Feature extraction

Till Korten

With material from

Robert Haase, Johannes Soltwedel and Marcelo Zoccoler, PoL, TU Dresden

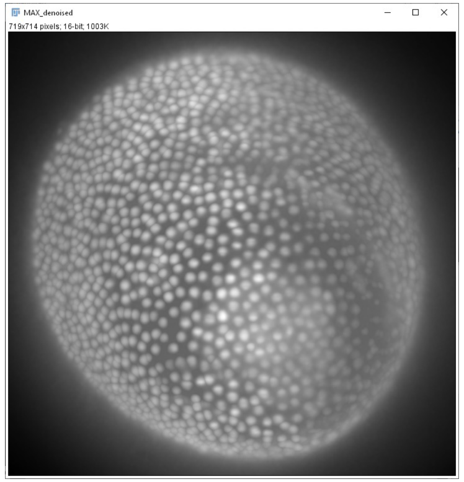
Benoit Lombardot, Scientific Computing Facility, MPI CBG

September 2023

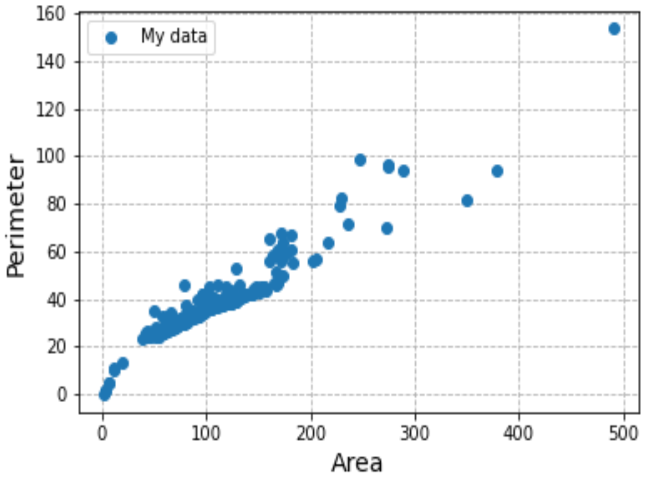
@TillKorten

Feature extraction

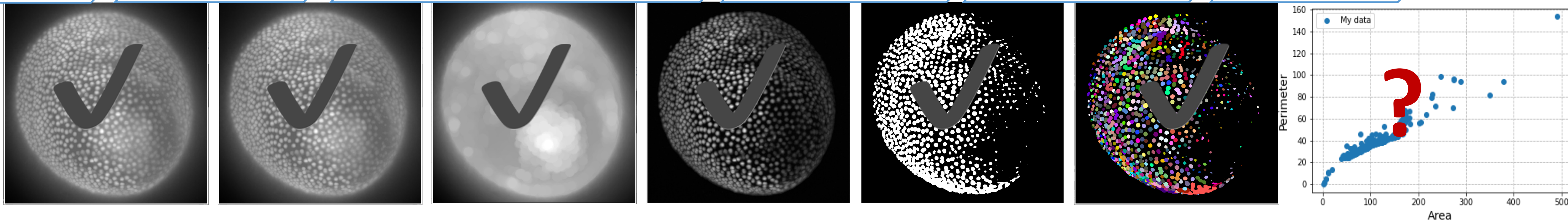
- Feature extraction is a *late* processing step in image analysis.
- It can be used for images, or segmented/labelled images



Feature Extraction



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



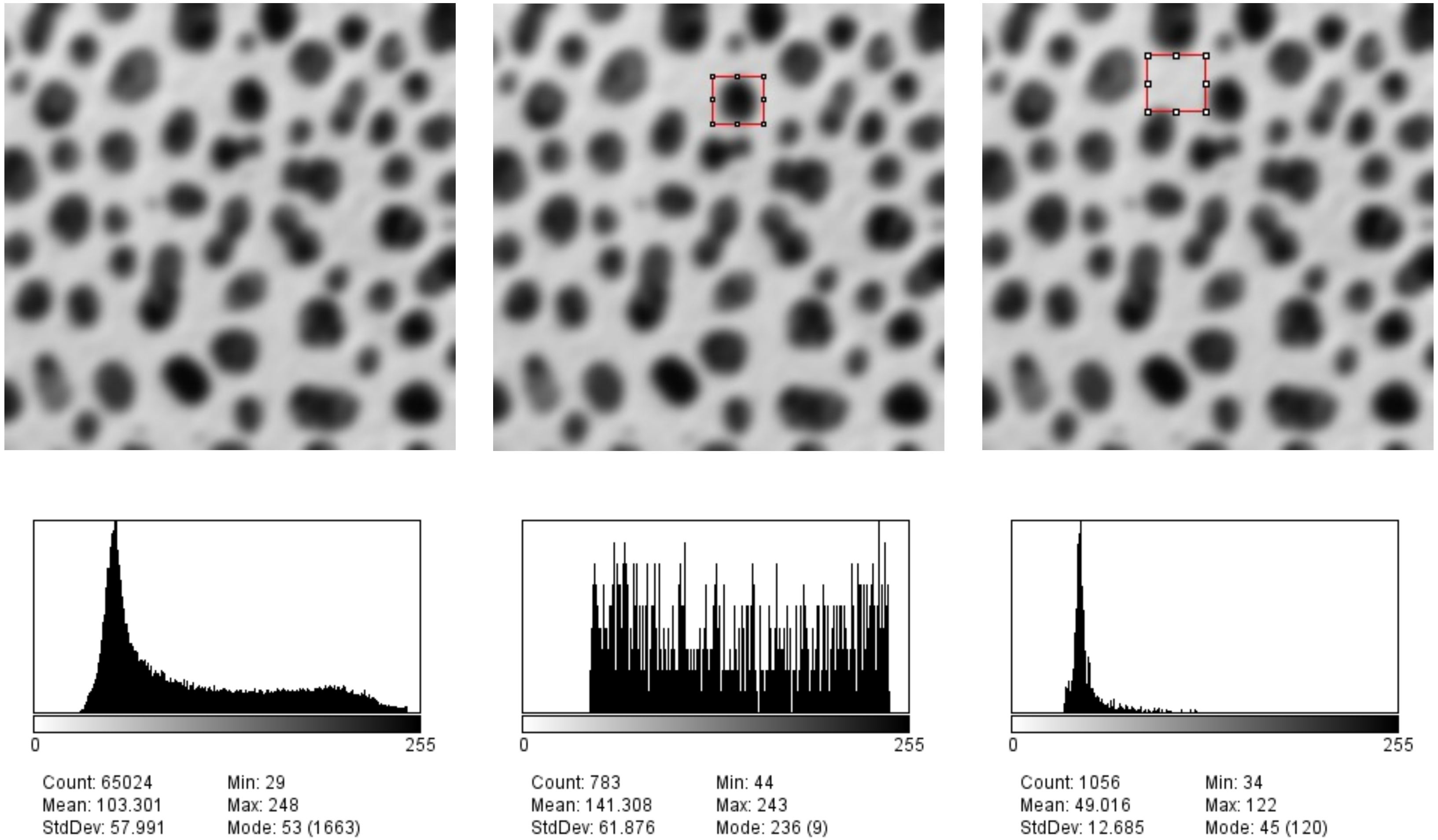
Feature extraction

- 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
- **Topological**
  - Number of neighbors
- **Others**
  - Overlap
  - Colocalization
- **Mixed features**
  - Center of mass
  - Local minima / maxima
  - Distance to neighbors
  - Average intensity in neighborhood

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

Center of mass

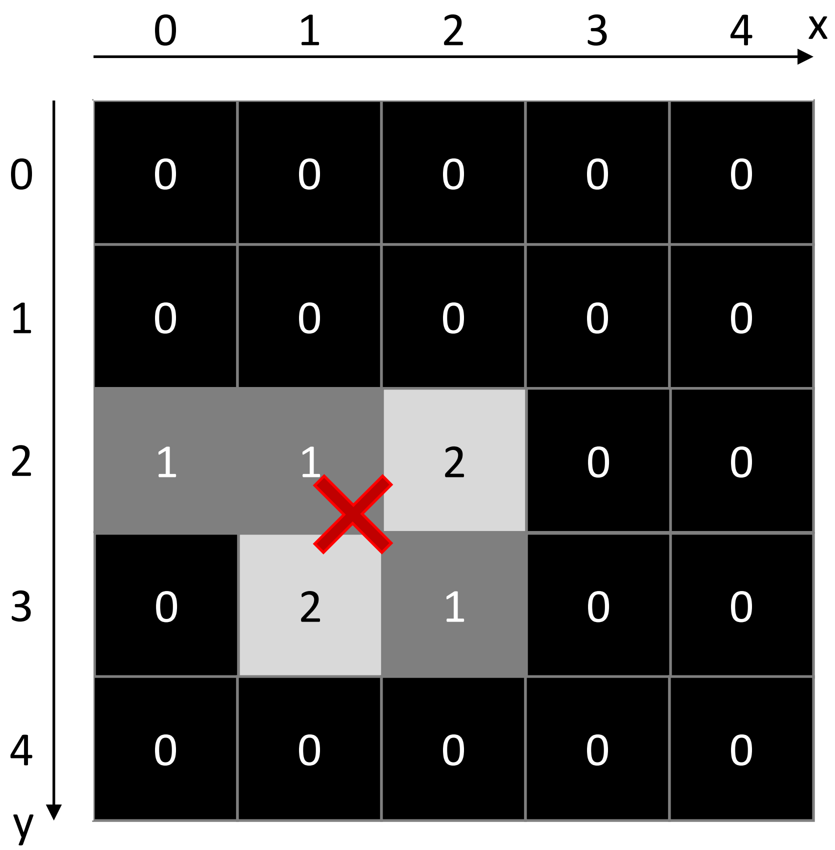
- Relative position in an image weighted by pixel intensities
  - x, y … pixel coordinates
  - w … image width
  - h … image height
  - µ … mean intensity
  - $g_{x,y}$ … pixel grey value
  - x_m, y_m … center of mass coordinates

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

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

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

"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

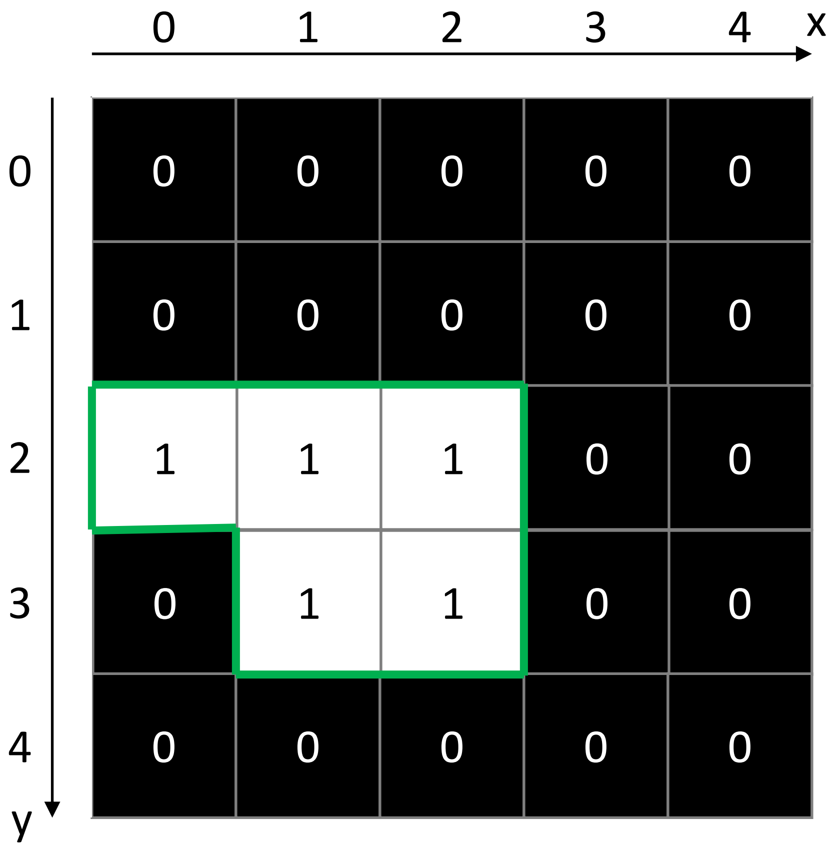
y \ x	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0	0	0
2	1	1	1	0	0
3	0	1	1	0	0
4	0	0	0	0	0

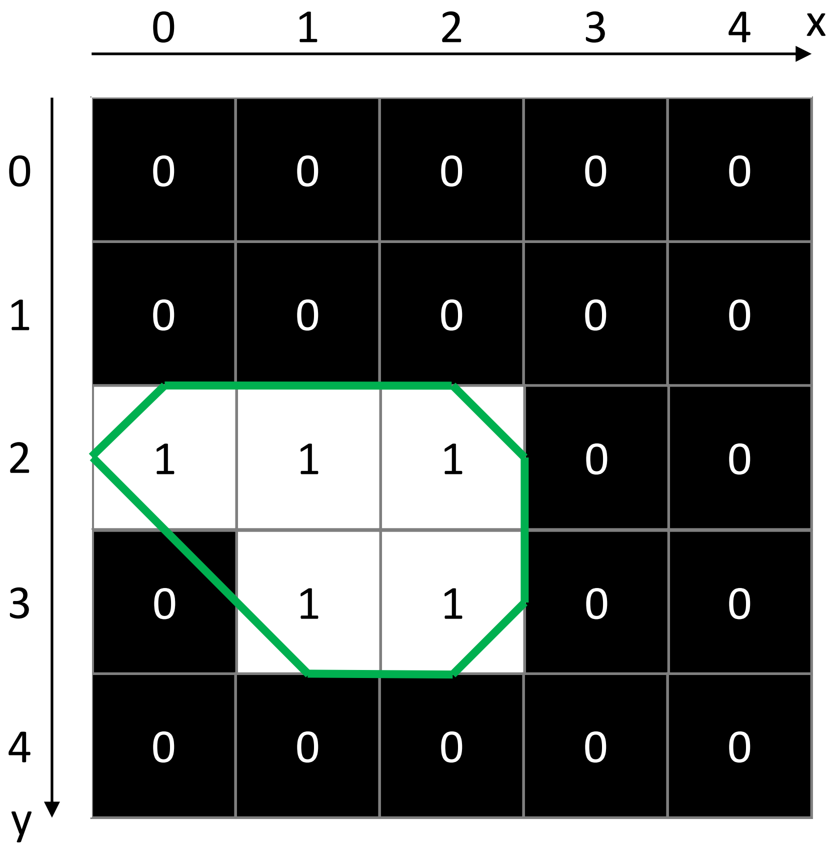
x_m = 1/5 (1·0 + 1·1 + 1·2 + 1·1 + 1·2) = 1.2

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

Perimeter

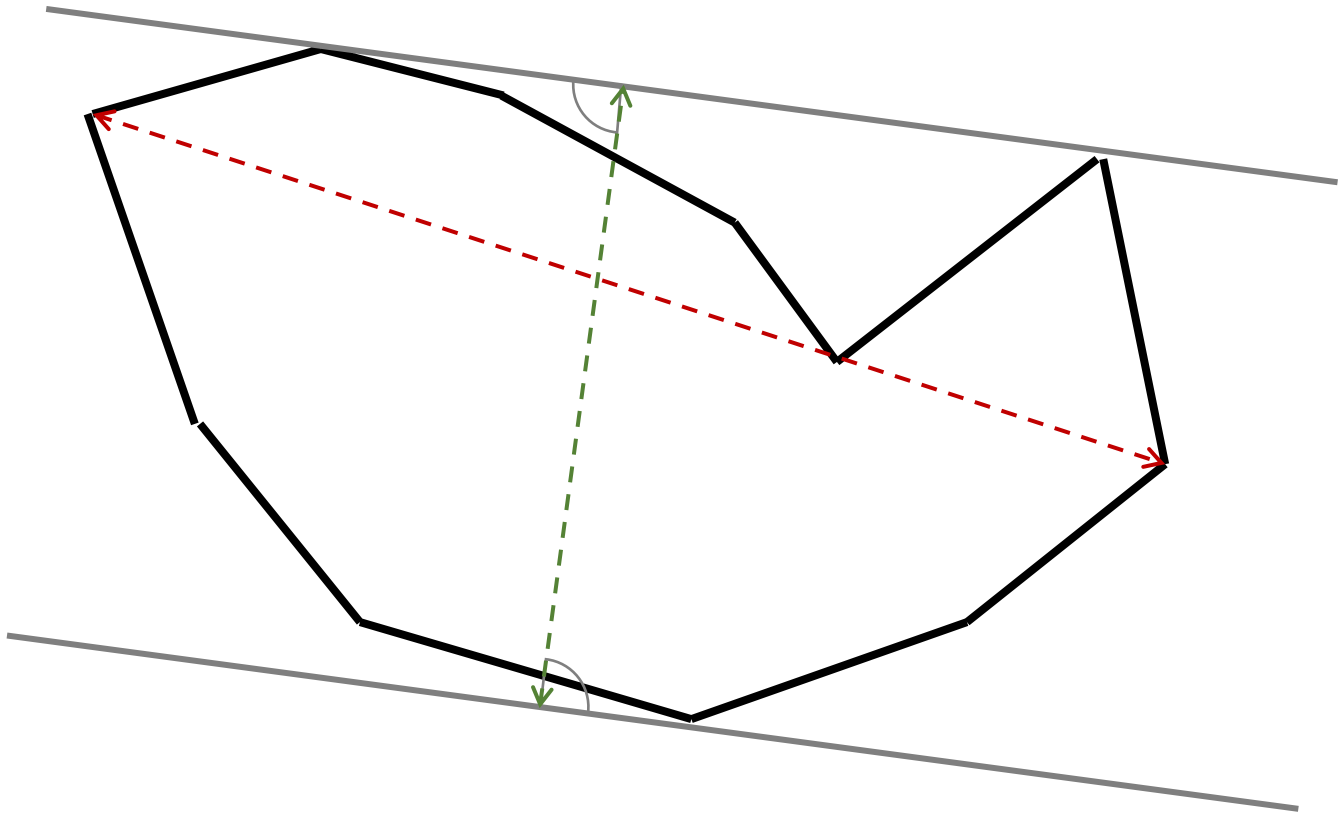
- Length of the outline around an object
- Depends on the actual implementation





Feret’s diameter

- Feret’s diameter describes the maximum distance between any two points of an outline.
- The minimum caliper (“Minimum Feret”) describes the shortest distance, the object would fit through.
- Feret and Minimum Feret do not need to be perpendicular to each other!

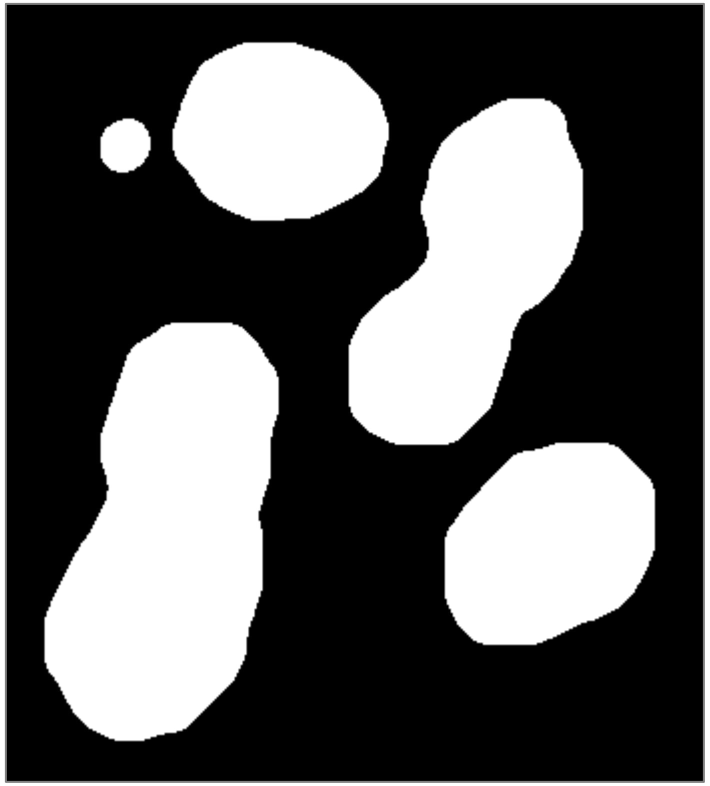


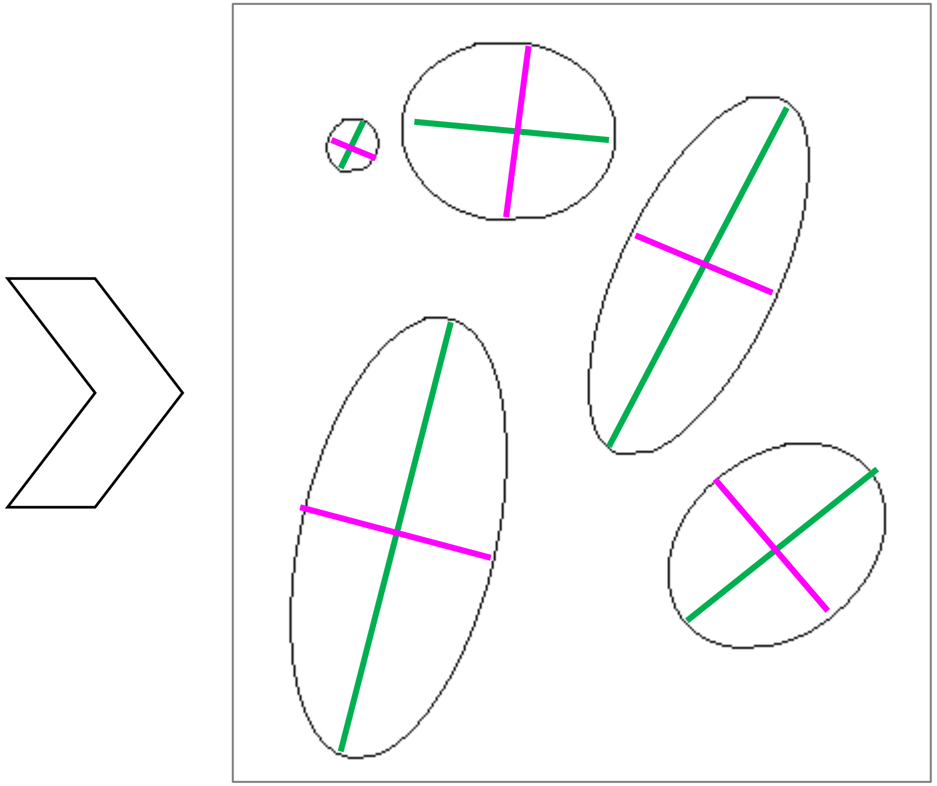
Fit ellipse

- 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

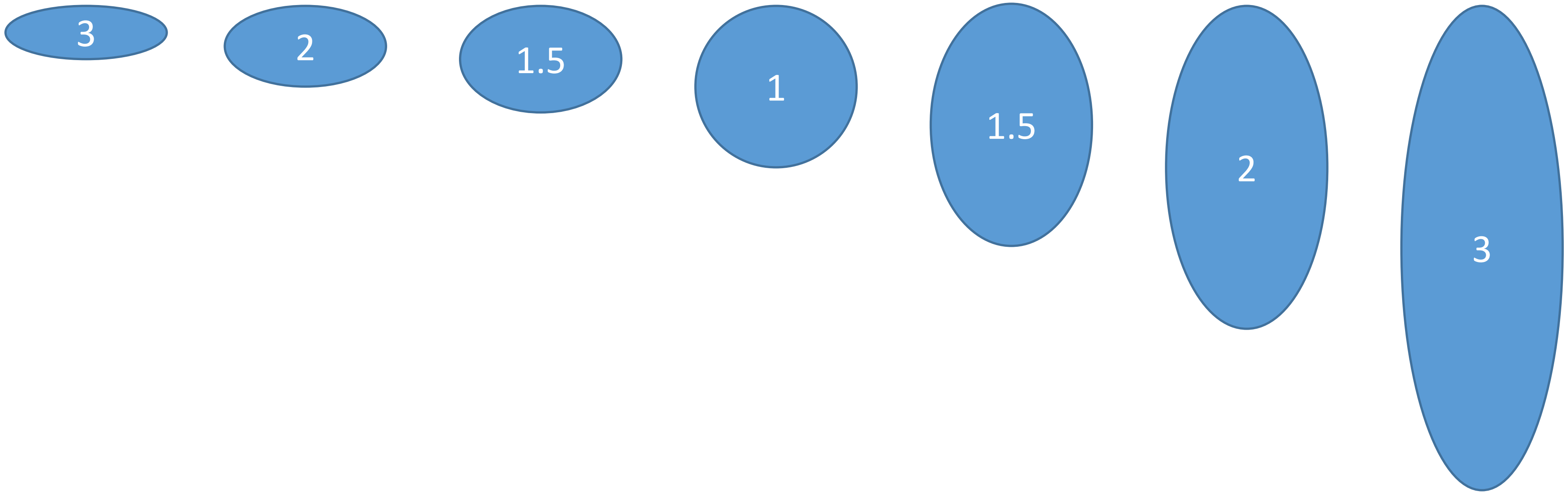




Aspect ratio

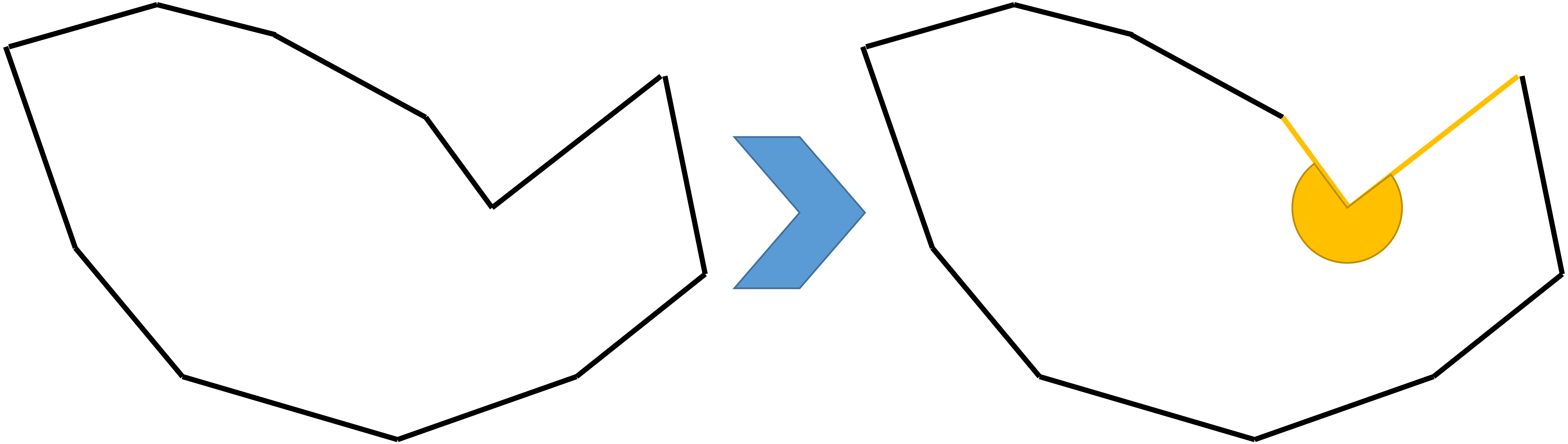
- The aspect ratio describes the elongation of an object.

AR = major / minor



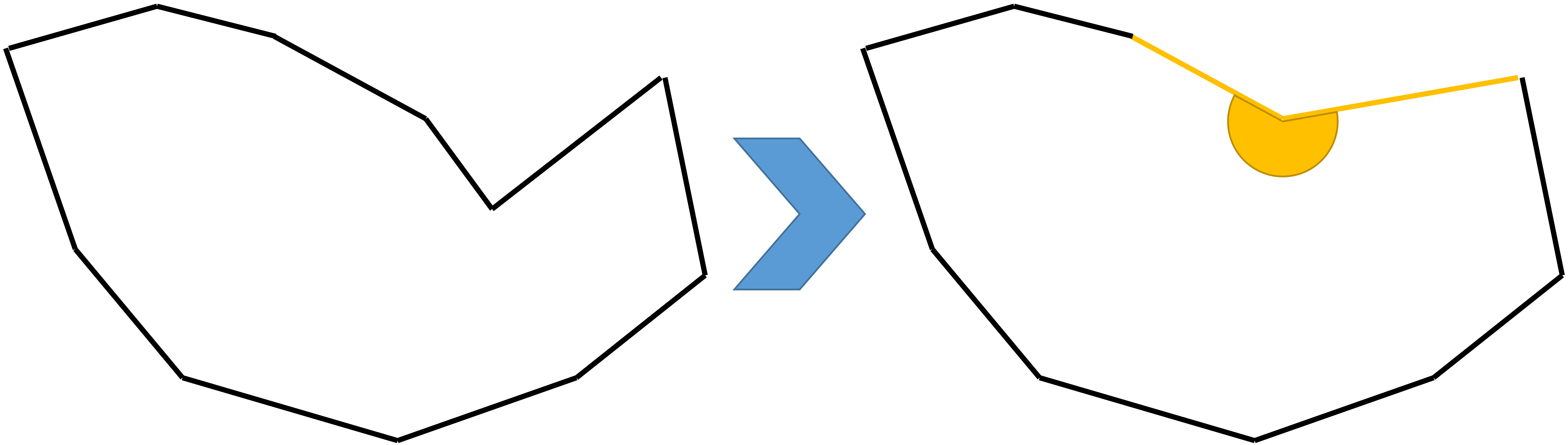
Convex hull

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



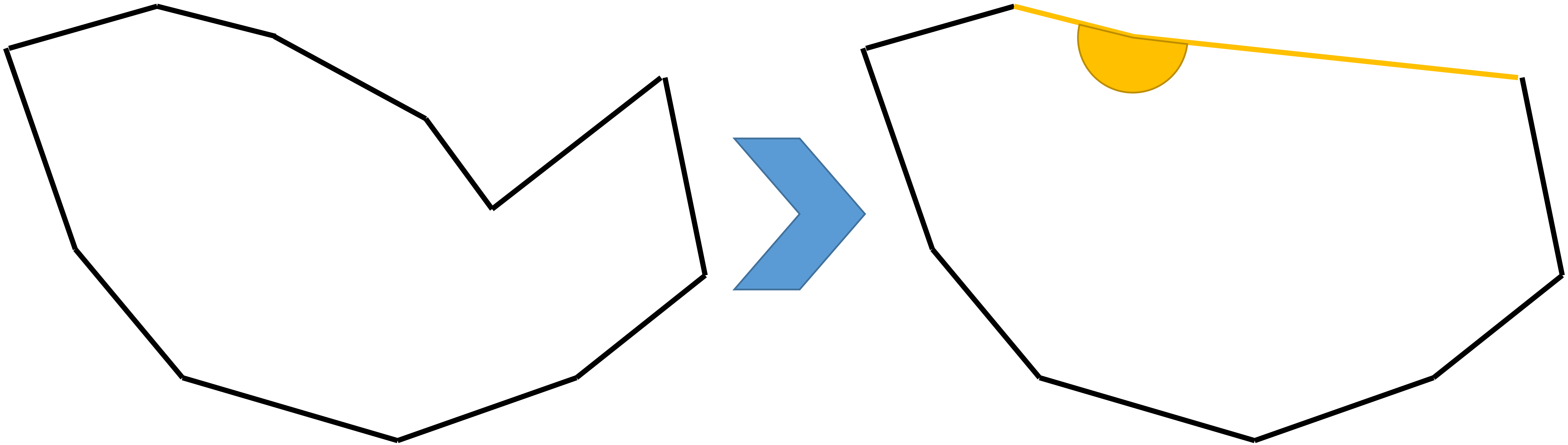
Convex hull

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



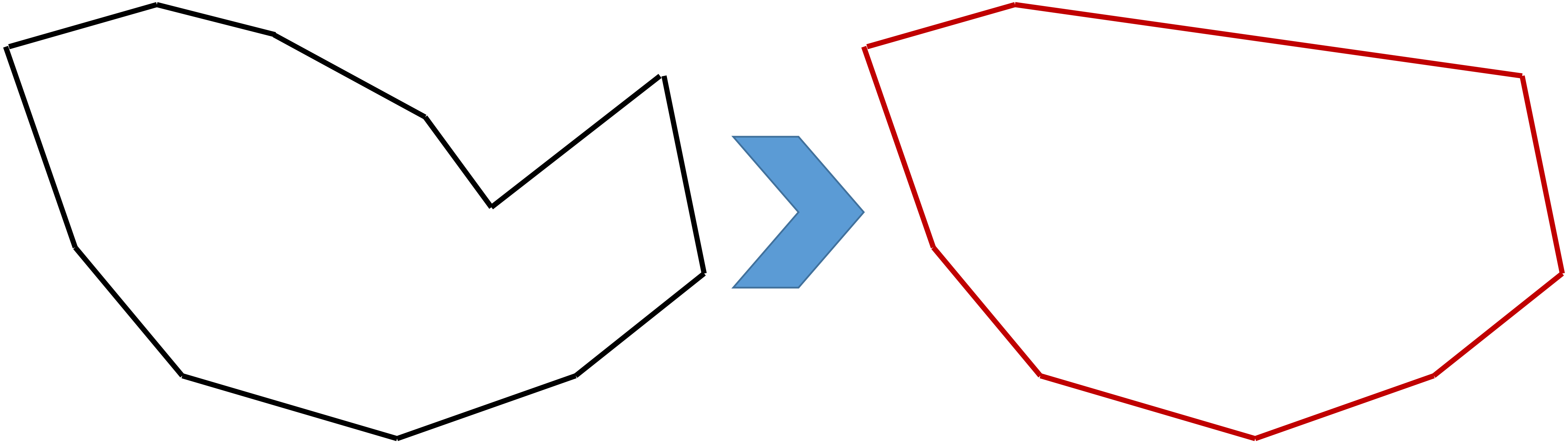
Convex hull

- 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.

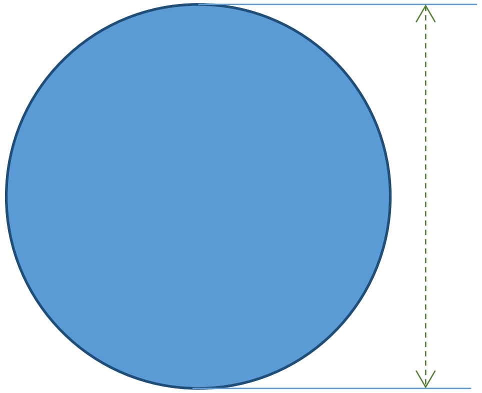


$$solidity = \frac{A}{A_{convexHull}}$$

Roundness and circularity

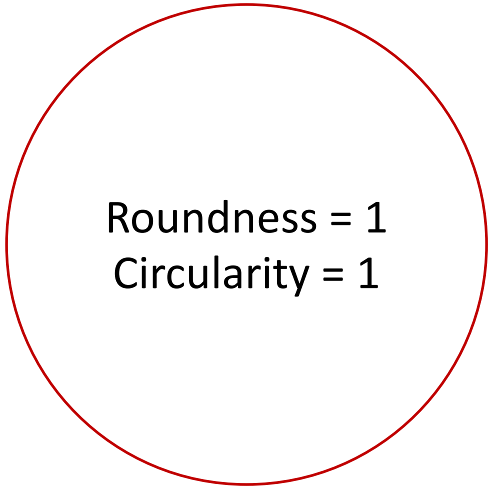
- The definition of a circle leads us to measurements of circularity and roundness.
- In case you use these measures, define them correctly. They are not standardized!

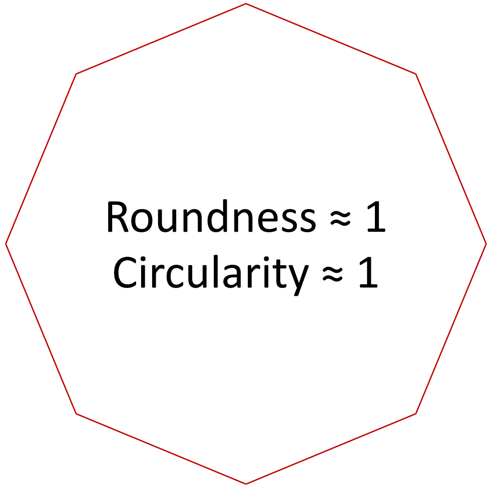
Diameter	d
Circumference	$C = \pi d$
Area	$A = \dfrac{\pi d^2}{4}$

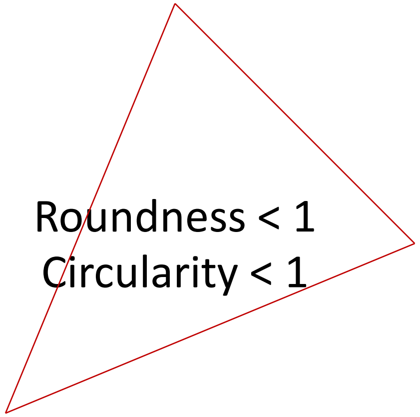


$$roundness = \frac{4 \ * A}{\pi \ major^2}$$

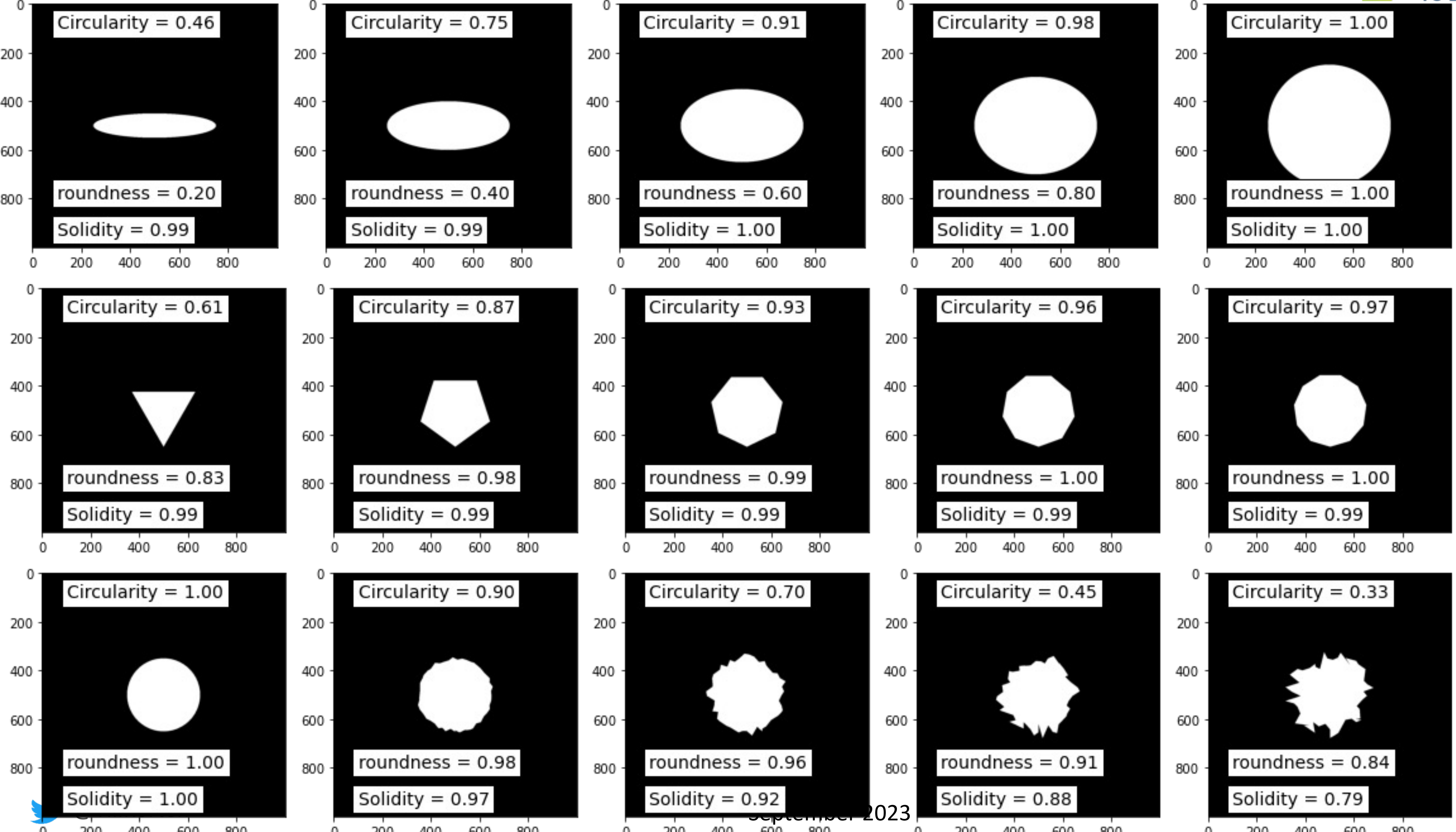
$$circularity = \frac{4\pi \ * A}{perimeter^2}$$







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 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.

Exercise: Quantiative measurements

- Use the given feature extraction notebook to apply some basic statistics to measurements

```
# analyse objects
properties = measure.regionprops(label_image, intensity_image=image)
```

```
statistics = {
    'area':       [p.area                for p in properties],
    'mean':       [p.mean_intensity      for p in properties],
    'major_axis': [p.major_axis_length   for p in properties]
}
```

```
dataframe = pd.DataFrame(statistics)
dataframe
```

	area	mean	major_axis	aspect_ratio
0	429	191.440559	34.779230	2.088249
1	183	179.846995	20.950530	1.782168
2	658	205.604863	30.198484	1.067734
3	433	217.515012	24.508791	1.061942
4	472	213.033898	31.084766	1.579415
...	...	...	...	...
57	213	184.525822	18.753879	1.296143
58	79	184.810127	18.287489	3.173540
59	88	182.727273	21.673692	4.021193
60	52	189.538462	14.335104	2.839825
61	48	173.833333	16.925660	4.417297

62 rows × 4 columns

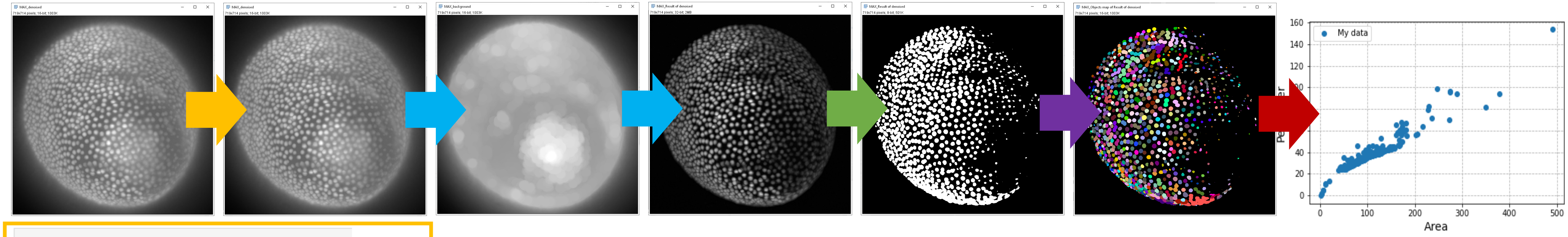
- How many objects are in it?

- How large is the largest object?

- What are mean and standard deviation of the intensity in the image?

- What are mean and standard deviation of the area of the segmented objects?

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)
```