Image Processing and Filtering

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

April 2022

@haesleinhuepf @zoccolermarcelo

Last week: Python algorithms

PoL Physics of Life TU Dresden

- Conditions: if-else statement:

```
if condition :
    # do something if
    # condition is true

else :
    # do something else if
    # condition is false

# do in any case
```

- Loops: for, while
  - Generate arrays within for loops

```
# for loops
for i in range(0, 5):
    print(i)
```

```
0
1
2
3
4
```

```
# we start with an empty list
numbers = []

# and add elements
for i in range(0, 5):
    numbers.append(i * 2)

print(numbers)
```

```
[0, 2, 4, 6, 8]
```

- Functions:

```
def sum_numbers(a, b):

    result = a + b

    return result
```

name (parameters)

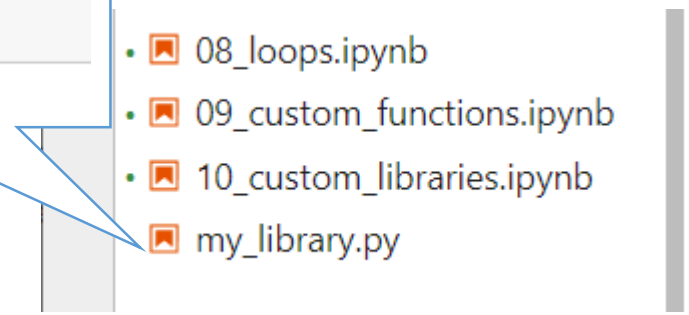
body commands

return statement (optional)

- Libraries:

```
from my_library import square

square(5)
```

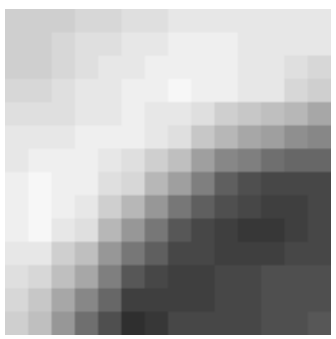
```
25
```

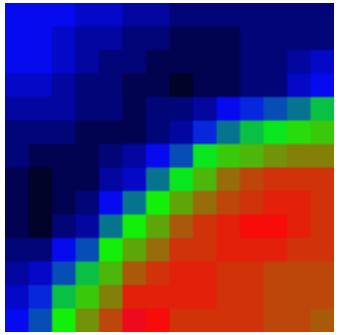


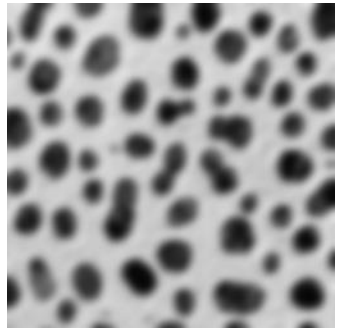
Lecture overview

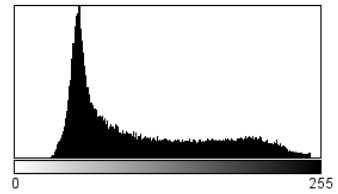
Today

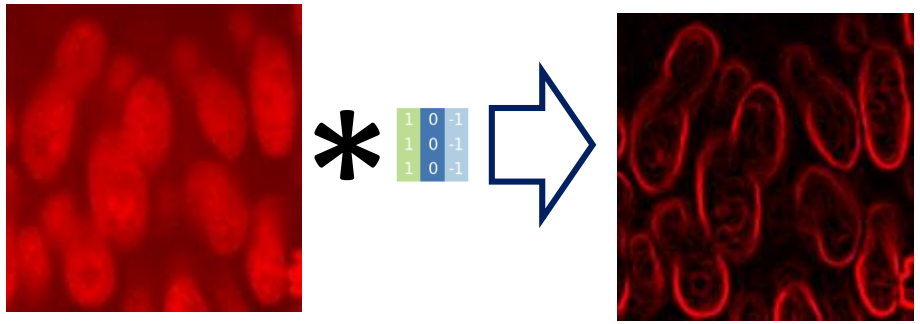
- Image Visualization
  - Pixels, Voxels
  - Resolution versus pixel size
  - Lookup tables/ Colormaps
  - Histograms
  - Brightness and Contrast
  - Introduction to napari
- Image Processing
  - Filters
  - Morphological Operations
  - Noise Removal
  - Background Subtraction
- Practicals

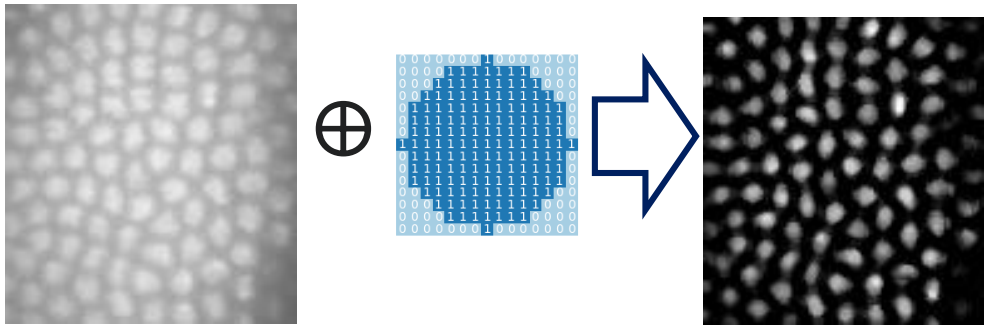


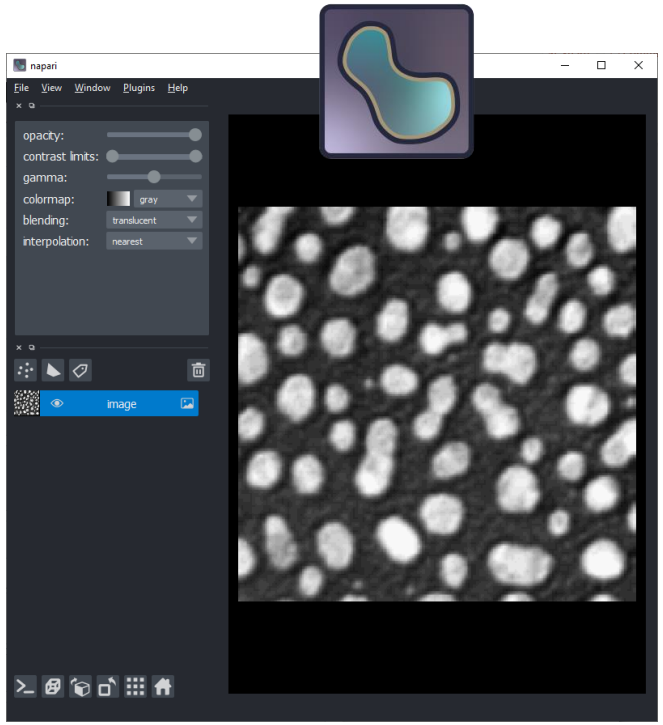












Lecture overview

Image by Pezibear from Pixabay,
Pixabay License - Free for commercial use - No attribution required

PoL
Physics of Life
TU Dresden

CENTER FOR
SYSTEMS BIOLOGY
DRESDEN

Image Visualization

Marcelo Leomil Zoccoler

With material from

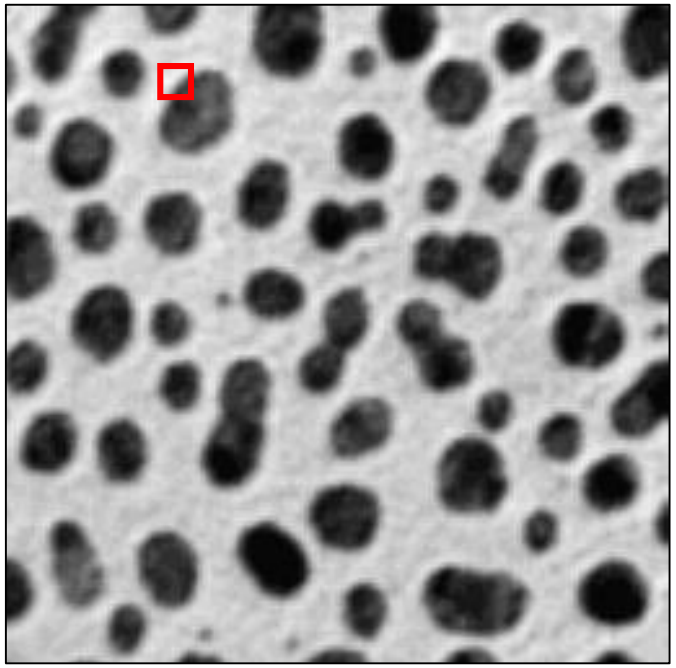
Robert Haase, PoL

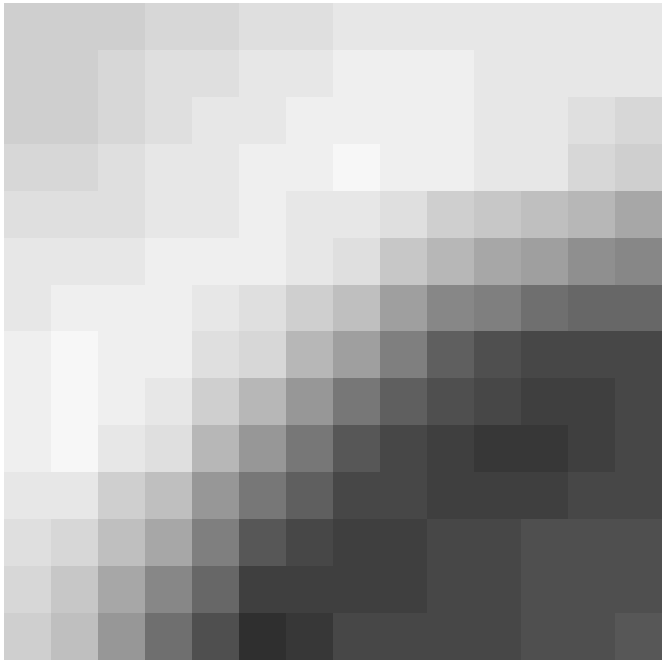
April 2022

@haesleinhuepf @zoccolermarcelo

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!

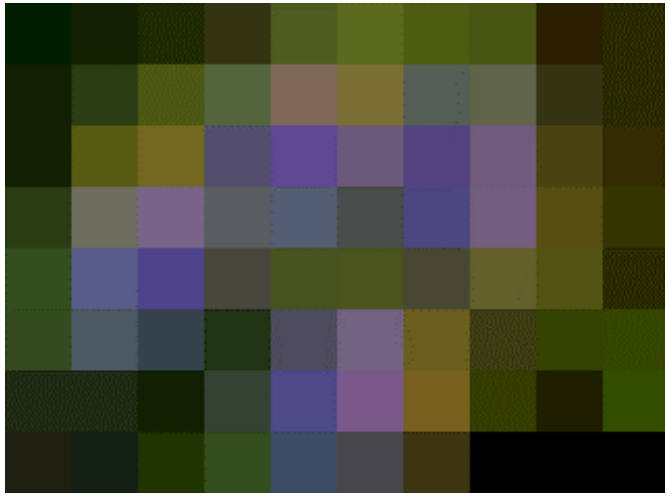




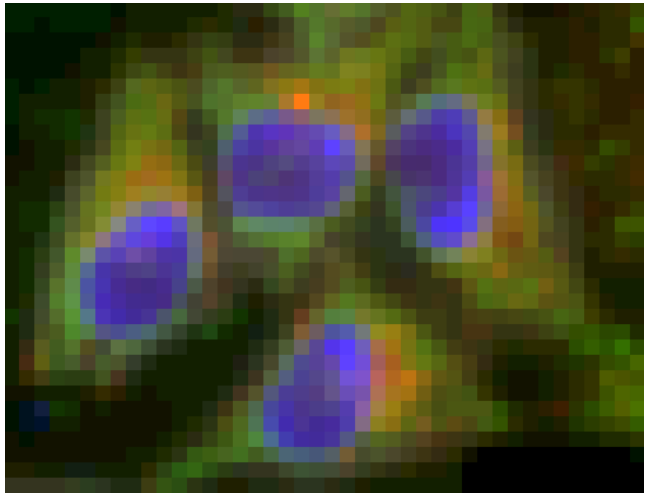
48	48	48	40	40	32	32	24	24	24	24	24	24	24
48	48	40	32	32	24	24	16	16	16	24	24	24	24
48	48	40	32	24	24	16	16	16	16	24	24	32	40
40	40	32	24	24	16	16	8	16	16	24	24	40	48
32	32	32	24	24	16	24	24	32	48	56	64	72	88
24	24	24	16	16	16	24	32	56	72	88	96	112	120
24	16	16	16	24	32	48	64	96	120	128	144	152	152
16	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

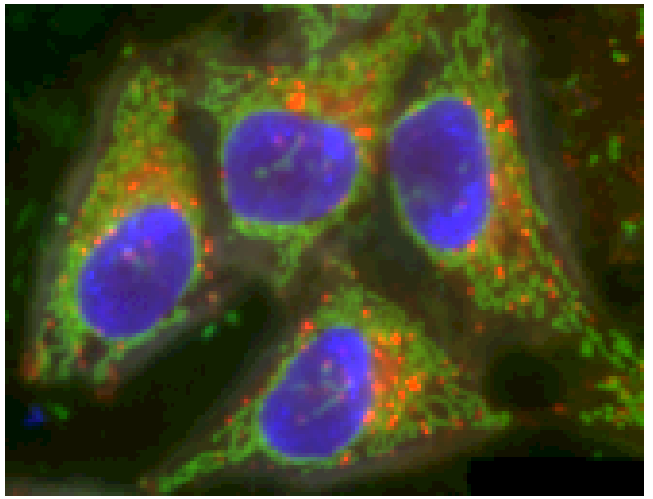
- Pixel size is a digital property of an image.
- You configure it during the imaging session at the microscope.



Pixel size: 3.3 µm



Pixel size: 0.8 µm

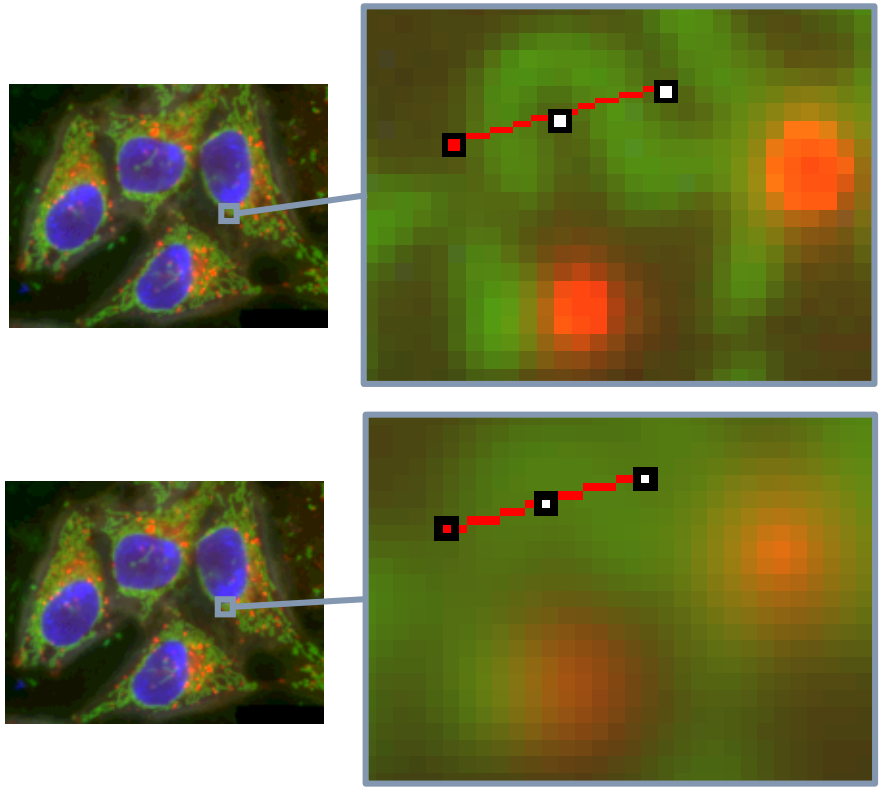


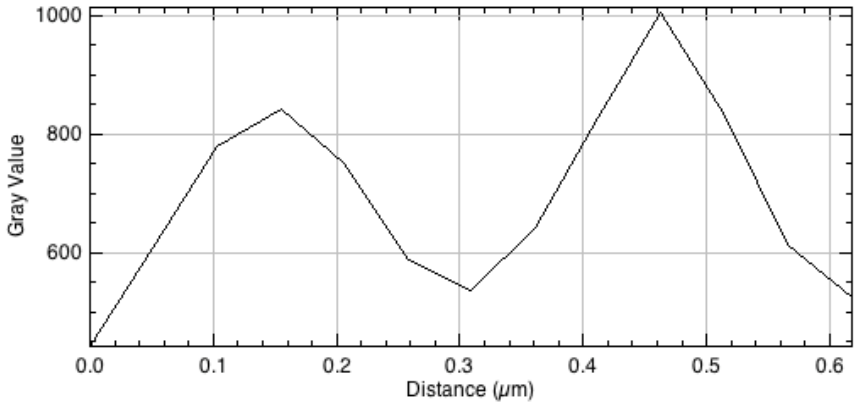
Pixel size: 0.05 µm

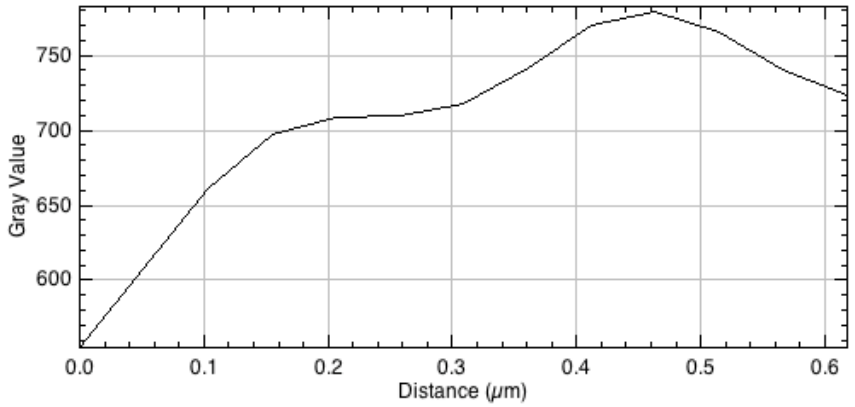
- We are not talking about resolution!

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.

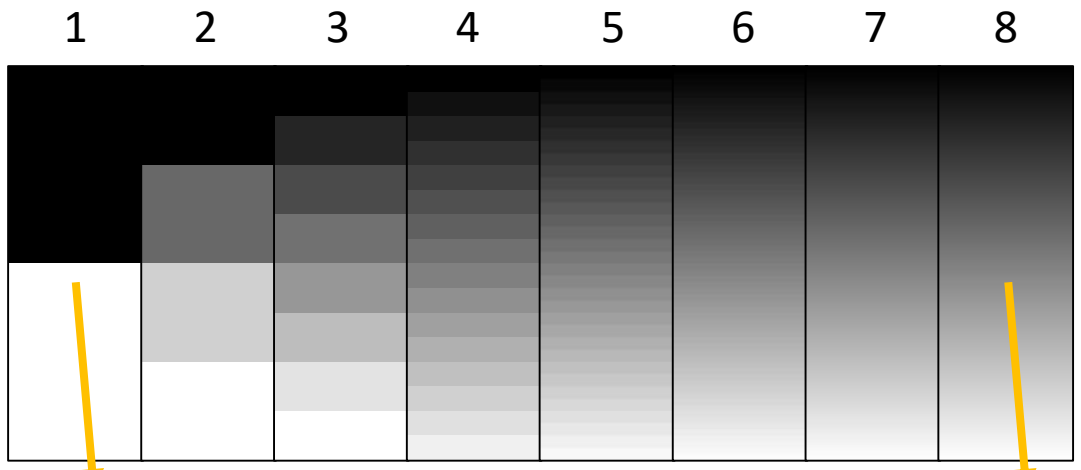


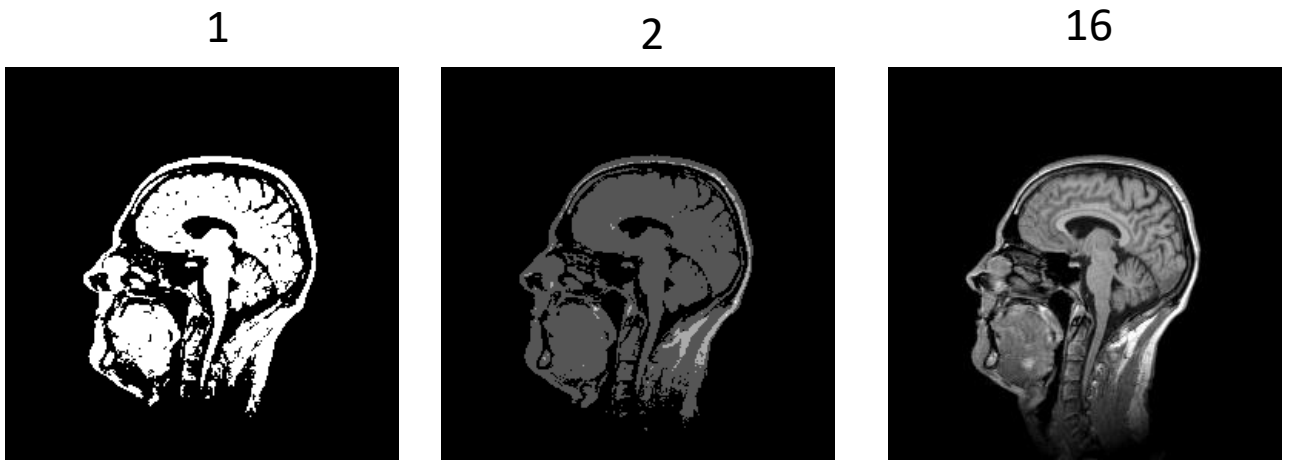




Bit-depth

- 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')
```

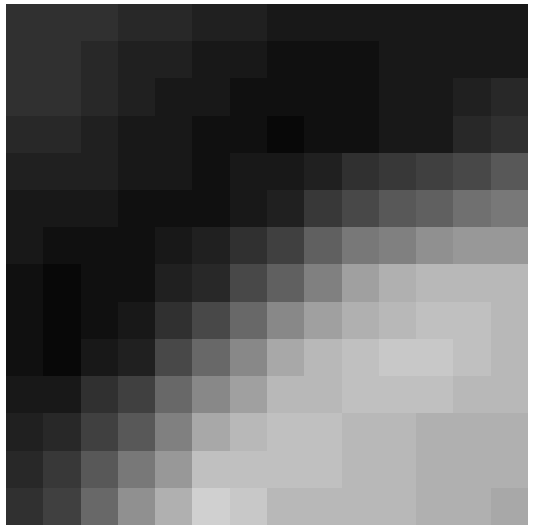
```
image_16bit = image.astype('uint16')
```

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

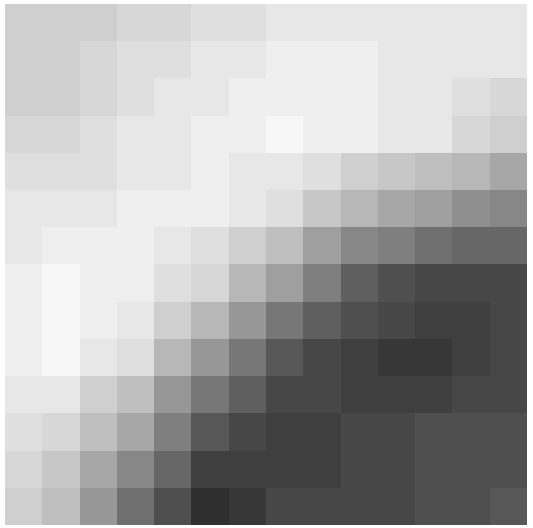
Pixel value	Display color
0 1 2 … 255	

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



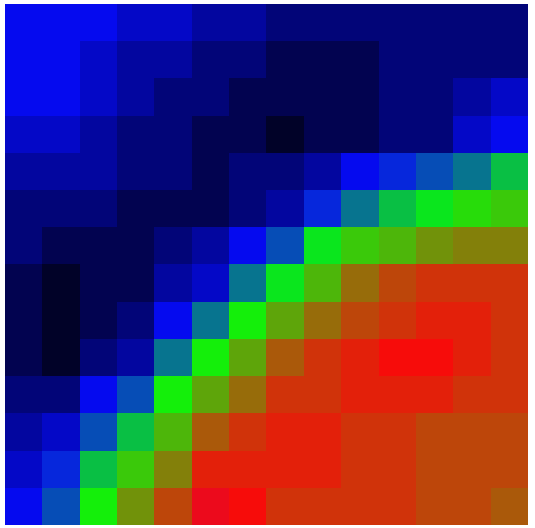
Pixel value	Display color
0 1 2 … 255	

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



Pixel value	Display color
0 1 2 … 255	

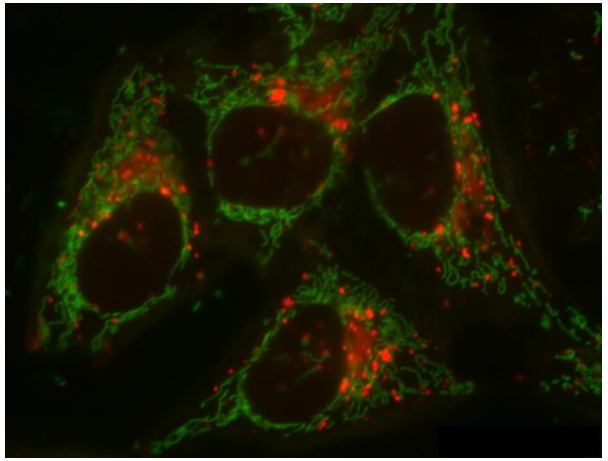
```
plt.imshow(image, cmap='jet')
```



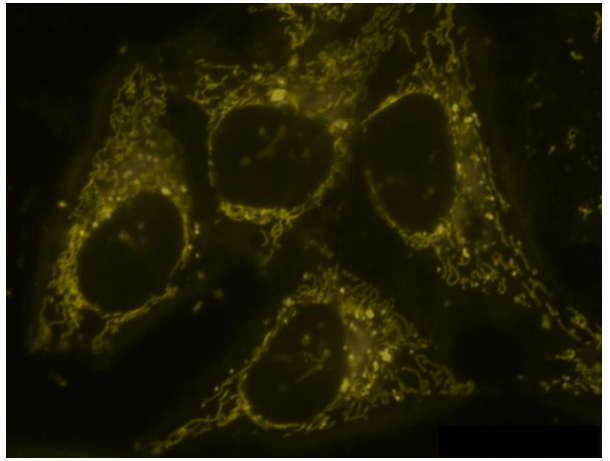
Lookup tables

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

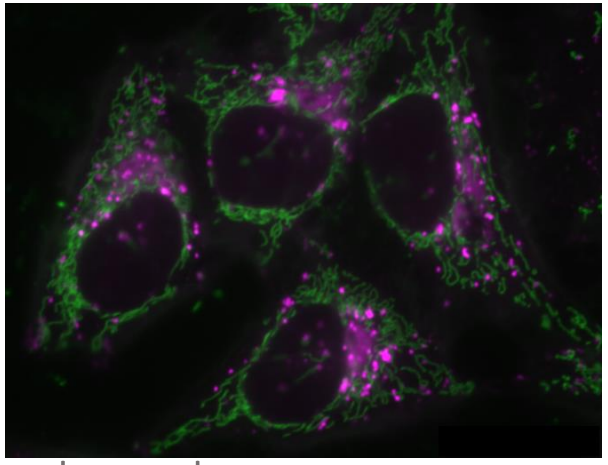
Default view

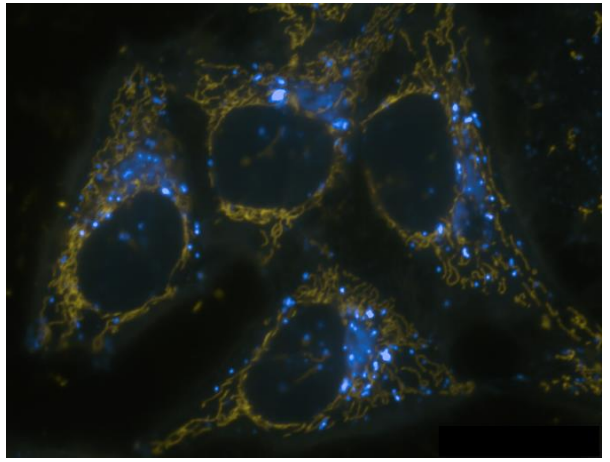


Red/green blind people see it like this



Replace red with magenta!





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)
```

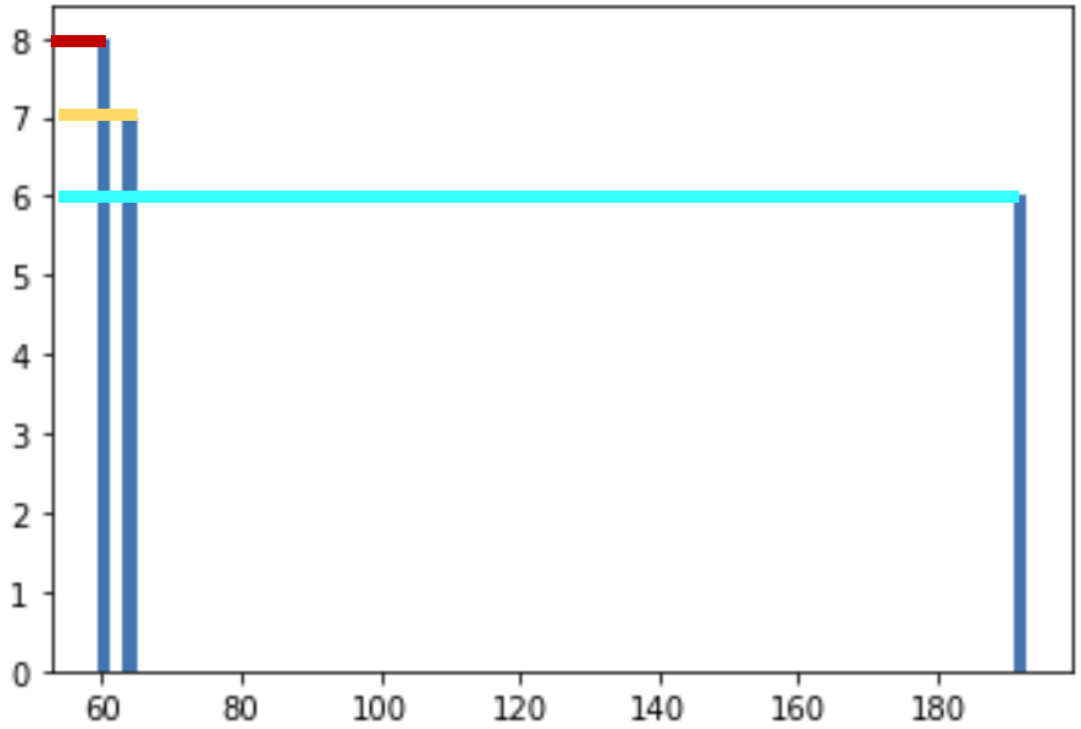
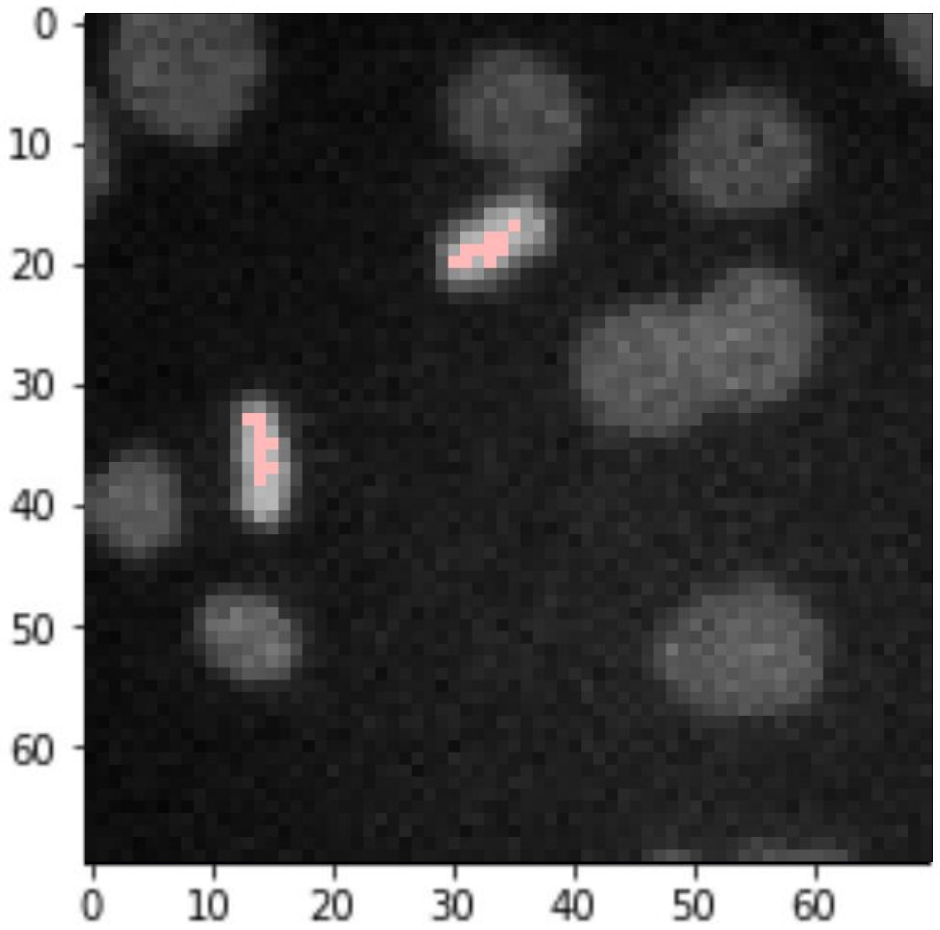
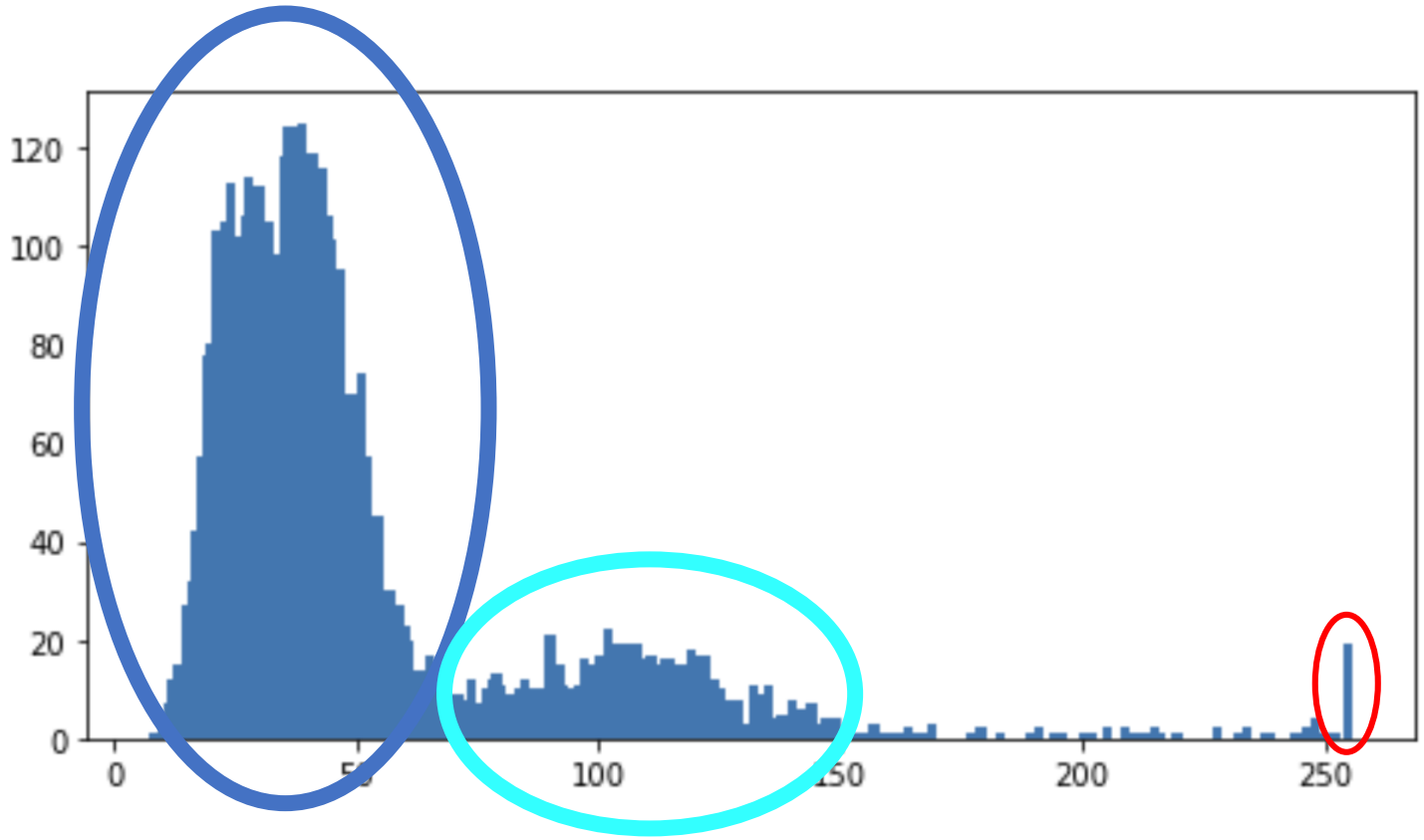


Image Histogram

Example with a slightly larger image:

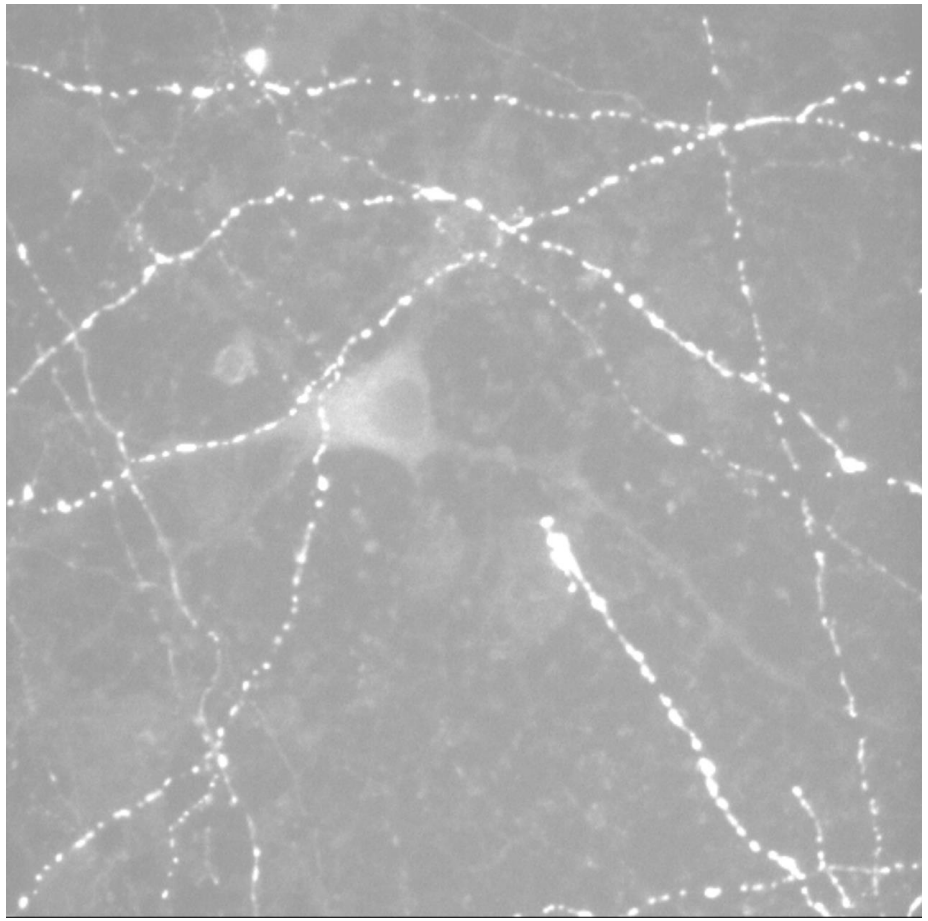




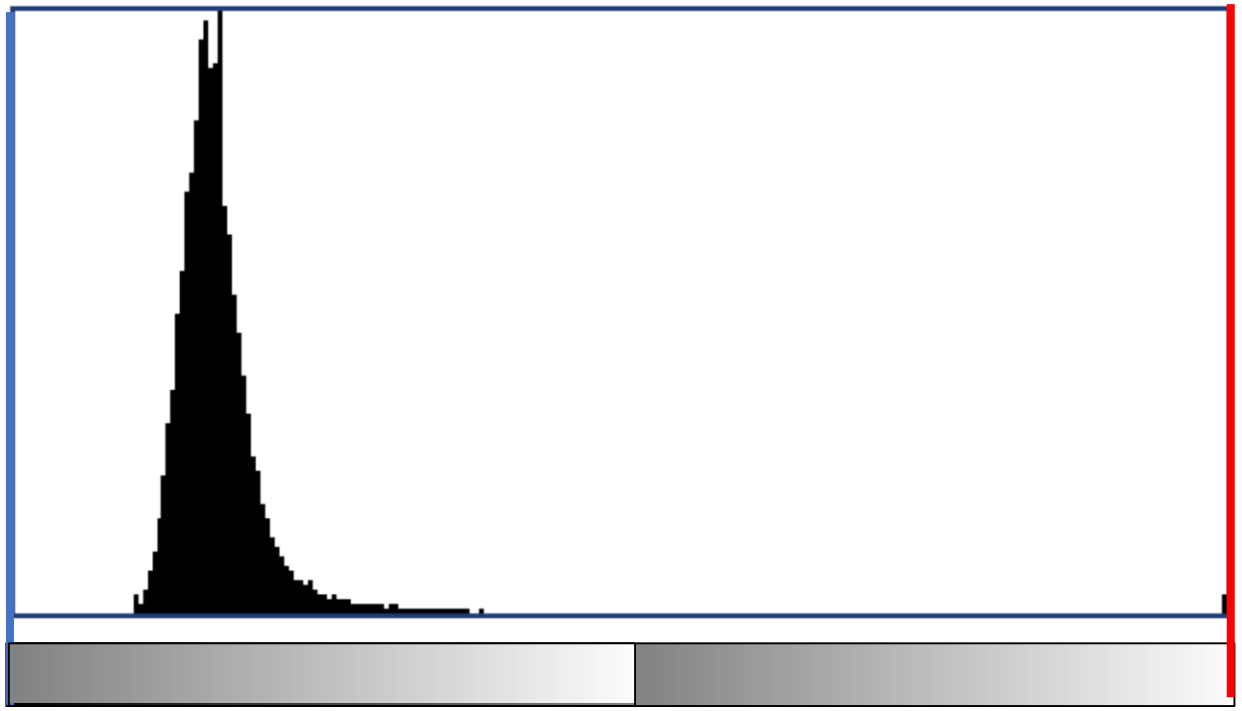
Visualisation: brightness / contrast

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

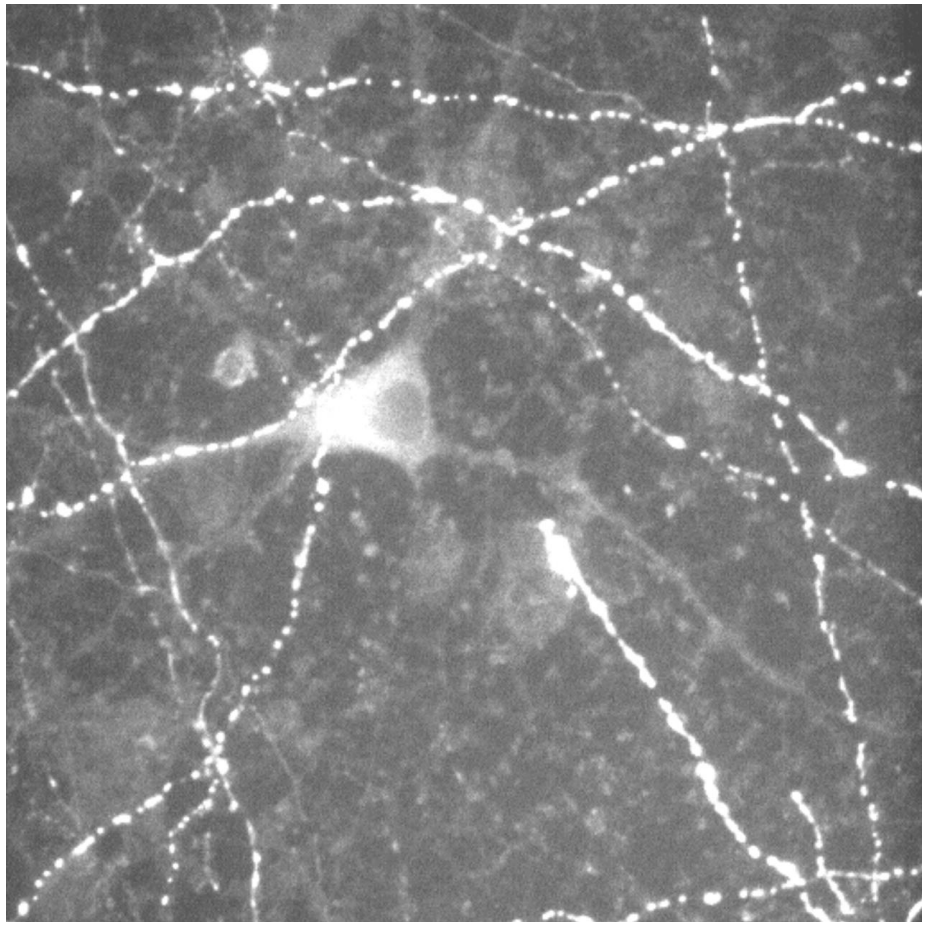


```
plt.imshow(image, cmap='gray', vmin=0, vmax=255)
```

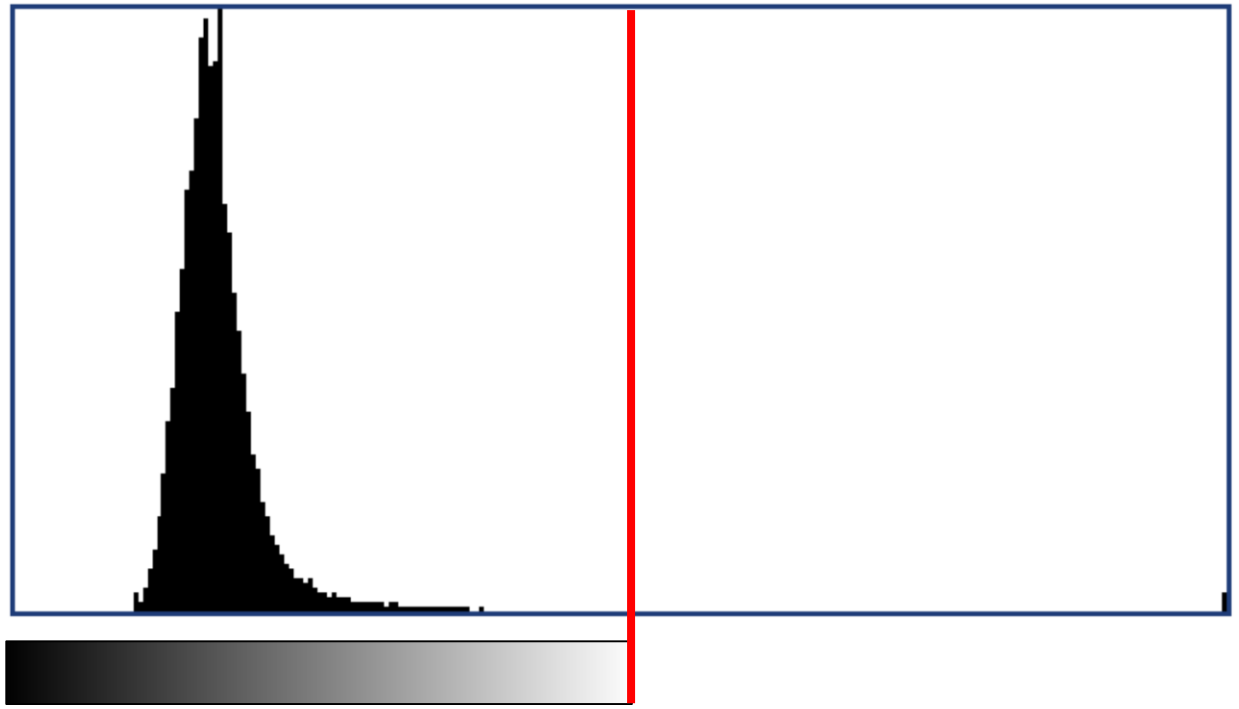
Visualisation: brightness / contrast

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?



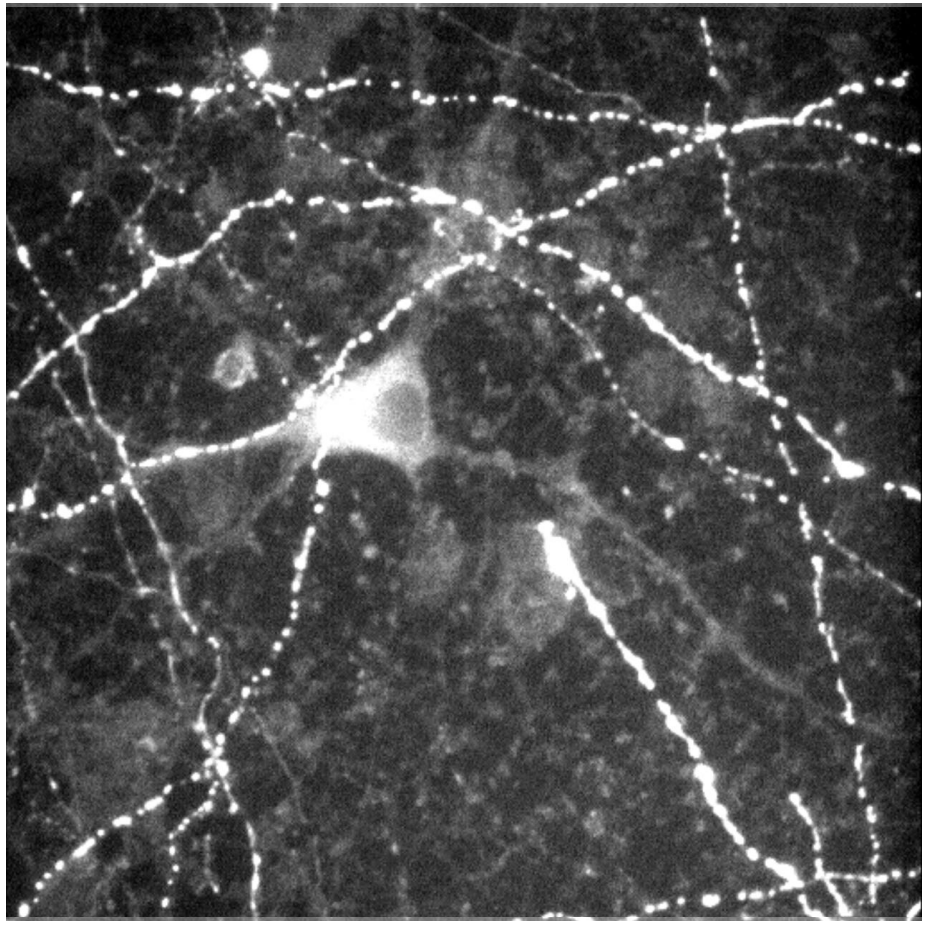
Decrease only vmax:
Contrast Increased



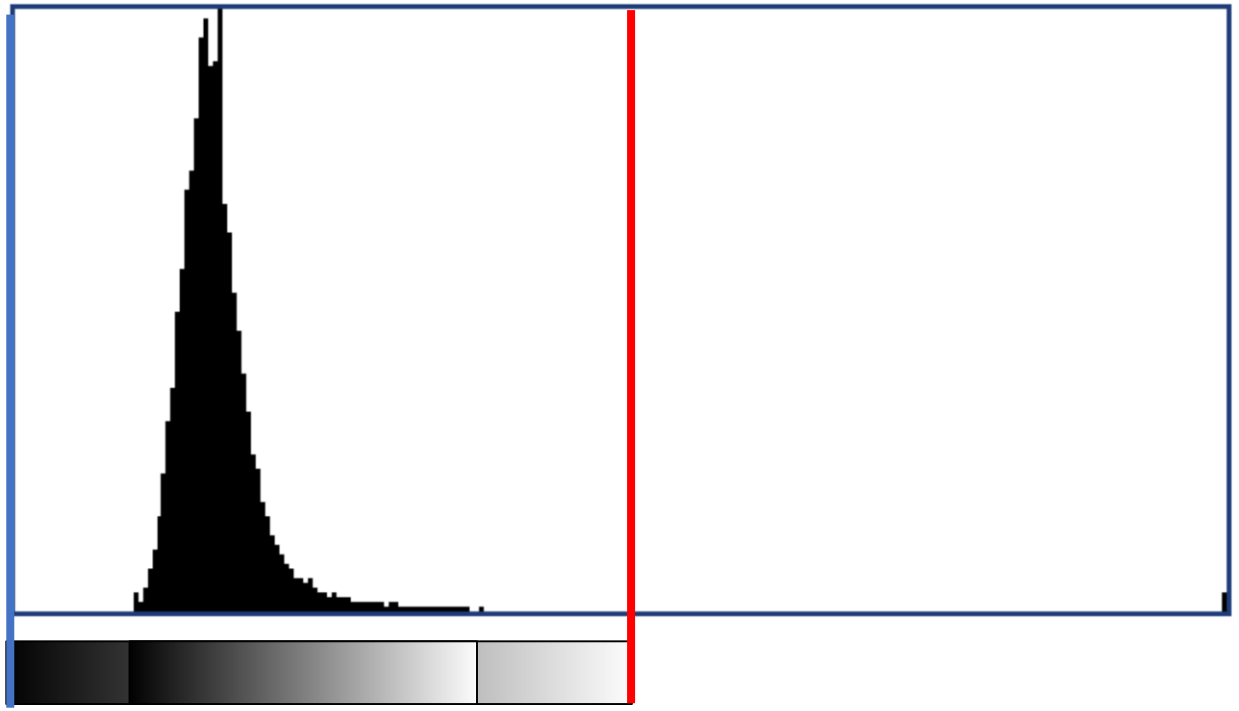
Visualisation: brightness / contrast

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?



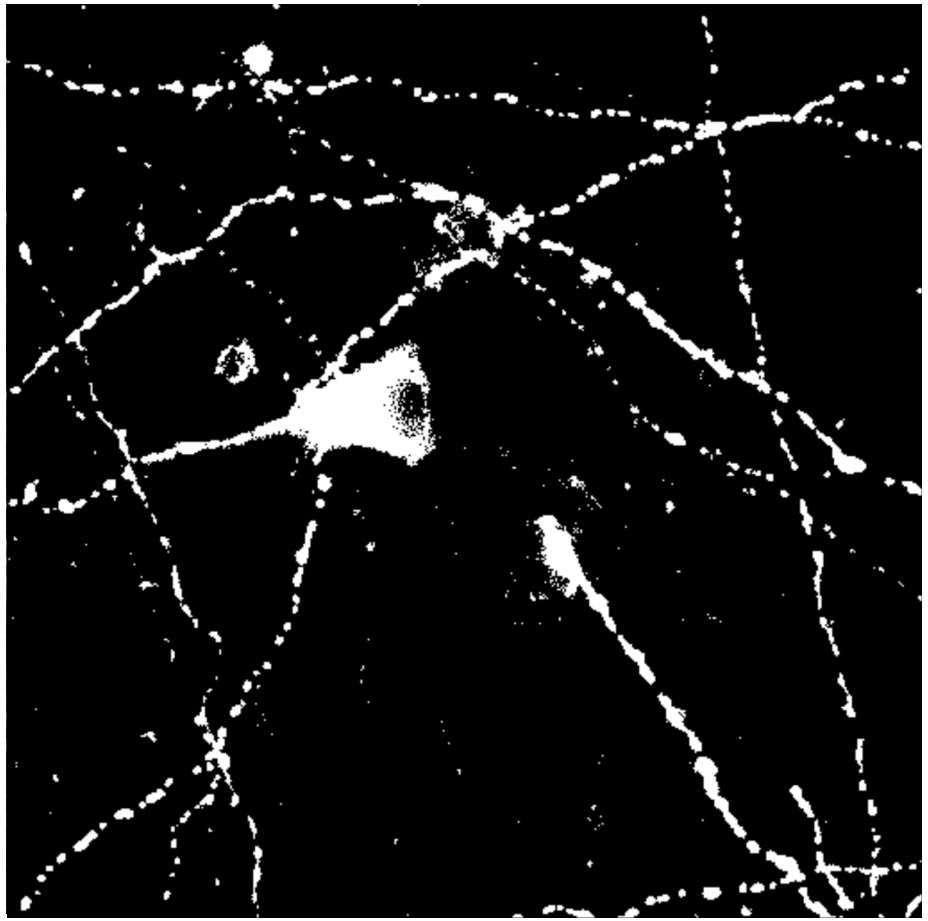
Window vmin and vmax around histogram:
Contrast Increased

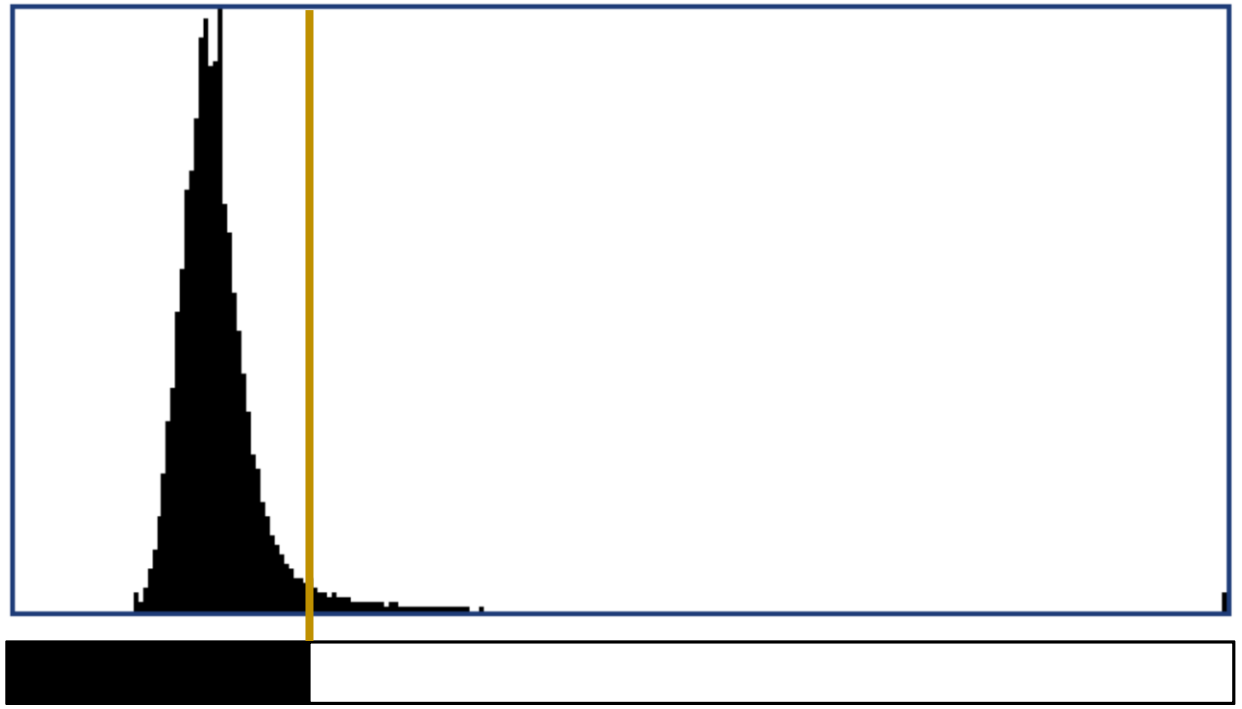


Don’t do such a narrow window right before acquisition!
If your signal changes over time/sample, you will lose information.

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)

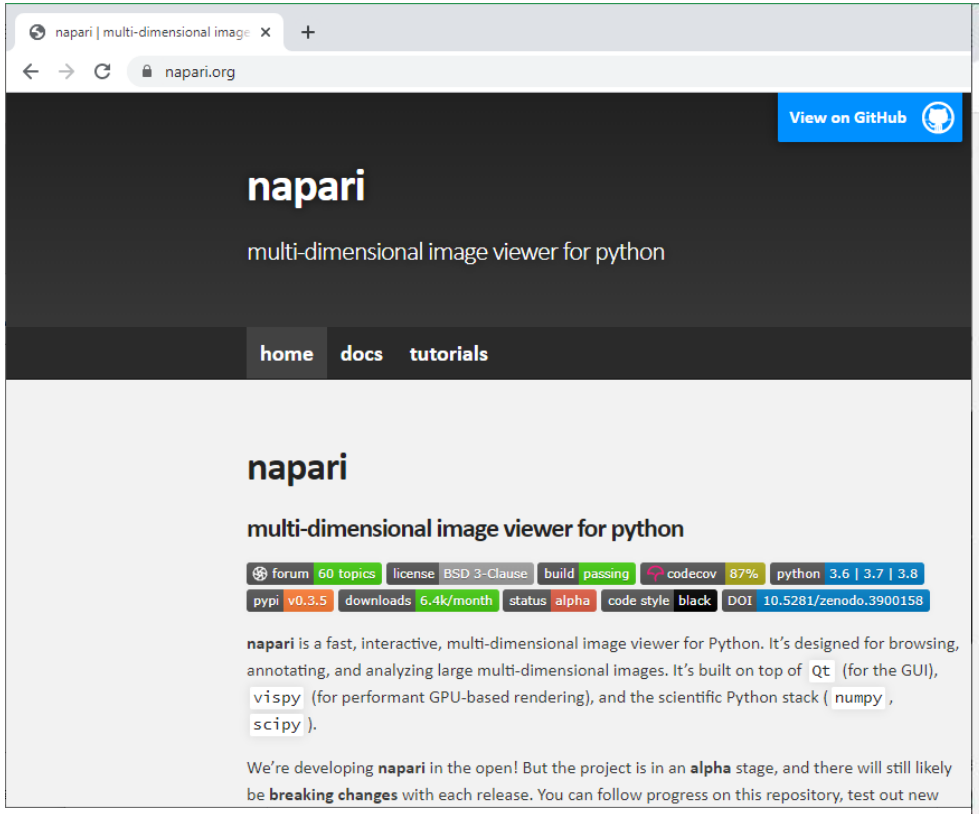


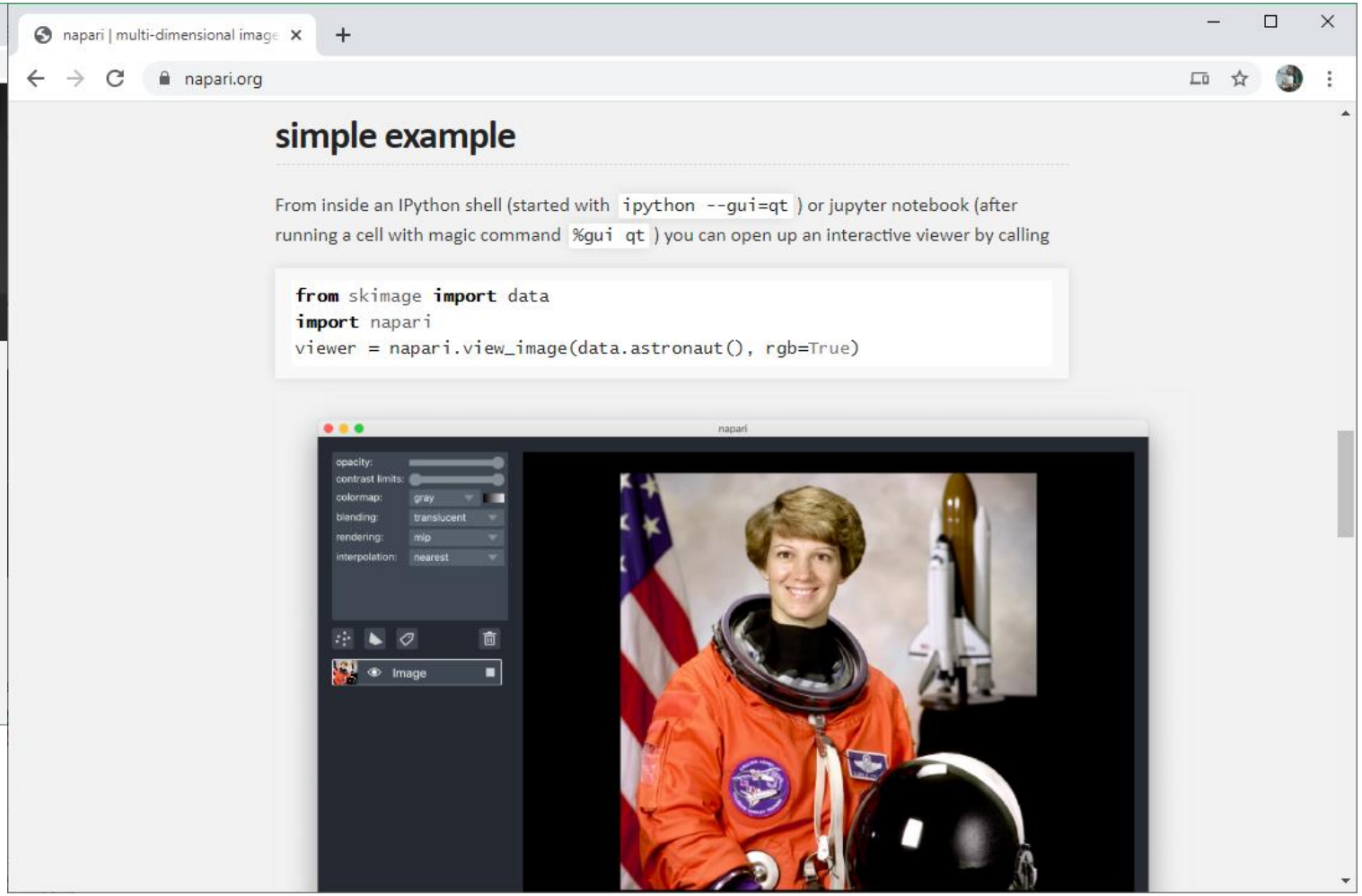


```
image_binary = image > value
```

napari

- Multi-dimensional image viewer in python
- https://napari.org/





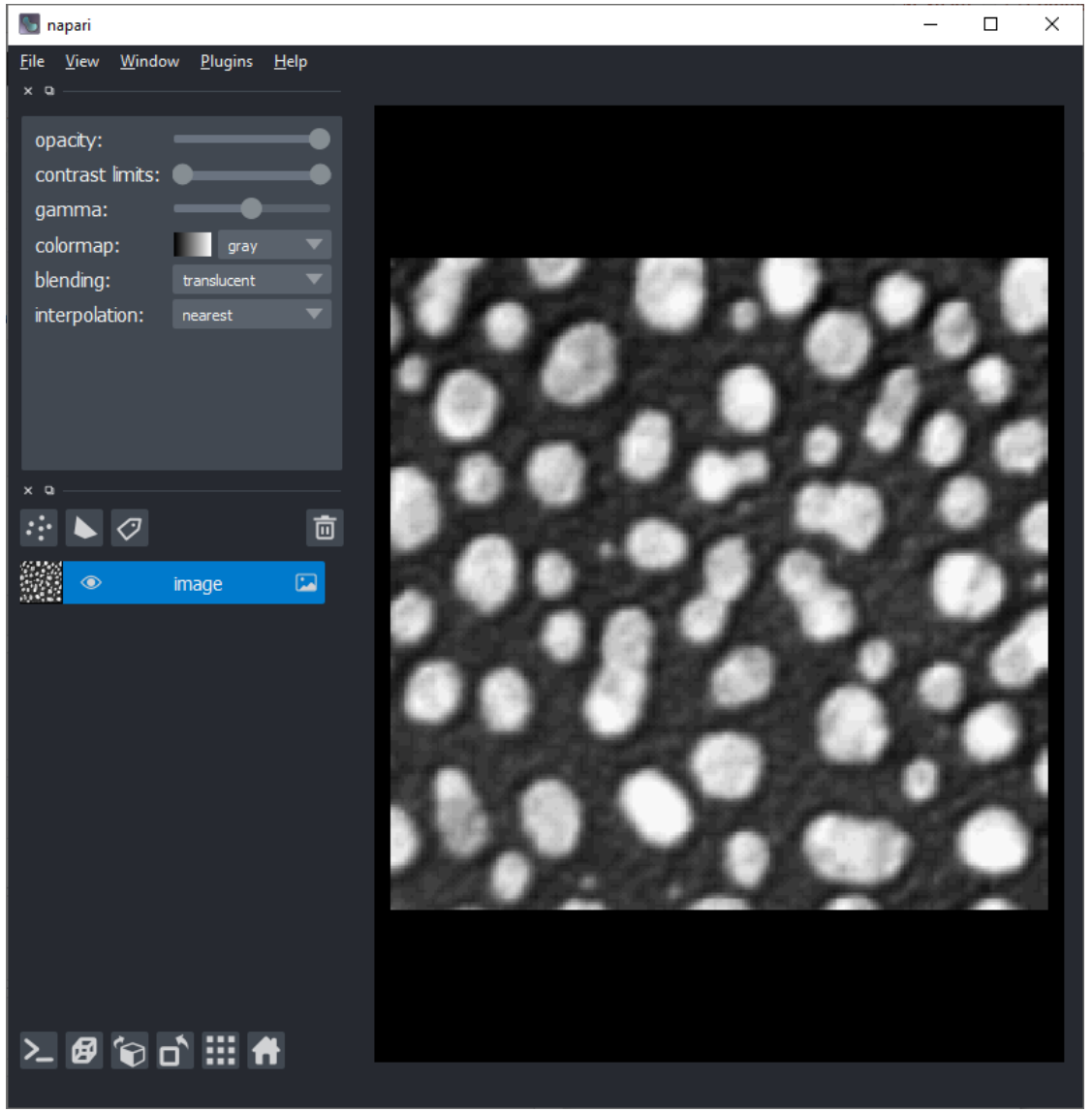
napari

- Open a window

```
import napari

# Create an empty viewer
viewer = napari.Viewer()
```

- Add an image (layer)

```
# Add a new layer containing an image
viewer.add_image(image)
```



Add multiple images as layers

- Add layers to napari to visualize intermediate processing results on top of each other or side by side.

- Change layer visualization within napari…

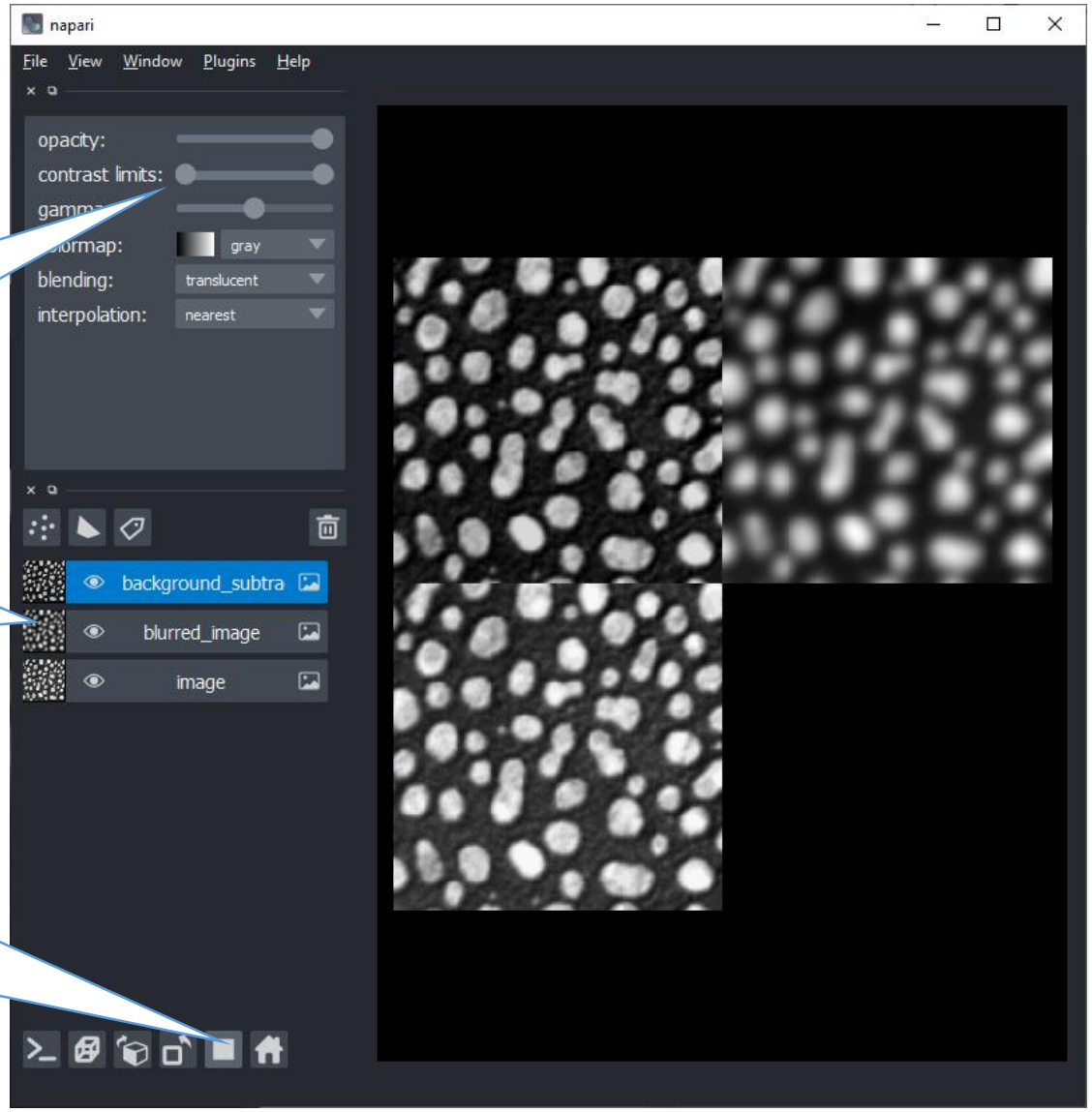
… or via code in a jupyter notebook:

```
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

Find more image layer attributes here:
https://napari.org/api/napari.layers.Image.html#napari.layers.Image

PoL
Physics of Life
TU Dresden

CENTER FOR
SYSTEMS BIOLOGY
DRESDEN

Image Processing: Filters

Marcelo Leomil Zoccoler

With material from

Robert Haase, PoL

Mauricio Rocha Martins, Norden lab, MPI CBG

April 2022

@haesleinhuepf @zoccolermarcelo

Filters

- An image processing filter is an operation on an image.
- It takes an image and produces a new image out of it.
- Filters change pixel values.

- There is no “best” filter. Which filter fits your needs, depends on the context.
- Filters do not do magic. They can not make things visible which are not in the image.

- Application examples
  - Noise-reduction
  - Artefact-removal
  - Contrast enhancement
  - Correct uneven illumination

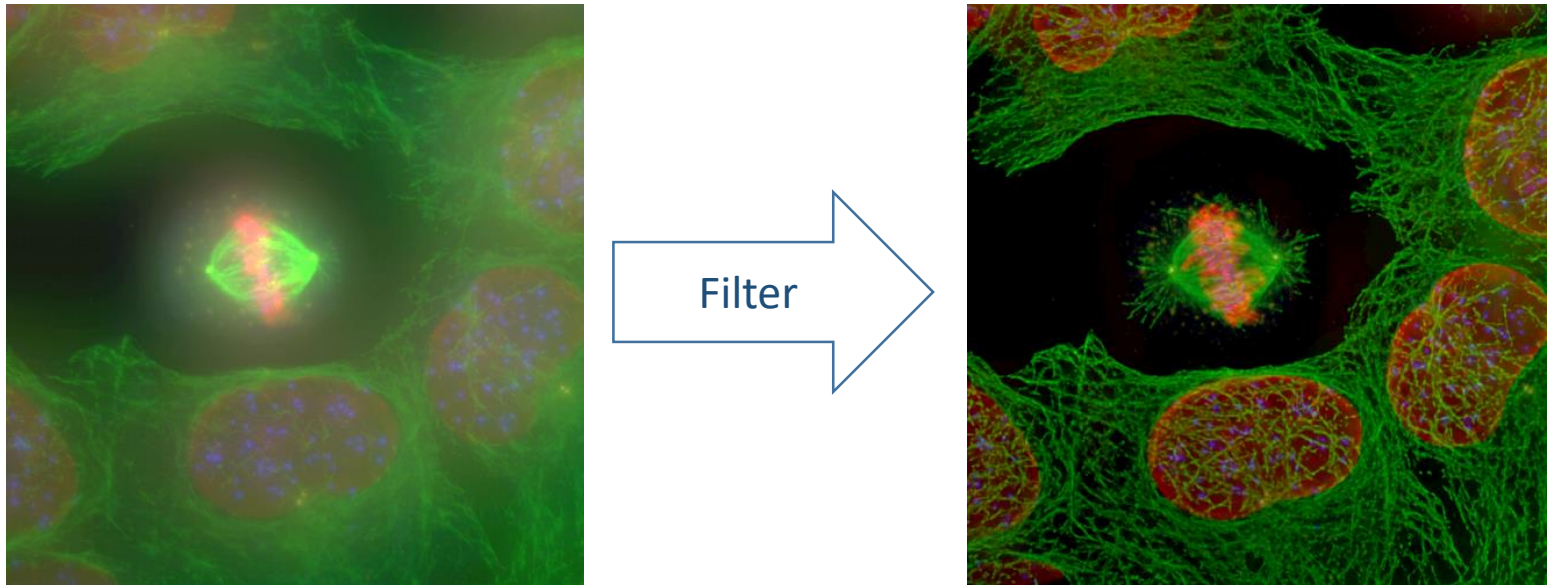


Image source: Alex Bird / Dan White MPI CBG

Effects harming image quality

- Noise is a general term for unwanted modifications that a signal may suffer during capture, storage, transmission, processing, or conversion.
- In microscopy, image quality suffers from
  - shot noise: Statistical variation of the photons arriving at the camera
  - dark noise: Statistical variation of how many electrons are generated if a photon arrives in a camera pixel (temperature dependent).
  - read out noise: introduced by the electronics, especially the analog-digital-converter
  - Physical/optical effects: aberrations, defocus
  - Biological/physiological/structural effects: motion, diffusion

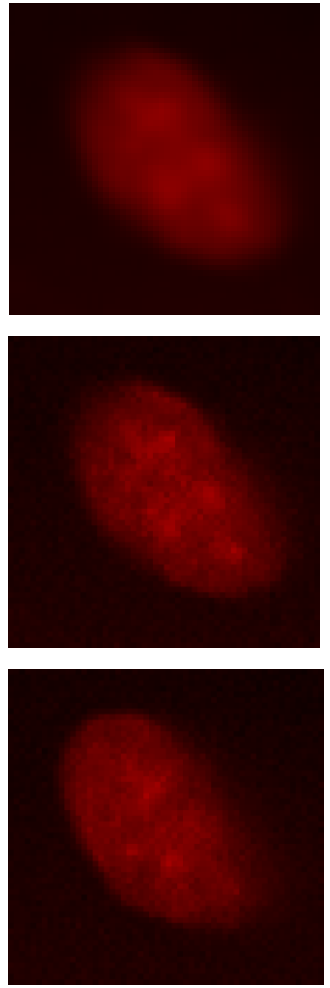


Image source: Mauricio Rocha Martins (Norden/Myers lab, MPI CBG)

Noise

- When dealing with noise in microscopic image processing, we mostly mean the noise visible in the images.

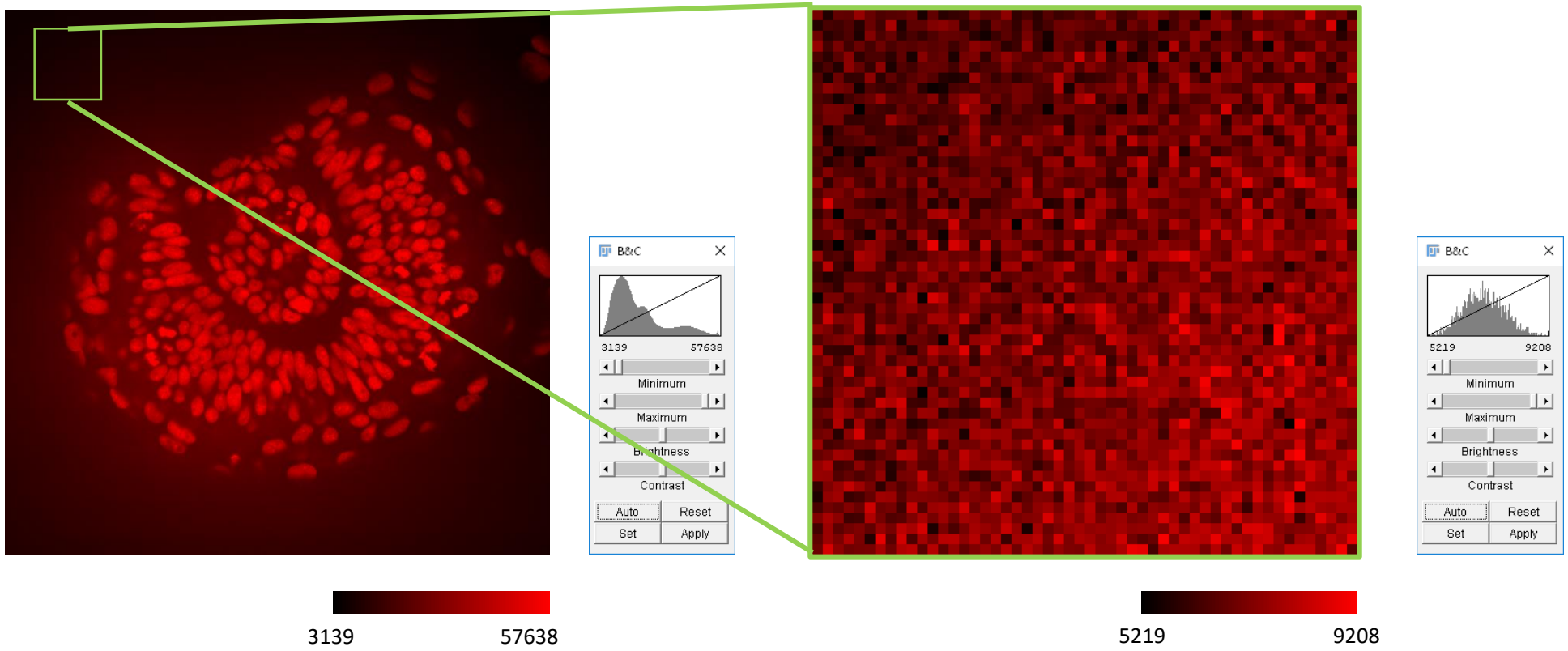


Image source: Mauricio Rocha Martins (Norden/Myers lab, MPI CBG)

Image correction / noise removal

- We need to remove the noise to help the computer *interpreting* the image

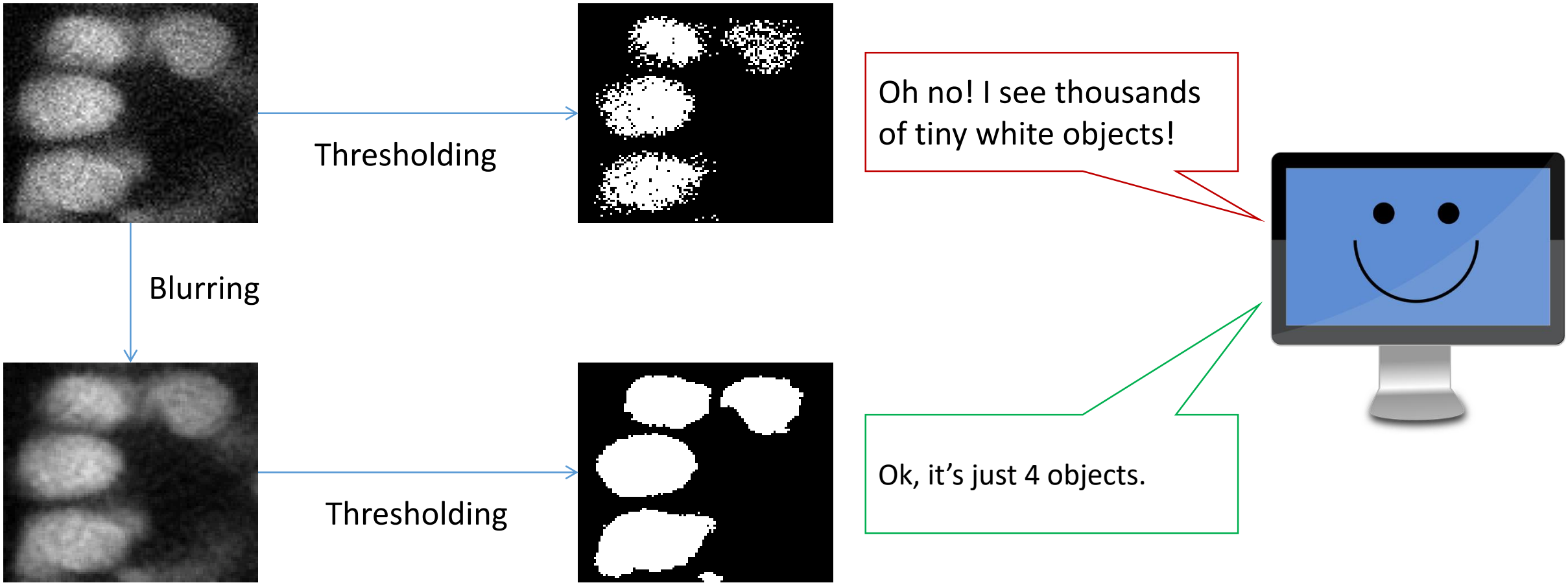
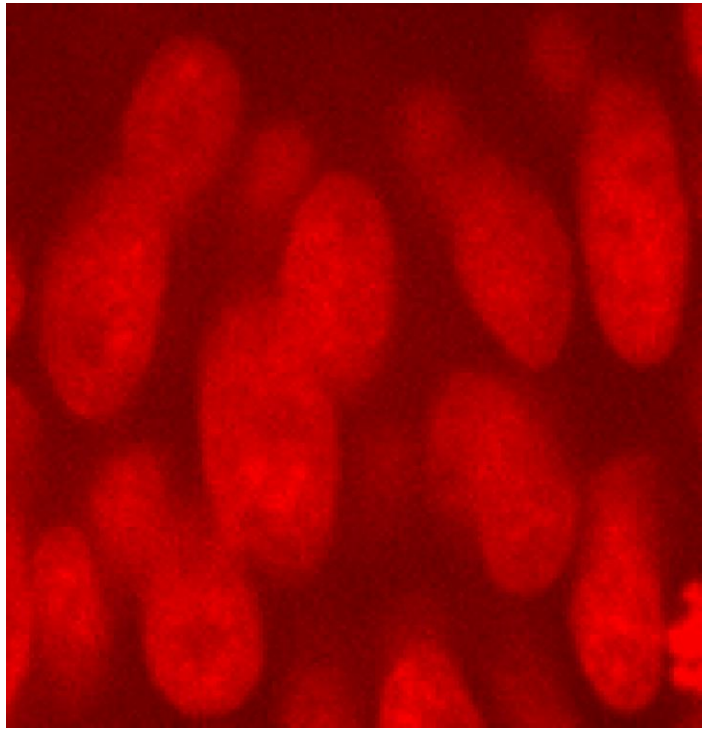
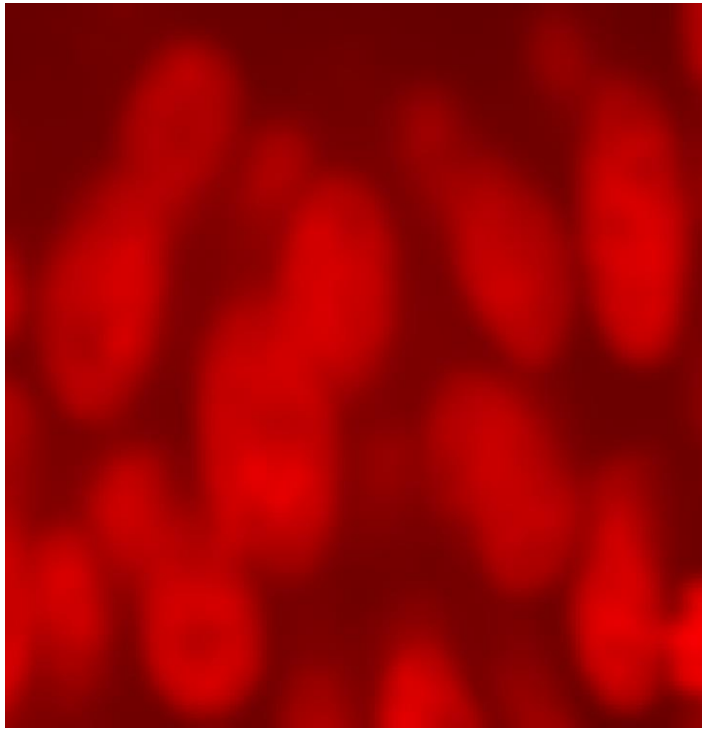


Image correction / noise removal

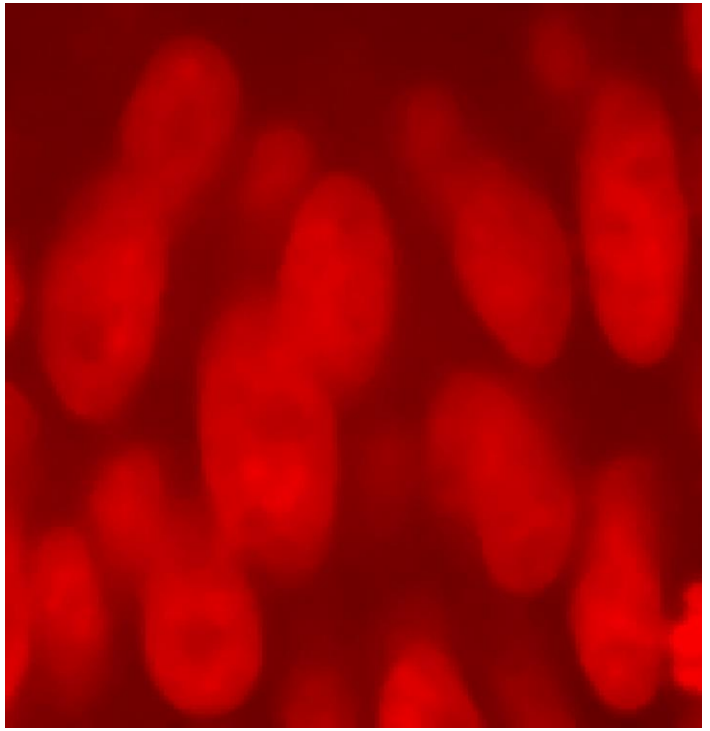
- Noise removal using *filters*



Original image



Gaussian blur filtered

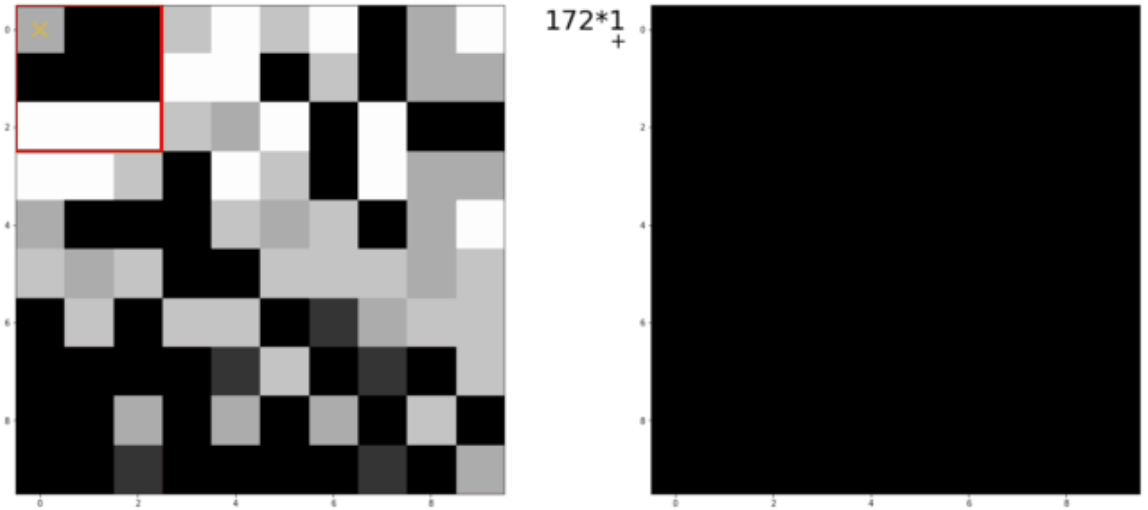


Median filtered

Image source: Mauricio Rocha Martins (Norden/Myers lab, MPI CBG)

Linear Filters

- Linear filters replace each pixel value with a linear combination of surrounding pixels
  - Basically, linear filtering is a Convolution
  - It needs a kernel (weight template)
  - Result: new image where each pixel is replaced by the weighted sum of pixel values in the neighbourhood.



- Kernels are matrices describing a linear filter

Mean filter, 3x3 kernel

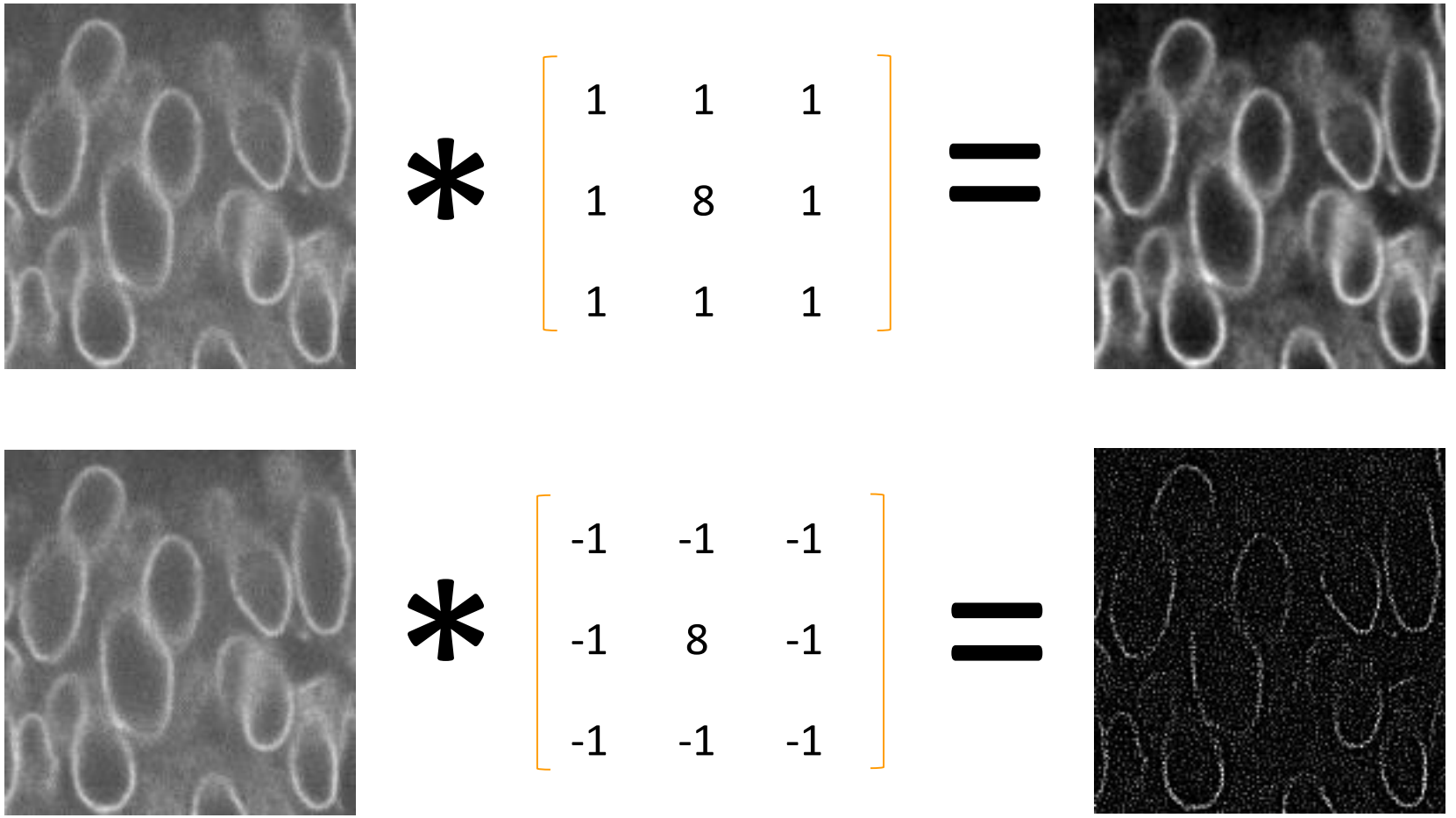
$$\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

Animation source: Dominic Waithe, Oxford University
https://github.com/dwaithe/generalMacros/tree/master/convolution_ani

Linear filters

- Terminology:
  - “We convolve an image with a kernel.”
  - Convolution operator: *

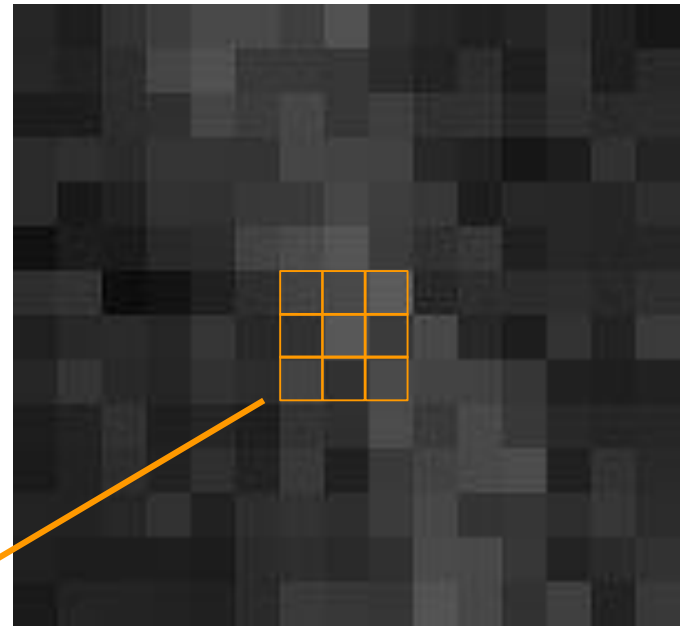
- Examples
  - Mean
  - Gaussian blur
  - Sobel-operator
  - Laplace-filter

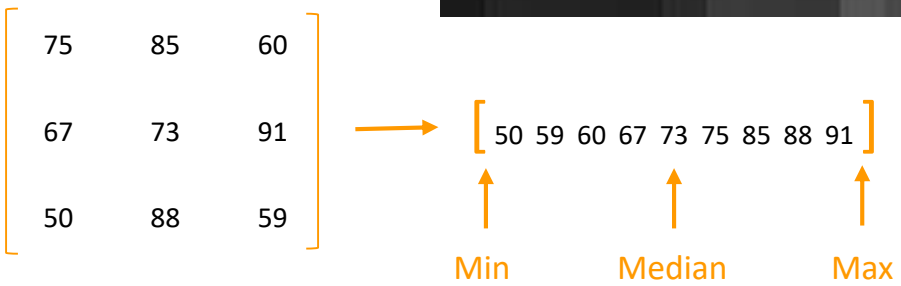


Nonlinear Filters

- Non linear filters also replace pixel value inside as rolling window but in a non linear function.

- Examples: order statistics filters
  - Min
  - Median
  - Max
  - Variance
  - Standard deviation





Filters

In python, for linear filters, we could apply a kernel to an image like this:

```
kernel = np.array(
    [[1/9, 1/9, 1/9],
     [1/9, 1/9, 1/9],
     [1/9, 1/9, 1/9]])
```

```
from scipy.ndimage import convolve

output = convolve(image, kernel)
```

But scikit-image already has several of these filters implemented and optimized.

```
from skimage.filters import gaussian

output = gaussian(image, sigma = 1)
```

Then there is no need to provide a kernel. And you can also directly apply non-linear filters.

```
from skimage.filters import median

output = median(image)
```

Available filters can be searched like this:

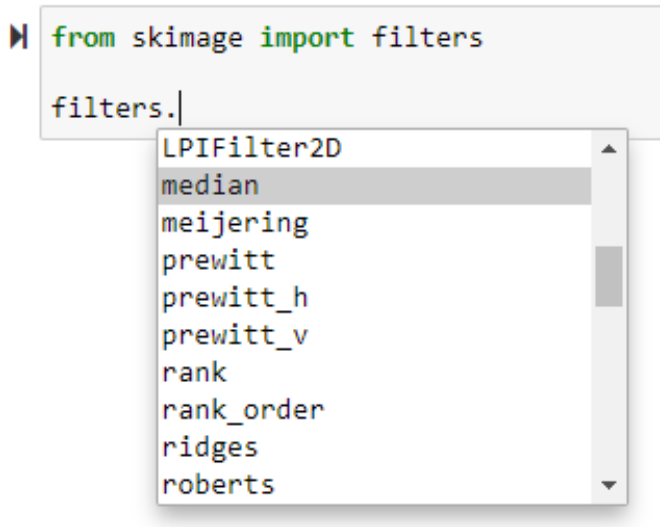


Image Processing: Morphological Operations

Marcelo Leomil Zoccoler

With material from

Robert Haase, PoL

Mauricio Rocha Martins, Norden lab, MPI CBG

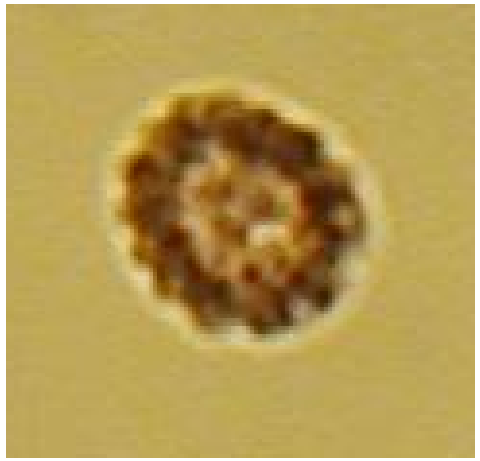
April 2022

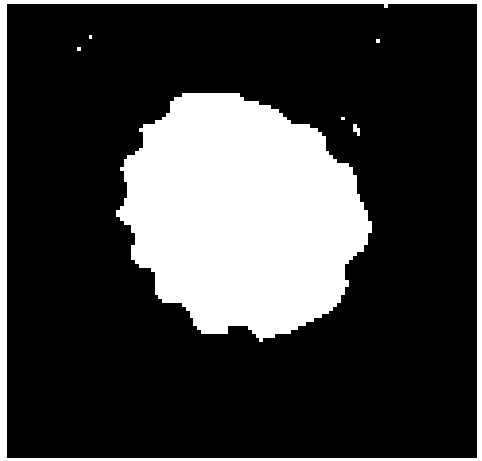
@haesleinhuepf @zoccolermarcelo

Refining masks

- Binary mask images may not be perfect immediately after thresholding.
- There are ways of refining them

Thresholding → Binary closing (Fill holes) → Binary opening






Background removal

- Differentiating objects is easier if their background intensity is equal.

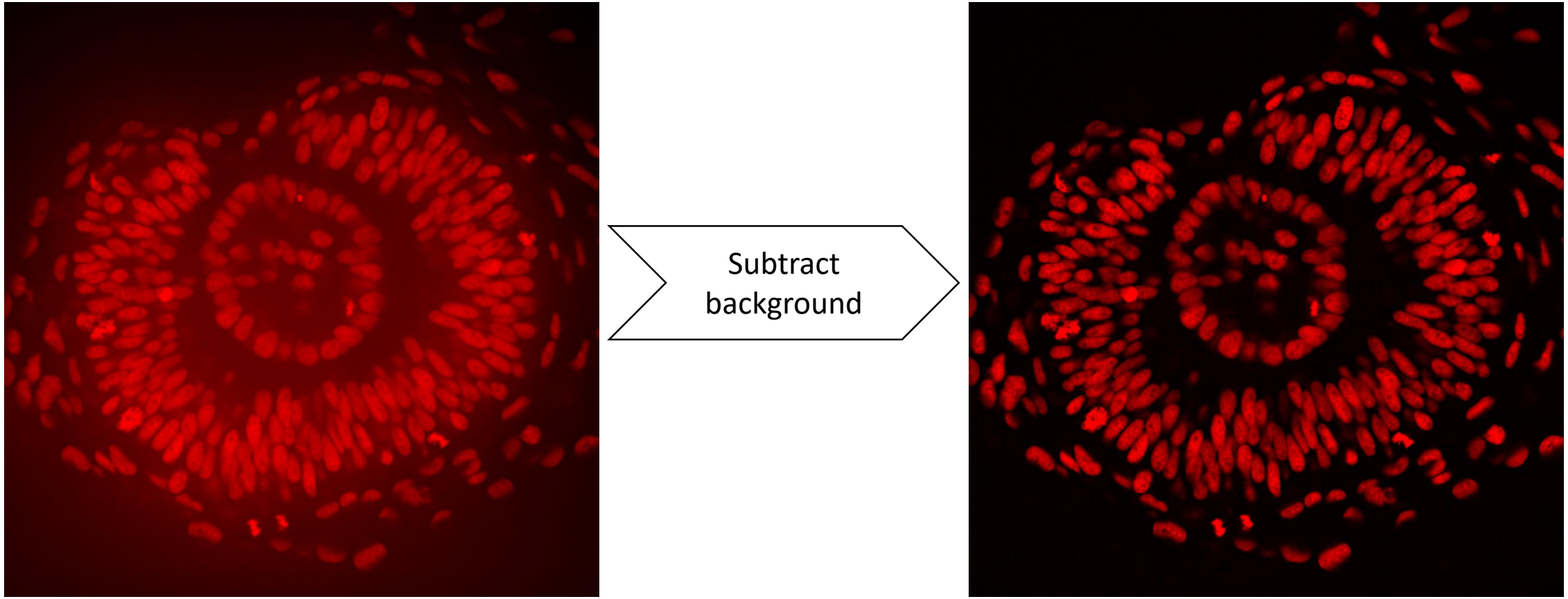
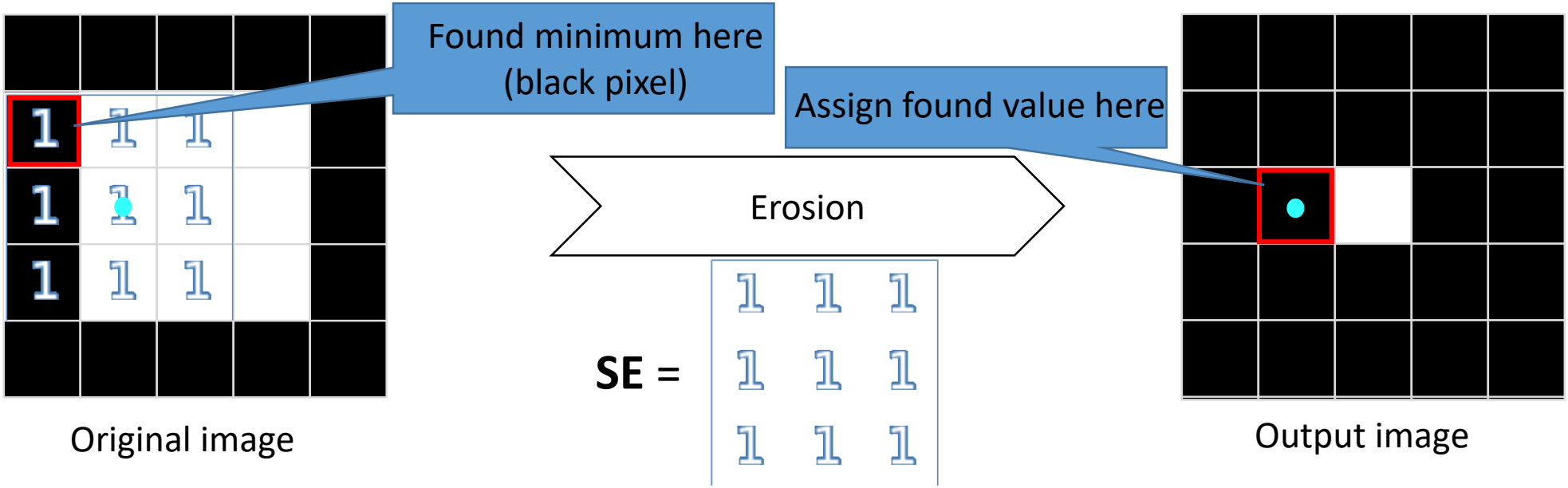


Image source: Mauricio Rocha Martins (Norden/Myers lab, MPI CBG)

Refining masks: Erosion

- Erosion: Every pixel with at least one black neighbor becomes black.



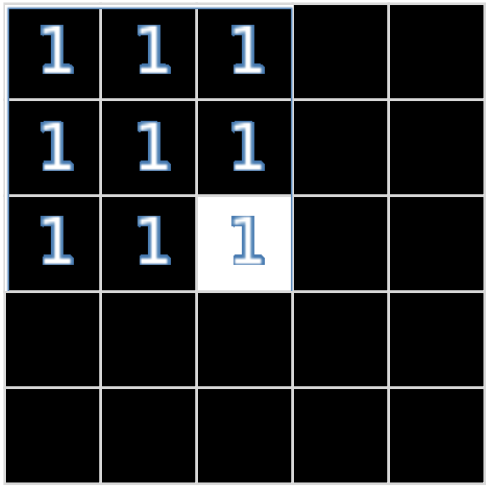
Erosion is essentially a minimum filter whose extent is defined by the kernel (structural element or **SE**) size and shape.

Dilation is essentially a maximum filter whose extent is defined by the kernel (structural element or **SE**) size and shape.

For an example with a grayscale image, check out this gif: https://en.wikipedia.org/wiki/Erosion_(morphology)#/media/File:Grayscale_Morphological_Erosion.gif

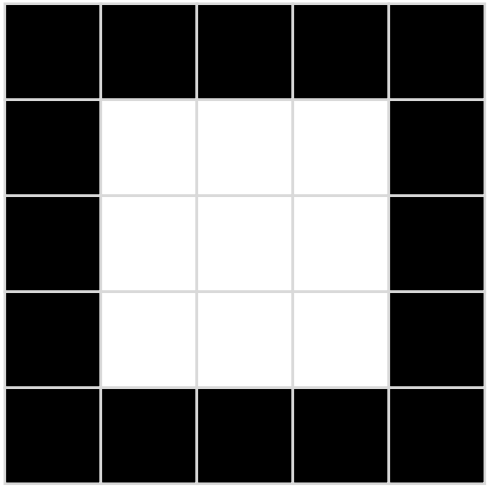
Refining masks: Dilation

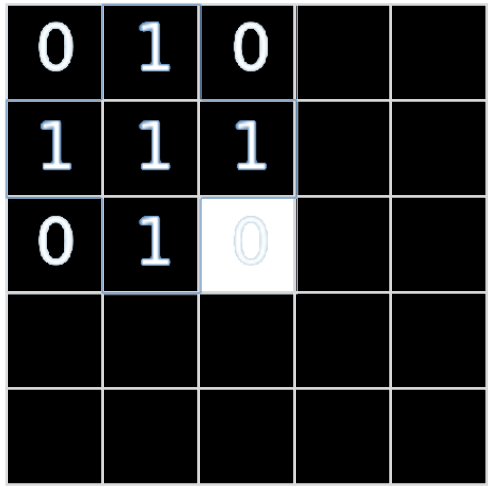
- Dilation: Every pixel with at least one white neighbor becomes white.



Dilation

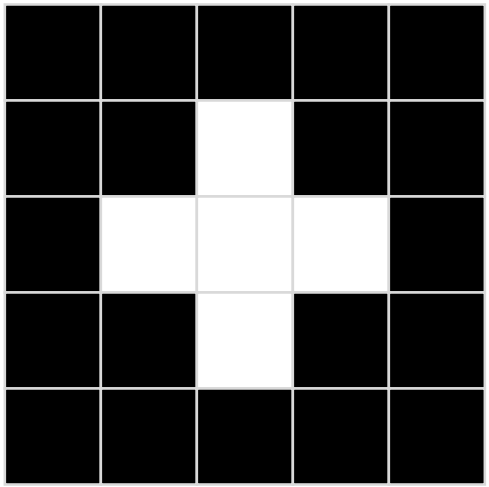
8-connected neighborhood
Moore-Neighborhood





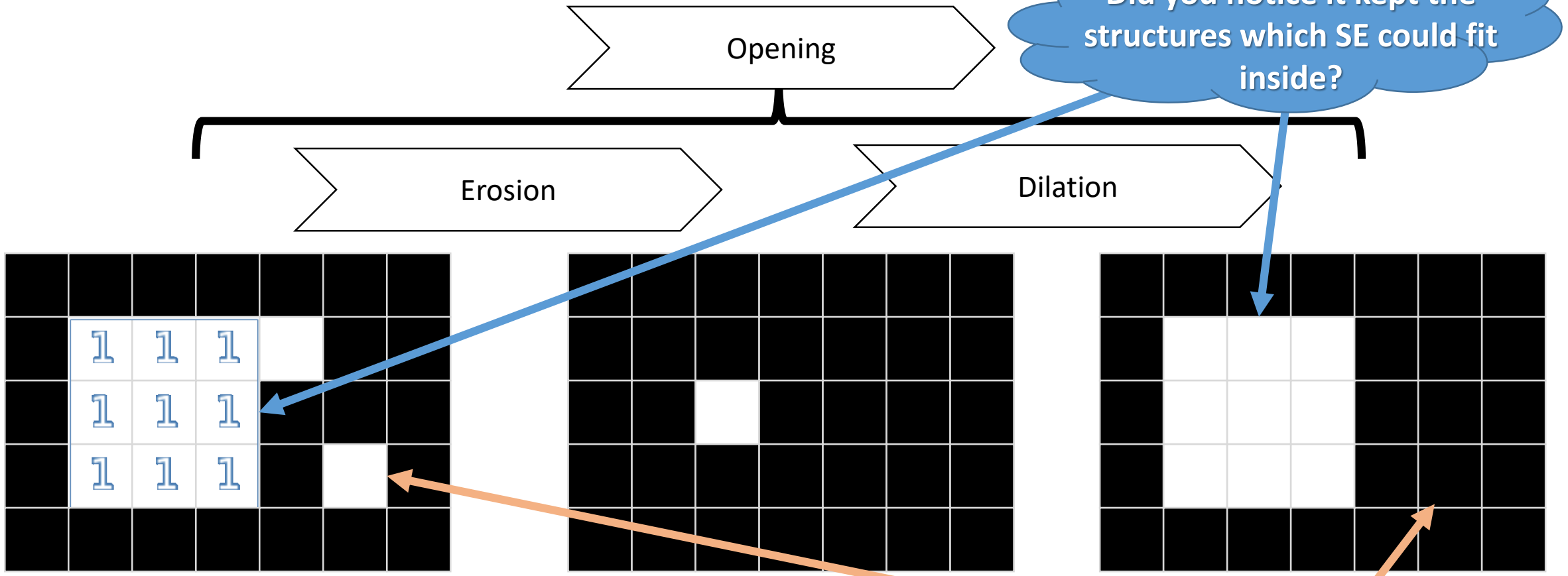
Dilation

4-connected neighborhood
von-Neumann-Neighborhood



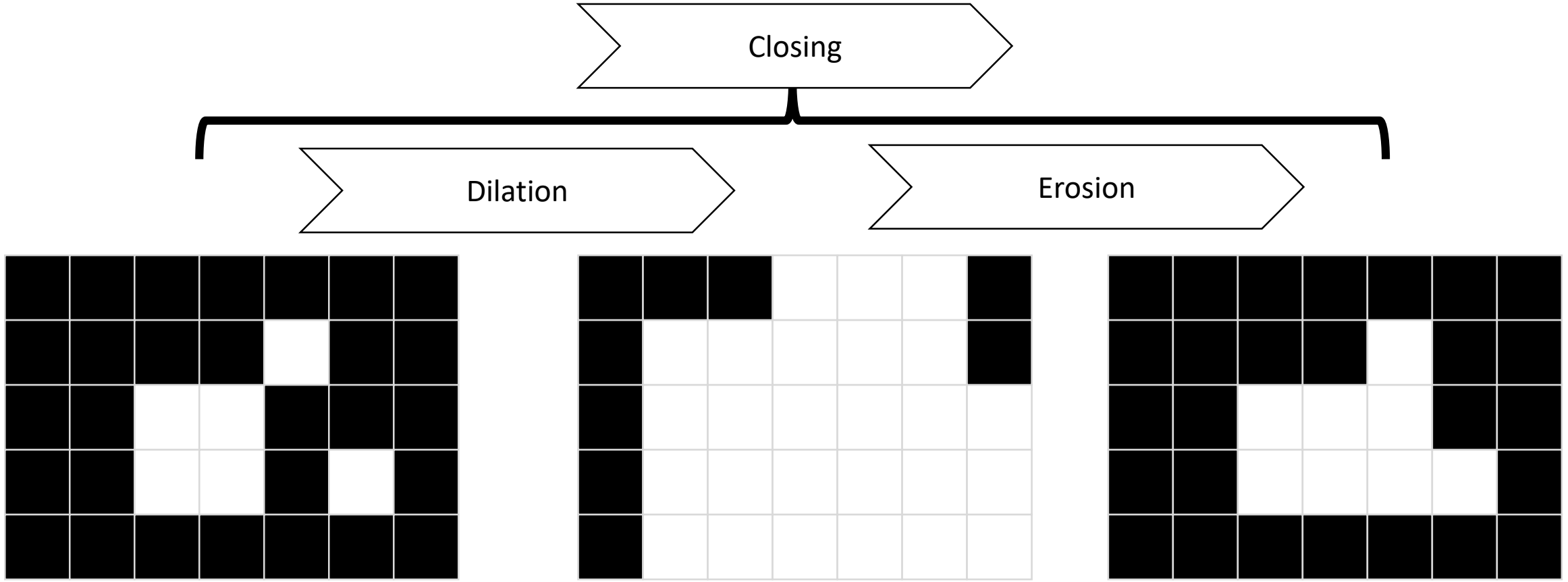
Refining masks: Erosion & Dilation

- Erosion and dilation combined allow correcting outlines.



- It can separate white (high intensity) structures that are weakly connected
- It may erase small white structures
- It tends to better preserve area of structures

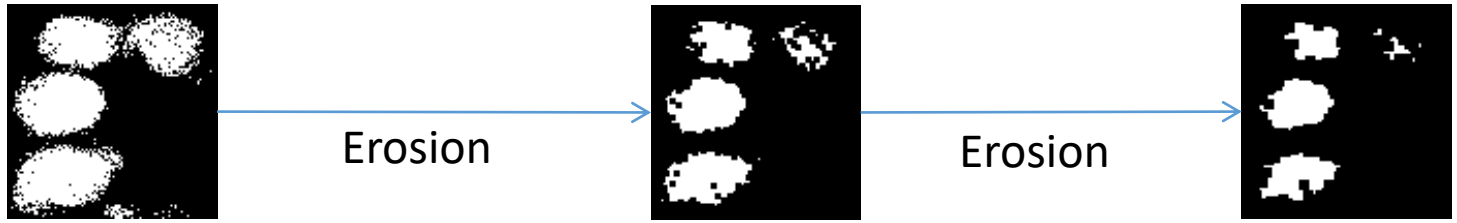
Refining masks: Erosion & Dilation



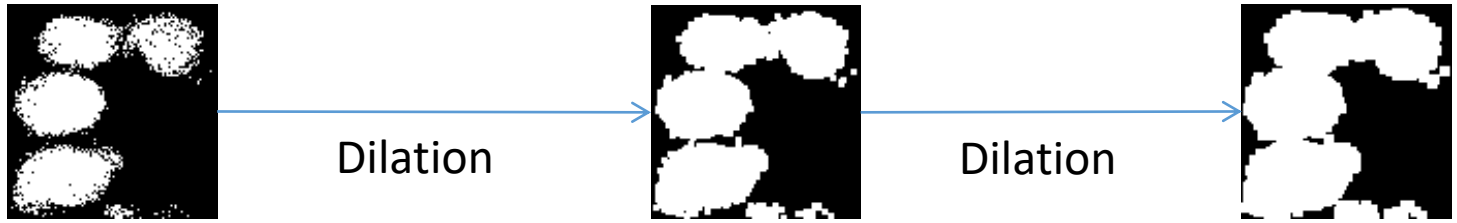
- It can connect white (high intensity) structures that are nearby
- It may close small holes inside structures
- It tends to better preserve area of structures

Refining masks: Erosion / dilation

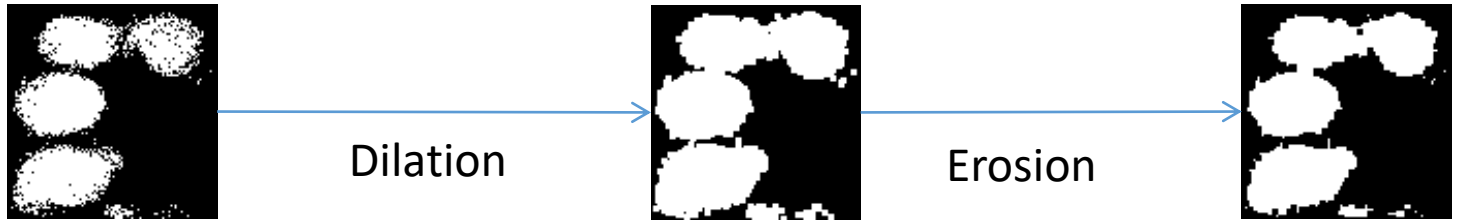
- Erosion: Set all pixels to black which have at least one black neighbor.



- Dilation: Set all pixels to white which have at least one white neighbor.



- Closing: Dilation + Erosion



- Opening: Erosion + Dilation

Morphological Operations in Python

Scikit-image has a sub-package called morphology

```
from skimage import morphology
```

You must define a SE first (also called footprint):

```
SE = morphology.square(3)
SE
```

```
array([[1, 1, 1],
       [1, 1, 1],
       [1, 1, 1]], dtype=uint8)
```

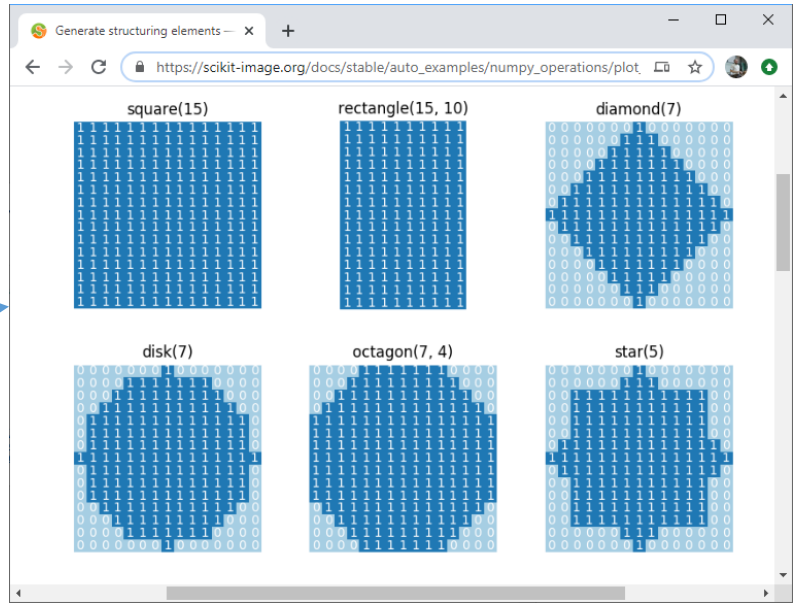
For a binary image, you apply it like this:

```
output = morphology.binary_dilation(binary_image, SE)
```

For a grayscale image, you apply it like this:

```
output = morphology.dilation(image, SE)
```

It can have other shapes/sizes:



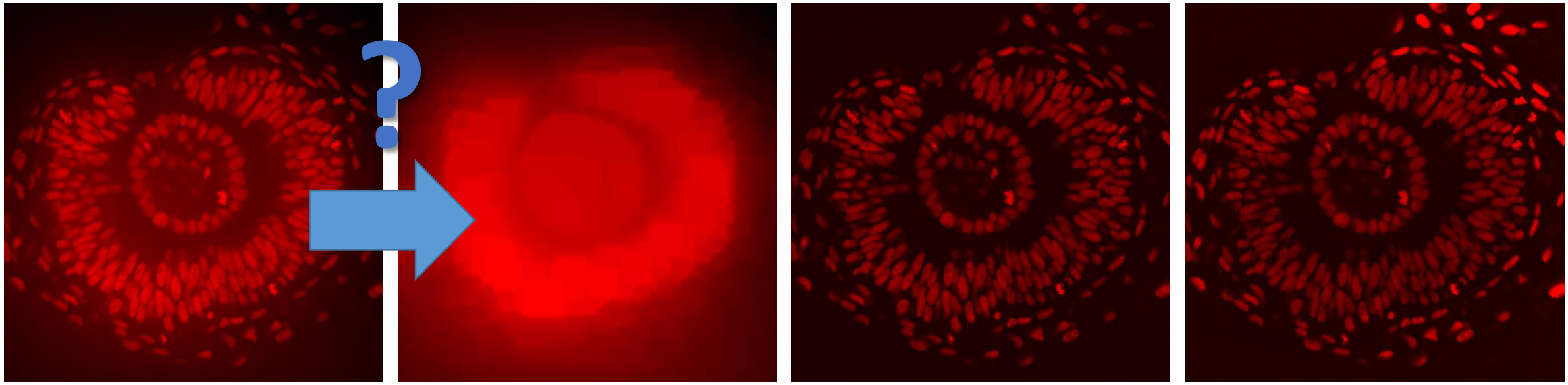
https://scikit-image.org/docs/stable/auto_examples/numpy_operations/plot_structuring_elements.html#sphx-glr-auto-examples-numpy-operations-plot-structuring-elements-py

Background removal

- Depending on the effect we want to correct for, it might make sense to divide an image by its background.

Do you remember that opening kept the structures which the SE could fit inside?

And do you remember that opening deletes structures smaller than SE?

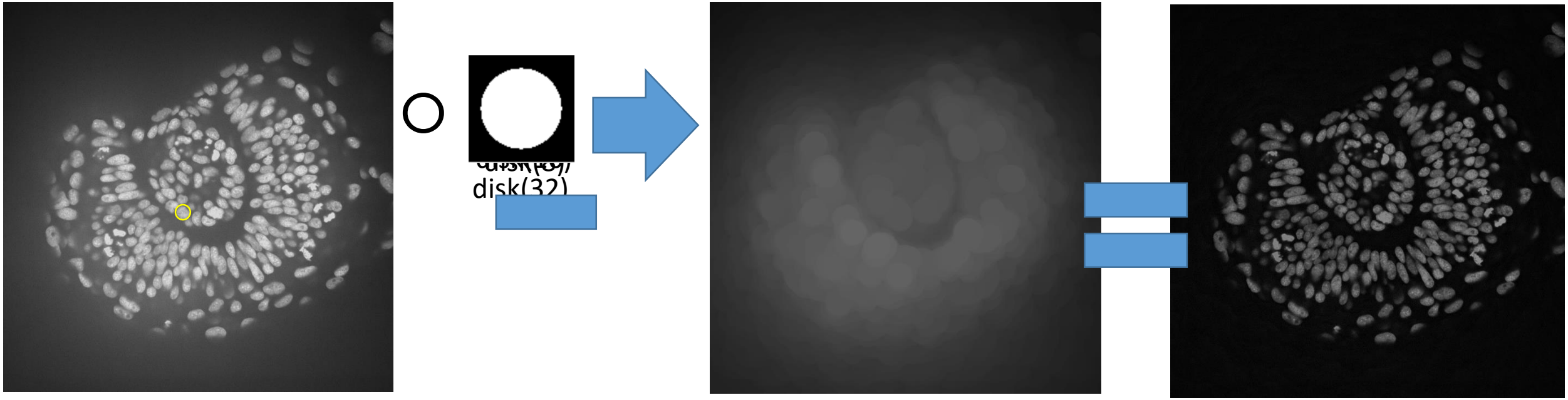


Original image | Extracted background | Subtracted background | Image divided by background

Image source: Mauricio Rocha Martins (Norden/Myers lab, MPI CBG)

Background removal

Opening



Original image

This is a good estimation of the background

Strucutres have a radius ≈ 12

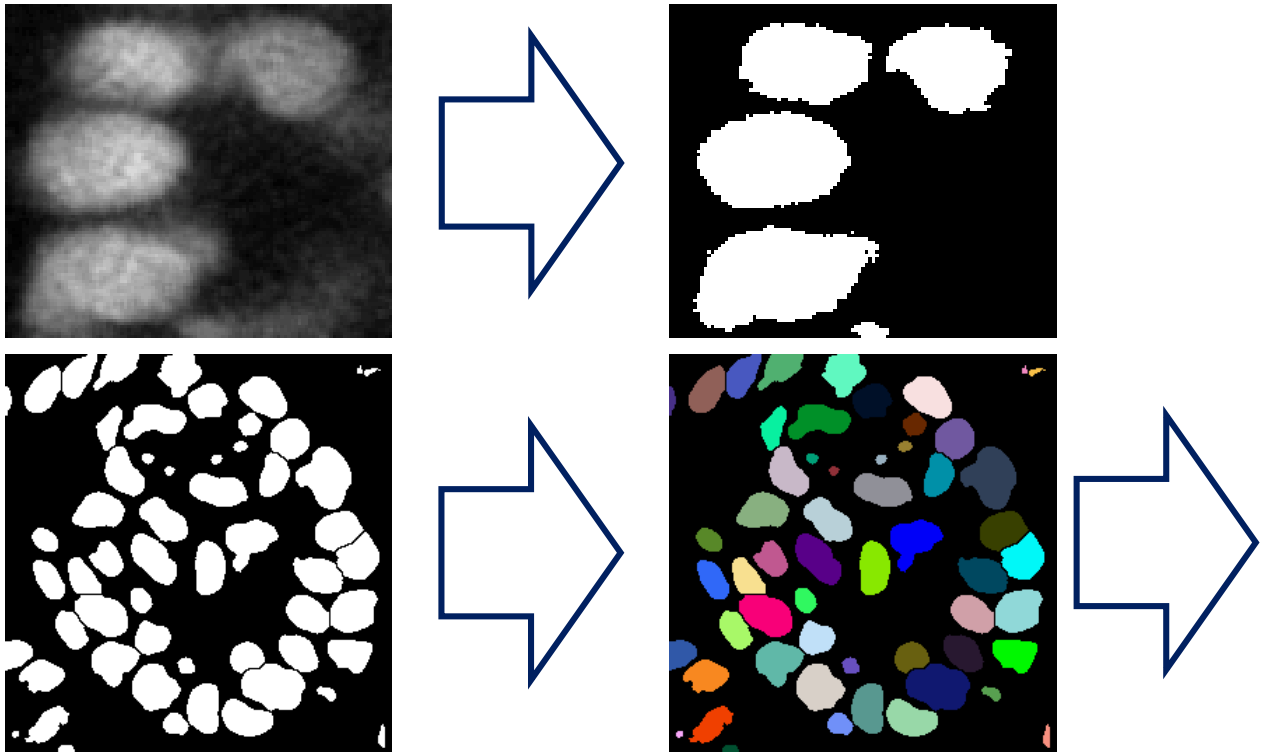
You have just learned the white tophat filter!

Summary

- Image visualization
  - Pixel size, colormaps, bit-depth
  - Image histogram
  - Brightness/Contrast
  - Binarization
  - Napari
- Image Filtering
  - Noise Removal
  - Edge Detection
- Morphological Operations
  - Mask Refinement
  - Background subtraction
- Tools
  - Python
    - Scikit-image
    - Napari

Coming up next

- Image Segmentation
  - Thresholding
  - Connected component analysis
  - Feature extraction



Results

	Area	X	Y
1	658	108.8298	13.0684
2	477	247.3092	14.0451
3	501	174.4381	33.6916
4	660	73.8045	41.7788
5	448	138.9688	55.2455
6	520	28.8481	57.9981
7	506	8.3617	98.0217
8	676	175.2855	99.5828
9	545	37.5569	122.7312
10	593	224.4410	127.8558

Extra: Image visualization: subplots with matplotlib

PoL Physics of Life TU Dresden

- Use matplotlib to put images side-by-side

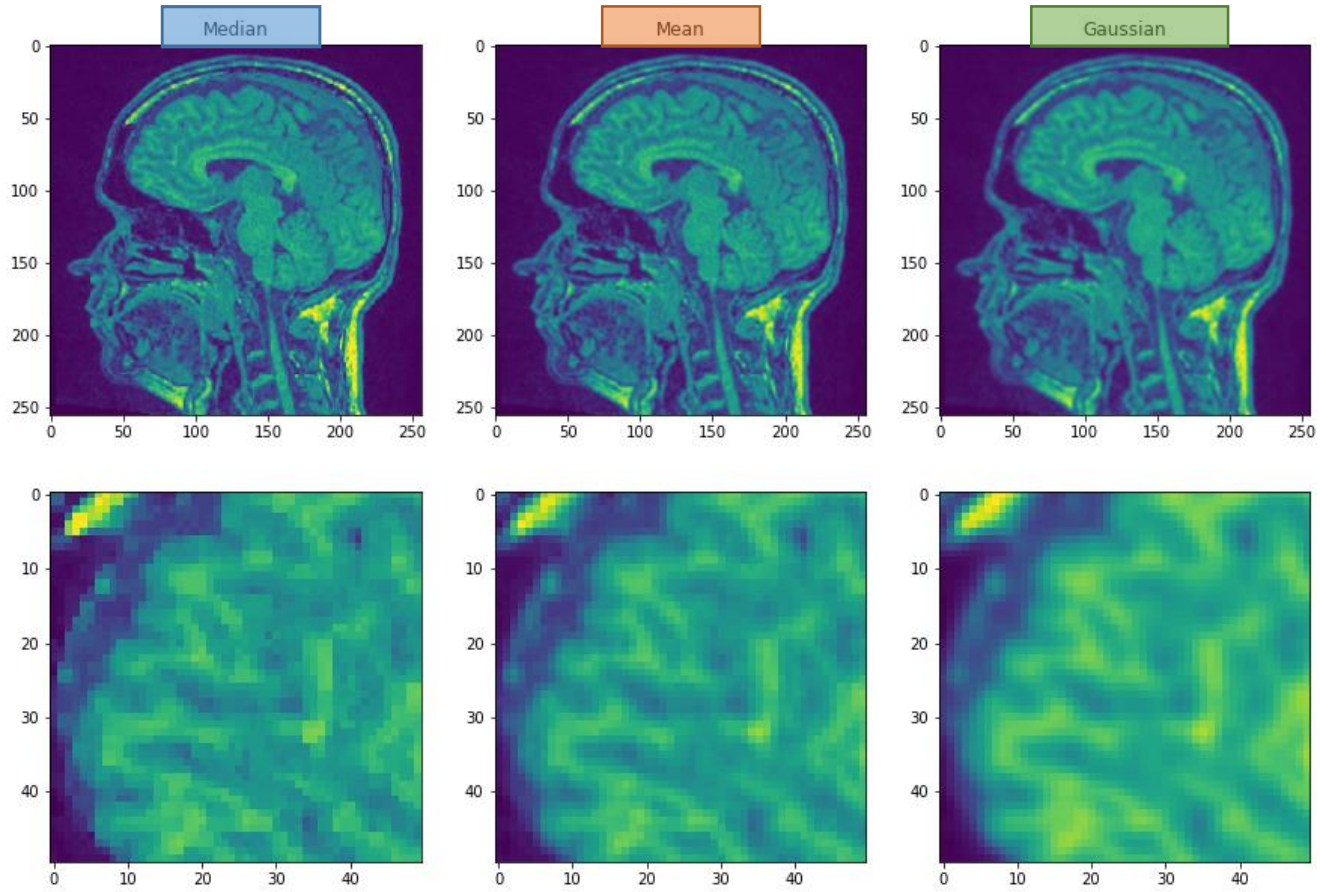
```
median_filtered = filters.median(noisy_mri, disk(1))
mean_filtered = filters.rank.mean(noisy_mri, disk(1))
gaussian_filtered = filters.gaussian(noisy_mri, sigma=1)

fig, axs = plt.subplots(2, 3, figsize=(15,10))

# first row
axs[0, 0].imshow(median_filtered)
axs[0, 0].set_title("Median")
axs[0, 1].imshow(mean_filtered)
axs[0, 1].set_title("Mean")
axs[0, 2].imshow(gaussian_filtered)
axs[0, 2].set_title("Gaussian")

# second row
axs[1, 0].imshow(median_filtered[50:100, 50:100])
axs[1, 1].imshow(mean_filtered[50:100, 50:100])
axs[1, 2].imshow(gaussian_filtered[50:100, 50:100])
```

rows

columns

row

column



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