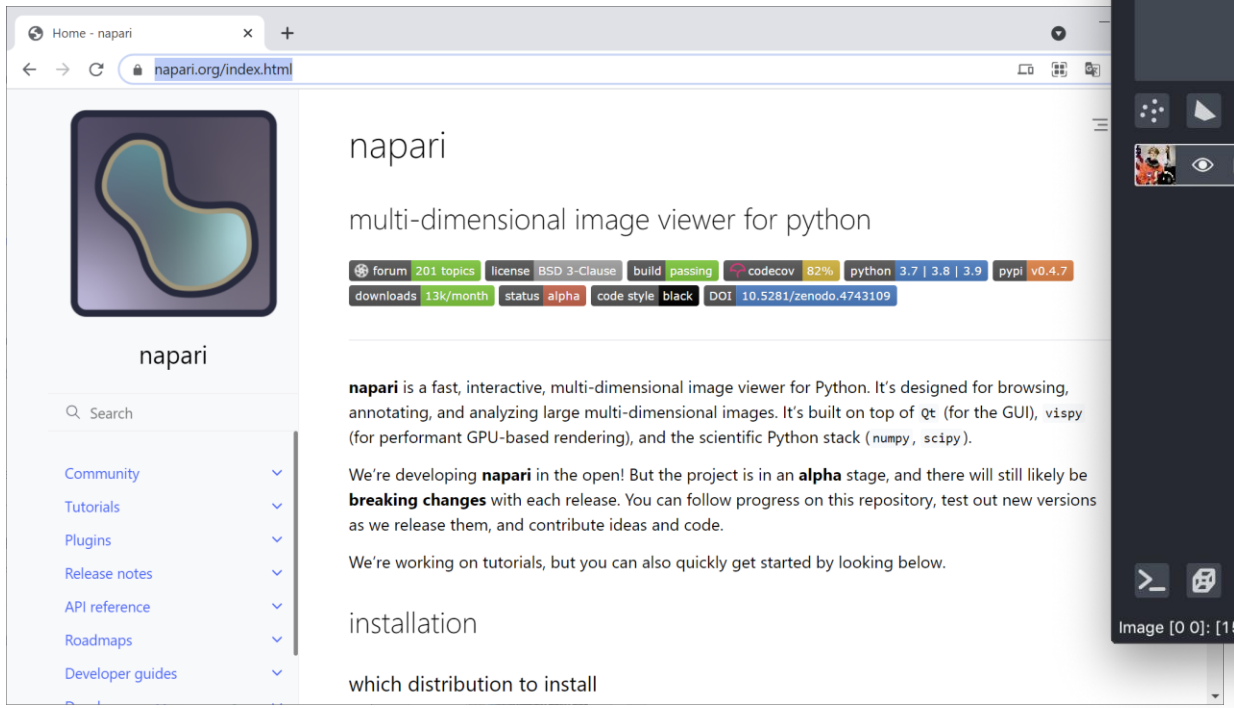
Marcelo L. Zoccoler

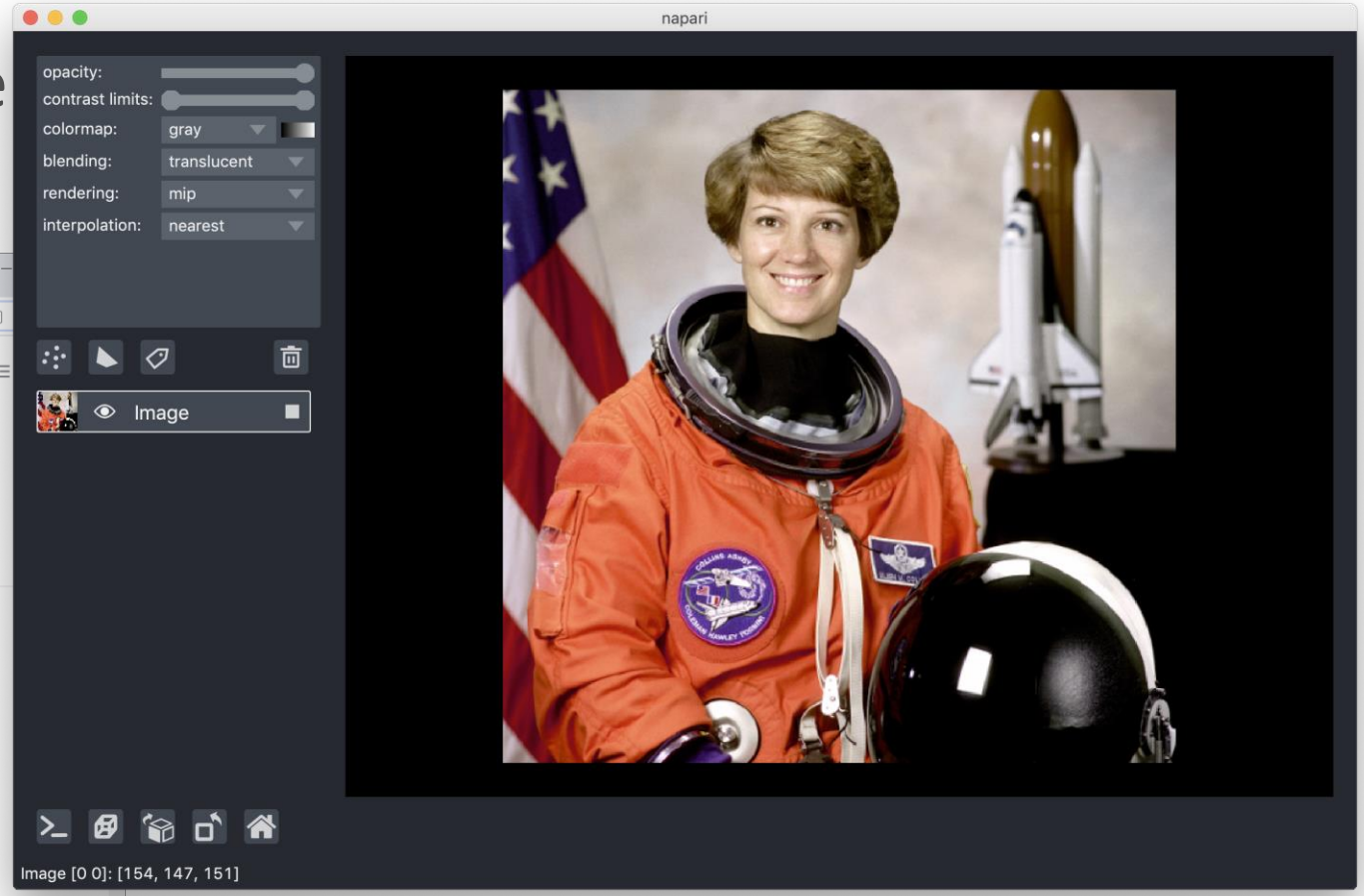
With material from:

Robert Haase, PoL; Guillaume Witz, Universität Bern

Napari: 3D viewer for Python

- Multi-dimensional image
- https://napari.org/





Napari: 3D viewer for Python

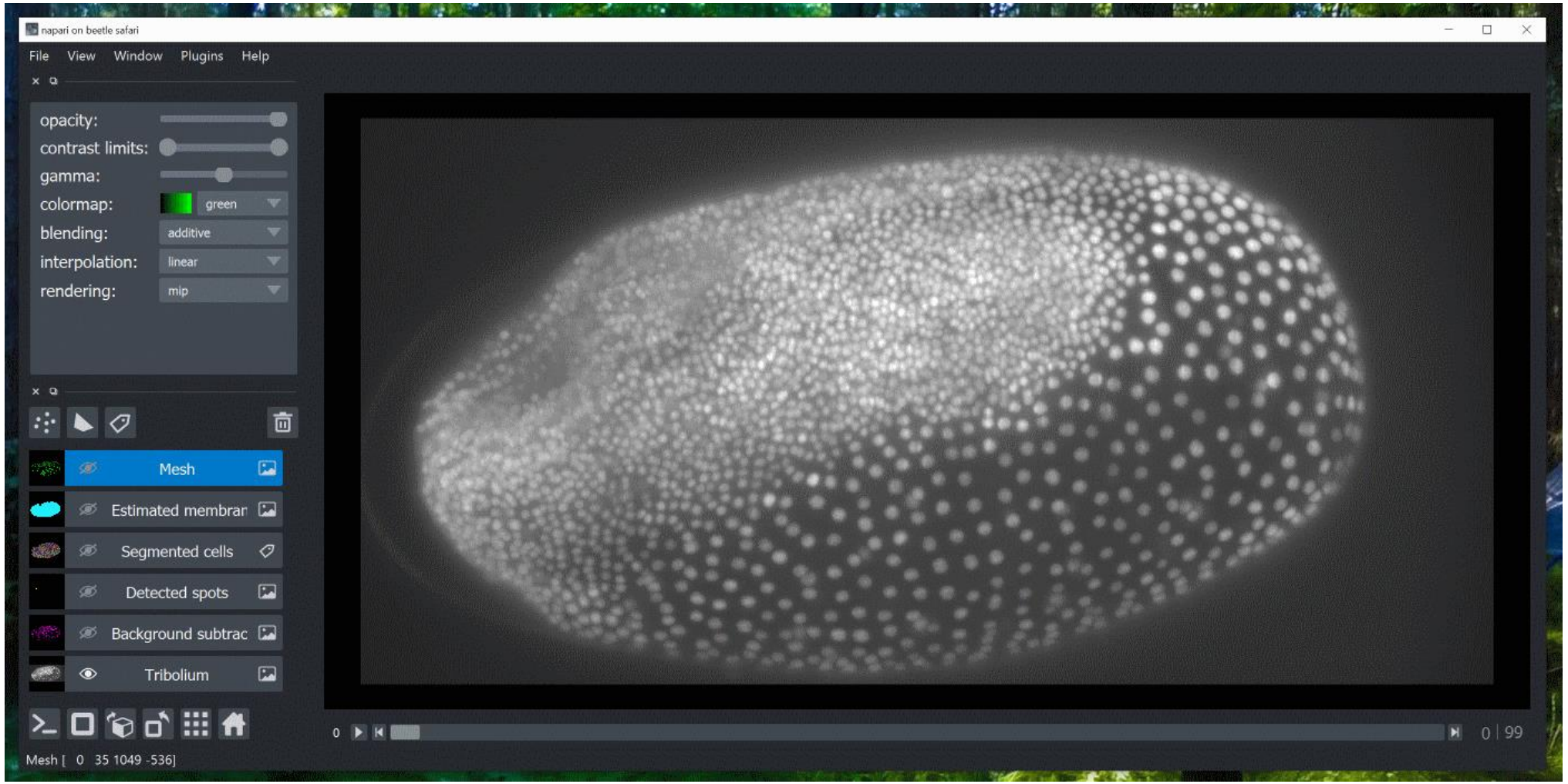


Image data source: Daniela Vorkel, Myers lab, MPI-CBG/CSBD

https://napari.org/

Napari user interface

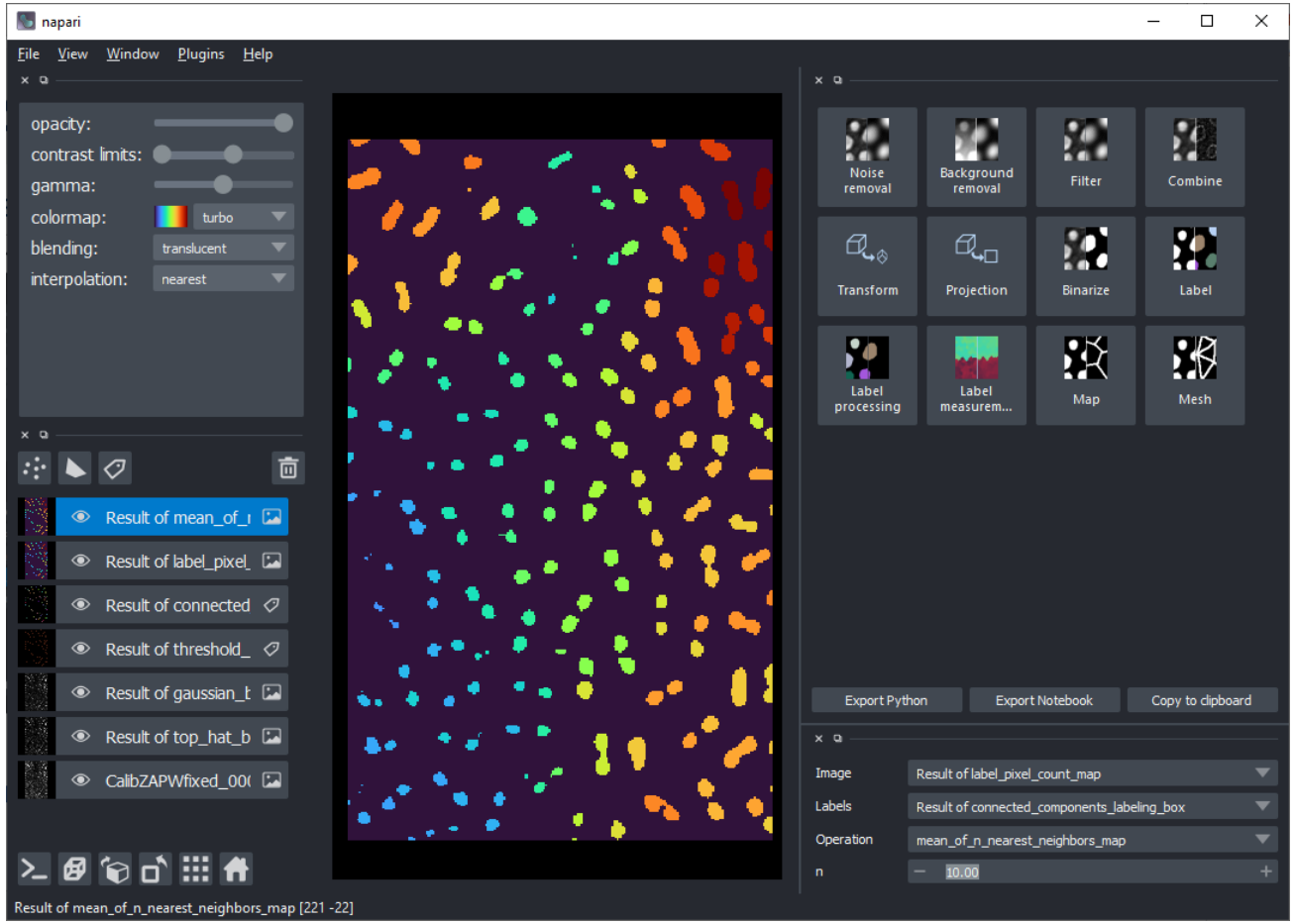
View configuration / tools

```
layer.opacity = 0.5
```

Layers

```
layer.visible = False
```

Viewer controls



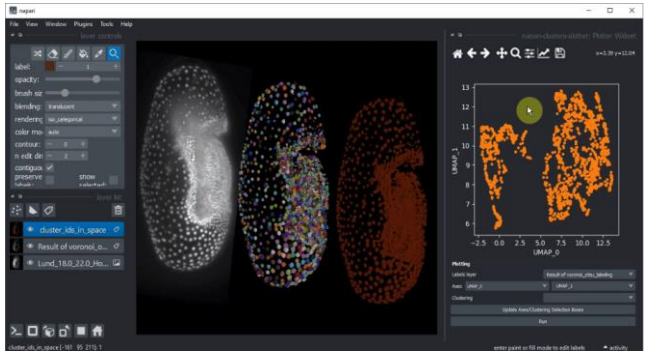
Dock widgets (custom plugins)

Function widgets (custom plugins)

https://napari.org/tutorials/fundamentals/viewer.html

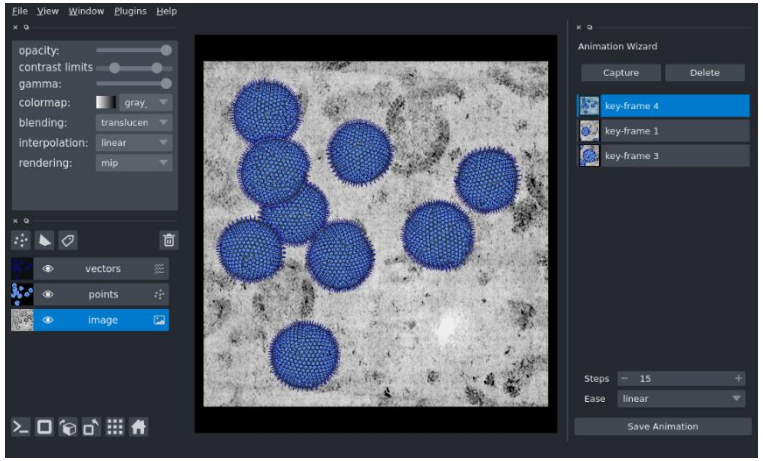
napari plugins

clusters-plotter



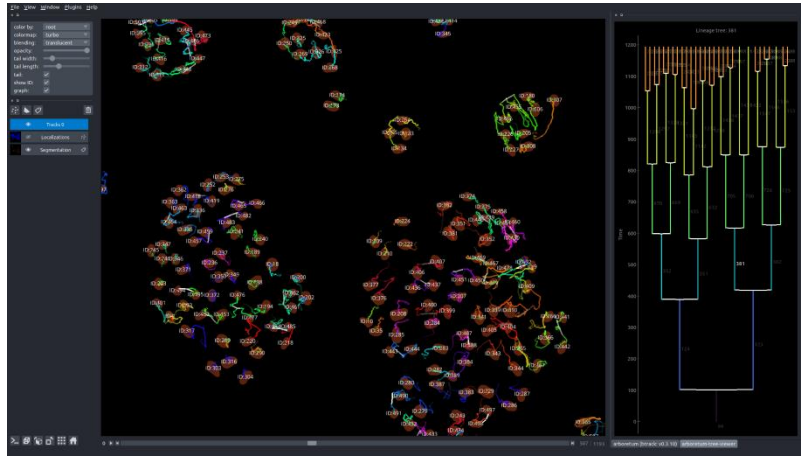
https://github.com/BiAPoL/napari-clusters-plotter

animation



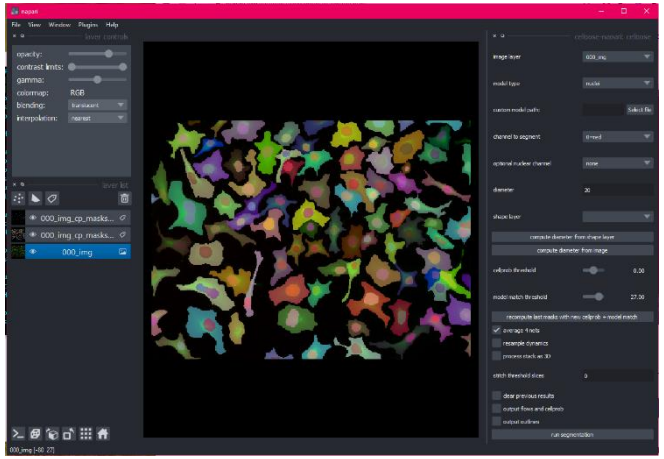
https://github.com/napari/napari-animation

arboretum



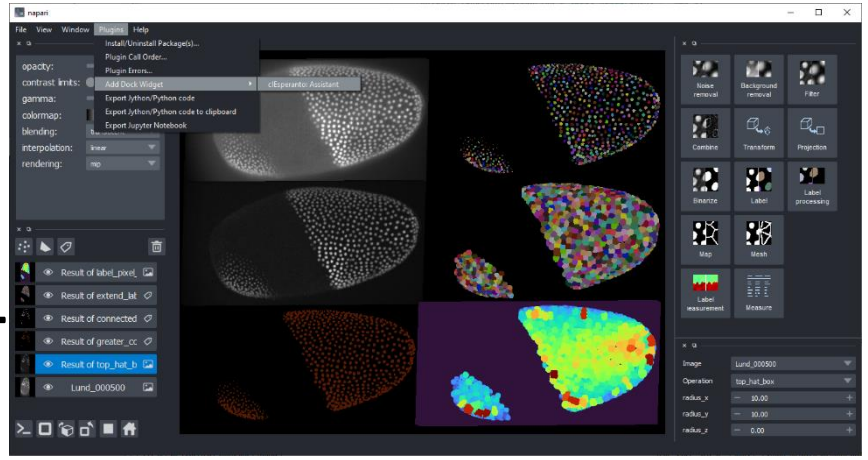
https://github.com/quantumjot/arboretum

cellpose



https://cellpose-napari.readthedocs.io/en/latest/

clesperanto

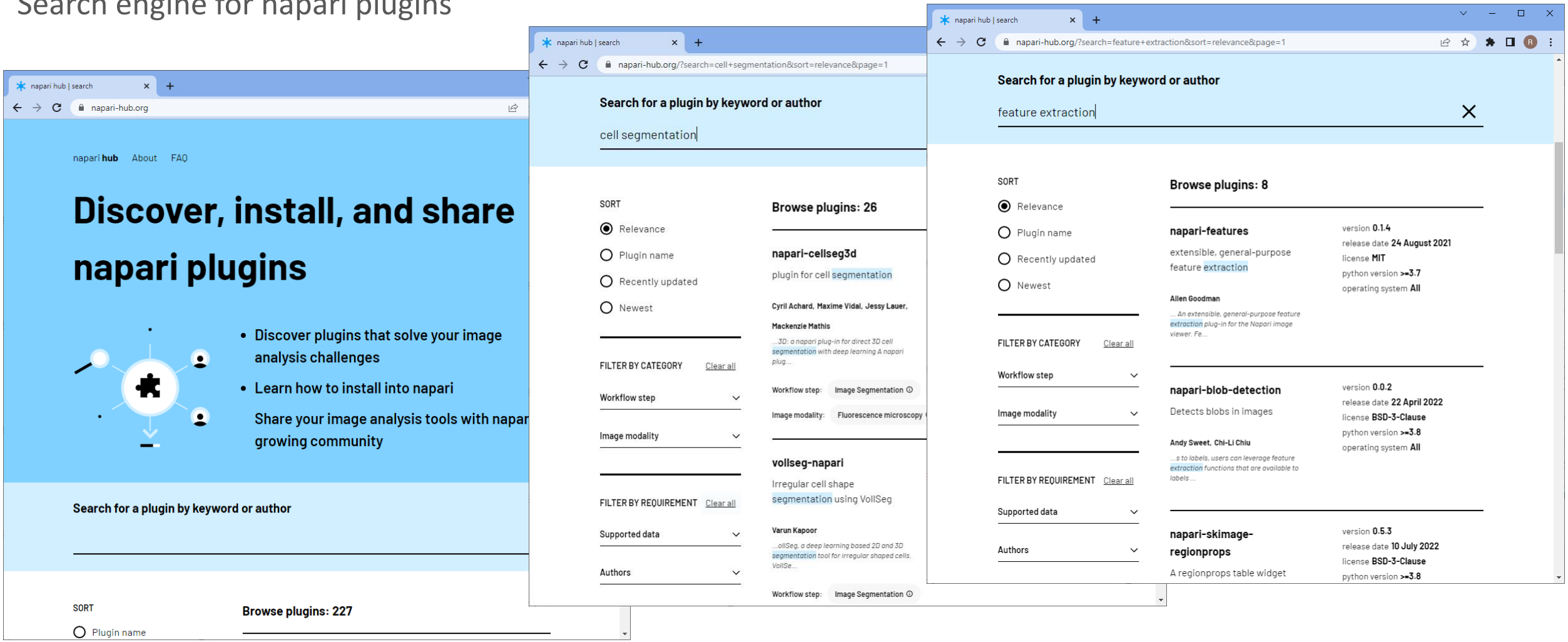


https://github.com/clEsperanto/napari_pyclesperanto_assistant

In development: https://github.com/topics/napari-plugin
Released: https://pypi.org/search/?q=&o=&c=Framework+%3A%3A+napari

The Napari Hub

- Search engine for napari plugins



https://www.napari-hub.org/

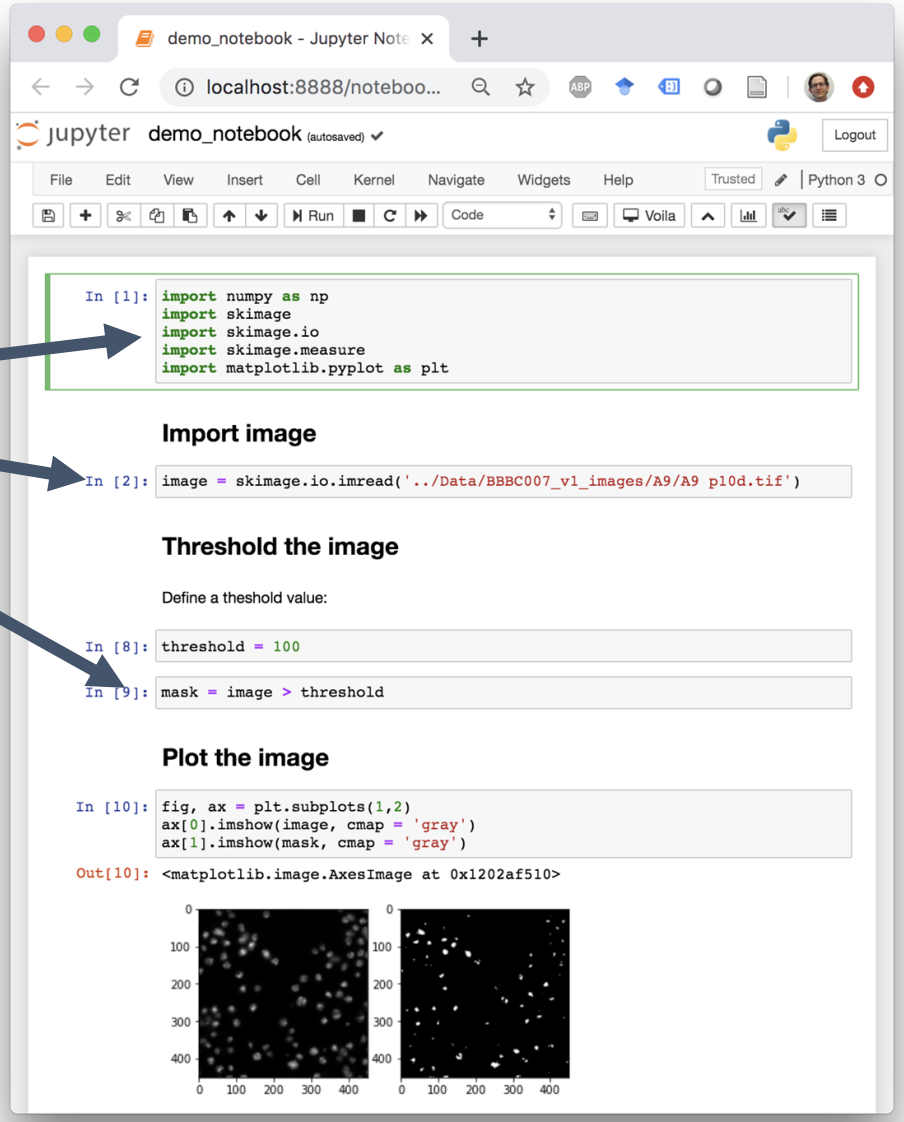
Napari from jupyter notebooks

What is a jupyter notebook?

A text file (easily sent around)

Rendered by Jupyter in the browser

Split into sections called cells



Napari from jupyter notebooks

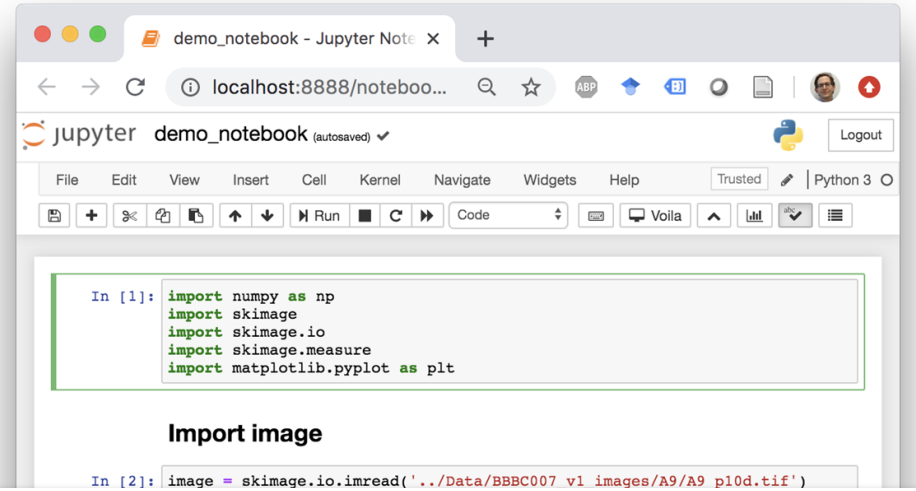
A text file (easily sent around)

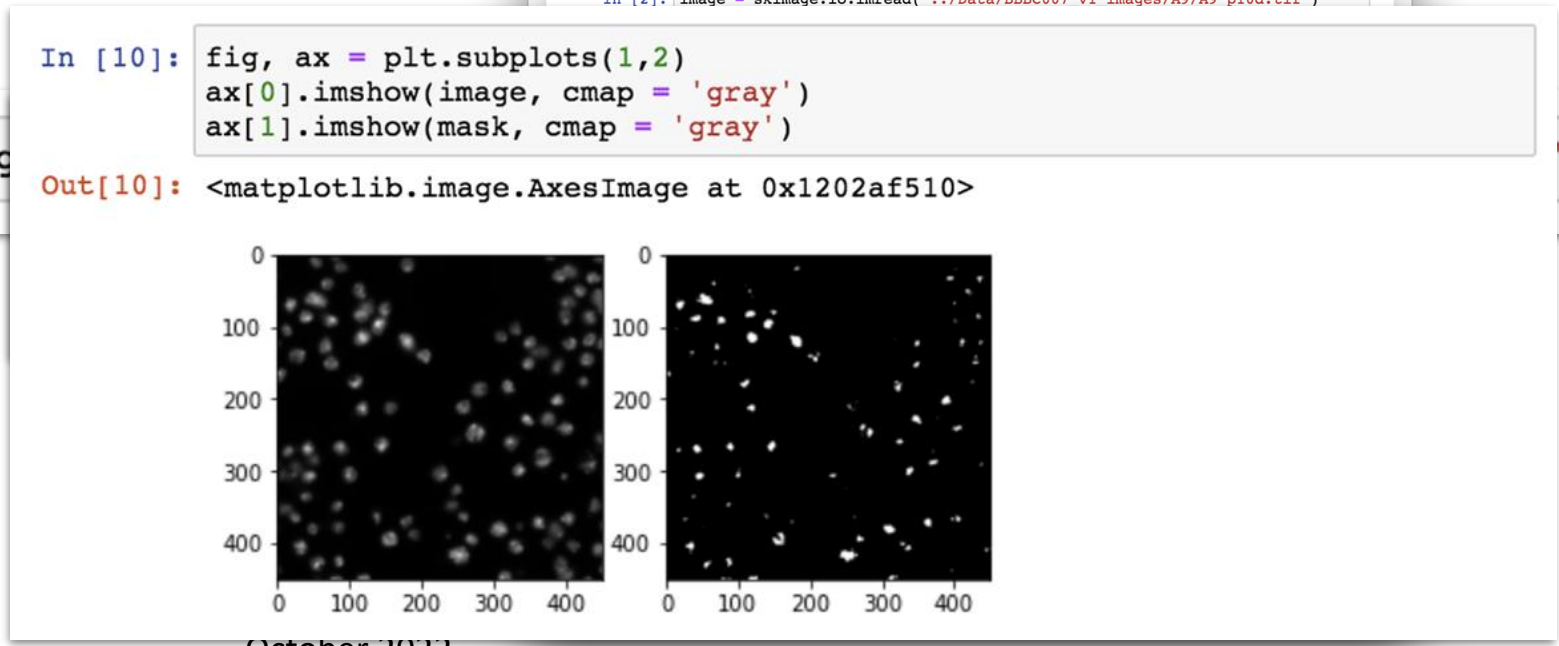
Rendered by Jupyter in the browser

Split into sections called cells

Cells can contain:

- Code
- Formatted text
- Rich output





Why using notebooks with napari?

PoL Physics of Life TU Dresden

- Easy data interaction and visualization with napari:
    - Call functions from code and visualize results in napari
    - Great for visualizing 3D (and more) data
    - Each processing step result can be displayed as a separate layer
- Data annotation

Scripting napari in notebooks

- Add layers to napari to visualize intermediate processing results on top of each other or side by side.
- Change layer visualization within napari…

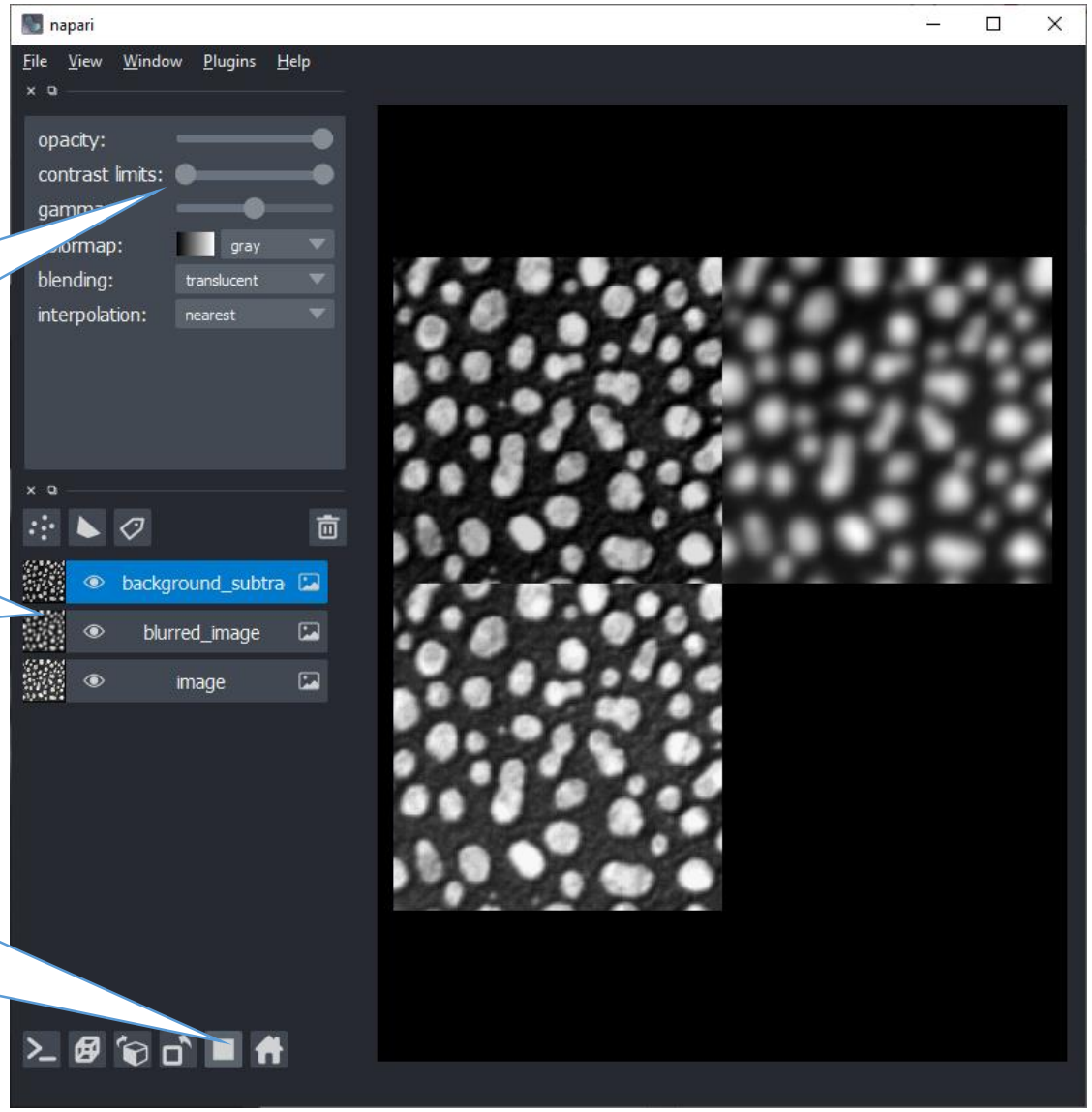
  … or via code in a jupyter notebook:

```
viewer.layers[0].contrast_limits
```

```
[0, 255]
```

1. Access the viewer
2. Access the layers
3. Choose a layer (by index or name)
4. “Press TAB” and check out available properties

```
viewer.layers[0].contrast_limits = [30,170]
```



Change Brightness and contrast here

Toggle visualization of layers here

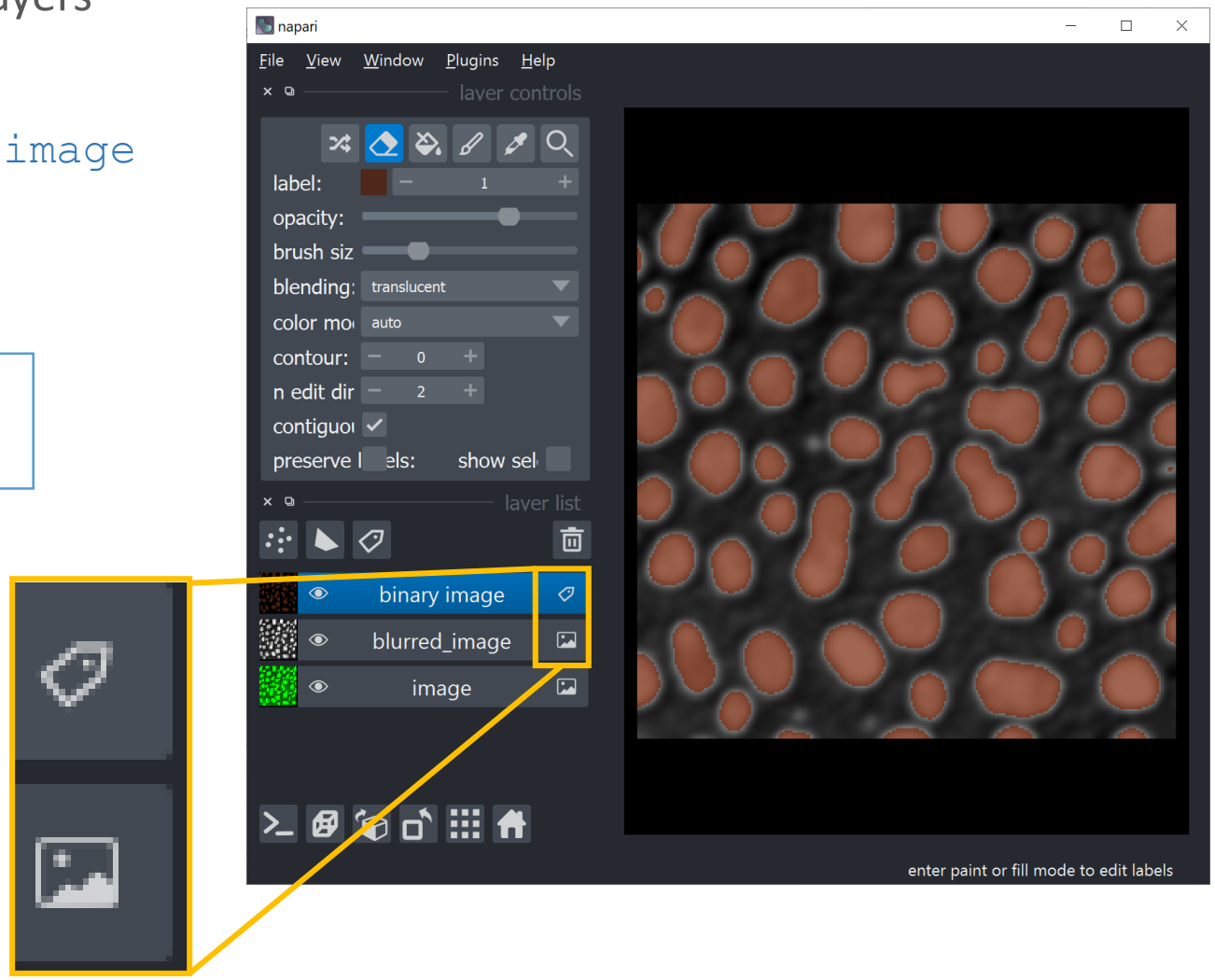
Gallery view

Visualizing image segmentation

- Binary images and label images visualized as label layers

```
# Add a new labels layer containing an image
viewer.add_labels(binary_image,
                  name="binary image")
```

Name your layers to keep track of what they contain



Labels Layer

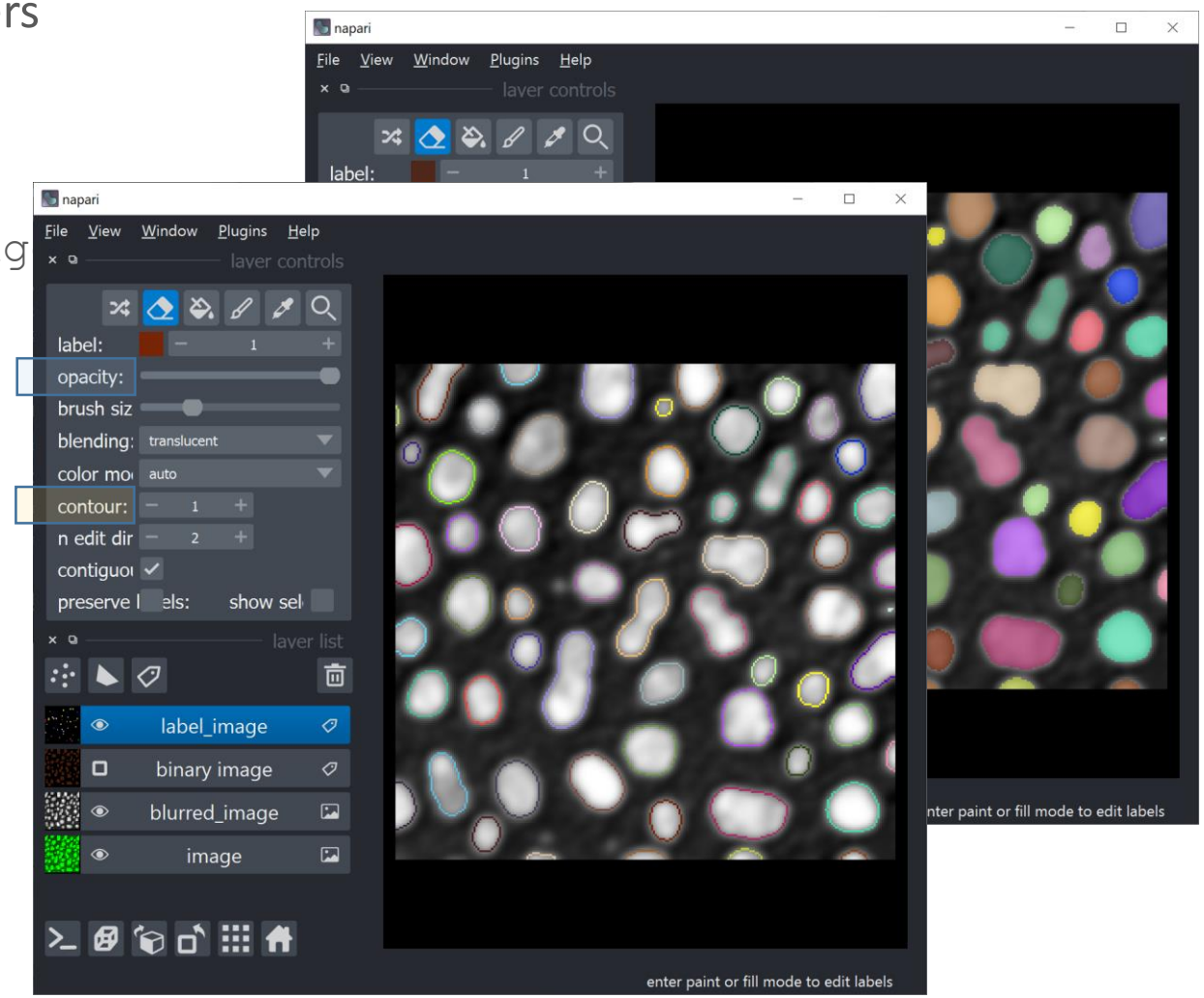
Image Layer

Visualizing image segmentation

- Binary images and label images visualized as label layers

- `# add labels to viewer`
- `label_layer = viewer.add_labels(label_imag`

- Visualize contours instead of the overlay
- `label_layer.contour = 1`
- `label_layer.opacity = 1`

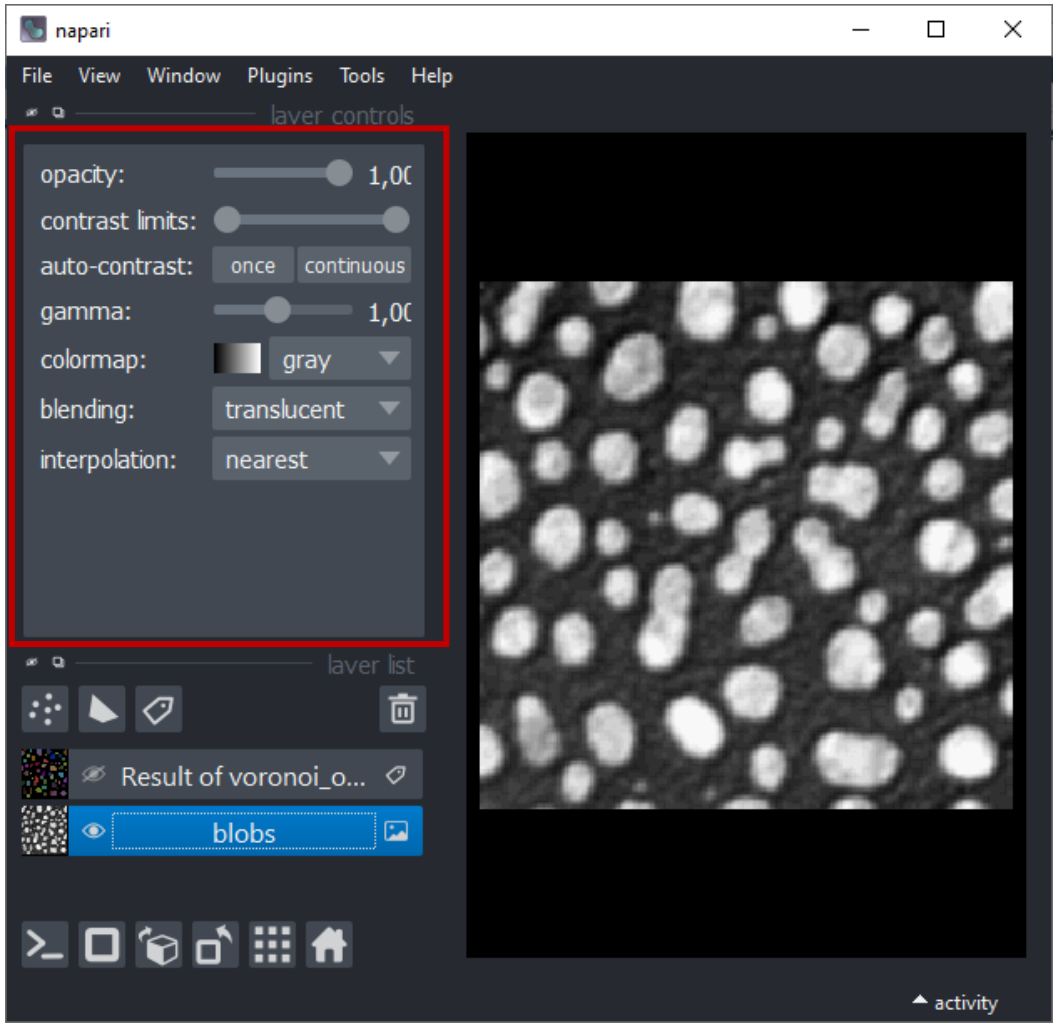


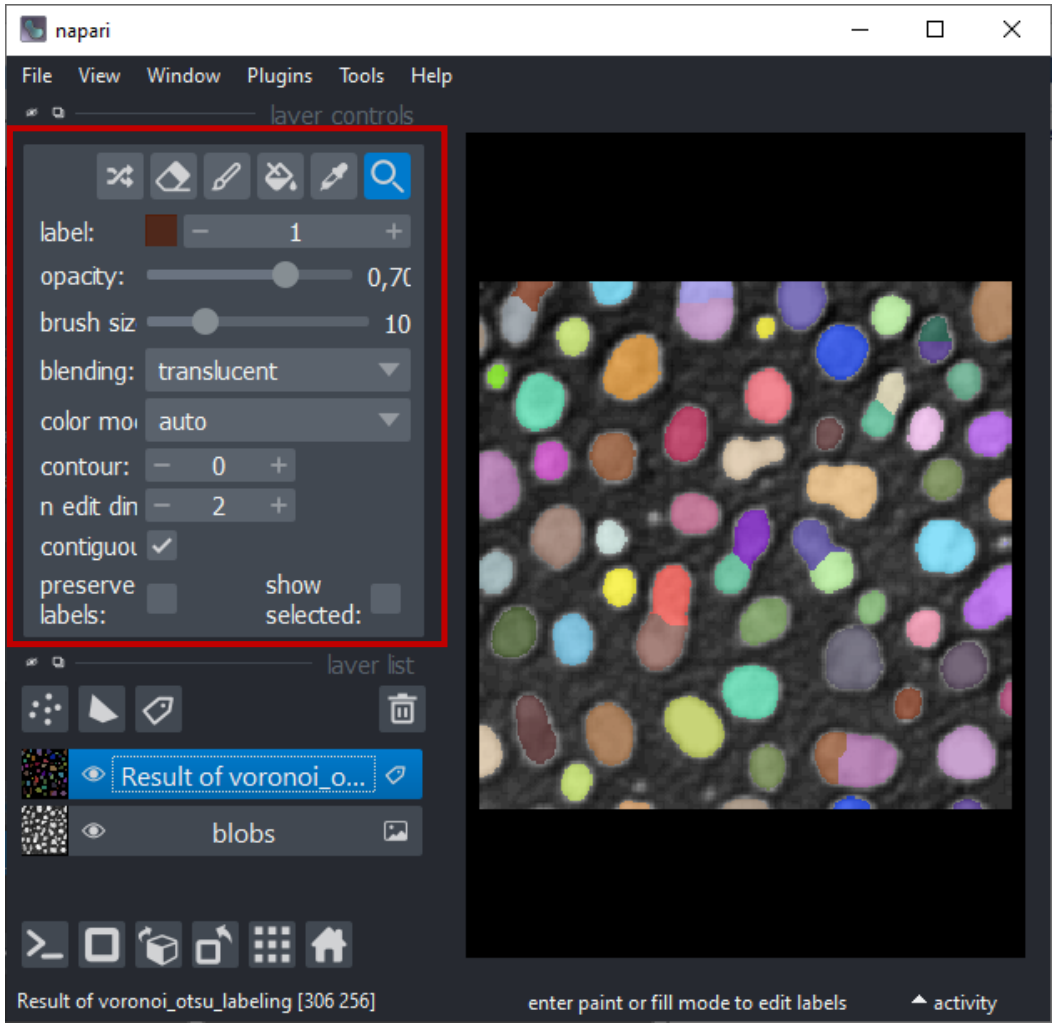
http://nbviewer.jupyter.org/github/BiAPoL/on_the_fly_image_processing_napari/blob/master/01_napari.ipynb

Visualizing image segmentation

PoL Physics of Life TU Dresden

Different layers have different configurations





Points layers

- There is also other layer types
  - Shapes
  - Points
  - Surfaces
  - Tracks
  - Vectors

```
# add points to viewer
label_layer = viewer.add_points(points,
  face_color='green', symbol='cross', size=5)
```