Image thresholding and feature extraction

Johannes Müller

With material from

Robert Haase, BiaPoL, PoL TU Dresden

Marcelo Zoccoler, BiaPol, PoL, TU Dresden

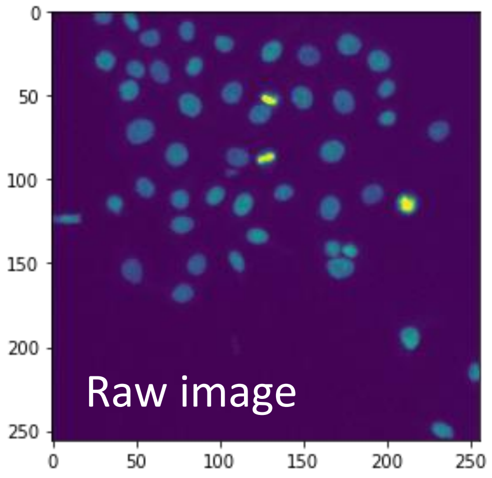
Benoit Lombardot, Scientific Computing Facility, MPI CBG

May 2022

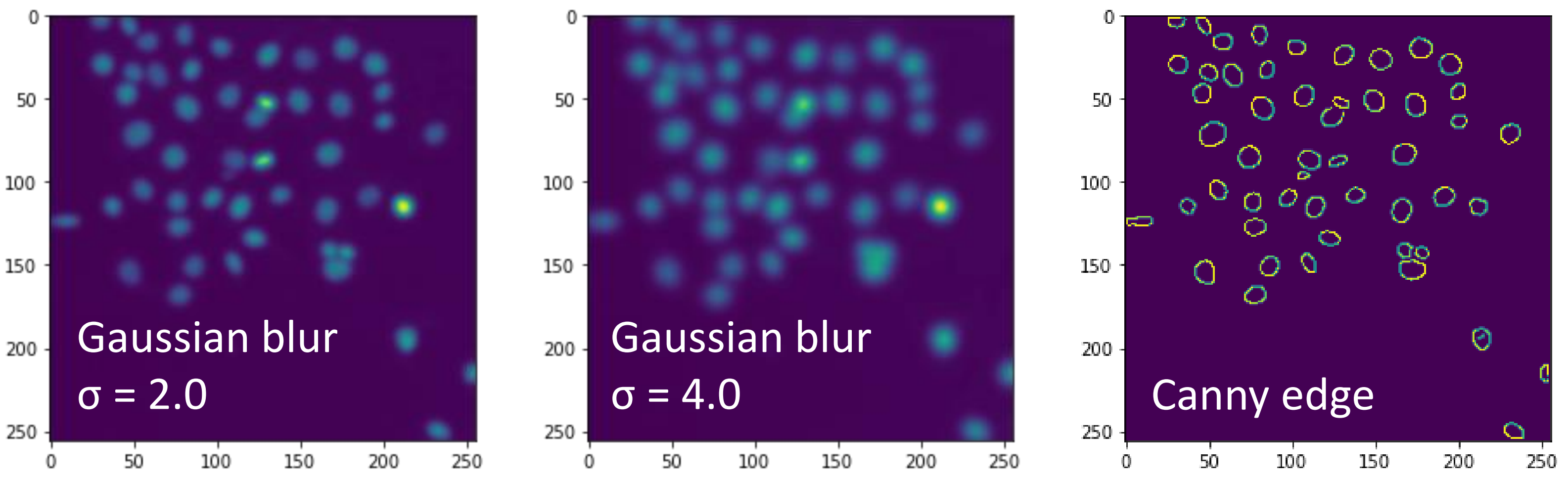
@jm_mightypirate

Recap: Filters & Histograms

- **Image filtering**



Filtered images (aka *feature maps*)

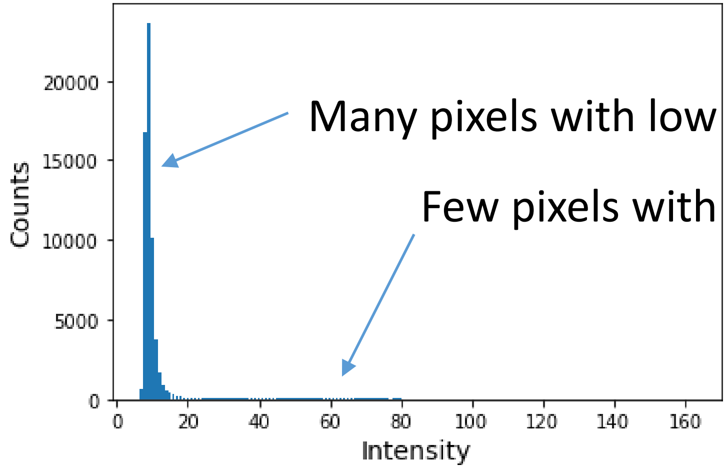



```python
from skimage import filters, feature
```

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

- **Histograms**

Count number of pixels within specific ranges ("bins") of intensity



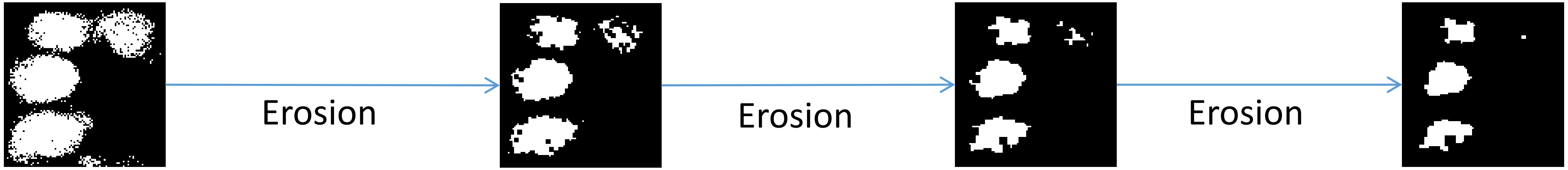
Many pixels with low intensity → background

Few pixels with higher intensity → foreground

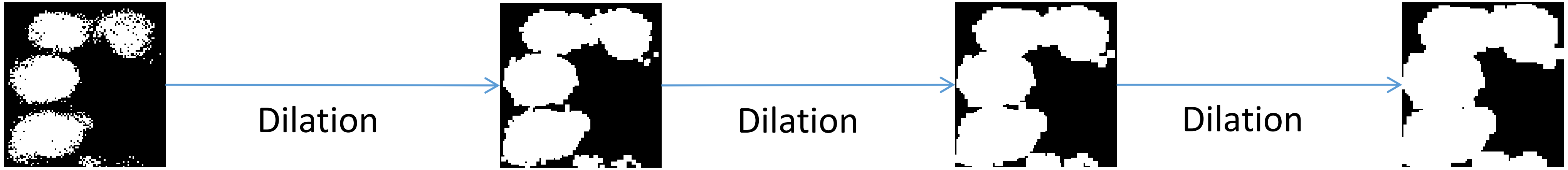
```python
histogram = skimage.exposure.histogram(image)
plt.bar(histogram[1], histogram[0])
```

Recap: Filters on binary images

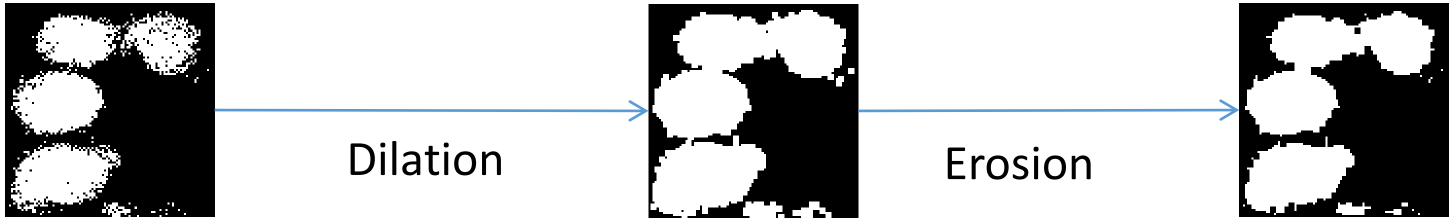
- Erosion: Set all pixels to black which have at least one black neighbor.



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



- Closing: Dilation + Erosion



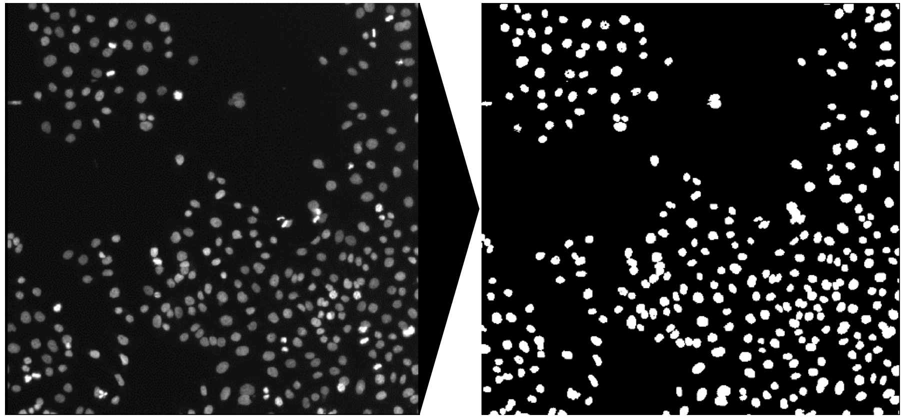
- Opening: Erosion + Dilation

Lecture overview

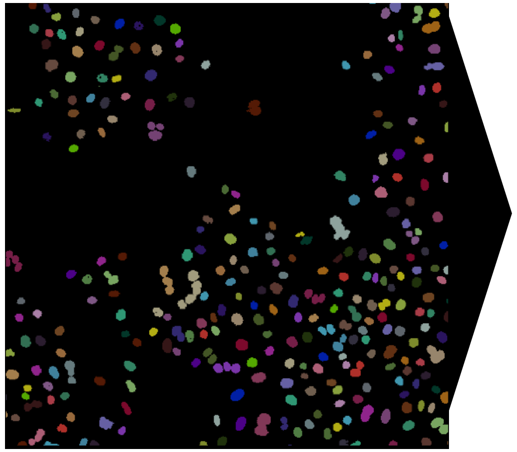
Today’s content:

Complete workflow from raw images toward extracting quantitative features from image data

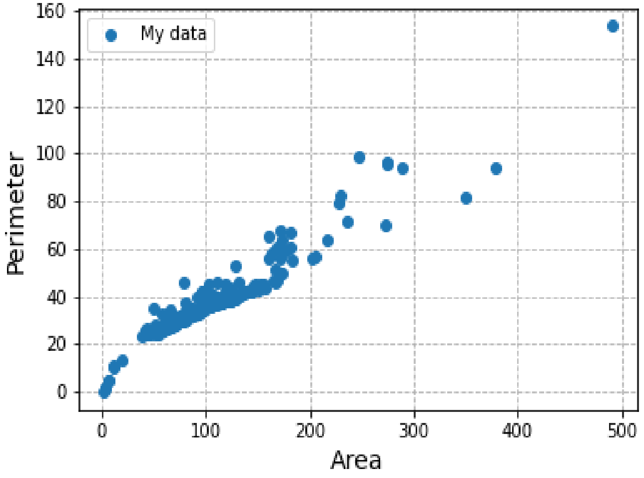
- Creating binary masks from intensity images (**Segmentation**)
  → Thresholding
- Create label images from binary masks
  → Connected-component analysis
- Measure features from label images
  → Shape, Intensity, etc.
  → Visualize measured features



Labelled image



Measurement



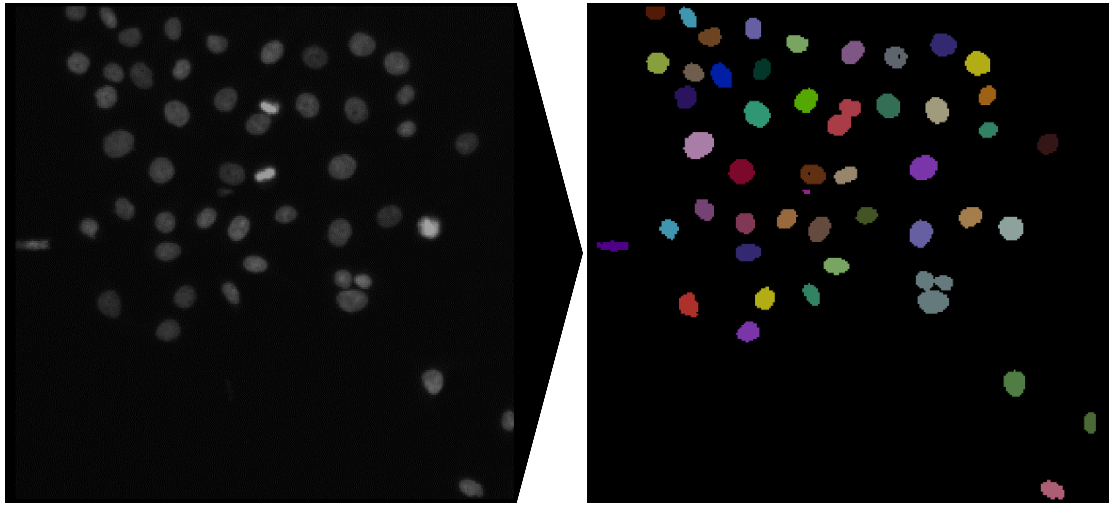
Segmentation

Aim:

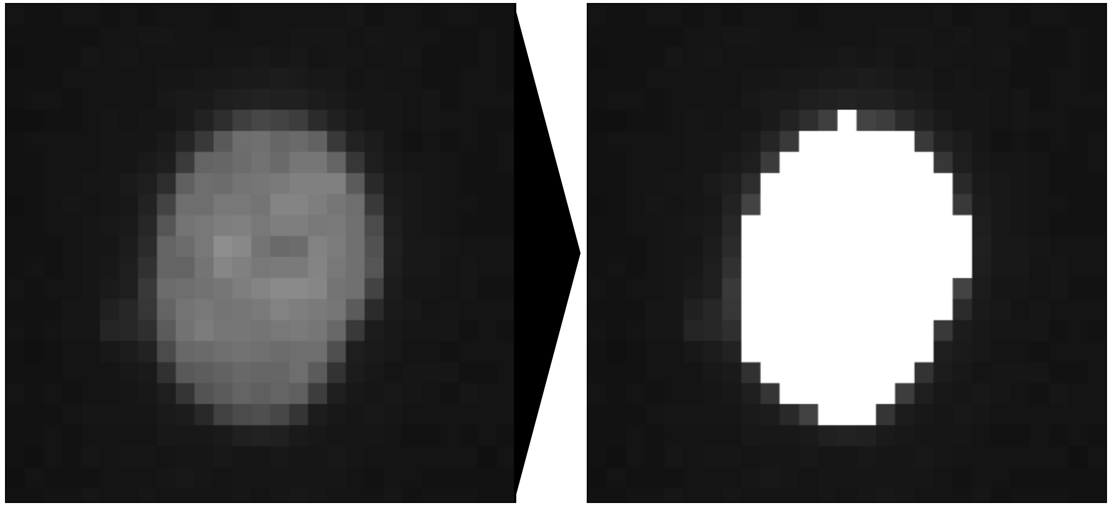
Separate background from foreground

Vocabulary:

- **Segmentation:**
  Assigning a meaningful *label* to each pixel
- **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

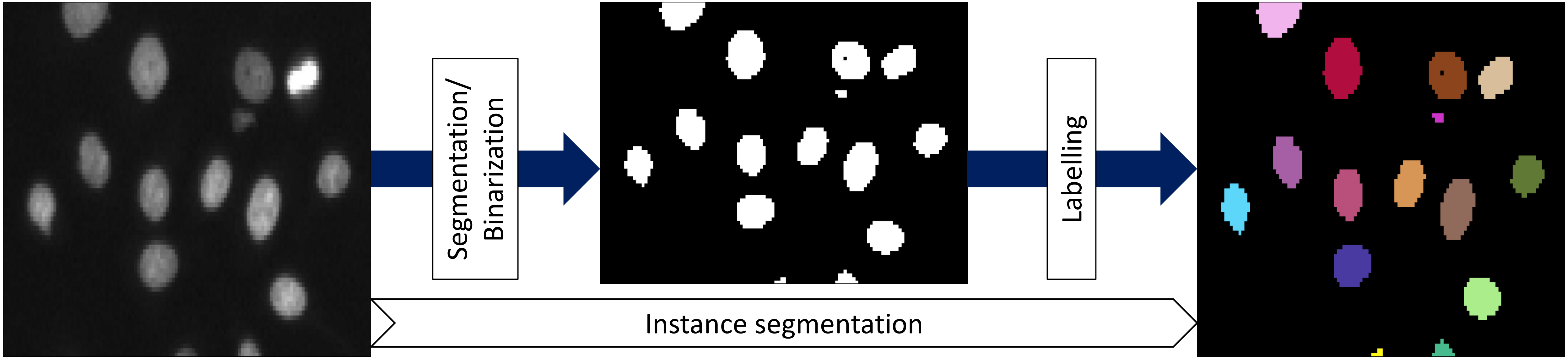


Semantic segmentation

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:

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

```
False
True
False
```

```
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]])
```

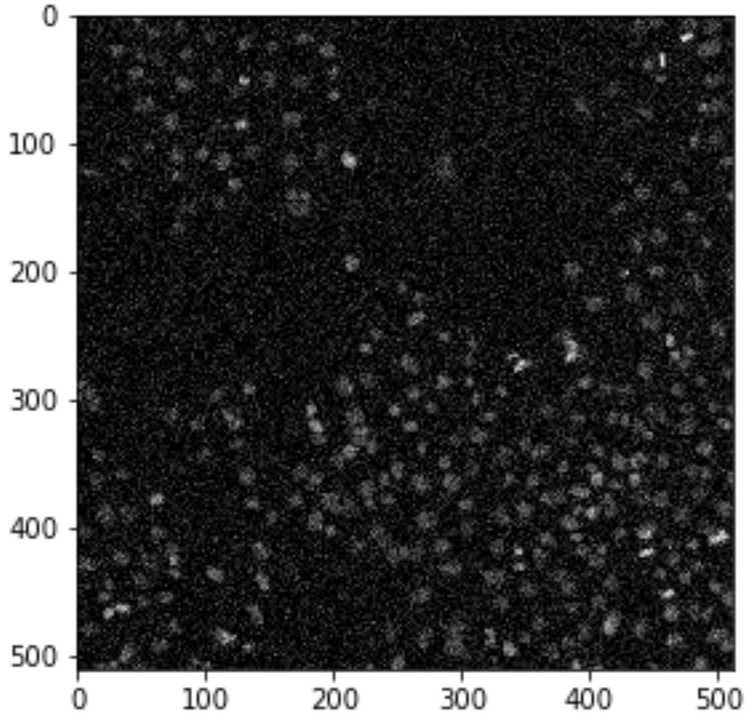
In this case, “image” is a *numpy array* → some operations are automatically applied to every pixel!

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

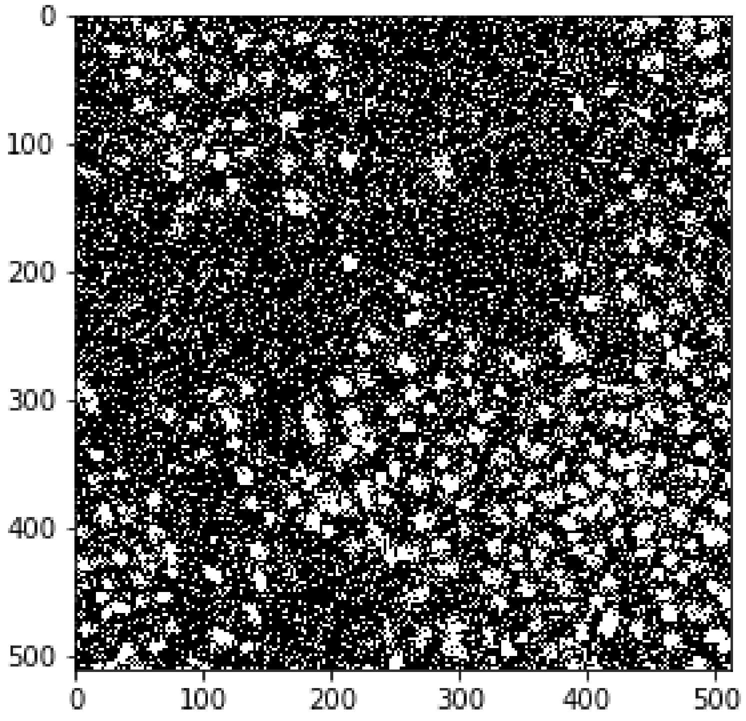
```
binary = image > threshold
```

Reminder: pre-processing!

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



Noisy image



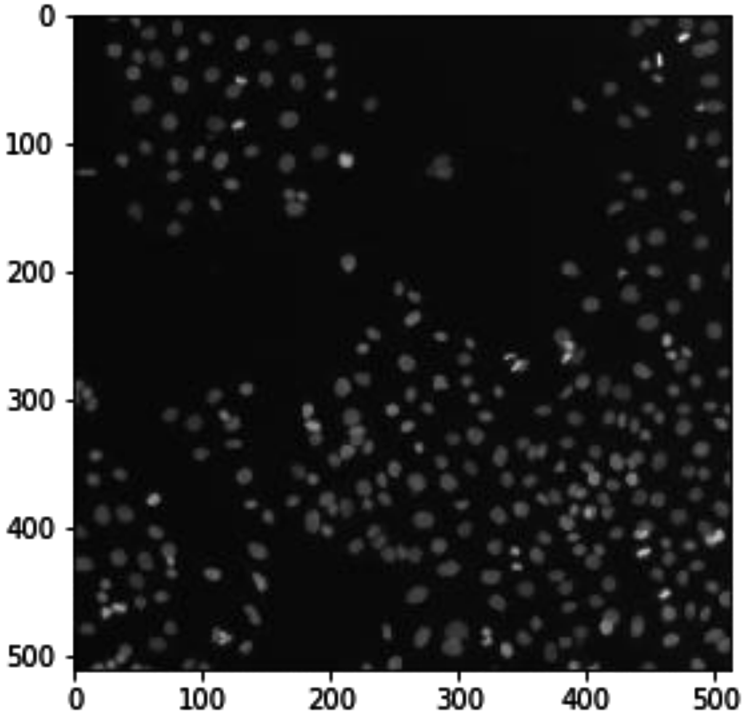
Thresholded image

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

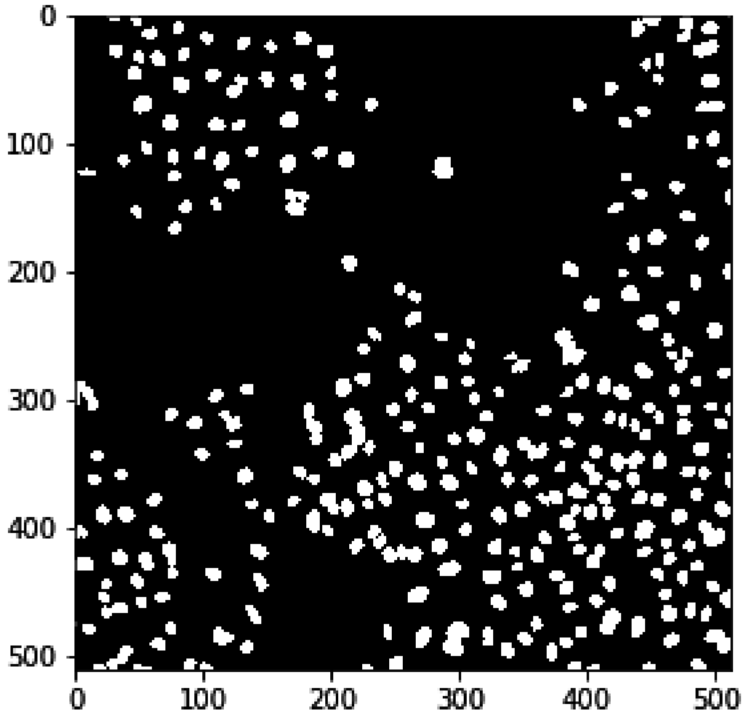
Image filtering *filters* relevant information for subsequent operations from the image!

Reminder: pre-processing!

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



Filtered image



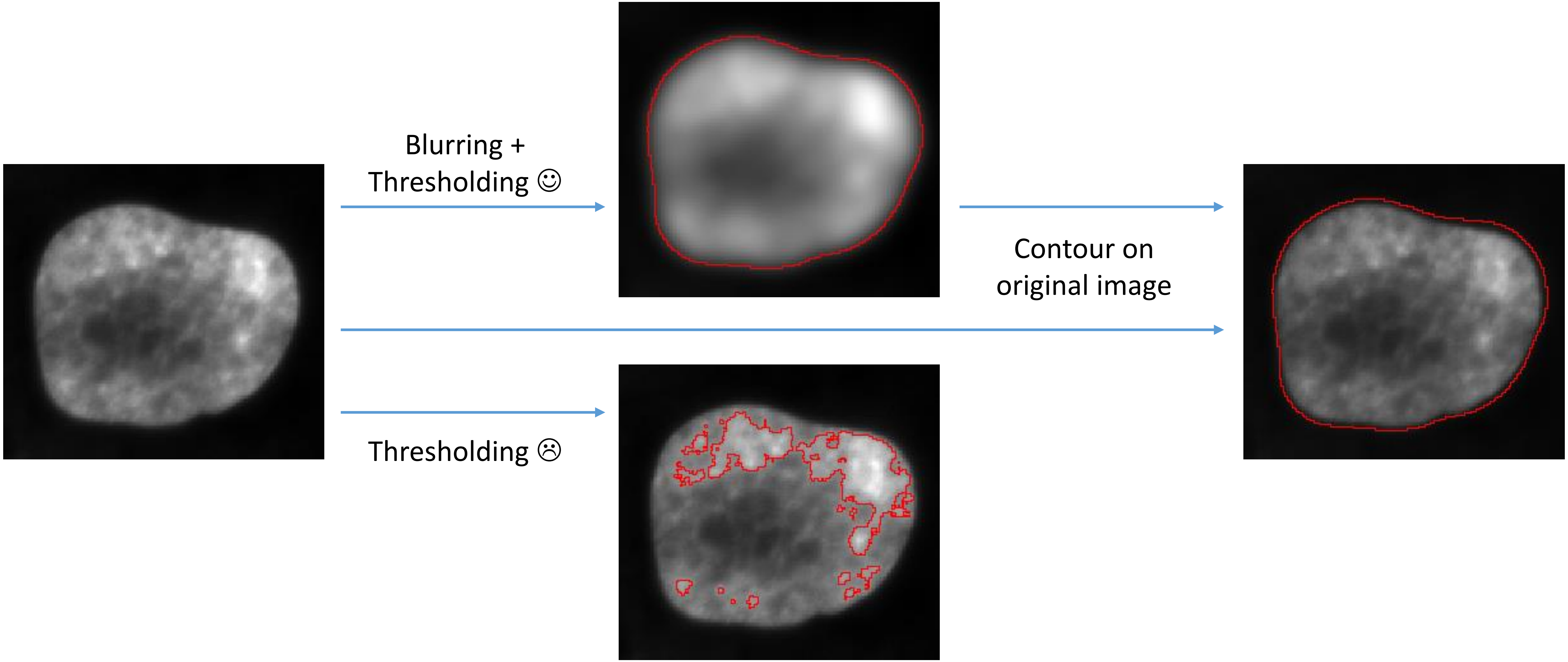
Thresholded image

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

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

Low-pass filtering to improve thresholding results

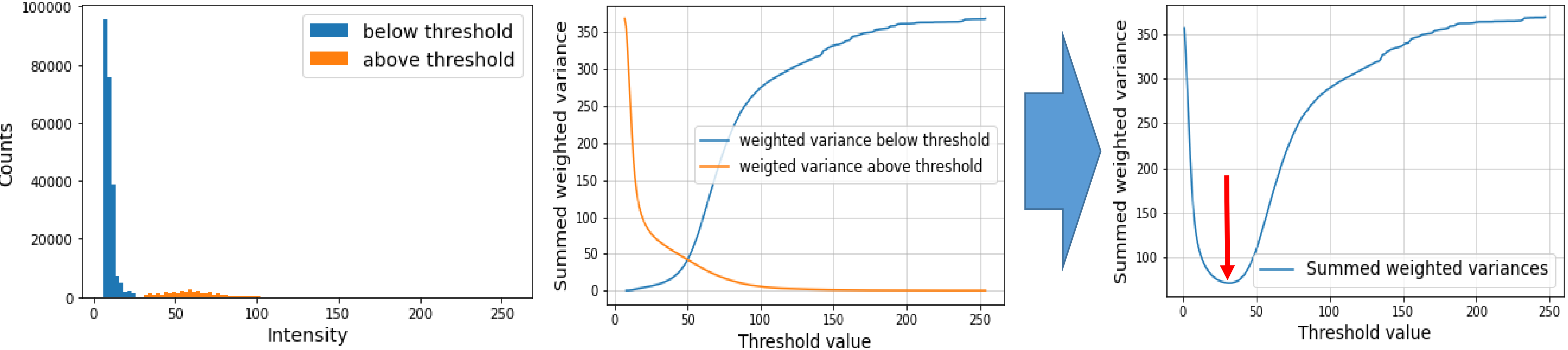
- In case thresholding algorithms outline the wrong structure, blurring in advance may help.
- However: **Do not** continue processing the blurred image, continue with the original!



Threshold algorithms

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

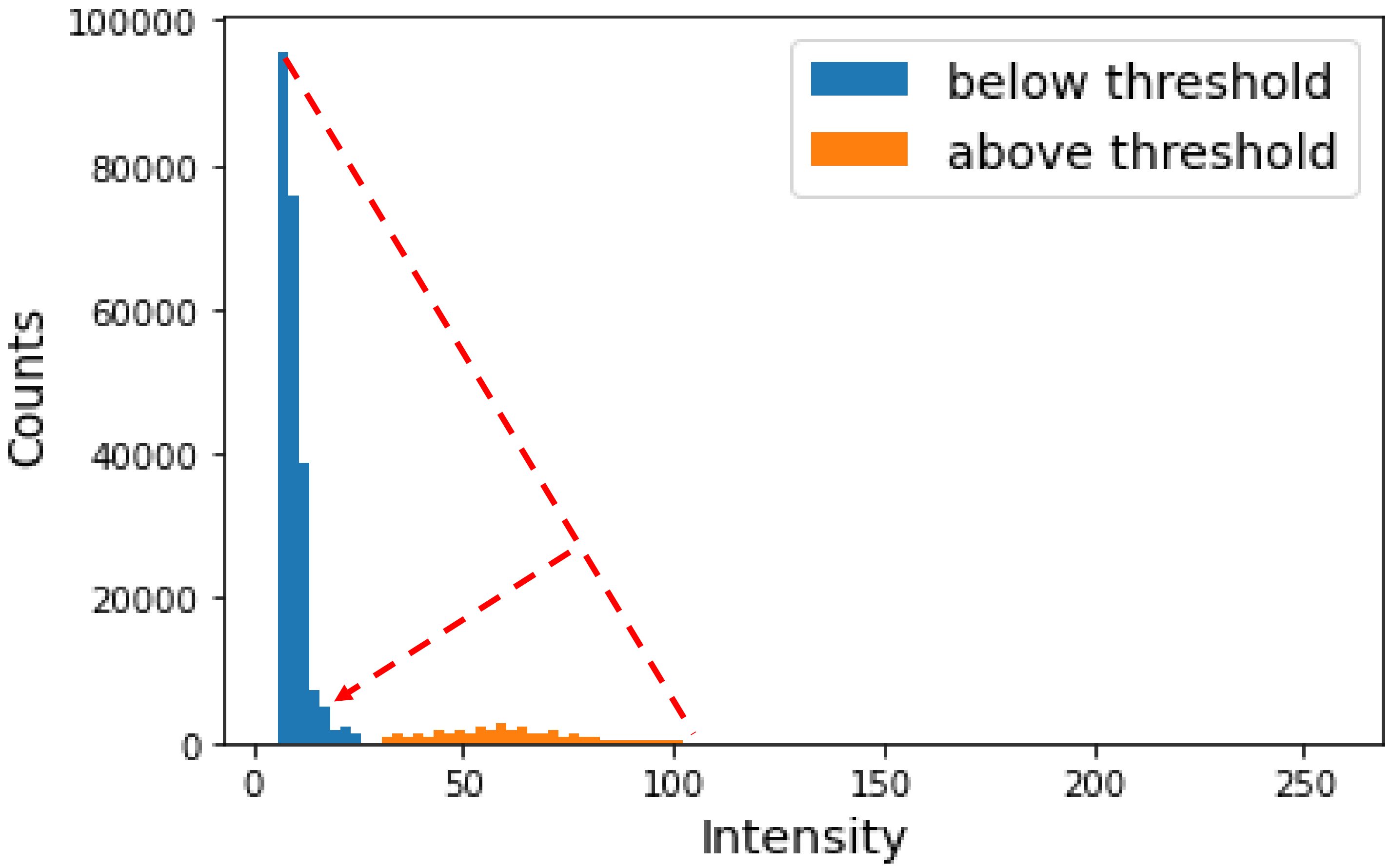
$$Var(I) = \frac{1}{n_I}\sum\left(I - mean(I)\right)^2 \quad \rightarrow \quad Var_{w,sum} = \frac{n_A}{n_I}\cdot Var(A) + \frac{n_B}{n_I}\cdot Var(B)$$



- **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 parameter of I belong to foreground. (Possibilities: Mean, Median, Quartiles, etc.)

```
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

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

@jm_mightypirate

Thresholding: Pitfalls

```
binary = image > a_good_threshold_value_of_my_choice
```

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

Inter-observer variability

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:

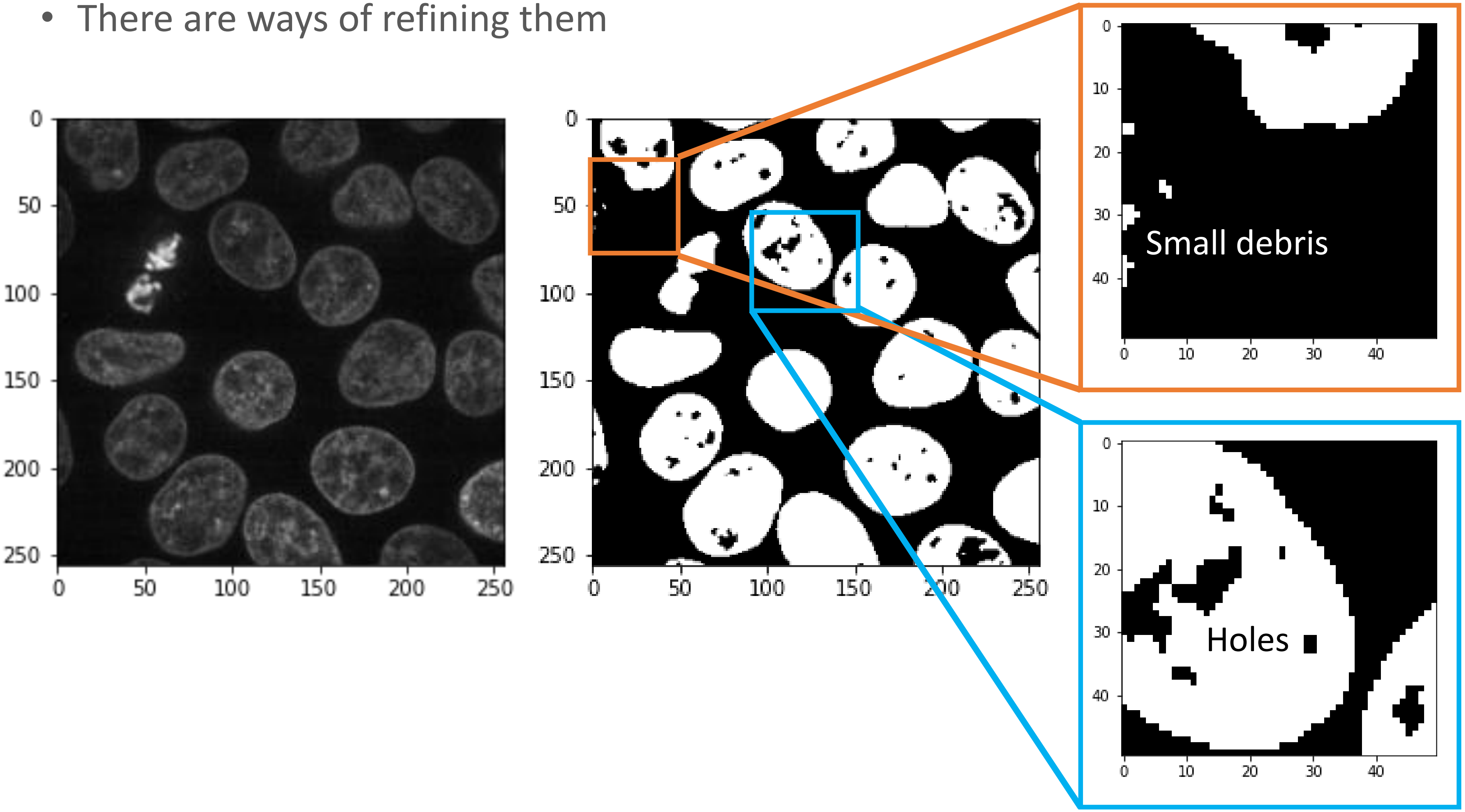
…and stick to it for the whole study. Using a new method for every image impairs reproducibility!

Do not over-engineer

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

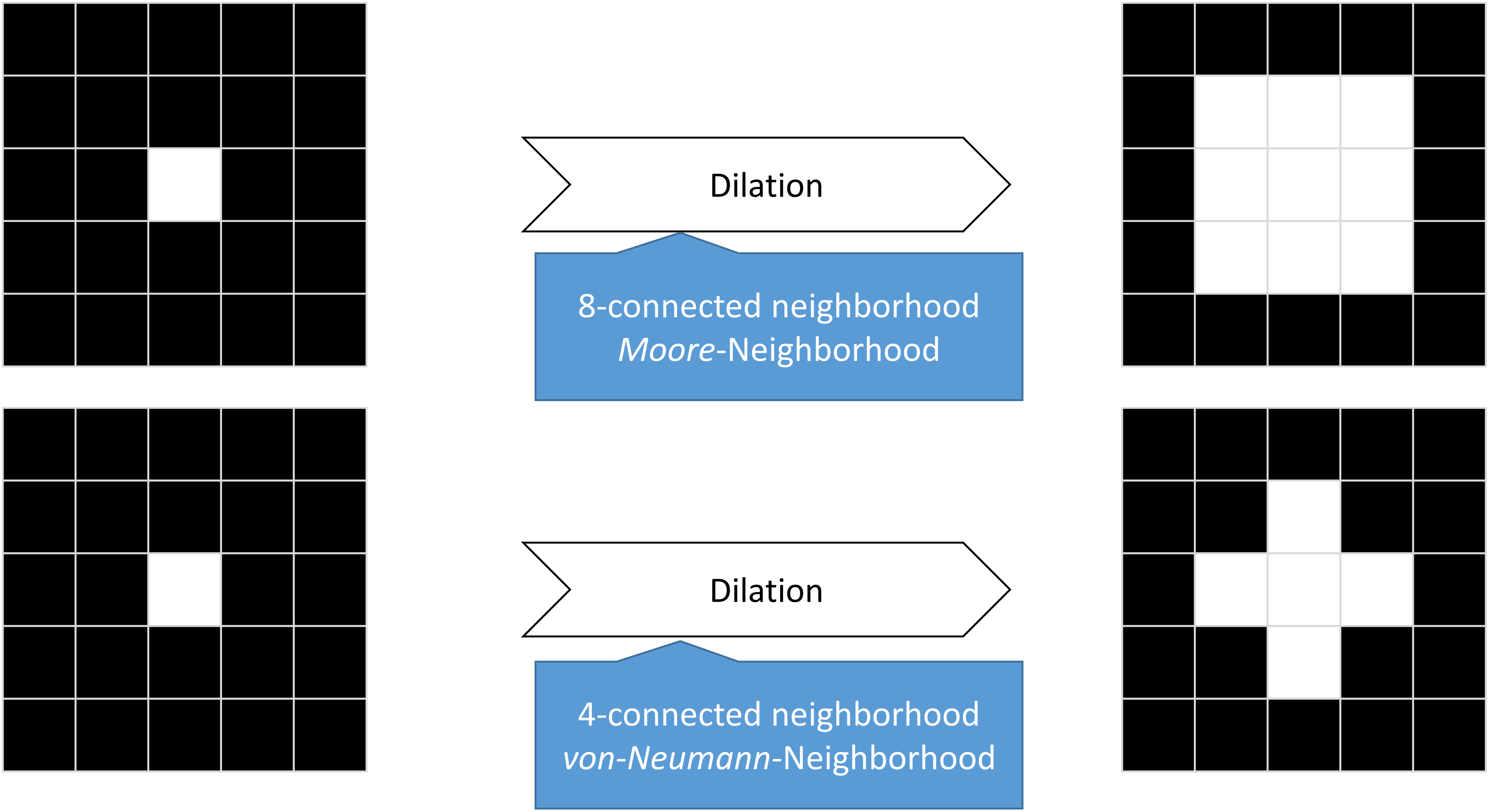
Refining masks

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



Refining masks: Dilation

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

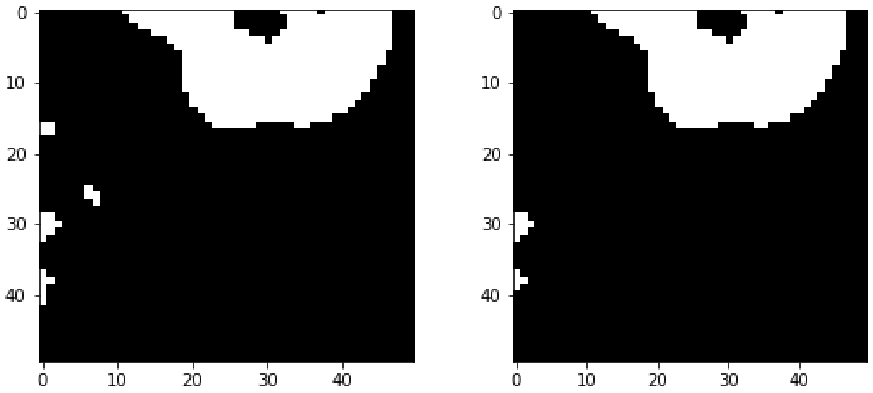


Refining masks: Opening & Closing

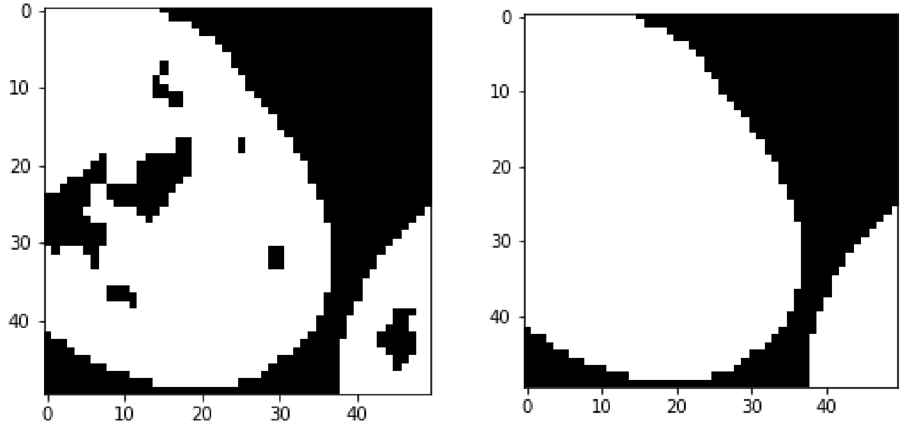
PoL
Physics of Life
TU Dresden

```python
from skimage import morphology
```

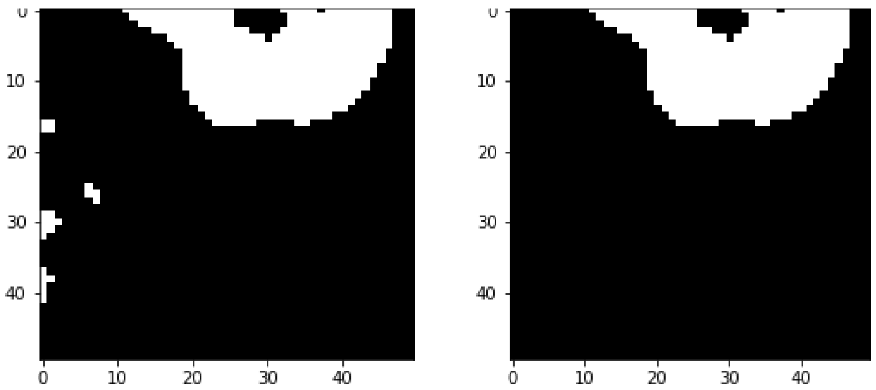
```python
opened = morphology.binary_opening(binary)
```



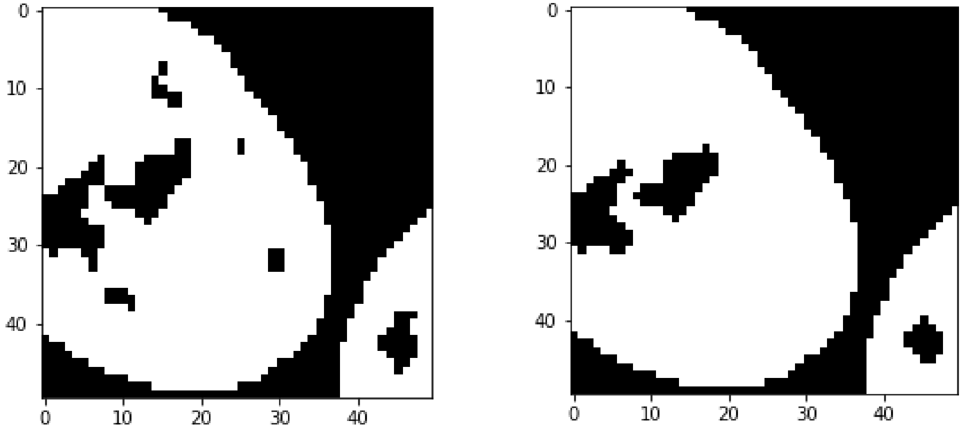
```python
closed = morphology.area_closing(binary, area_threshold=100)
```



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

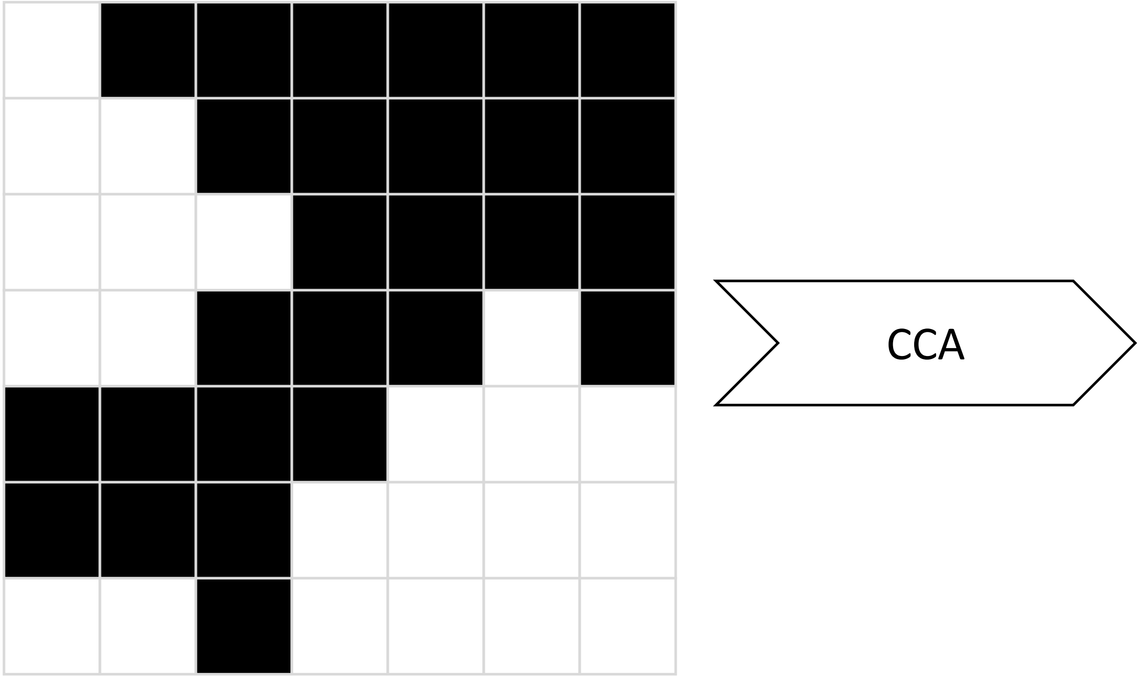


```python
closed = morphology.binary_closing(binary)
```



Connected components labelling

- In order to allow the computer differentiating objects, connected components analysis (CCA) is used to mark pixels belonging to different objects with different numbers
- Background pixels are marked with 0.
- The maximum intensity of a labelled map corresponds to the number of objects.



1	0	0	0	0	0	0
1	1	0	0	0	0	0
1	1	1	0	0	0	0
1	1	0	0	0	3	0
0	0	0	0	3	3	3
0	0	0	3	3	3	3
2	2	0	3	3	3	3

Label map visualisation

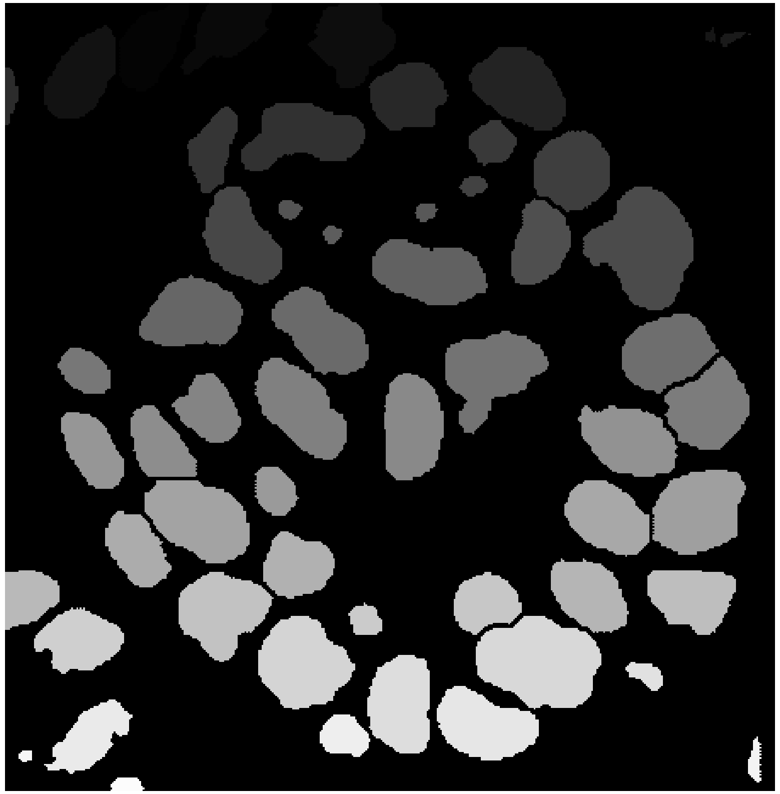
- Label maps can be nicely visualized with the right lookup table

Grey

Pixel value	Display color
0 1 2 … 255	

Glasbey

Pixel value	Display color
0 1 2 … 255	



Reminder: Vocabulary

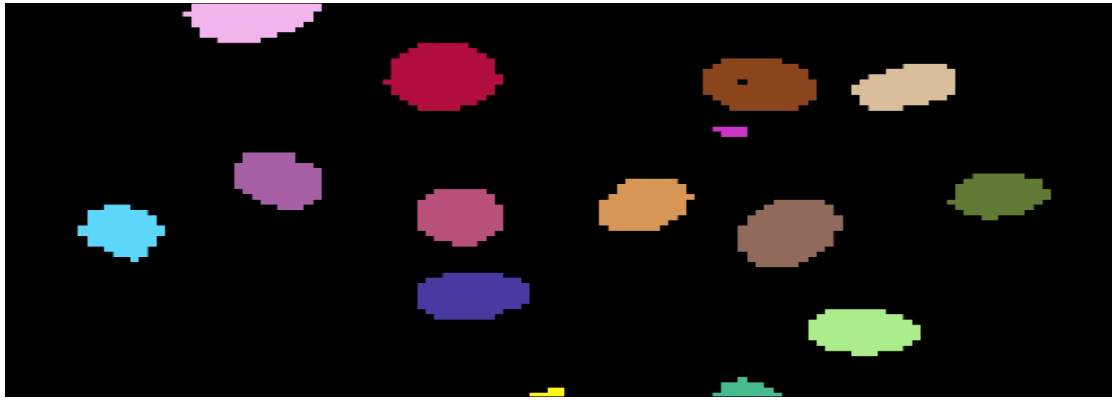
1	0	0	0	0	0	0
1	1	0	0	0	0	0
1	1	1	0	0	0	0
1	1	0	0	0	3	0
0	0	0	0	3	3	3
0	0	0	3	3	3	3
2	2	0	3	3	3	3

Instance segmentation

Label 1 = Nucleus #1

Label 2 = Nucleus #2

Label 3 = Nucleus #3

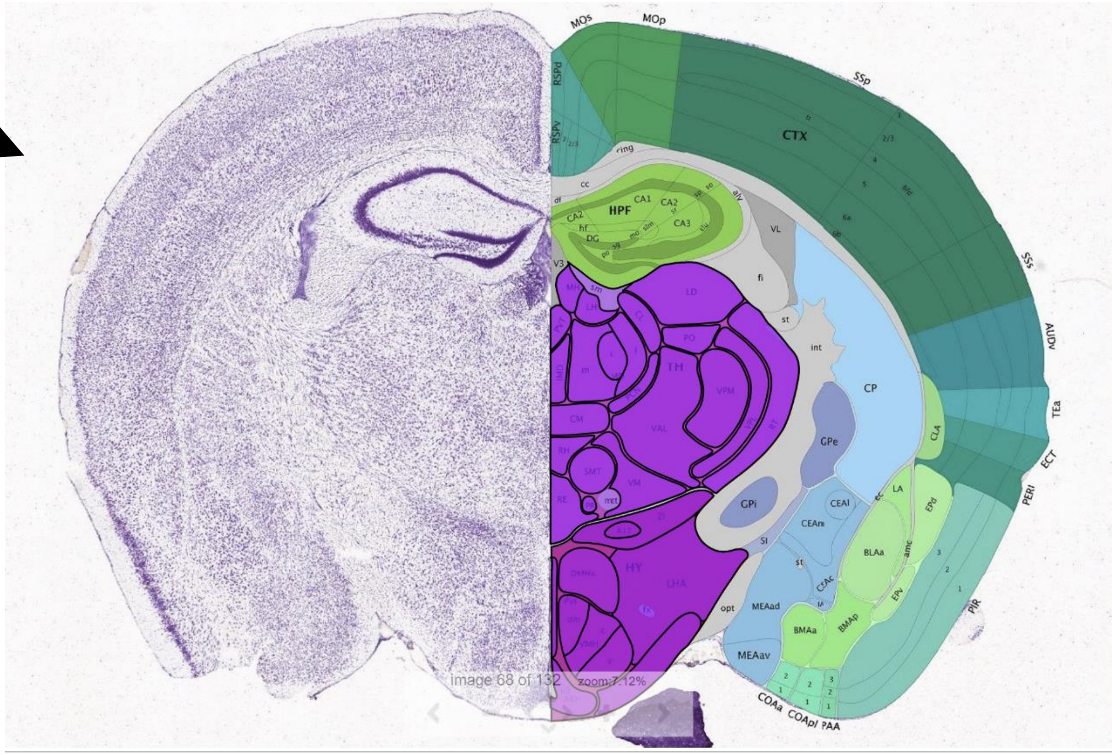


Semantic segmentation

Label 1 = Cortex

Label 2 = Cerebellum

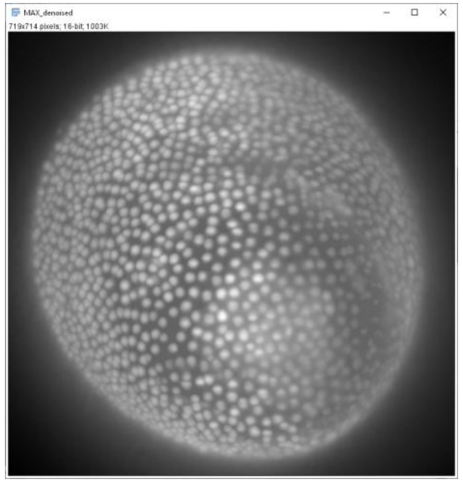
Label 3 = Hippocampus



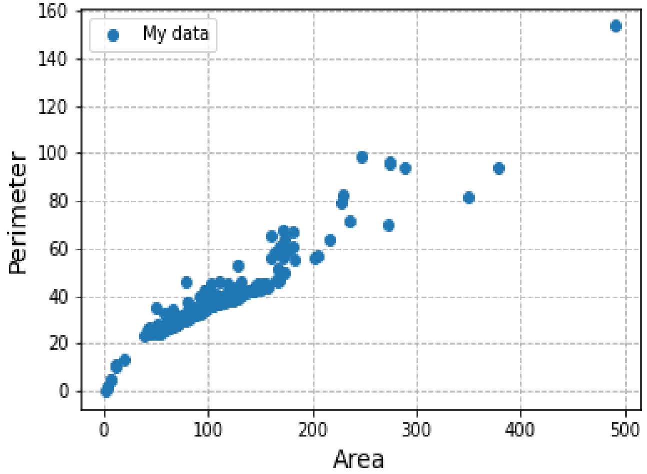
Allen adult mouse atlases,
https://atlas.brain-map.org

Feature extraction

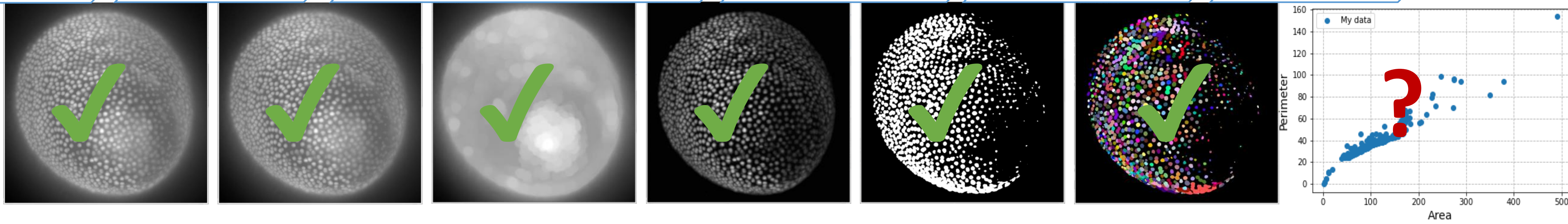
- Feature extraction is a *late* processing step in image analysis.
- It can be used for images, or segmented/labelled images



Feature Extraction



Acquisition → Denoising → Background subtraction → Segmentation → Labeling → Feature Extraction



@jm_mightypirate

May 2022

Image data source: Daniela Vorkel, Myers lab, MPI CBG

Feature extraction

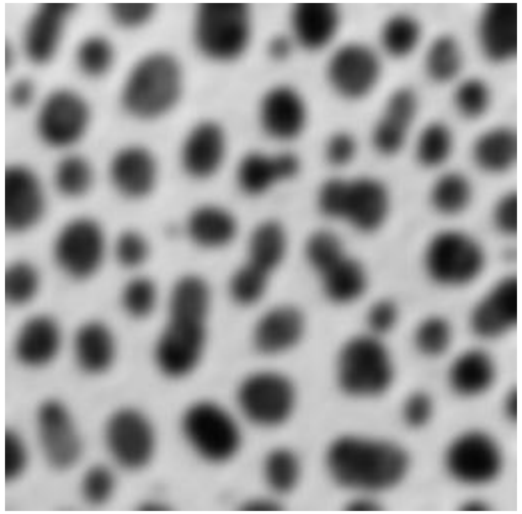
- A *feature* is a countable or measurable property of an image or object.
- Goal of feature extraction is finding a minimal set of features to describe an object well enough to differentiate it from other objects.

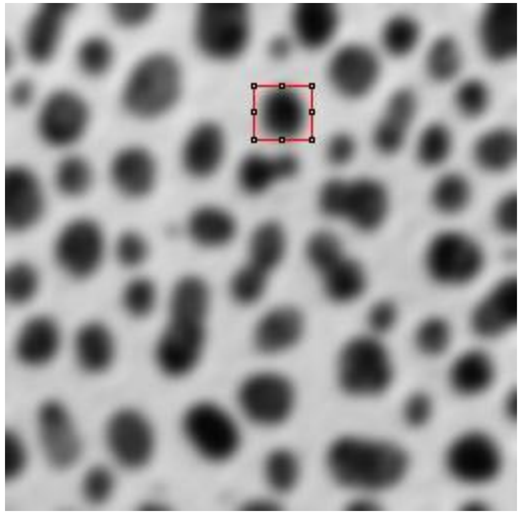
- **Intensity based**
  - Mean intensity
  - Standard deviation
  - Total intensity
  - Textures
- **Shape based /spatial**
  - Area / Volume
  - Roundness
  - Solidity
  - Circularity / Sphericity
  - Elongation
  - Centroid
  - Bounding box
- **Spatio-temporal**
  - Displacement,
  - Speed,
  - Acceleration
- **Others**
  - Overlap
  - Colocalization
  - Neighborhood

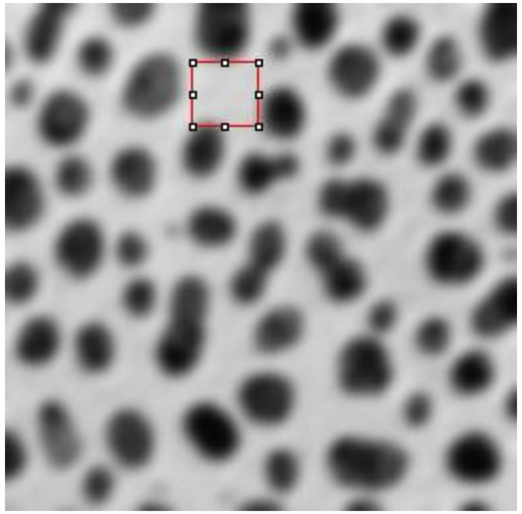
- **Mixed features**
  - Center of mass
  - Local minima / maxima

Intensity based features

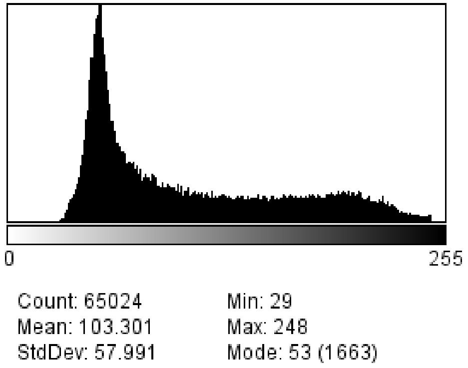
- Min / max
- Median
- Mean
- Mode
- Variance
- Standard deviation

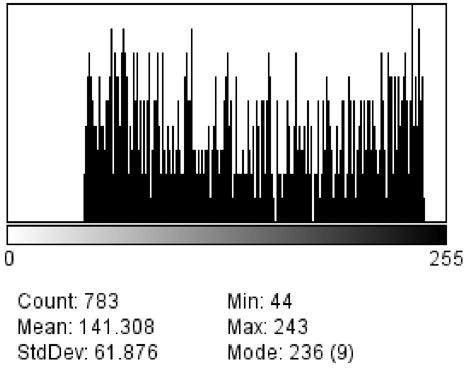


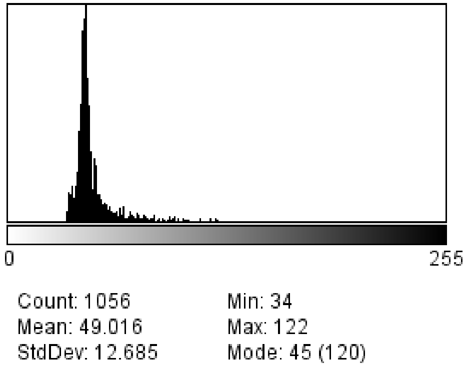




- Can be derived from pixel values
- Don’t take spatial relationship of pixels into account







- See also:
  - descriptive statistics
  - histogram

Bounding rectangle / bounding box

- Position and size of the smallest rectangle containing all pixels of an object
  - x_b, y_b … position of the bounding box
  - w_b … width of the bounding box
  - h_b … height of the bounding box

variable	value
x_b	0
y_b	2
w_b	3
h_b	2

y \ x	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0	0	0
2	1	1	1	0	0
3	0	1	1	0	0
4	0	0	0	0	0

Center of mass

- Relative position in an image weighted by pixel intensities

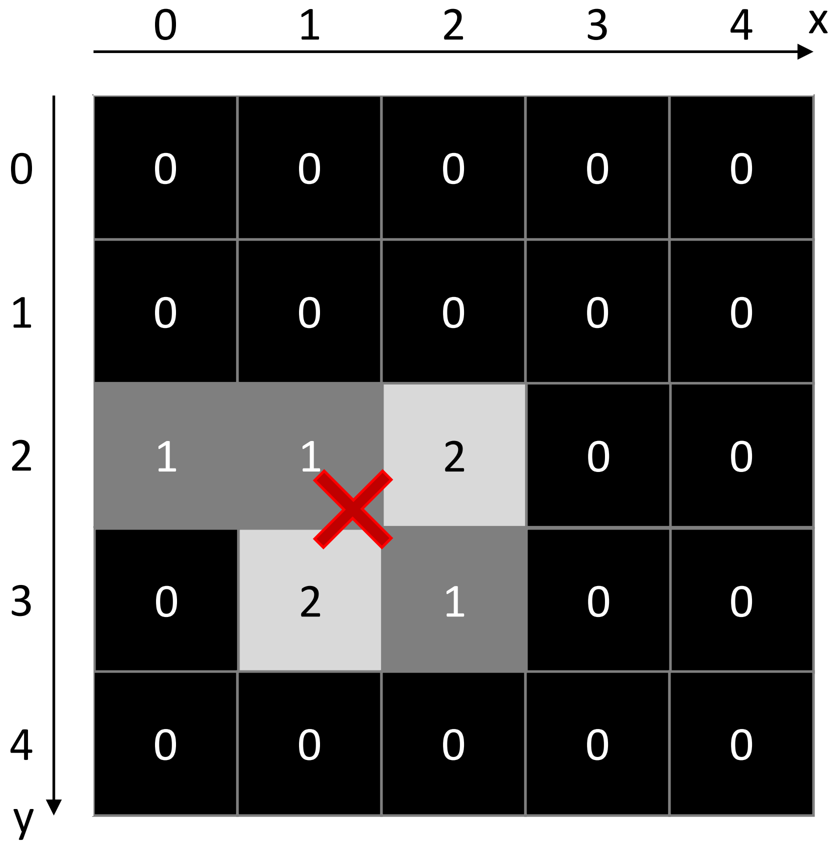
  - x, y … pixel coordinates
  - w … image width
  - h … image height
  - µ … mean intensity
  - $g_{x,y}$ … pixel grey value
  - x_m, y_m … center of mass coordinates

$$\mu = \frac{1}{wh} \sum_{y=0}^{h-1} \sum_{x=0}^{w-1} g_{x,y}$$

$$x_m = \frac{1}{wh\mu} \sum_{y=0}^{h-1} \sum_{x=0}^{w-1} x \, g_{x,y}$$

$$y_m = \frac{1}{wh\mu} \sum_{y=0}^{h-1} \sum_{x=0}^{w-1} y \, g_{x,y}$$

"sum intensity"
"total intensity"



x_m = 1/7 (1·0 + 1·1 + 2·2 + 2·1 + 1·2) = 1.3

y_m = 1/7 (1·2 + 1·2 + 2·3 + 2·2 + 1·3) = 2.4

Center of geometry / centroid

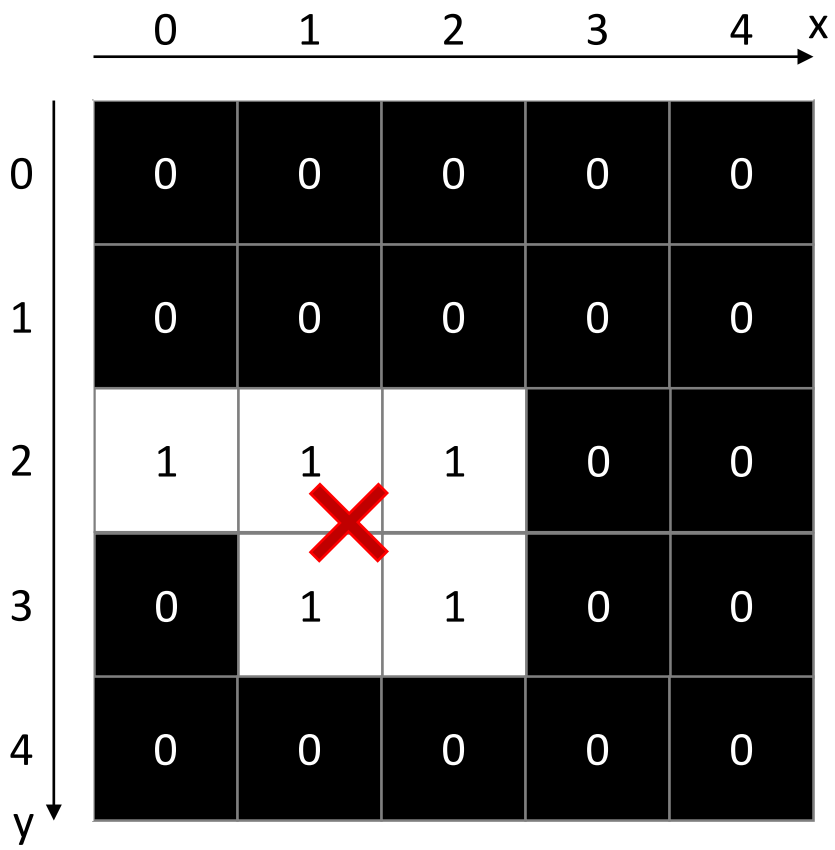
- Relative position in an image weighted by pixel intensities
- Special case of center of mass for binary images
  - x, y … pixel coordinates
  - w … image width
  - h … image height
  - µ … mean intensity
  - $g_{x,y}$ … pixel grey value, integer in range [0;1]
  - x_m, y_m … center of mass coordinates

$$\mu = \frac{1}{wh}\sum_{y=0}^{h-1}\sum_{x=0}^{w-1} g_{x,y}$$

$$x_m = \frac{1}{wh\mu}\sum_{y=0}^{h-1}\sum_{x=0}^{w-1} x\, g_{x,y}$$

$$y_m = \frac{1}{wh\mu}\sum_{y=0}^{h-1}\sum_{x=0}^{w-1} y\, g_{x,y}$$

Number of white pixels

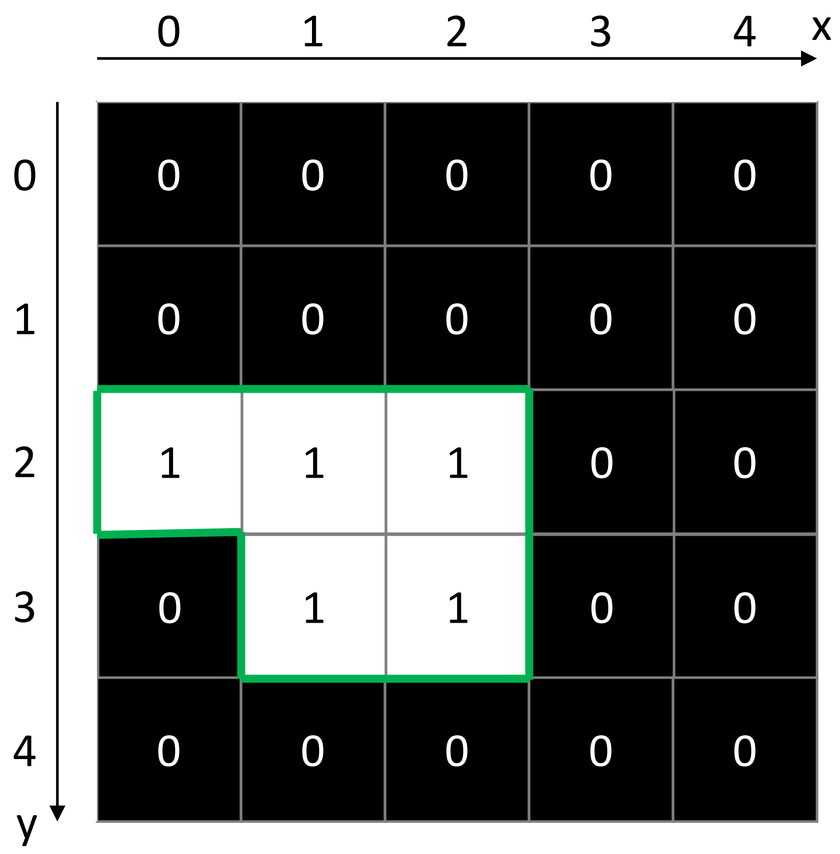


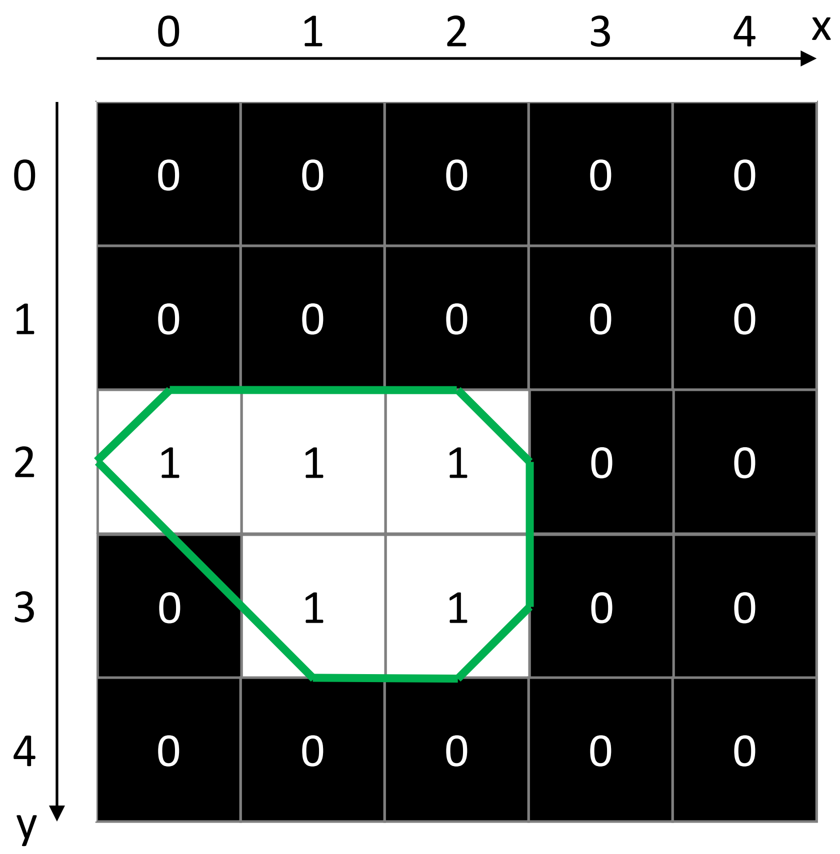
x_m = 1/5 (1·0 + 1·1 + 1·2 + 1·1 + 1·2) = 1.2

y_m = 1/5 (1·2 + 1·2 + 1·3 + 1·2 + 1·3) = 2.4

Perimeter

- Length of the outline around an object
- Depends on the actual implementation



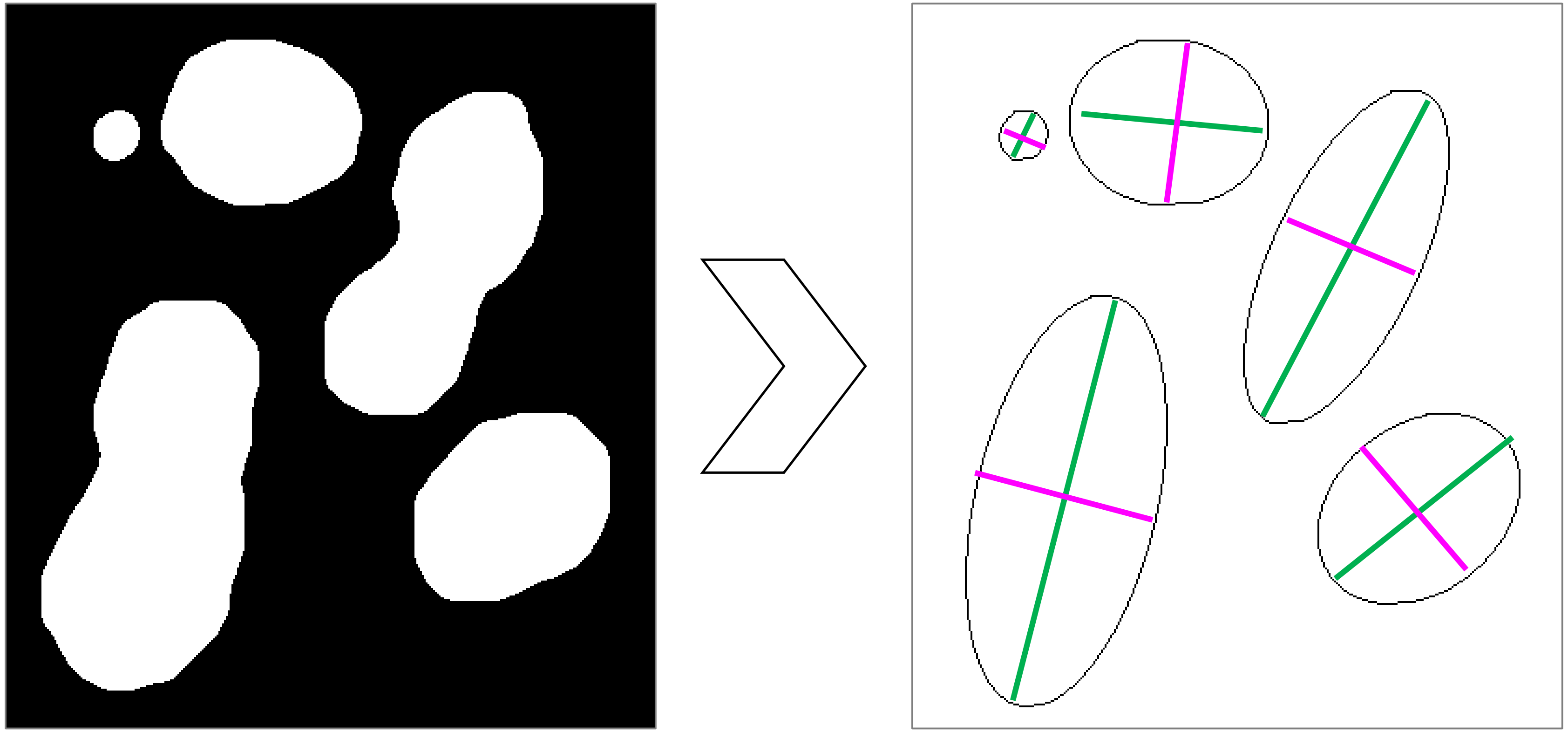


Fit ellipse

- For every object, find the optimal ellipse simplifying the object.

- Major axis … long diameter
- Minor axis … short diameter

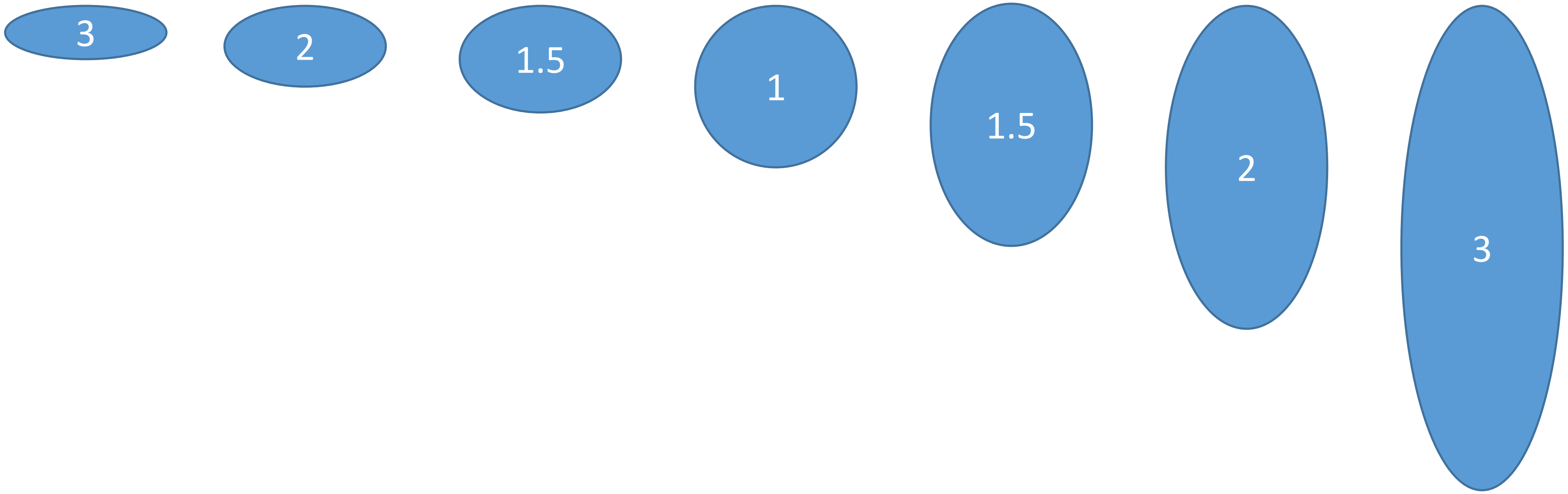
- Major and minor axis are perpendicular to each other



Aspect ratio

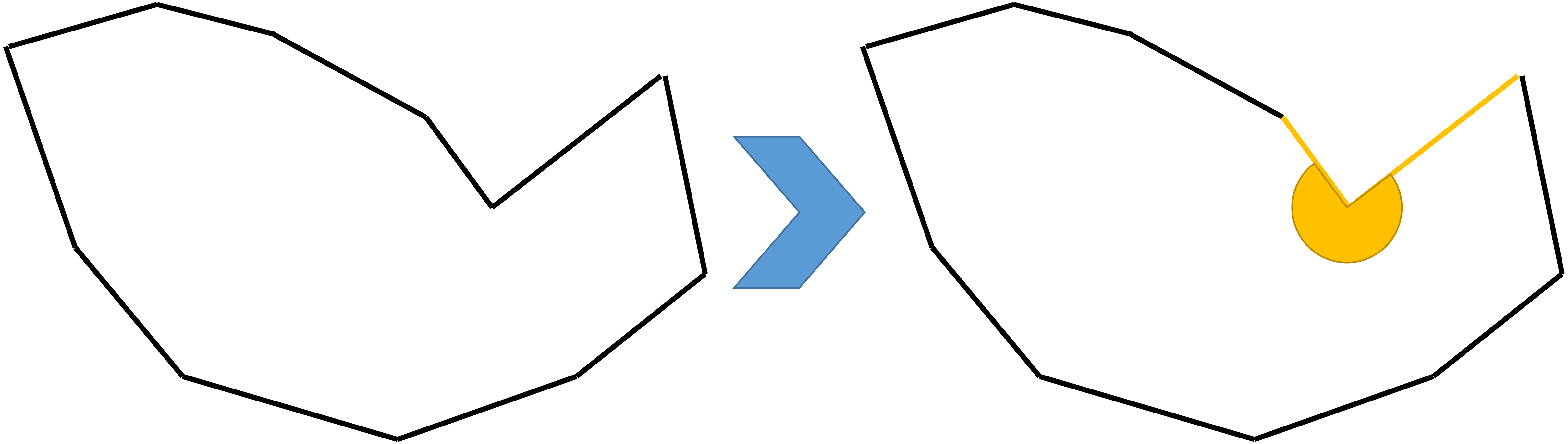
- The aspect ratio describes the elongation of an object.

AR = major / minor



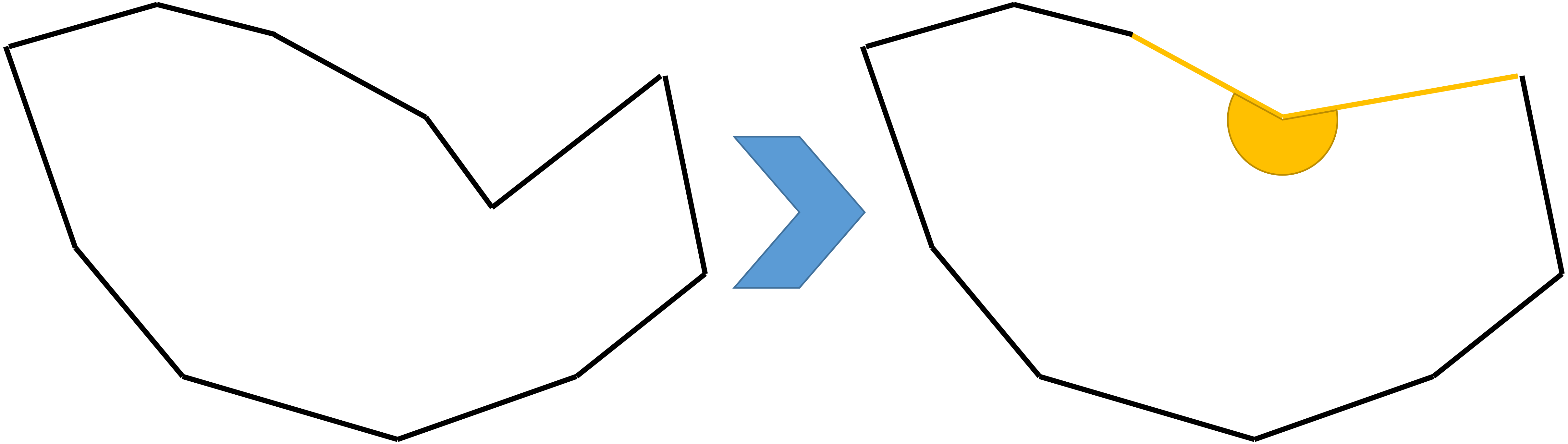
Convex hull

- By removing all concave corners of an object, we retrieve its convex hull.



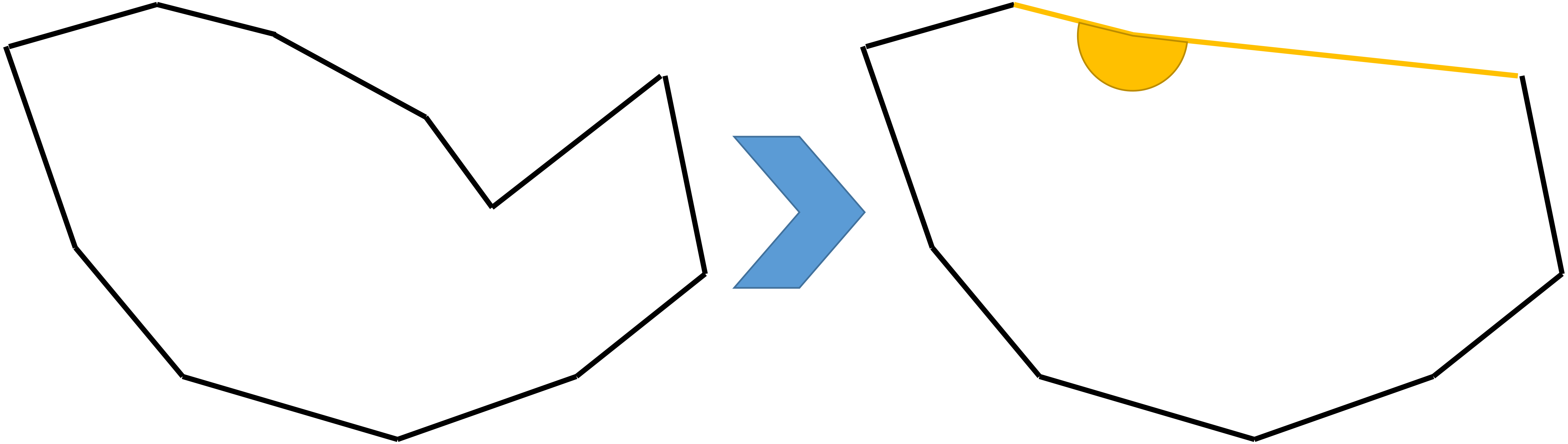
Convex hull

- By removing all concave corners of an object, we retrieve its convex hull.



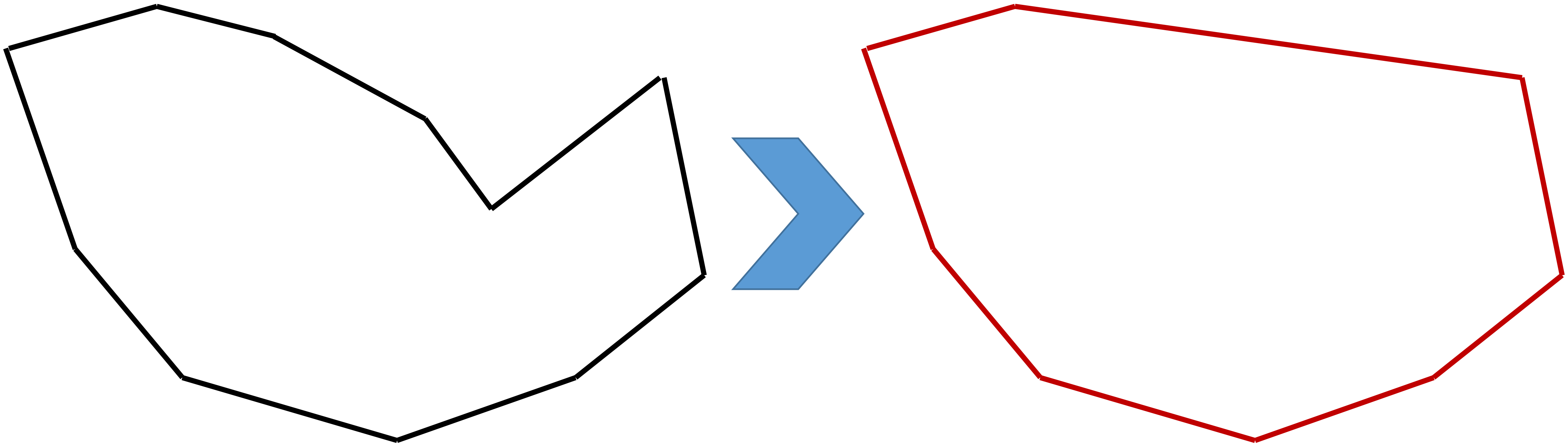
Convex hull

- By removing all concave corners of an object, we retrieve its convex hull.



Convex hull

- By removing all concave corners of an object, we retrieve its convex hull.

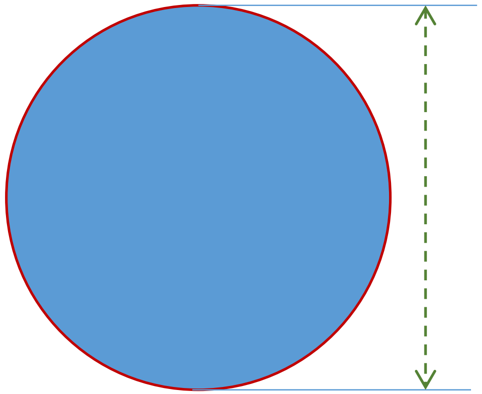


$$solidity = \frac{A}{A_{convexHull}}$$

Roundness and circularity

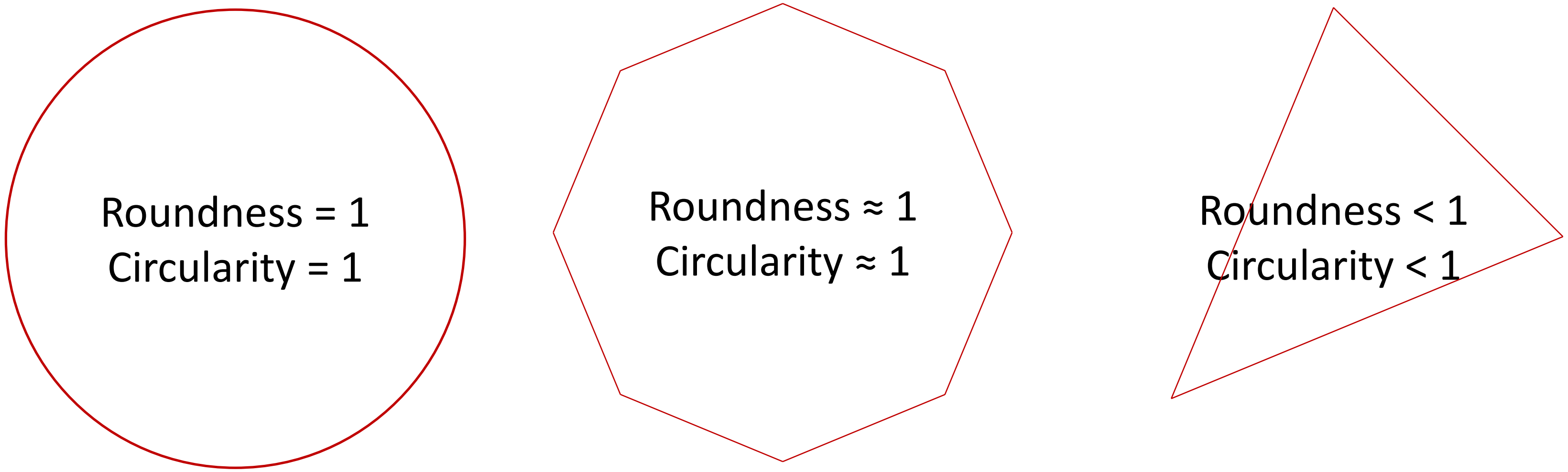
- The definition of a circle leads us to measurements of circularity and roundness.
- In case you use these measures, define them correctly. They are not standardized!

Diameter	d
Circumference	$C = \pi d$
Area	$A = \frac{\pi d^2}{4}$

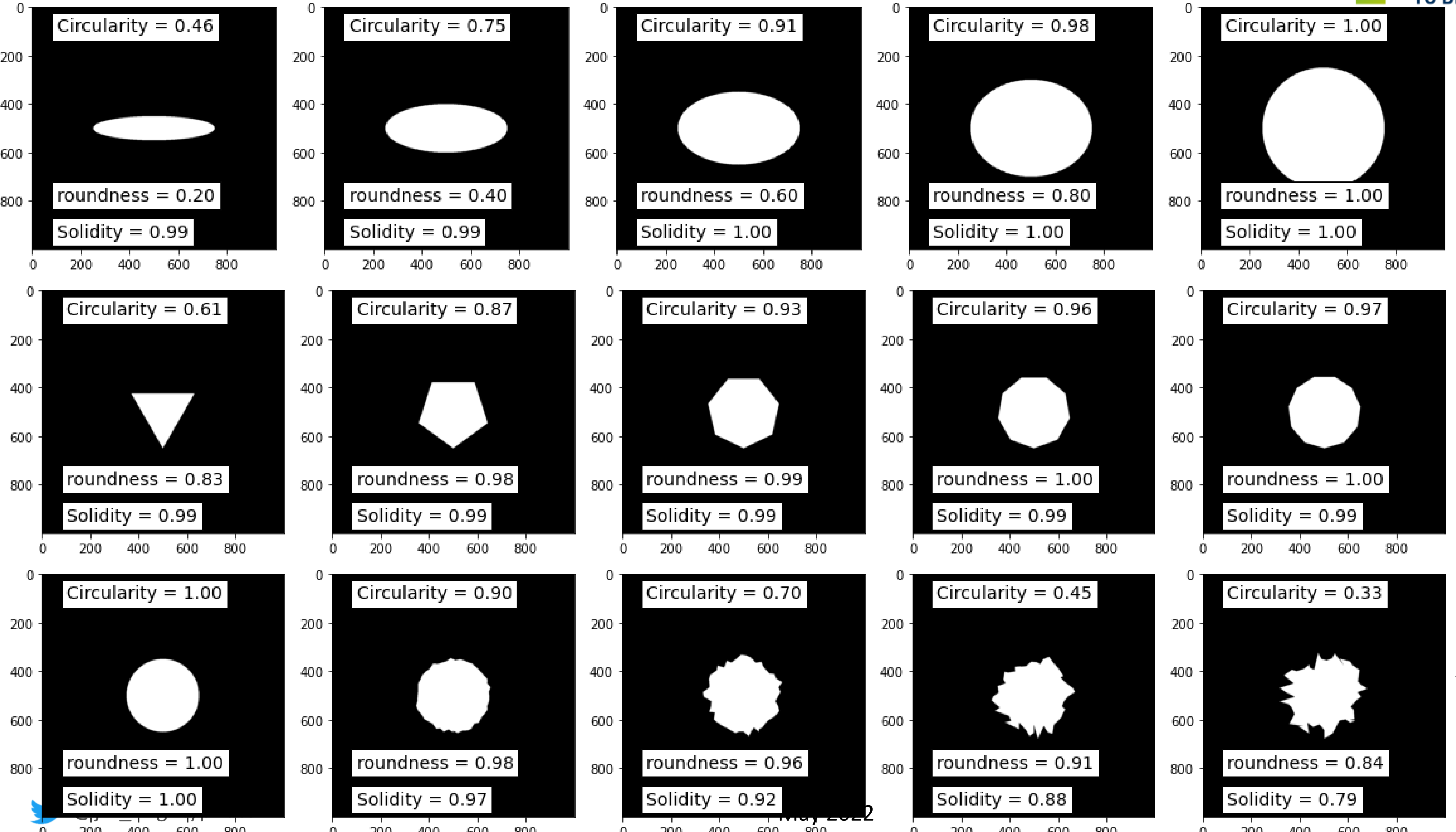


$$roundness = \frac{4 * A}{\pi\ major^2}$$

$$circularity = \frac{4\pi * A}{perimeter^2}$$



Roundness and circularity



$$roundness = \frac{4 * A}{\pi \, major^2}$$

$$circularity = \frac{4\pi * A}{perimeter^2}$$

$$solidity = \frac{A}{A_{convexHull}}$$

Feature extraction in Python

- **In Fiji**: *Analyze > Analyze Particles…*
- **In Python:**

```python
from skimage import measure
```

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

`skimage.measure.blur_effect` (image[, h_size, …])	Compute a metric that indicates the strength of blur in an image (0 for no blur, 1 for maximal blur).
`skimage.measure.euler_number` (image[, …])	Calculate the Euler characteristic in binary image.
`skimage.measure.find_contours` (image[, …])	Find iso-valued contours in a 2D array for a given level value.
`skimage.measure.grid_points_in_poly` (shape, verts)	Test whether points on a specified grid are inside a polygon.
`skimage.measure.inertia_tensor` (image[, mu])	Compute the inertia tensor of the input image.
`skimage.measure.inertia_tensor_eigvals` (image)	Compute the eigenvalues of the inertia tensor of the image.
`skimage.measure.label` (label_image[, …])	Label connected regions of an integer array.
`skimage.measure.regionprops` (label_image[, …])	Measure properties of labeled image regions.
`skimage.measure.regionprops_table` (label_image)	Compute image properties and return them as a pandas-compatible table.

area : int

Number of pixels of the region.

area_bbox : int

Number of pixels of bounding box.

area_convex : int

Number of pixels of convex hull image, which is the smallest convex polygon that e

area_filled : int

Number of pixels of the region will all the holes filled in. Describes the area of the i

axis_major_length : float

The length of the major axis of the ellipse that has the same normalized second ce the region.

axis_minor_length : float

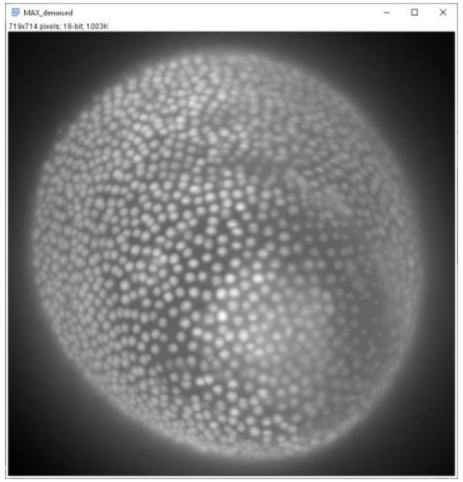
The length of the minor axis of the ellipse that has the same normalized second ce the region.

Feature extraction

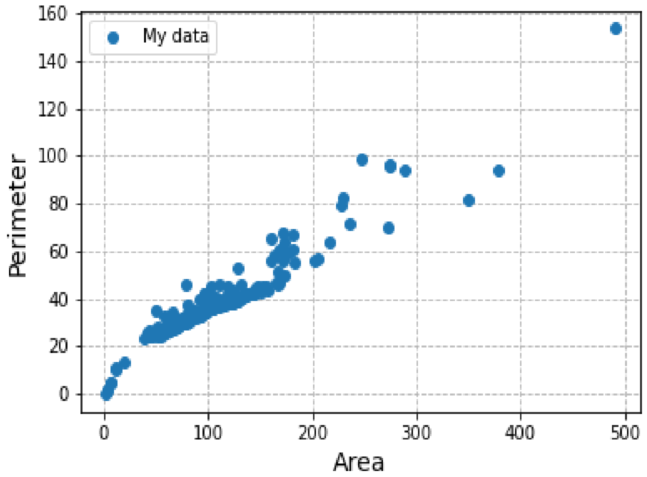
- Feature extraction is a *late* processing step in image analysis.
- It can be used for images, or segmented/labelled images

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

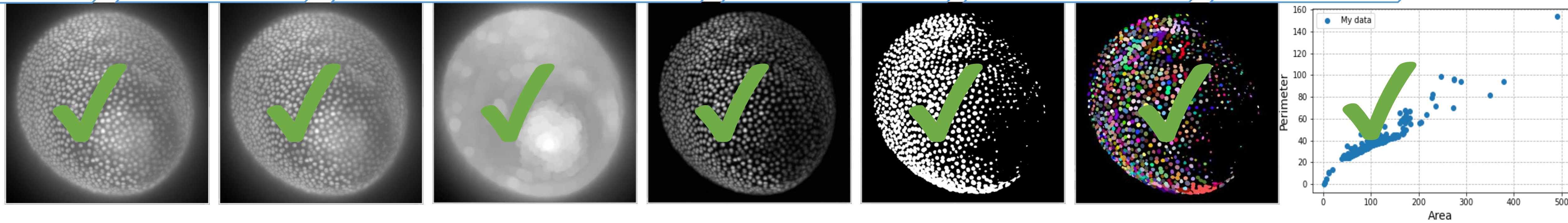
```
filtered = filters.gaussian(image, sigma=5)
```



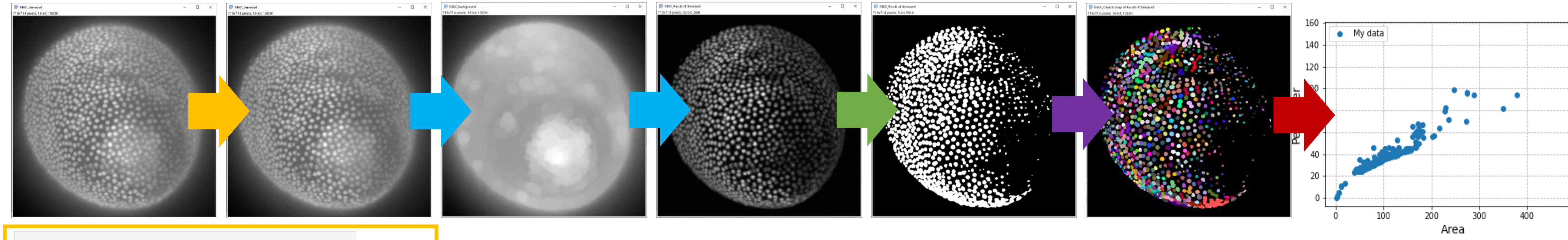
Feature Extraction



Acquisition → Denoising → Background subtraction → Segmentation → Labeling → Feature Extraction



Summary




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

```
filtered = filters.gaussian(image, sigma=5)
```

Filtering the image reduces pixel noise

```
bg_subtracted = morphology.white_tophat(image, footprint=footprint)
```

Top-hat filtering removes the background

Thresholding binarizes the image

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

Connected-components analysis groups pixels to objects

```
labels = measure.label(binary)
```

Feature extraction allows descriptive statistics

```
measurements = measure.regionprops_table(labels, properties=properties)
```

Next lecture

- **Field trip!**

https://gather.town/app/nq1oQrNJ1UIQ5t01/imagesc-island