PoL
Physics of Life
TU Dresden

CENTER FOR
SYSTEMS BIOLOGY
DRESDEN

# Image thresholding

Johannes Müller

With material from
Robert Haase, BiaPoL, PoL TU Dresden
Marcelo Zoccoler, BiaPol, PoL, TU Dresden
Benoit Lombardot, Scientific Computing Facility, MPI CBG

@jm_mightypirate

# Segmentation

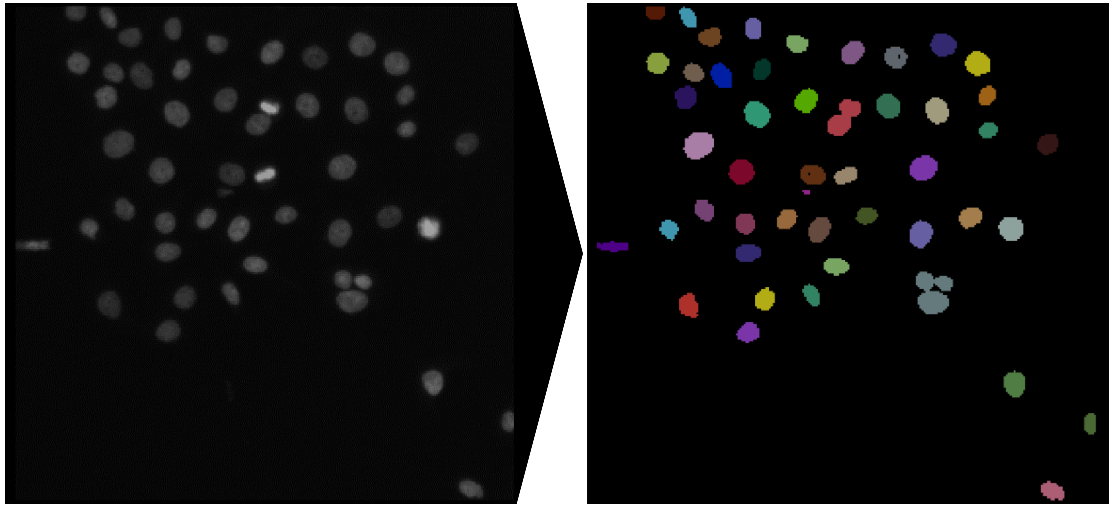
## Aim:

Separate background from foreground

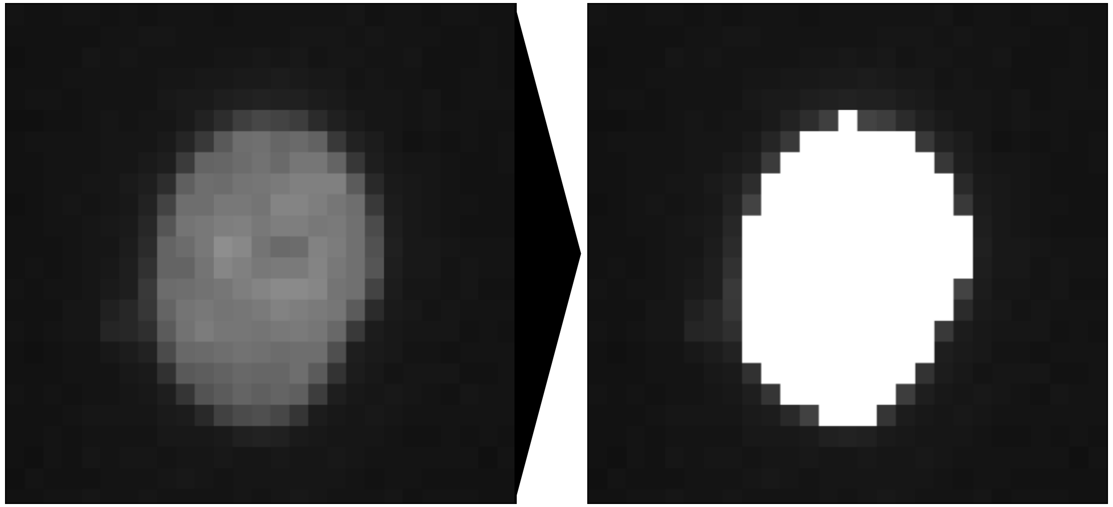
## Vocabulary:

- **Segmentation:**
  → Assigning a meaningful *label* to each pixel
  → Segmentation is a *classification* problem
- **Semantic segmentation:**
  Differentiate pixels into multiple *classes* (e.g., membrane, nucleus, cytosol, etc.)
- **Instance segmentation:**
  Differentiate multiple occurrences of the same class into separate instances of this class (e.g., separate *label* for each cell in image)

https://scikit-image.org/docs/stable/api/skimage.data.html



Instance segmentation

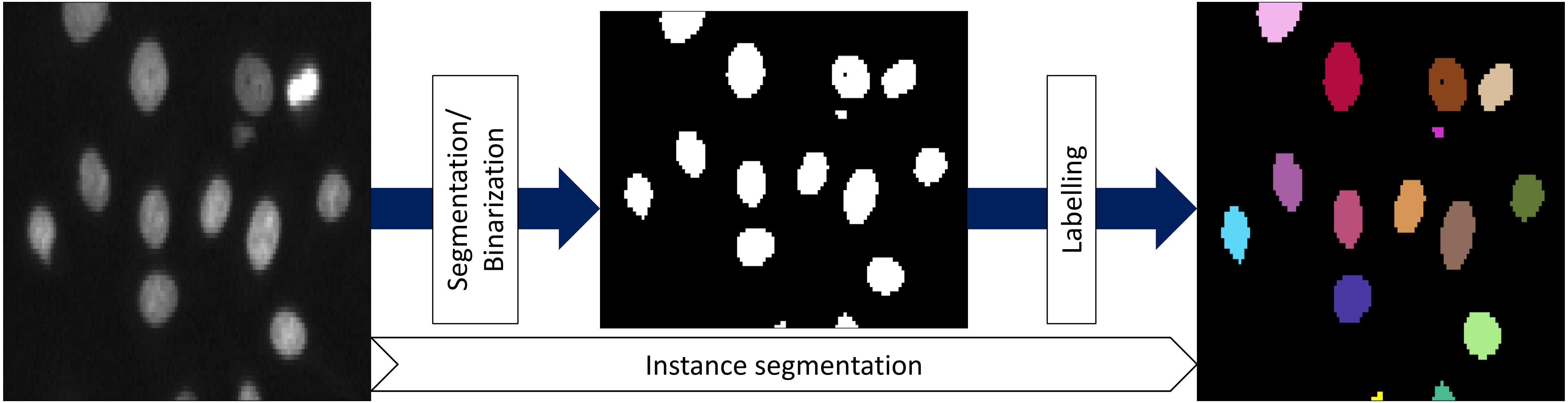


Semantic segmentation

# Segmentation and labelling

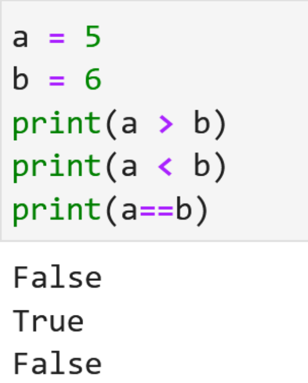
Analyzing properties (*features*) of individual objects in images requires instance segmentation

- Methods
  - Thresholding + connected components labeling
  - Spot detection + seeded watershed
  - Edge detection based
  - Machine learning



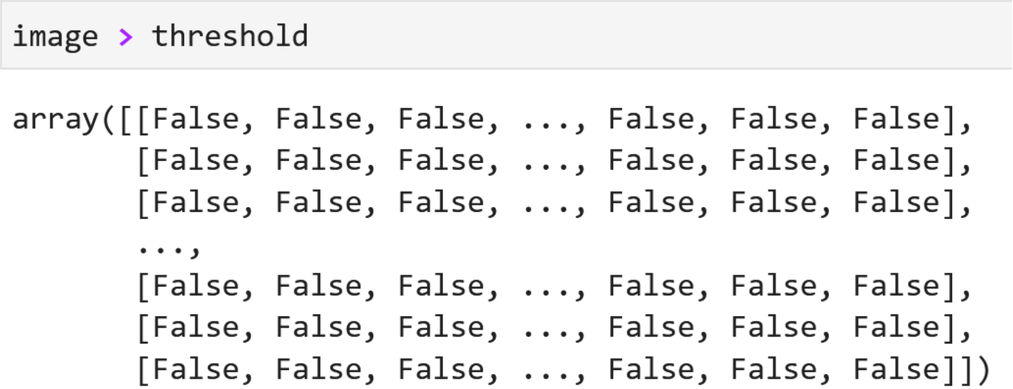
# Thresholding in Python

- Applying a threshold to an image requires to compare every pixel to the threshold value
- We can compare values in Python with:



```python
a = 5
b = 6
print(a > b)
print(a < b)
print(a==b)
```

```
False
True
False
```

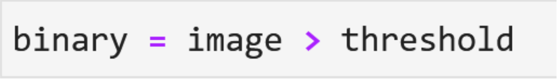


```python
image > threshold
```

```
array([[False, False, False, ..., False, False, False],
       [False, False, False, ..., False, False, False],
       [False, False, False, ..., False, False, False],
       ...,
       [False, False, False, ..., False, False, False],
       [False, False, False, ..., False, False, False],
       [False, False, False, ..., False, False, False]])
```

In this case, “image” is a *numpy array* → some operations are automatically applied to every pixel!

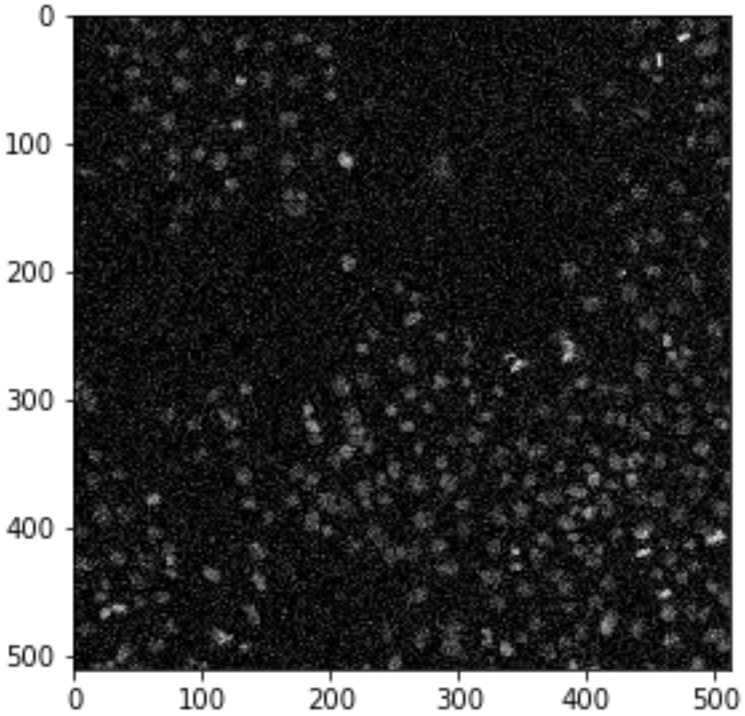
- We can then simply store the output of this element-wise comparison in a new variable:



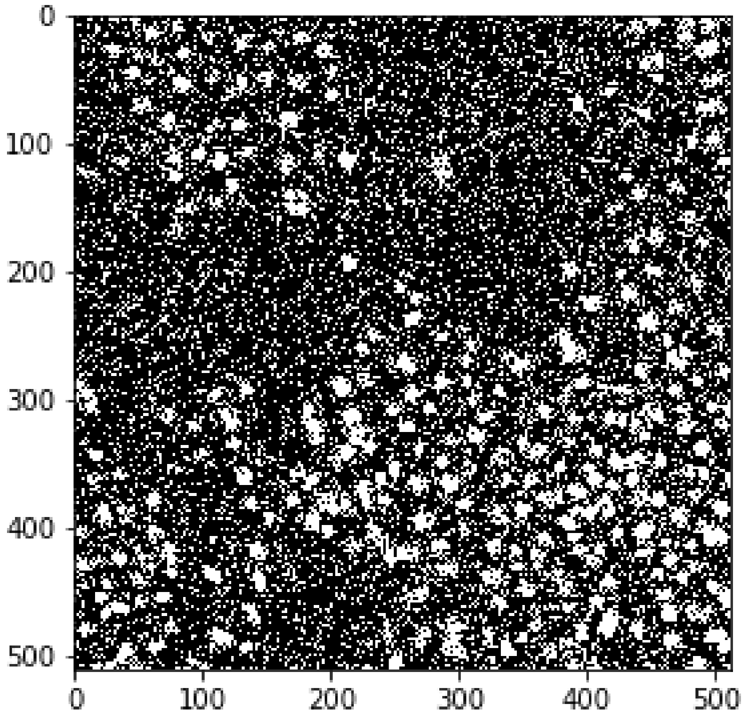
```python
binary = image > threshold
```

# Reminder: pre-processing!

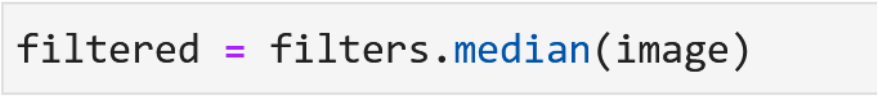
- Before we can create masks, we need to pre-process images:
  - Noise removal
  - Background subtraction



Noisy image



Thresholded image

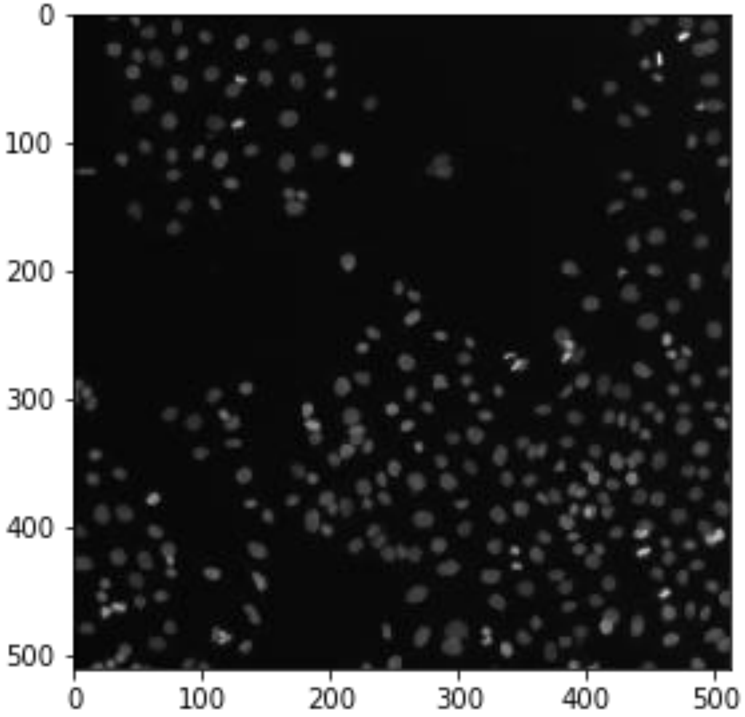


```
filtered = filters.median(image)
```

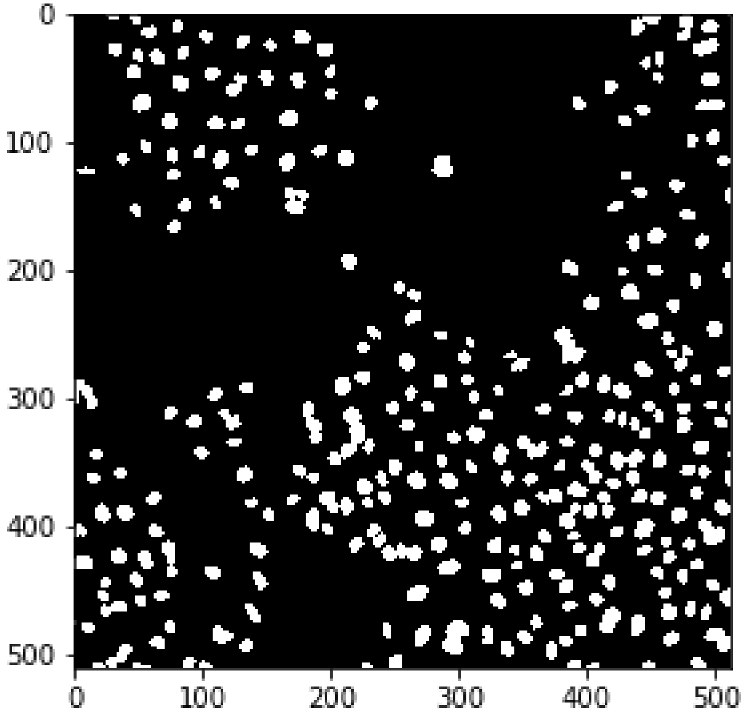
Image filtering *filters* relevant information for subsequent operations from the image!

# Reminder: pre-processing!

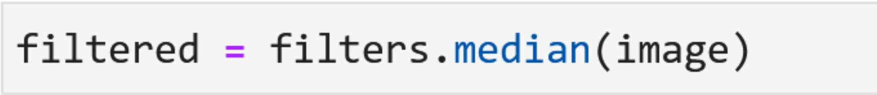
- Before we can create masks, we need to pre-process images.
  - Noise removal
  - Background subtraction



Filtered image



Thresholded image

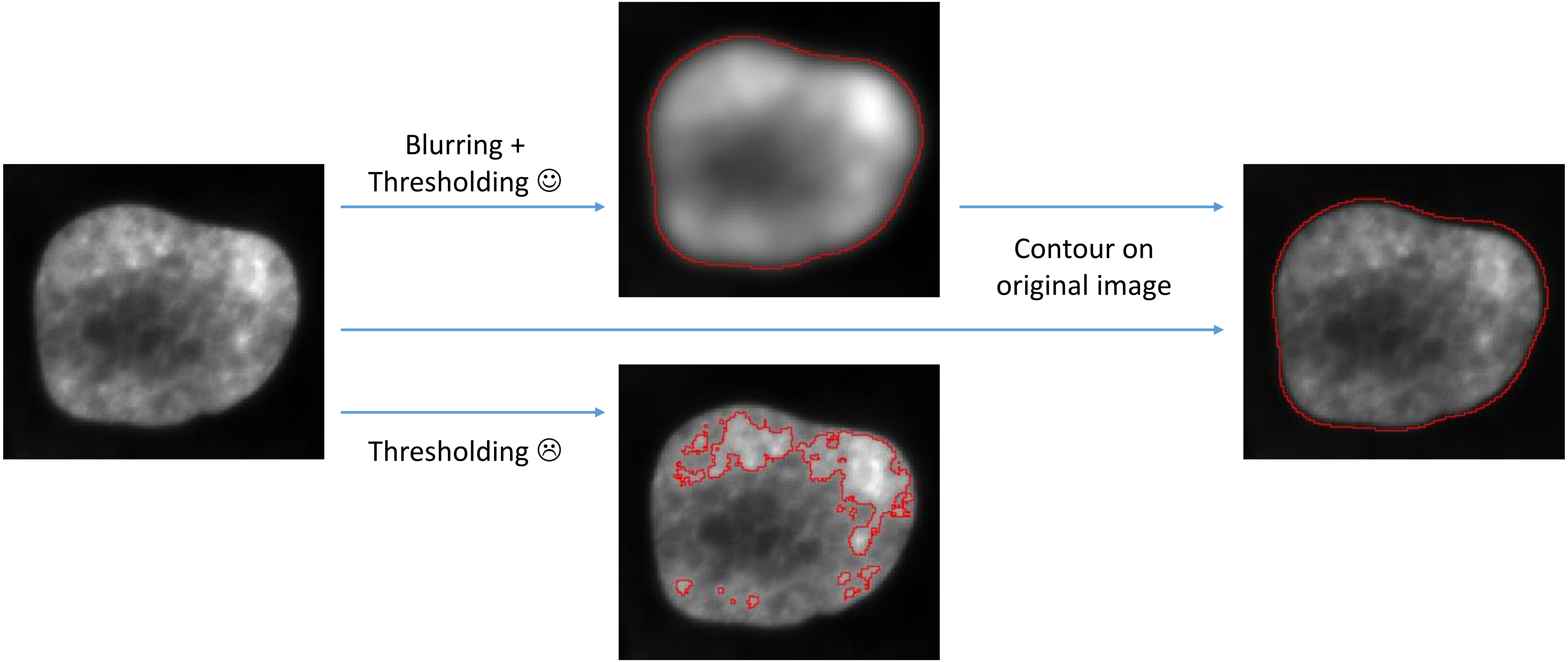


```
filtered = filters.median(image)
```

Image filtering *filters* relevant information for subsequent operations from the image!

# *Low-pass filtering* to improve thresholding results

- In case thresholding algorithms outline the wrong structure, <u>blurring in advance</u> may help.
- However: **Do not** continue processing the blurred image, continue with the original!



Blurring + Thresholding ☺

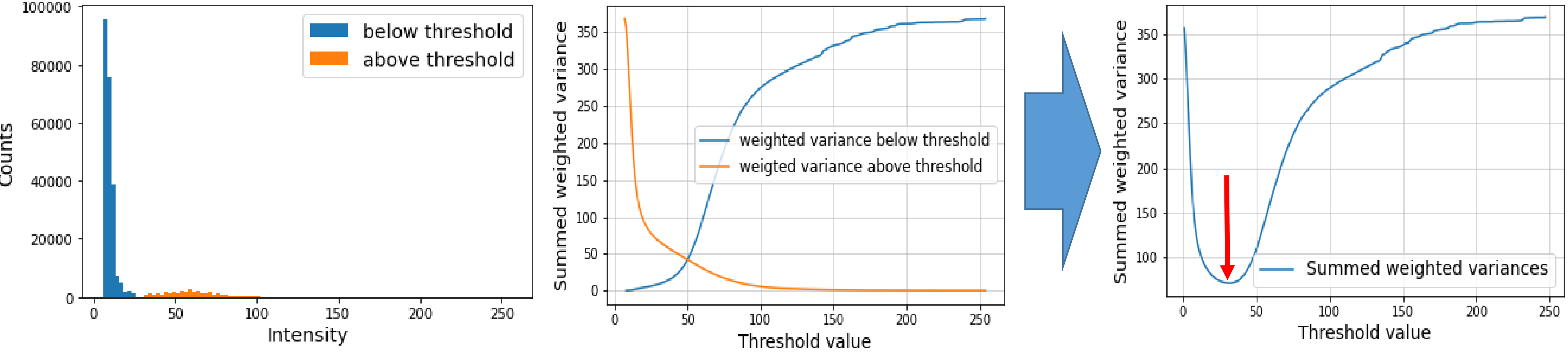
Contour on original image

Thresholding ☹

# Threshold algorithms

- **Otsu-thresholding** (Otsu et Al. 1979): Find threshold so that the summed, weighted variance $Var_{w,sum}$ becomes minimal:

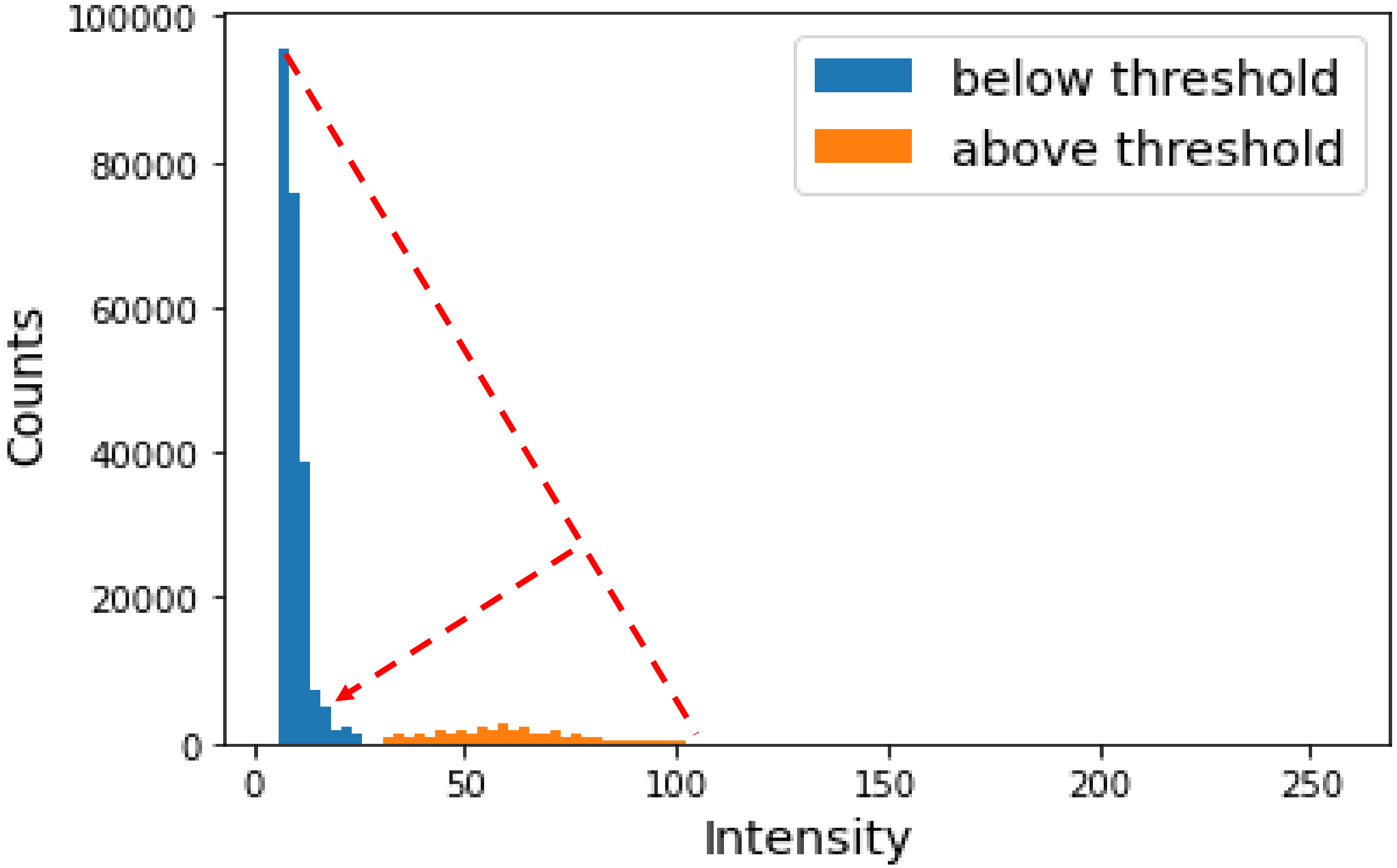
$$Var(I) = \frac{1}{n_I}\sum\left(I - mean(I)\right)^2 \quad \rightarrow \quad Var_{w,sum} = \frac{n_A}{n_I}\cdot Var(A) + \frac{n_B}{n_I}\cdot Var(B)$$



- **Statistical thresholding**: Pixels above statistical parameter of I belong to  foreground. (Possibilities: Mean, Median, Quartiles, etc.)

# Threshold algorithms

- **Triangle thresholding**: Draw a line between histogram point with max. counts and max. intensity and find point in histogram with maximal distance to this line (----)



# Thresholding with scikit-image

```
threshold = filters.threshold_otsu(image)
```

- **Otsu-thresholding** (Otsu et Al. 1979): Find threshold so that the summed, weighted variance $Var_{w,sum}$ becomes minimal.

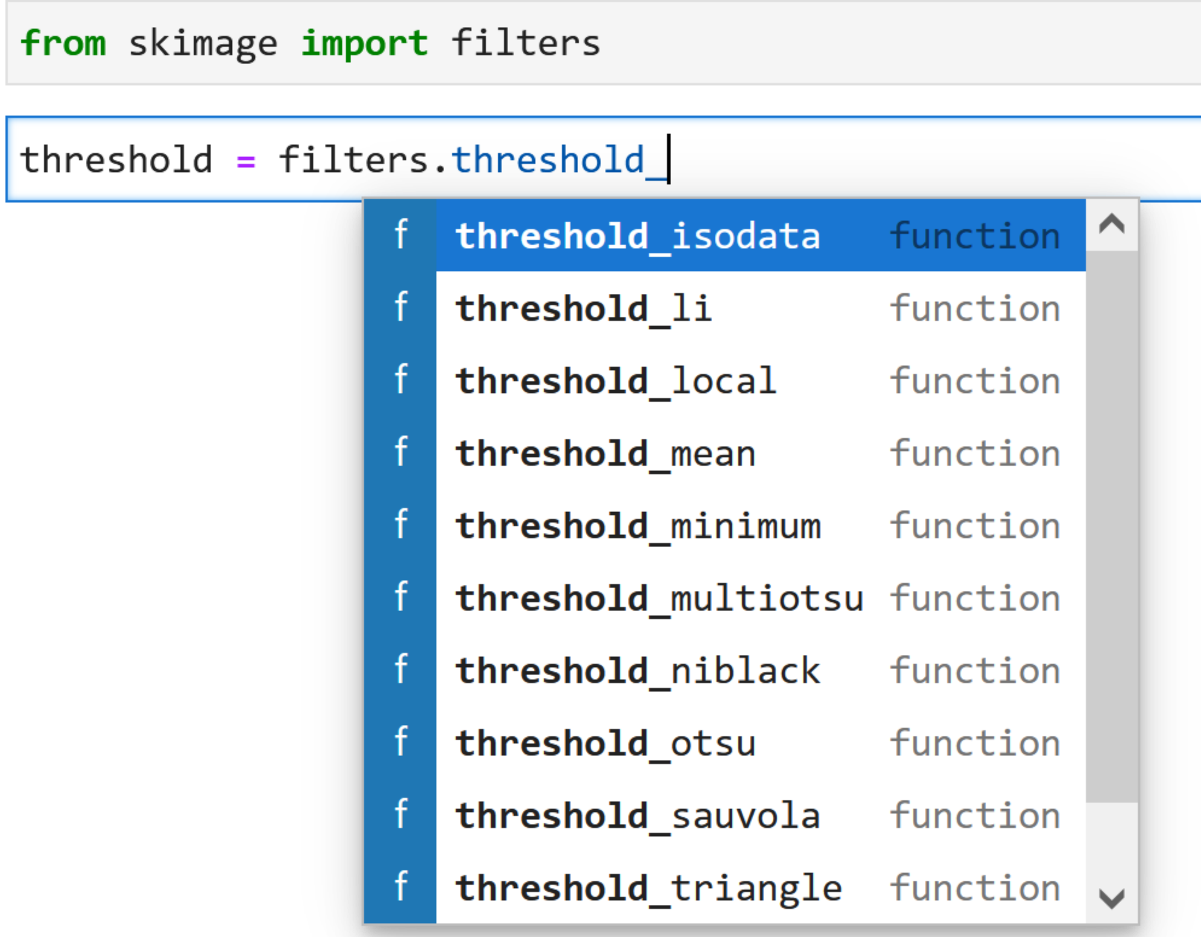
```
threshold = filters.threshold_mean(image)
```

- **Statistical thresholding**: Pixels above statistical

```
threshold = filters.threshold_triangle(image)
```

- **Triangle thresholding**: Draw a line between histogram point with max. counts and max. intensity and find point in histogram with maximal distance to this line.

**Explore more threshold options in scikit-image with:**



```
from skimage import filters
```

```
threshold = filters.threshold_
```

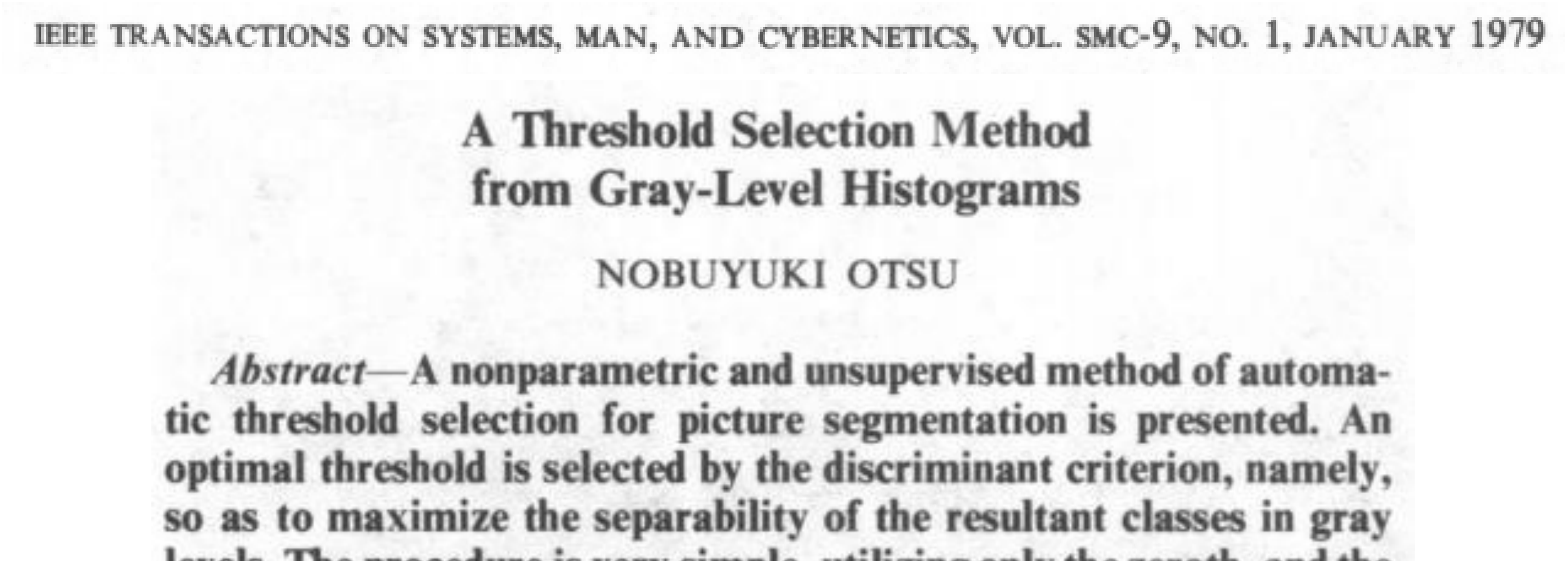
```
f  threshold_isodata    function
f  threshold_li         function
f  threshold_local      function
f  threshold_mean       function
f  threshold_minimum    function
f  threshold_multiotsu  function
f  threshold_niblack    function
f  threshold_otsu       function
f  threshold_sauvola    function
f  threshold_triangle   function
```

# Thresholding: Citing

- Cite the thresholding method of your choice properly

!!!
Important for citing the otsu thresholding used in processing

*We segmented the cell nuclei in the images using the Otsu thresholding method (Otsu et Al. 1979) implemented in scikit-image (van der Walt et Al. 2014).*



IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, VOL. SMC-9, NO. 1, JANUARY 1979

**A Threshold Selection Method from Gray-Level Histograms**

NOBUYUKI OTSU

*Abstract*—A nonparametric and unsupervised method of automatic threshold selection for picture segmentation is presented. An optimal threshold is selected by the discriminant criterion, namely, so as to maximize the separability of the resultant classes in gray

# Thresholding: Pitfalls

```
binary = image > a_good_threshold_value_of_my_choice
```

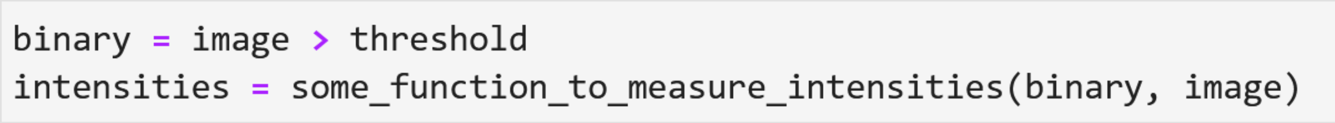
Inter-observer variability

**Never use manual thresholding!**

- Different observers come to different results when selecting a “good” threshold value

➢ You may come to different results when selecting a threshold value repeatedly

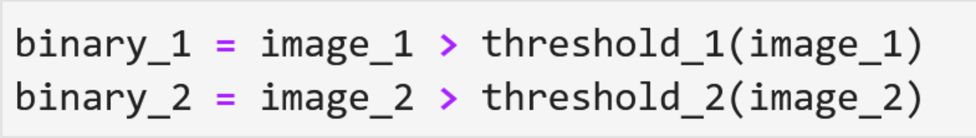
Intra-observer variability



```
binary = image > threshold
intensities = some_function_to_measure_intensities(binary, image)
```

**Avoid thresholding an image and afterwards measure intensities in the same image**

- You would measure the threshold you entered



```
binary_1 = image_1 > threshold_1(image_1)
binary_2 = image_2 > threshold_2(image_2)
```

**Chose one threshold algorithm:**

…and stick to it for the whole study. Using a new method for every image impairs reproducibility!

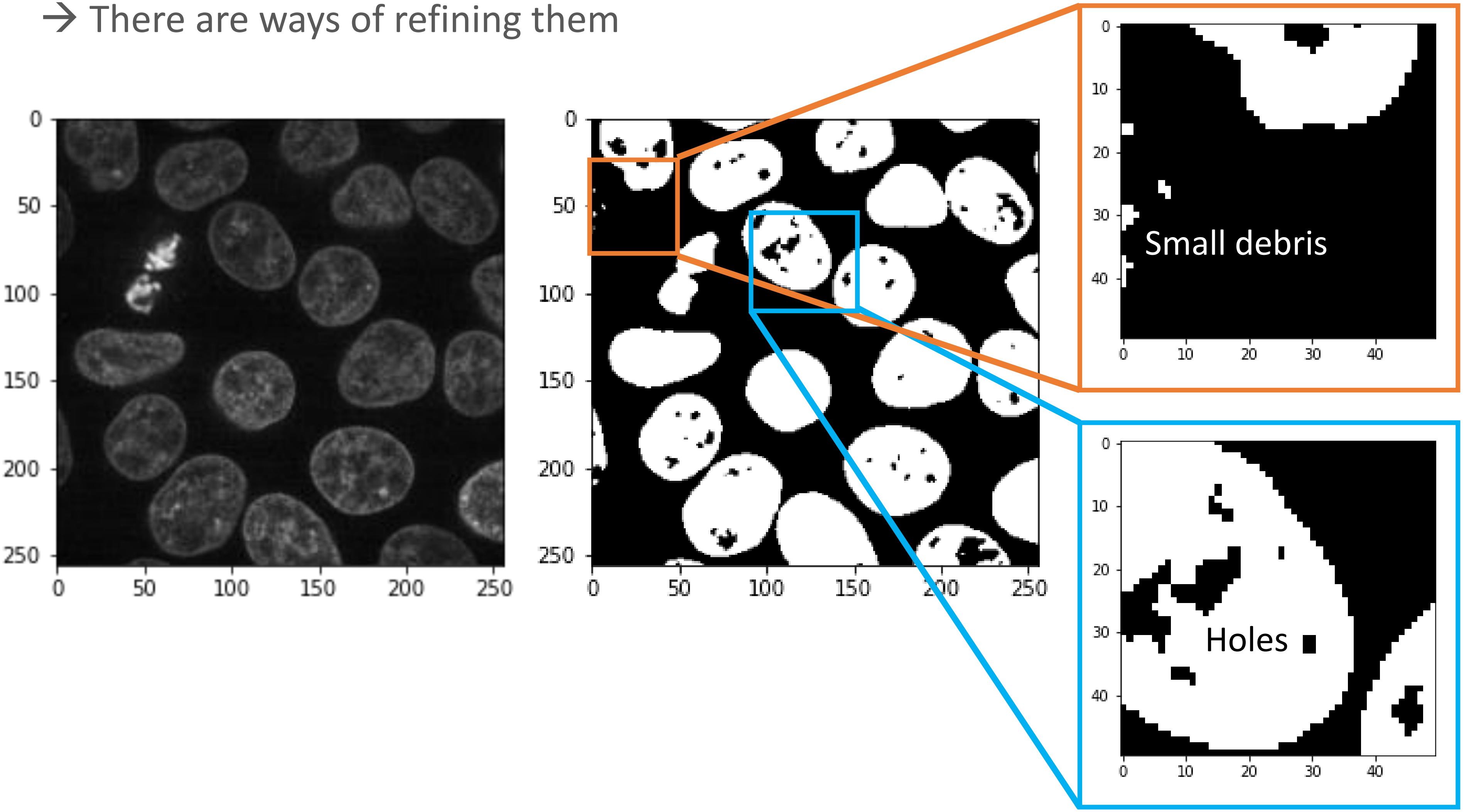
**Do not over-engineer**

There will be always images where thresholding fails – better report the errors!

# Refining masks

Binary mask images may not be perfect immediately after thresholding.

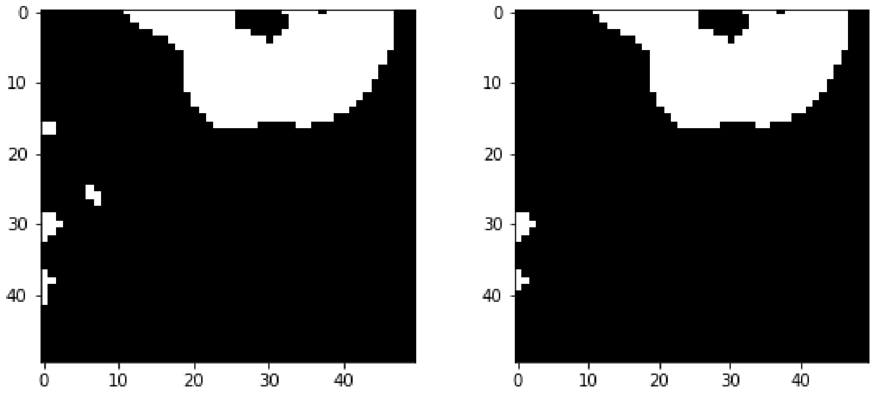
→ There are ways of refining them



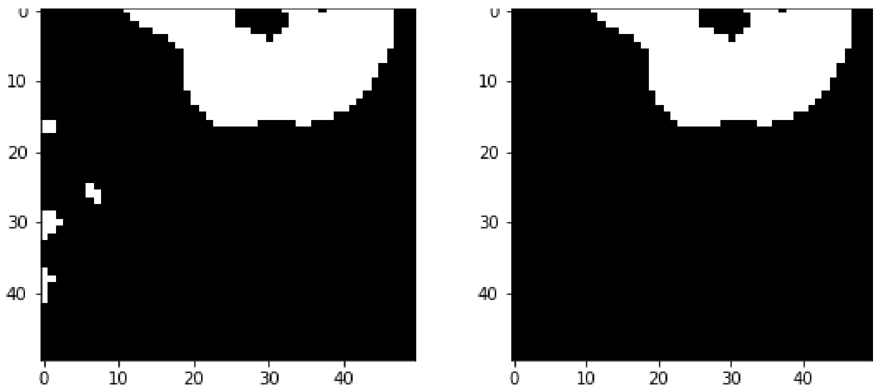
# Refining masks: Opening & Closing

```python
from skimage import morphology
```

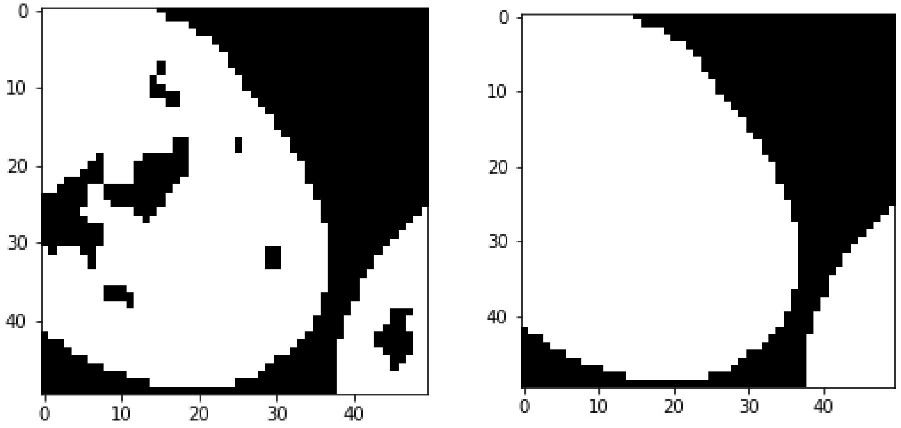
```python
opened = morphology.binary_opening(binary)
```



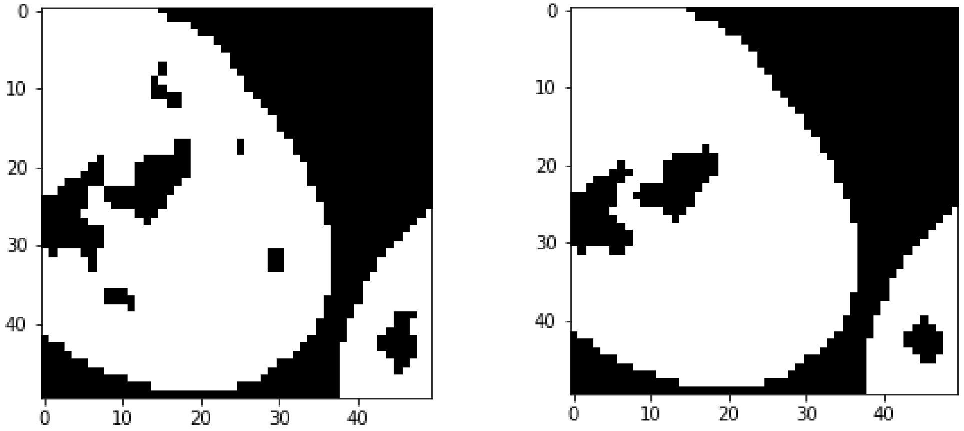
```python
opened = morphology.area_opening(binary, area_threshold=20)
```



```python
closed = morphology.area_closing(binary, area_threshold=100)
```

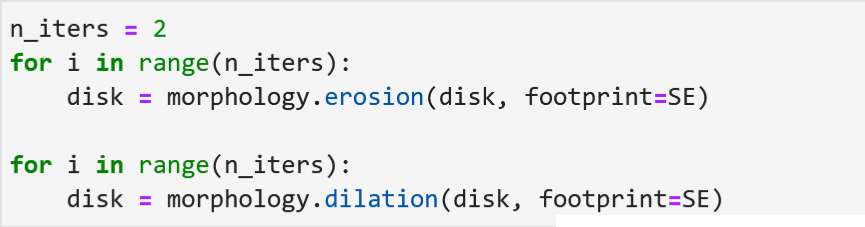


```python
closed = morphology.binary_closing(binary)
```



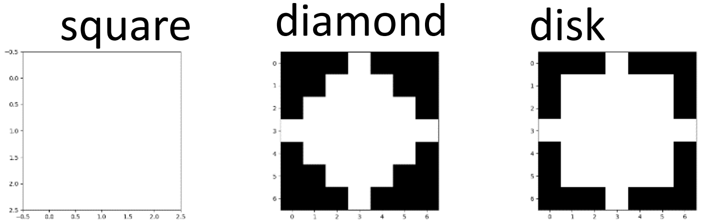
# Pitfalls

- Eroding/dilating binary images with structural element tends to reproduce the structural element

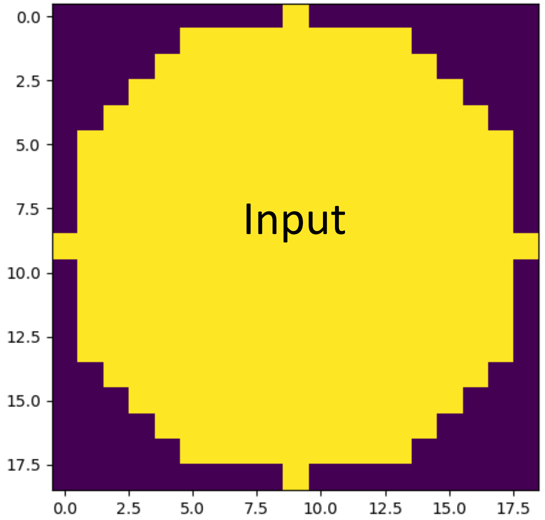


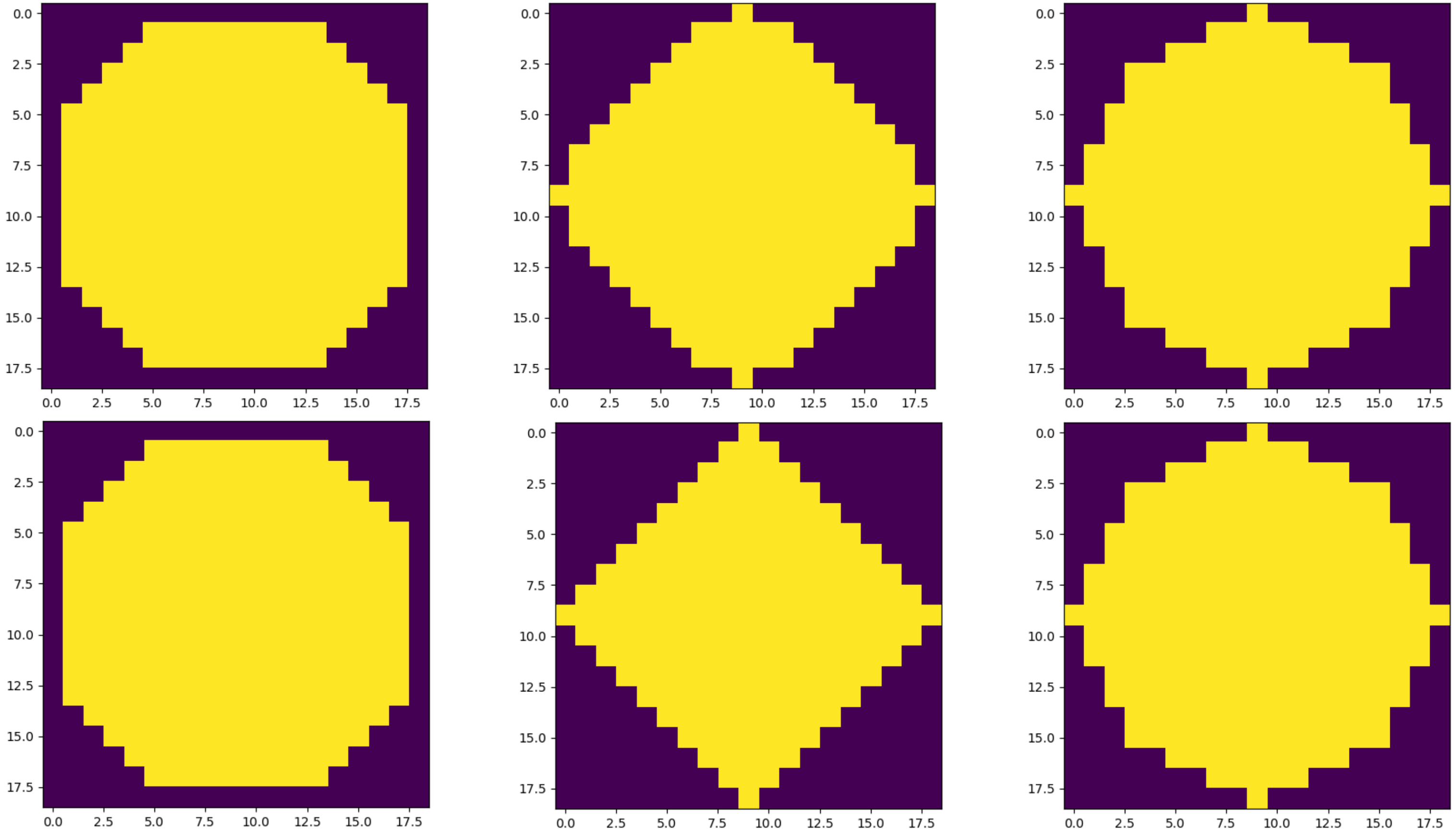
```
n_iters = 2
for i in range(n_iters):
    disk = morphology.erosion(disk, footprint=SE)

for i in range(n_iters):
    disk = morphology.dilation(disk, footprint=SE)
```



NOTE:
The number of erosion iteration gives different results based on the input image data.





n_iters=2

n_iters=3

square

diamond

disk

@jm_mightypirate

October 2022