Image thresholding

Till Korten

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

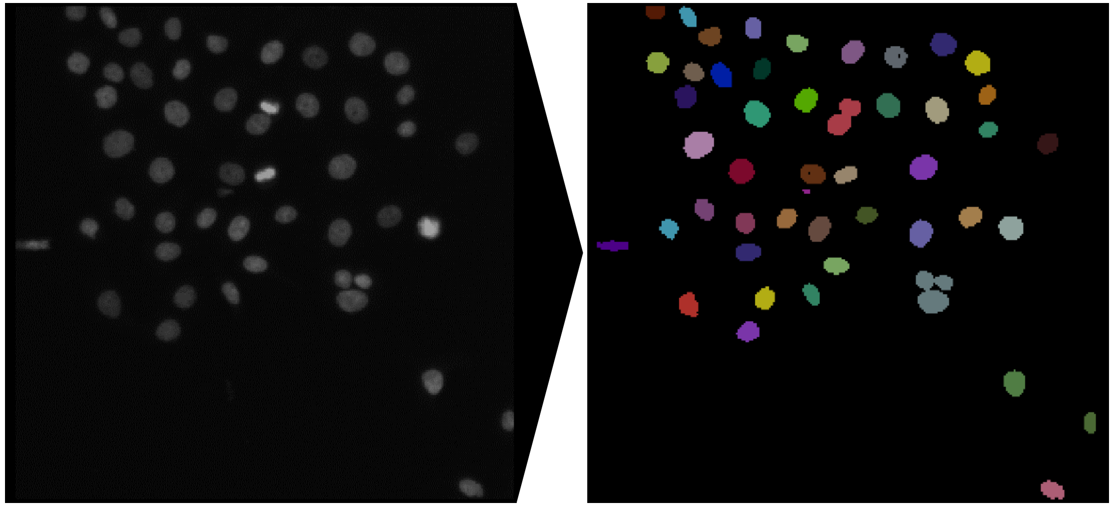
Segmentation

Aim:

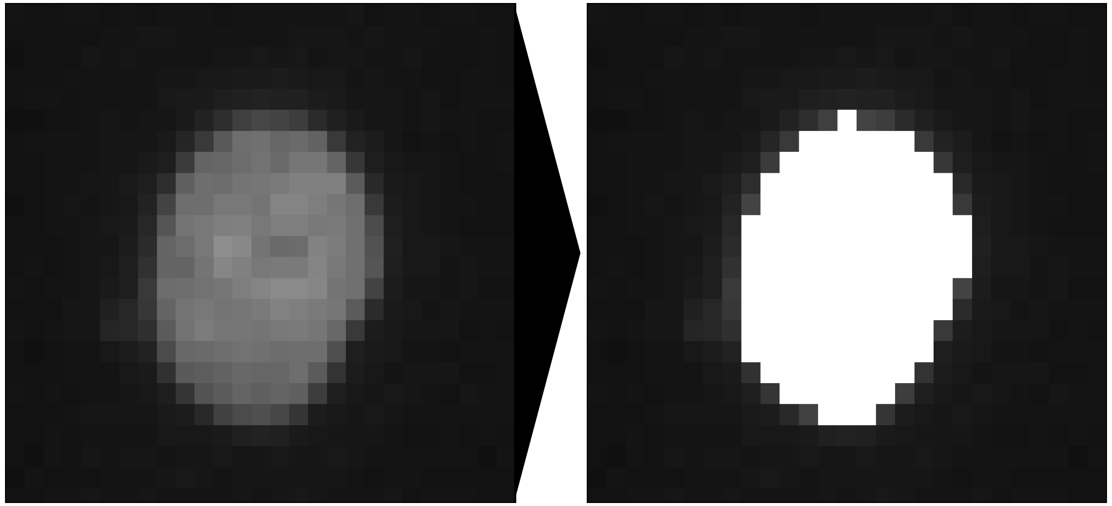
Separate background from foreground

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)



Instance segmentation



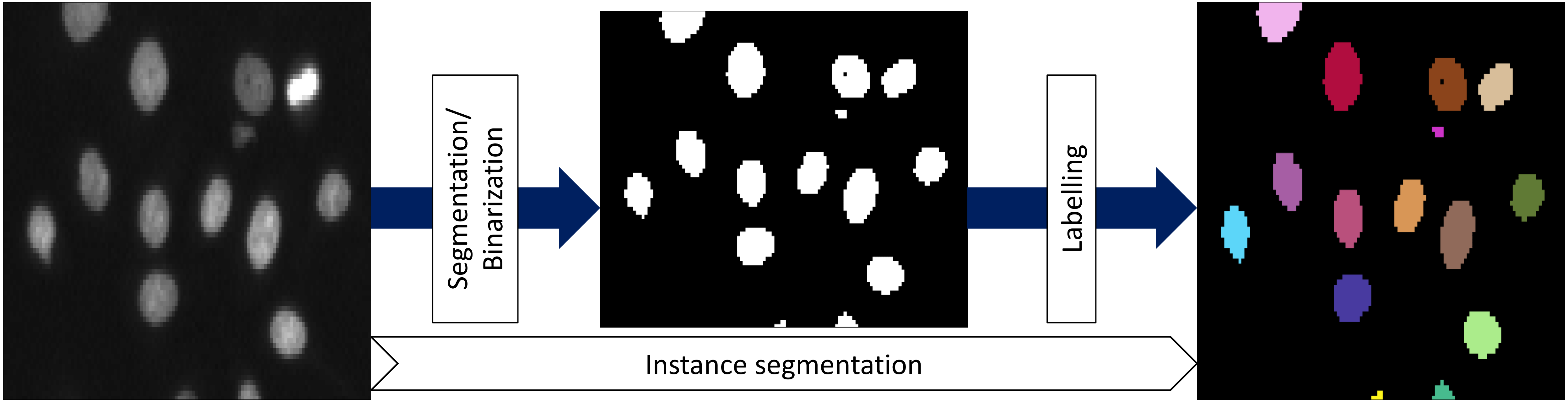
Semantic segmentation

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

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



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

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]])
```

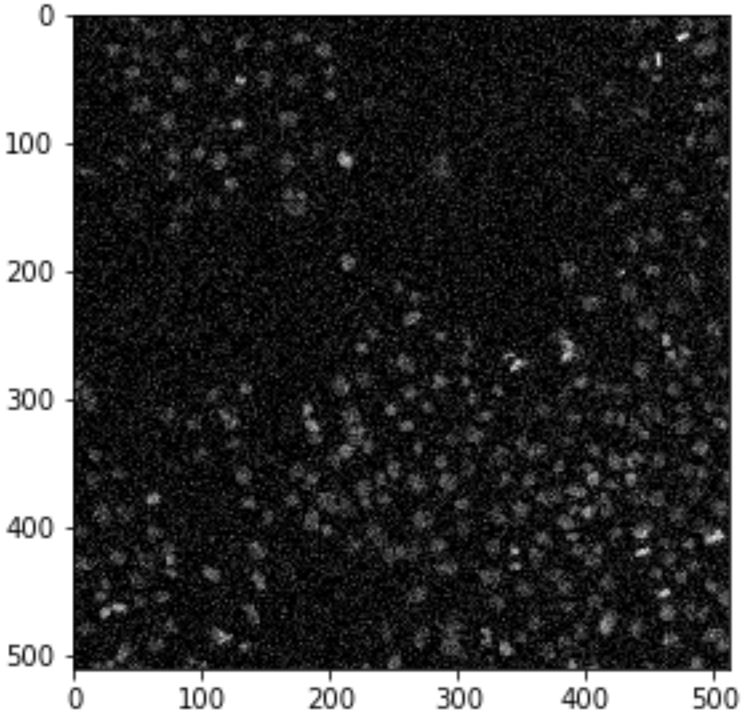
In this case, “image” is a *numpy array* → some operations are automatically applied to every pixel!

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

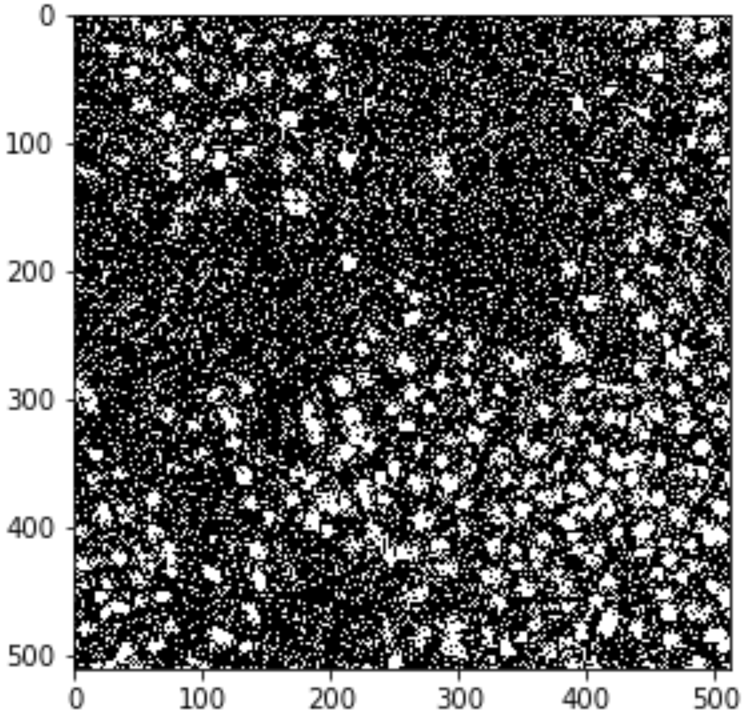
```python
binary = image > threshold
```

Reminder: pre-processing!

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



Noisy image



Thresholded image

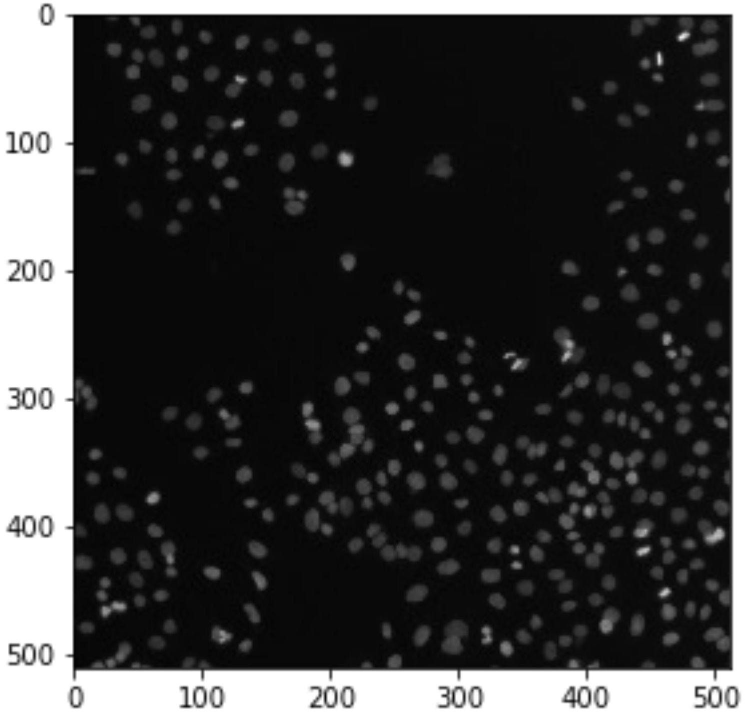
```
filtered = filters.median(image)
```

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

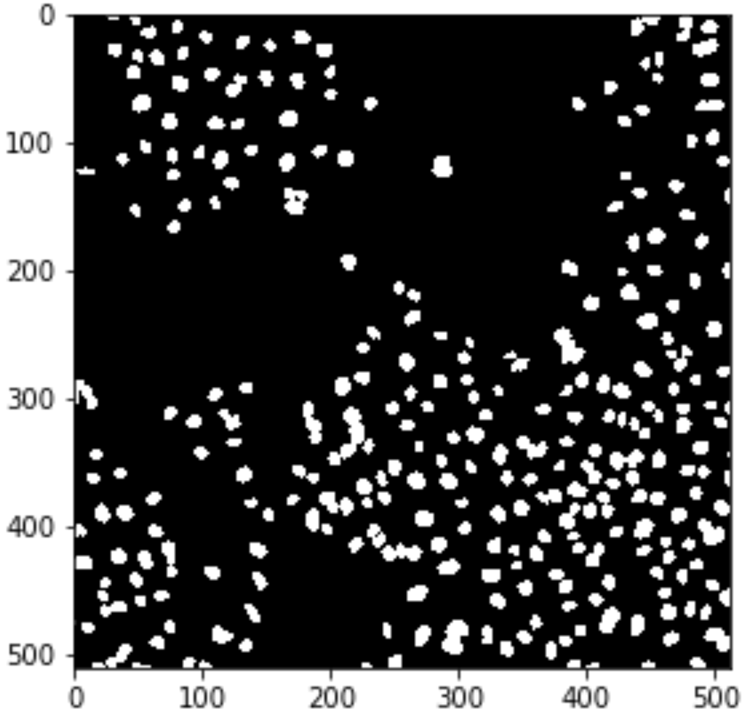
https://scikit-image.org/docs/stable/api/skimage.data.html

Reminder: pre-processing!

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



Filtered image



Thresholded image

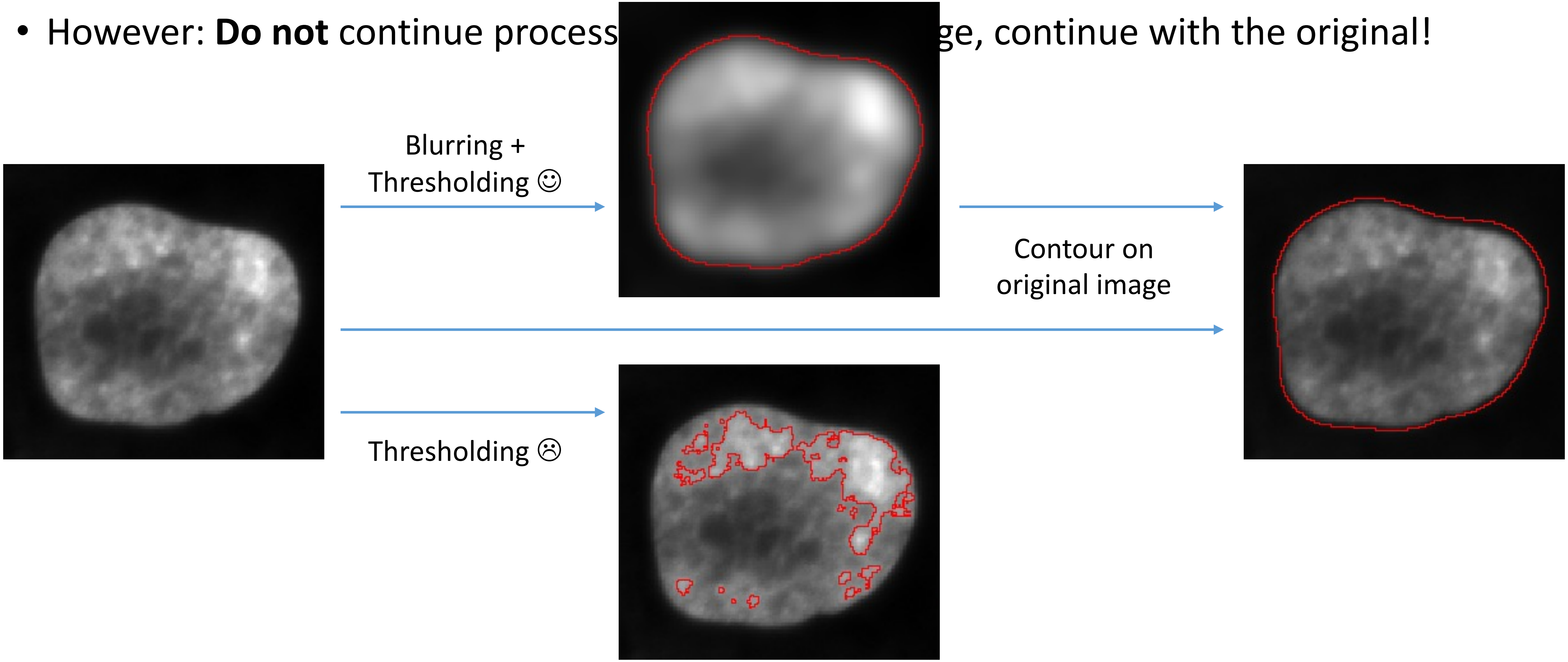
```python
filtered = filters.median(image)
```

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

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

Low-pass filtering to improve thresholding results

- In case thresholding algorithms outline the wrong structure, blurring in advance may help.
- However: **Do not** continue process ge, continue with the original!



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)
```

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

Do not over-engineer

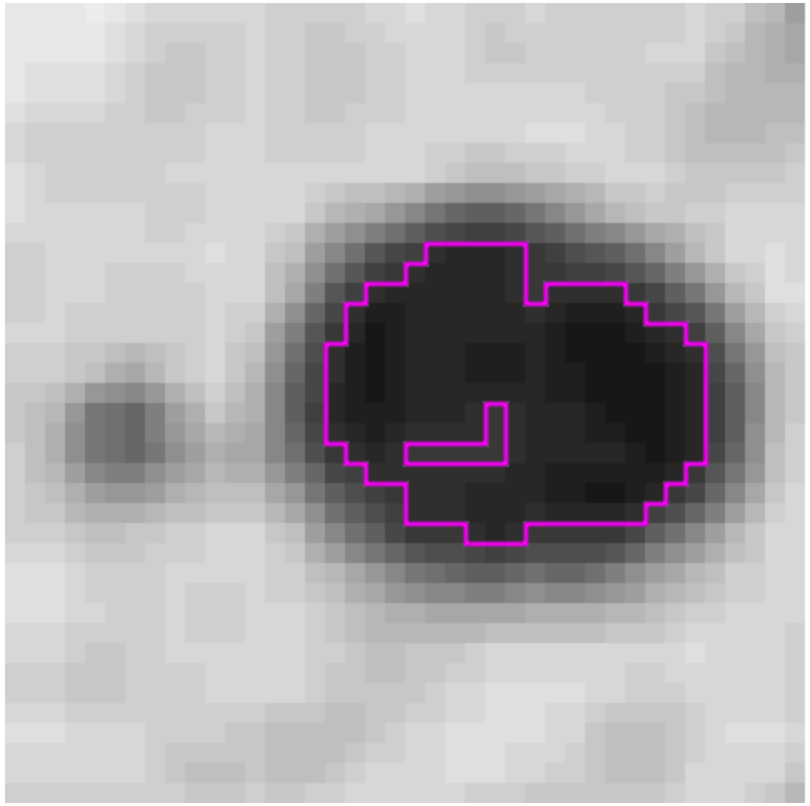
There will be always images where thresholding fails – better report the errors!

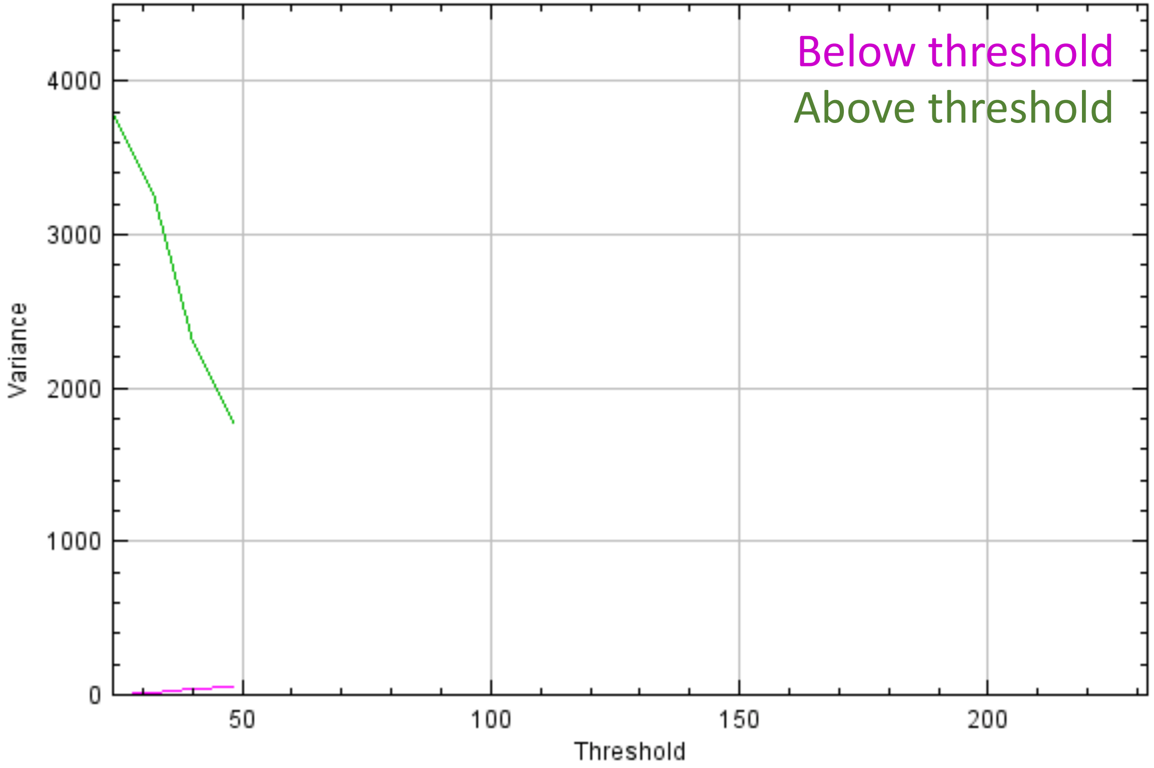
Otsu's method

- Searching for a threshold where the variance in both classes (above/below threshold) becomes minimal.

$$Var(I) = \sum_{i \in I} g_i - \bar{g}_I \qquad \bar{g}_I = \sum_{i \in I} \frac{g_i}{n_I}$$

$Var(I)$ … Variance in image I
g_i … grey value of a pixel i
$\bar{g}_I$ … mean grey value of the whole image I
n_I … number of pixels in Image I



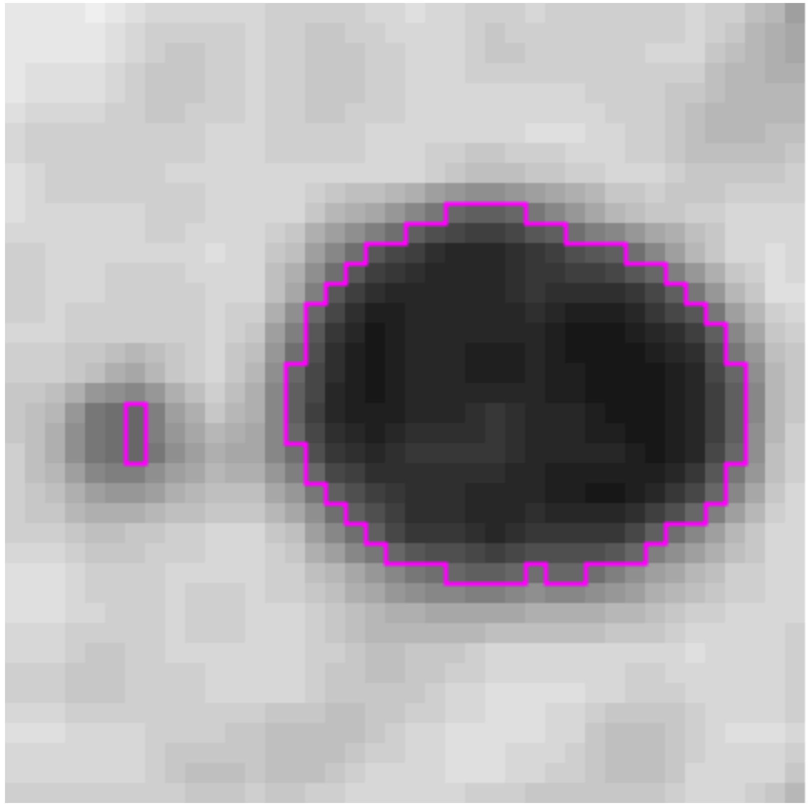


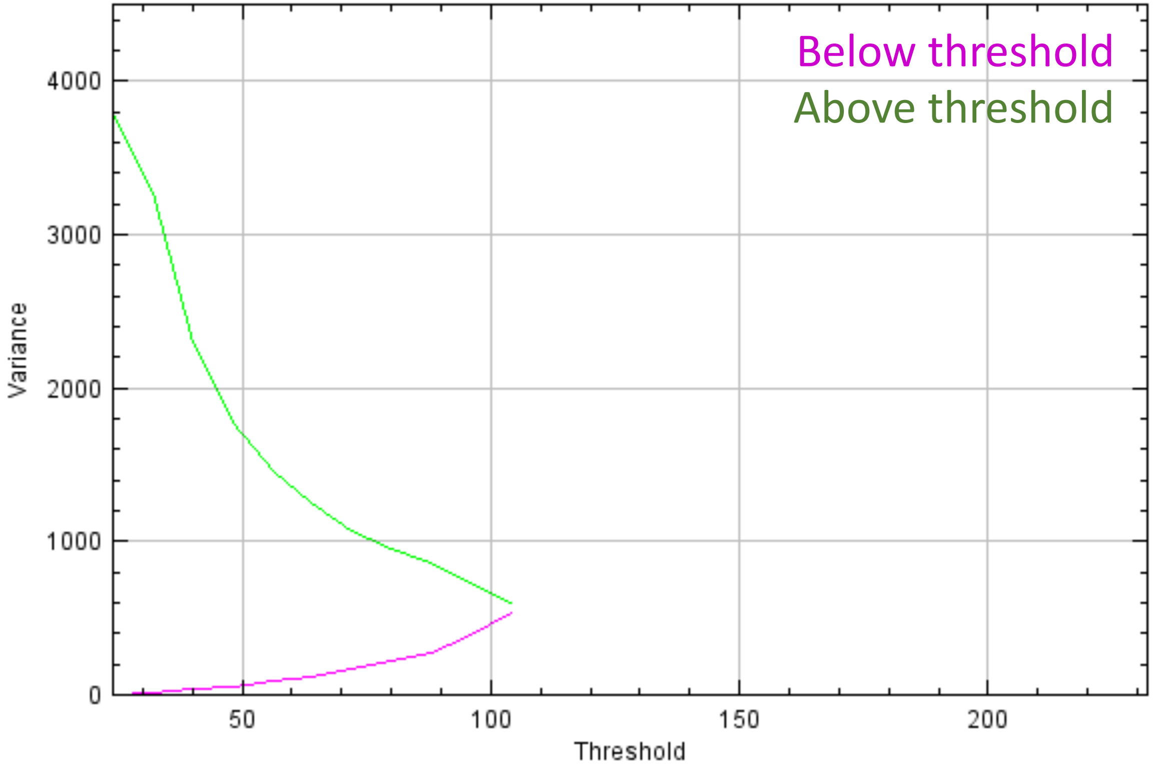
Otsu's method

- Searching for a threshold where the variance in both classes (above/below threshold) becomes minimal.

$$Var(I) = \sum_{i \in I} g_i - \bar{g}_I \qquad \bar{g}_I = \sum_{i \in I} \frac{g_i}{n_I}$$

$Var(I)$ … Variance in image I
g_i … grey value of a pixel i
$\bar{g}_I$ … mean grey value of the whole image I
n_I … number of pixels in Image I



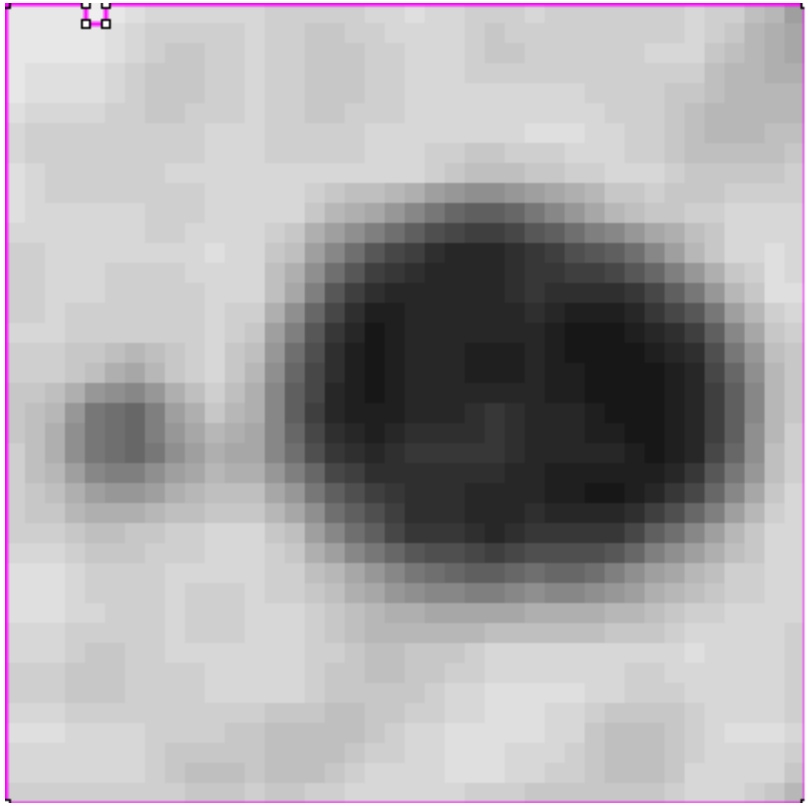


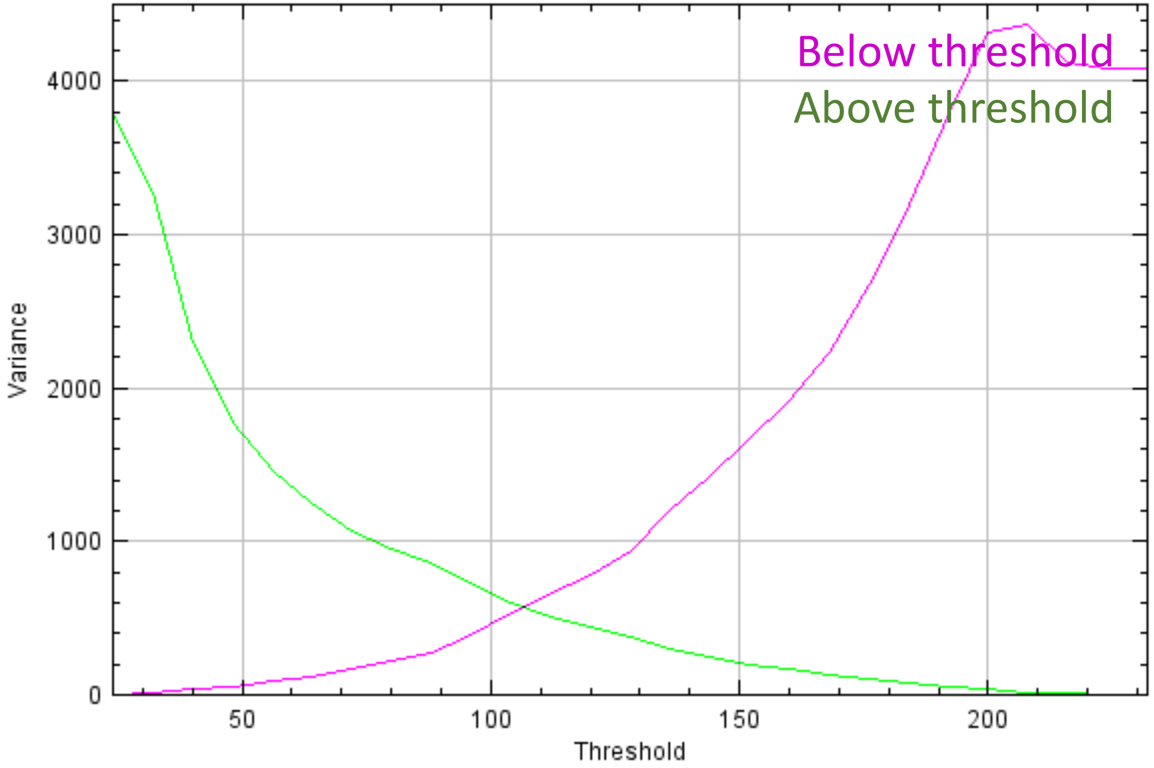
Otsu's method

- Searching for a threshold where the variance in both classes (above/below threshold) becomes minimal.

$$Var(I) = \sum_{i \in I} g_i - \bar{g}_I \qquad \bar{g}_I = \sum_{i \in I} \frac{g_i}{n_I}$$

$Var(I)$ … Variance in image I
g_i … grey value of a pixel i
$\bar{g}_I$ … mean grey value of the whole image I
n_I … number of pixels in Image I

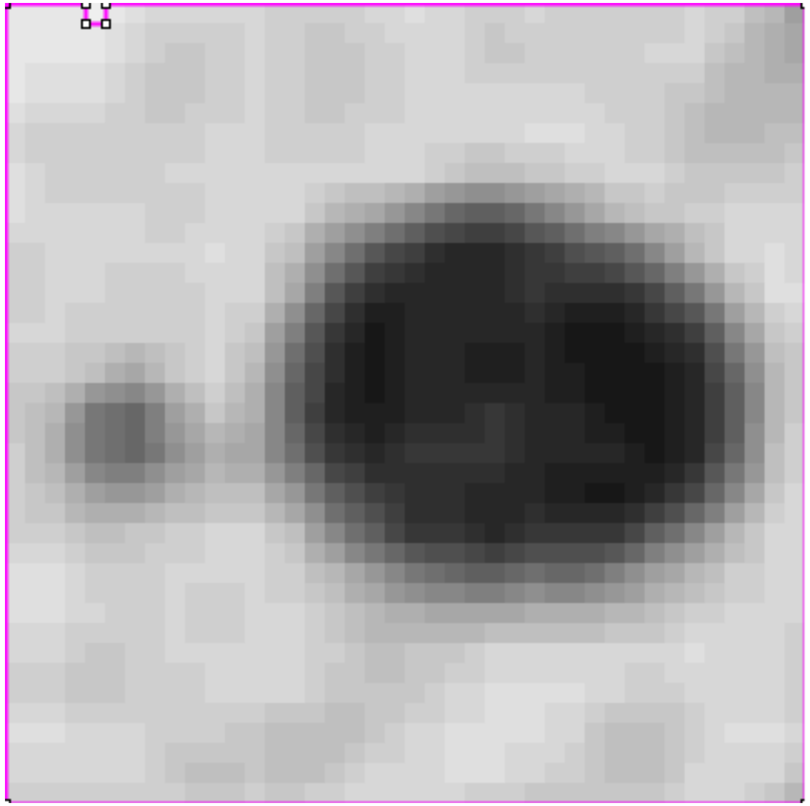


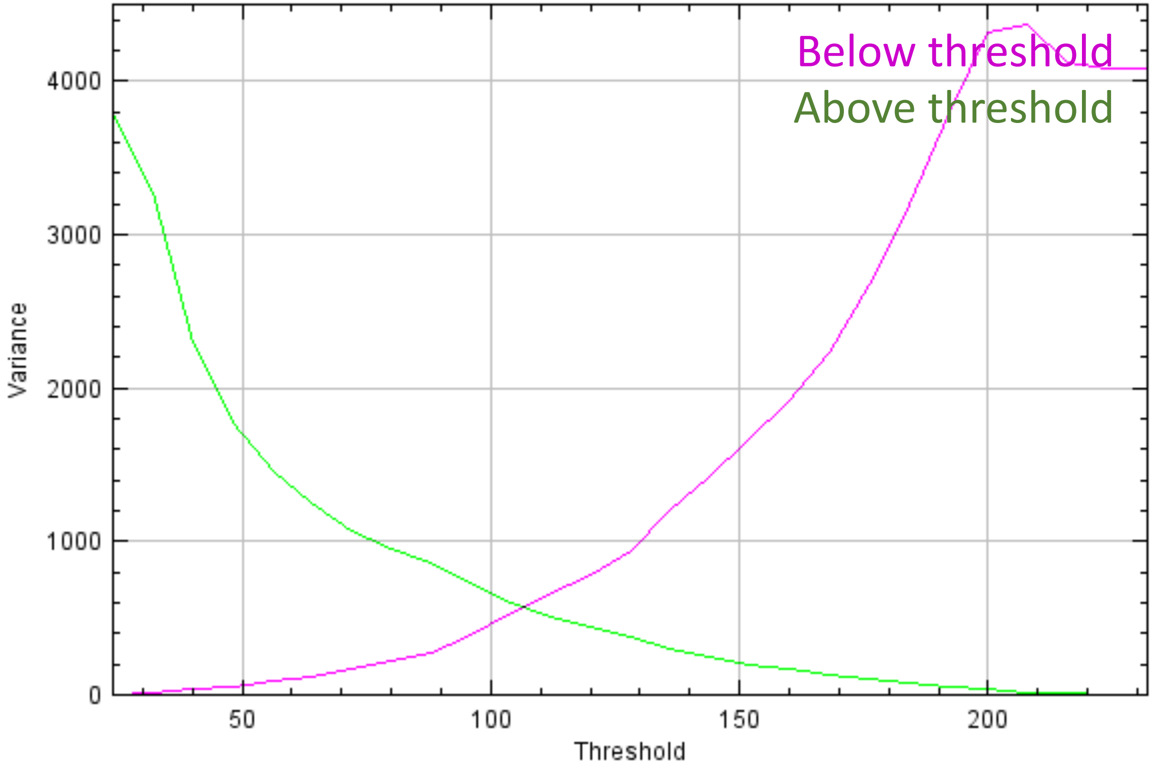


Otsu's method

- Searching for a threshold where the variance in both classes (above/below threshold) becomes minimal.
- Weighted (!) sum variance

$$Var'(I) = \frac{n_A}{n_I} Var(A) + \frac{n_B}{n_I} Var(B) \qquad I = A \cup B$$

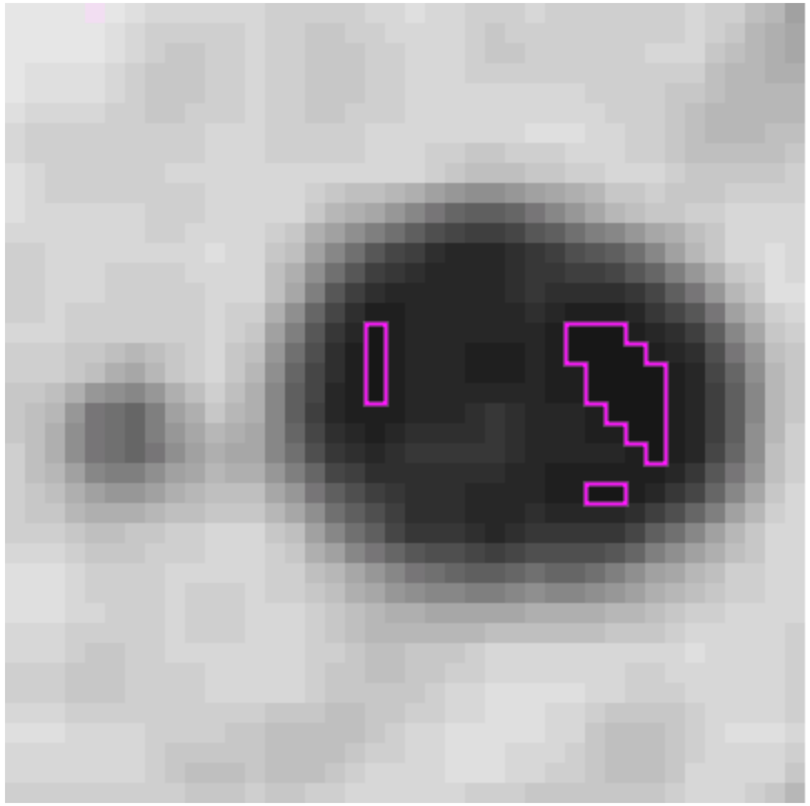


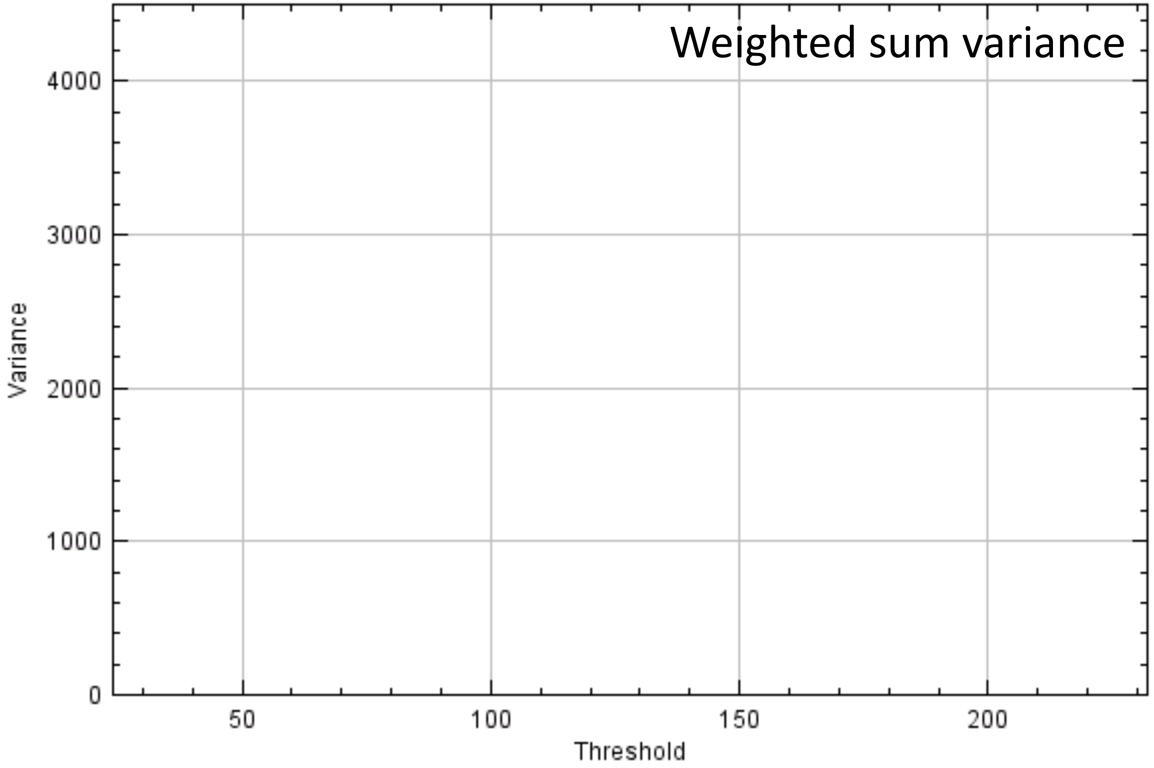


Otsu's method

- Searching for a threshold where the variance in both classes (above/below threshold) becomes minimal.
- Weighted (!) sum variance

$$Var'(I) = \frac{n_A}{n_I} Var(A) + \frac{n_B}{n_I} Var(B) \qquad I = A \cup B$$

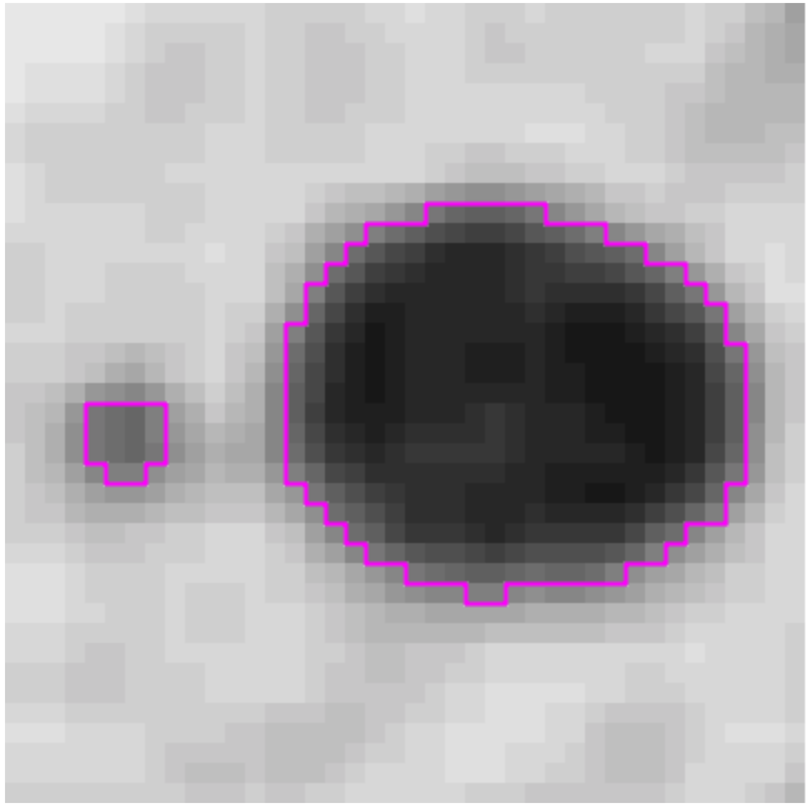


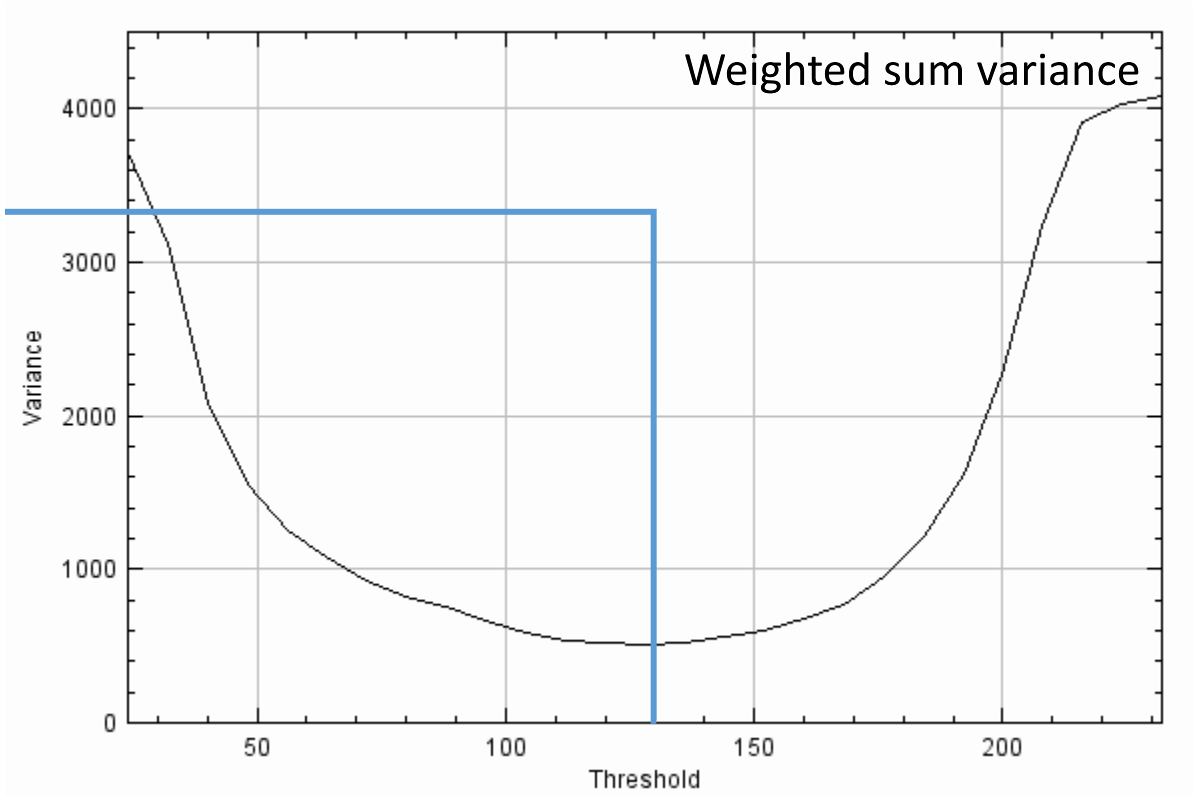


Otsu's method

- Searching for a threshold where the variance in both classes (above/below threshold) becomes minimal.
- Weighted (!) sum variance

$$Var'(I) = \frac{n_A}{n_I} Var(A) + \frac{n_B}{n_I} Var(B) \qquad I = A \cup B$$





See also: http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html

Thresholding: Citing

- Cite the thresholding method of your choice properly

We segmented the cell nuclei in the images using the Otsu thresholding method (Otsu et Al. 1979) implemented in Fiji (Schindelin et Al. 2012).

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

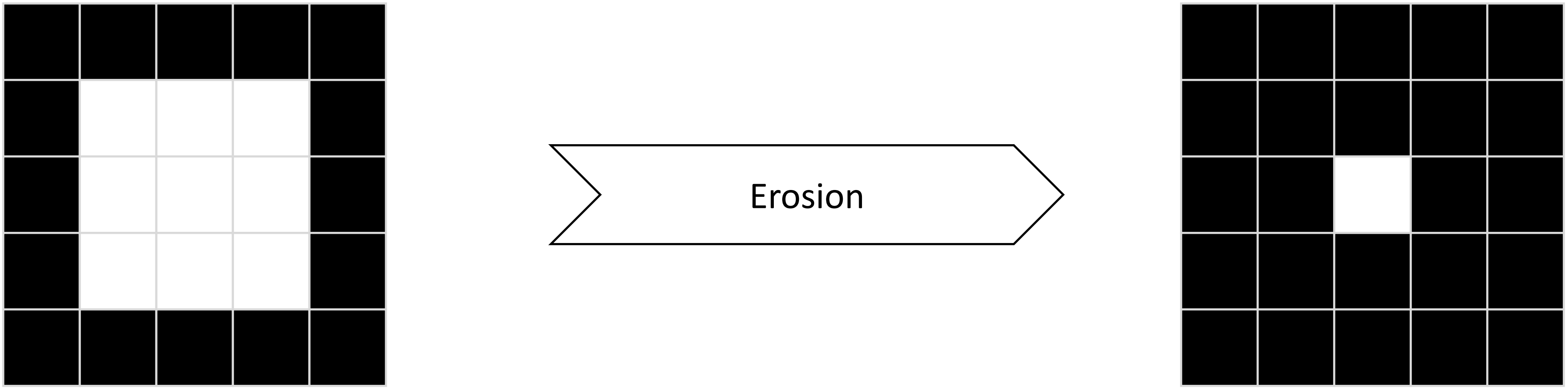
Refining masks

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

Thresholding → Binary closing (Fill holes) → Binary opening

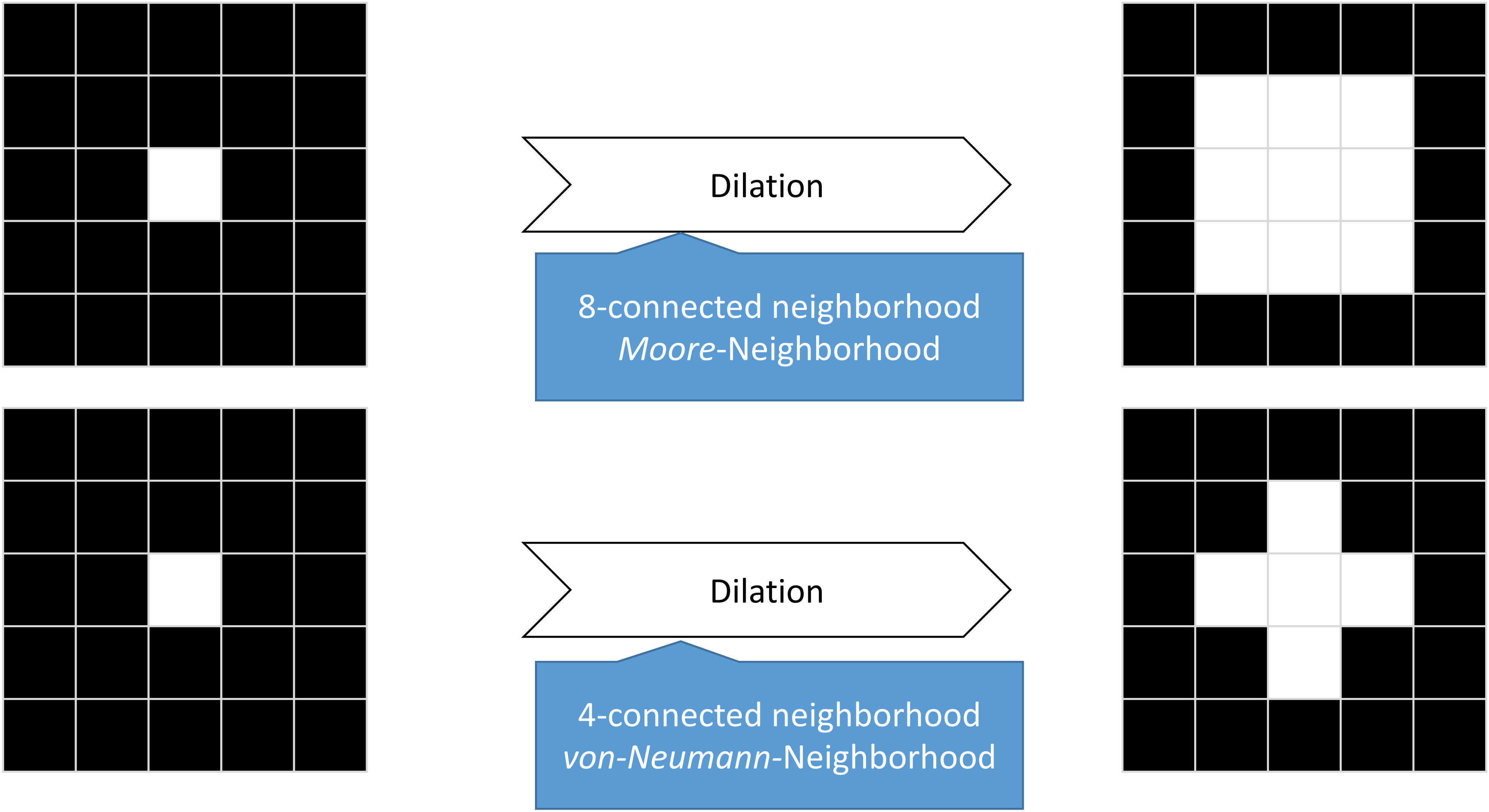
Refining masks: Erosion

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



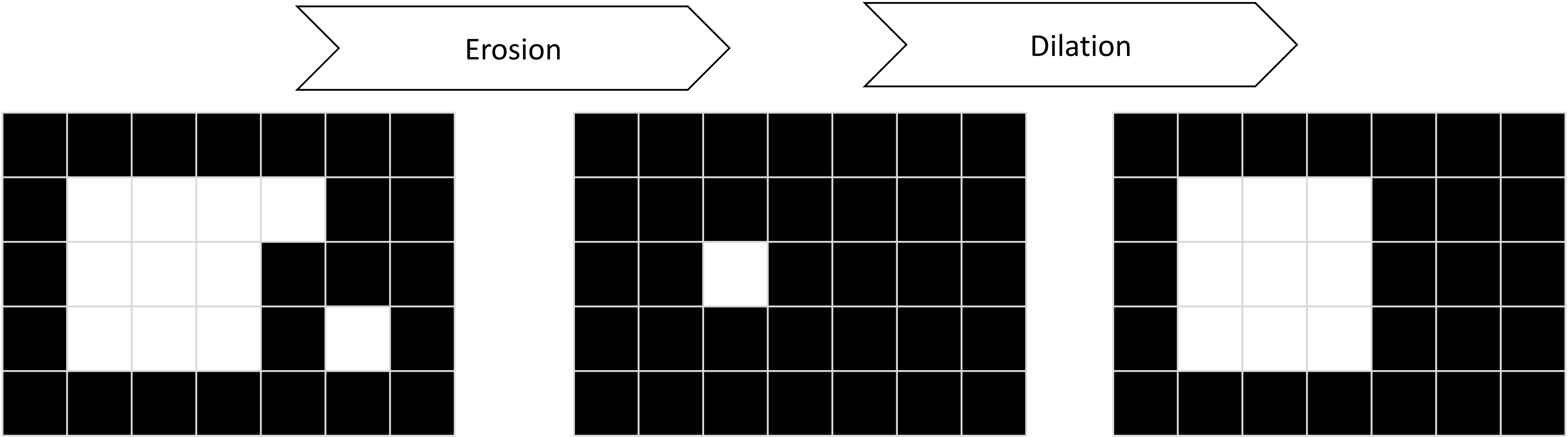
Refining masks: Dilation

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



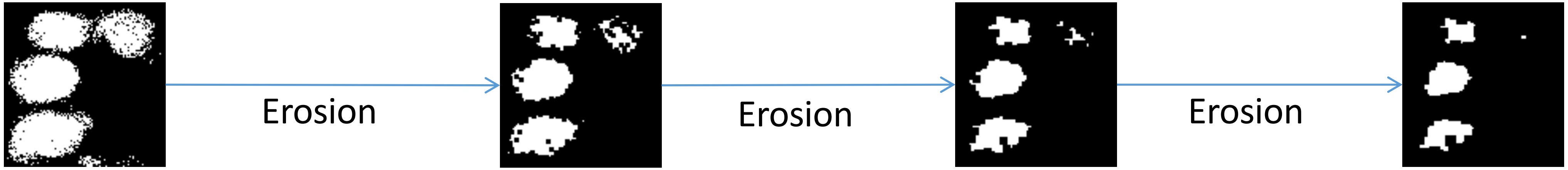
Refining masks: Erosion & Dilation

- Erosion and dilation combined allow correcting outlines.



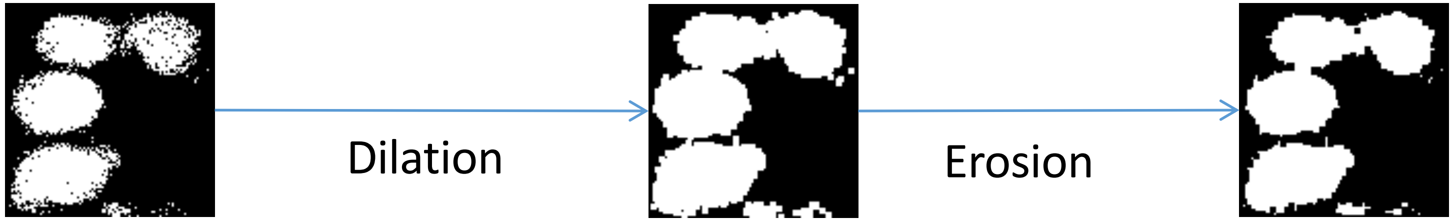
Refining masks: Erosion / dilation

- 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

Exercises for lunch: