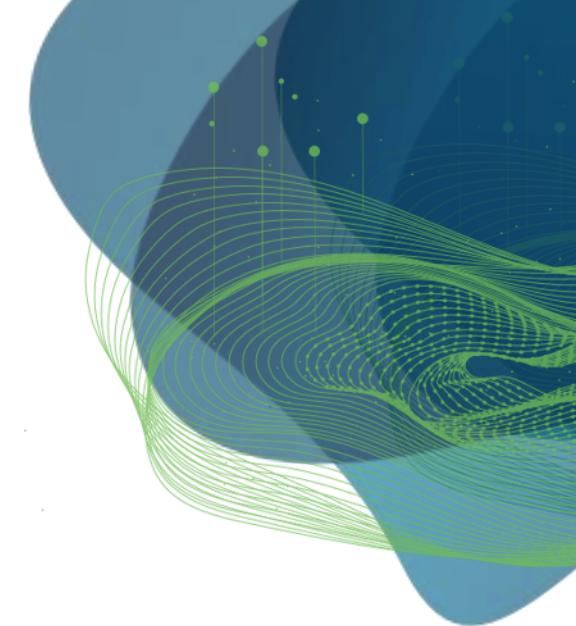




DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS  
AND ARTIFICIAL INTELLIGENCE



# Surface reconstruction

Robert Haase

Using materials from Alba Villaronga Luque and Jesse Veenvliet (MPI CBG Dresden), Marcelo Leomil Zoccoler, Johannes Soltwedel and Mara Lampert, PoL, TU Dresden

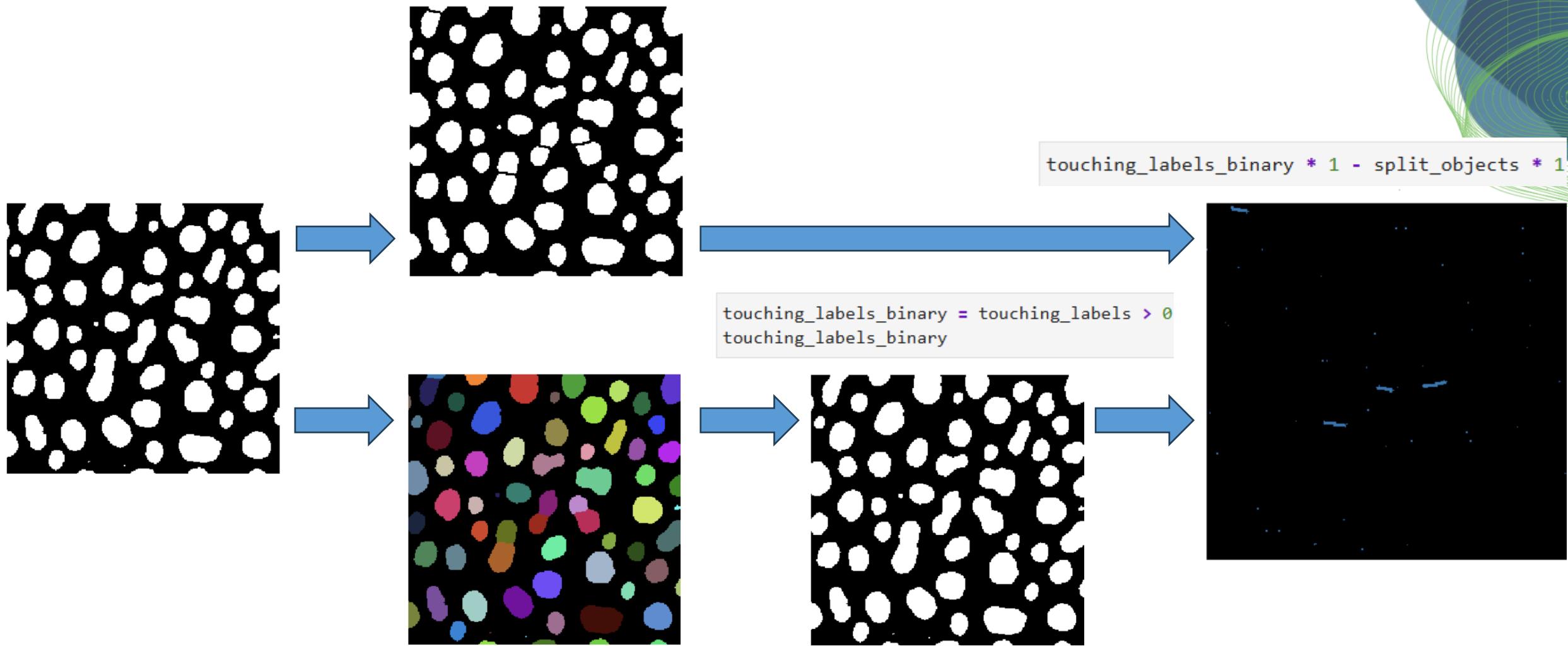
GEFÖRDERT VOM



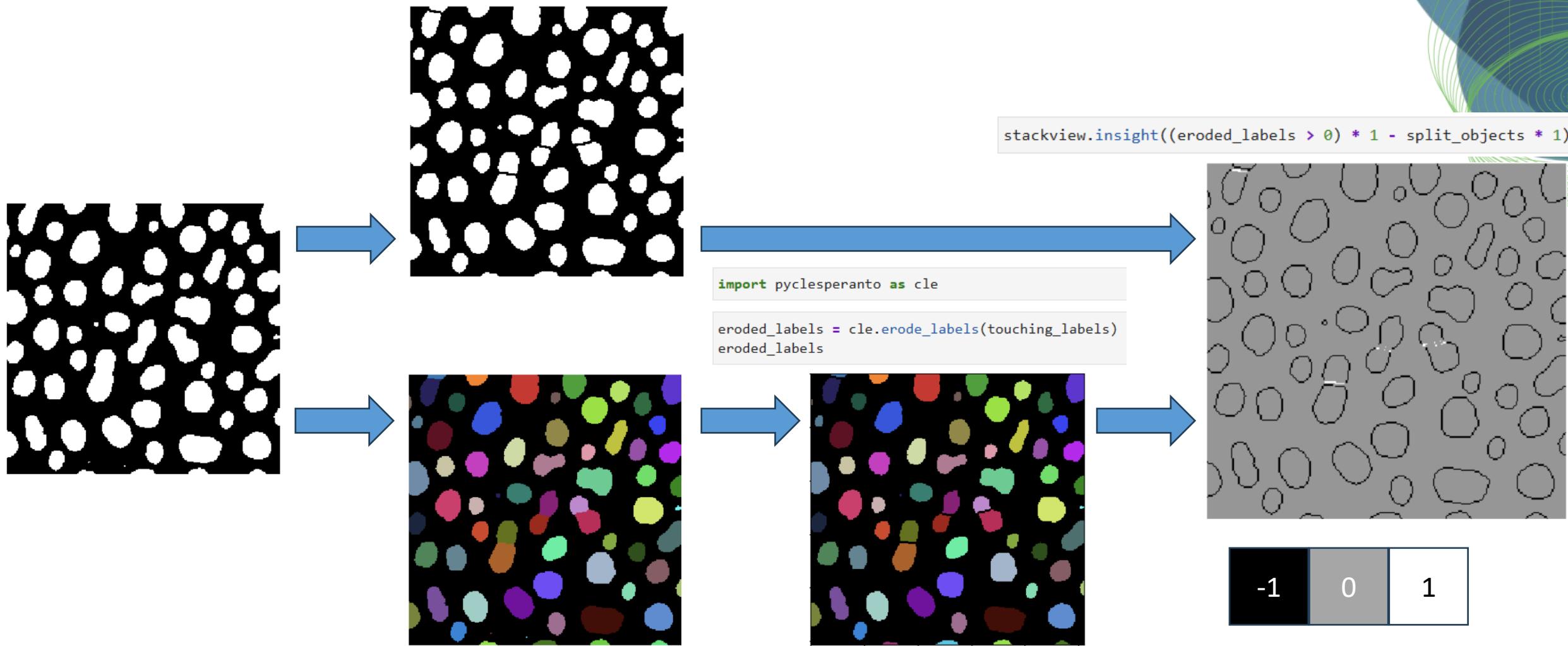
Bundesministerium  
für Bildung  
und Forschung

Diese Maßnahme wird gefördert durch die Bundesregierung aufgrund eines Beschlusses des Deutschen Bundestages.  
Diese Maßnahme wird mitfinanziert durch Steuermittel auf der Grundlage des von den Abgeordneten des Sächsischen Landtags beschlossenen Haushaltes.

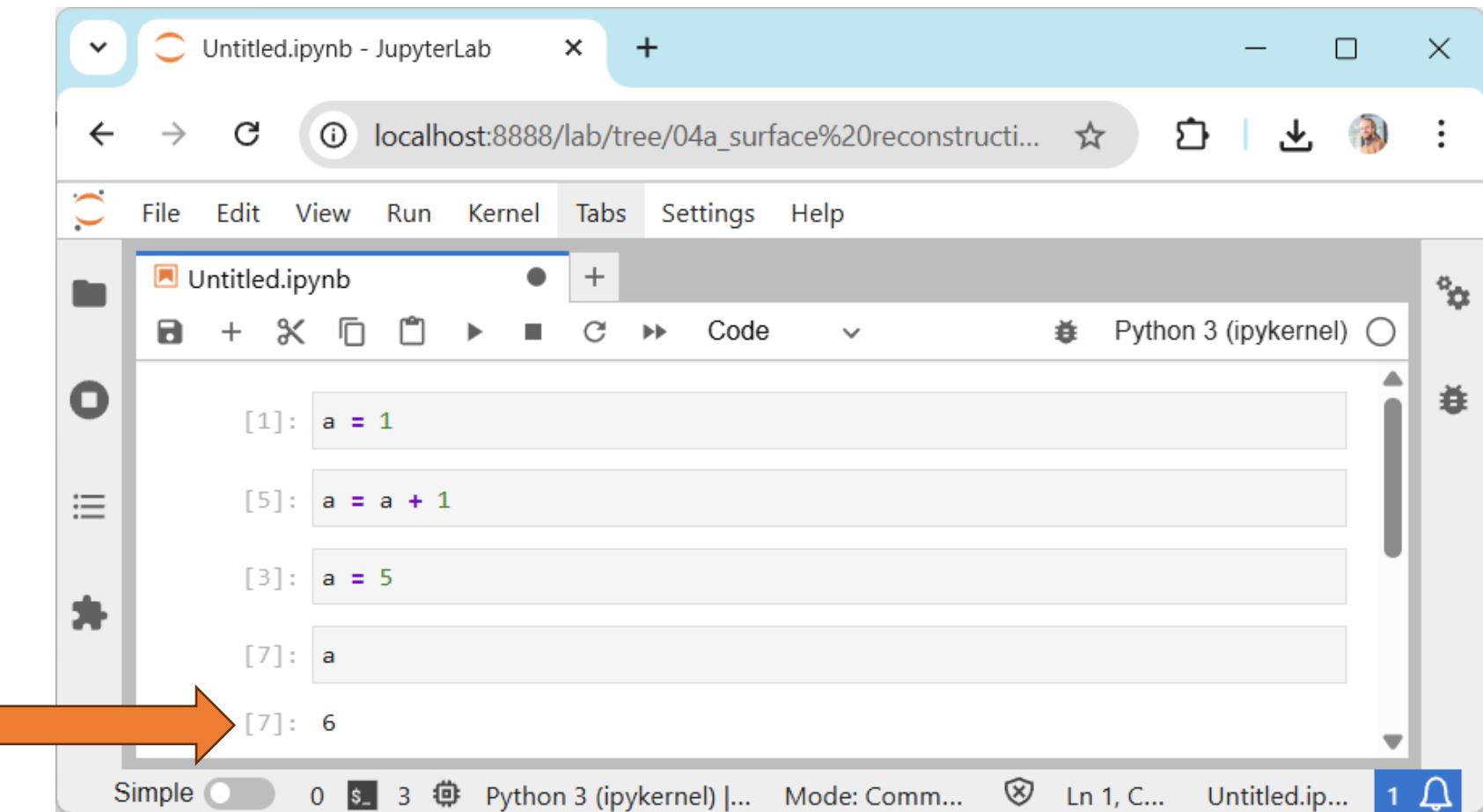
# Follow up: Watershed exercise



# Follow up: Watershed exercise



# Quiz: What is wrong with this result?



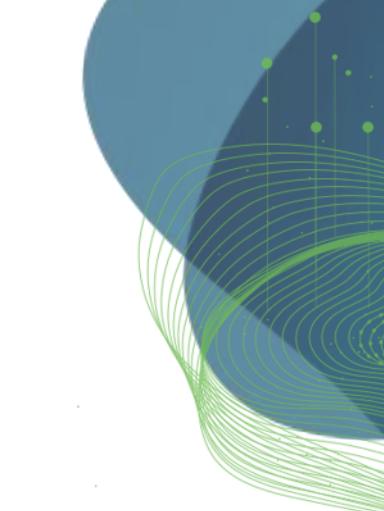
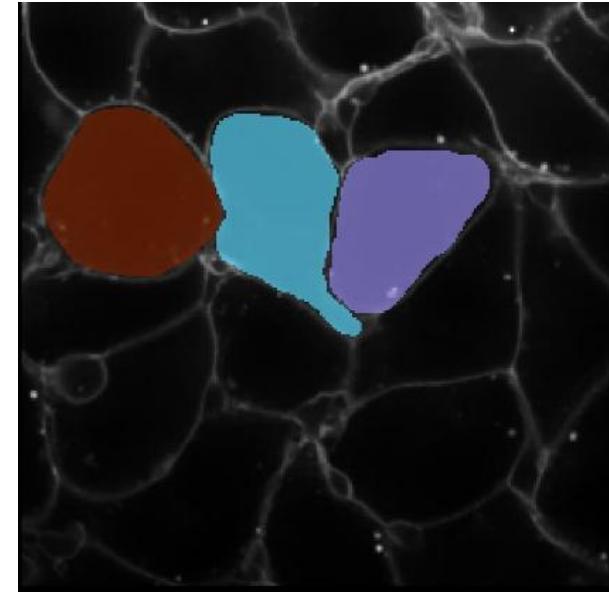
The screenshot shows a JupyterLab interface with the title bar "Untitled.ipynb - JupyterLab". The URL in the address bar is "localhost:8888/lab/tree/04a\_surface%20reconstructi...". The menu bar includes File, Edit, View, Run, Kernel, Tabs, Settings, and Help. A sidebar on the left shows a file tree with "Untitled.ipynb" selected. The main area displays a code cell history:

```
[1]: a = 1
[5]: a = a + 1
[3]: a = 5
[7]: a
[7]: 6
```

An orange arrow points to the output cell [7]: 6, indicating the error in the result.

# Sparse Jaccard Index

This label image  
shows a ...



Instance  
segmentation



Semantic  
segmentation



Sparse  
semantic  
segmentation

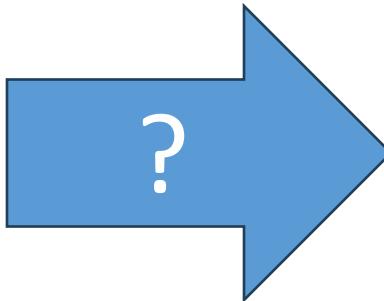


Sparse  
instance  
segmentation



# Quiz: Recap

- How is this operation called?



Dilation



Erosion



Opening

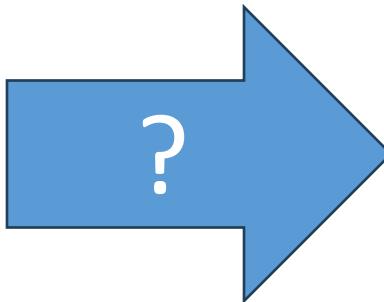


Closing



# Quiz: Recap

- How is this operation called?



Dilation



Erosion



Opening



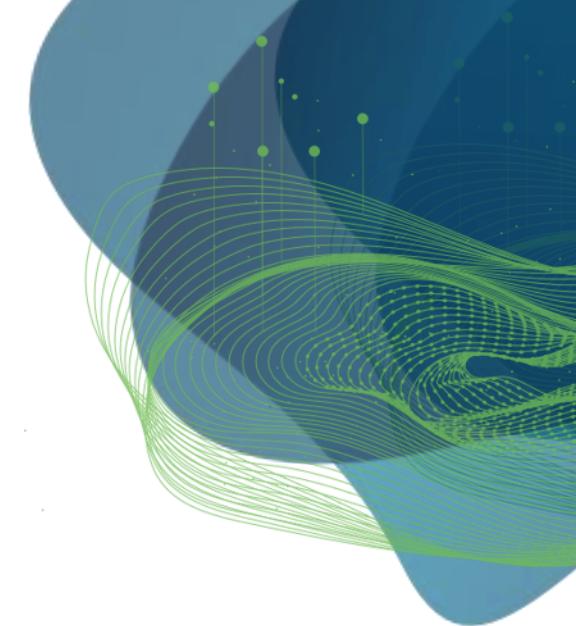
Closing





DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS  
AND ARTIFICIAL INTELLIGENCE



# Surface reconstruction

Robert Haase

Using materials from Alba Villaronga Luque and Jesse Veenvliet (MPI CBG Dresden), Marcelo Leomil Zoccoler, Johannes Soltwedel and Mara Lampert, PoL, TU Dresden

GEFÖRDERT VOM

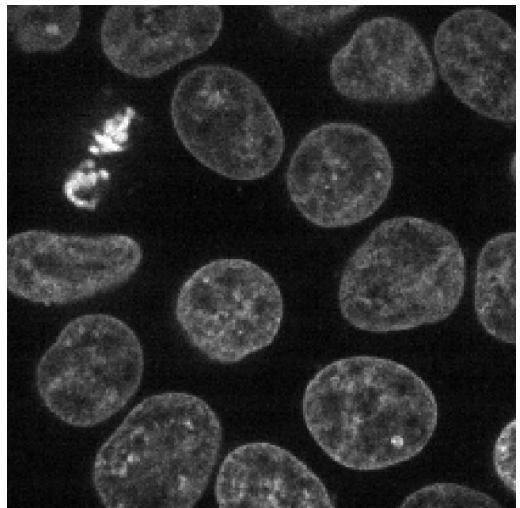


Bundesministerium  
für Bildung  
und Forschung

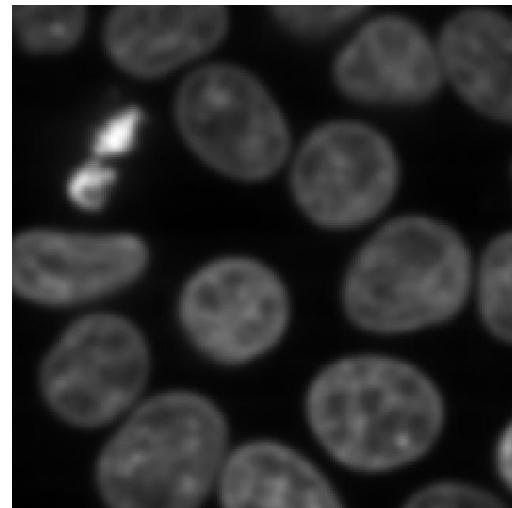
Diese Maßnahme wird gefördert durch die Bundesregierung aufgrund eines Beschlusses des Deutschen Bundestages.  
Diese Maßnahme wird mitfinanziert durch Steuermittel auf der Grundlage des von den Abgeordneten des Sächsischen Landtags beschlossenen Haushaltes.

# Motivation: Surface reconstruction

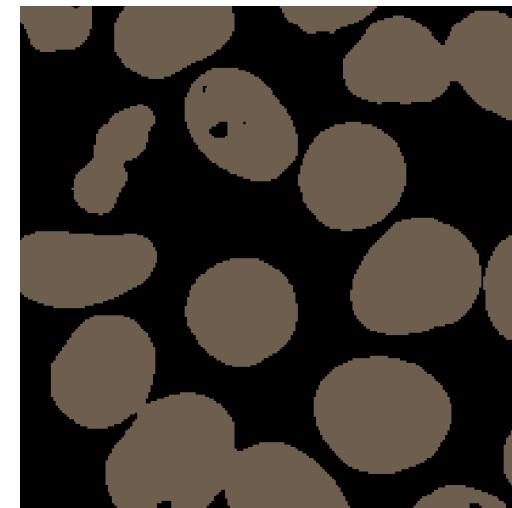
- Pixel and voxel arrays can be huge in memory
- Processing 3D arrays is time-consuming



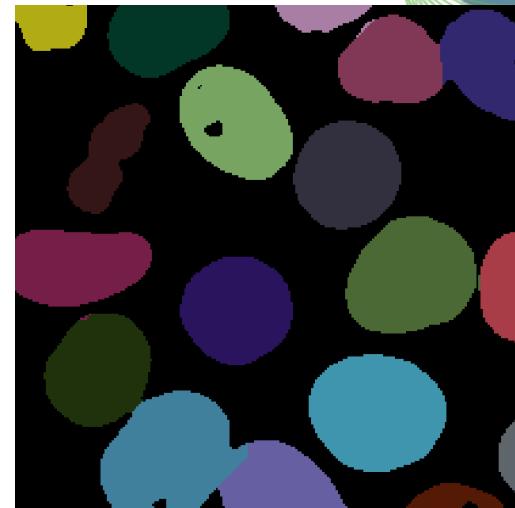
1024 x1024 x 100  
16-bit image



1024 x1024 x 100  
16-bit image



1024 x1024 x 100  
8-bit image



1024 x1024 x 100  
16-bit image

How much memory does  
this workflow cost?

700 MB

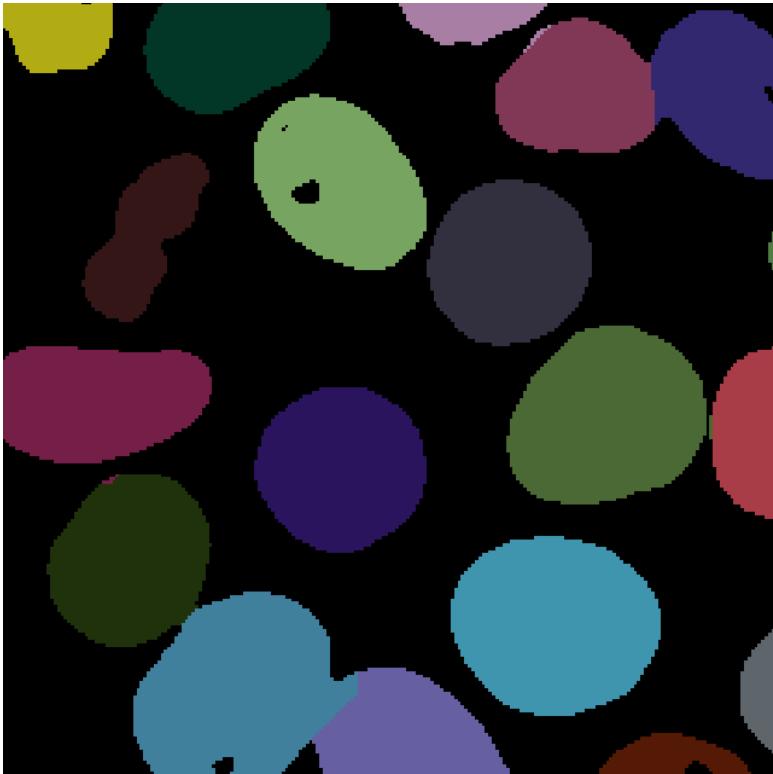
400 MB

4 GB

7 GB

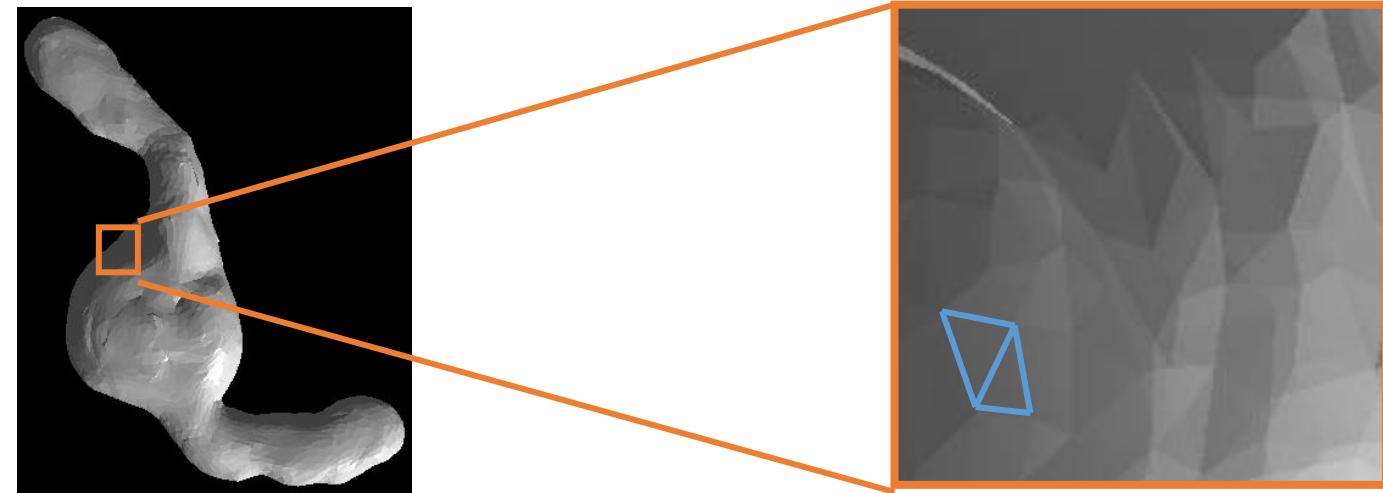
# Motivation: Surface reconstruction

- Pixel and voxel borders introduce artifacts, potentially problematic for measurements, e.g. surface area



# Surface meshes

- Points on a surfaces connected by triangles form a surface mesh



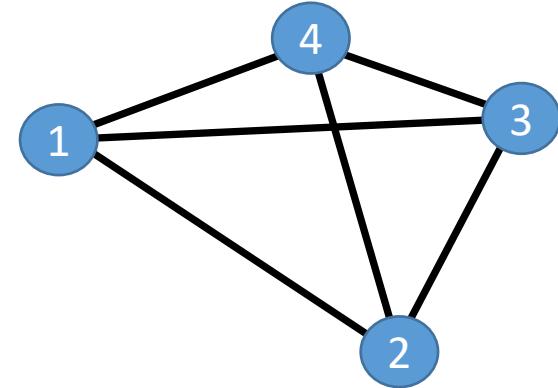
“Vertices” / points

Point x	Point y	Point z
$x_1$	$y_1$	$z_1$
$x_2$	$y_2$	$z_2$
$x_3$	$y_3$	$z_3$
$x_4$	$y_4$	$z_4$
...	...	...

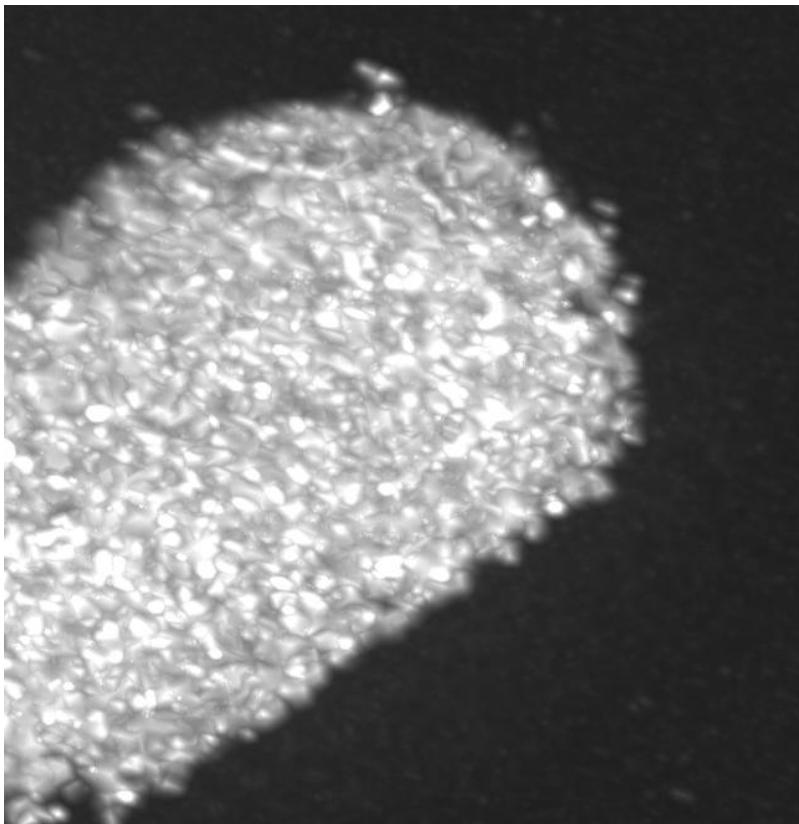
+

“Faces” / Triangles

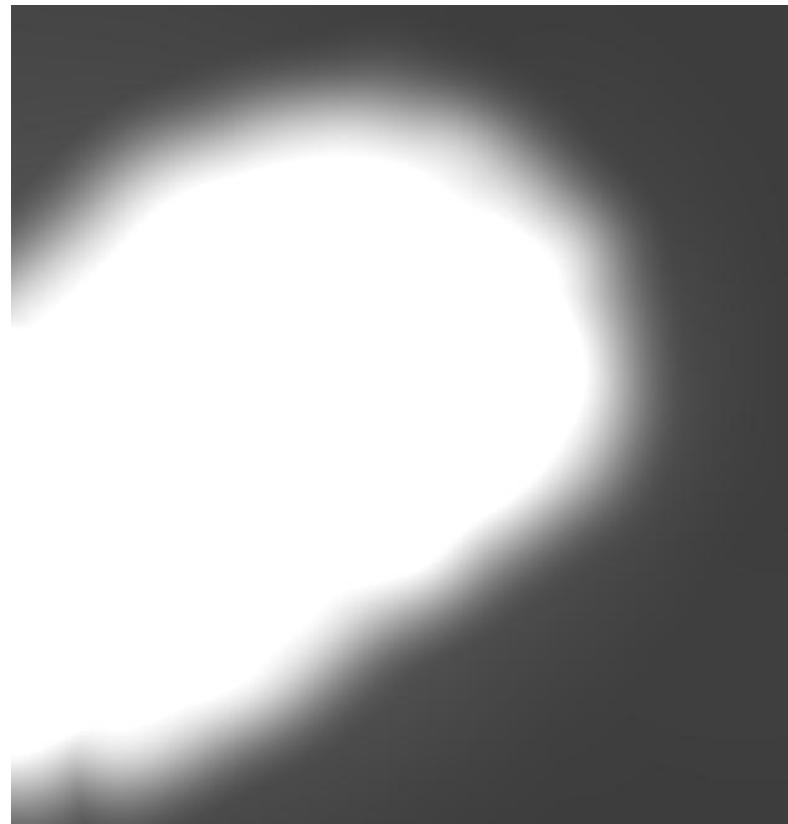
Point 1	Point 2	Point 3
1	2	3
1	2	4
2	3	4
1	3	4



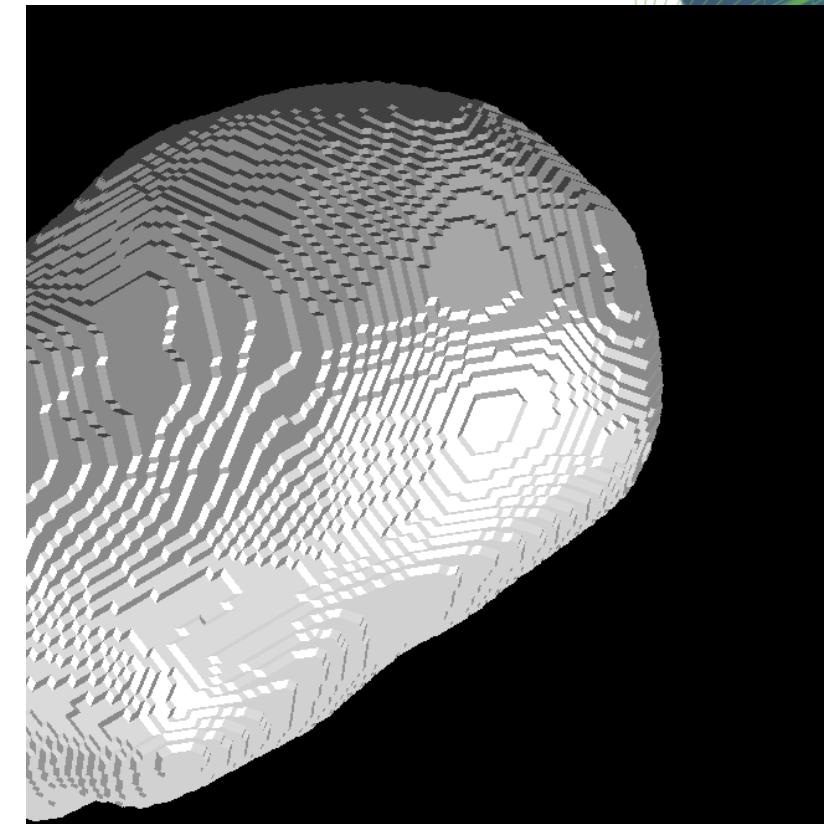
# Surface reconstruction



3D image of nuclei



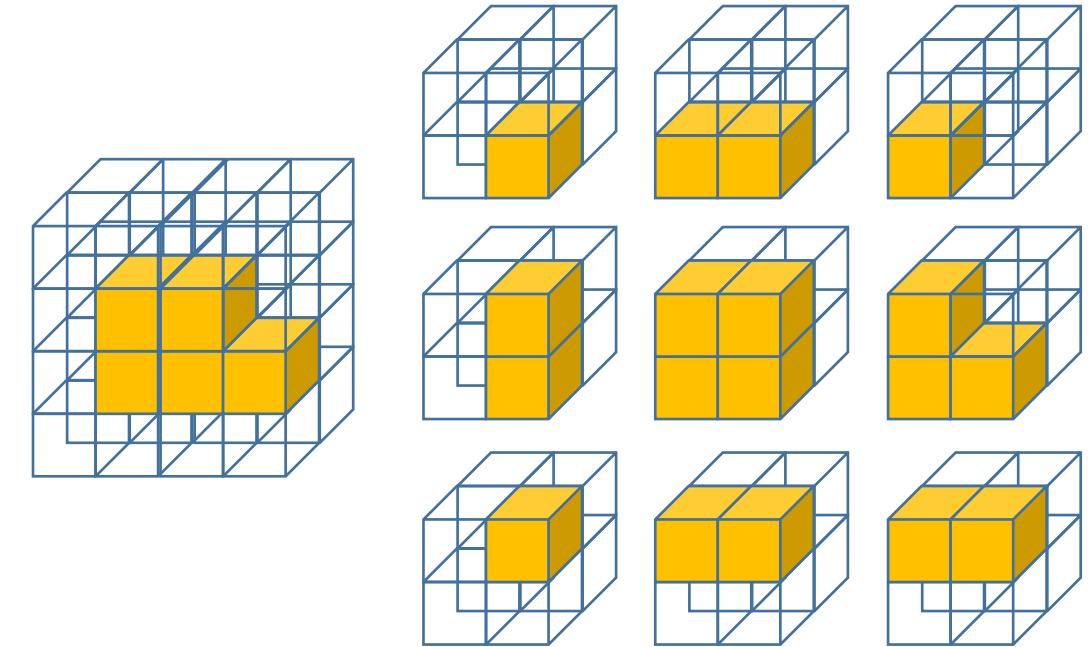
Gaussian filtered



Binary 3D image  
(visualized as surface mesh)

# Marching cubes algorithm

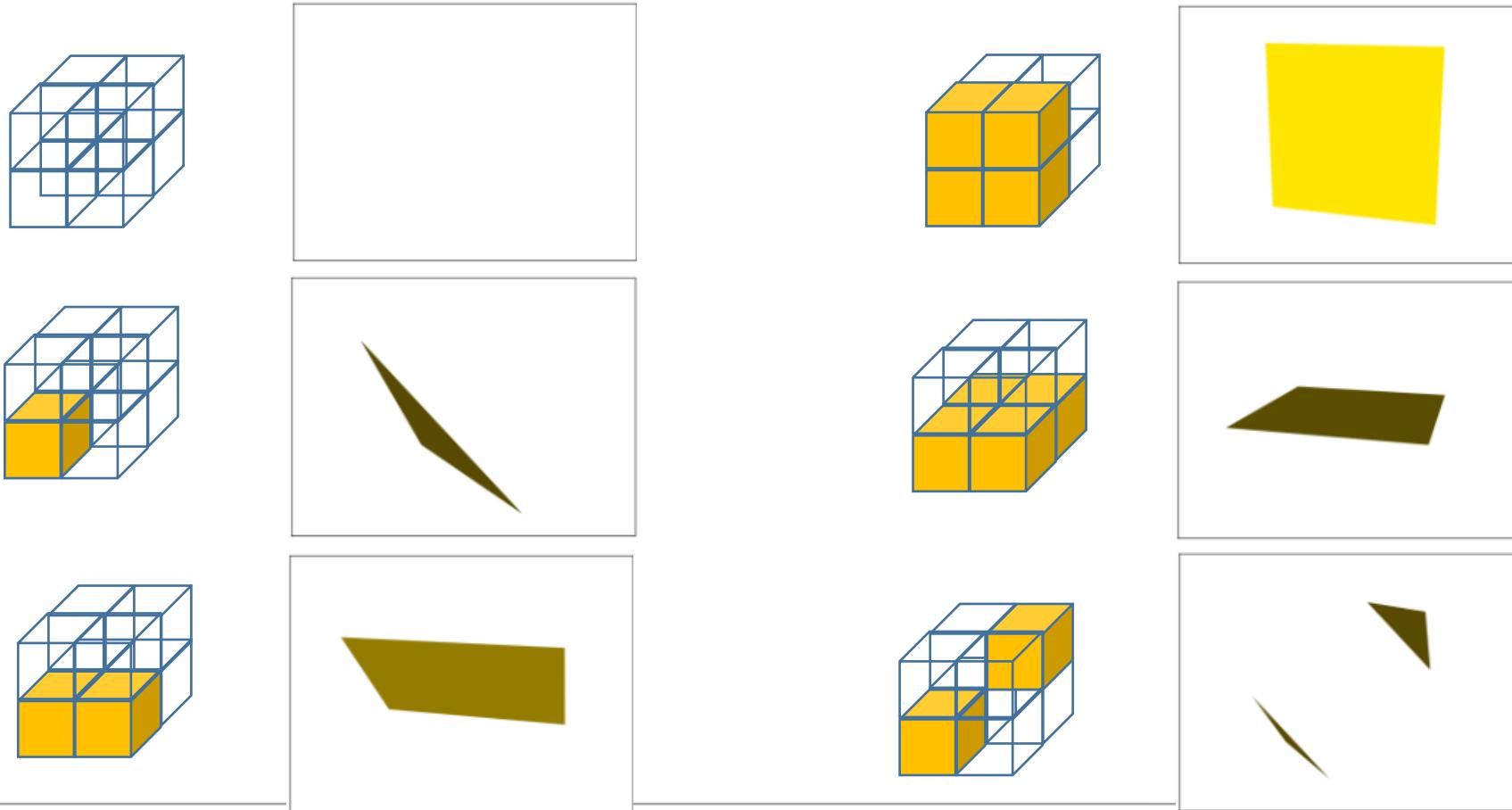
- Starting point: 3D binary image
- Cuts the image in small cubes and iterates over them



Split into cubes

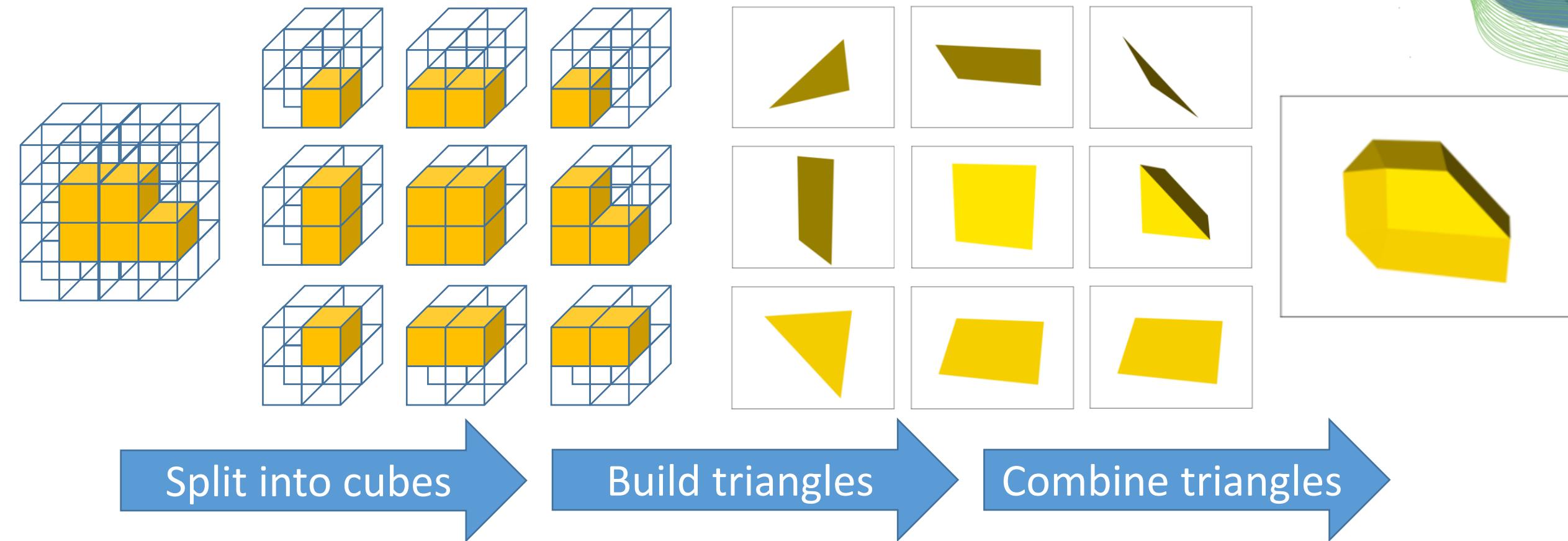
# Marching cubes algorithm

- Starting point: 3D binary image
- Cuts the image in small cubes and iterates over them



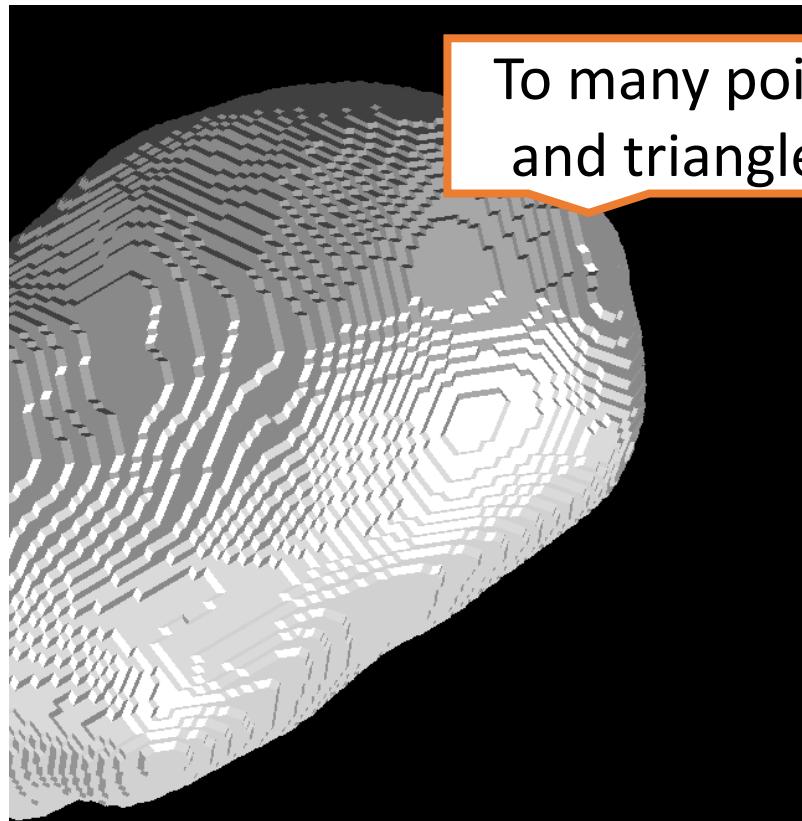
# Marching cubes algorithm

- Starting point: 3D binary image
- Cuts the image in small cubes and iterates over them

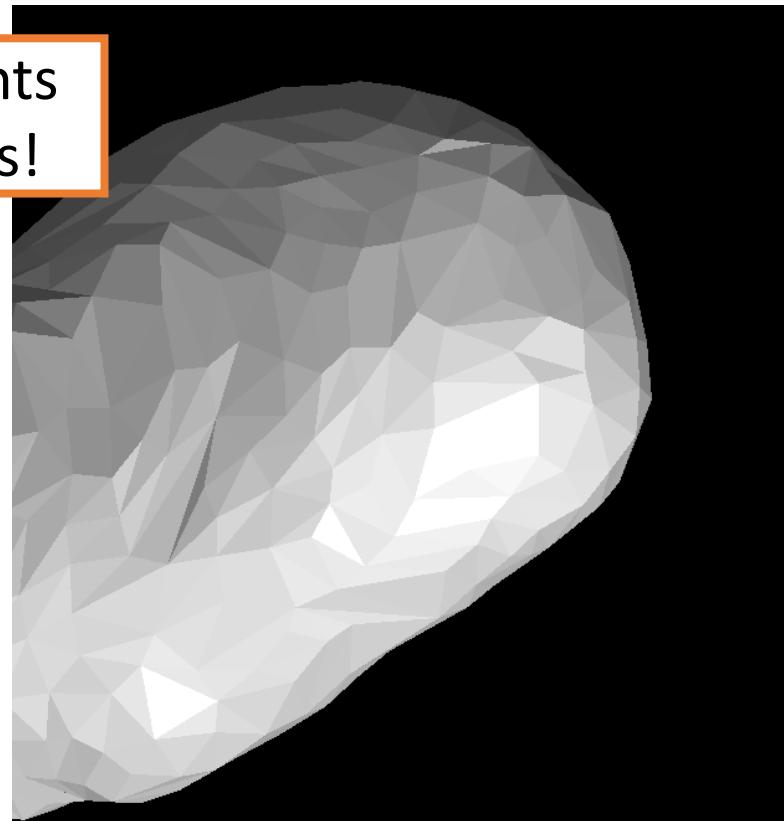


# Surface post-processing

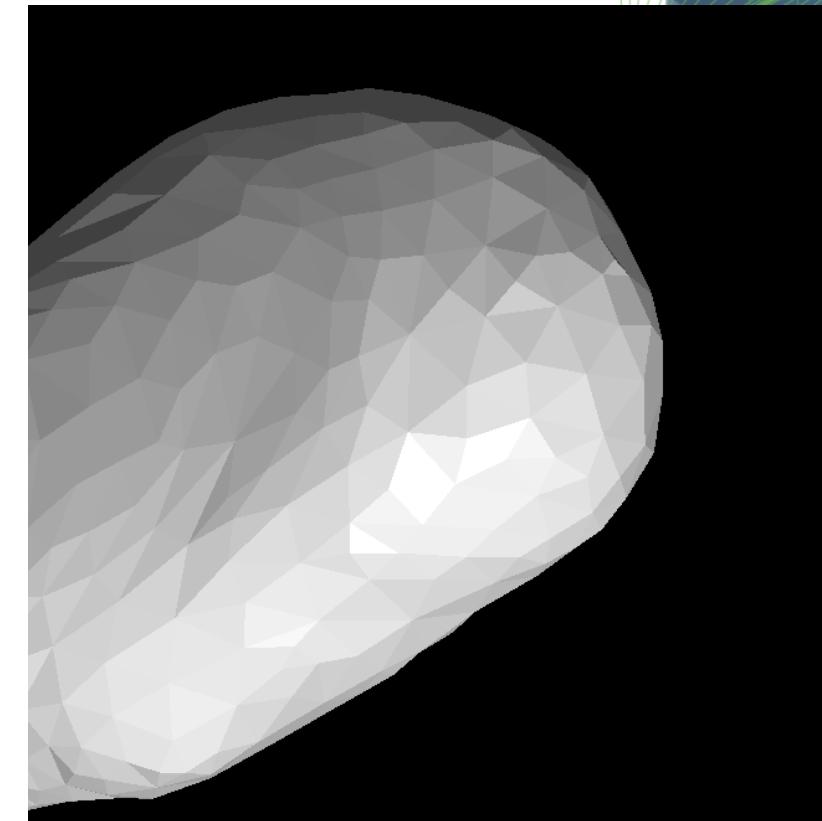
- Necessary to better match biological reality.



Marching cubes result



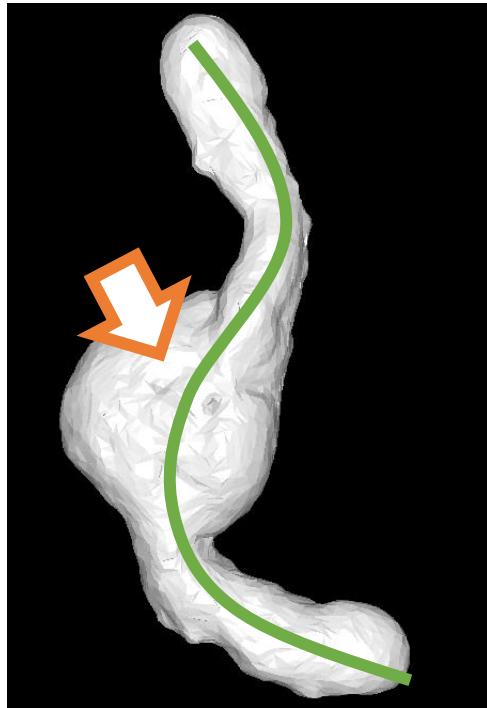
Simplified mesh  
(less points, locally averaged)



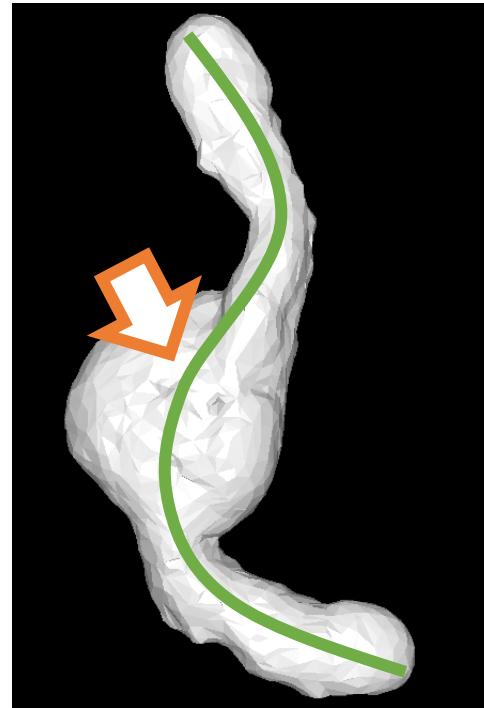
Smoothed mesh  
(position locally planarized)

# Surface post-processing

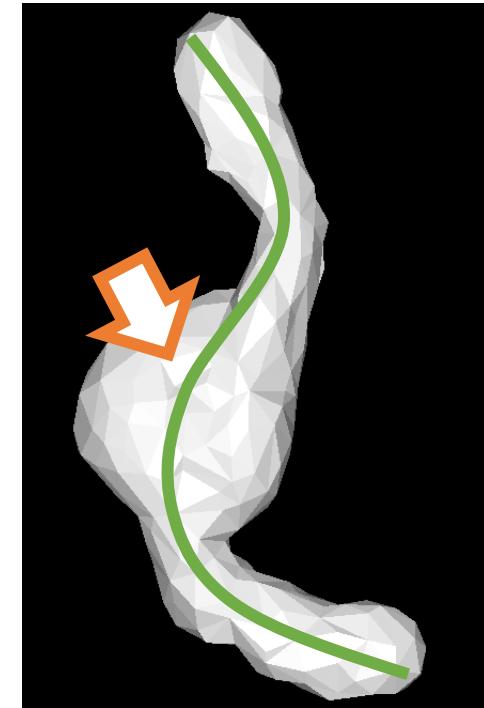
- Every processing step has consequences errors of later measurements
- Depends on desired measurement



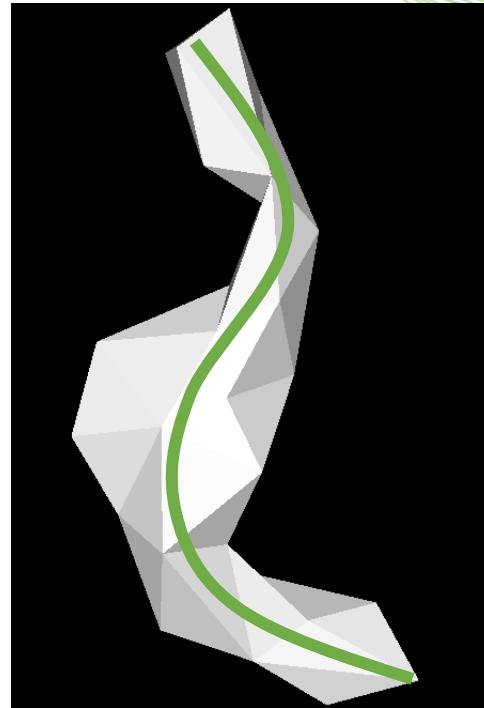
Surface mesh



Simplified by factor 0.5



Simplified by factor 0.05



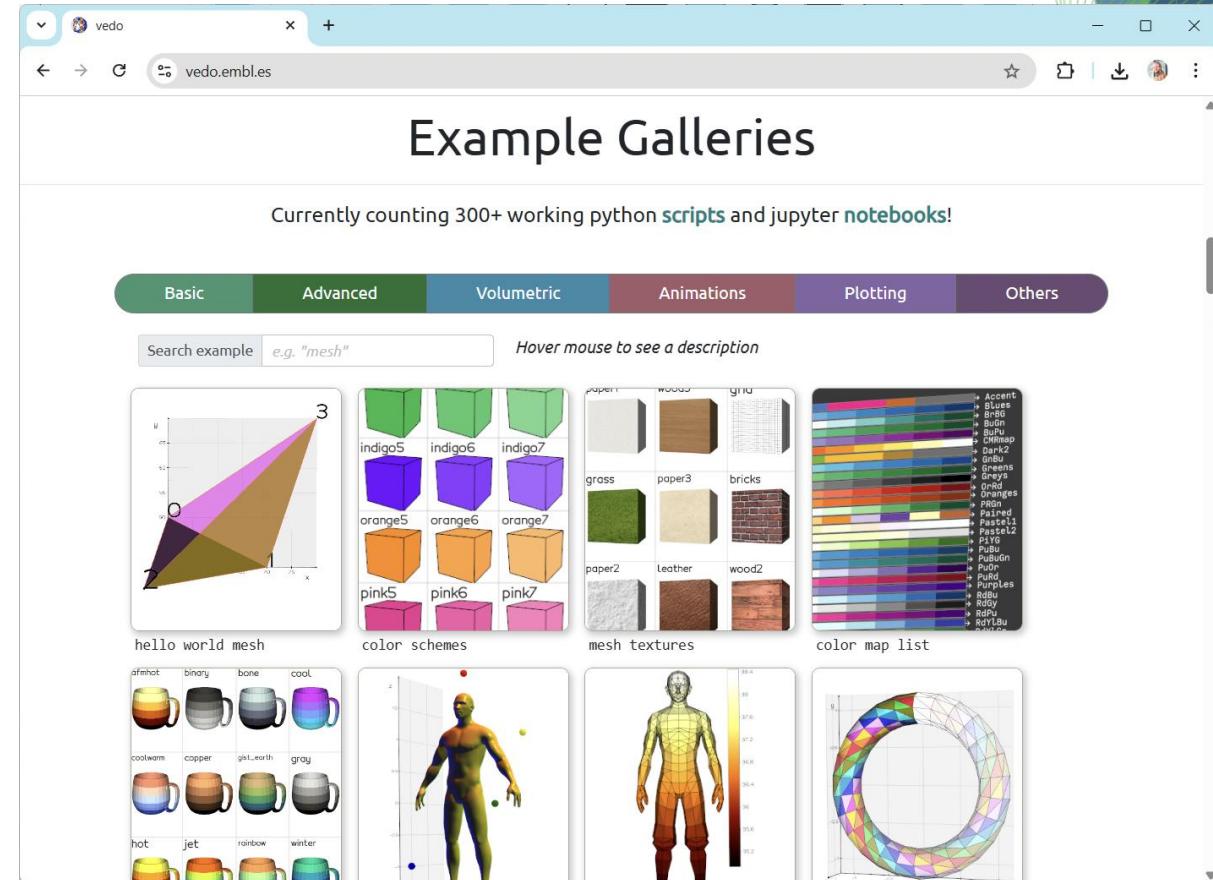
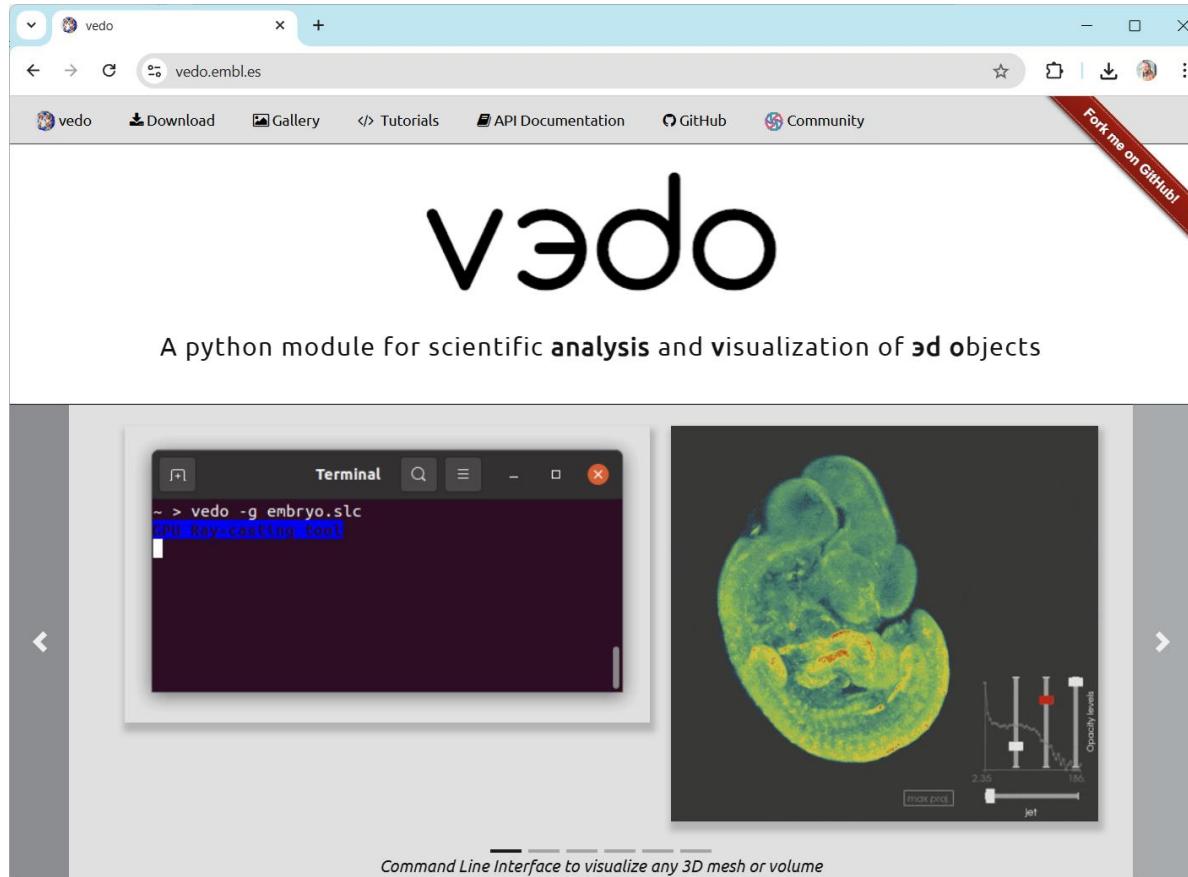
Simplified by factor 0.01

Number of small concave regions

Total length

# Surface processing: vedo

- Open source mesh + point cloud processing library (MIT licensed)



# Surface post-processing

- Meshes are lists of points [vertices] and triangles [faces]

```
[8]: mesh.points
```

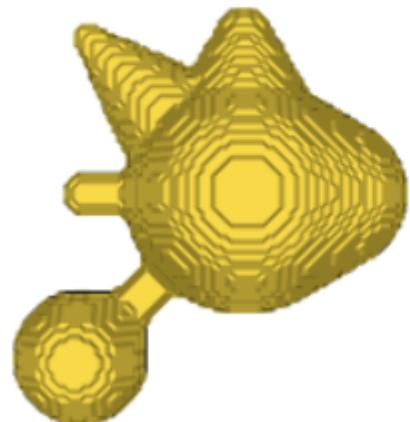
```
[8]: array([[ 47. ,  44. , -25.5],  
           [ 46.5,  44. , -26. ],  
           [ 47. ,  43.5, -26. ],  
           ...,  
           [ 51. ,  56. , -74.5],  
           [ 52. ,  56. , -74.5],  
           [ 53. ,  56. , -74.5]], dtype=float32)
```

```
[9]: mesh.cells[:10]
```

```
[9]: [[2, 1, 0],  
      [4, 3, 0],  
      [0, 3, 2],  
      [6, 5, 4],  
      [4, 5, 3].
```

```
[5]: mesh
```

```
[5]:
```



Mesh: vedo.mesh.Mesh

bounds  
(x/y/z) 2.500 ... 83.50  
2.500 ... 88.50  
-74.50 ... -25.50

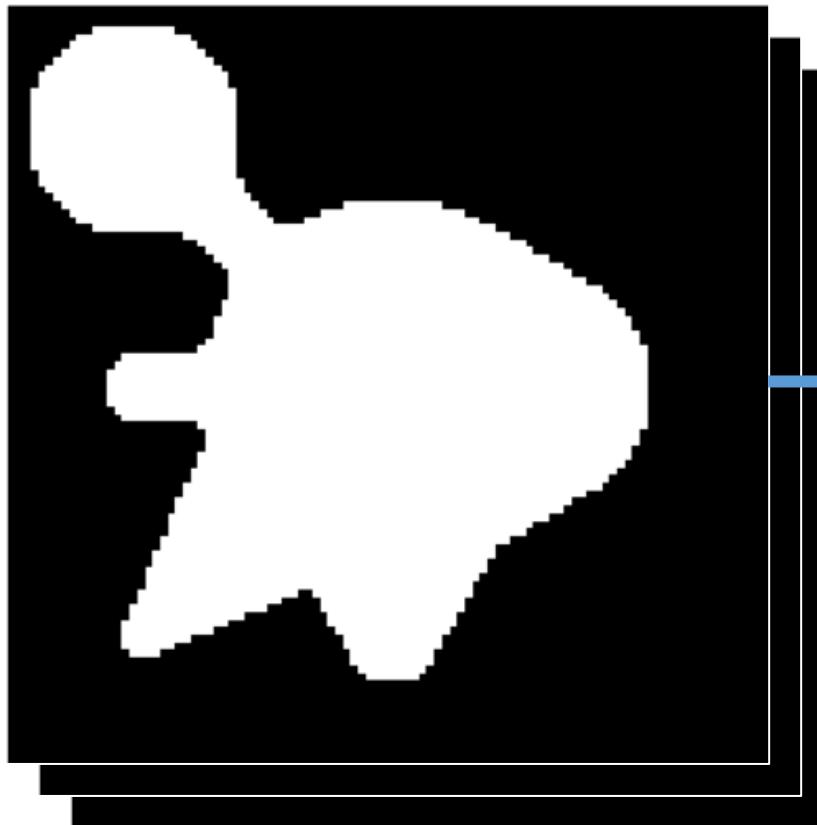
center of mass (42.6, 46.6, -50.0)

average size 31.277

nr. points / faces 19040 / 38076

# Surface reconstruction with vedo

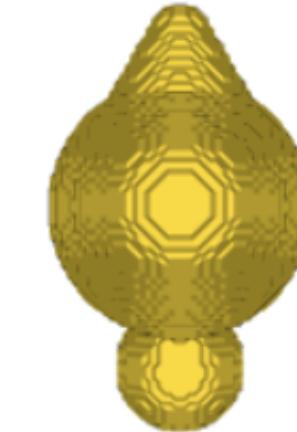
- Turn binary and/or label images into surface meshes



```
[3]: verts, faces, normals, values = marching_cubes(binary_image)

mesh = vedo.mesh.Mesh((verts, faces))
mesh
```

```
[3]:
```



Mesh: vedo.mesh.Mesh

**bounds** 25.50 ... 74.50  
(x/y/z) 2.500 ... 88.50  
2.500 ... 83.50

**center of mass** (50.0, 46.6, 42.6)

**average size** 31.277

**nr. points / faces** 19040 / 38076

# Processing surface meshes with vedo

- Object oriented: mesh... [hit Shift-Tab, to learn more!]

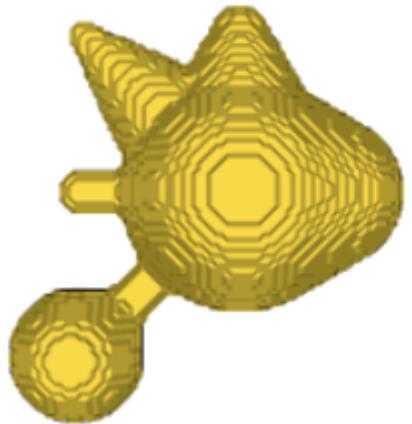
```
[4]: mesh.r
[4]: f render_lines_as_tubes    function
      f render_points_as_spheres function
      i rendered_at                 instance
      f reorient                   function
      f resample_data_from        function
      f reverse                     function
      f rotate                      function
      f rotate_x                    function
      f rotate_y                    function
      f rotate_z                    function
```

# Processing surface meshes with vedo

- Object oriented: mesh...

```
[4]: mesh.rotate_y(90)
```

```
[4]:
```



Mesh: vedo.mesh.Mesh

**bounds**  
(x/y/z) 2.500 ... 83.50  
2.500 ... 88.50  
-74.50 ... -25.50

**center of mass** (42.6, 46.6, -50.0)

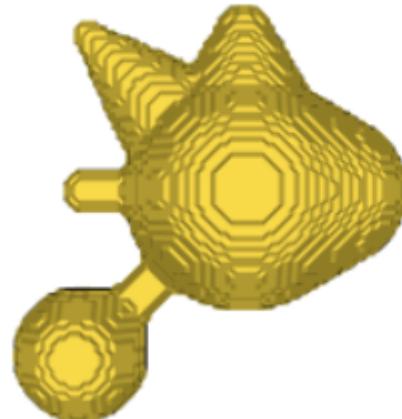
**average size** 31.277

**nr. points / faces** 19040 / 38076

**Pitfall:** vedo uses in-place operations. Calling a function modifies data!

```
[5]: mesh
```

```
[5]:
```



Mesh: vedo.mesh.Mesh

**bounds**  
(x/y/z) 2.500 ... 83.50  
2.500 ... 88.50  
-74.50 ... -25.50

**center of mass** (42.6, 46.6, -50.0)

**average size** 31.277

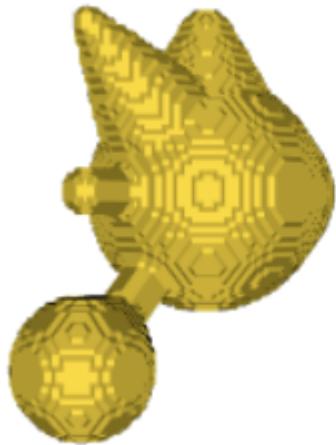
**nr. points / faces** 19040 / 38076

# Processing surface meshes with vedo

- Copy objects to prevent changing the original data

```
[6]: rotated = mesh.copy().rotate_y(angle=45)  
rotated
```

[6]:



Mesh: vedo.mesh.Mesh

**bounds** -37.83 ... 28.64  
(x/y/z) 2.500 ... 88.50  
-99.35 ... -32.88

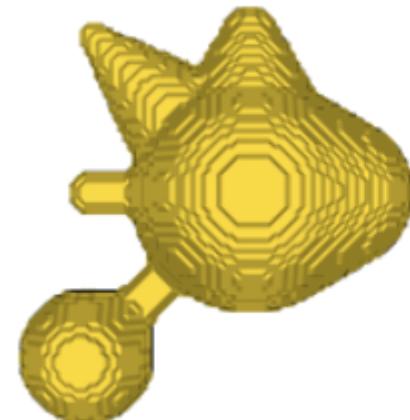
**center of mass** (-5.24, 46.6, -65.5)

**average size** 31.277

**nr. points / faces** 19040 / 38076

```
[7]: mesh
```

[7]:



Mesh: vedo.mesh.Mesh

**bounds** 2.500 ... 83.50  
2.500 ... 88.50  
(x/y/z) -74.50 ... -25.50

**center of mass** (42.6, 46.6, -50.0)

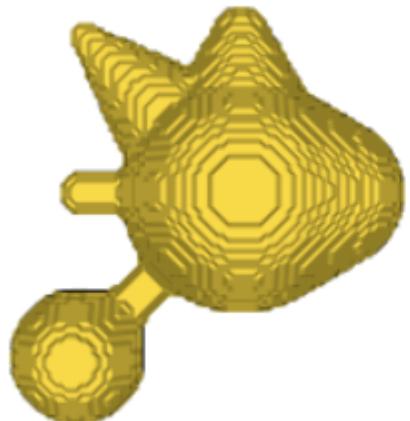
**average size** 31.277

**nr. points / faces** 19040 / 38076

# Surface mesh processing

- Surface mesh simplification
- To prevent the computer freezing

```
[7]: mesh
```



Mesh: vedo.mesh.Mesh

bounds  
(x/y/z)  
2.500 ... 83.50  
2.500 ... 88.50  
-74.50 ... -25.50

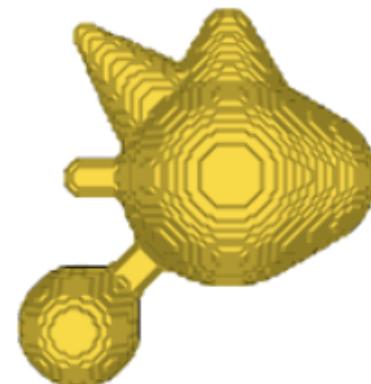
center of mass (42.6, 46.6, -50.0)

average size 31.277

nr. points / faces 19040 / 38076

```
[13]: decimated_mesh = mesh.copy().decimate(fraction=0.5)
decimated_mesh
```

```
[13]:
```



Mesh: vedo.mesh.Mesh

bounds  
(x/y/z)  
2.500 ... 83.50  
2.500 ... 88.50  
-74.50 ... -25.50

center of mass (39.9, 43.6, -48.5)

average size 31.100

nr. points / faces 9521 / 19038

```
[16]:
```



Mesh: vedo.mesh.Mesh

bounds  
(x/y/z)  
4.806 ... 81.54  
5.490 ... 89.67  
-74.54 ... -25.58

center of mass (40.2, 46.8, -50.3)

average size 33.785

nr. points / faces 22 / 37

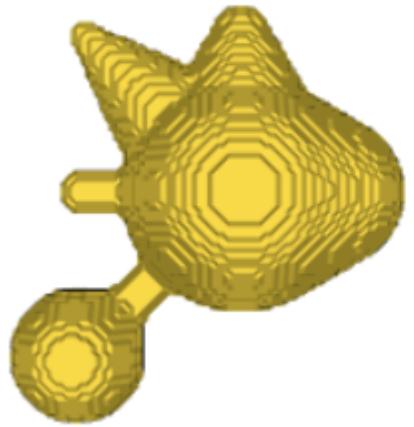
Quiz: What fraction created this surface mesh?

24

# Surface mesh processing

- Surface mesh smoothing

```
[7]: mesh
```



Mesh: vedo.mesh.Mesh

**bounds** 2.500 ... 83.50  
2.500 ... 88.50  
(x/y/z) -74.50 ... -25.50

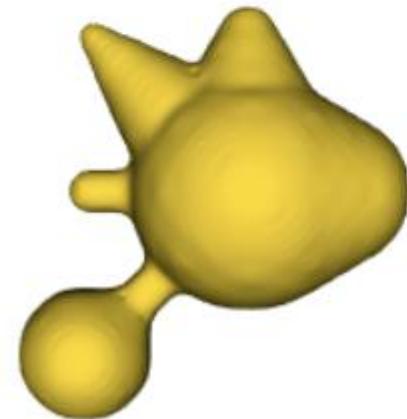
**center of mass** (42.6, 46.6, -50.0)

**average size** 31.277

**nr. points / faces** 19040 / 38076

```
[27]: smoothed_mesh = mesh.copy().smooth(niter=15,  
pass_band=0.0001,  
edge_angle=15,  
feature_angle=60,  
boundary=False)
```

```
smoothed_mesh
```



Mesh: vedo.mesh.Mesh

**bounds** 2.386 ... 83.61  
2.383 ... 88.59  
(x/y/z) -74.58 ... -25.42

**center of mass** (42.6, 46.6, -50.0)

**average size** 31.277

**nr. points / faces** 19040 / 38076

# Surface mesh processing

ChatGPT 4o

Share

[28]: `mesh.smooth?`

**Signature:**

```
mesh.smooth(  
    niter=15,  
    pass_band=0.1,  
    edge_angle=15,  
    feature_angle=60,  
    boundary=False,  
) -> Self
```

**Docstring:**

Adjust mesh point positions using the so-called "Windowed Sinc" method.

**Arguments:**

`niter : (int)`  
number of iterations.

`pass_band : (float)`  
set the `pass_band` value for the windowed sinc filter.

`edge_angle : (float)`  
edge angle to control smoothing along edges (either interior or boundary).

`feature_angle : (float)`  
specifies the feature angle for sharp edge identification.

`boundary : (bool)`  
specify if boundary should also be smoothed or kept unmodified

In surface smoothing, what is the set the `pass_band` option for the windowed sinc filter?

In surface smoothing (particularly in image or volumetric data processing), the `pass_band` option for a **windowed sinc filter** typically defines the **normalized cutoff frequency** of the filter — that is, it controls how much of the frequency spectrum is preserved.

## Key Concepts:

- The **sinc filter** is an ideal low-pass filter in the frequency domain.
- The **windowed sinc filter** applies a tapering function (like Hamming or Blackman window) to limit the sinc kernel spatially.
- The `pass_band` parameter defines the **highest frequency allowed to pass without attenuation**, expressed as a fraction of the Nyquist frequency (which is half the sampling rate).

## Typical Usage:

- A `pass_band` value of 1.0 allows all frequencies up to Nyquist to pass (i.e., no smoothing).
- A `pass_band` value of 0.1 to 0.5 applies increasing smoothing, with **lower values producing smoother results**.
- It is usually a float between 0 and 1.

# Surface mesh processing

```
[28]: mesh.smooth?
```

**Signature:**

```
mesh.smooth(  
    niter=15,  
    pass_band=0.1,  
    edge_angle=15,  
    feature_angle=60,  
    boundary=False,  
) -> Self
```

**Docstring:**

Adjust mesh point positions using the so-called "Windowed Sinc" method.

**Arguments:**

niter : (int)  
 number of iterations.

pass\_band : (float)  
 set the pass\_band value for the windowed sinc filter.

edge\_angle : (float)  
 edge angle to control smoothing along edges (either interior or boundary).

feature\_angle : (float)  
 specifies the feature angle for sharp edge identification.

boundary : (bool)  
 specify if boundary should also be smoothed or kept unmodified

## Explanation for pass\_band option in smooth function #1248

Open

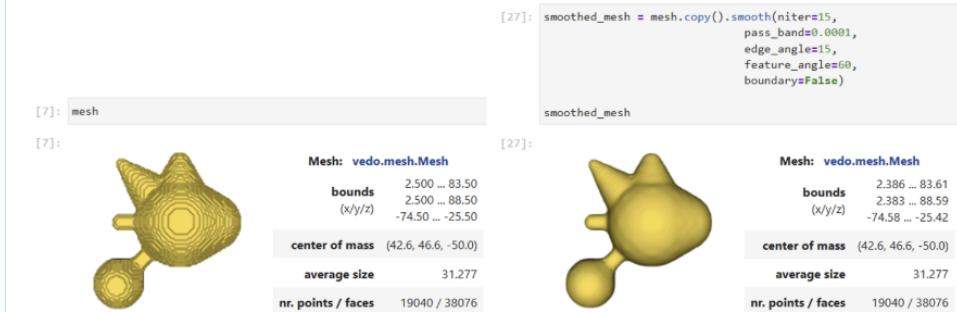


haesleinhuepf opened 3 days ago

Hi @marcomusy,

I hope you are doing well!

I'm wondering how to explain the `pass_band` option of the `Mesh.smooth` function. I suspect that this parameter allows us to remove edge/corner artifacts introduced by the marching-cubes algorithm.



However, I struggle to explain (myself) what this `pass_band` option actually does. The docstring unfortunately only says "(float) set the pass\_band value for the windowed sinc filter." ChatGPT explains it like "The `pass_band` parameter defines the highest frequency allowed to pass without attenuation, expressed as a fraction of the Nyquist frequency (which is half the sampling rate)." Do you by chance know a better explanation? Or do you have publicly available training materials explaining this? I'm searching for a nice explanation / analogy for students in a course of the university.

Thanks!

Best,  
Robert

# Surface mesh processing



marcomusy 2 days ago

Owner ...

Hi Robert! Hope you're doing fine too!

This filter is better described [here](#).

Its purpose is to reduce the "high frequency noise" in the position of the vertices without modifying the topology of the mesh.

An analogy would be cutting off the high values of in the fourier transform space and then anti-transform to the original space. Another similar example of this is given in 3D by [this example](#):

```
vedo --run lowpassfilter
```

In the specific case of a polygonal mesh something similar is achieved through the socalled "window sinc algorithm". The windowed sinc function is

$$\text{sinc}(x) = \sin(\pi * x) / (\pi * x)$$

Windowed Sinc Kernel:

$$h(x) = \text{sinc}(x) * w(x)$$

(where  $w(x)$  is the window function)

the smoothed vertex position becomes:

$$p_v' = \sum(w_{vu} * p_u) \text{ for all } u \in N(v)$$

(where  $p_v'$  is the new position of vertex  $v$ ,  $p_u$  is the position of neighbor  $u$ ,  $w_{vu}$  are weights from the kernel, and  $N(v)$  is the set of neighboring vertices)

The pass\_band parameter (0 to 2) controls smoothing intensity; lower values increase smoothing by filtering out higher frequencies.

I'll try to improve the description of the method in the vedo docs! Hope this helps!

Detailed Description

adjust point positions using a windowed sinc function interpolation kernel

**vtkWindowedSincPolyDataFilter** adjusts point coordinates using a windowed sinc function interpolation kernel. The effect is to "relax" or "smooth" the mesh, making the cells better shaped and the vertices more evenly distributed. Note that this filter operates on the lines, polygons, and triangle strips composing an instance of **vtkPolyData**. Vertex or poly-vertex cells are never modified.

The algorithm proceeds as follows. For each vertex  $v$ , a topological and geometric analysis is performed to determine which vertices are connected to  $v$ , and which cells are connected to  $v$ . Then, a connectivity array is constructed for each vertex. (The connectivity array is a list of lists of vertices that directly attach to each vertex, the so-called smoothing stencil.) Next, an iteration phase begins over all vertices. For each vertex  $v$ , the coordinates of  $v$  are modified using a windowed sinc function interpolation kernel. Taubin describes this methodology in the IBM tech report RC-2040 (#90237, dated 3/12/96) "Optimal Surface Smoothing as Filter Design" G. Taubin, T. Zhang and G. Golub. (Zhang and Golub are at Stanford University.)

This report discusses using standard signal processing low-pass filters (in particular windowed sinc functions) to smooth polyhedra. The transfer functions of the low-pass filters are approximated by Chebyshev polynomials. This facilitates applying the filters in an iterative diffusion process (as opposed to a kernel convolution). The more smoothing iterations applied, the higher the degree of polynomial approximating the low-pass filter transfer function. Each smoothing iteration, therefore, applies the next higher term of the Chebyshev filter approximation to the polyhedron. This decoupling of the filter into an iteratively applied polynomial is possible since the Chebyshev polynomials are orthogonal, i.e. increasing the order of the approximation to the filter transfer function does not alter the previously calculated coefficients for the low order terms.

Note: Care must be taken to avoid smoothing with too few iterations. A Chebyshev approximation with too few terms is a poor approximation. The first few smoothing iterations represent a severe scaling and translation of the data. Subsequent iterations cause the smoothed polyhedron to converge to the true location and scale of the object. We have attempted to protect against this by automatically adjusting the filter, effectively widening the pass band. This adjustment is only possible if the number of iterations is greater than 1. Note that this sacrifices some degree of smoothing for model integrity. For those interested, the filter is adjusted by searching for a value sigma such that the actual pass band is  $k_{pb} + \sigma$  and such that the filter transfer function evaluates to unity at  $k_{pb}$ , i.e.  $f(k_{pb}) = 1$

To improve the numerical stability of the solution, and minimize the scaling and translation effects, the algorithm can translate and scale the position coordinates to within the unit cube [-1, 1], perform the smoothing, and translate and scale the position coordinates back to the original coordinate frame. This mode is controlled with the `NormalizeCoordinatesOn()` / `NormalizeCoordinatesOff()` methods. For legacy reasons, the default is `NormalizeCoordinatesOff`.

This implementation is currently limited to using an interpolation kernel based on Hamming windows. Other windows (such as Hann, Blackman, Kaiser, Lanczos, Gaussian, and exponential windows) could be used instead.

There are some special instance variables used to control the execution of this filter. (These ivars basically control what vertices can be smoothed, and the creation of the connectivity array.) The `BoundarySmoothing` ivar enables/disables the smoothing operation on vertices that are on the "boundary" of the mesh. A boundary vertex is one that is surrounded by a semi-cycle of polygons (or used by a single line).

Another important ivar is `FeatureEdgeSmoothing`. If this ivar is enabled, then interior vertices are classified as either "simple", "interior edge", or "fixed", and smoothed differently. (Interior vertices are manifold vertices surrounded by a cycle of polygons; or used by two line cells.) The classification is based on the number of feature edges attached to  $v$ . A feature edge occurs when the angle between the two surface normals of a polygon sharing an edge is greater than the `FeatureAngle` ivar. Then, vertices used by no feature edges are classified "simple", vertices used by exactly two feature edges are classified "interior edge", and all others are "fixed" vertices.

Once the classification is known, the vertices are smoothed differently. Corner (i.e., fixed) vertices are not smoothed at all. Simple vertices are smoothed as before. Interior edge vertices are smoothed only along their two connected edges, and only if the angle between the edges is less than the `EdgeAngle` ivar.

The total smoothing can be controlled by using two ivars. The `NumberOfIterations` determines the maximum number of smoothing passes. The `NumberOfIterations` corresponds to the degree of the polynomial that is used to approximate the windowed sinc function. Ten or twenty iterations is usually necessary. Contrast this with `vtkSmoothPolyDataFilter` which usually requires 100 to 200 smoothing iterations. `vtkSmoothPolyDataFilter` is also not an approximation to an ideal low-pass filter, which can cause the geometry to shrink as the amount of smoothing increases.

# Surface mesh processing



marcomusy 2 days ago

Owner ...

Hi Robert! Hope you're doing fine too!

This filter is better described [here](#).

Its purpose is to reduce the "high frequency noise" in the position of the vertices without modifying the topology of the mesh.

An analogy would be cutting off the high values of in the fourier transform space and then anti-transform to the original space. Another similar example of this is given in 3D by [this example](#):

```
vedo --run lowpassfilter
```



In the specific case of a polygonal mesh something similar is achieved through the socalled "window sinc algorithm". The windowed sinc function is

$$\text{sinc}(x) = \sin(\pi * x) / (\pi * x)$$

Windowed Sinc Kernel:

$$h(x) = \text{sinc}(x) * w(x)$$

(where  $w(x)$  is the window function)

the smoothed vertex position becomes:

$$p_v' = \sum(w_{vu} * p_u) \text{ for all } u \in N(v)$$

(where  $p_v'$  is the new position of vertex  $v$ ,  $p_u$  is the position of neighbor  $u$ ,  $w_{vu}$  are weights from the kernel, and  $N(v)$  is the set of neighboring vertices)

The pass\_band parameter (0 to 2) controls smoothing intensity; lower values increase smoothing by filtering out higher frequencies.

I'll try to improve the description of the method in the vedo docs! Hope this helps!



haesleinhuepf 2 days ago

Author ...

Awesome thanks for the feedback! You think it would be ok to call it a low-pass filter that modifies the positions of vertices?



marcomusy 2 days ago

Owner ...

Yes! it is a signal processing-based smoothing method. Just ensure the context clarifies that it operates on mesh geometry, not raw signals, to avoid possible confusion.



1

Lesson learned: Reaching out to developers in case anything is unclear. Most are happy to answer questions and about feedback allowing them to improve docs.

# View surface meshes in Napari

```
[5]: viewer = napari.Viewer(ndisplay=3)

[6]: def to_napari_surface_tuple(vedo_mesh):
    import numpy as np
    return (vedo_mesh.points, np.asarray(vedo_mesh.cells))

viewer.add_surface(to_napari_surface_tuple(surface))

napari.utils.nbscreenshot(viewer)
```



Start Napari in 3D-mode

Surface meshes in Napari are tuples of (vertices, faces)

# View surface meshes in Napari

- You can modify the view in napari, by changing camera parameters.

```
[7]: viewer.camera.angles = [0,0,0]

napari.utils.nbscreenshot(viewer)
```

```
[7]:
```



Use SHIFT-Tab  
for more options

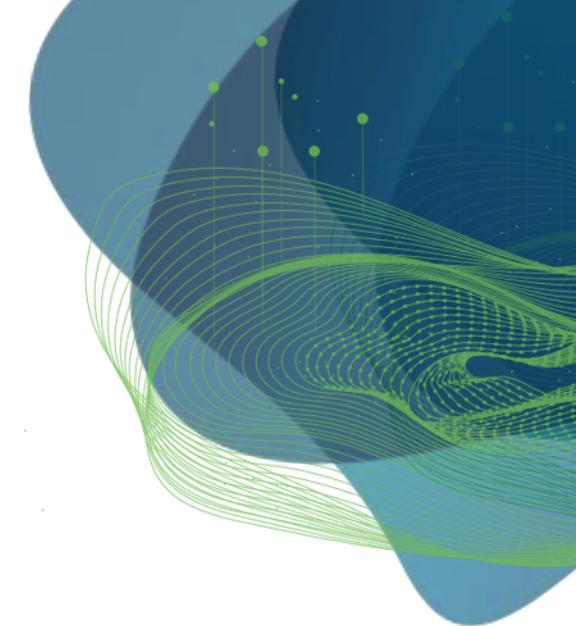
```
[ ]: viewer.camera.
```

```
reset
schema
schema_json
set_view_direction
up_direction
update
update_forward_refs
validate
view_direction
zoom
```



DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS  
AND ARTIFICIAL INTELLIGENCE



# Feature extraction

Robert Haase

GEFÖRDERT VOM



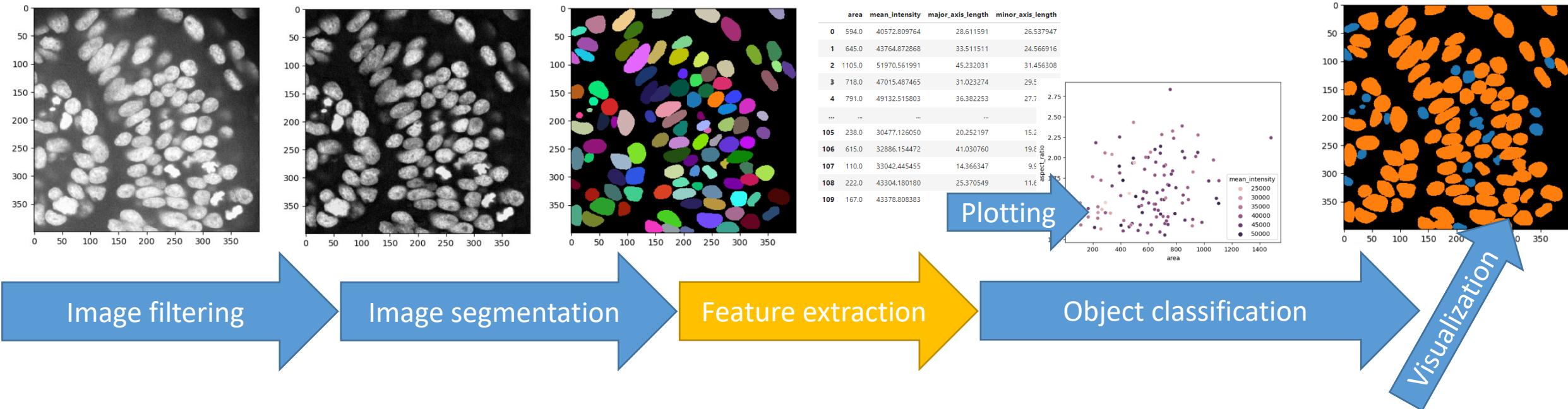
Bundesministerium  
für Bildung  
und Forschung



Diese Maßnahme wird gefördert durch die Bundesregierung  
aufgrund eines Beschlusses des Deutschen Bundestages.  
Diese Maßnahme wird mitfinanziert durch Steuermittel auf  
der Grundlage des von den Abgeordneten des Sächsischen  
Landtags beschlossenen Haushaltes.

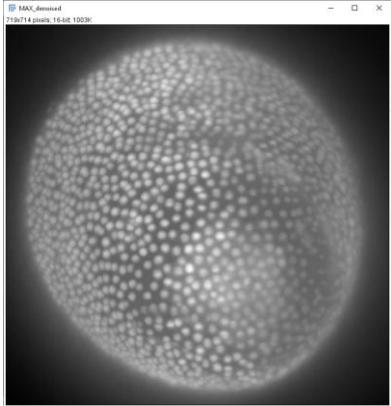
# Lecture overview: Bio-image Analysis

- Image Data Analysis workflows
- Goal: **Quantify observations, substantiate conclusions with numbers**

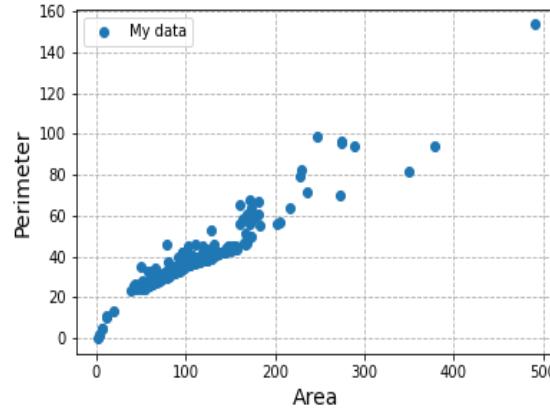


# Feature extraction

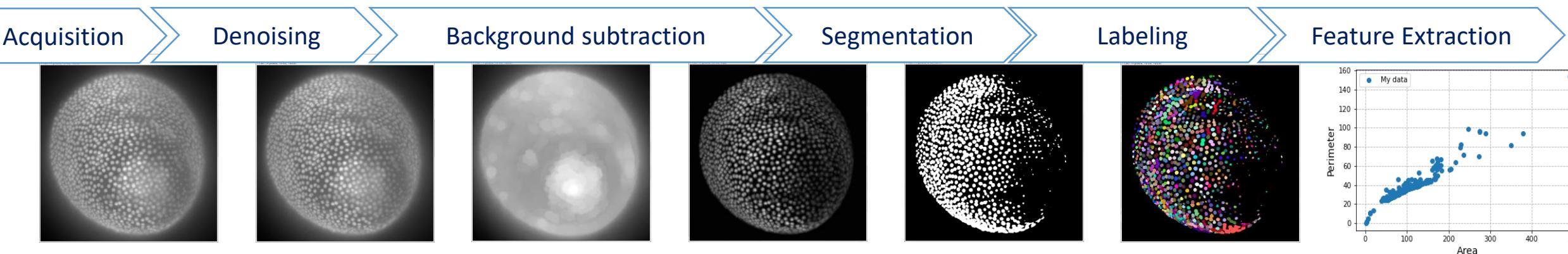
- Feature extraction is a *late* processing step in image analysis.
- It can be used for images or



Feature Extraction



- or segmented/labelled images



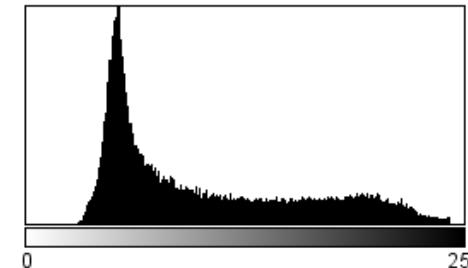
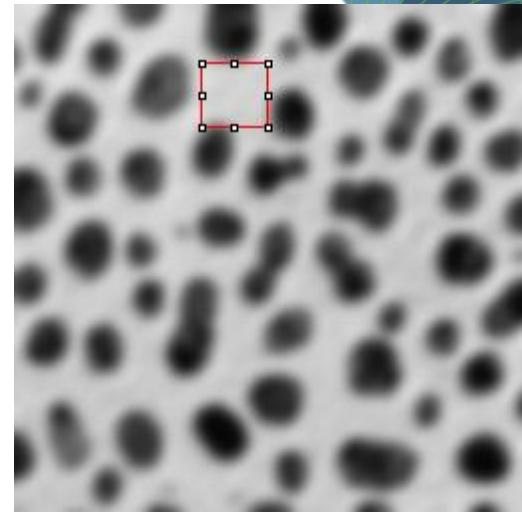
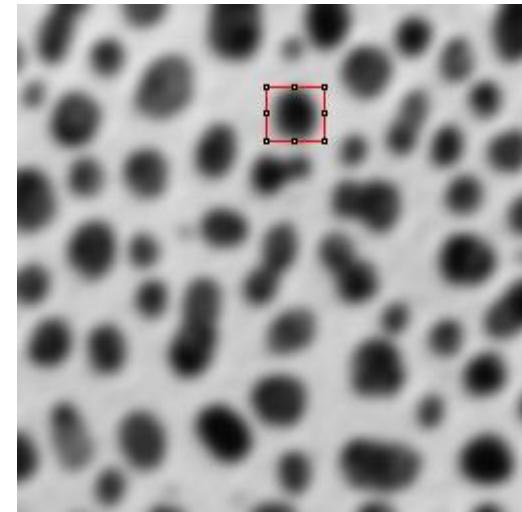
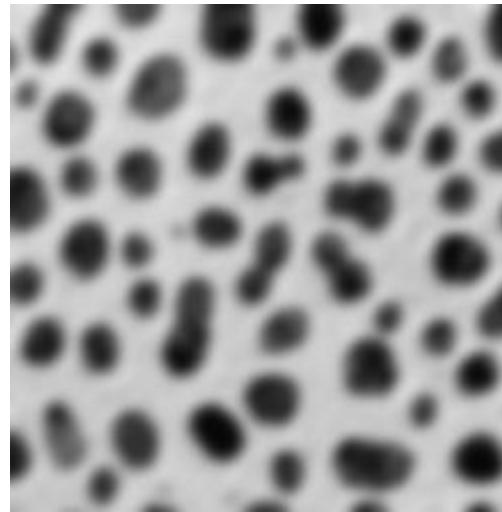
# Feature extraction



- 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
- **Mixed features**
  - Center of mass
  - Local minima / maxima
  - Distance to neighbors
  - Average intensity in neighborhood
- **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

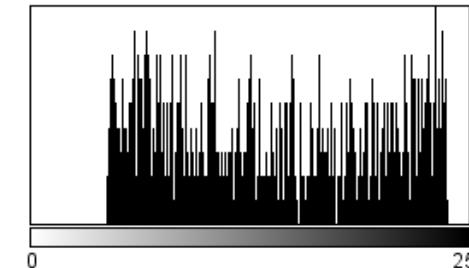
# Intensity based features

- 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



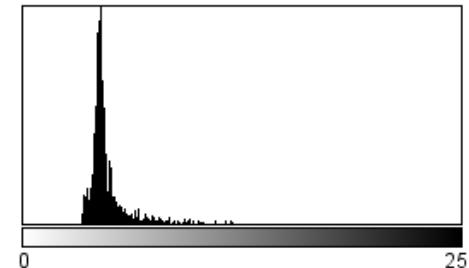
Count: 65024  
Mean: 103.301  
StdDev: 57.991

Min: 29  
Max: 248  
Mode: 53 (1663)



Count: 783  
Mean: 141.308  
StdDev: 61.876

Min: 44  
Max: 243  
Mode: 236 (9)



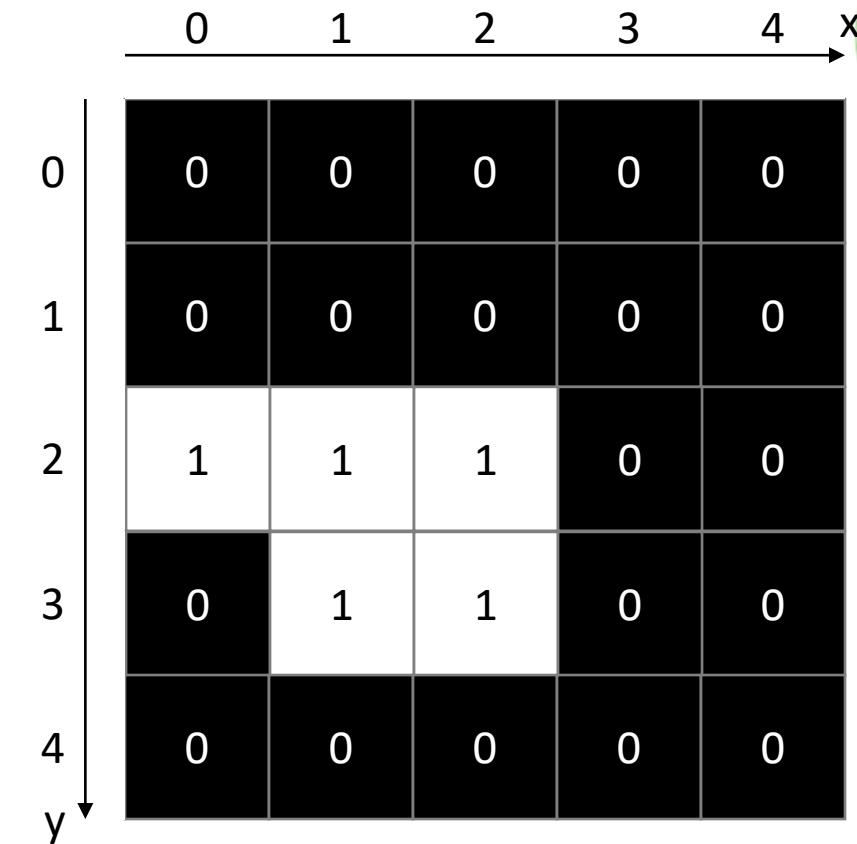
Count: 1056  
Mean: 49.016  
StdDev: 12.685

Min: 34  
Max: 122  
Mode: 45 (120)

# 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



# Center of mass

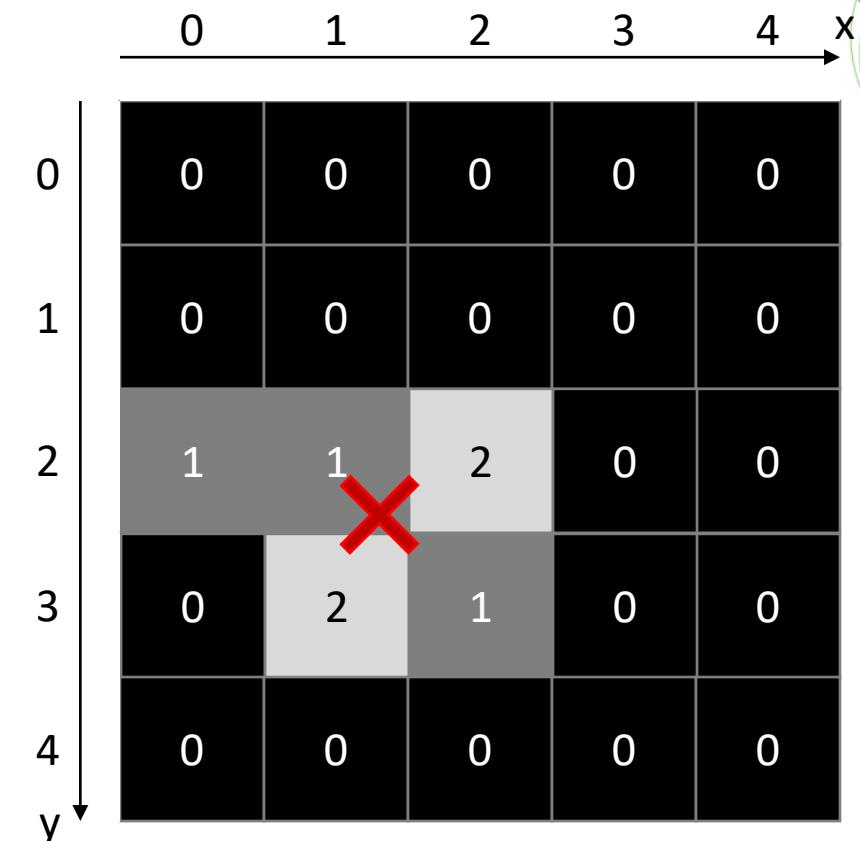
- Relative position in an image weighted by pixel intensities

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

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

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

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



$$x_m = 1/7 (1 \cdot 0 + 1 \cdot 1 + 2 \cdot 2 + 2 \cdot 1 + 1 \cdot 2) = 1.3$$

$$y_m = 1/7 (1 \cdot 2 + 1 \cdot 2 + 2 \cdot 3 + 2 \cdot 2 + 1 \cdot 3) = 2.4$$

# Center of geometry / centroid

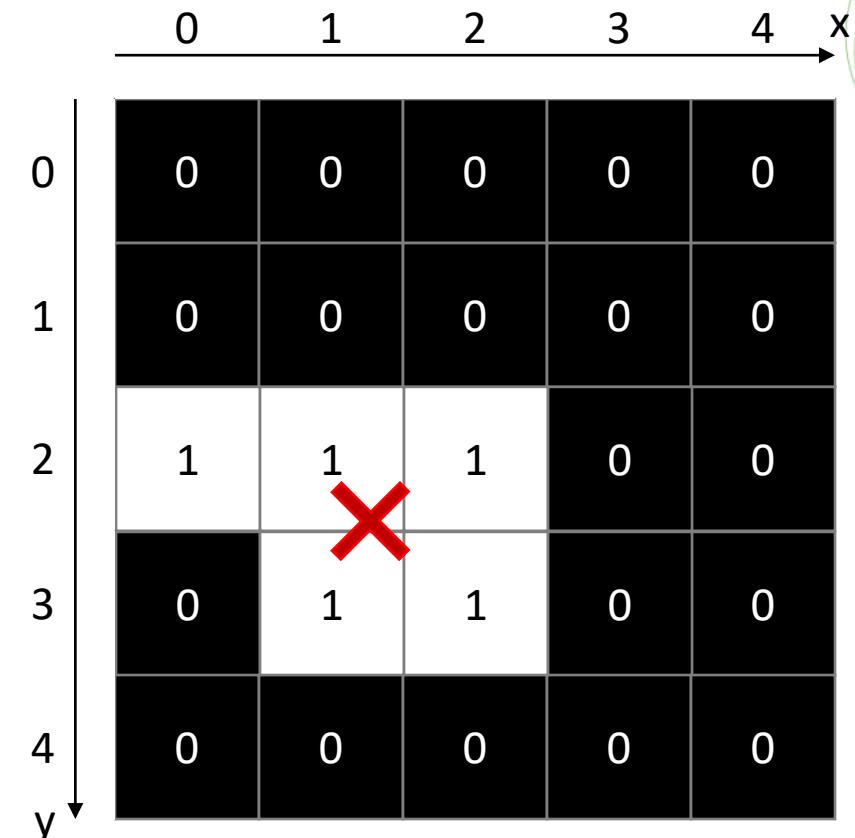
- Relative position in an image weighted by pixel intensities
- Special case of center of mass for binary images
  - $x, y$  ... pixel coordinates
  - $w$  ... image width
  - $h$  ... image height
  - $\mu$  ... mean intensity
  - $g_{x,y}$  ... pixel grey value, integer in range [0;1]
  - $x_m, y_m$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

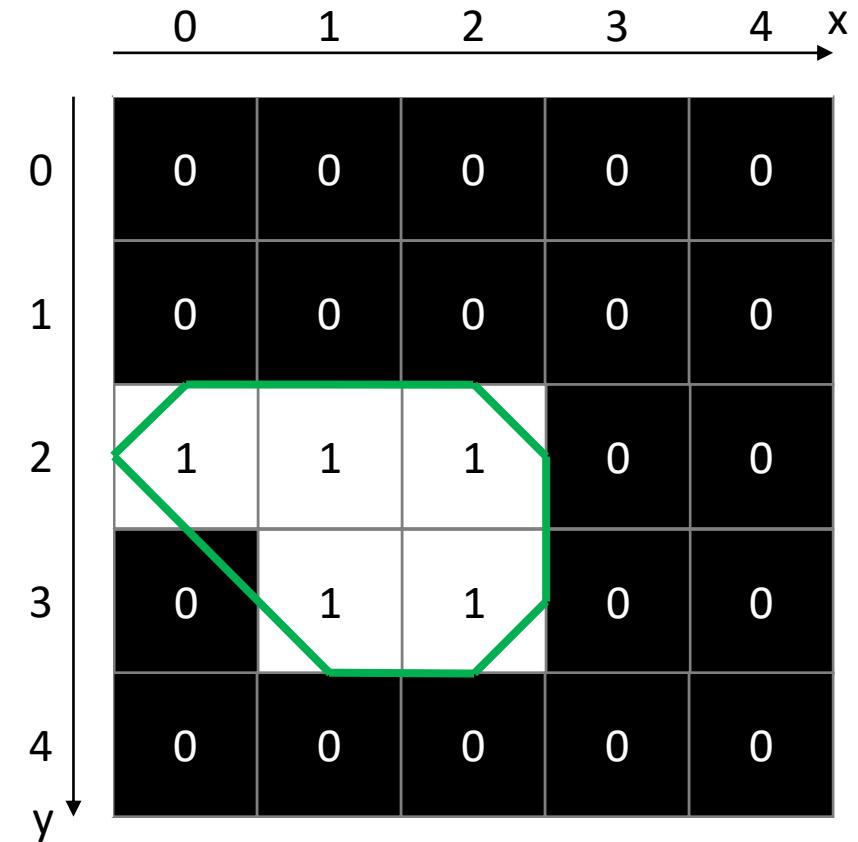
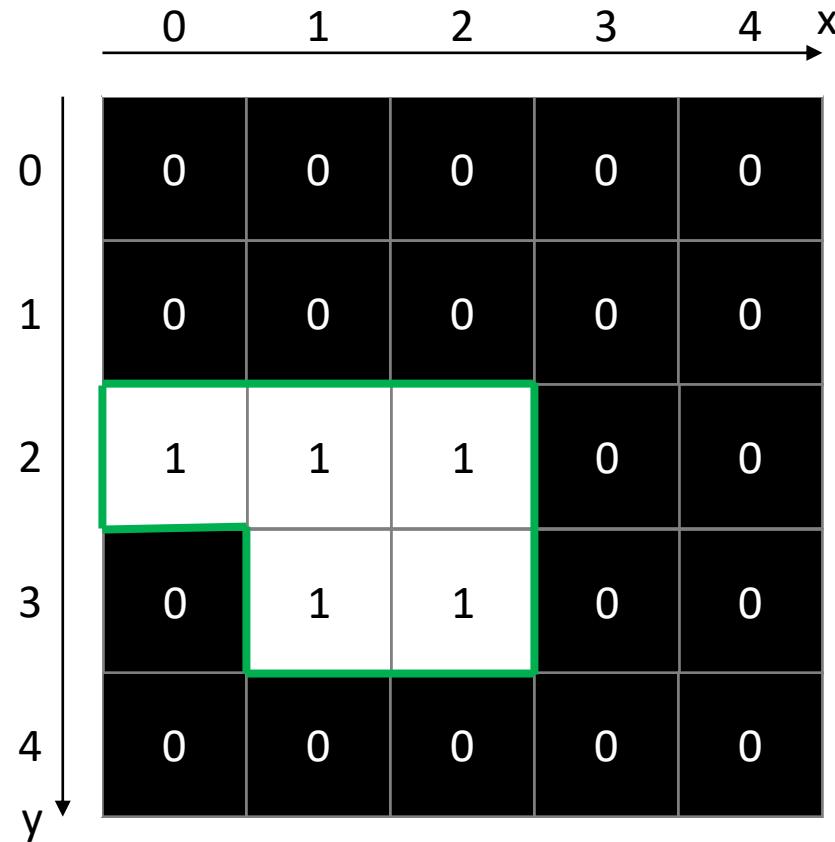


$$x_m = 1/5 (1 \cdot 0 + 1 \cdot 1 + 1 \cdot 2 + 1 \cdot 1 + 1 \cdot 2) = 1.2$$

$$y_m = 1/5 (1 \cdot 2 + 1 \cdot 2 + 1 \cdot 3 + 1 \cdot 2 + 1 \cdot 3) = 2.4$$

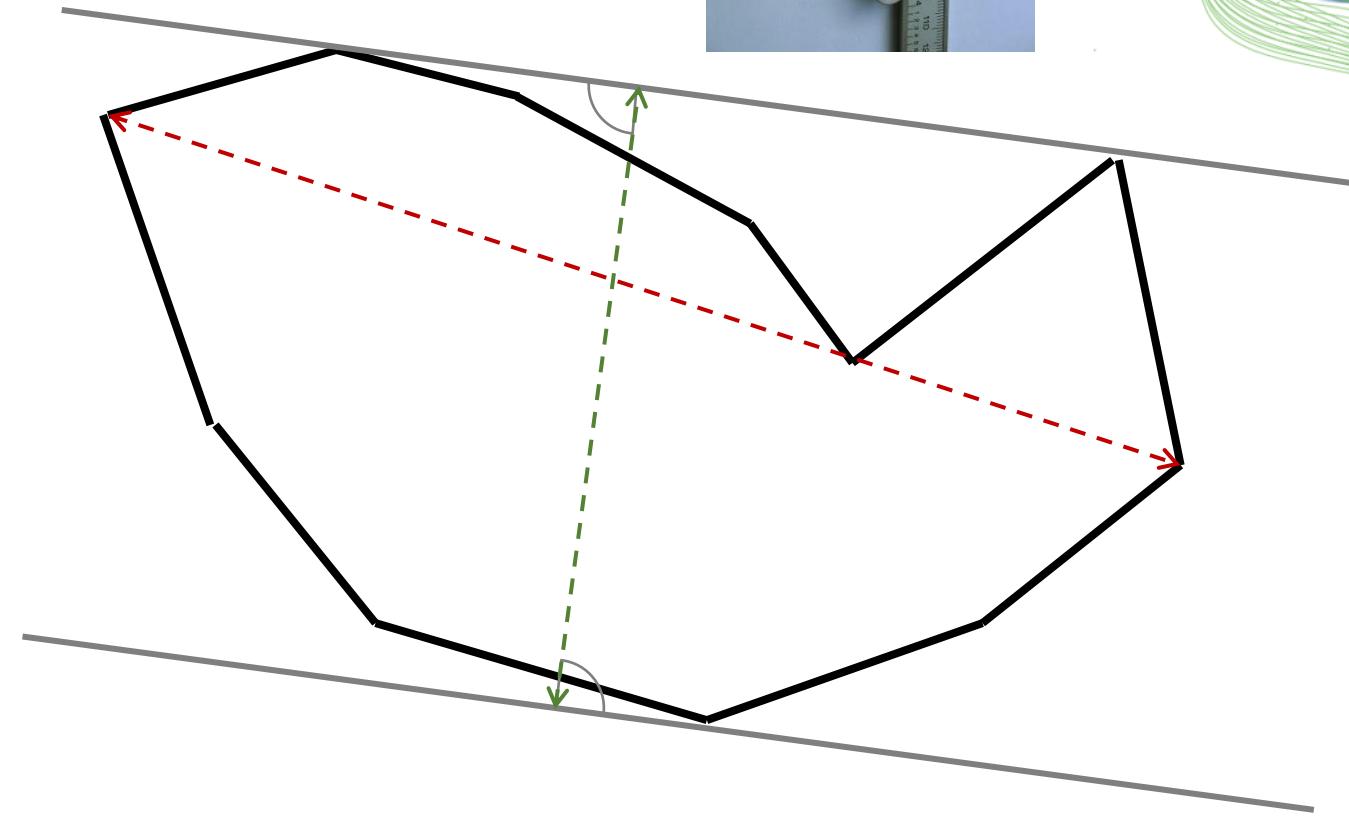
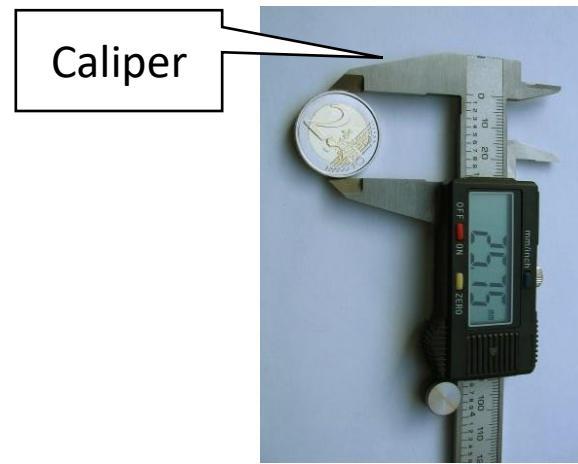
# Perimeter

- 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!



# Feret's diameter

- Feret's diameter (L.R. Feret, 1931) is often cited, but impossible to read online ...
- The term “Feret’s Diameter” was established in the 1970s

La Grosseur des grains des matières pulvérulentes  
Von: L.R. Feret  
Gedrucktes Buch, Deutsch, 19uu  
Verlag: [Eidgen. Materialprüfungsanstalt a. d. Eidgen. Technischen Hochschule], [Zürich], 19uu  
Mehr Daten anzeigen

Ausleihen in Deutsche Nationalbibliothek Leipzig der Nähe von Leipzig, Deutschland  
Ausleihen  
1,3 Kilometer entfernt

Exemplar in einer Bibliothek Filtern nach: Beliebige Medienart Beliebige Ausgabe Im Umkreis von 200+ km suchen  
1 edition in 2 libraries

Bevorzogene Bibliotheken Alle Bibliotheken  
2 Bibliotheken in der Nähe von Leipzig, Deutschland Nur Öffentliche Bibliotheken  
Deutsche Nationalbibliothek Leipzig  
1,3 Kilometer von Ihrem aktuellen Standort Deutscher Platz 1, Leipzig, 04103, Deutschland  
Ausleihen Bevorzugte Bibliothek  
Regional oder National

LA GROSSEUR DES GRAINS DES MATIÈRES PULVÉRULENTES  
par  
L. R. FERET  
Ancien Elève de l'Ecole Polytechnique,  
Chef du Laboratoire des Ponts et Chaussées de Boulogne-sur-Mer  
BOULOGNE-SUR-MER (France)

SOMMAIRE

AUSZUG

### DIE KORNGRÖSSE PULVERFÖRMIGER STOFFE

Zur Kennzeichnung der linearen Grösse von Körnern einer bestimmten Kornfraktion, unabhängig von der Größenordnung und dem zur Abscheidung benutzten Verfahren, scheint am geeignetesten das Mittel aus einer genügenden Anzahl von Messungen des Abstandes je zweier an entgegengesetzten Seiten des Umrisses der Körner gelegter Tangenten, die parallel zu einer beliebigen, aber für alle Messungen gleichen Richtung verlaufen. Die Messung geschieht unabhängig von der Lage der Körner zu der gewählten Richtung der Tangenten.

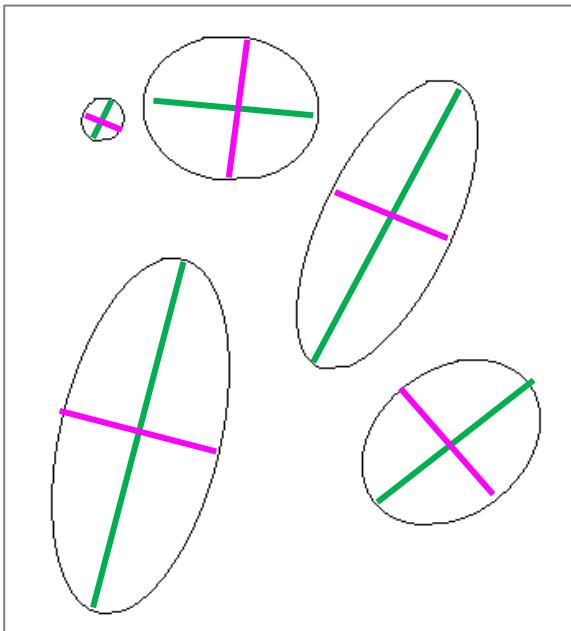
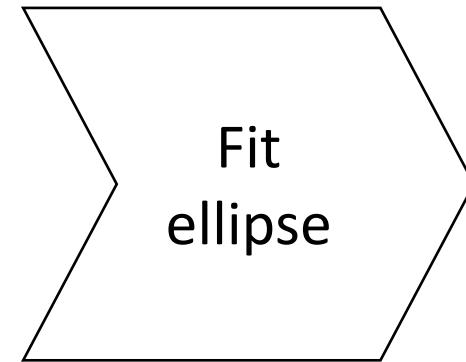
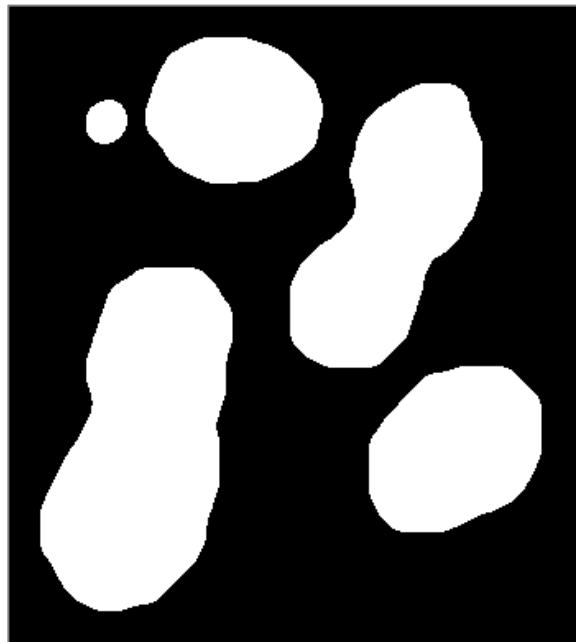
Auf Grund des so erhaltenen Mittelwertes, der als *mittlere Kornbreite* bezeichnet wird, baut Verfasser mittelst geometrischer Progressionen, die auf der Normalreihe von *Renard* beruhen, eine Einteilung nach Kornbreiten für das ganze Gebiet der gekörnten und staubförmigen Materialien auf. Die verschiedenen Kornklassen sind gekennzeichnet durch die Grenzwerte der entsprechenden *mittleren Kornbreiten* und ausserdem durch Namen, die so ausgewählt wurden, dass sie leicht in alle Sprachen eingeführt werden können.

Diese Einteilung wird vervollständigt durch eine Definition der *Kornzusammensetzung* unter Hinweis auf die Bestimmung der letzteren, entweder, ob diese Bestimmung in strenger Uebereinstimmung mit der allgemeinen Einteilung oder auf einfachere Weise im Hinblick auf gewisse gebräuchliche Anwendungen geschieht.

# Minor / major axis

- For every object, find the optimal ellipse simplifying the object.
- Major axis ... long diameter
- Minor axis ... short diameter

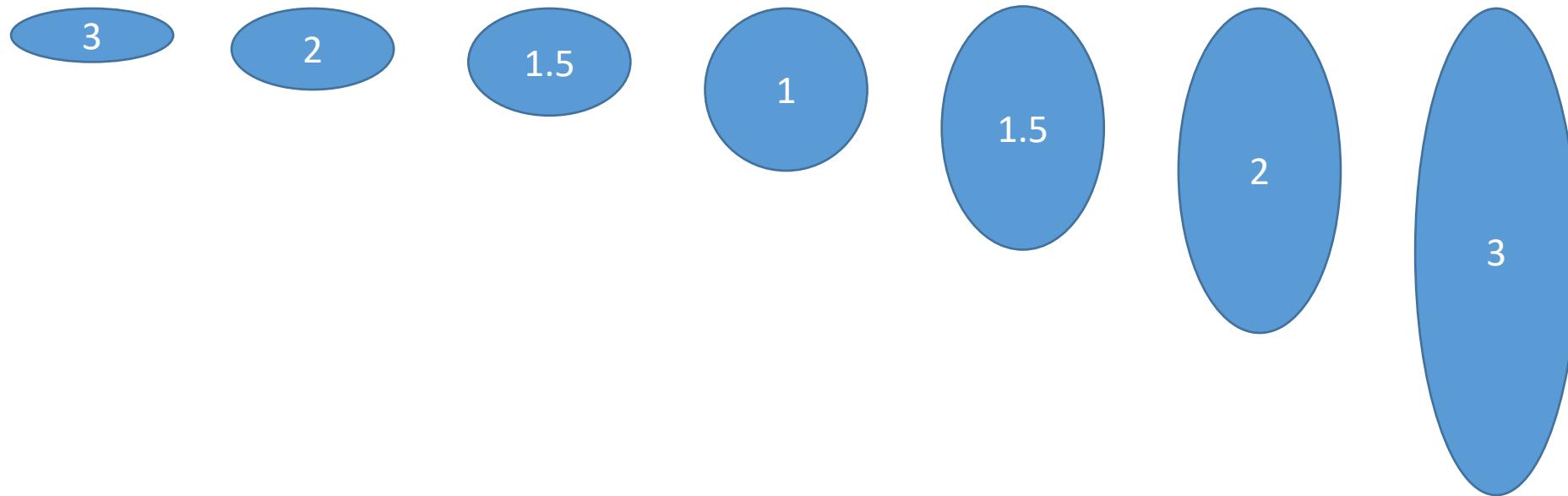
- Major and minor axis are perpendicular to each other



# Aspect ratio

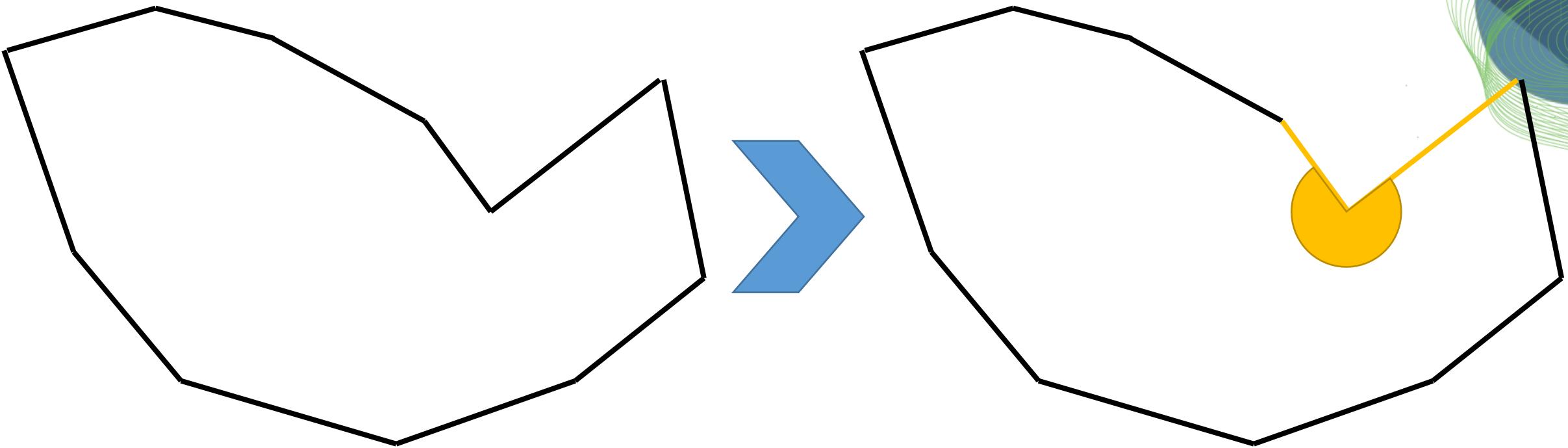
- The aspect ratio describes the elongation of an object.

$$AR = \text{major} / \text{minor}$$



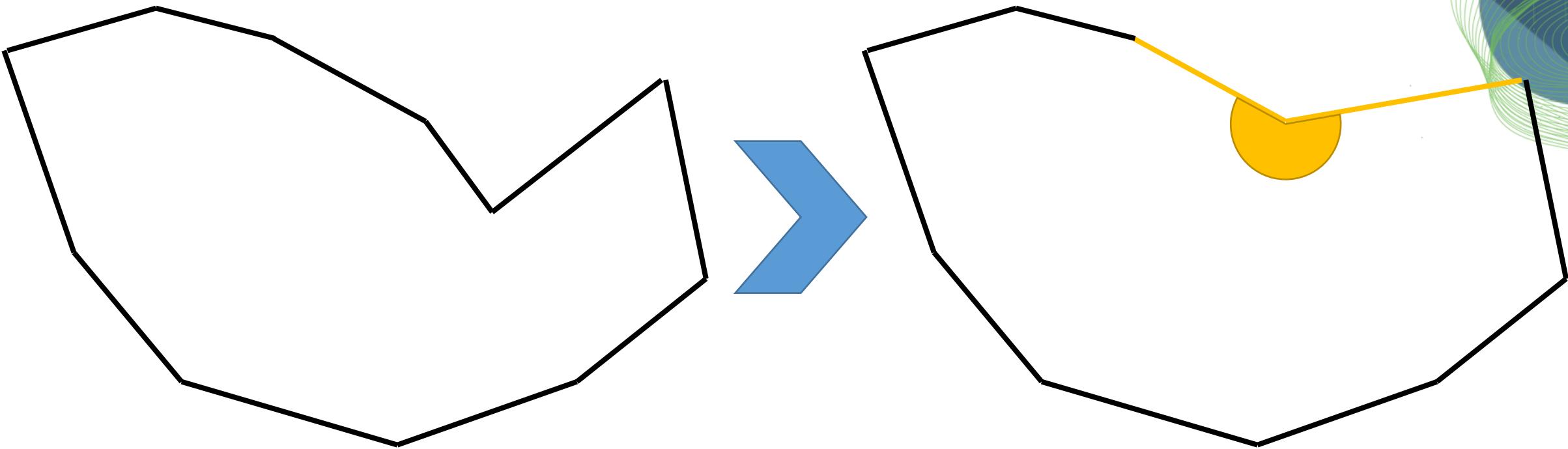
# Convex hull

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



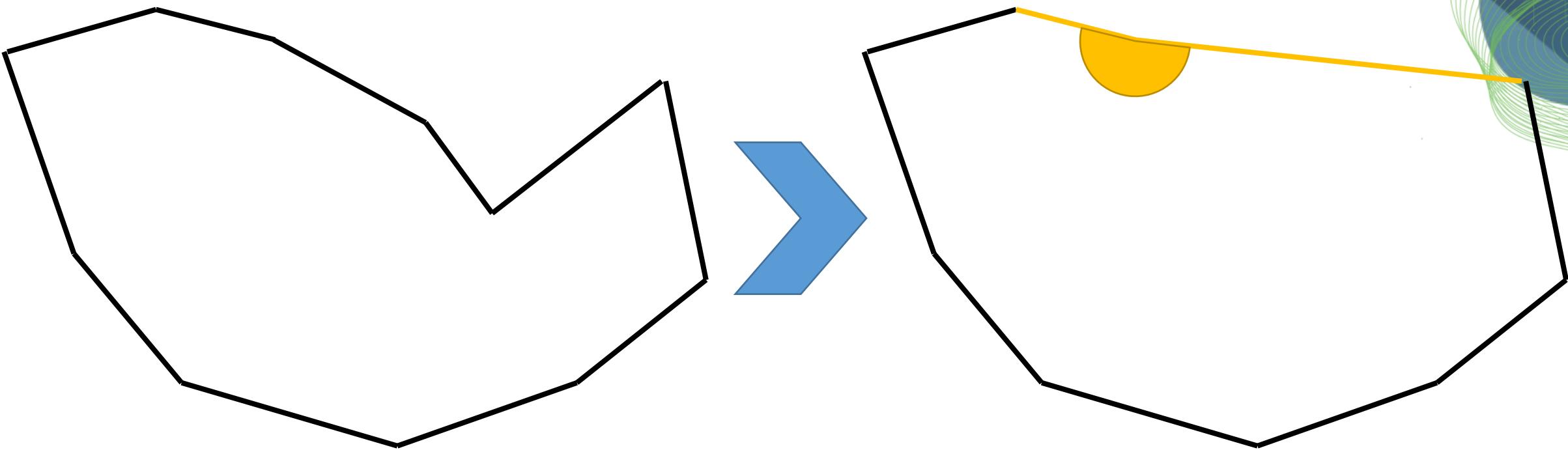
# Convex hull

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



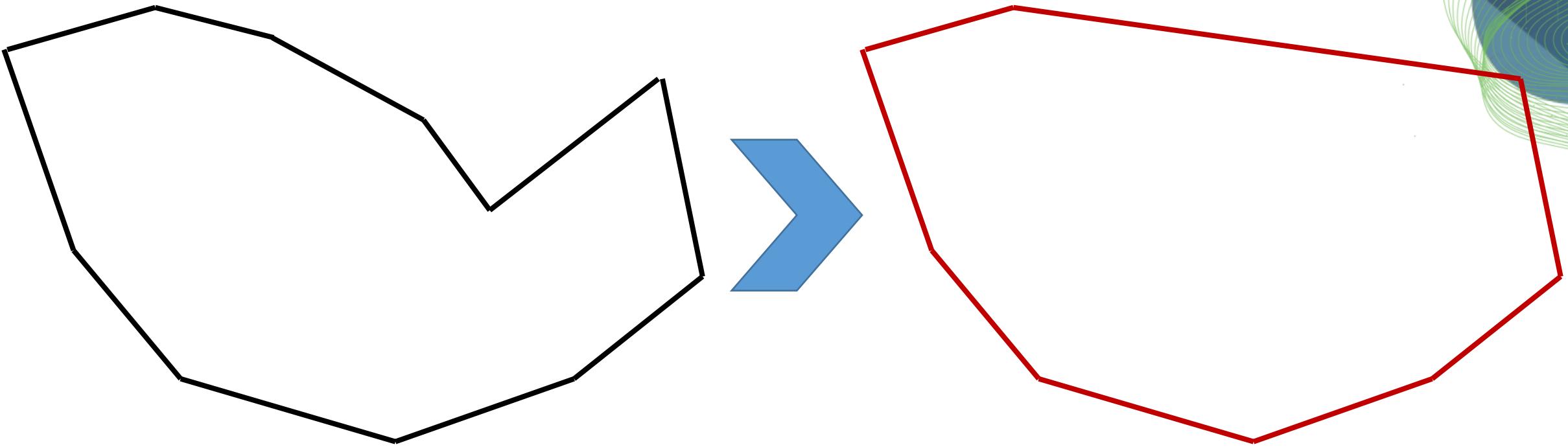
# Convex hull

- 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

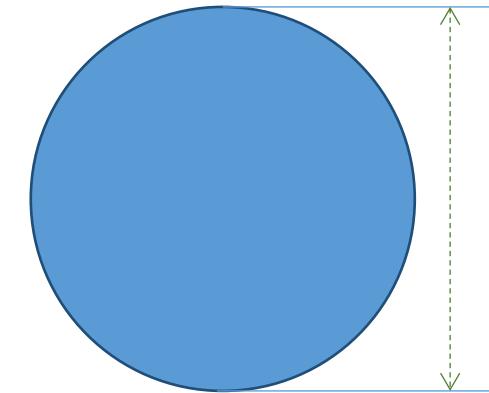
$d$

Circumference

$C = \pi d$

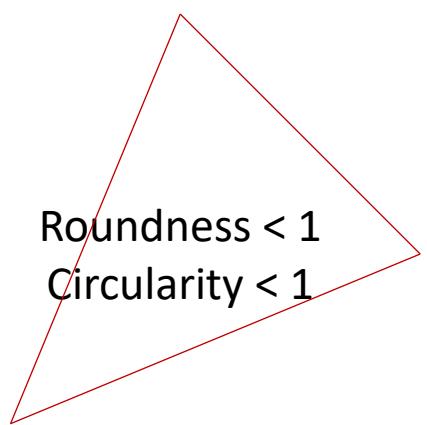
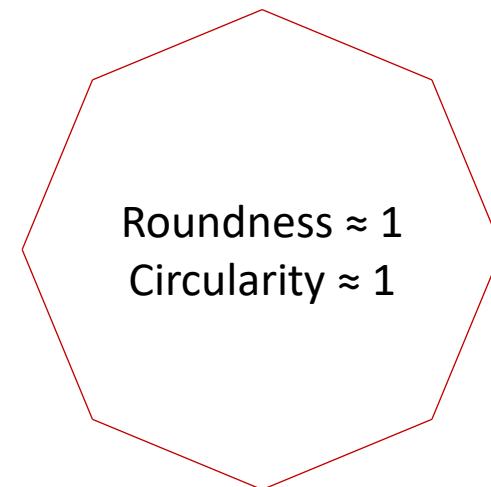
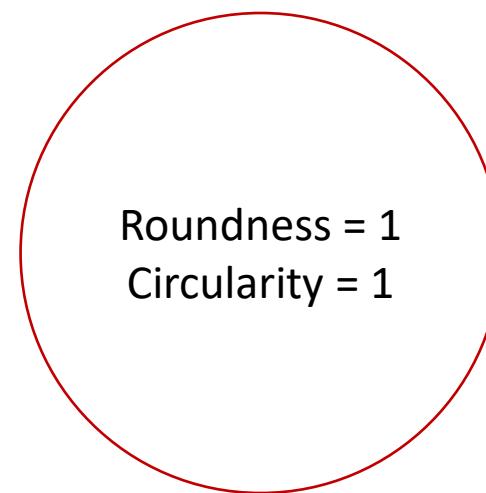
Area

$A = \frac{\pi d^2}{4}$



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

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



# Feature extraction in Python

- In Python: `from skimage import measure`

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

`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 encloses the region.

`area_filled` : int

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

`axis_major_length` : float

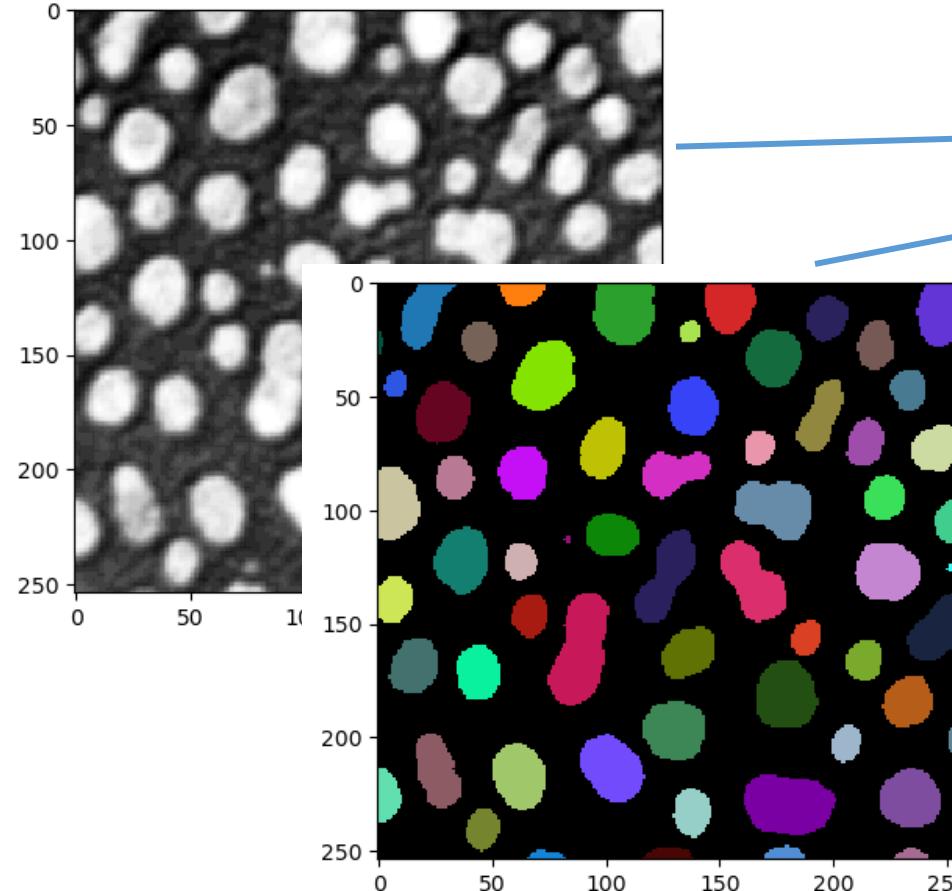
The length of the major axis of the ellipse that has the same normalized second central moments as the region.

`axis_minor_length` : float

The length of the minor axis of the ellipse that has the same normalized second central moments as the region.

# Feature extraction in Python

- The transition from image data to tabular data / pandas DataFrames



[4]: df = pd.DataFrame(regionprops\_table(label\_image,  
intensity\_image=image,  
properties=["label", "mean\_intensity", "area",  
"major\_axis\_length", "minor\_axis\_length"]))

[4]:

	label	mean_intensity	area	perimeter	major_axis_length	minor_axis_length
0	1	191.440559	429.0	89.012193	34.779230	16.654732
1	2	179.846995	183.0	53.556349	20.950530	11.755645
2	3	205.604863	658.0	95.698485	30.198484	28.282790
3	4	217.515012	433.0	77.455844	24.508791	23.079220
4	5	213.033898	472.0	83.798990	31.084766	19.681190
...	...	...	...	...	...	...

# Feature extraction in Python

- The transition from image data to tabular data / pandas DataFrames

```
[4]: df = pd.DataFrame(regionprops_table(label_image,
                                         intensity_image=image,
                                         properties=["label", "mean_intensity", "area", "perimeter",
                                         "major_axis_length", "minor_axis_length"]))
df
```

```
[4]:    label  mean_intensity   area  perimeter  major_axis_length  minor_axis_length
  0      1        191.440559  429.0     89.012193           34.779230       16.654732
  1      2        179.846995  183.0     53.556349           20.950530       11.755645
  2      3        205.604863  658.0     95.698485           30.198484       28.282790
  3      4        217.515012  433.0     77.455844           24.508791       23.079220
  4      5        213.033898  472.0     83.798990           31.084766       19.681190
  ...     ...

```

# Feature extraction in Python

- Customized features passed as function(s).

```
[5]: def standard_deviation_intensity(region, intensities):
    return np.std(intensities[region])

df = pd.DataFrame(regionprops_table(label_image,
                                      intensity_image=image,
                                      properties=["label", "mean_intensity", "area", "perimeter",
                                                   "major_axis_length", "minor_axis_length"],
                                      extra_properties=[standard_deviation_intensity]))
df
```

```
[5]:   label  mean_intensity  area  perimeter  major_axis_length  minor_axis_length  standard_deviation_intensity
  0      1        191.440559  429.0     89.012193          34.779230           16.654732                29.7931
  1      2        179.846995  183.0     53.556349          20.950530           11.755645                21.2705
  2      3        205.604863  658.0     95.698485          30.198484           28.282790                29.3922
  3      4        217.515012  433.0     77.455844          24.508791           23.079220                35.8523
  4      5        213.033898  472.0     83.798990          31.084766           19.681190                28.7410
  ...    ...            ...    ...          ...             ...              ...                  ...

```

# Feature extraction in Python

- Customized features computed afterwards.

Quiz: Why didn't we compute standard deviation of the intensity like this?

```
[6]: df['roundness'] = 4 * df['area'] / np.pi / pow(df['major_axis_length'], 2)
df['circularity'] = 4 * np.pi * df['area'] / pow(df['perimeter'], 2)
```

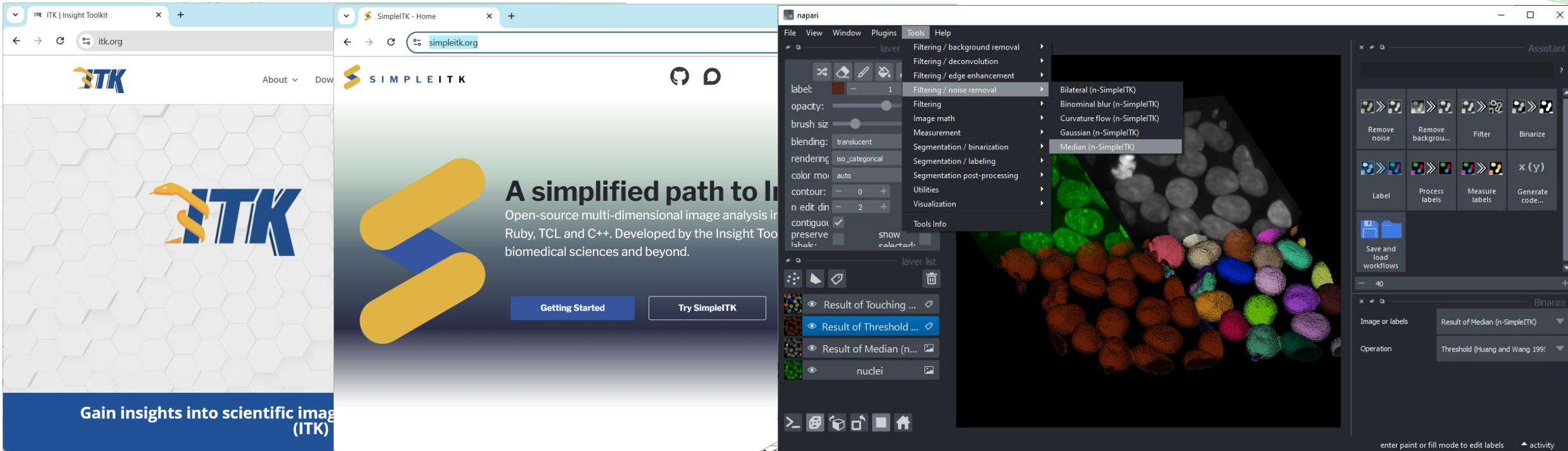
```
df
```

```
[6]:
```

	label	mean_intensity	area	perimeter	major_axis_length	minor_axis_length	standard_deviation_intensity	roundness	circularity
0	1	191.440559	429.0	89.012193	34.779230	16.654732	29.793138	0.451572	0.680406
1	2	179.846995	183.0	53.556349	20.950530	11.755645	21.270534	0.530849	0.801750
2	3	205.604863	658.0	95.698485	30.198484	28.282790	29.392255	0.918683	0.902871
3	4	217.515012	433.0	77.455844	24.508791	23.079220	35.852345	0.917813	0.906963
4	5	213.033898	472.0	83.798990	31.084766	19.681190	28.741080	0.621952	0.844645
...	...	...	...	...	...	...	...	...	...

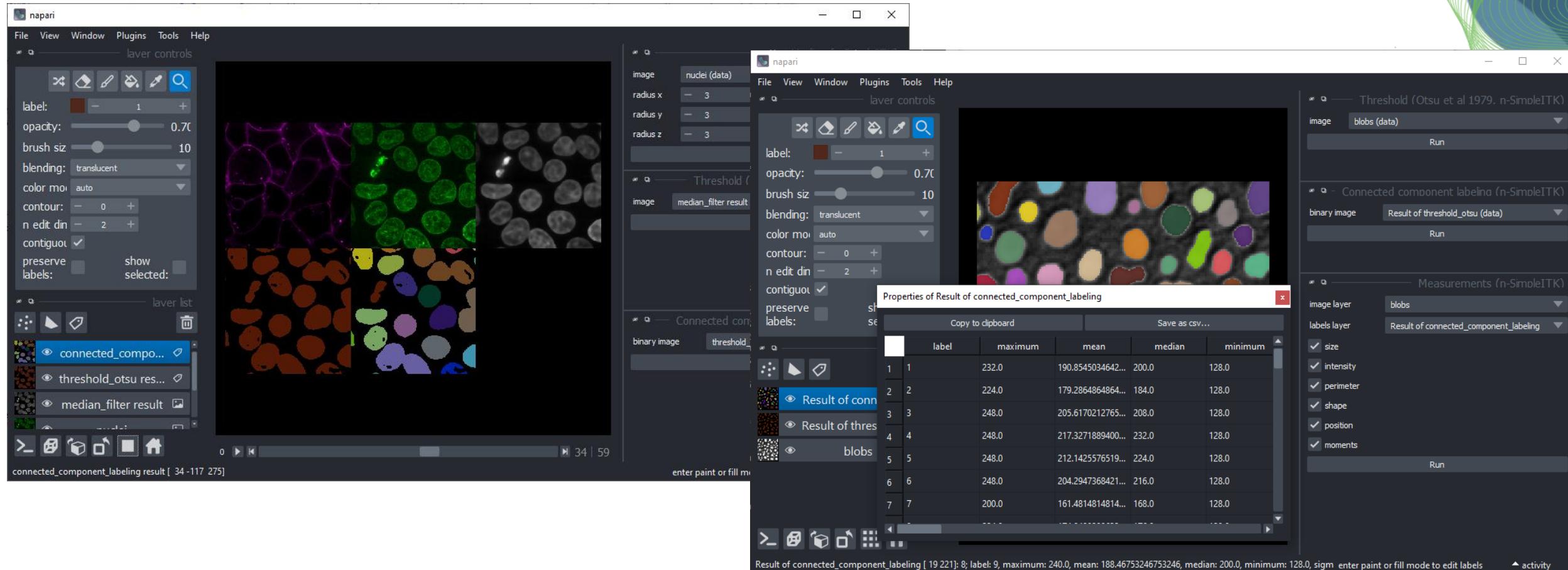
# SimpleITK

- ITK: Insight Toolkit, a [medical] image processing library, written in C++, originating in the 80s.
- SimpleITK: A Python wrapper around ITK
- Napari-simpleitk-image-processing: A Napari Plugin and simplification wrapper around simple-itk.



# SimpleITK in Napari

- Menu Tools > Measurement Tables > Measurements (n-SimpleITK)



# SimpleITK

- The napari-plugin for creating tables can also be called from Python.
- **Recommended for working with 3D data.**

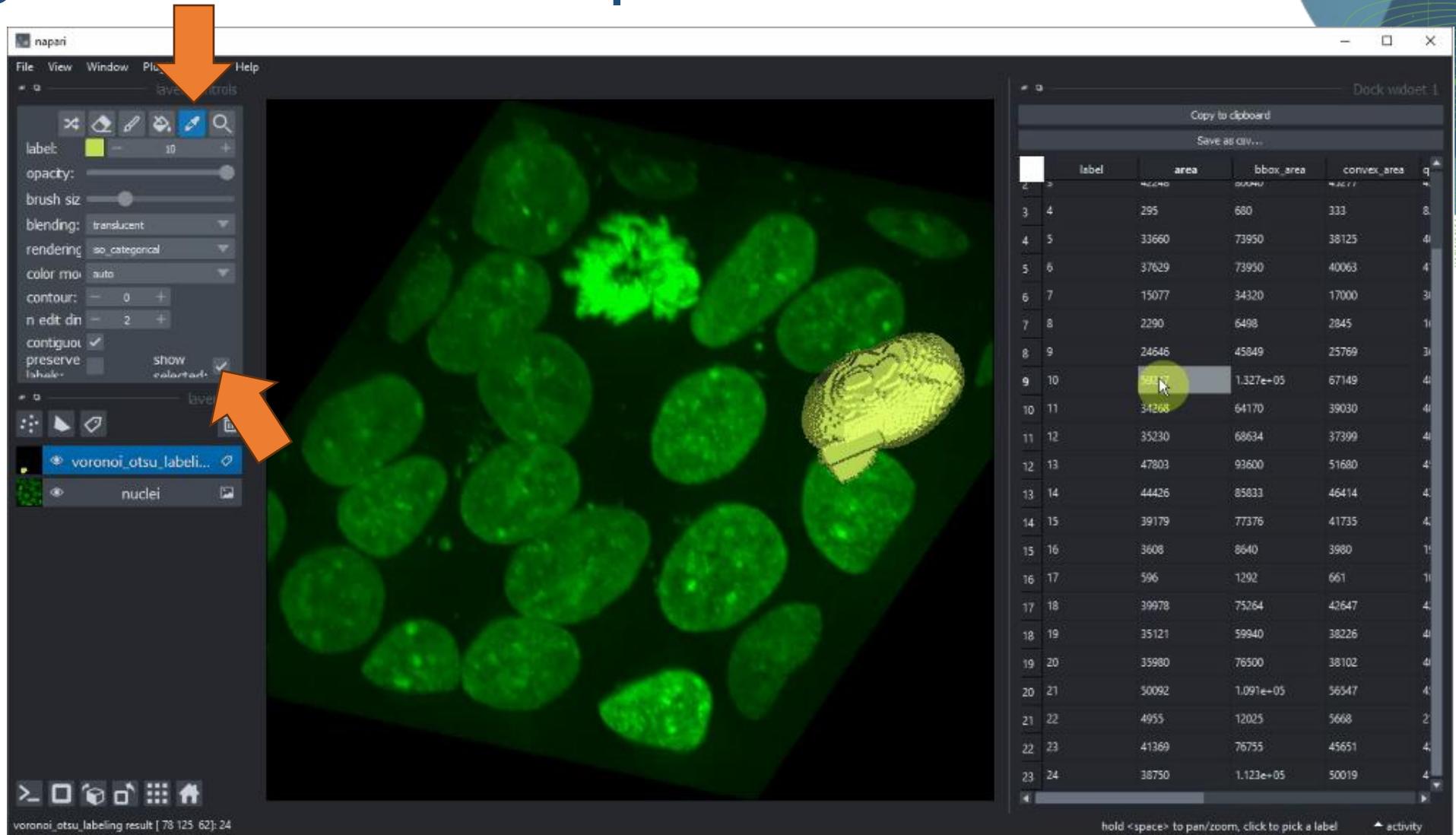
```
statistics = label_statistics(blobs, labels,
                             intensity=True,
                             size=True,
                             shape=True,
                             perimeter=True,
                             position=True,
                             moments=True)

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

	label	maximum	mean	median	minimum	sigma	sum	variance	bbox_0	bbox_1
0	1	224.0	137.526132	136.0	112.0	13.360739	157880.0	178.509343	0	0
1	2	232.0	193.014354	200.0	128.0	28.559077	80680.0	815.620897	11	0
2	3	224.0	179.846995	184.0	128.0	21.328889	32912.0	454.921516	53	0

# Exploring features in Napari

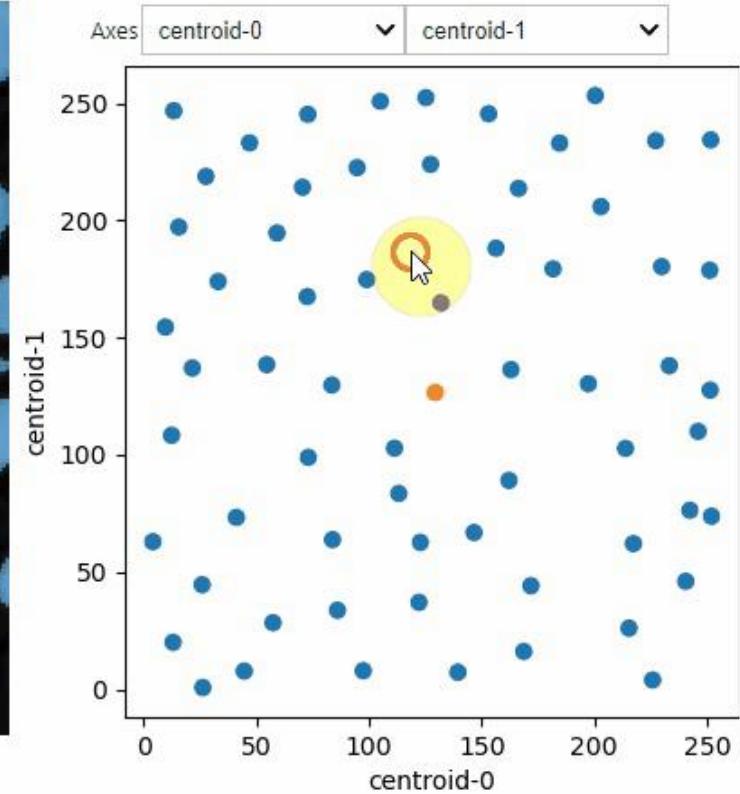
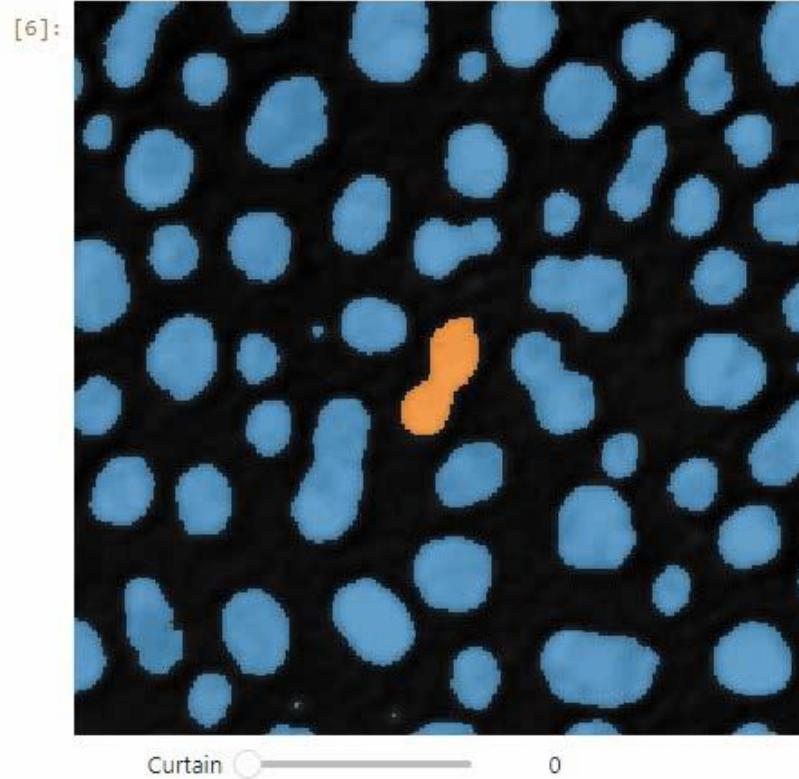
- Select table rows and view corresponding object in 2D/3D space



# Selecting objects according to their properties

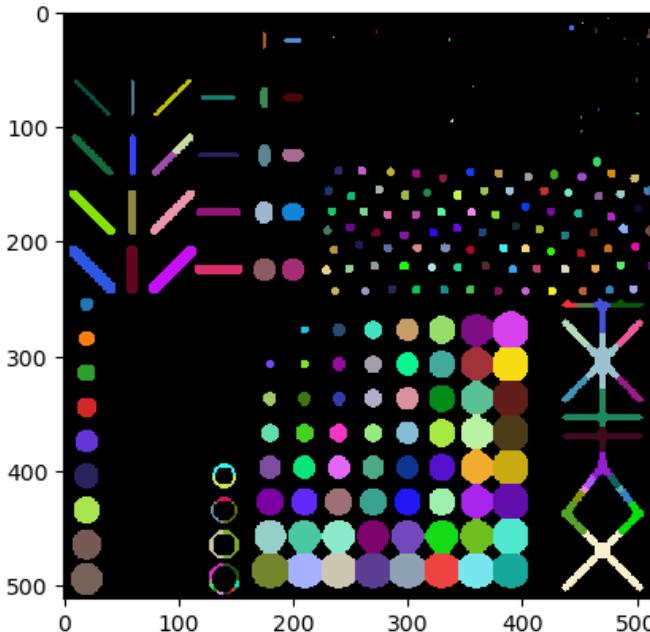
- Understanding what features *mean* may require interactive user interfaces

```
[6]: stackview.clusterplot(image=image,
                           labels=labeled_image,
                           df=df,
                           column_x="area",
                           column_y="aspect_ratio",
                           zoom_factor=1.6,
                           alpha=0.7)
```

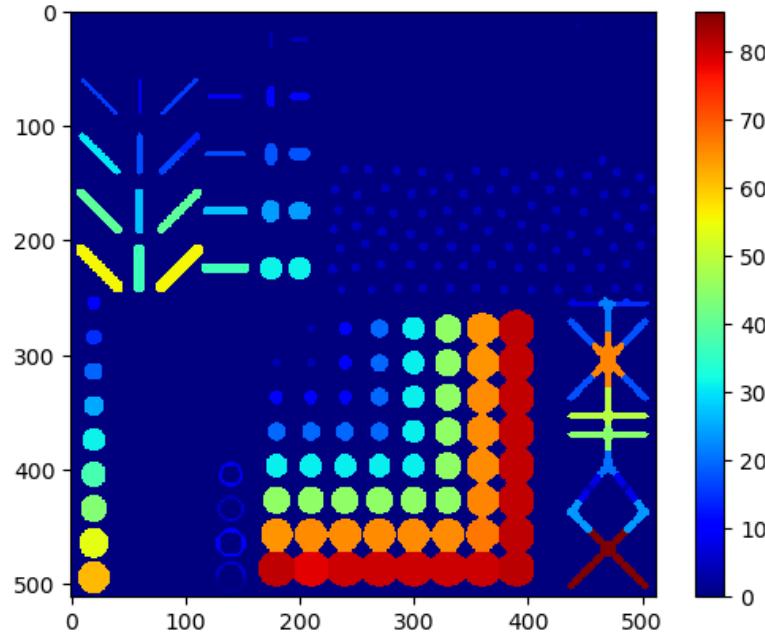


# Parametric images

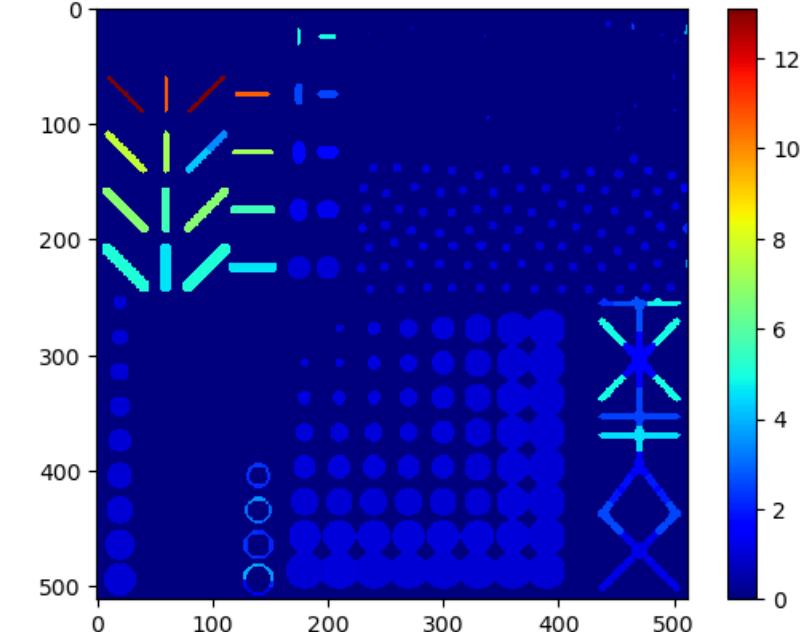
- Visualizing quantitative measurements in image space.



Label image



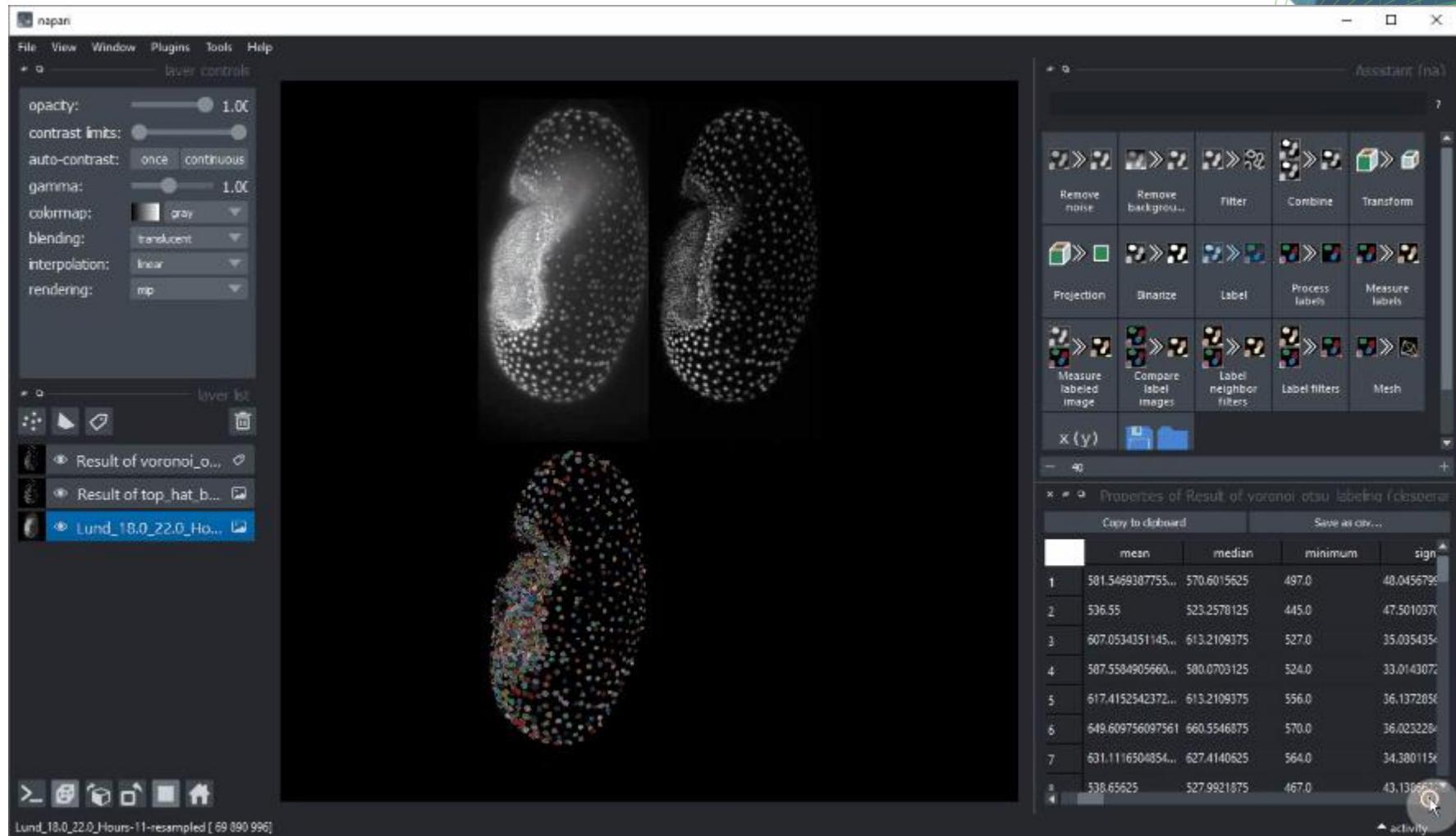
Pixel count image



Aspect ratio image

# Exploring features in Napari

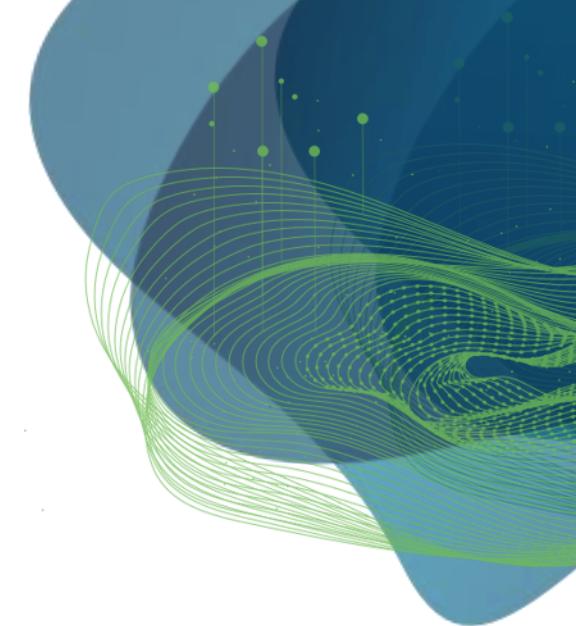
- Double-click on table column to retrieve a parametric map image





DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS  
AND ARTIFICIAL INTELLIGENCE



# Exercises

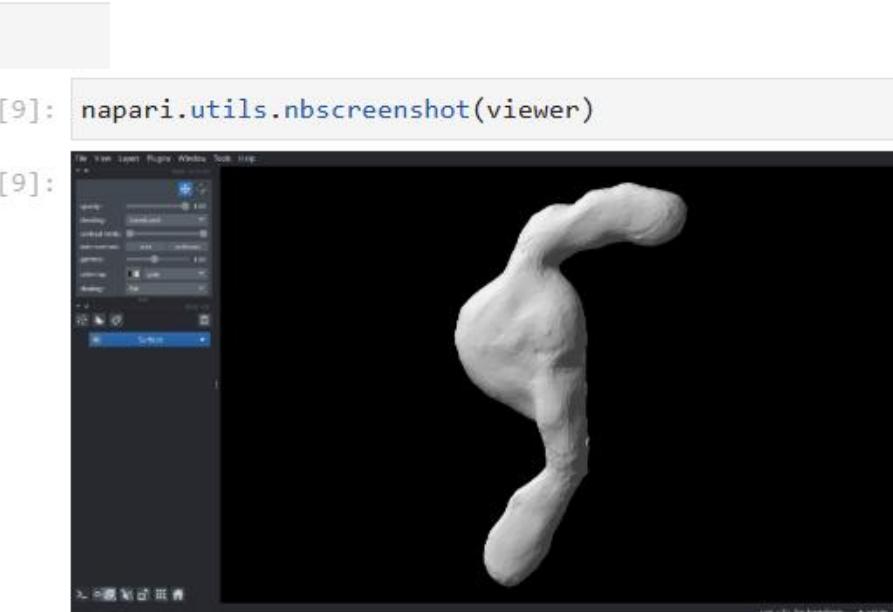
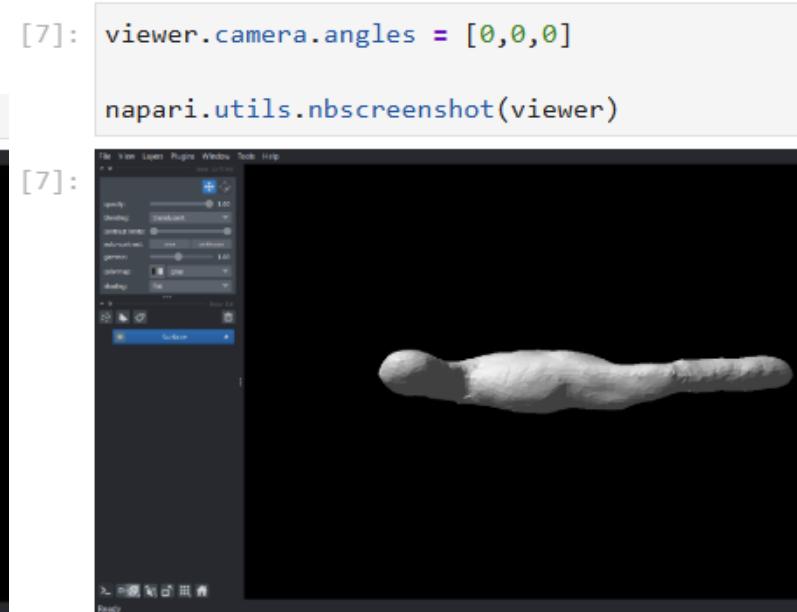
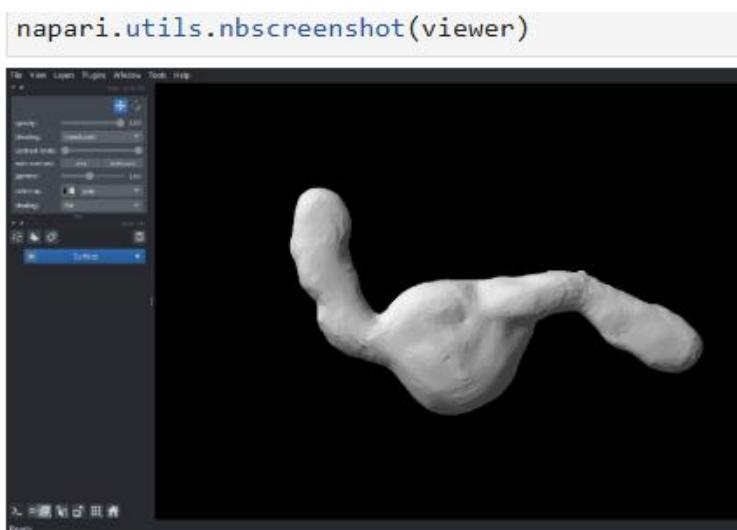
Robert Haase

GEFÖRDERT VOM



# Exercise: Surface meshes

- Creating, storing, processing surface mesh data
- Task: Reproduce a specific view in Napari.
- Challenge: Try to code the view-adaptation in Napari in one minute or less.



# Exercise: Quantitative measurements

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

```
[5]: df = pd.DataFrame(regionprops_table(image , label_image,
                                         perimeter = True,
                                         shape = True,
                                         position=True,
                                         moments=True))

df
```

	label	area	bbox_area	equivalent_diameter	convex_area	max_intensity	mean_intensity	min_intensity	perimeter	perimeter
0	1	429.0	750.0	23.371345	479.0	232.0	191.440559	128.0	89.012193	89.012193
1	2	183.0	231.0	15.264430	190.0	224.0	179.846995	128.0	53.556349	53.556349
2	3	658.0	756.0	28.944630	673.0	248.0	205.604863	120.0	95.698485	95.698485
3	4	433.0	529.0	23.480049	445.0	248.0	217.515012	120.0	77.455844	77.455844
4	5	472.0	551.0	24.514670	486.0	248.0	213.033898	128.0	83.798990	83.798990
...	...	...	...	...	...	...	...	...	...	...
57	58	213.0	285.0	16.468152	221.0	224.0	184.525822	120.0	52.284271	52.284271
58	59	79.0	108.0	10.029253	84.0	248.0	184.810127	128.0	39.313708	39.313708
59	60	88.0	110.0	10.585135	92.0	216.0	182.727273	128.0	45.692388	45.692388
60	61	52.0	75.0	8.136858	56.0	248.0	189.538462	128.0	30.692388	30.692388
61	62	48.0	68.0	7.817640	53.0	224.0	173.833333	128.0	33.071068	33.071068

62 rows × 86 columns

## Exercises

Make a table with only `area`, `mean_intensity`, `standard_deviation_intensity` and `label`.

[ ]:

How many objects are in the dataframe?

[ ]:

How large is the largest object?

[ ]:

What is the mean intensity of the brightest object?

[ ]:

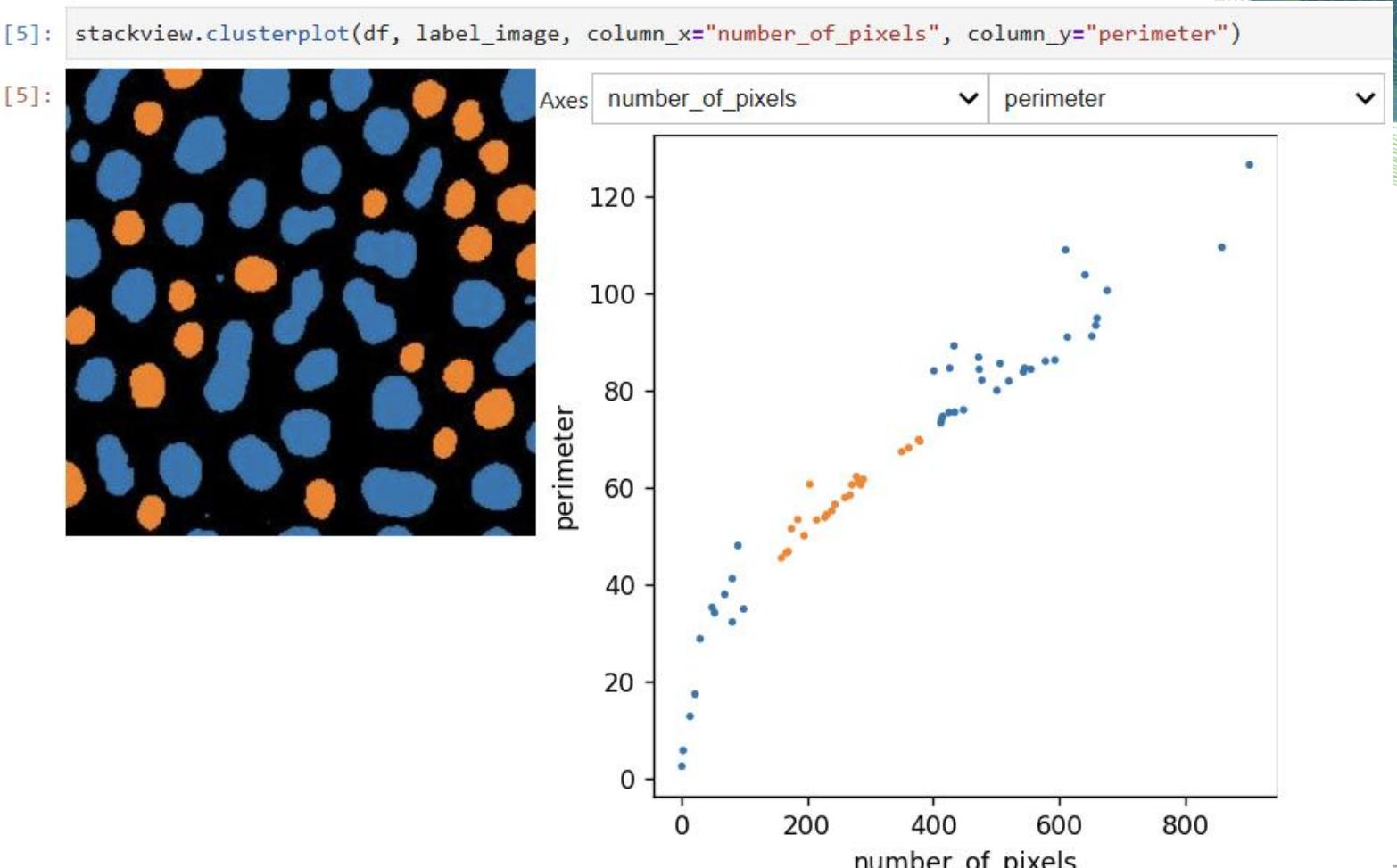
What are mean and standard deviation intensity of the image?

[ ]:



# Exercise: Parametric maps

- Interpreting parameters: What's the difference between elongation and flatness?
- Interactive tools might help.





DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS  
AND ARTIFICIAL INTELLIGENCE

# Complex exercise

Robert Haase

GEFÖRDERT VOM



Bundesministerium  
für Bildung  
und Forschung



Diese Maßnahme wird gefördert durch die Bundesregierung  
aufgrund eines Beschlusses des Deutschen Bundestages.  
Diese Maßnahme wird mitfinanziert durch Steuermittel auf  
der Grundlage des von den Abgeordneten des Sächsischen  
Landtags beschlossenen Haushaltes.



# Complex exercise

- Scenario: Imagine a biologist sent you some data (images + maybe corresponding label image). They ask you to write an image-analysis workflow for processing these images [+ more images of similar kind].

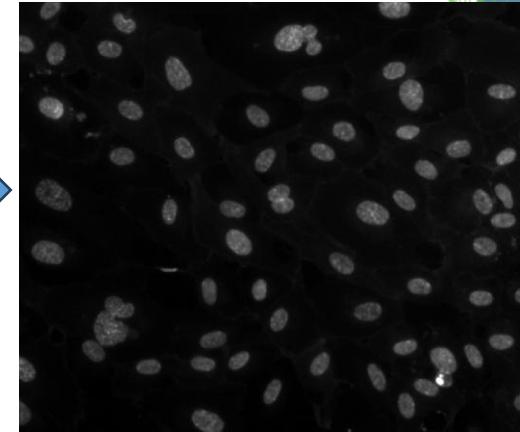
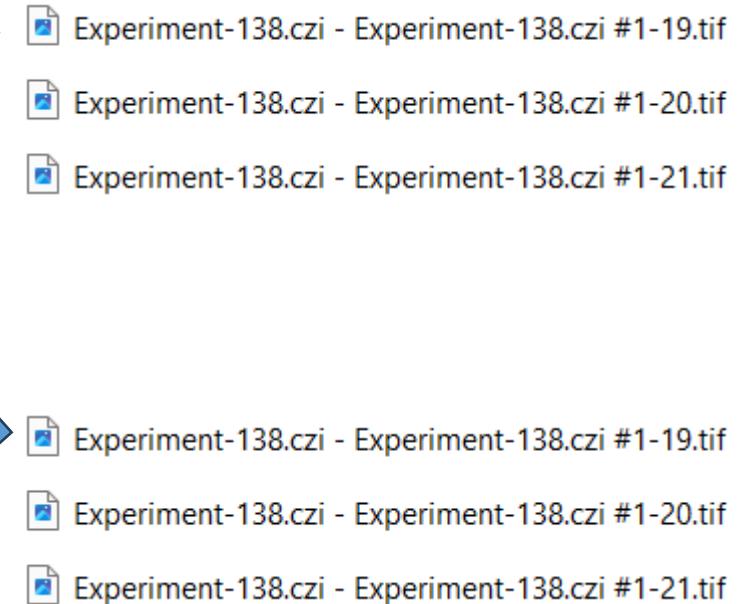
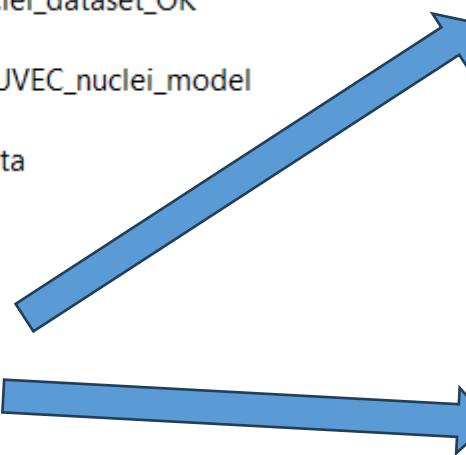
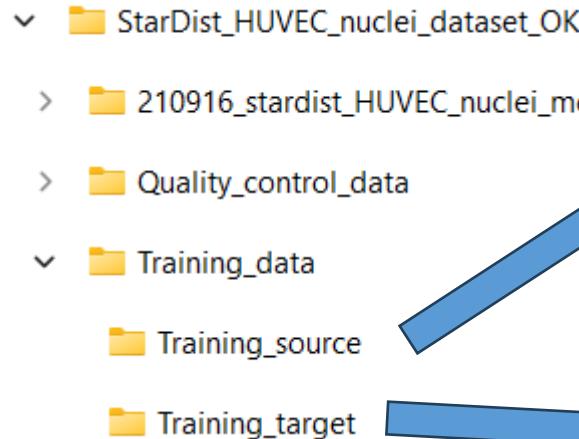
Screenshot of a web browser showing a GitHub repository page for 'ScaDS/BIDS-lecture-2025'. The page displays a README file and a CC-BY-4.0 license. Below these, a section titled 'Assignments' contains text explaining the assignment requirements and a list of student IDs.

As explained in the lecture on May 9th, every enrolled student has to submit an image analysis workflow + documentation. The workflows are intended to process different datasets. Each student gets their own dataset. Download the zipfile corresponding to your MatNr and unzip it. The password was given to you in the lecture on May 9th. [3795650](#) [3781296](#) [3799815](#) [3745356](#) [3711464](#) [3781088](#) [3782680](#) [3769826](#) [3790403](#) [3746612](#) [3787475](#) [3706879](#) [3792009](#) [3796977](#) [3796590](#) [3795611](#) [3795409](#) [3792423](#) [3771050](#) [2163012](#) [3777628](#) [3780527](#) [3765110](#) [3720384](#) [3781461](#) [3763803](#) [3795674](#) [3781887](#) [3733406](#) [3770023](#) [3750893](#) [3761687](#)



# Complex exercise

- Unzip your zip-file and download the data. Example:



# Complex exercise

- Scientific tasks
  - Develop an image-segmentation workflow, which reproduces label images (doesn't need to be perfect)
  - Extract features from these images, at least area/volume of objects + 1 more feature
  - Plot area [or volume] against another feature.
  - Visualize parametric image showing area / volume of the objects.

# Complex exercise

- Engineering tasks
  - Setup a software environment
  - Setup an image processing workflow
  - Setup a data analysis / visualization workflow
  - Setup a quality assurance procedure
- Documentation tasks
  - Installation instructions
  - User guide
  - Documentation of used data
  - Explanation of the used algorithms



Act as if you would communicate with a biologist, with limited image-analysis, conda, and programming skills.

# Complex exercise

- Submission
  - Submit a password-protected ZIP file to [robert.haase@uni-leipzig.de](mailto:robert.haase@uni-leipzig.de) (Why password protected: The virus scanner cannot reject python files in encrypted zip-files)
  - Allowed file formats: ipynb, py, docx, pdf, md, csv, yml, json, xml, txt, tif
  - **Deadline: July 4<sup>th</sup> 2025**
- Hint
  - Send this ZIP file to a friend and ask them to run the analysis. If they can follow your instructions successfully, without communicating with you, proceed to final submission.

# Complex exercise

- Checklist
  - The software environment is reproducible
  - There are instructions how to run the code.
  - There is example data in the zip file. Use a small fraction of the entire data (< 10 MB)  
(note: you cannot submit a 500MB ZIP file via email)
  - The code is well documented / commented
  - Segmentation results are visualized
  - There is code that stores segmentation results, e.g. as label image .tif file
  - The quality of the segmentation result is measured
  - Used algorithms are cited, and explained in some sentences
  - There is code that stores extracted features / measurements as CSV-file
  - One could associate rows in this CSV file with segmented objects in saved segmentation results.
  - Resulting plots and visualizations have reasonable axis labels and are well explained
  - The copyright of re-used data and code are respected

# Complex exercise

- Option A: You do the analysis yourself. Minor question-answer interactions with ChatGPT etc. are allowed.
- Option B: You use a Large Language Model to do the whole job.  
If you choose this option, you need to submit all prompts you used.