PoL
Physics of Life
TU Dresden

CENTER FOR
SYSTEMS BIOLOGY
DRESDEN

Image Processing: Filters

Till Korten

With material from

Robert Haase, Pol; Mauricio Rocha Martins, Norden lab, MPI CBG; Dominic Waithe, Oxford University; Benoit Lombardot, Scientific Computing Facility, MPI CBG; Alex Bird, Dan White, MPI CBG

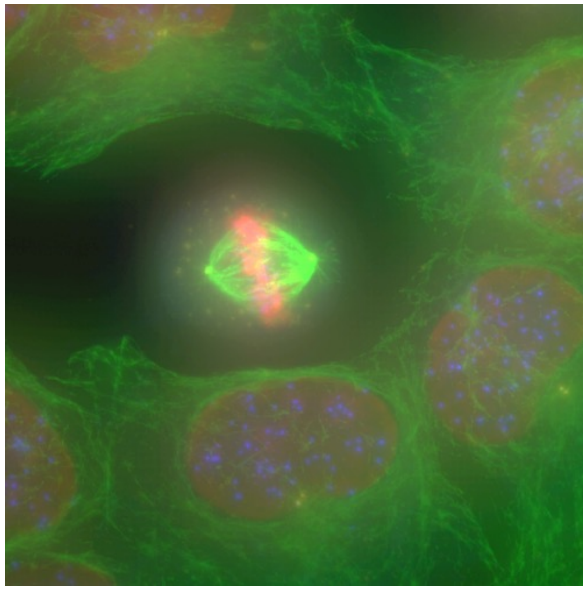
December 2022

@TillKorten

Filters are usually used to enhance features of interest

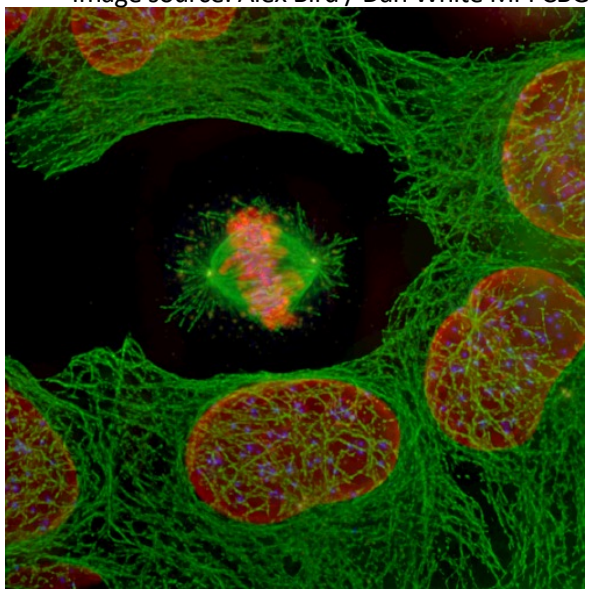
PoL Physics of Life TU Dresden

- 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.



Filter

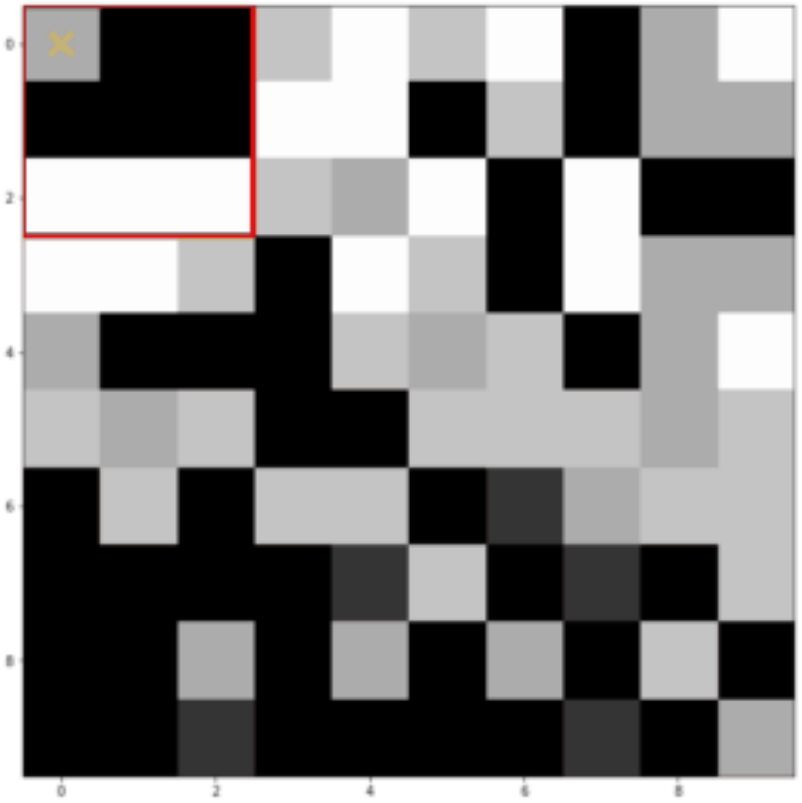
Image source: Alex Bird / Dan White MPI CBG



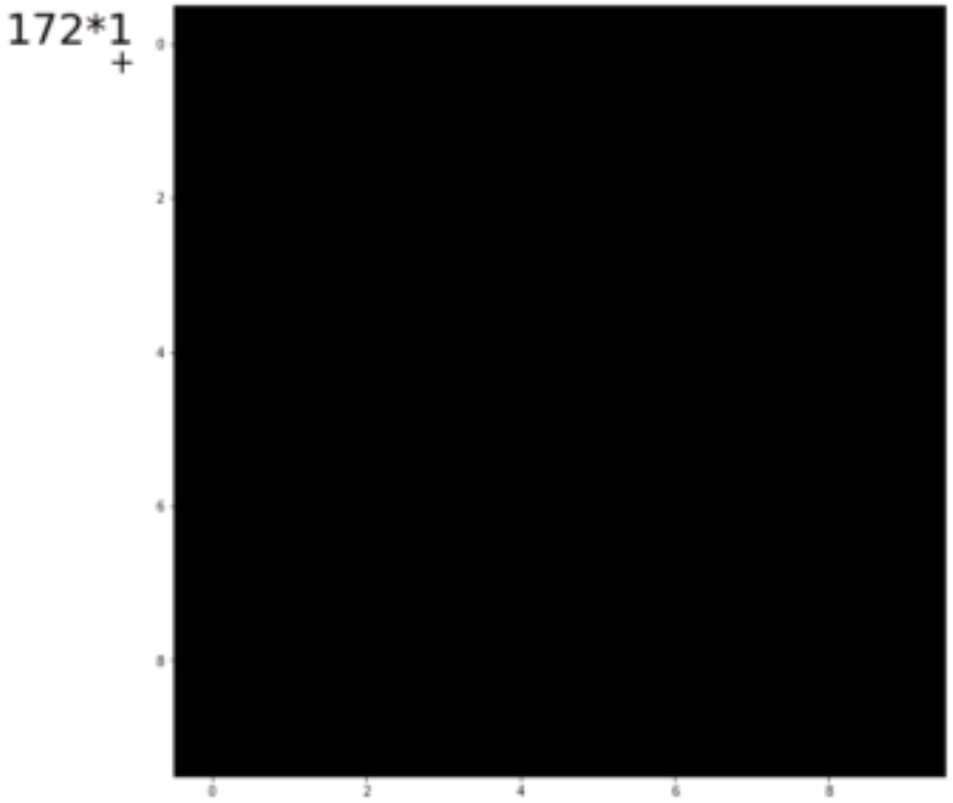
Filters apply an algorithm to each pixel of an image

Example: mean filter

Input image



Output image

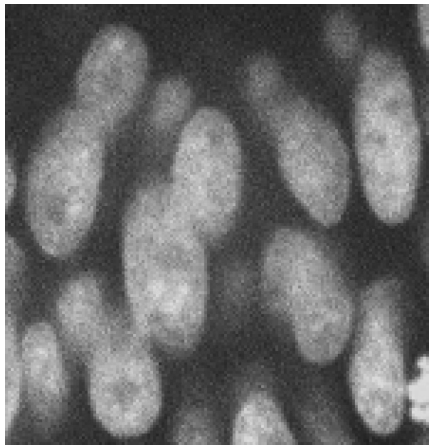


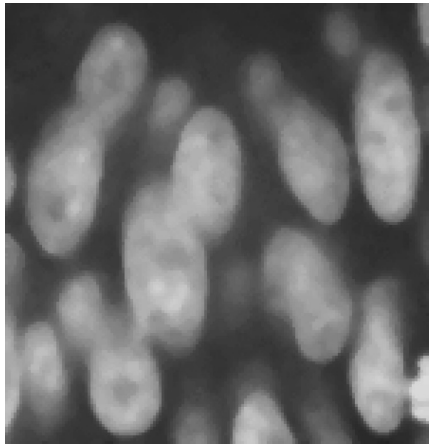
Animation source: Dominic Waithe, Oxford University
https://github.com/dwaithe/generalMacros/tree/master/convolution_ani

Common applications for image filters

PoL Physics of Life TU Dresden

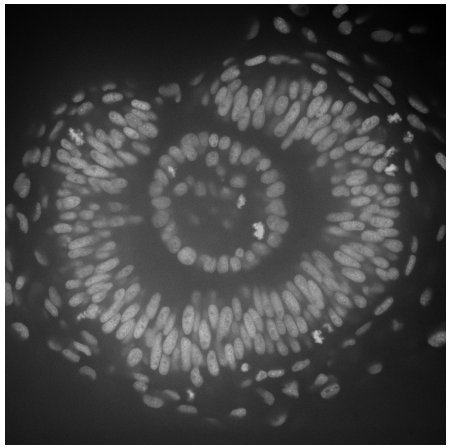
Noise reduction

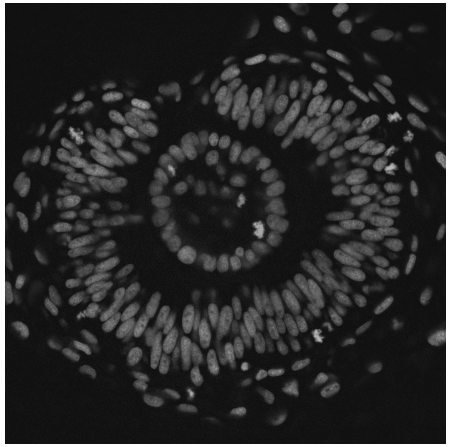




Mauricio Rocha Martins (Norden/Myers lab, MPI CBG)

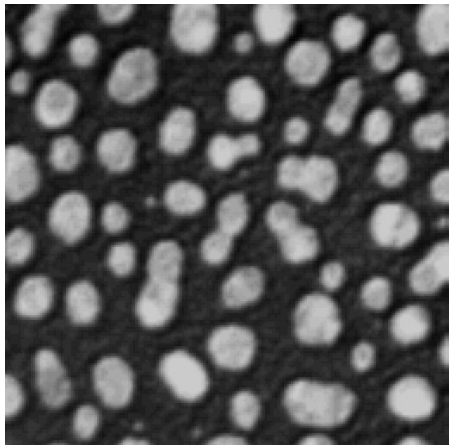
Background removal

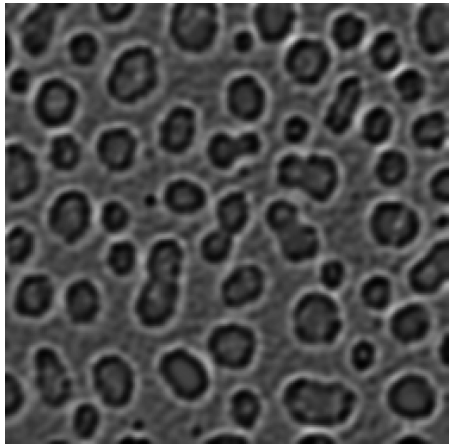




Mauricio Rocha Martins (Norden/Myers lab, MPI CBG)

Edge detection





https://imagej.nih.gov/ij/images/

De-convolution

Stub Mandrel, CC BY-SA 4.0, via Wikimedia Commons

Noise reduction

- In microscopy, noise is usually comprised of
  - shot noise: Statistical variation of the photons arriving at the camera
  - dark noise: noise produced by thermal fluctuations in the camera chip
  - read out noise: introduced by the electronics, especially the analog-digital-converter

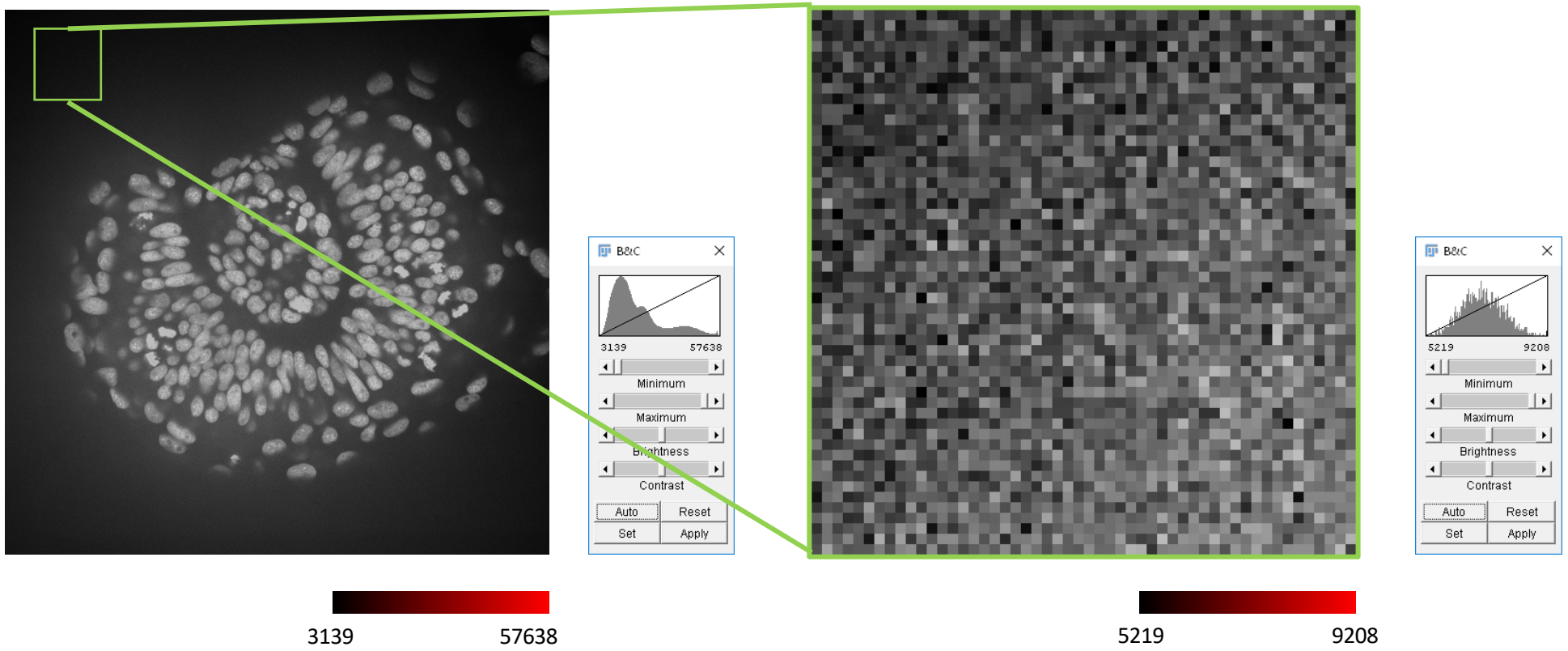


Image source: Mauricio Rocha Martins (Norden/Myers lab, MPI CBG)

Reducing noise by lightly blurring the image

PoL Physics of Life TU Dresden

```
from skimage.filters import median, gaussian
from skimage.morphology import disk

zfish_gaussian = gaussian(cropped_zfish, sigma=1.5, preserve_range=True)
zfish_median = median(cropped_zfish, footprint=disk(1.5))
```

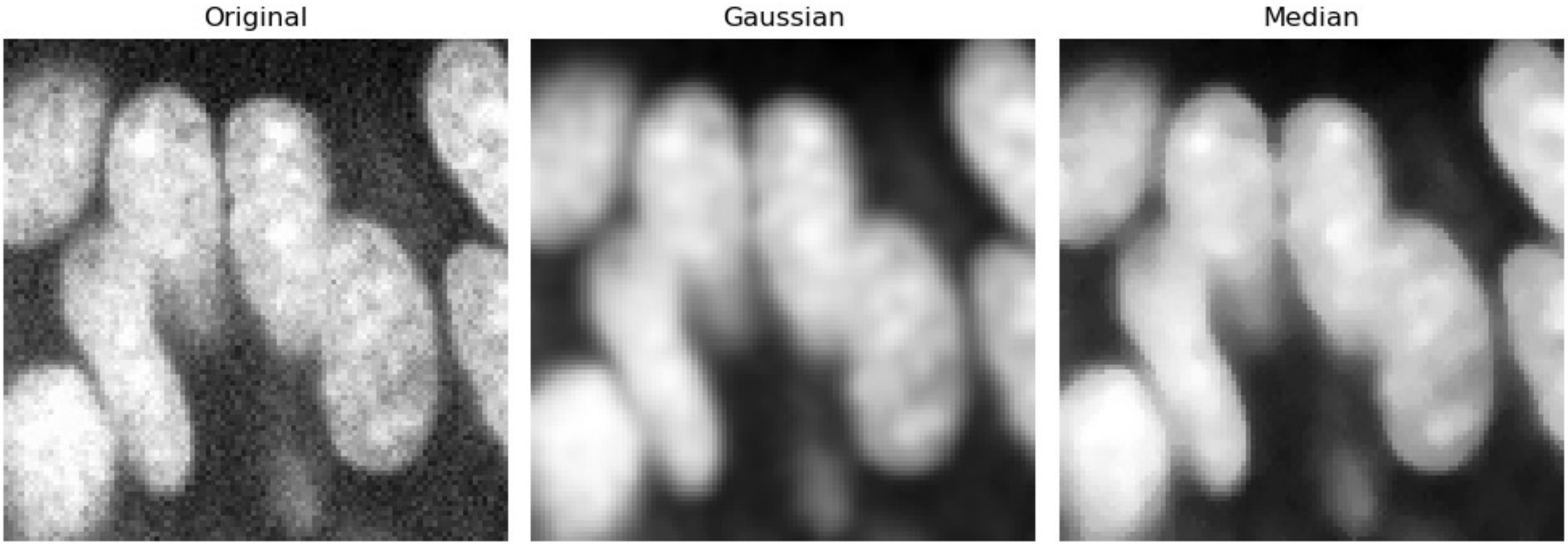
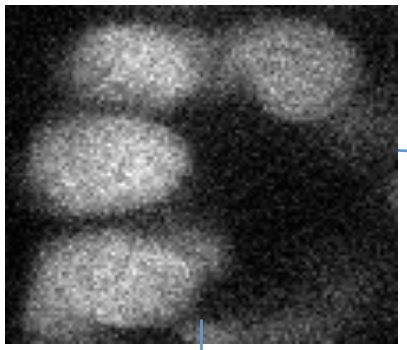


Image source: Mauricio Rocha Martins (Norden/Myers lab, MPI CBG)

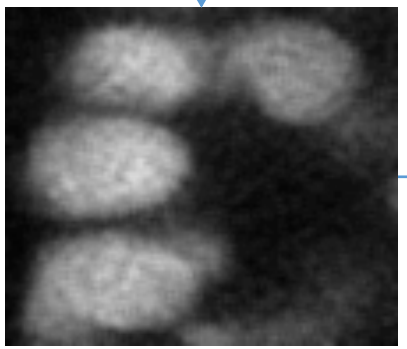
Removing noise helps the computer *interpret* the image



Thresholding

Oh no! I see thousands of tiny white objects!

Blurring



Thresholding

Ok, it's just 4 objects.

Background removal

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

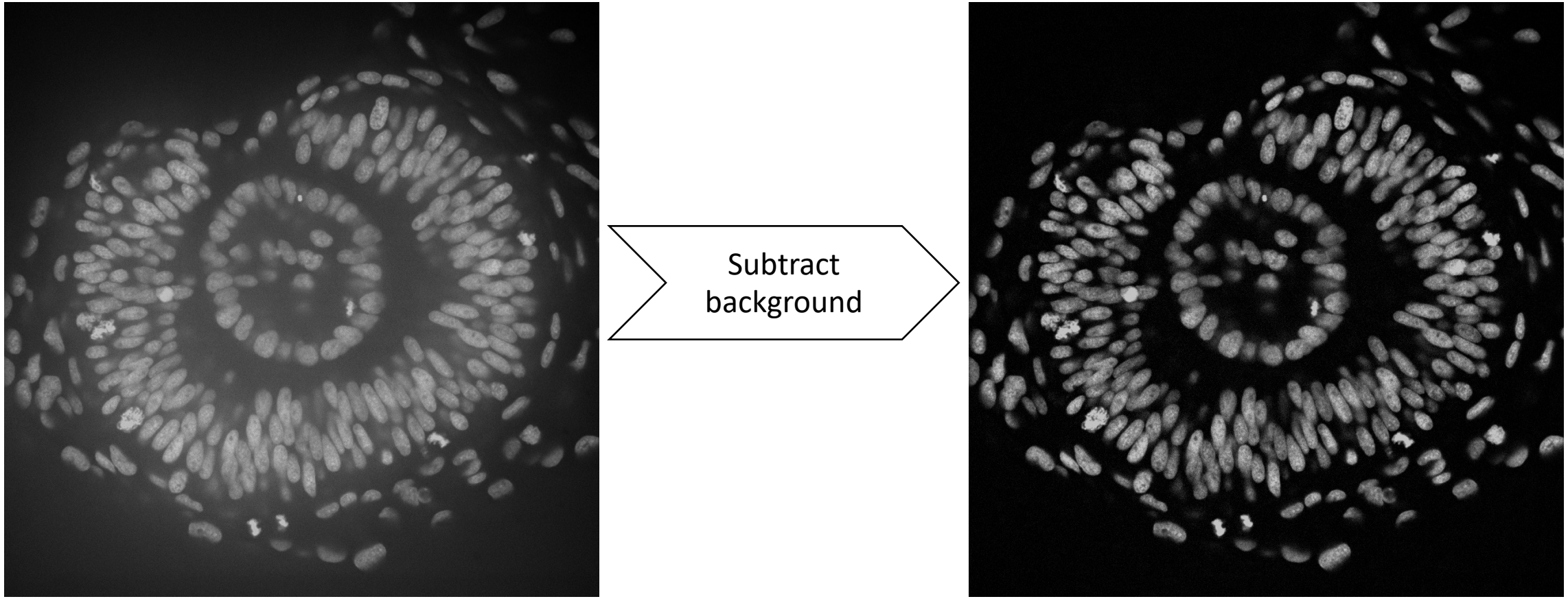


Image source: Mauricio Rocha Martins (Norden/Myers lab, MPI CBG)

Removing background with the white_tophat filter

PoL Physics of Life TU Dresden

```
from skimage.morphology import disk, white_tophat

zfish_tophat = white_tophat(zfish_image, footprint=disk(50))

background_tophat = zfish_image - zfish_tophat
```

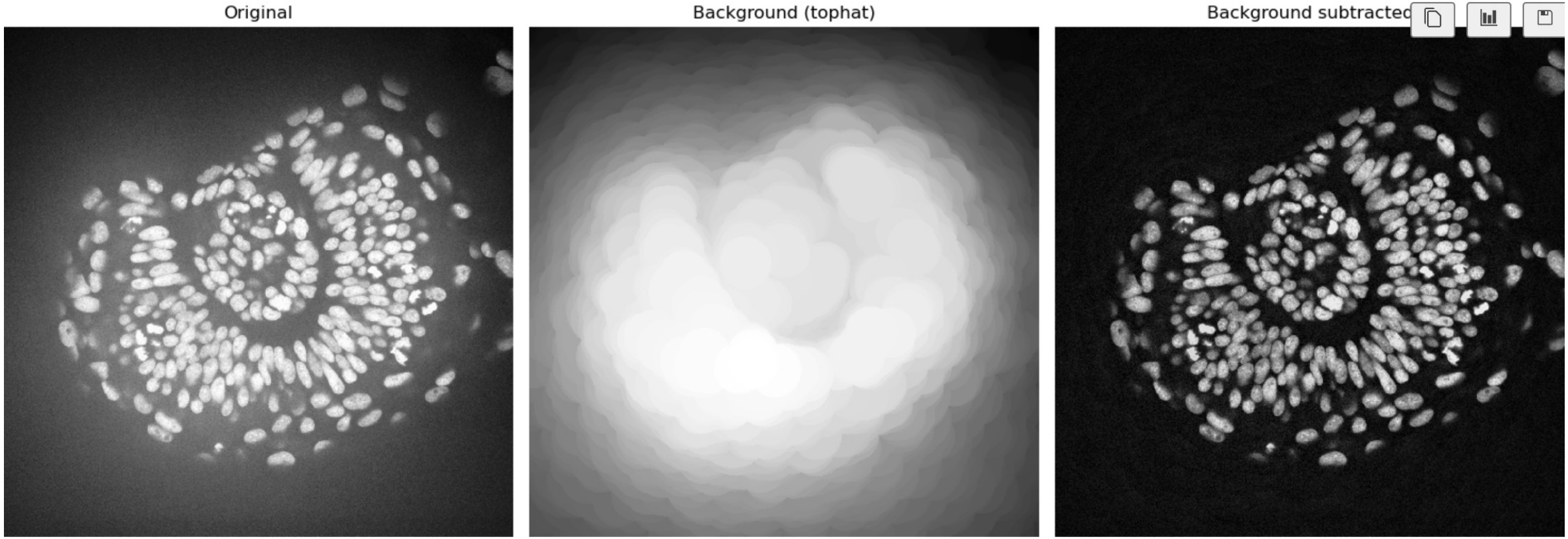
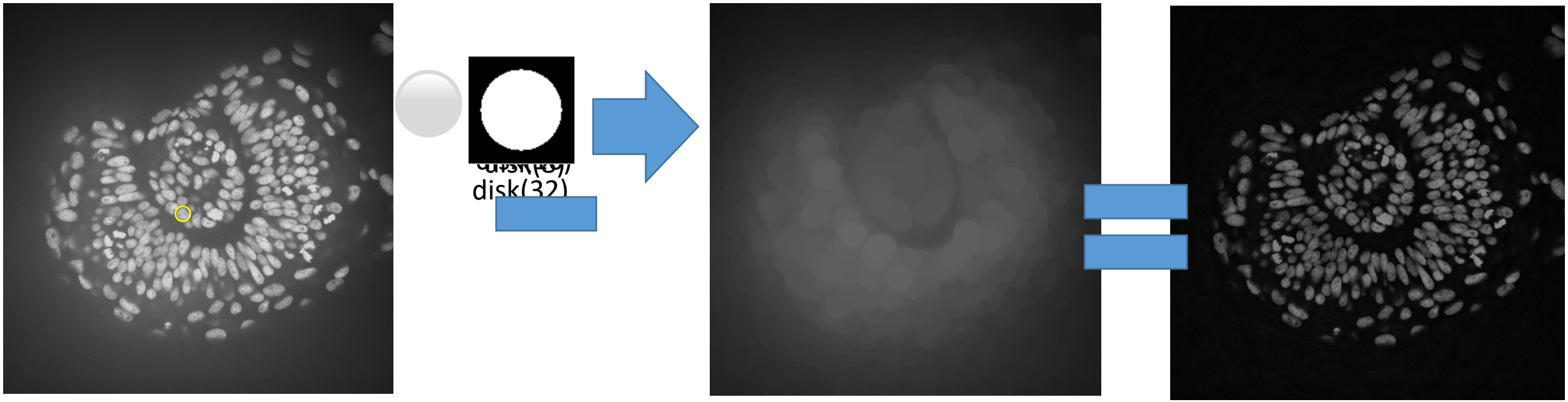


Image source: Mauricio Rocha Martins (Norden/Myers lab, MPI CBG)

The effect of footprint size

white_tophat



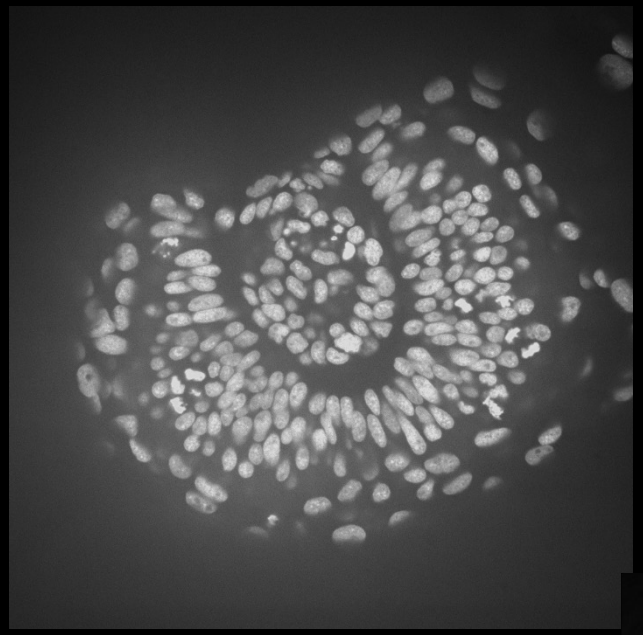
Original image

This is a good estimation of the background

Structures have a radius ≈ 12

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

The effect of footprint size



What happens for a small structuring element (e.g., disk (2))?

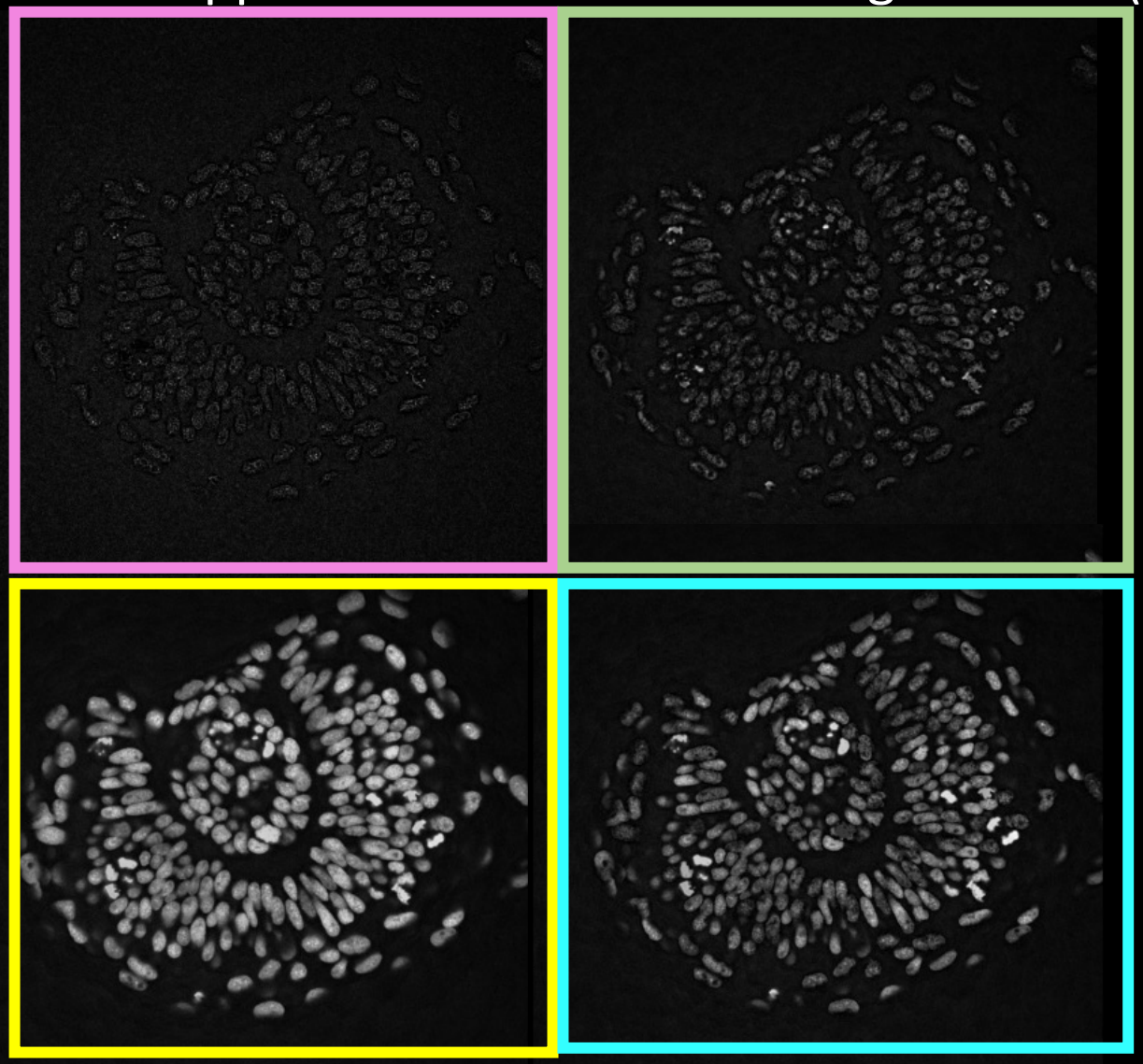
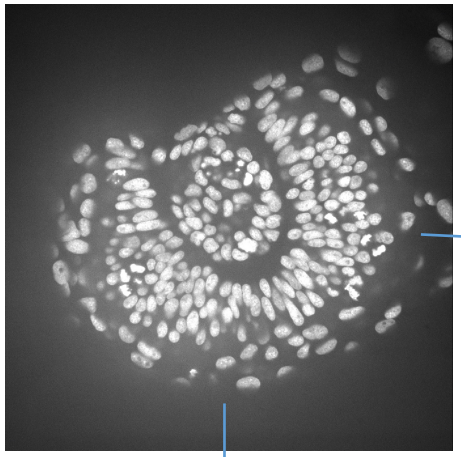
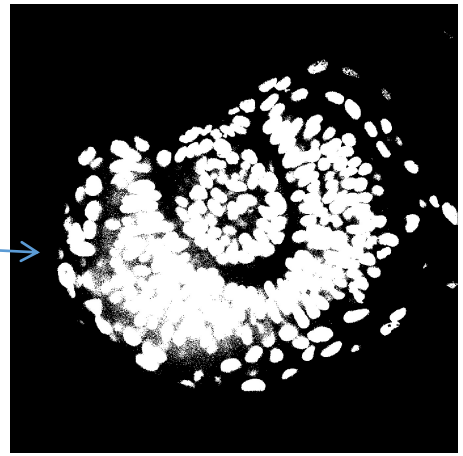


Image source: Mauricio Rocha Martins (Norden/Myers lab, MPI CBG)

Removing background helps the computer *interpret* the image

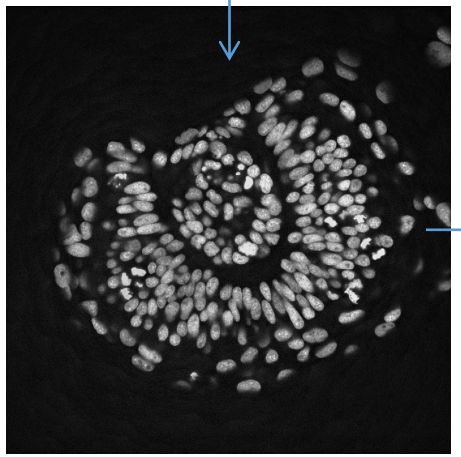


Thresholding

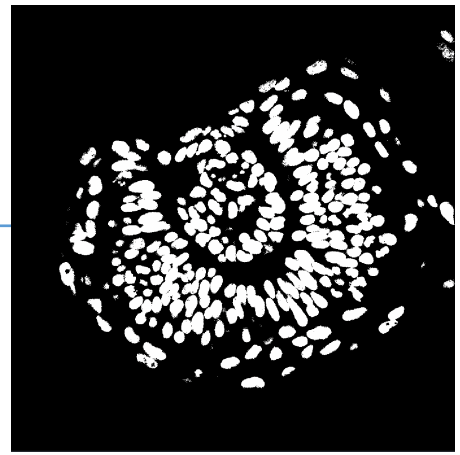


Oh no! I cannot distinguish some objects!

Background removal



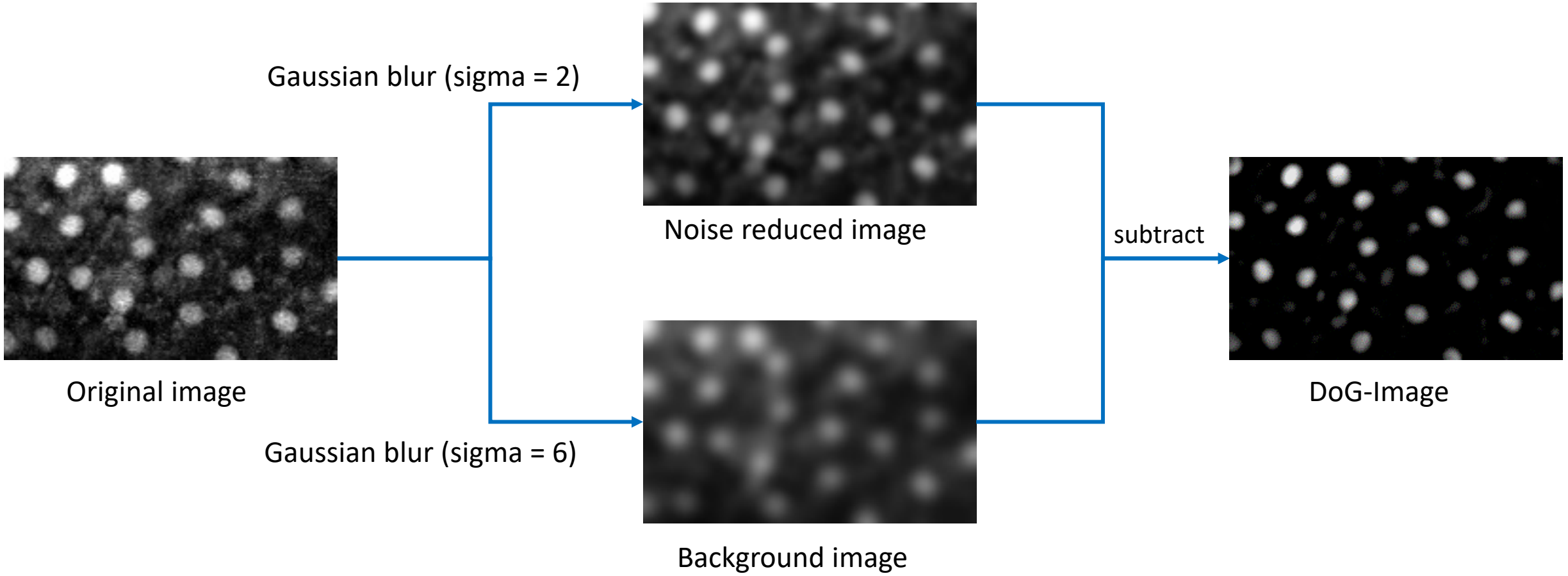
Thresholding



Ok, now I can distinguish most objects.

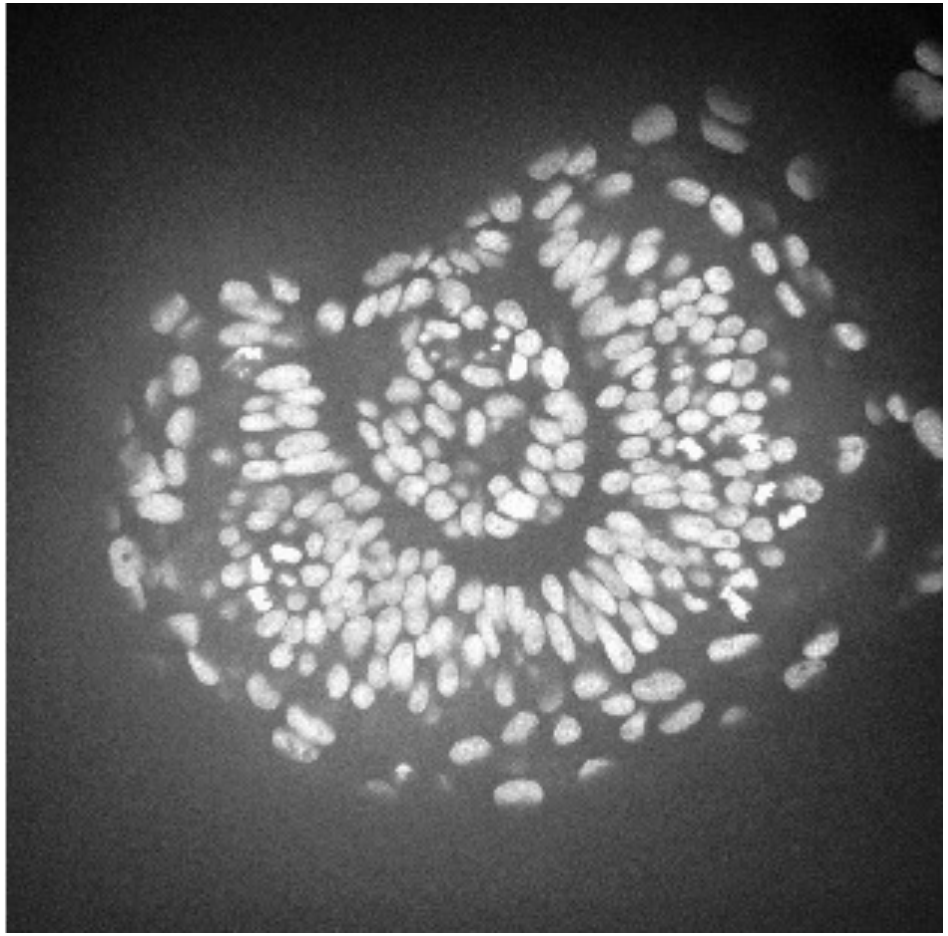
Difference-of-Gaussian

- Combine noise reduction with background removal.

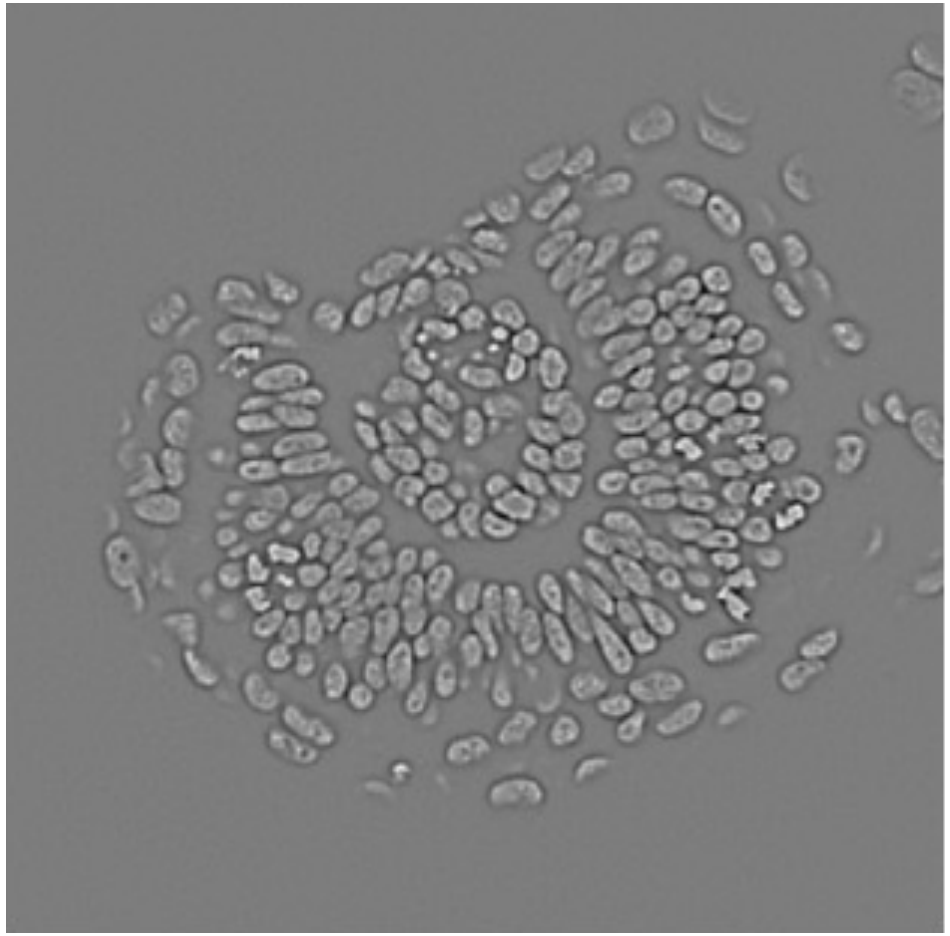


Edge detection

- Detecting the edges of objects can be useful in identifying object boundaries



Detect edges

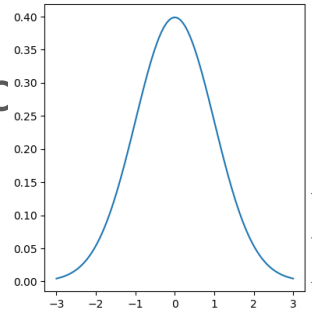


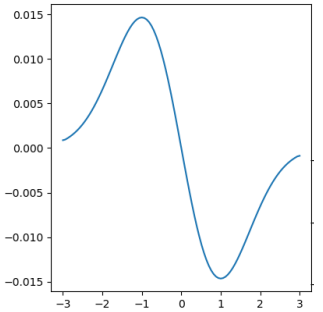
The Laplace-filter is frequently used to detect edges

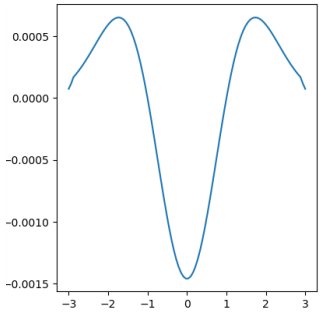
- *Second derivative of a Gaussian blur filter*
- Used for edge-detection and edge enhanc
- Also known as the Mexican-hat-filter

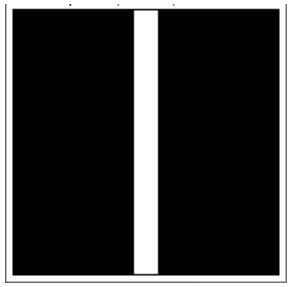
1st derivative

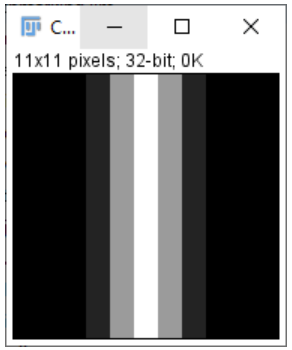
2nd derivative

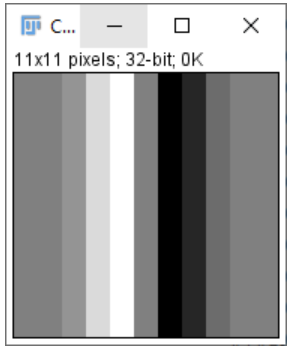


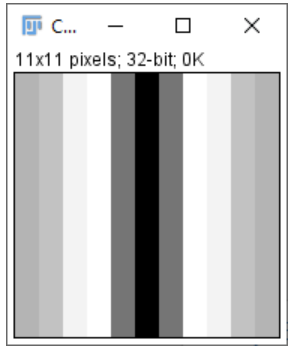


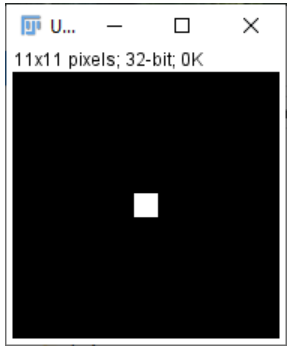


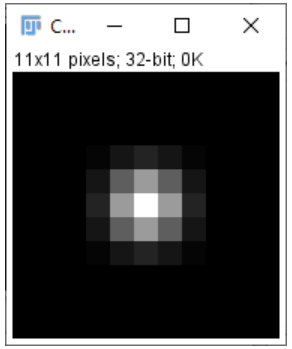


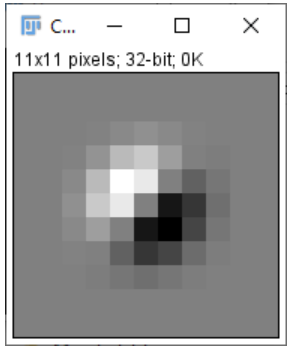


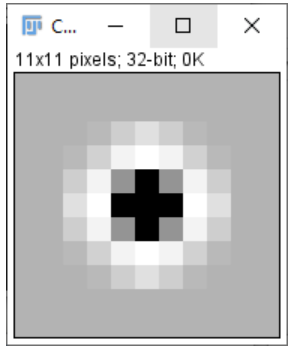




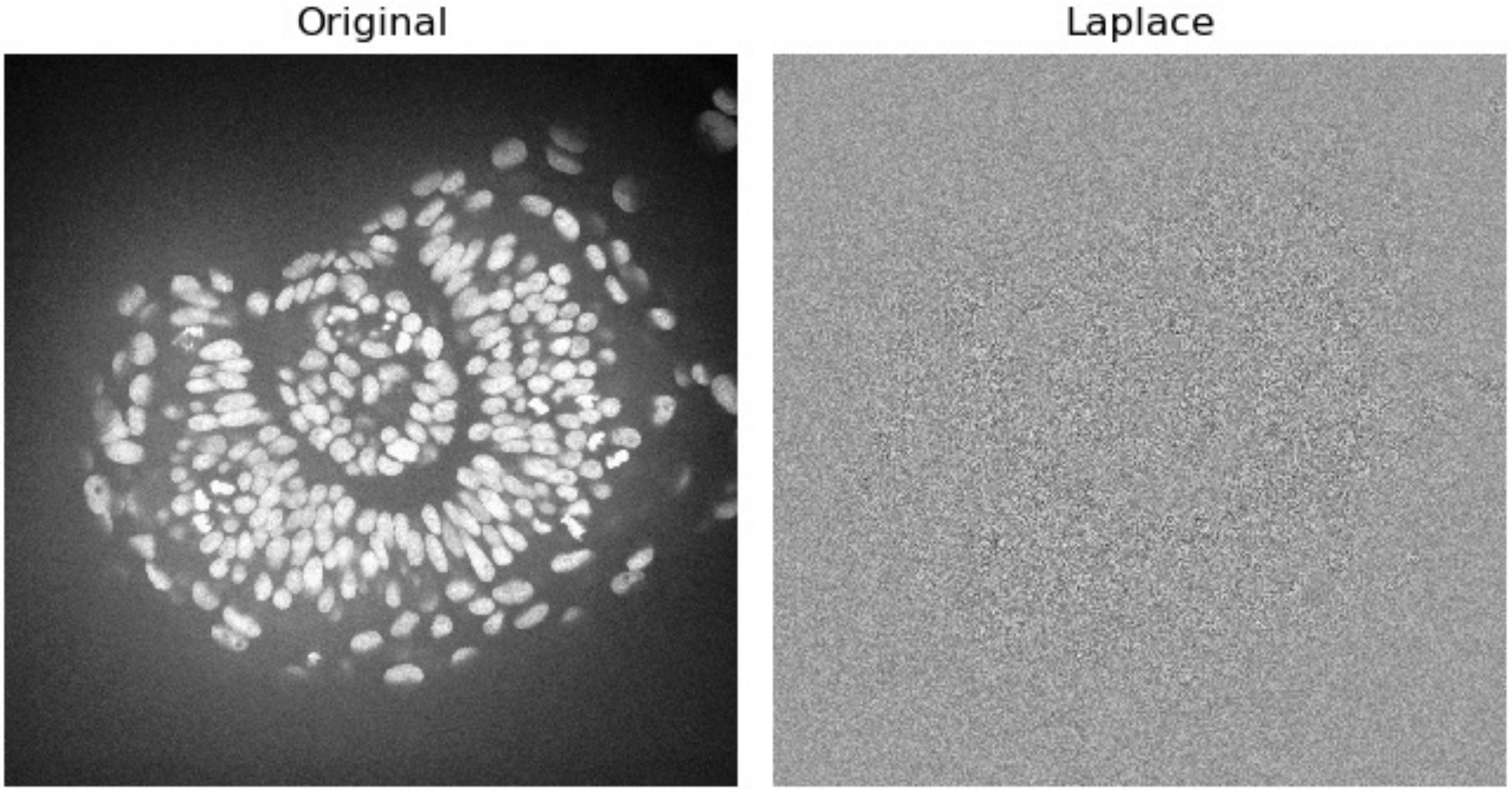




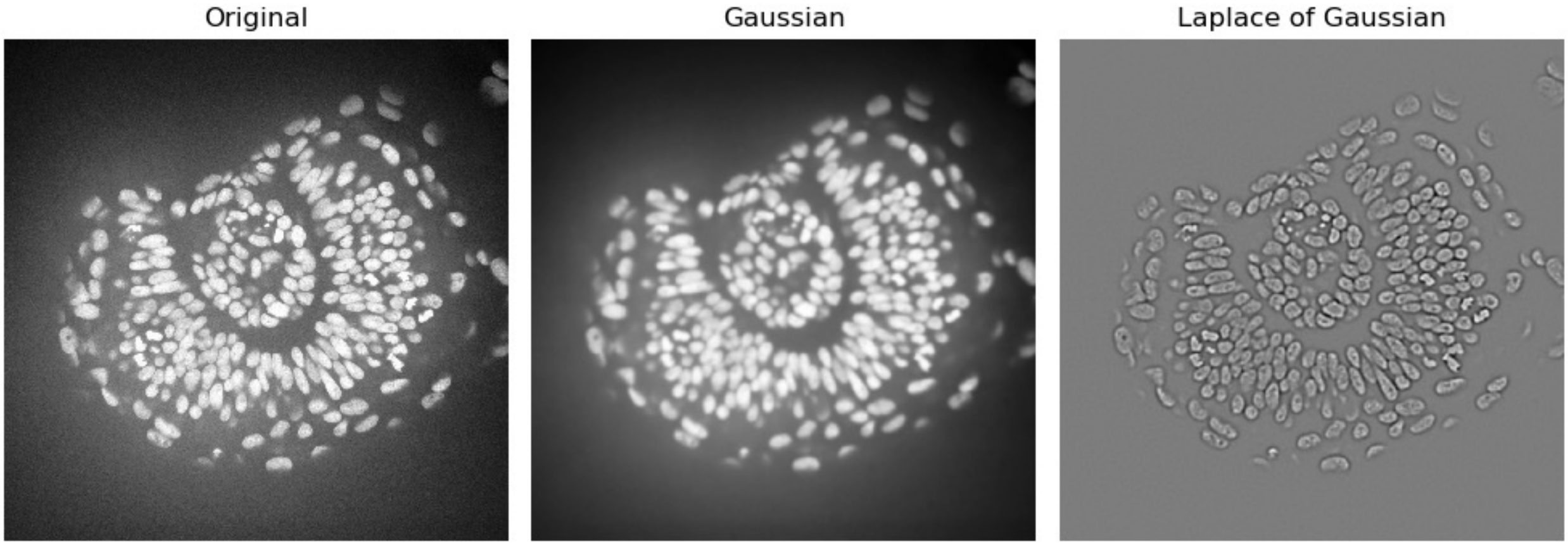




The Laplace filter does not work well on noisy data



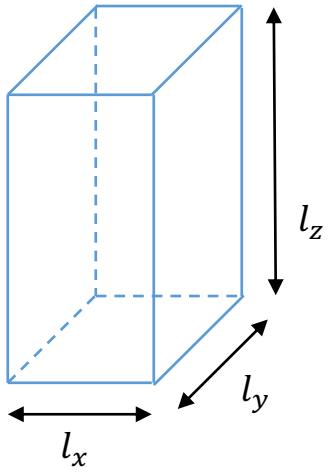
Reducing noise improves the output of the Laplace filter



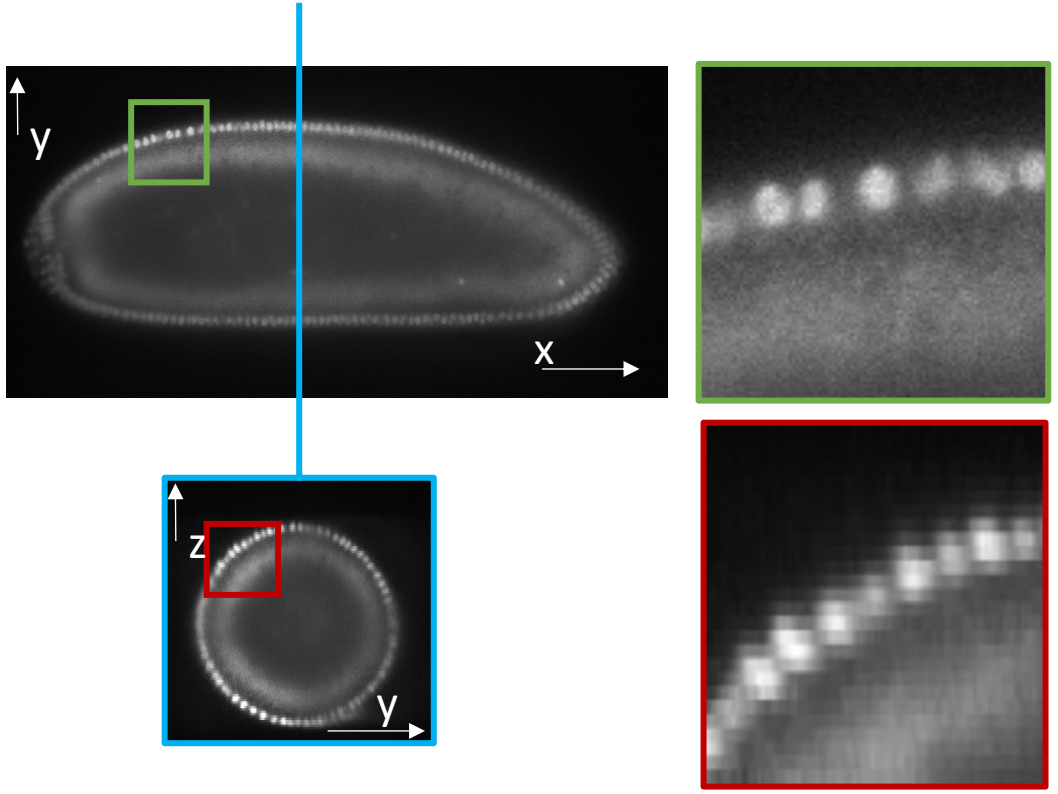
Volumetric Data

- Voxel: “Volume element”, usually anisotropic

- An-iso-tropy, from Greek:
  - ἀν- (not)
  - ἴσος *isos (equal)*
  - τρόπος *tropos (rotation, direction)*
- *Not the same in all directions*
- Usually in 3D image processing:



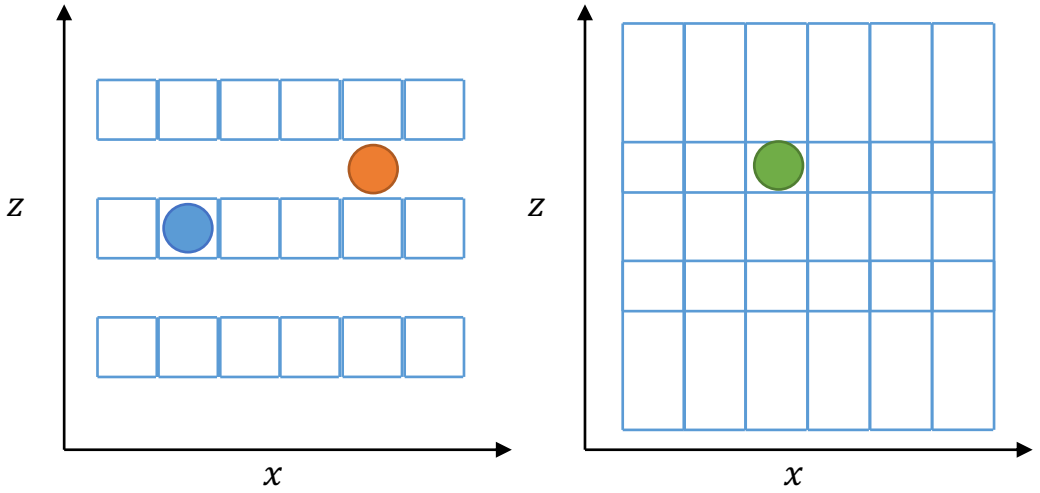
$$l_x = l_y \neq l_z$$



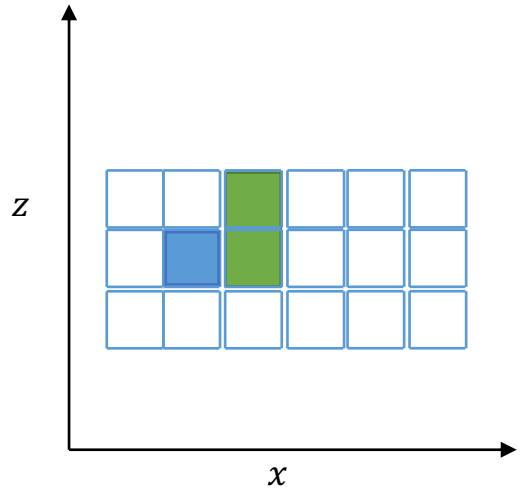
- Image analysts *love* to have isotropic voxels, but microscopes usually have a lower resolution in z than in x and y.

Slice distance versus slice thickness

What you may have measured using imaging:



What you see when processing 3D images:



- Slice distance and slice thickness may be different, but
- many image processing tools ignore that!

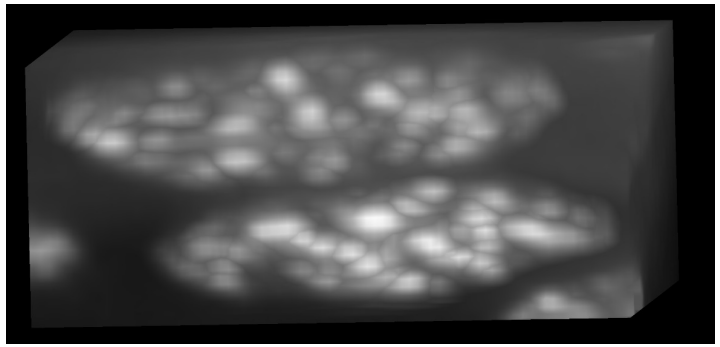
Isotropic Images

The visualization can be fixed in napari by providing a ‘scale’ argument.

```
viewer.add_image(image, name= 'image')
```

```
viewer.layers['image'].scale = [factor_z, factor_y, factor_x]
```

?



```
image.shape
```

(32, 61, 74)

```
reference_size = voxel_size_z

factor_z = voxel_size_z / reference_size
factor_y = voxel_size_y / reference_size
factor_x = voxel_size_x / reference_size
```

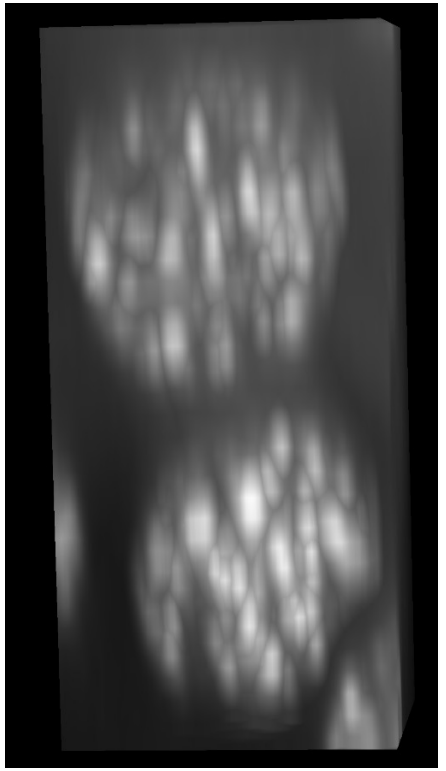
voxel_size_z = 1 µm
voxel_size_y = 0.2 µm
voxel_size_x = 0.2 µm

[5, 1, 1]

[1, 5, 5]

[0.2, 1, 1]

[1, 0.2, 0.2]



```
image.shape
```

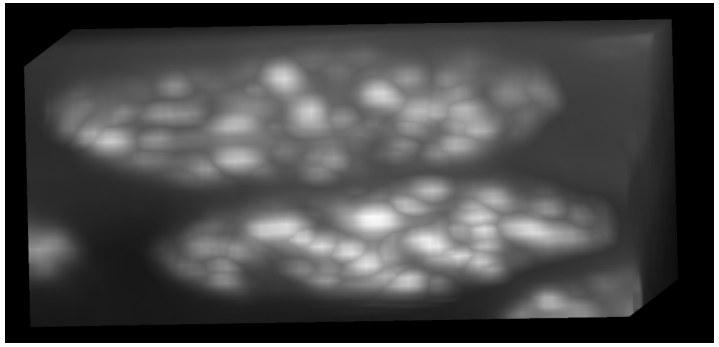
(32, 61, 74)

But that does not change voxel size, just adjusts visualization.

Filters in 3D Images

The functions are typically the same, but many of them may not account by default different voxel sizes.

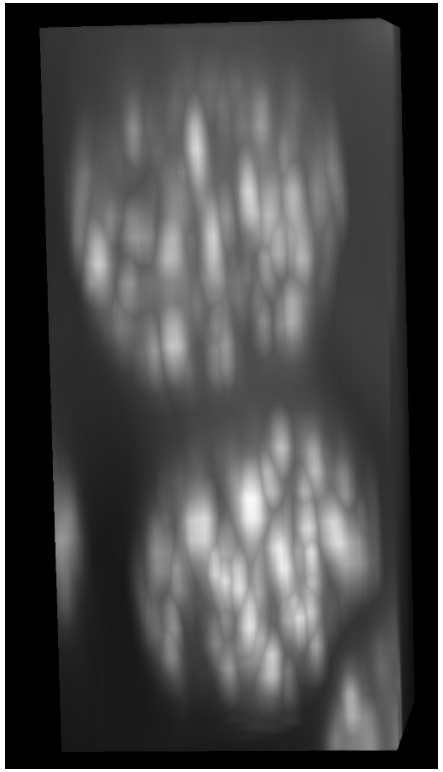
If the filter does not take voxel size into account, we would need to rescale the image before applying a filter.



```
image.shape
```

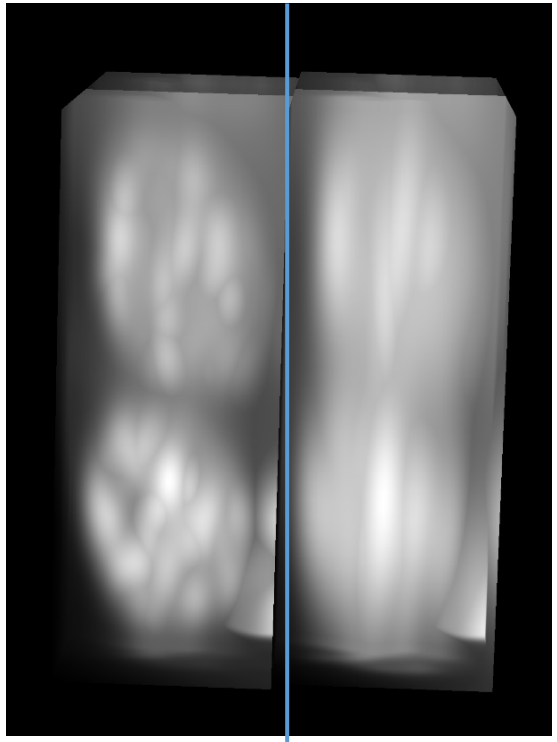
(32, 61, 74)

rescale



```
image.shape
```

(158, 61, 74)



Gaussian blur on rescaled image

Gaussian blur on image