



DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS  
AND ARTIFICIAL INTELLIGENCE

# Microscopy Image Processing

Robert Haase

Reusing materials from Mauricio Rocha Martins (Norden lab,  
MPI CBG); Dominic Waithe (Oxford University); Alex Bird, Dan  
White (MPI CBG), Marcelo Leomil Zoccoler, TU Dresden

GEFÖRDERT VOM



Bundesministerium  
für Bildung  
und Forschung



Diese Maßnahme wird gefördert durch die Bundesregierung  
aufgrund eines Beschlusses des Deutschen Bundestages.  
Diese Maßnahme wird mitfinanziert durch Steuermittel auf  
der Grundlage des von den Abgeordneten des Sächsischen  
Landtags beschlossenen Haushaltes.

# Recap quiz

- We write good documentation to enable others to re-do our experiment. This is good for ...

Repeatability



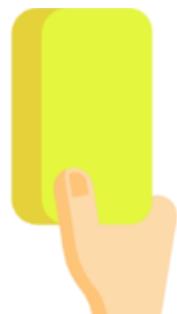
Reproducibility



Replicability

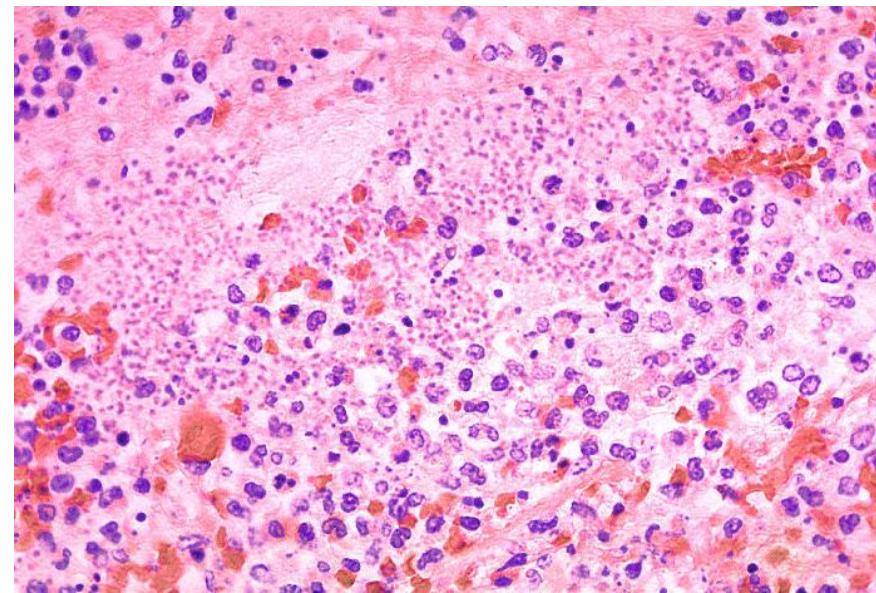


Reliability



# Recap quiz

- This image is from a ...



Transmitted light microscope



Fluorescence microscope

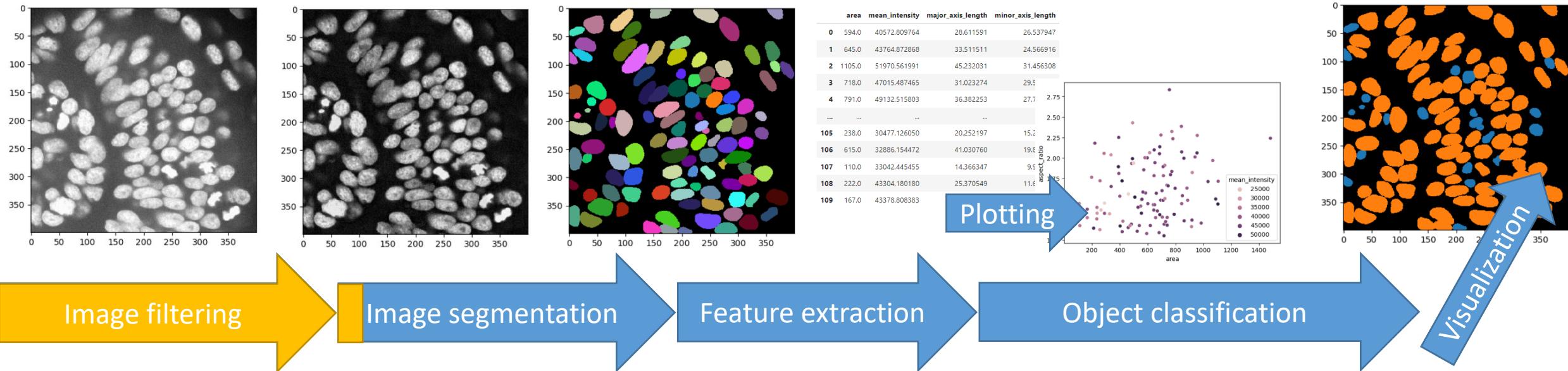


Electron microscope



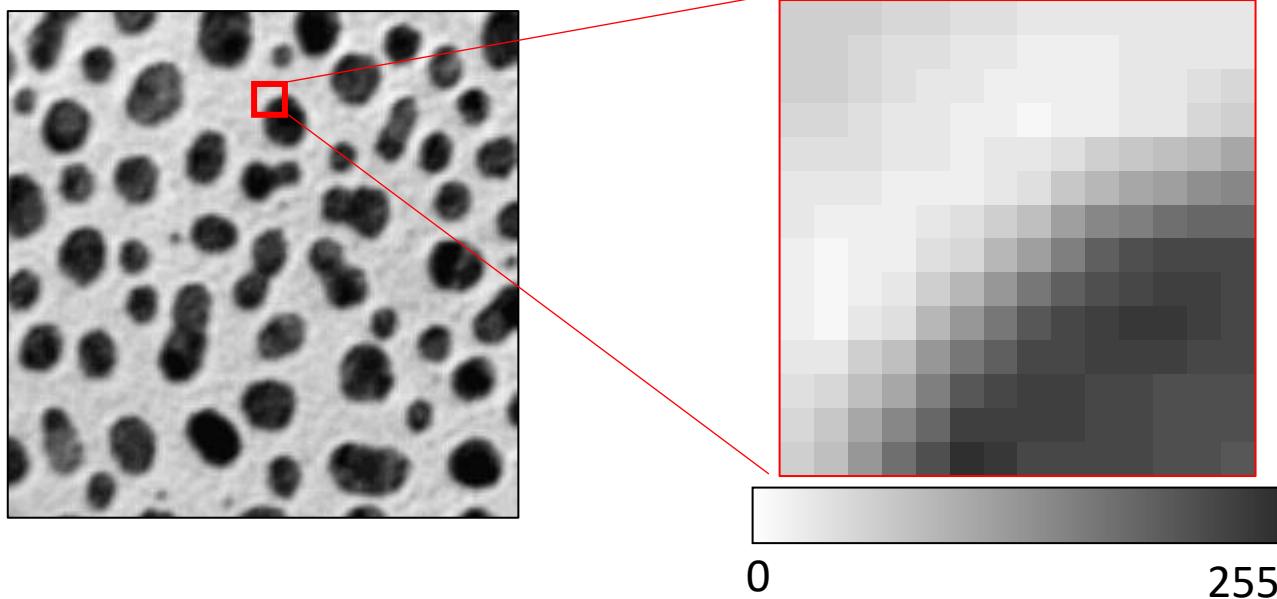
# Lecture overview: Bio-image Analysis

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



# Images and pixels

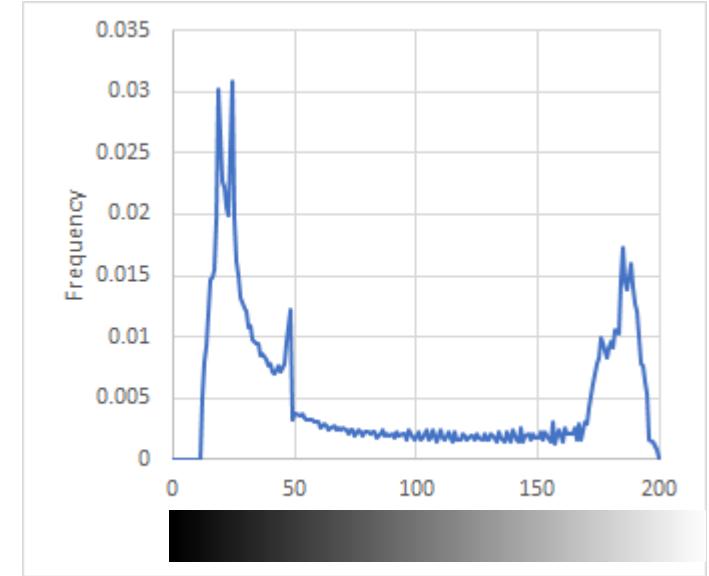
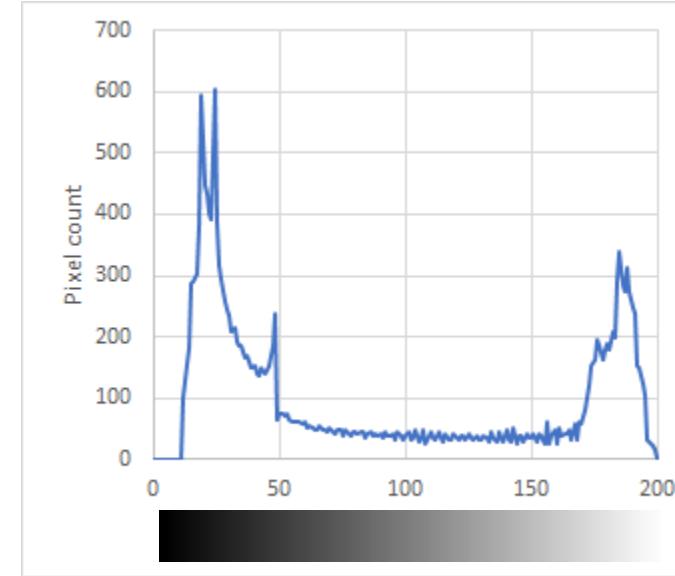
- An image is just a matrix of numbers: pixels: “picture element”
- The edges between pixels are an artefact of the imaging / digitization. They are not real!



48	48	48	40	40	32	32	24	24	24	24	24	24	24
48	48	40	32	32	24	24	16	16	16	16	24	24	24
48	48	40	32	24	24	16	16	16	16	16	24	24	32
40	40	32	24	24	16	16	8	16	16	16	24	24	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	176	176	176	168

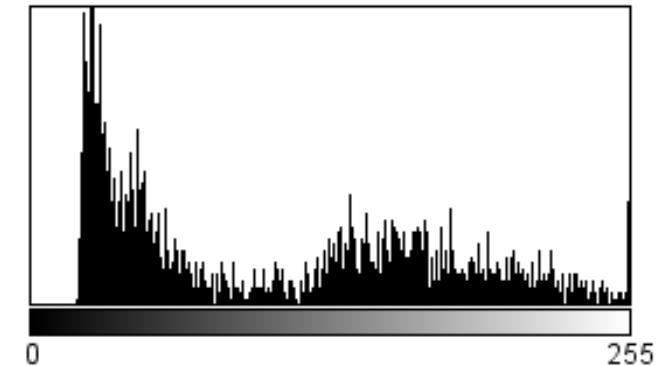
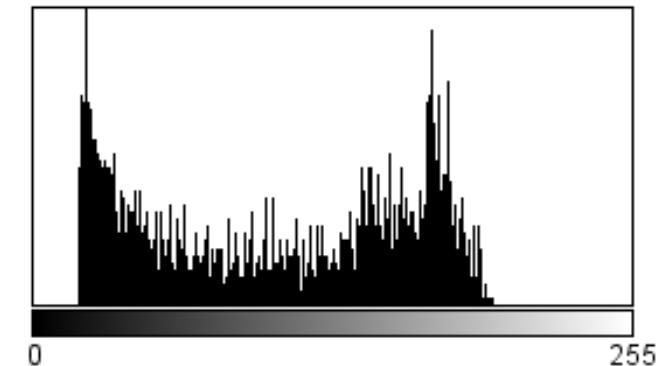
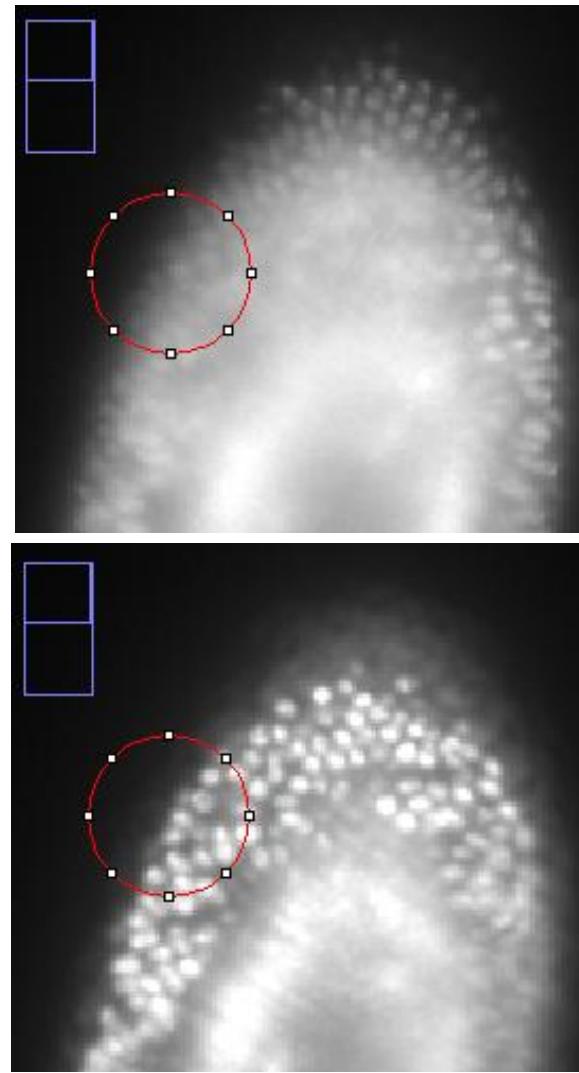
# Histograms

- A histogram shows the probability distribution of pixel intensities.
- The probability of a pixel having a certain intensity can be measured by counting pixels and calculating the frequency of the given intensity.
- Whenever you see a histogram, try to imagine the lookup-table on the X-axis



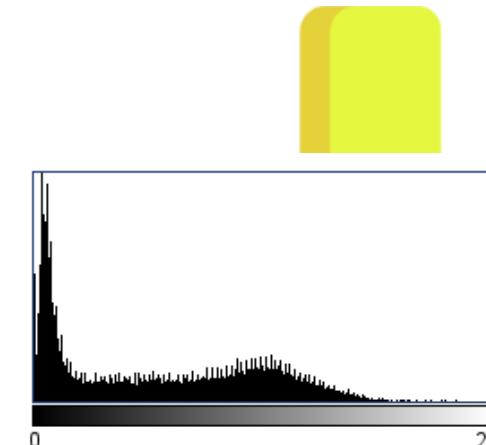
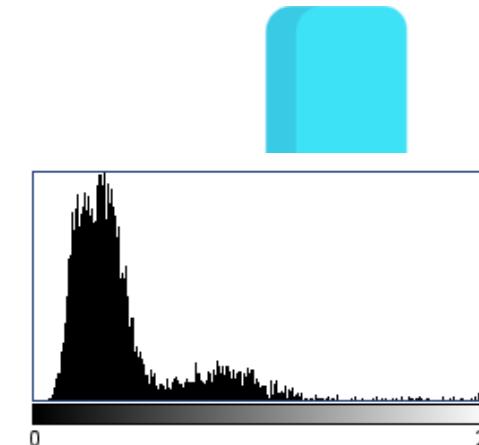
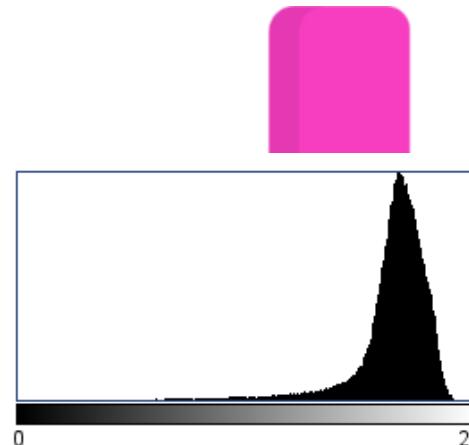
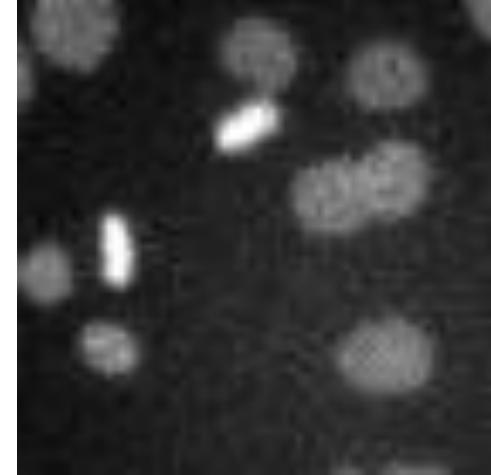
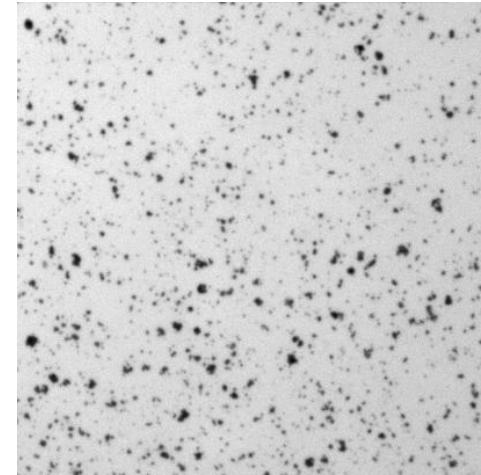
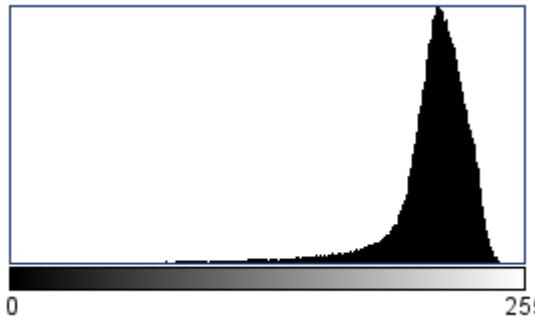
# Histograms

- Histograms are summaries of images
- Tell stories, e.g. about image quality



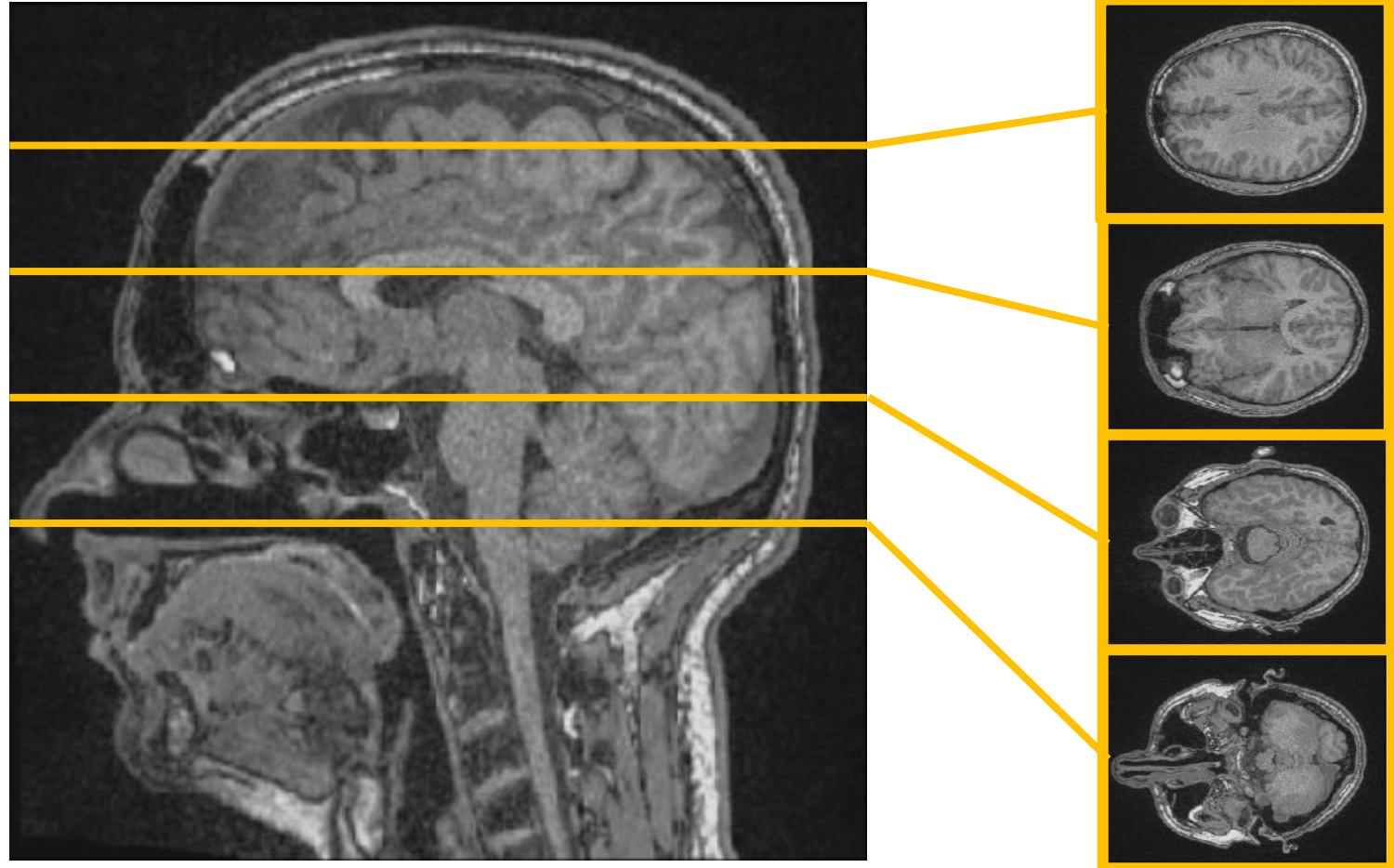
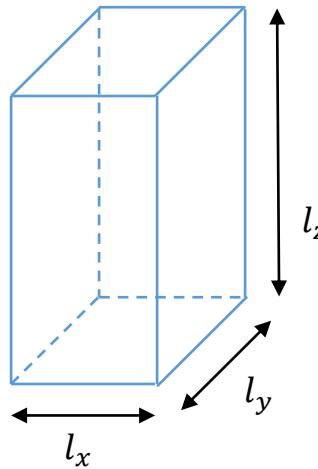
# Histograms

- To which image does this histogram belong to?



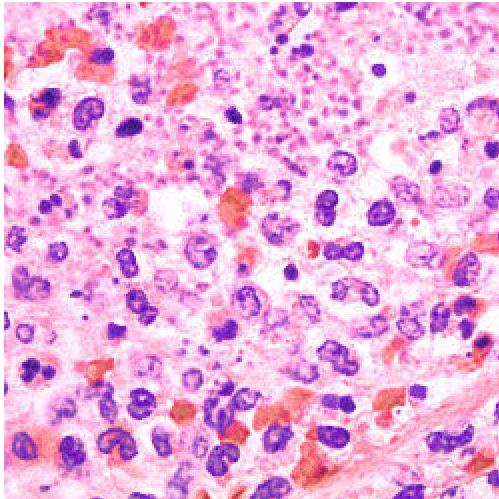
# Image stacks and voxels

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

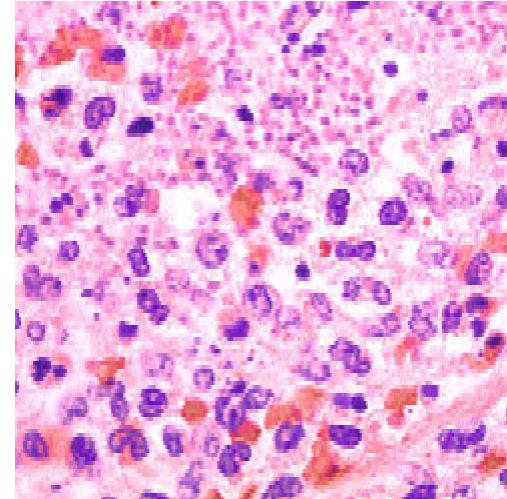


# Anisotropy

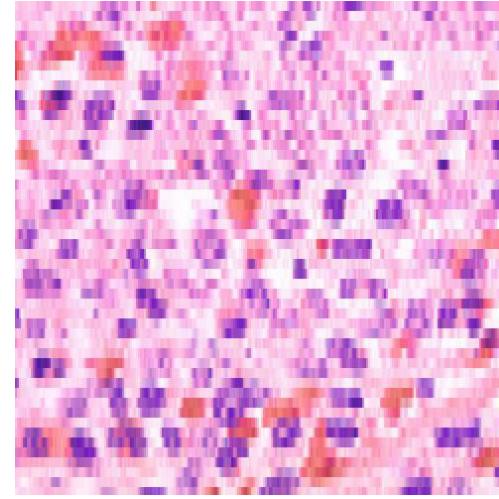
- Voxel size has immediate impact on image quality and thus, on processing / analysis results.



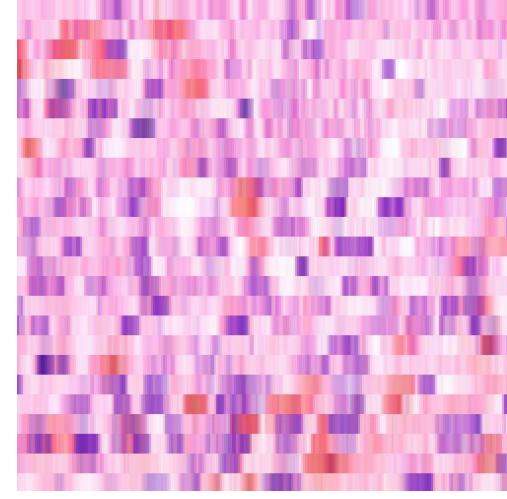
1:1  
250 x 250



1:2  
250 x 125



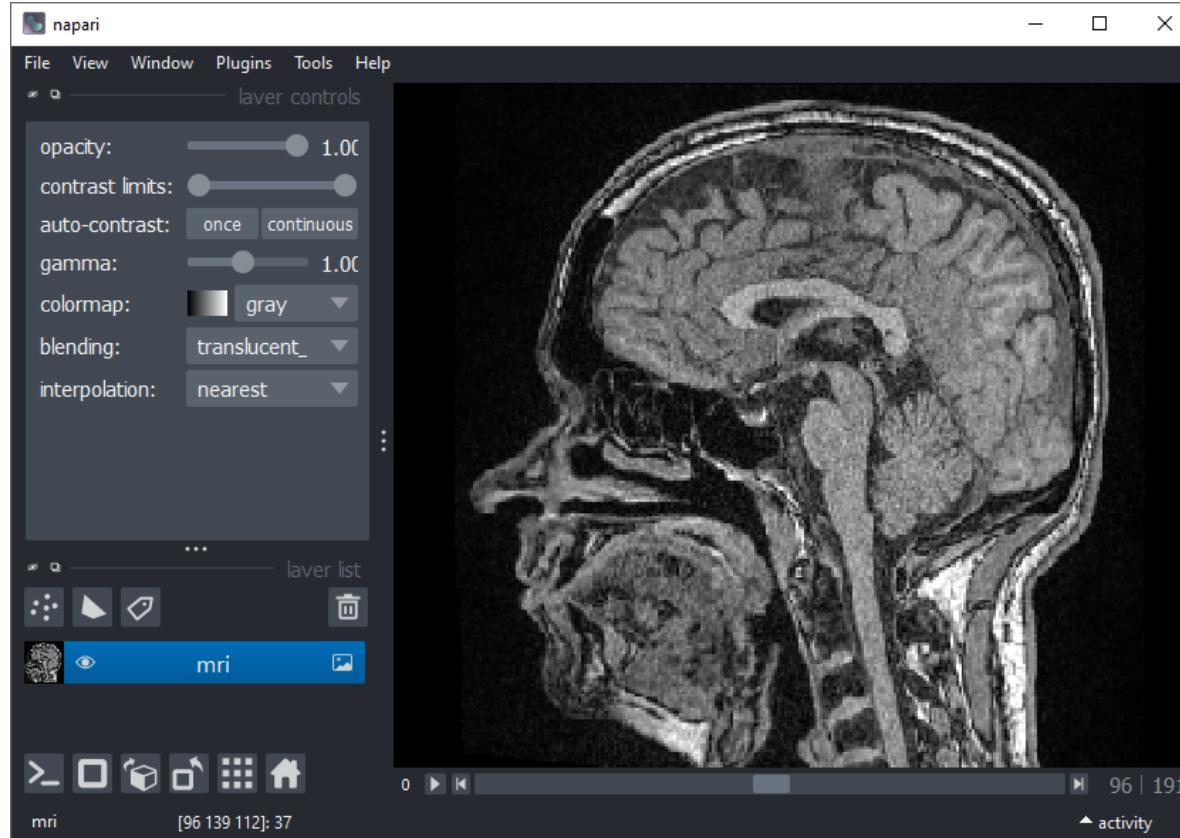
1:5  
250 x 50



1:10  
250 x 25

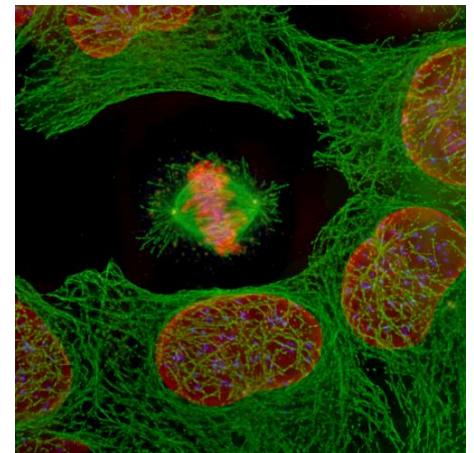
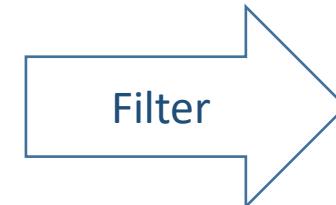
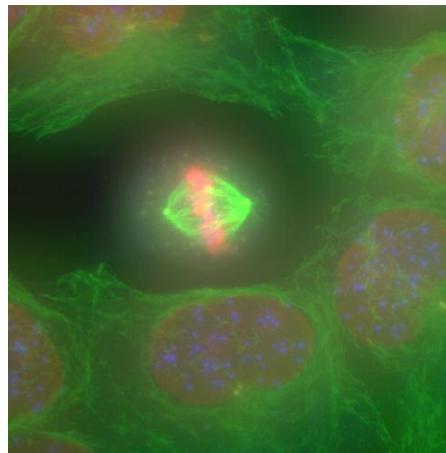
# 3D Image visualization

Interactive tools available



# Filters

- An image processing filter is an operation on an image.
- It consumes an image and produces a new image out of it.
- 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

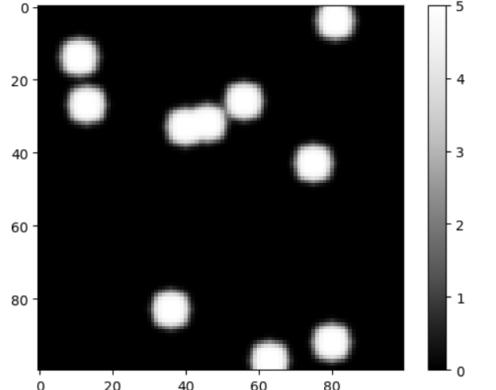


# Quiz

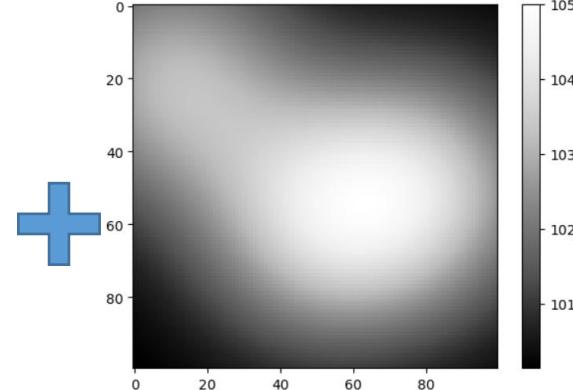
- What is noise?

# Effects harming image quality

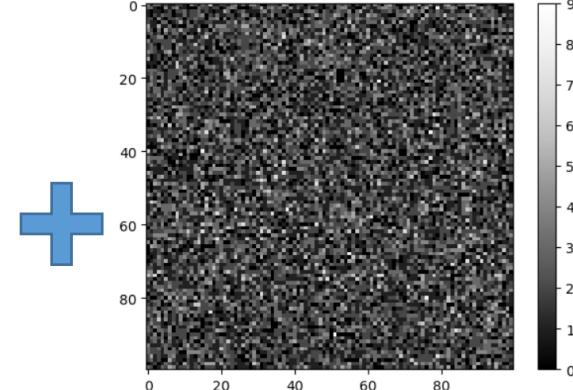
“nuclei”



“background”



“noise”



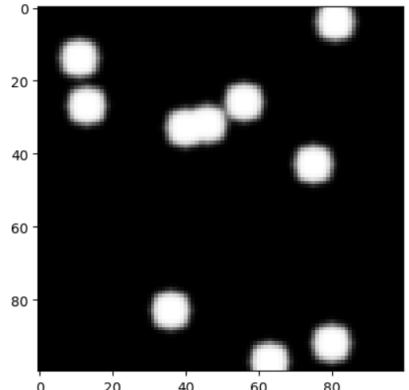
- Aberrations, defocus
- Motion blur

- Light from objects behind and in front of the scene (out-of-focus light)
- Dirt on the object slide
- Camera offset

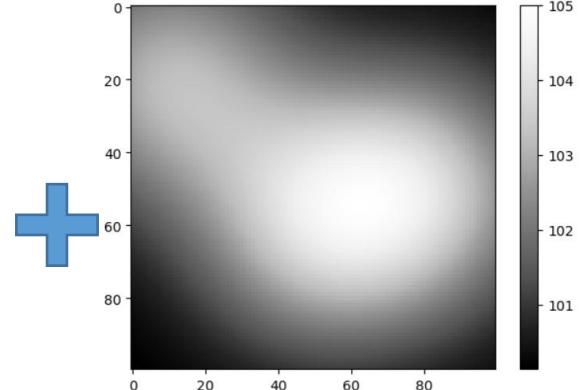
- Shot noise (arriving photons in the camera from the scene)
- Dark noise (electrons made from photons)
- Read-out-noise (electronics)

# Effects harming image quality

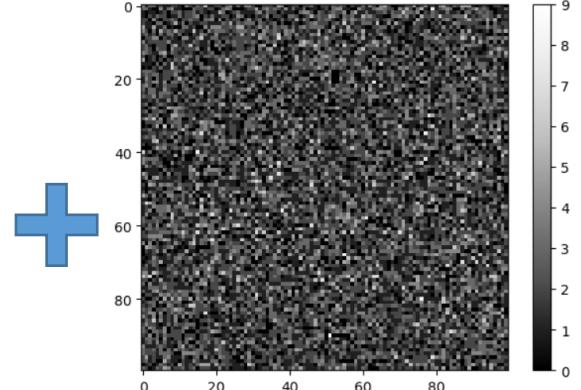
“nuclei”



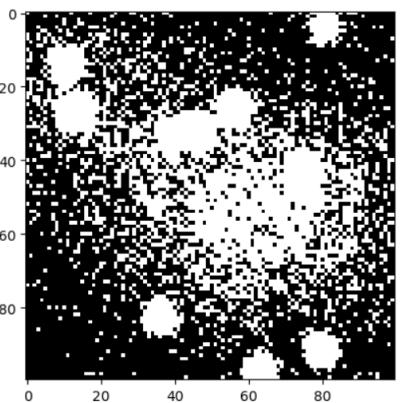
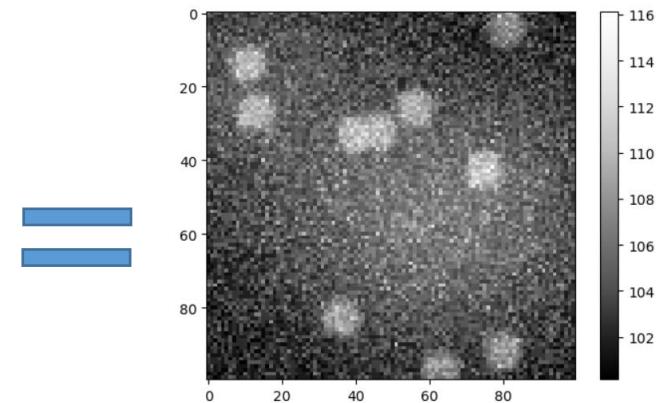
“background”



“noise”

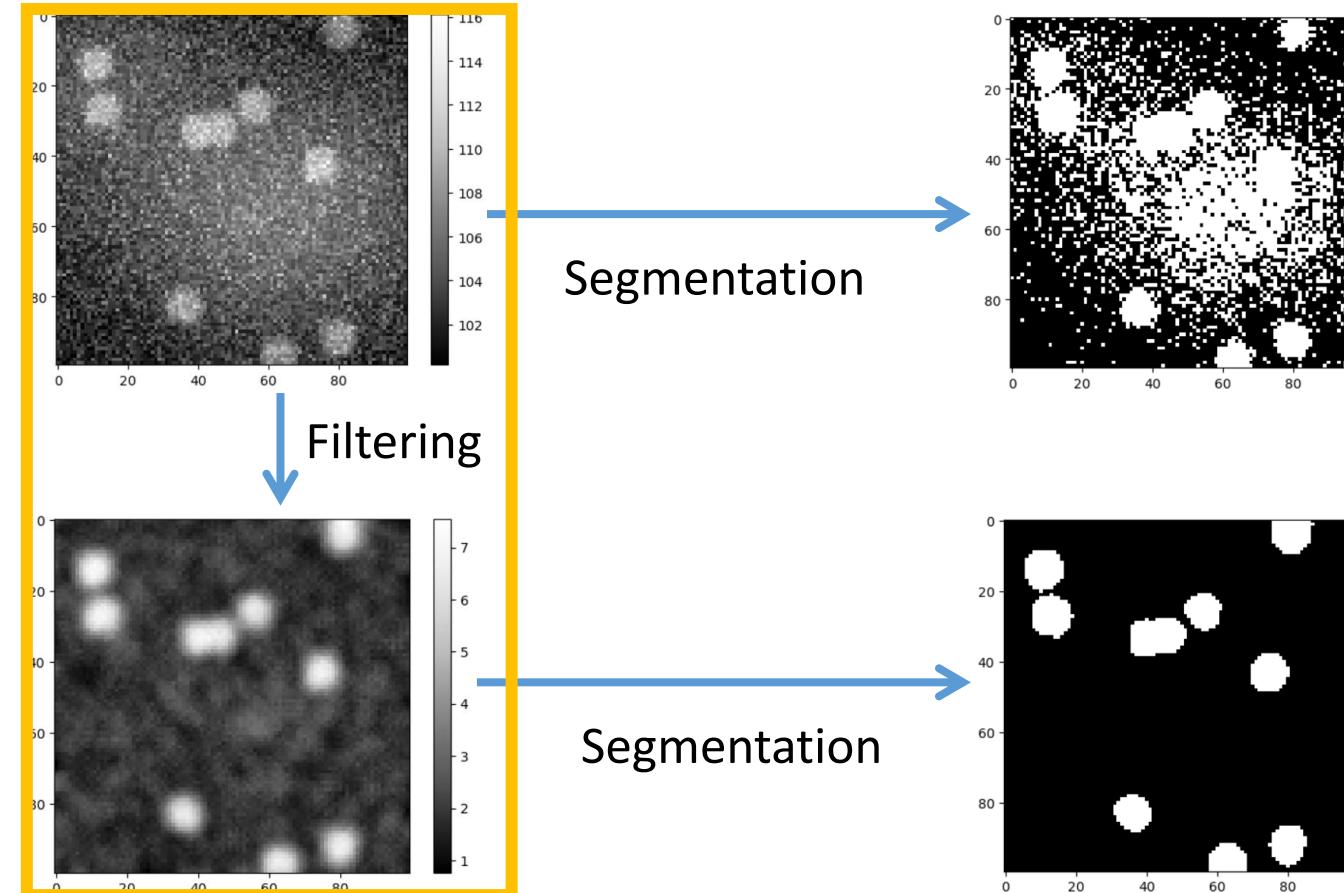


Segmentation



# Image filtering

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



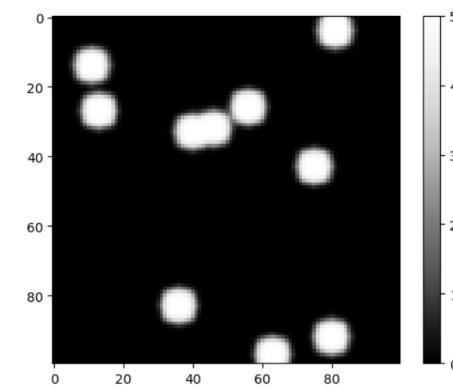
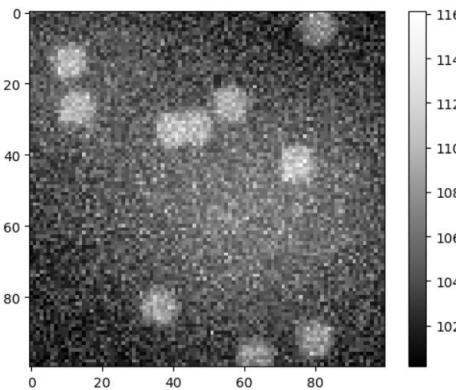
Oh no! I see thousands  
of tiny white objects!

Ok, it's just 9 objects.



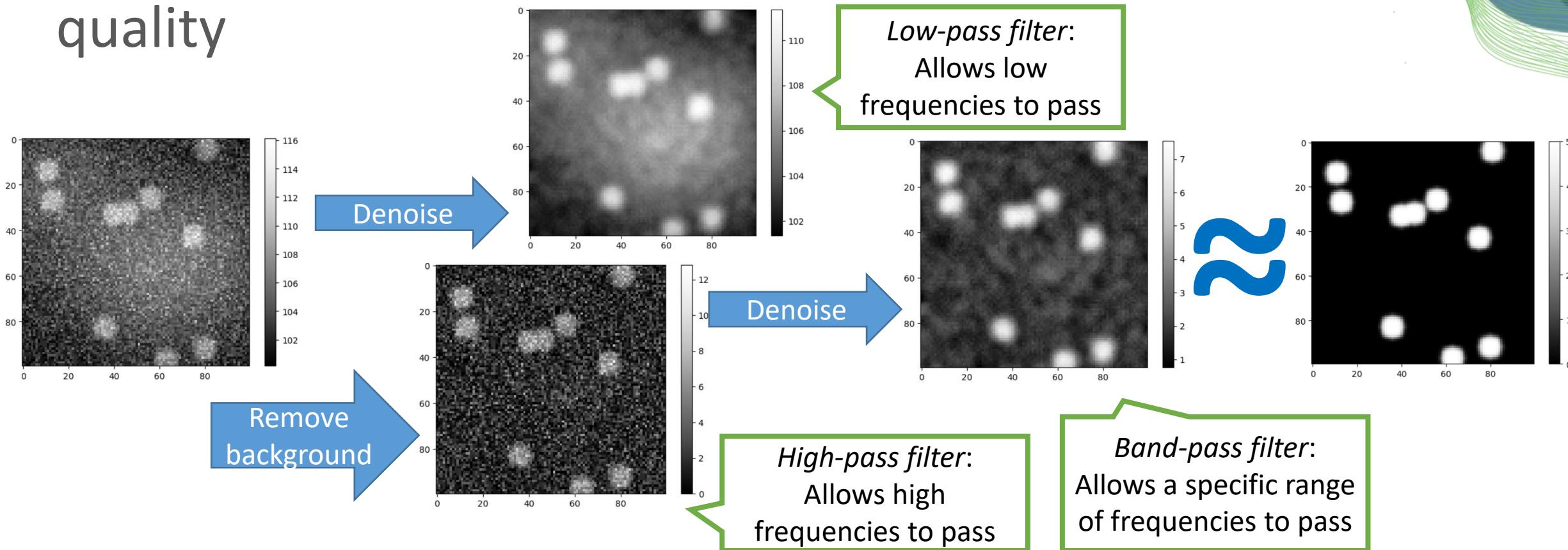
# Image filtering

- Attempt to invert / “undo” processes disturbing image quality



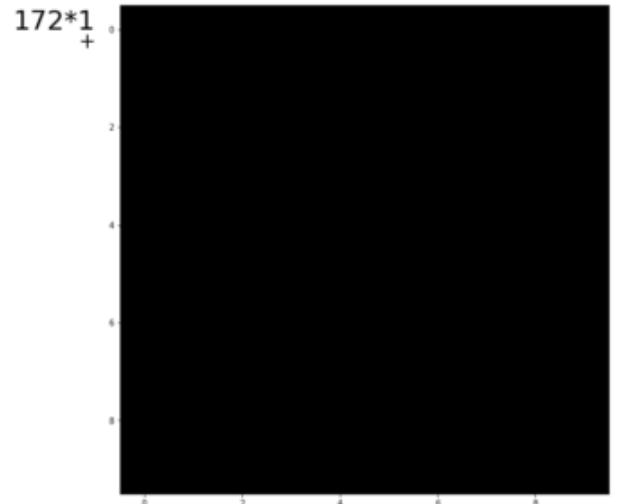
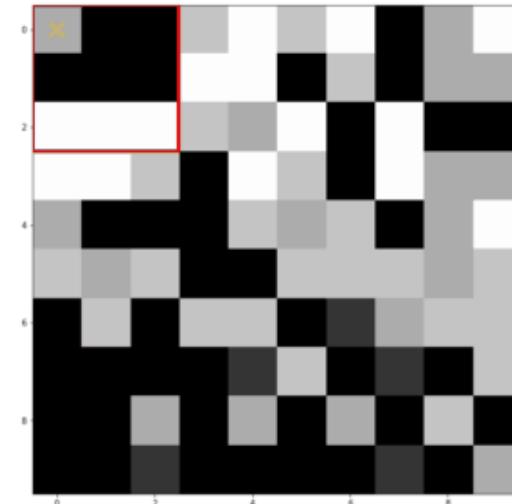
# Image filtering

- Attempt to invert / “undo” processes disturbing image quality



# Linear filters

- *Linear filters* compute each pixel value with a weighted linear combination of surrounding pixels
- Filter *kernels* are matrices describing a linear filter
- This multiplication of surrounding pixels according to a matrix is called *convolution*



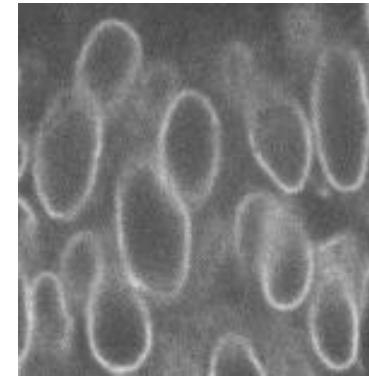
Mean filter, 3x3 kernel:

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

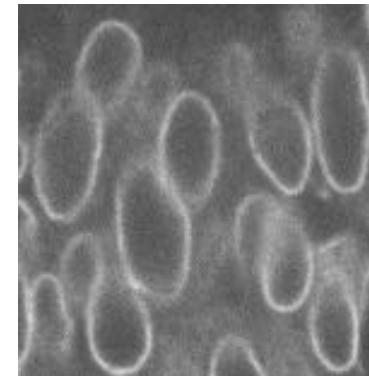
# Linear filters

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

- Examples
  - Mean
  - Gaussian blur
  - Sobel
  - Laplace



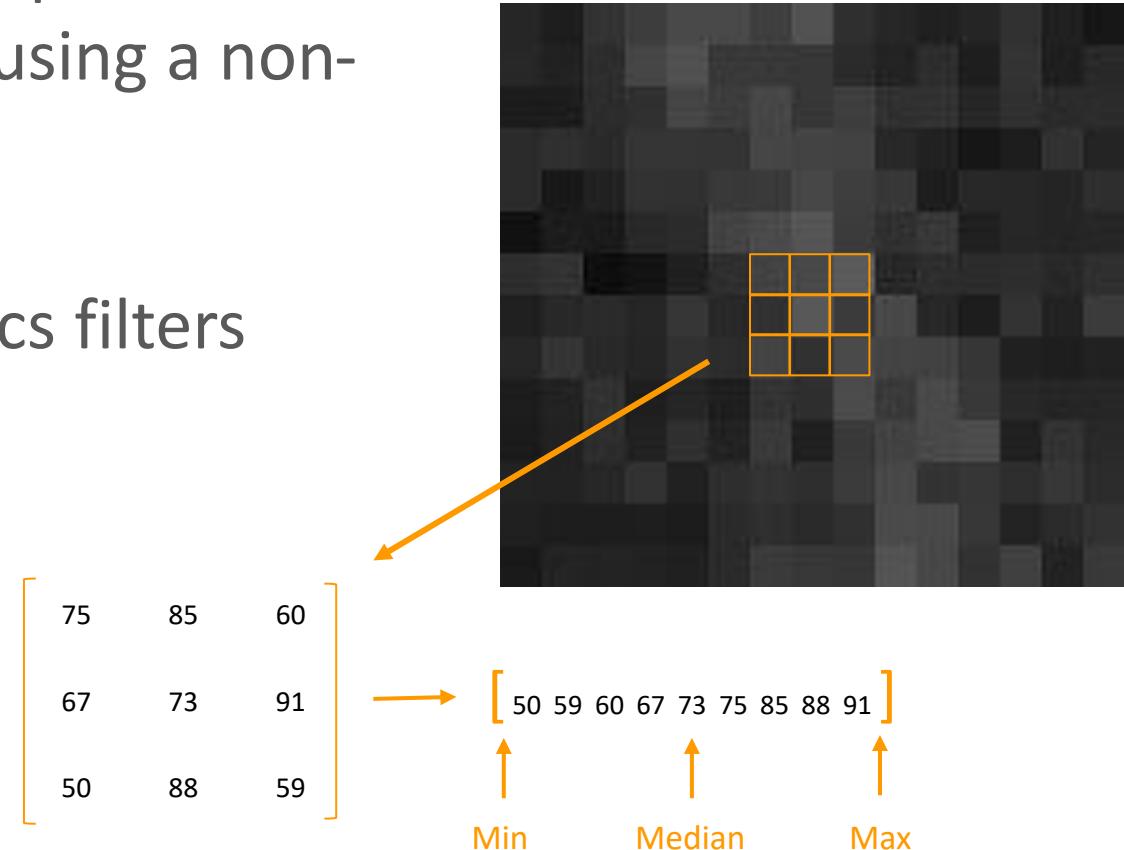
$$\text{Input} \quad * \quad \begin{bmatrix} 1 & 1 & 1 \\ 1 & 8 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \text{Output}$$



$$\text{Input} \quad * \quad \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} = \text{Output}$$

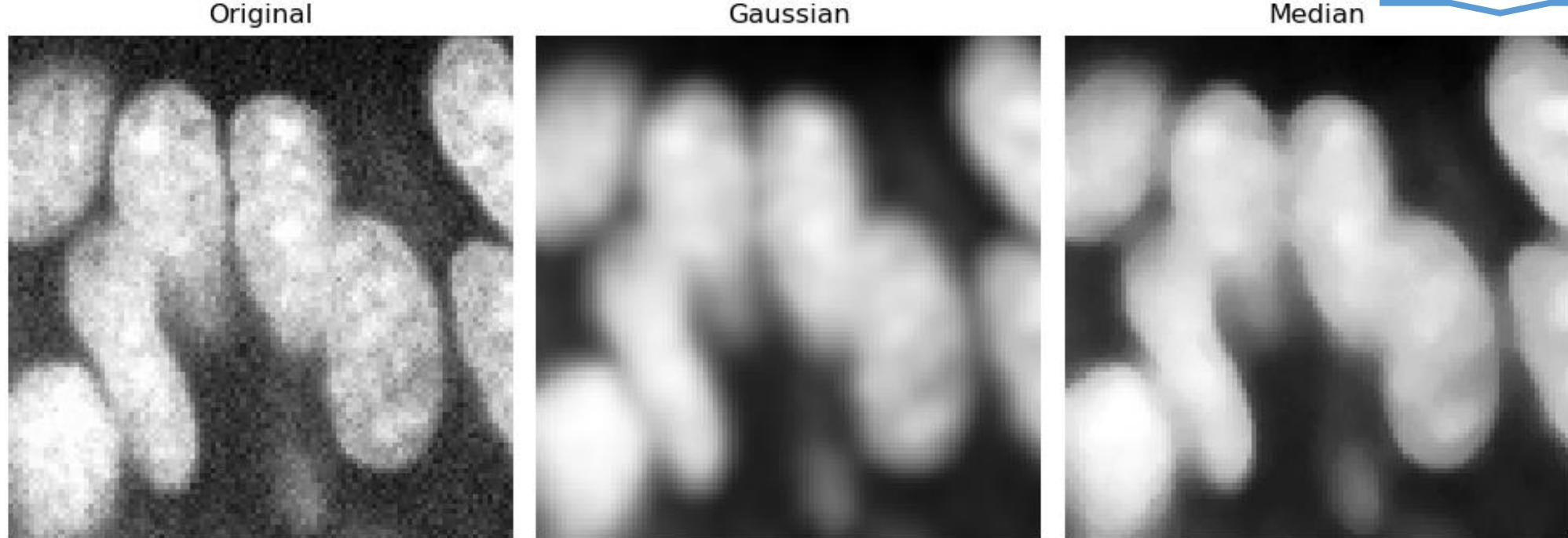
# Non-linear Filters

- Non-linear filters also replace pixel value inside a rolling window but using a non-linear function.
- Examples: descriptive statistics filters
  - Minimum
  - Median
  - Maximum
  - Variance
  - Standard deviation



# Noise removal

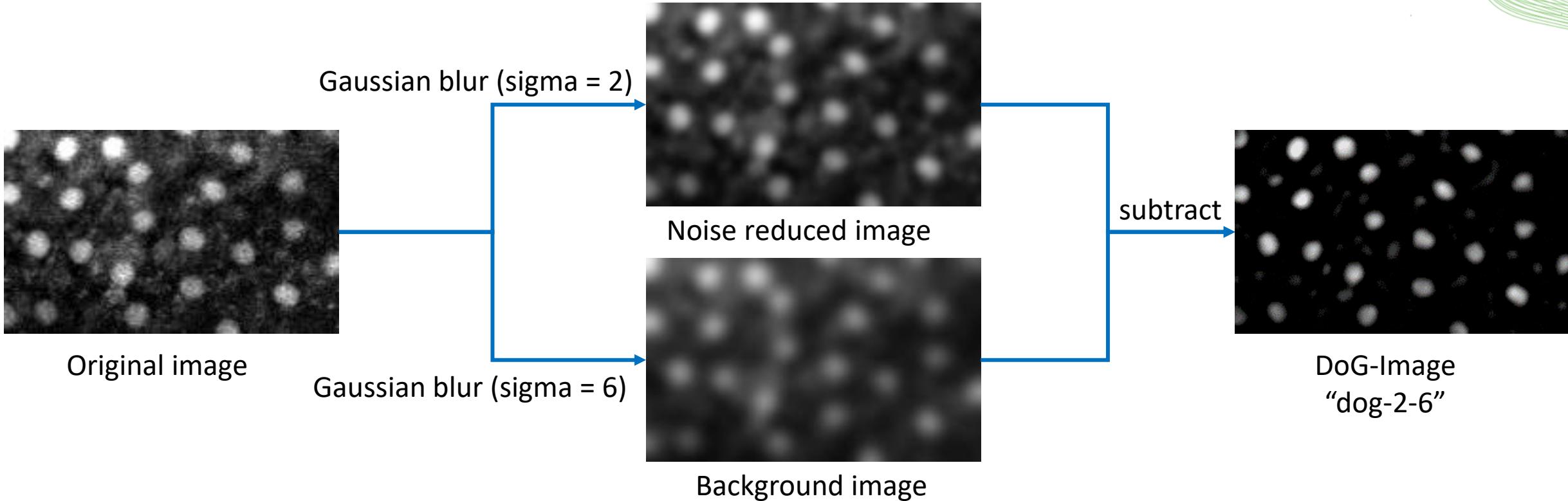
- Gaussian filter
- Median filter (computationally expensive)
- Deep learning (even more expensive, but state-of-the-art)



“Edge preserving”

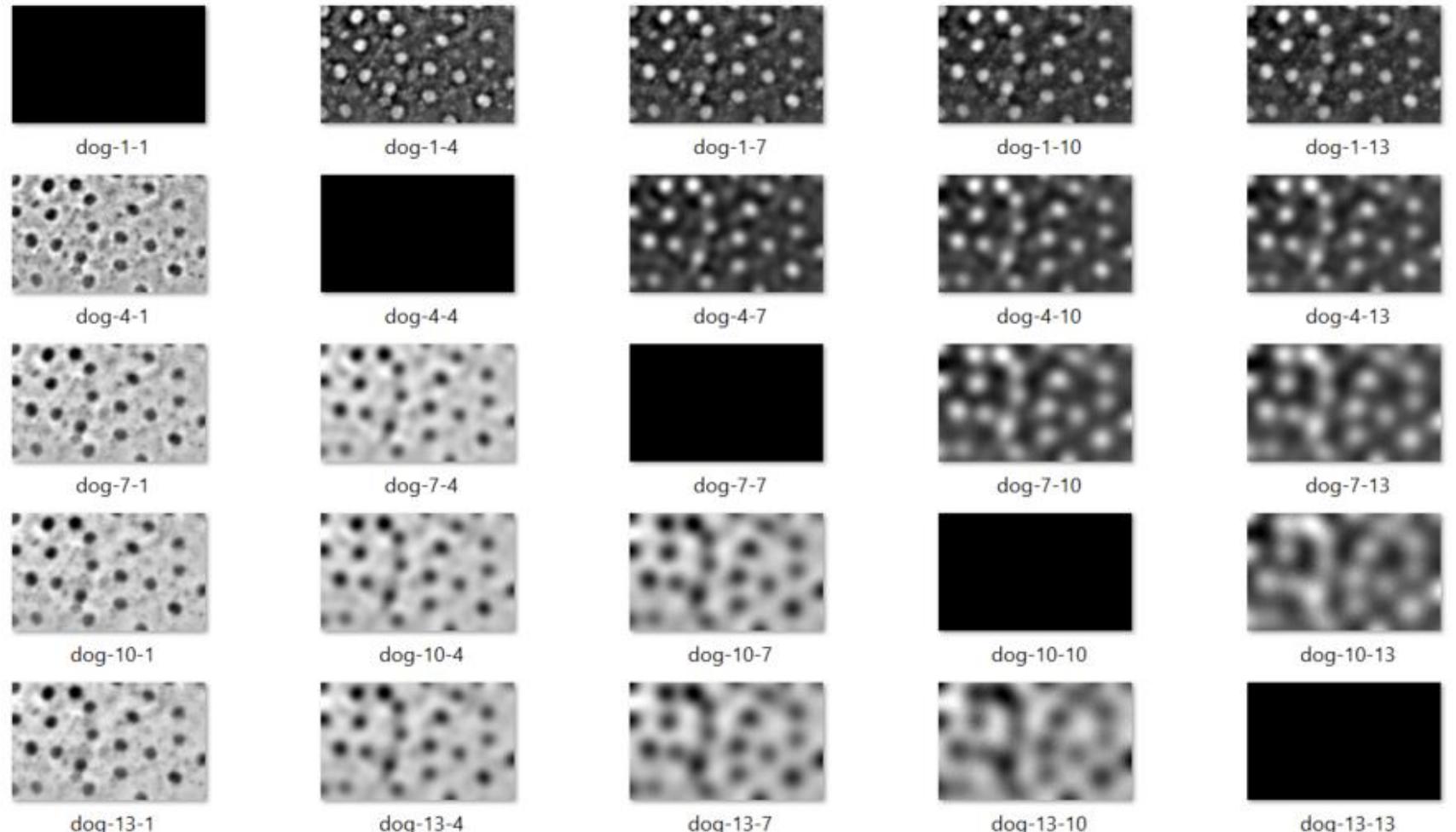
# Difference-of-Gaussian (DoG)

- Improve image in order to detect bright objects.
- Band-pass filter



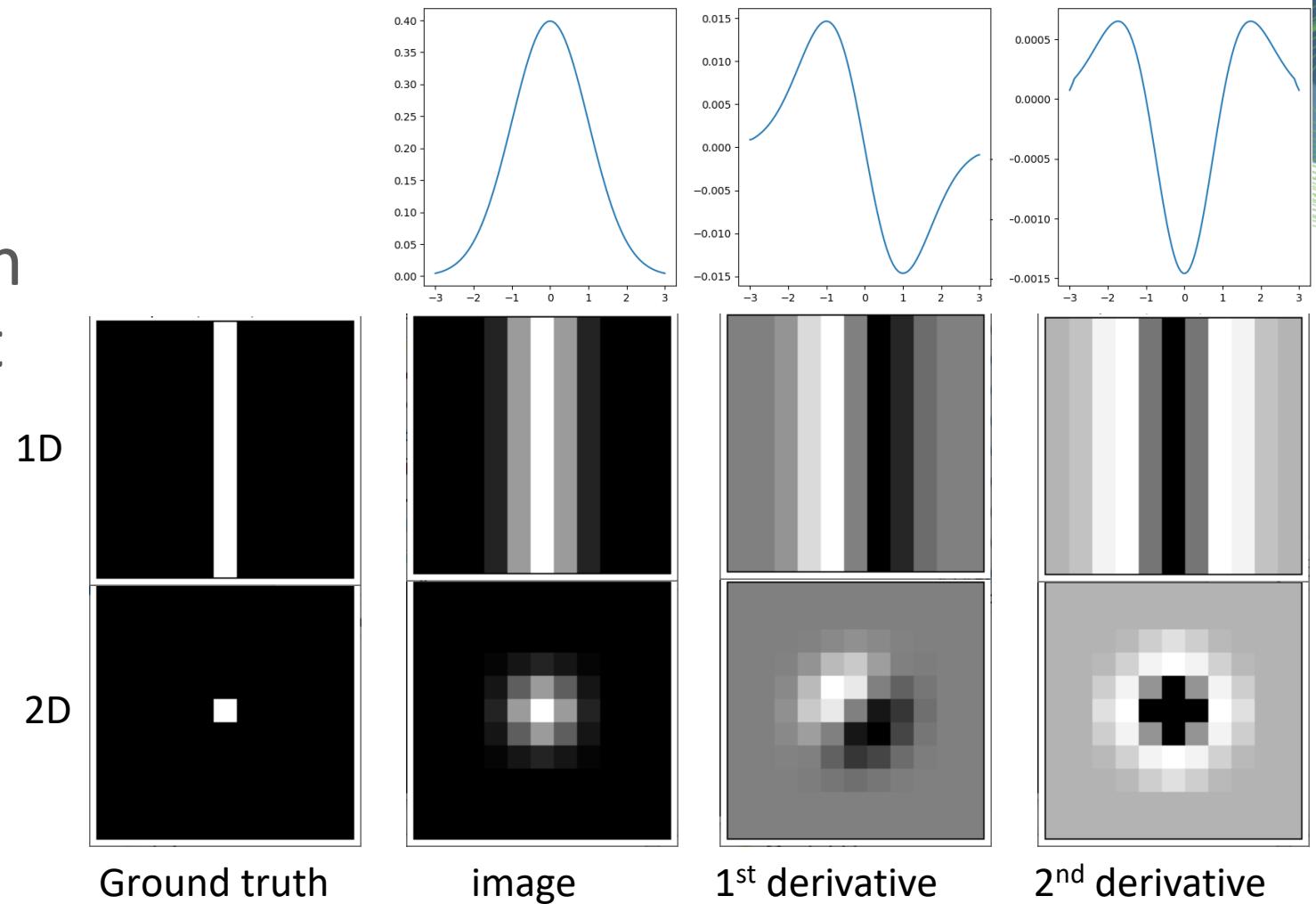
# Difference-of-Gaussian (DoG)

- Example DoG images



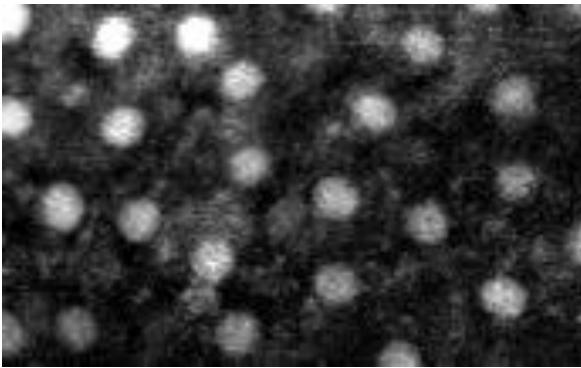
# Laplace-filter

- *Second derivative of a Gaussian filter*
- Used for edge-detection and edge enhancement
- Also known as the “Mexican-hat-filter”



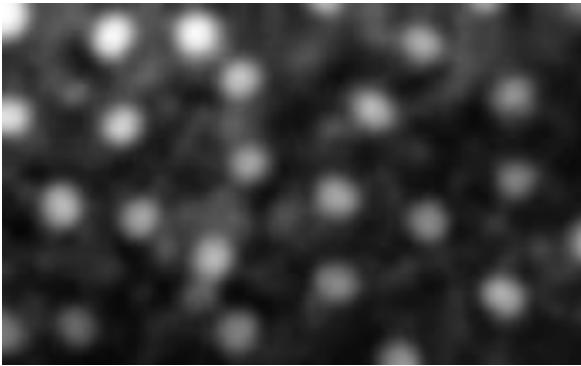
# Laplacian-of-Gaussian (LoG)

Laplace filter

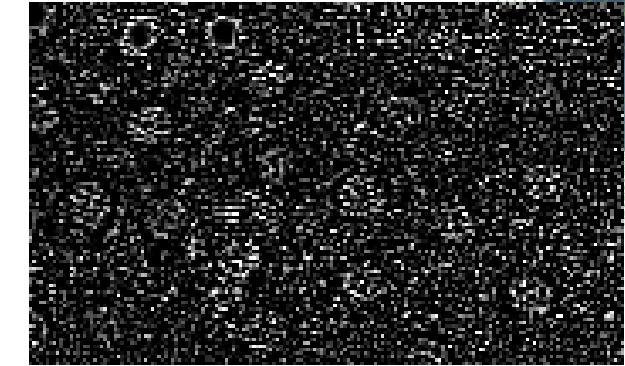


Gaussian filter

Laplacian of Gaussian filter

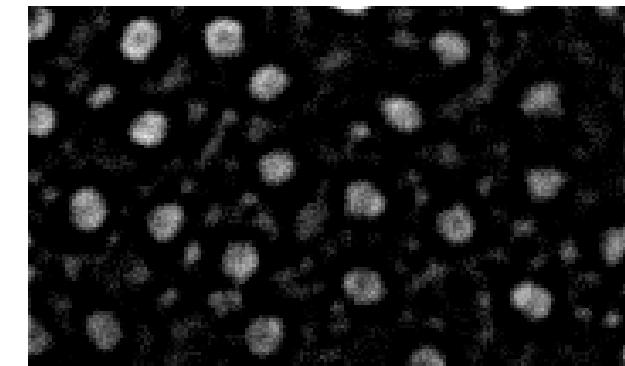


$$* \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} =$$



Laplace filtered image

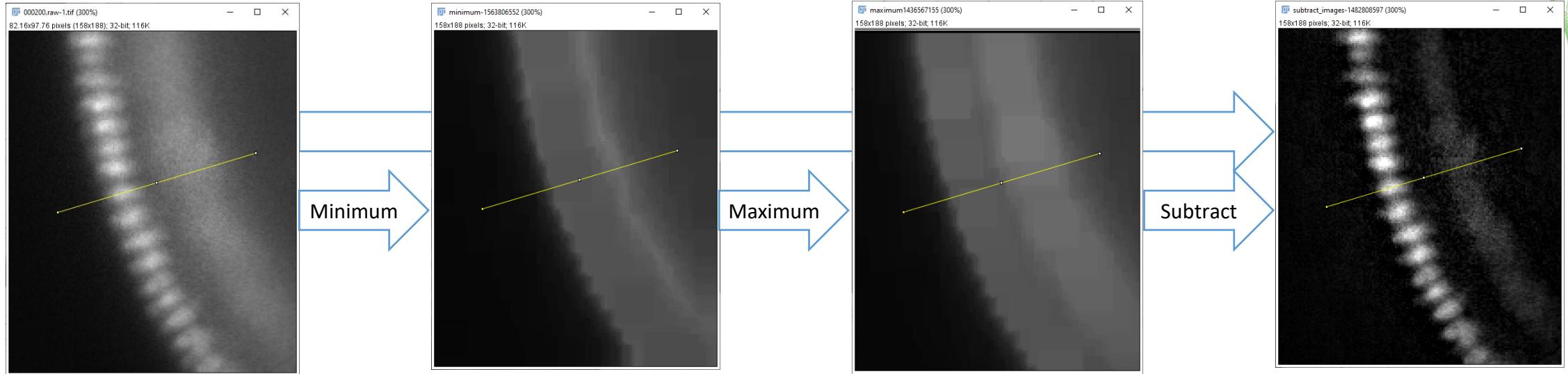
$$* \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} =$$



LoG image

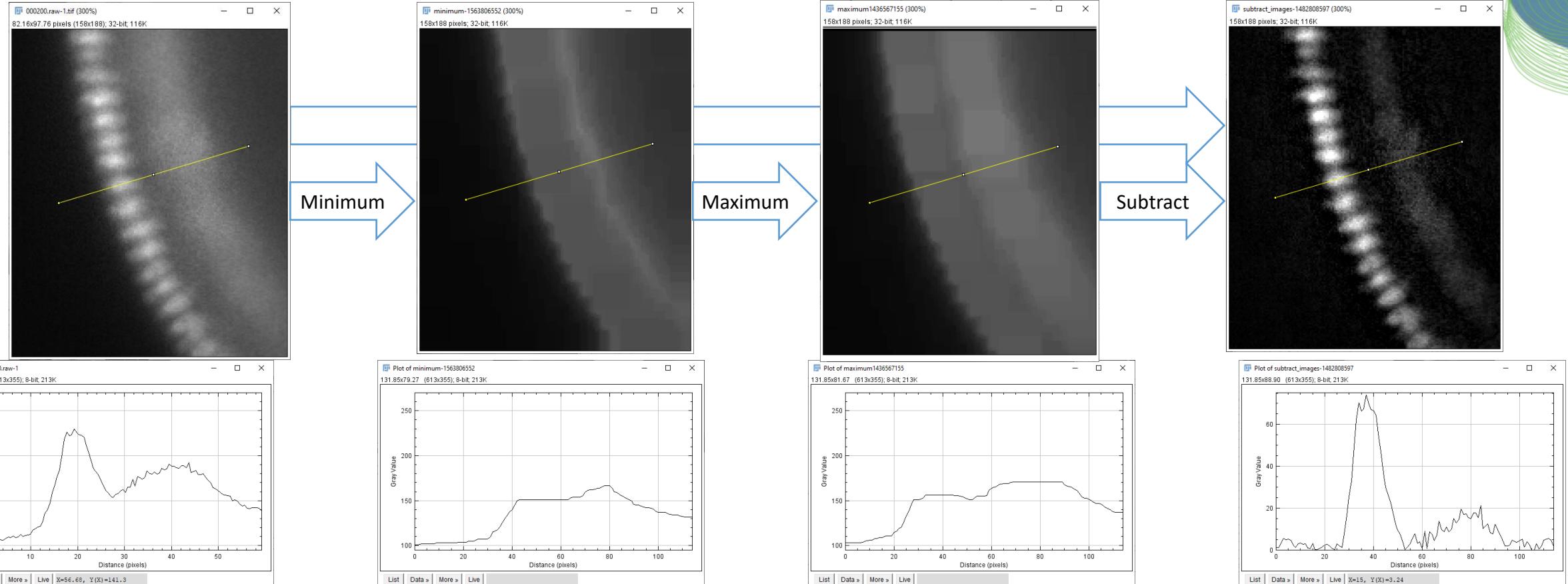
# Top-hat filter

- Background subtraction



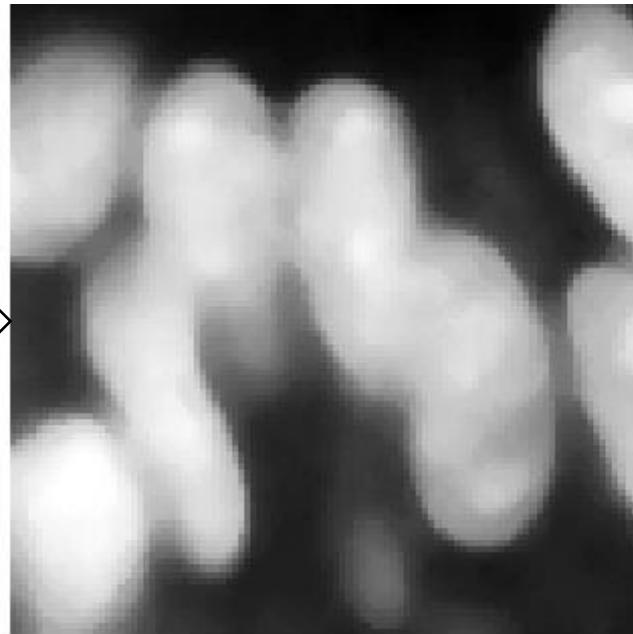
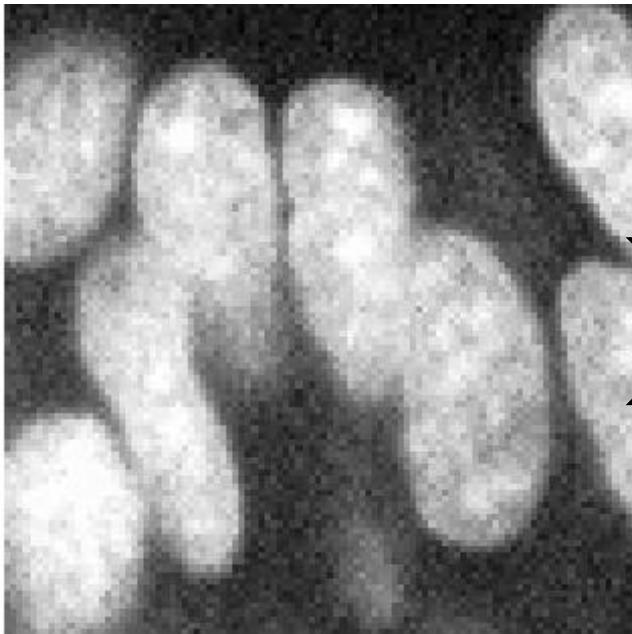
# Top-hat filter

- Background subtraction



# Quiz: Noise removal

- The median filter is a ...



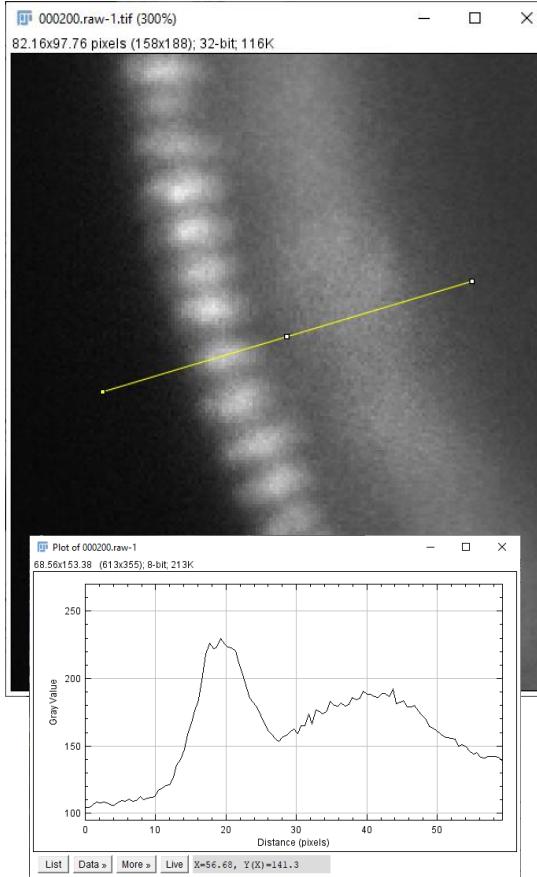
Linear  
filter



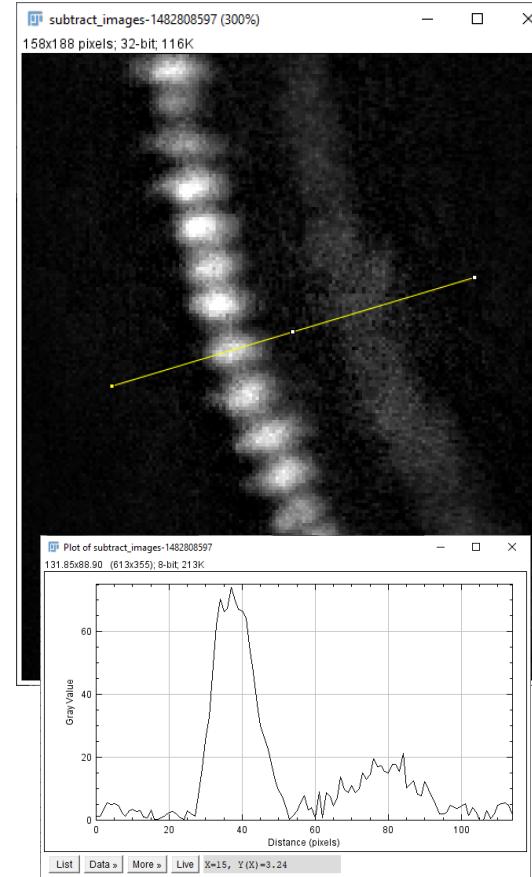
Non-linear  
filter

# Background removal

- Removing background from an image is a ... ?



Top-hat



Low-pass  
filter



High-pass  
filter



DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS  
AND ARTIFICIAL INTELLIGENCE

# Short detour: Image segmentation

Robert Haase

GEFÖRDERT VOM



Bundesministerium  
für Bildung  
und Forschung



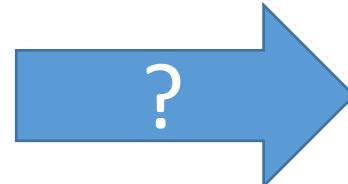
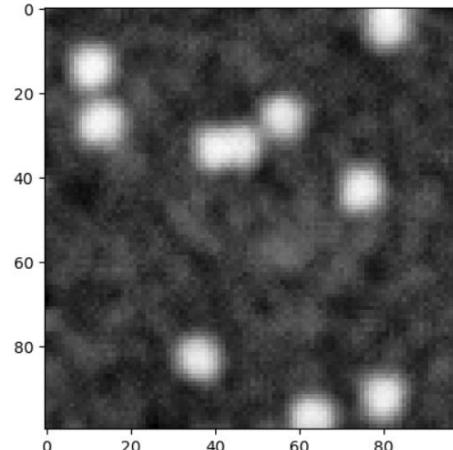
Diese Maßnahme wird gefördert durch die Bundesregierung  
aufgrund eines Beschlusses des Deutschen Bundestages.  
Diese Maßnahme wird mitfinanziert durch Steuermittel auf  
der Grundlage des von den Abgeordneten des Sächsischen  
Landtags beschlossenen Haushaltes.

# Short detour: Segmentation

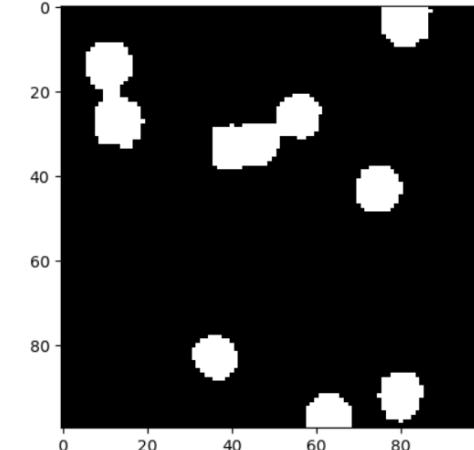
Binarization, e.g. via thresholding

- Very basic and yet efficient segmentation technique
- Histogram based, to determine an intensity threshold above which pixels become white
- Not state-of-the-art in many fields anymore

Intensity image



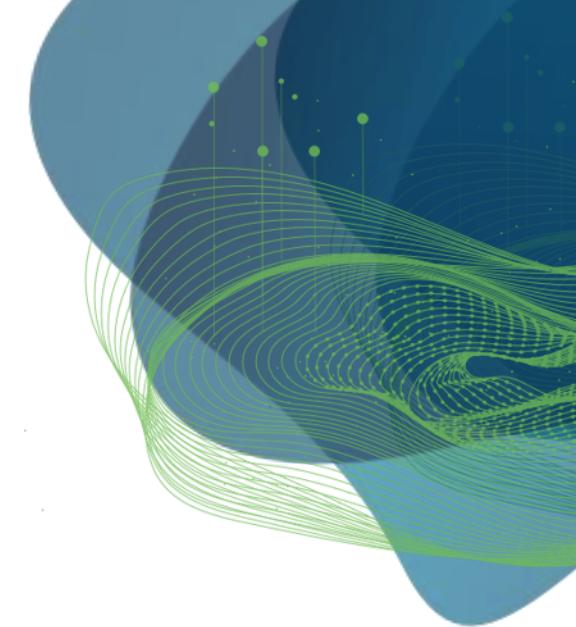
Binary image





DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS  
AND ARTIFICIAL INTELLIGENCE



# Image Processing: Morphological Operations

Robert Haase

With material from

Marcelo Leomil Zoccoler, Physic of Life, TU Dresden

GEFÖRDERT VOM



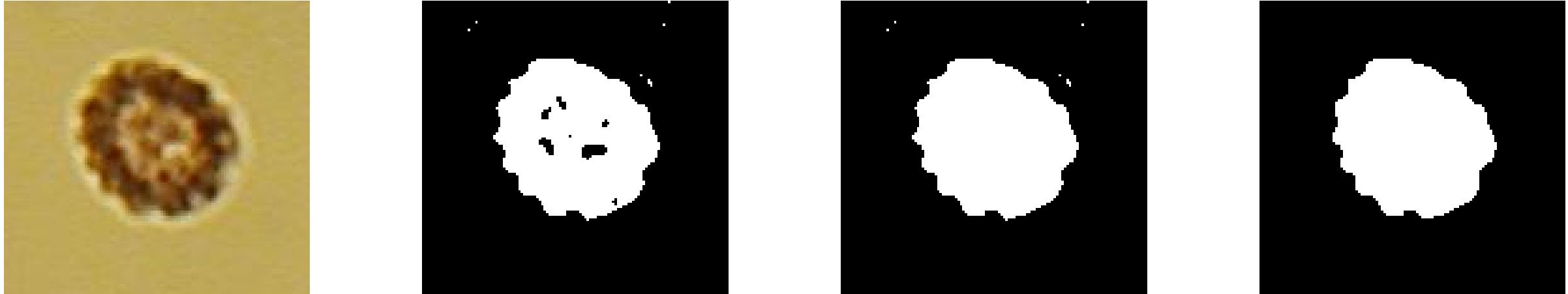
Bundesministerium  
für Bildung  
und Forschung



Diese Maßnahme wird gefördert durch die Bundesregierung  
aufgrund eines Beschlusses des Deutschen Bundestages.  
Diese Maßnahme wird mitfinanziert durch Steuermittel auf  
der Grundlage des von den Abgeordneten des Sächsischen  
Landtags beschlossenen Haushaltes.

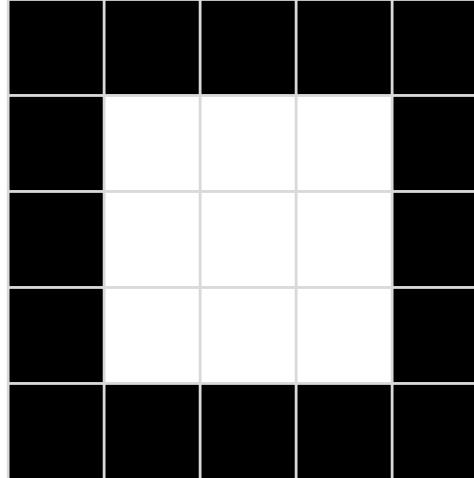
# Refining masks

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

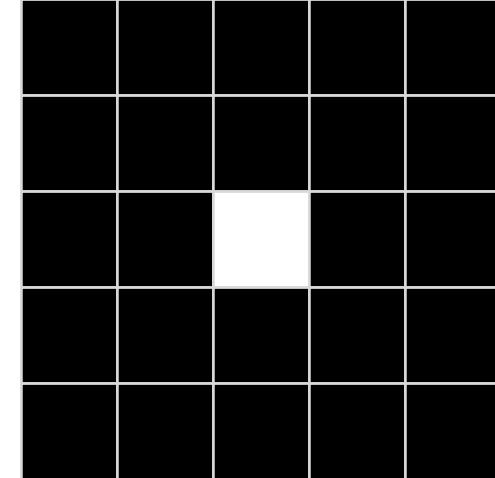


# Erosion

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

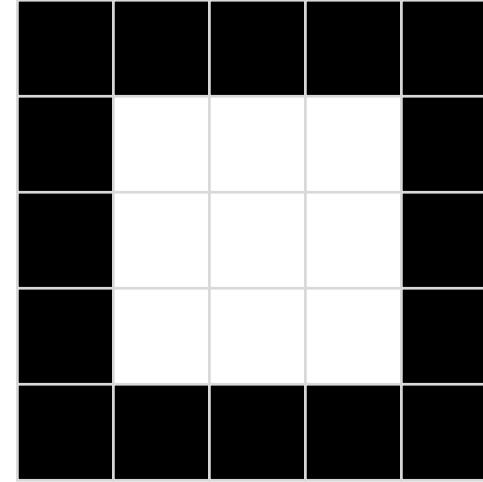
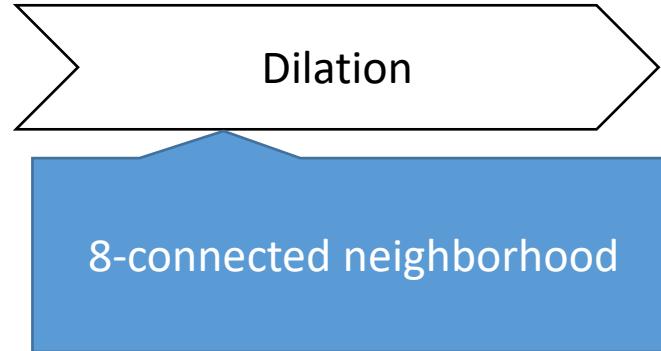
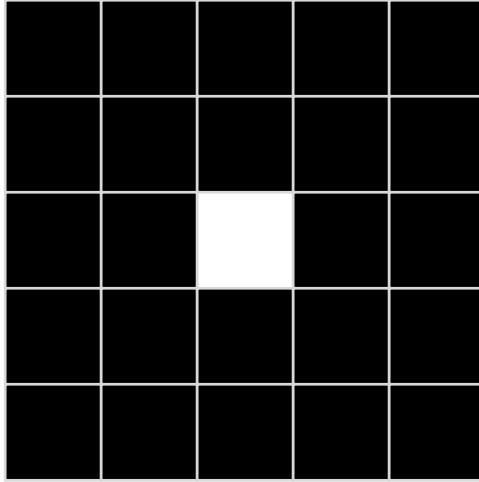


Erosion



# Dilation

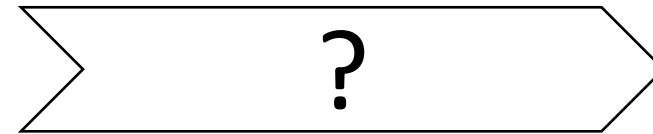
- Dilation: Every pixel with at least one white neighbor becomes white.



# Quiz

- What would be the equivalent image processing filter you learned before? Hint: assume black=0 and white=1.

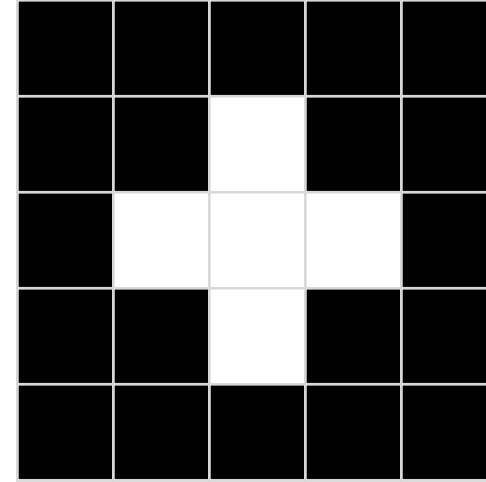
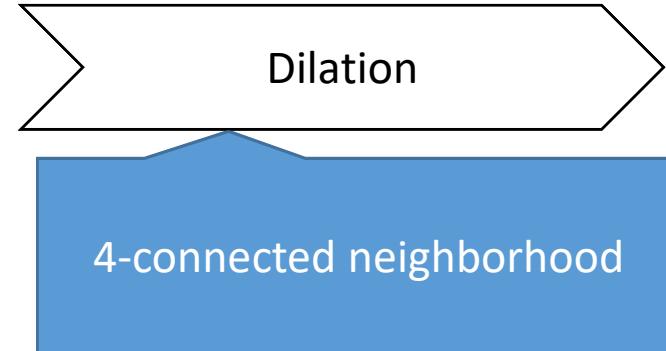
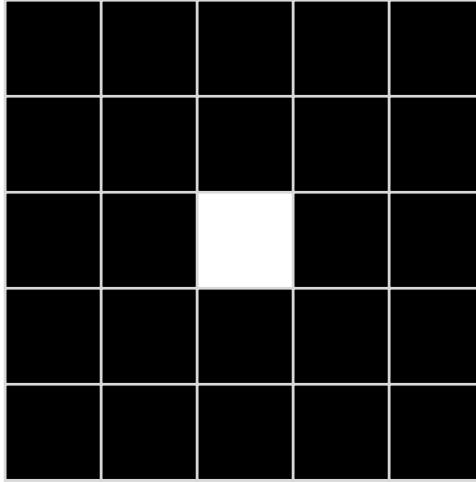
0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	0	0



0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

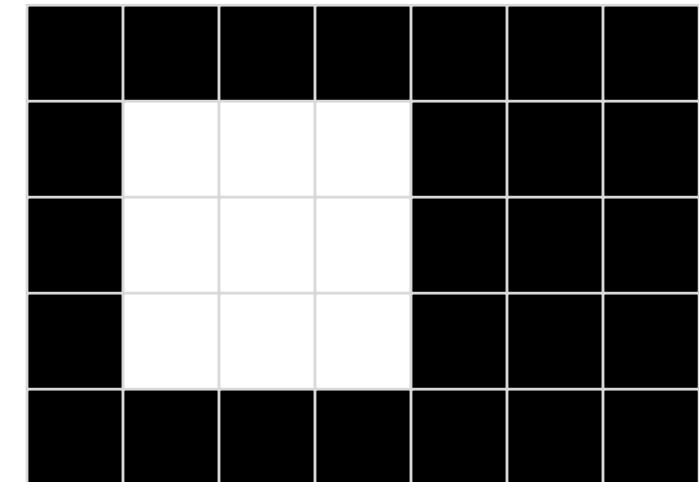
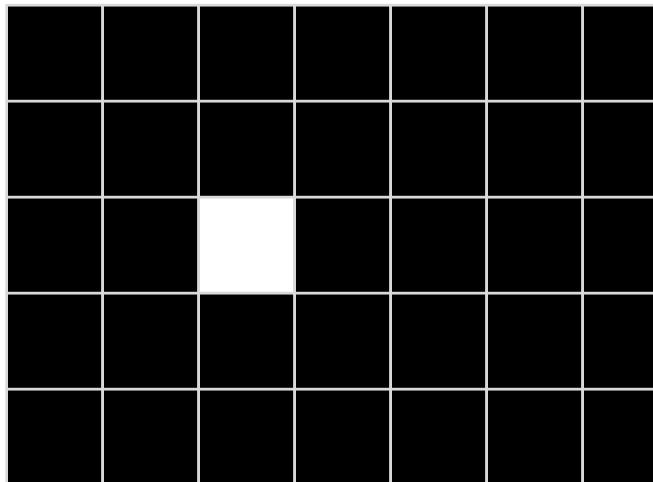
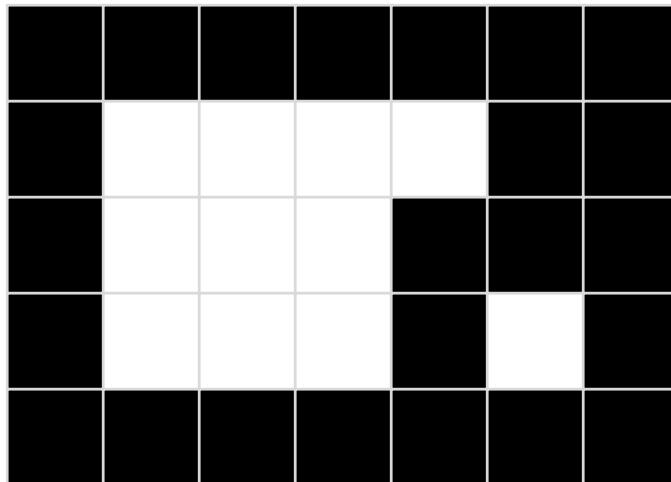
# Dilation

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



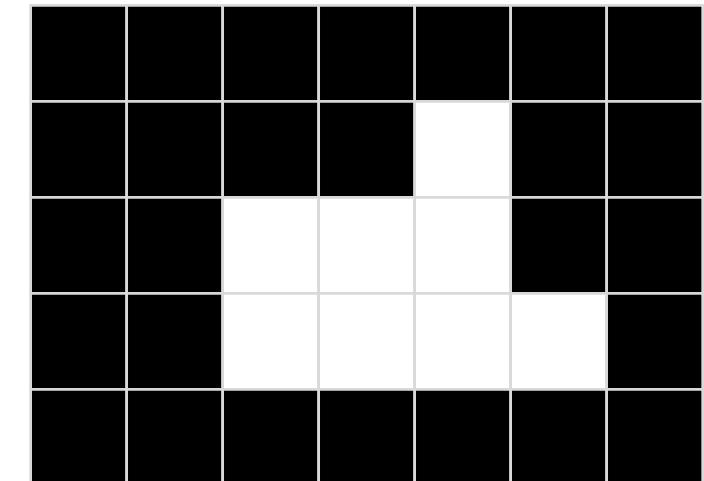
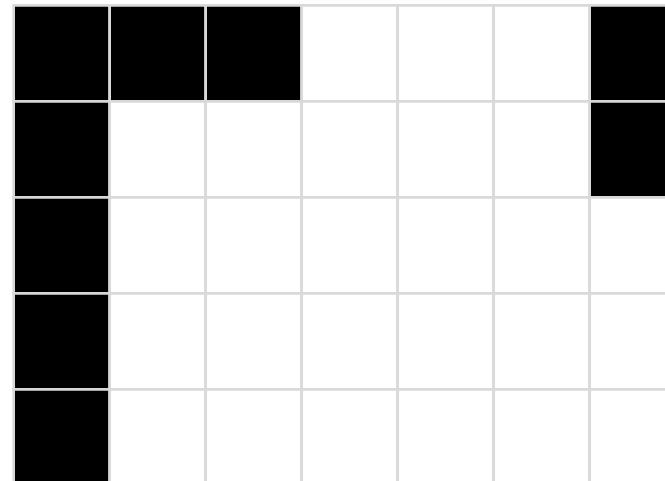
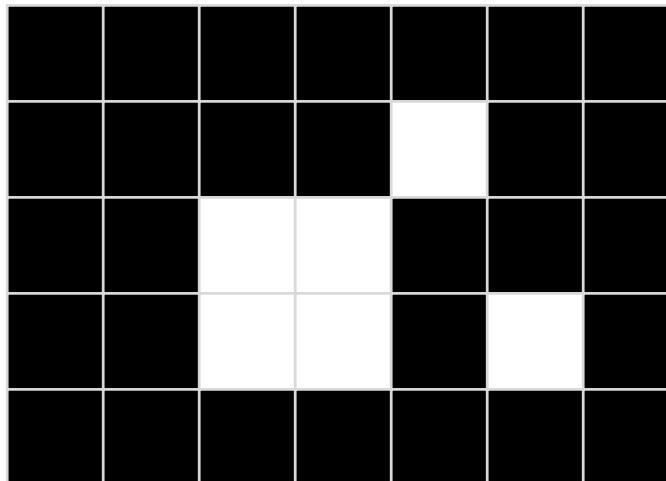
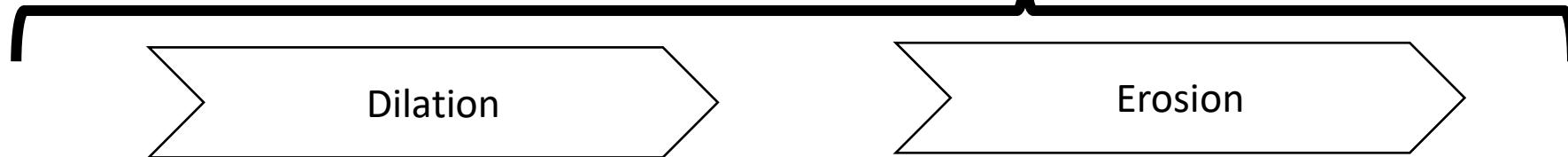
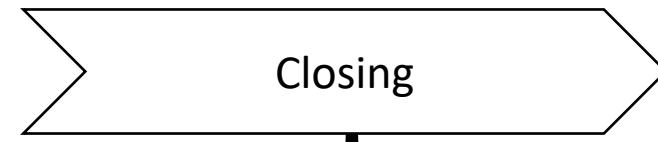
# Opening

- Erosion and dilation combined allow correcting outlines.
- Separates white structures
- Erases small white structures



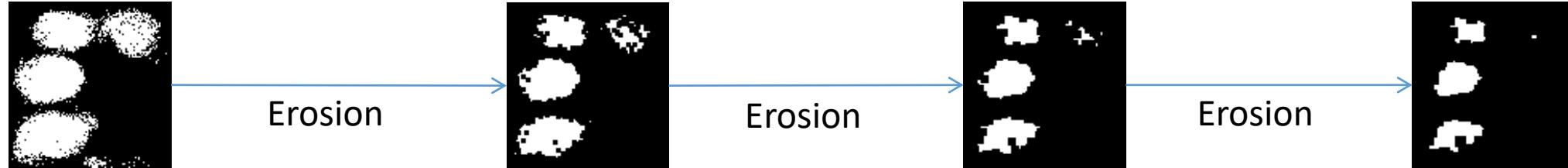
# Closing

- Erosion and dilation combined allow correcting outlines.
- Connects white structures
- Closes small holes in white structures



# Chaining erosion and 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



DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS  
AND ARTIFICIAL INTELLIGENCE

# Microscopy Image Processing in Python

Robert Haase



Robert Haase  
@haesleinhuepf  
BIDS Lecture 2/12  
April 25th 2025



Diese Maßnahme wird gefördert durch die Bundesregierung aufgrund eines Beschlusses des Deutschen Bundestages.  
Diese Maßnahme wird mitfinanziert durch Steuermittel auf der Grundlage des von den Abgeordneten des Sächsischen Landtags beschlossenen Haushaltes.

42

# Working with images in python

- Open images, e.g. using scikit-image

```
from skimage.io import imread  
  
image = imread("blobs.tif")
```

```
image  
  
array([[ 40,  32,  24, ..., 216, 200, 200],  
       [ 56,  40,  24, ..., 232, 216, 216],  
       [ 64,  48,  24, ..., 240, 232, 232],  
       ...,  
       [ 72,  80,  80, ..., 48,  48,  48],  
       [ 80,  80,  80, ..., 48,  48,  48],  
       [ 96,  88,  88, ..., 48,  48,  48]], dtype=uint8)
```

Images are *just* multi-dimensional arrays or “arrays of arrays”.

# Loading images

- Proprietary file formats, e.g. using *aicsimageio* or *bioio*

```
[3]: aics_image = AICSImage("data/PupalWing.czi")
aics_image
```

```
[3]: <AICSImage [Reader: CziReader, Image-is-in-Memory: False]>
```

```
[4]: aics_image.shape
```

```
[4]: (1, 1, 80, 520, 692)
```

```
[5]: aics_image.dims
```

```
[5]: <Dimensions [T: 1, C: 1, Z: 80, Y: 520, X: 692]>
```

```
[6]: aics_image.dims.order
```

```
[6]: 'TCZYX'
```

bioio is the successor  
of aicsimageio, but  
seems not to work yet

X and Y: width / height  
Z: Number of images in 3D stack  
T: time-points  
C: channels  
S: scenes

# Loading images

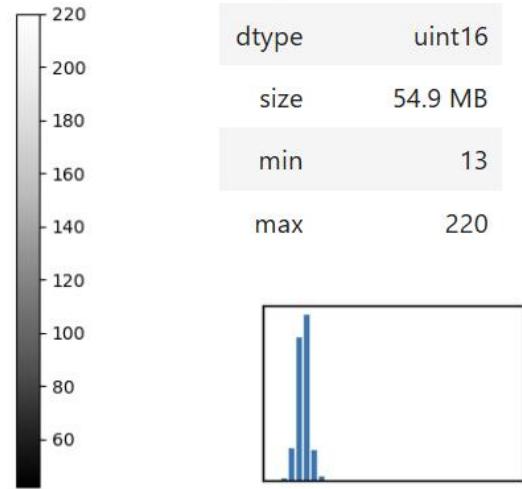
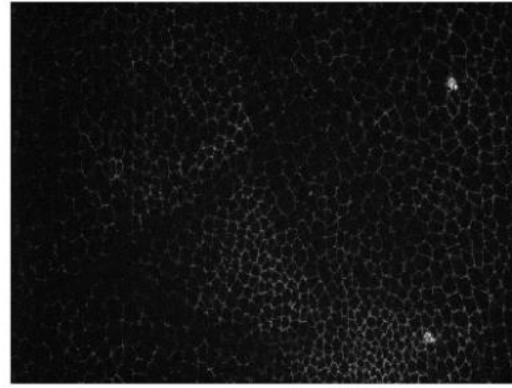
- Before you can work with such images, you need to extract a numpy-array first

```
[6]: np_image = aics_image.get_image_data("ZYX", T=0)  
np_image.shape
```

```
[6]: (80, 520, 692)
```

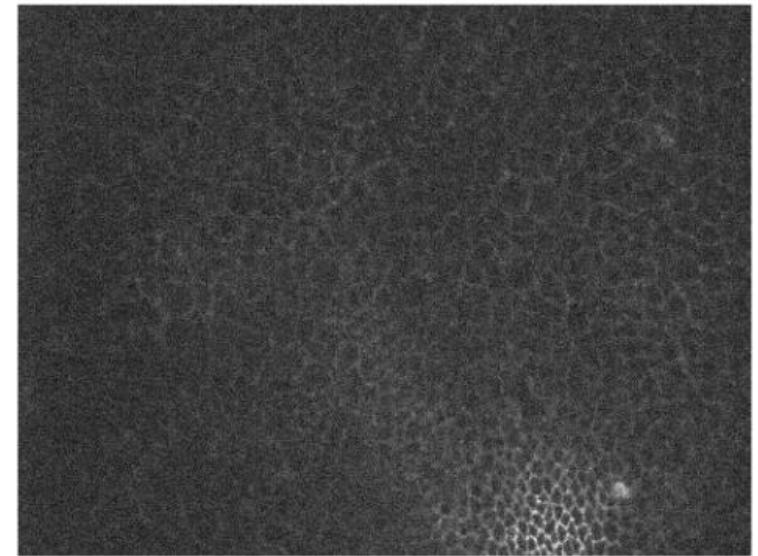
```
[7]: stackview.insight(np_image)
```

```
[7]:
```



Select a 2D plane from  
a 3D image stack

```
[9]: stackview.imshow(np_image[39])
```



# Visualizing images

```
[3]: from skimage.io import imread  
from matplotlib.pyplot import imshow  
import stackview
```

```
image = imread("data/mitosis_mod.tif")  
image
```

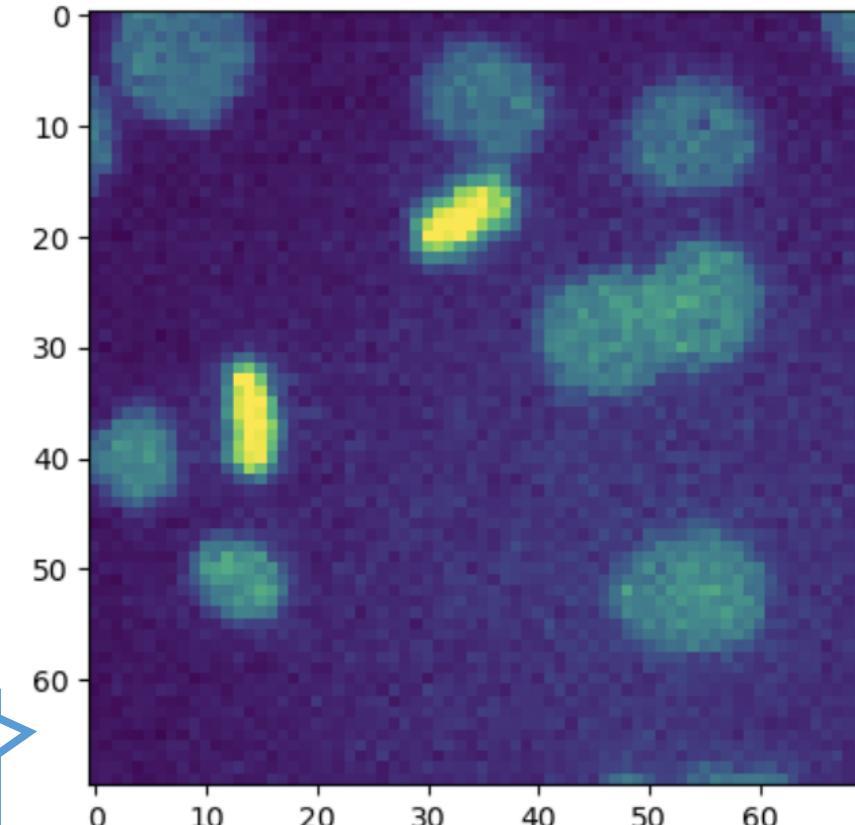
```
[3]: array([[ 19,  29,  44, ...,  88, 115, 113],  
          [ 19,  29,  55, ...,  87, 101, 112],  
          [ 25,  36,  61, ...,  90,  90, 106],  
          ...,  
          [ 20,  21,  20, ...,  57,  33,  40],  
          [ 22,  25,  25, ...,  41,  34,  40],  
          [ 20,  25,  18, ...,  41,  35,  39]], dtype=uint8)
```

Suboptimal image display

The default colormap has been criticized many times.

```
[2]: imshow(image)
```

```
[2]: <matplotlib.image.AxesImage at 0x2cd0e7cad90>
```



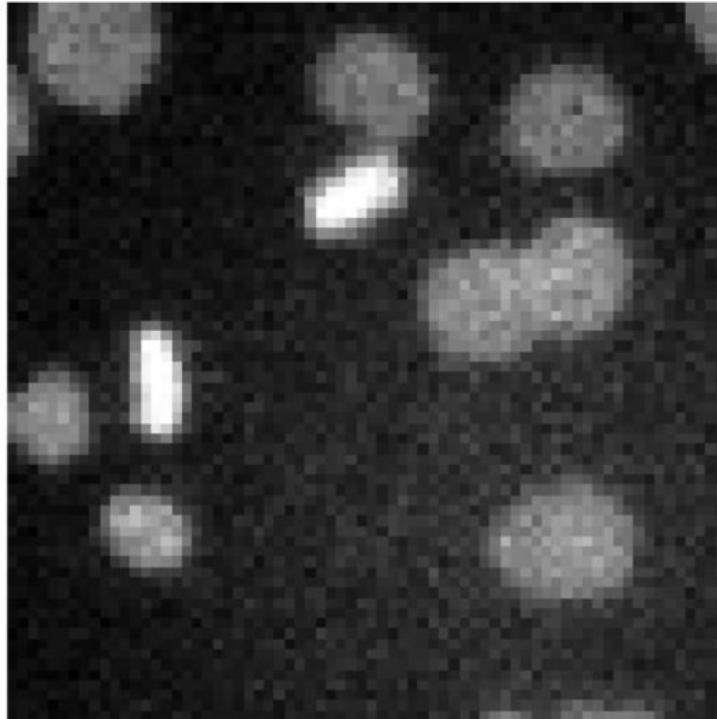
# Visualizing images

- Alternative: stackview

Disclaimer: I maintain this

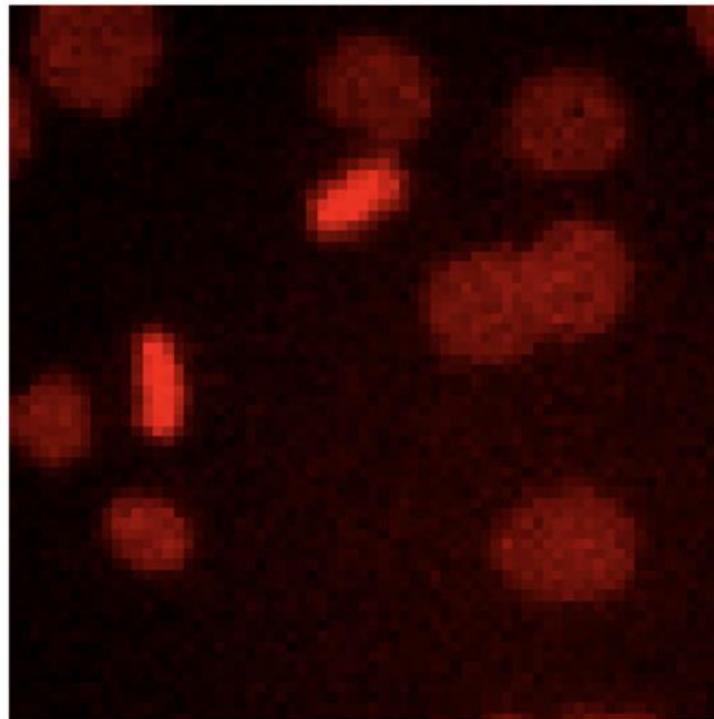
[3]:

```
import stackview  
stackview.imshow(image)
```



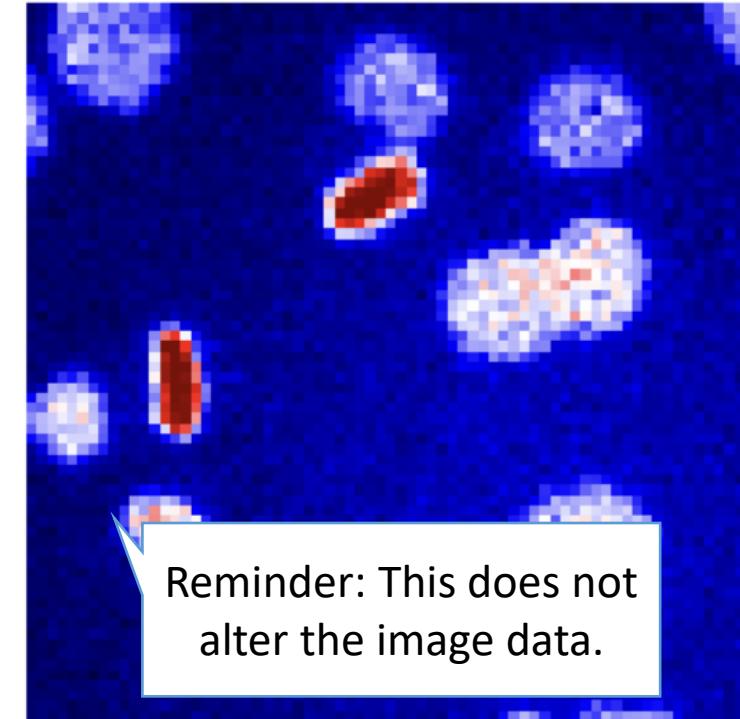
[5]:

```
stackview.imshow(image, colormap="pure_red")
```



[6]:

```
stackview.imshow(image, colormap="seismic")
```



Reminder: This does not alter the image data.

# Visualizing images

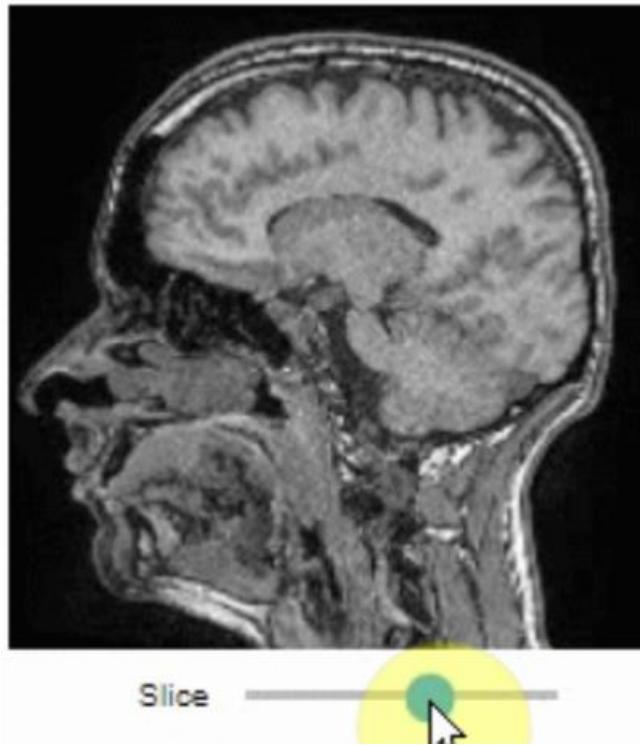
- Alternative: stackview

```
[6]: stackview.slice(mri_image)
```

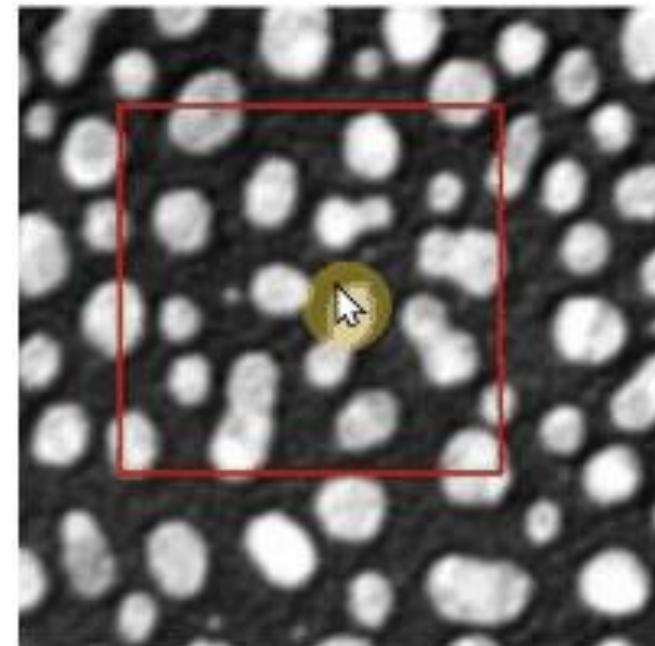


# Visualizing images

```
[6]: stackview.slice(mri_image)
```



```
[8]: stackview.histogram(image)
```

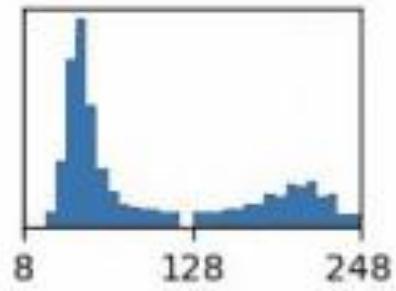


slice (... , 38:184, 39:190)

dtype uint8

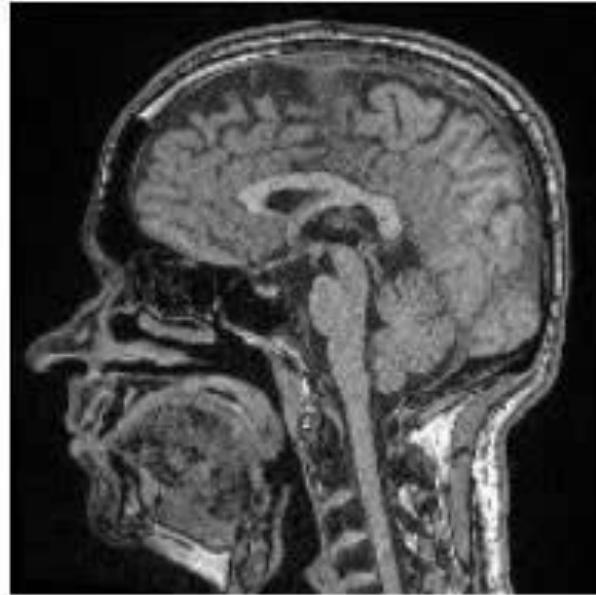
min 8

max 248

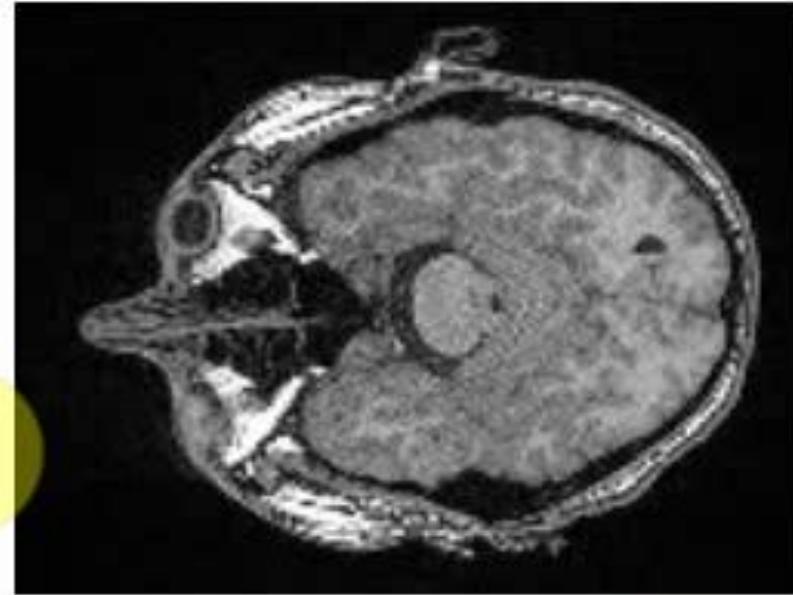


# Orthogonal planes

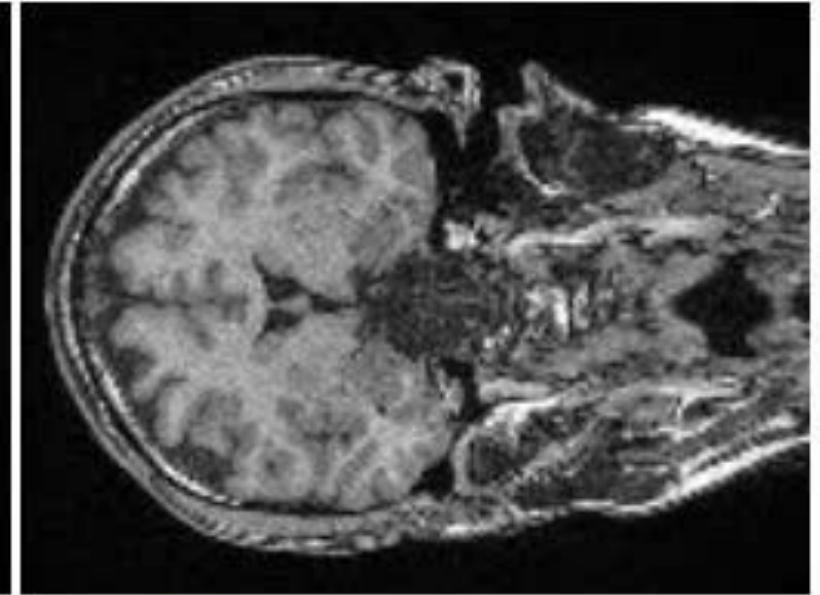
```
[10]: stackview.orthogonal(mri_image, crosshairs=False)
```



Z 60



Y 80



X 80

# Orthogonal planes

Quiz: What does this T stand for?

```
[3]: xy_plane = mri_image[100]  
stackview.imshow(xy_plane)
```



```
[4]: zx_plane = mri_image[:,128,:]  
stackview.imshow(zx_plane)
```



```
[5]: stackview.imshow(zx_plane.T)
```

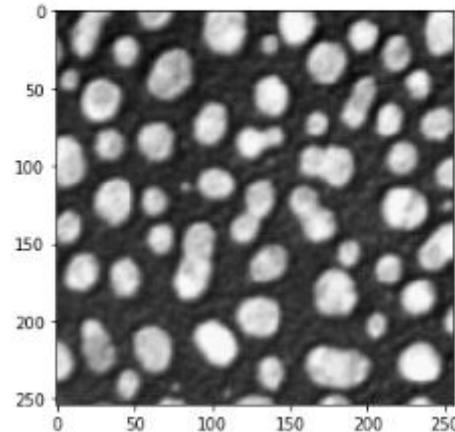


# Cropping and resampling images

- Indexing and cropping *numpy-arrays* works like with python arrays.

```
imshow(image)
```

```
<matplotlib.image.AxesImage at 0x
```

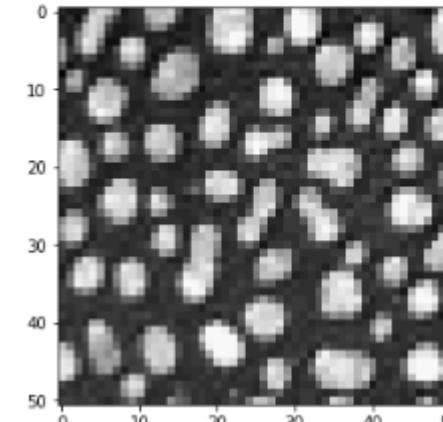


Original image

```
sampled_image = image[::5, ::5]
```

```
imshow(sampled_image)
```

```
<matplotlib.image.AxesImage at 0x
```

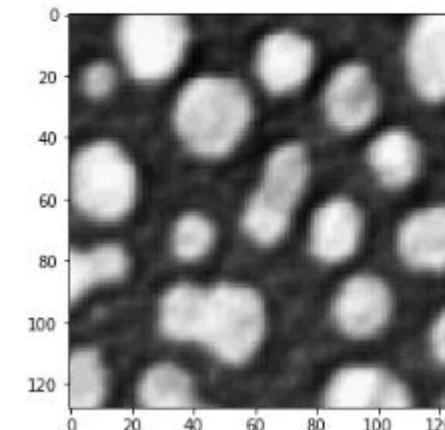


Sub-sampled image

```
cropped_image2 = image[0:128, 128:]
```

```
imshow(cropped_image2)
```

```
<matplotlib.image.AxesImage at 0x29e
```

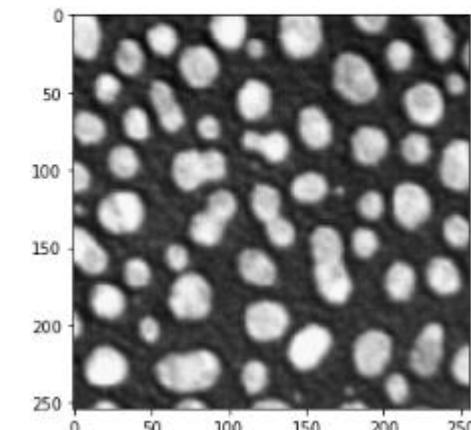


Cropped image

```
flipped_image = image[:, ::-1]
```

```
imshow(flipped_image)
```

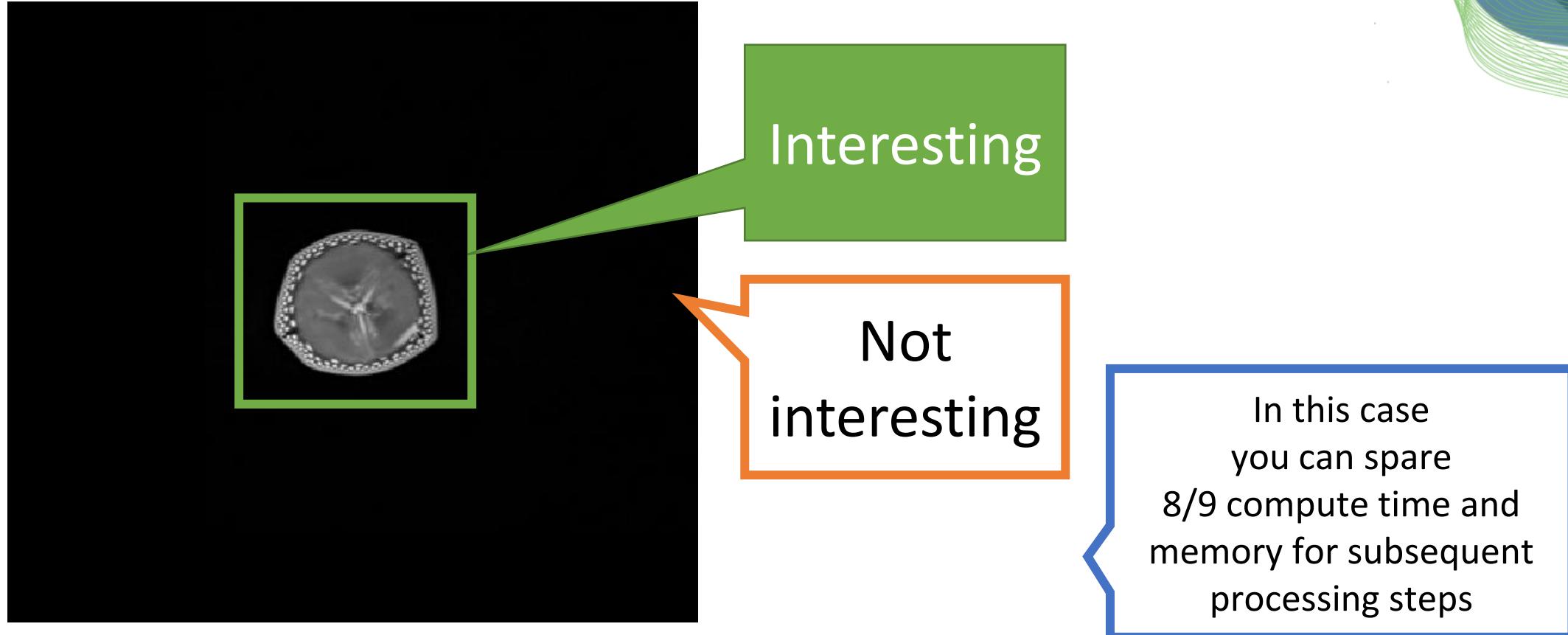
```
<matplotlib.image.AxesImage at 0x
```



Flipped image

# Cropping and resampling images

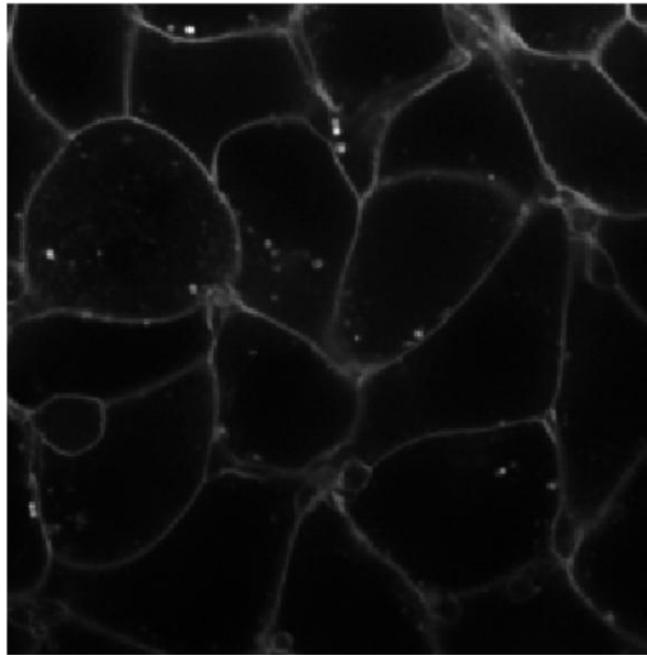
- Crop out the region you're interested in



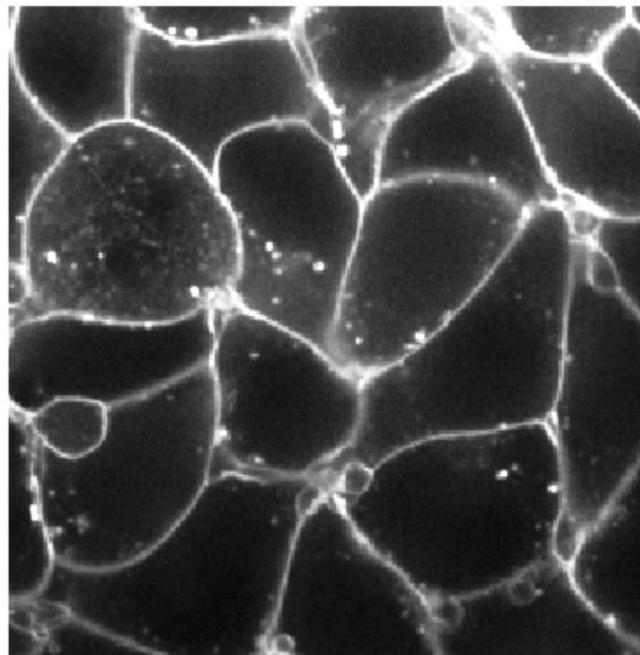
# Brightness, contrast, display-range

- After loading data, make sure you can see the structure you're interested in

```
[5]: stackview.imshow(image)
```



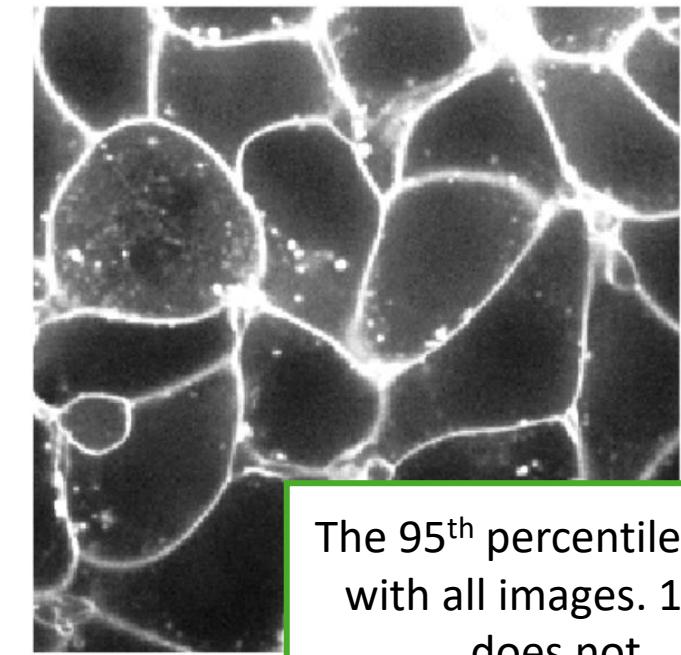
```
[9]: stackview.imshow(image, max_display_intensity=10000)
```



```
[6]: upper_limit = np.percentile(image, 95)  
upper_limit
```

```
[6]: 6580.0
```

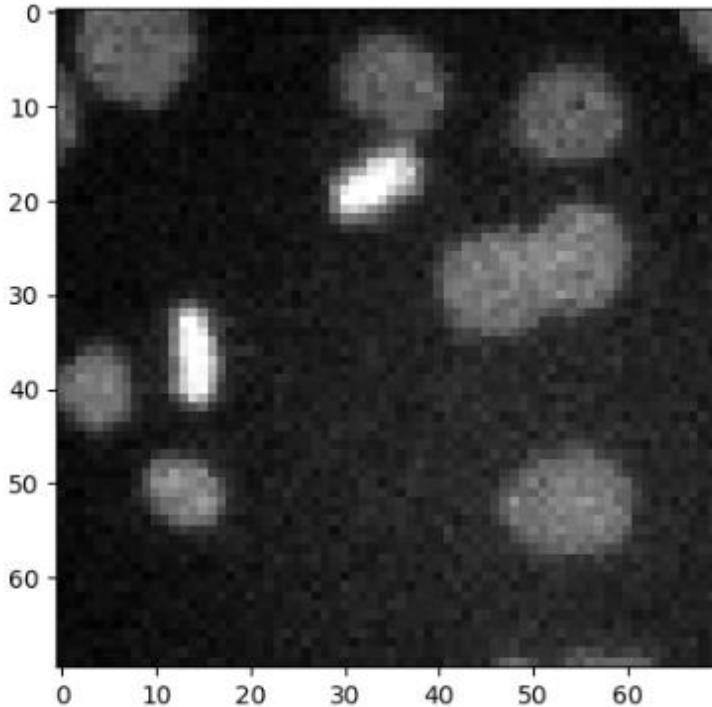
```
[7]: stackview.imshow(image, max_display_intensity=upper_limit)
```



The 95<sup>th</sup> percentile works  
with all images. 10000  
does not.

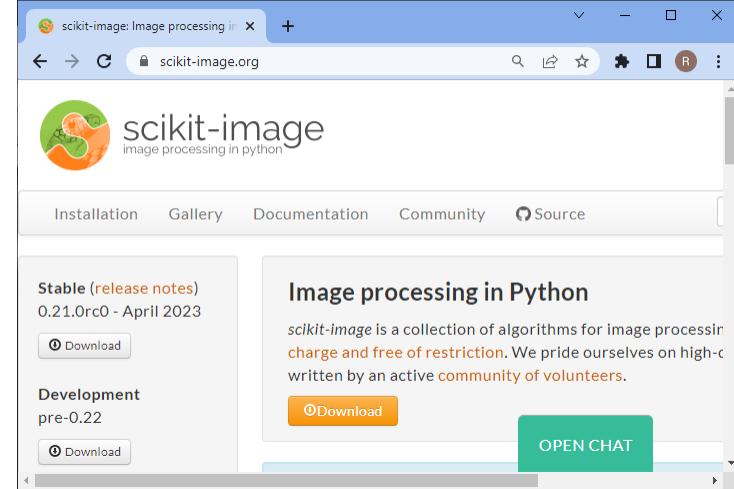
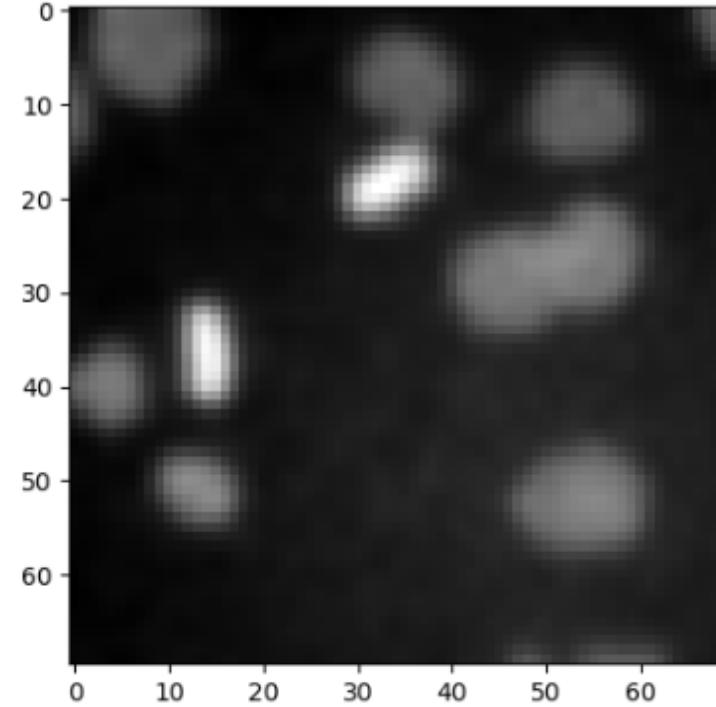
# Filters

... are just functions



```
denoised_gaussian = filters.gaussian(image3, sigma=1)  
plt.imshow(denoised_gaussian, cmap='gray')
```

<matplotlib.image.AxesImage at 0x283aab3ba90>

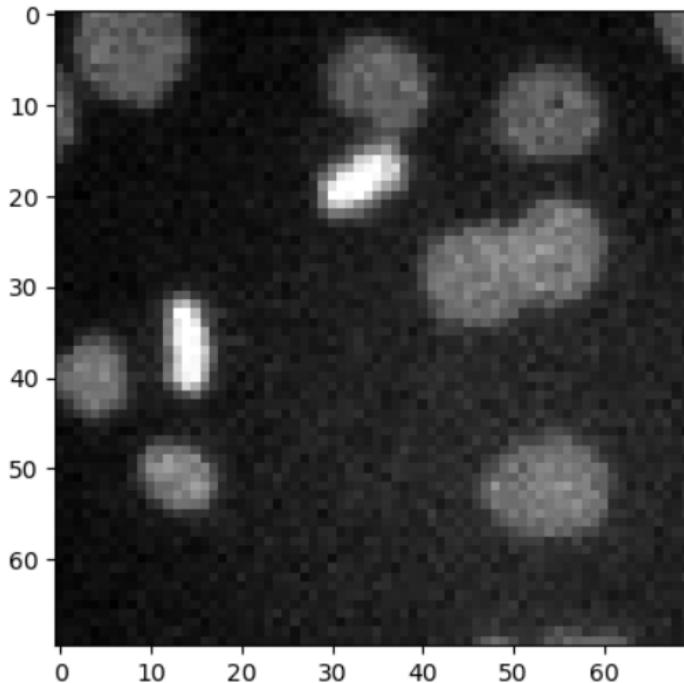


# Filters

- Use every opportunity and play with filter parameters to get an idea what they do.

```
plt.imshow(image3, cmap='gray')
```

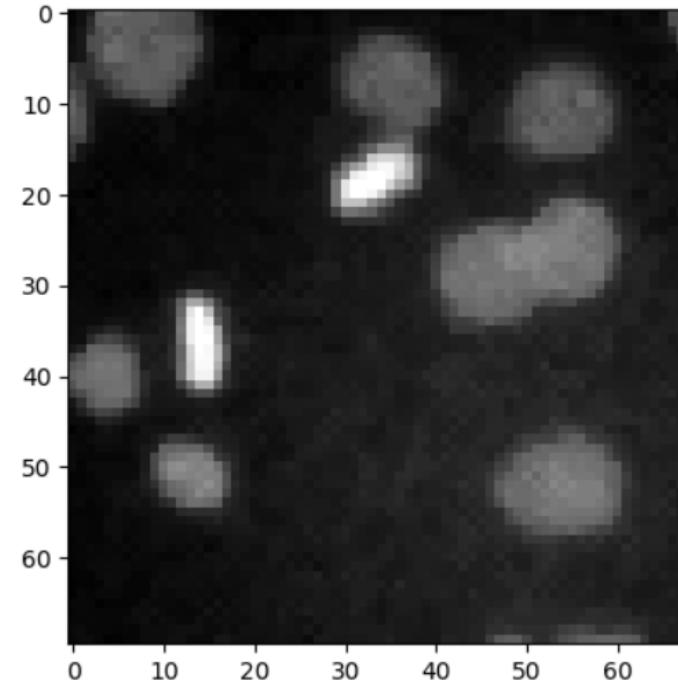
```
<matplotlib.image.AxesImage at 0x1d86893b6d0>
```



```
denoised_median = filters.median(image3, morphology.disk(1))
```

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

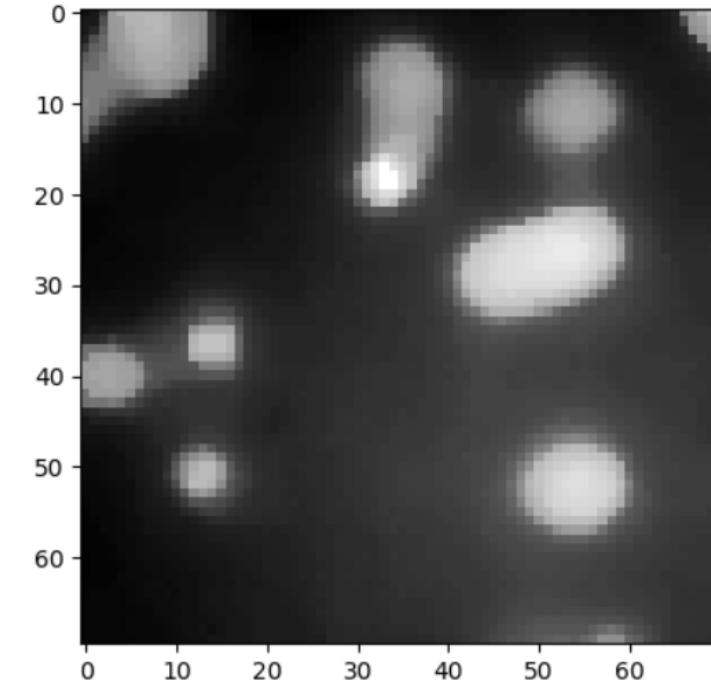
```
<matplotlib.image.AxesImage at 0x1d868a189d0>
```



```
denoised_median2 = filters.median(image3, morphology.disk(5))
```

```
plt.imshow(denoised_median2, cmap='gray')
```

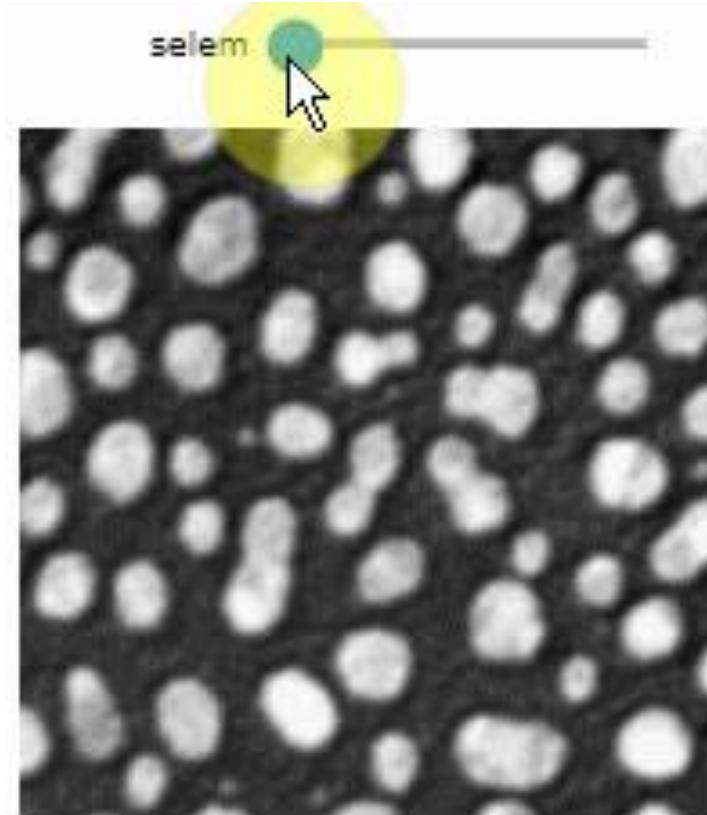
```
<matplotlib.image.AxesImage at 0x1d868ca7af0>
```



# Filters

- Use every opportunity and play with filter parameters to get an idea what they do.

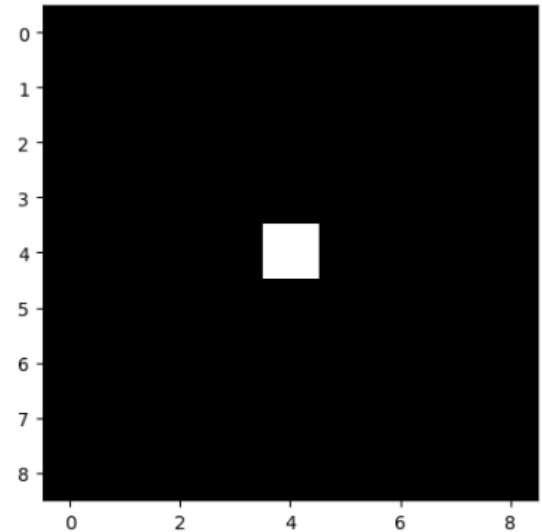
```
from skimage.filters.rank import maximum  
stackview.interact(maximum, slice_image)
```



# Filters

... may be custom functions

Recommendation: Apply custom filters to super simple images to see if they do the right thing.



```
def laplacian_of_gaussian(image, sigma):
    """
    Applies a Gaussian kernel to an image and the Laplacian afterwards.
    """

    # blur the image using a Gaussian kernel
    intermediate_result = filters.gaussian(image, sigma)

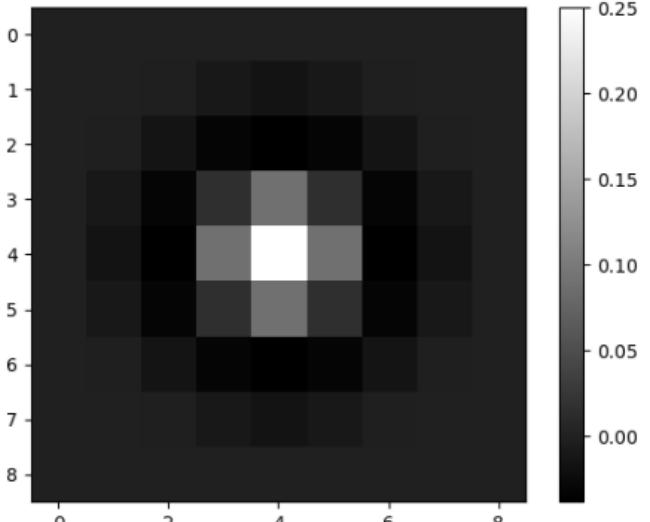
    # apply the mexican hat filter (Laplacian)
    result = filters.laplace(intermediate_result)

    return result
```

```
log_image1 = laplacian_of_gaussian(image2, sigma=1)

plt.imshow(log_image1, cmap='gray')
plt.colorbar()
```

<matplotlib.colorbar.Colorbar at 0x283a9679430>

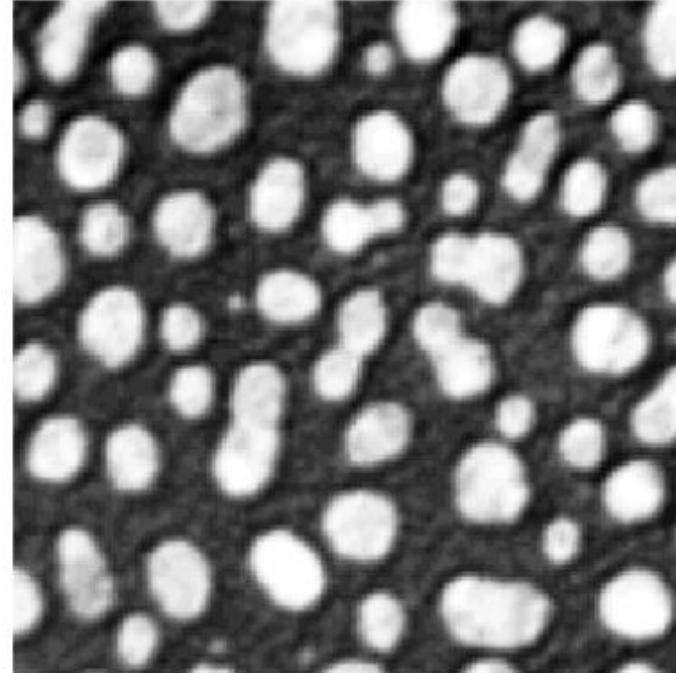


# Binarization / Thresholding

- Turn images into binary images (most basic form of segmentation)
- Basic **math** operation thanks to numpy
- Applied to every pixel individually

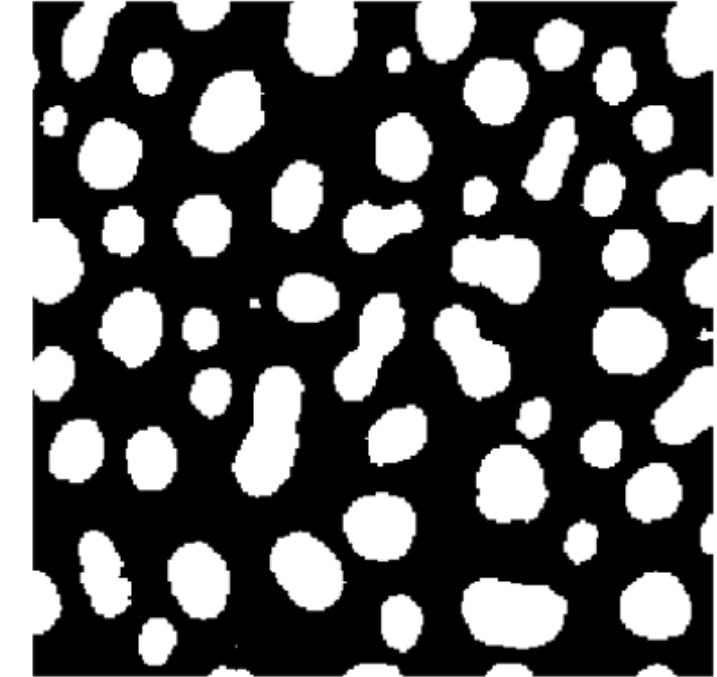
[2]:

```
image = imread("data/blobs.tif")
stackview.imshow(image)
```



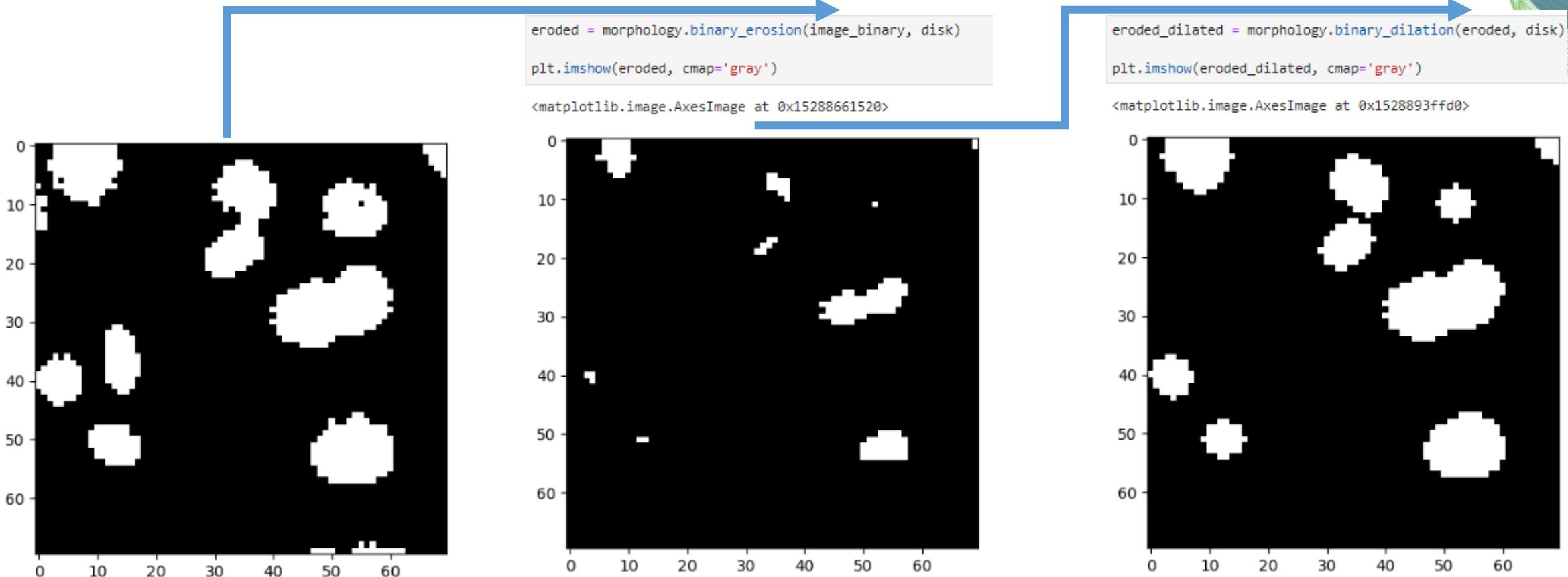
[3]:

```
binary_image = image > 128
stackview.imshow(binary_image)
```



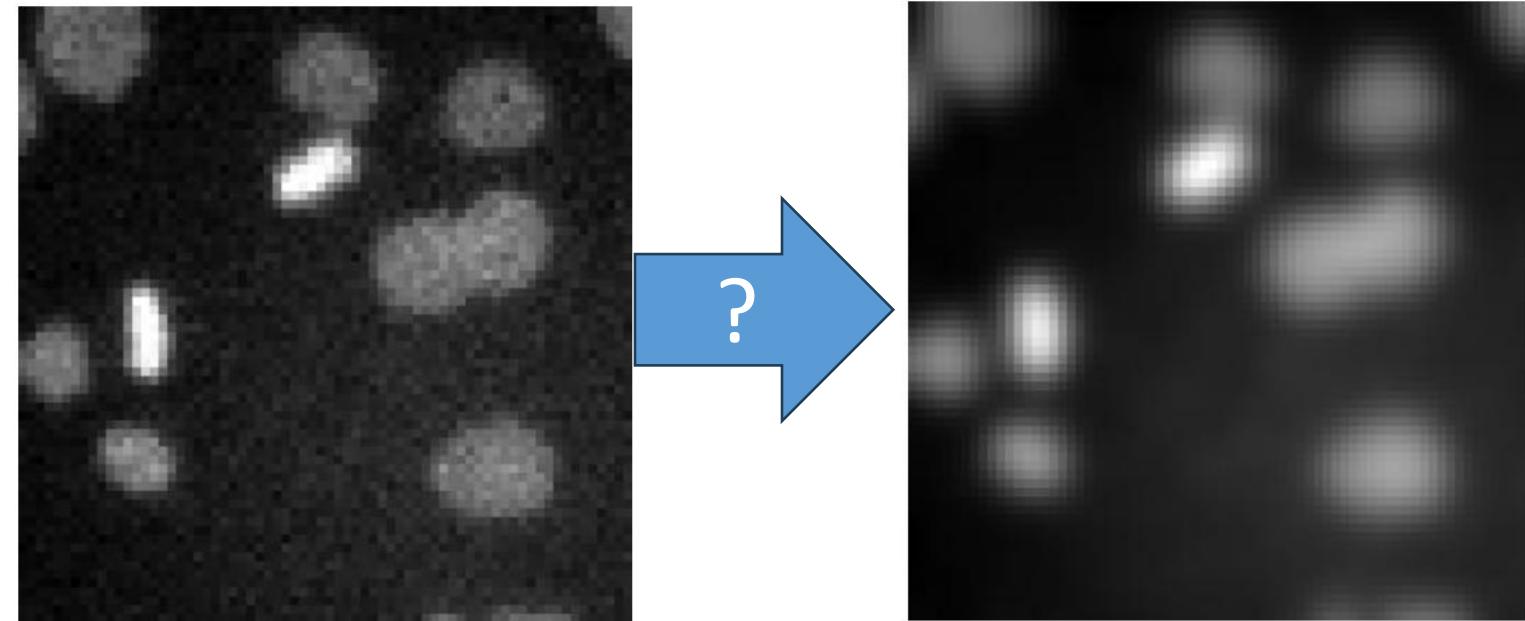
# Morphological operations

- To *morph* objects in binary images



# Quiz: Image processing

Which kind of operation is applied here?



Denoising



Background  
removal



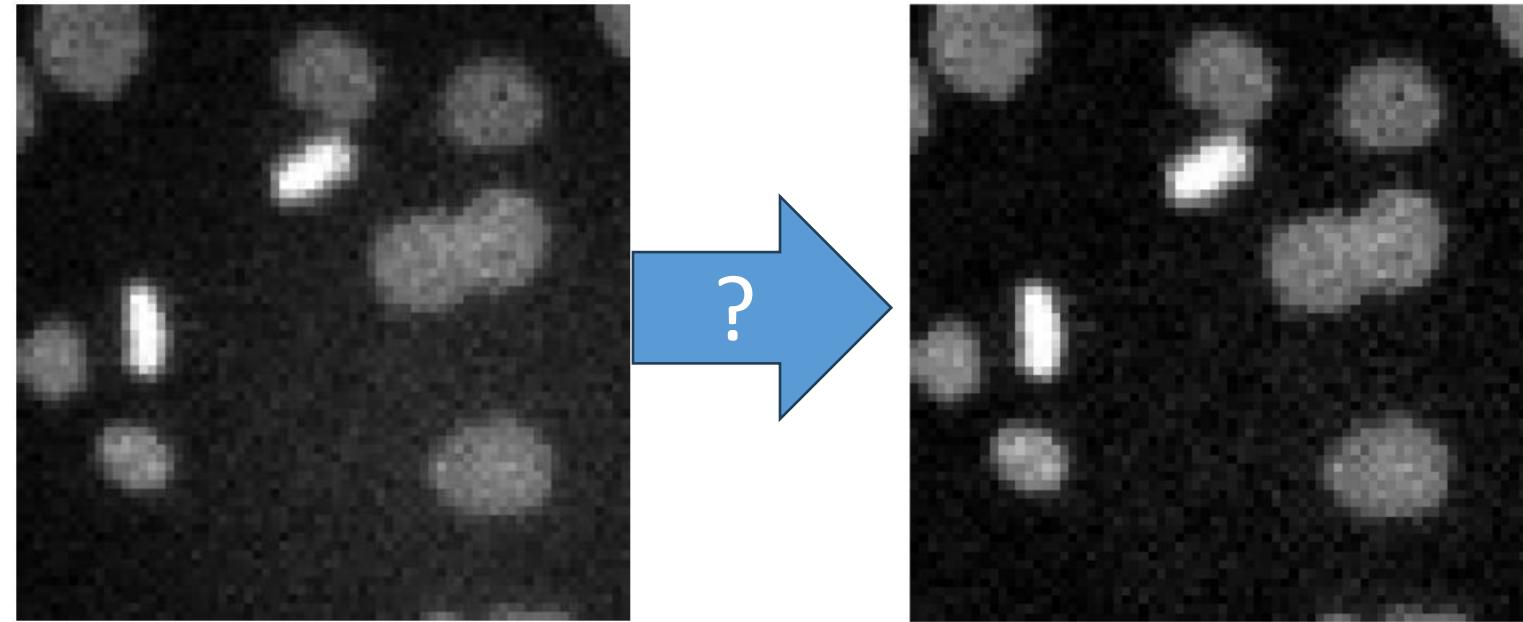
Binarization



Morphological  
operation

# Quiz: Image processing

Which kind of operation is applied here?



Denoising

Background removal

Binarization

Morphological operation





DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS  
AND ARTIFICIAL INTELLIGENCE

# Exercises

Robert Haase

GEFÖRDERT VOM



Bundesministerium  
für Bildung  
und Forschung



Diese Maßnahme wird gefördert durch die Bundesregierung  
aufgrund eines Beschlusses des Deutschen Bundestages.  
Diese Maßnahme wird mitfinanziert durch Steuermittel auf  
der Grundlage des von den Abgeordneten des Sächsischen  
Landtags beschlossenen Haushaltes.

# Hint: Jupyter code hints

- Press SHIFT-Tab within brackets behind a function to read its documentation.

The screenshot shows a Jupyter Notebook cell with the following code:

```
[10]: def custom_median(image, radius:int):
        return filters.median(image, morphology.disk(radius))

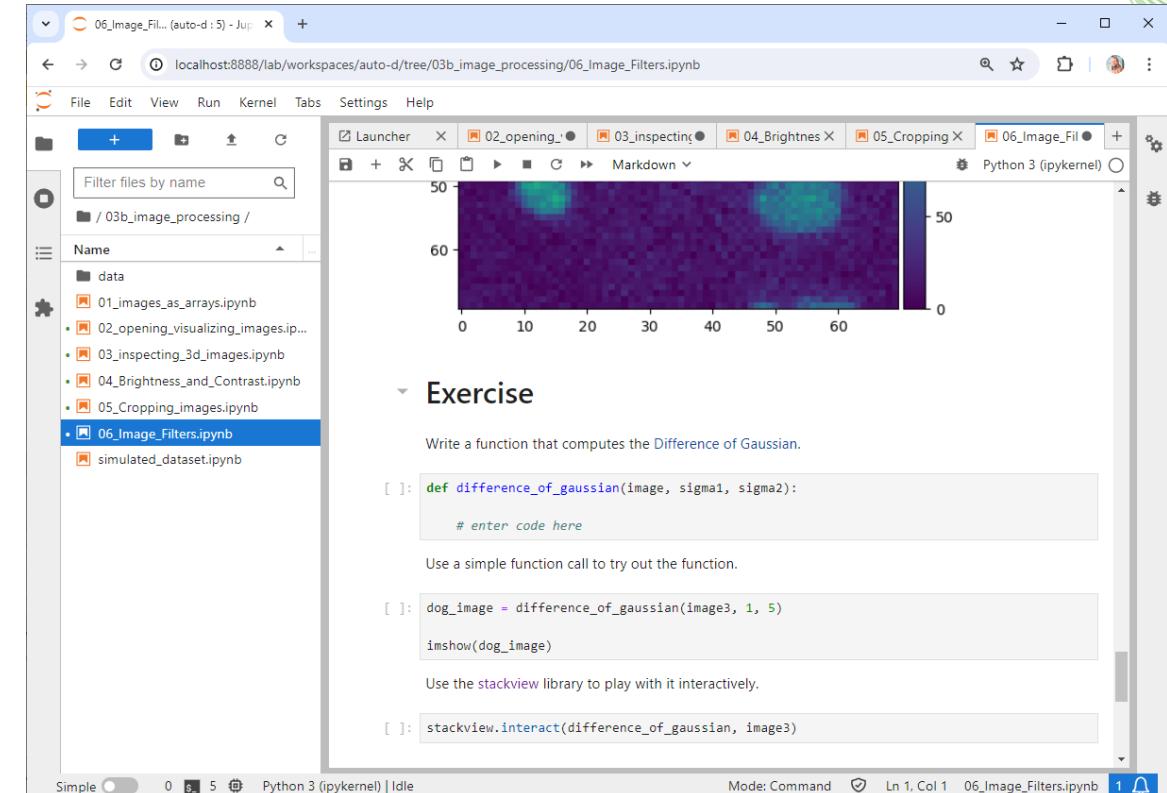
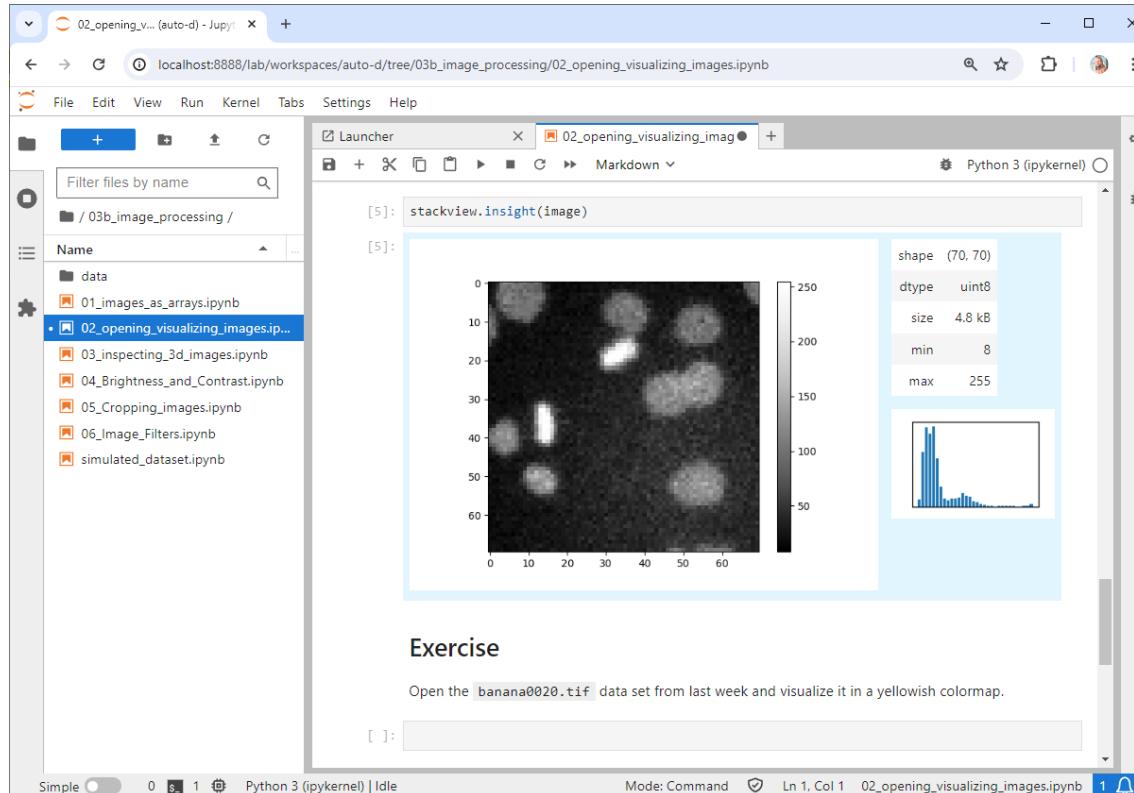
stackview.interact(custom_
```

Below the cell, there is a slider labeled "radius". To the right of the cell, a tooltip provides documentation for the `filters.median` function:

**Signature:**  
filters.median(  
 image,  
 footprint=None,  
 out=None,  
 mode='nearest',  
 cval=0.0,  
 behavior='ndimage',  
)  
**Docstring:**  
Return local median of an image.  
**Parameters**  
-----  
**image** : array-like  
 Input image.  
**footprint** : ndarray, optional  
 If ``behavior=='rank'``, ``footprint`` is a 2-D array of 1's and 0's.  
 If ``behavior=='ndimage'``, ``footprint`` is a N-D array of 1's and 0's  
 with the same number of dimension than ``image``.

# Exercise: image processing

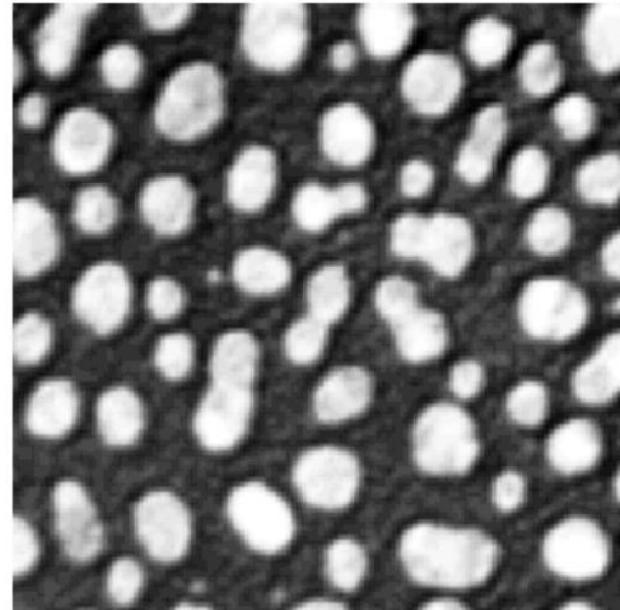
- Get started with loading, viewing, cropping and processing images



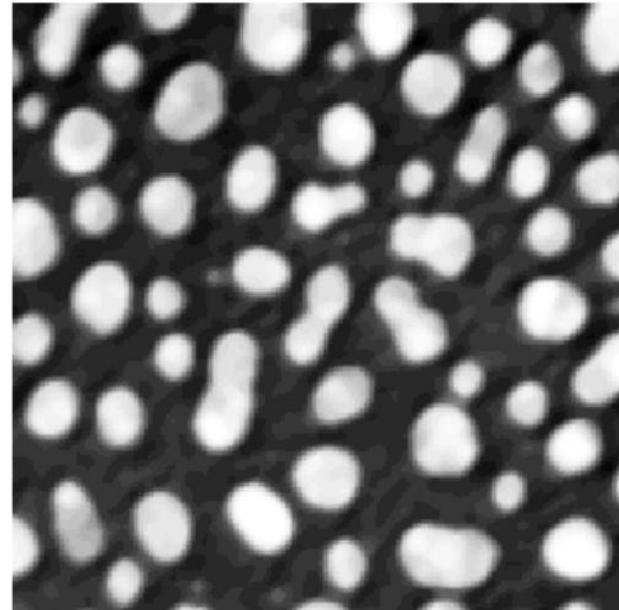
# Exercise: dependencies

- There is a Jupyter Notebook which doesn't work (anymore). Find out why.
- Fix it in two ways:
  - A) by changing the code
  - B) by *not* changing the code

```
[2]: image = imread("data/blobs.tif")
stackview.imshow(image)
```



```
[3]: filtered = median(image, selem=np.ones((7,7)))
stackview.imshow(filtered)
```



# Material repository

The screenshot shows a GitHub repository page for 'BIDS-lecture-2025'. The repository is public and has 3 forks and 3 stars. It contains a 'main' branch with several commits from 'haesleinhuepf' and other users. The 'About' section indicates no description, website, or topics provided. The 'Code' tab is selected, showing a list of files and their details. The 'README' file is linked, and the 'CC-BY-4.0 license' is mentioned.

Commits:

- haesleinhuepf added slides (yesterday)
- 01a\_setting\_up\_local\_enviro... The beginning (2 weeks ago)
- 01b\_setting\_up\_sc\_ulei\_environ... Update readme.md (2 weeks ago)
- 01c\_testing\_environment The beginning (2 weeks ago)
- 02a\_image\_processing added slides (yesterday)
- 02b\_dependency\_managem... added exercises for 2nd lecture (yesterday)
- .gitignore added exercises for 2nd lecture (yesterday)

About

No description, website, or topics provided.

- Readme
- CC-BY-4.0 license
- Activity
- Custom properties
- 3 stars
- 3 watching
- 0 forks

Report repository

Today: 9 notebooks. Invest your time wisely.

- [Microscopy Image Processing](#) (April 25th 2025)
  - Image visualization
  - Image filtering
  - Morphological operations
  - Exercises:
    - [Images are just arrays of numbers](#)
    - [Opening images and applying colormaps](#)
    - [Opening and handling proprietary file formats](#)
    - [Image windowing](#)
    - [Cropping images](#)
    - [Inspecting 3D images](#)
    - [Image processing filters](#)
    - [Basic thresholding](#)
    - [Morphological operations](#)