

Distributed & GPU-accelerated Image Processing

Robert Haase

Funded by



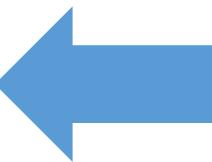
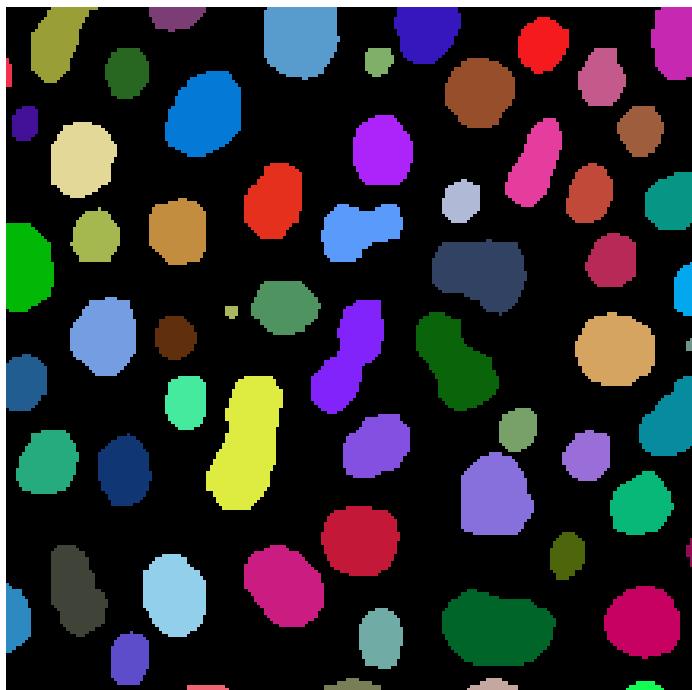
Bundesministerium
für Bildung
und Forschung



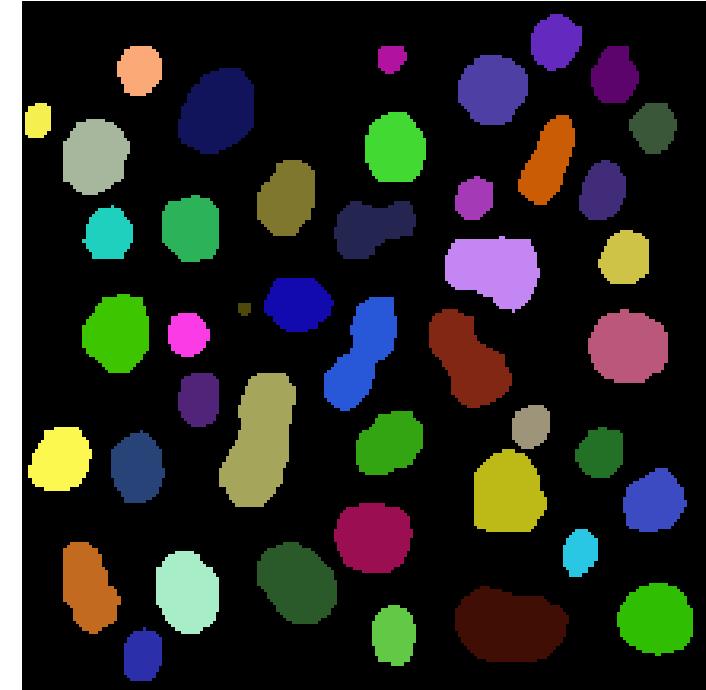
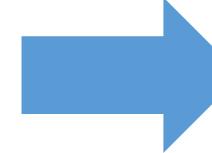
Diese Maßnahme wird gefördert durch die Bundesregierung
aufgrund eines Beschlusses des Deutschen Bundestages.
Diese Maßnahme wird mitfinanziert durch Steuermittel auf
der Grundlage des von den Abgeordneten des Sächsischen
Landtags beschlossenen Haushaltes.

Quiz:

- How can we estimate the number of cells in an image?

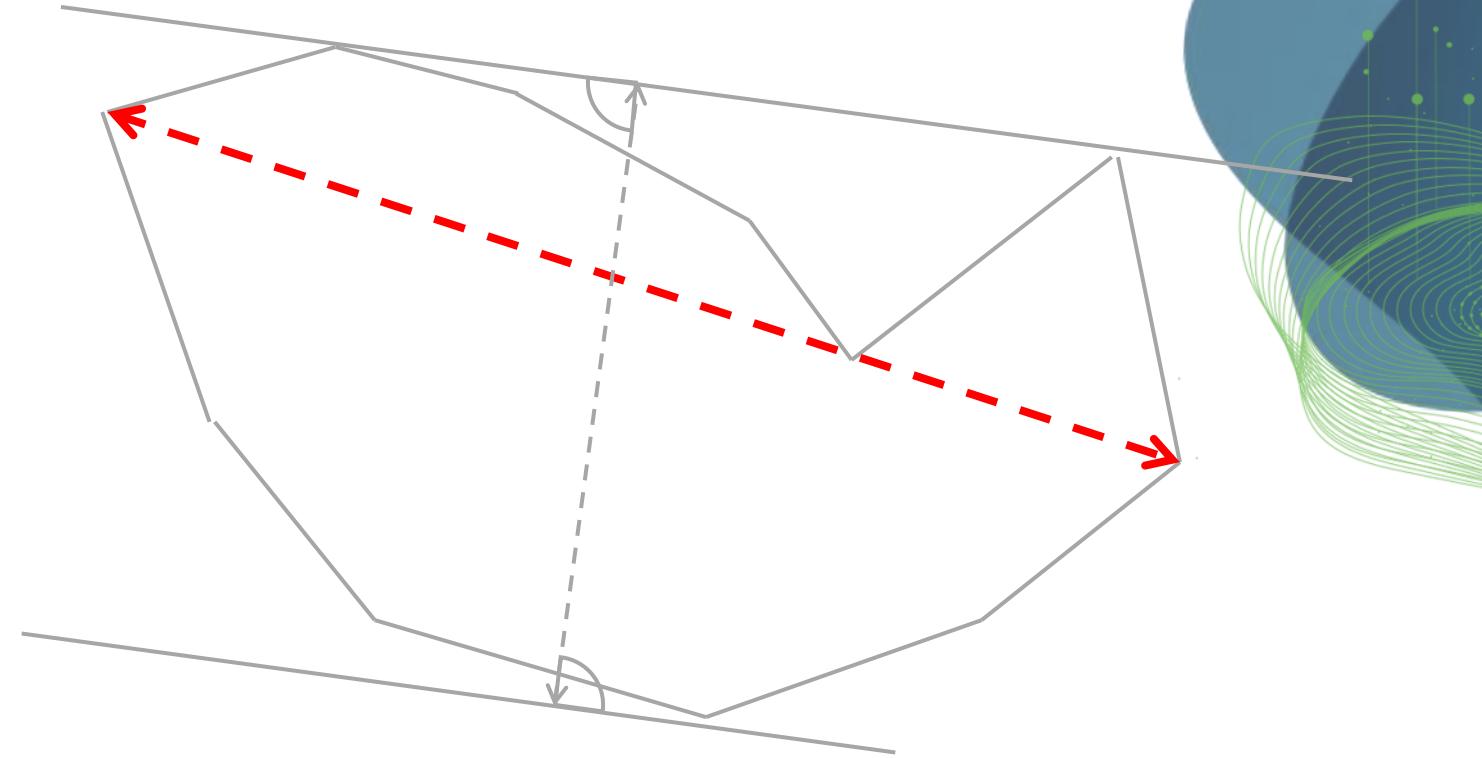


?



Quiz: Recap

How is this feature called?



Feret's diameter



Minimum Caliper



Minor Axis length



Major Axis length



Distributed & GPU-accelerated Image Processing

Robert Haase

Funded by



Bundesministerium
für Bildung
und Forschung

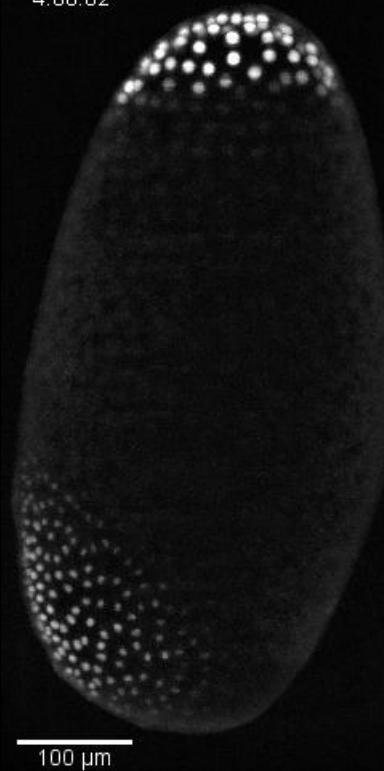


Diese Maßnahme wird gefördert durch die Bundesregierung
aufgrund eines Beschlusses des Deutschen Bundestages.
Diese Maßnahme wird mitfinanziert durch Steuermittel auf
der Grundlage des von den Abgeordneten des Sächsischen
Landtags beschlossenen Haushaltes.

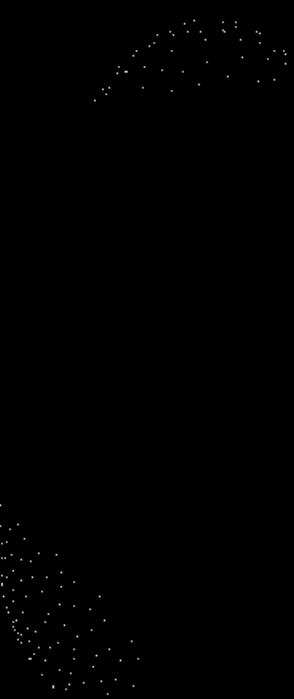
GPU accelerated image processing in life sciences

- ... to study embryo development

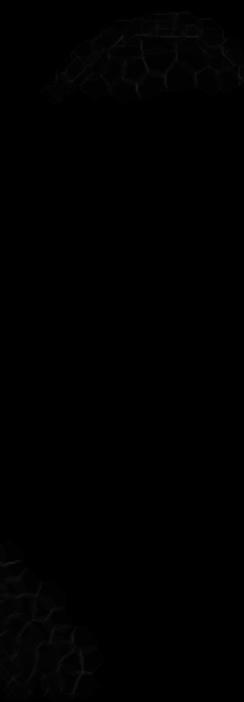
Tribolium castaneum nuclei-
GFP,
Background subtracted
4:00:02



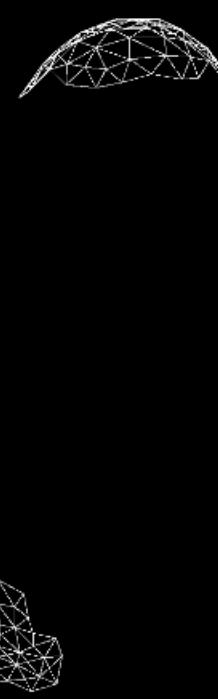
Spot detection (3D)



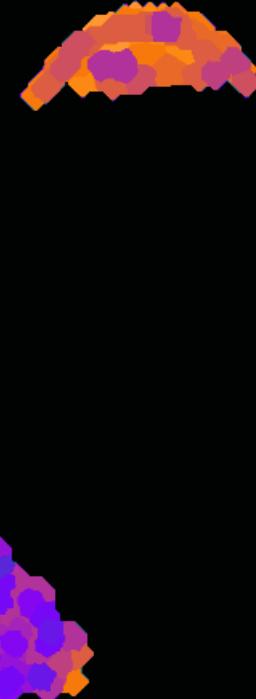
Theoretical membranes
(pseudo Voronoi map)



Neighbor mesh



Average centroid distance of
neighbors



0 μm 35

GPU accelerated image processing in life sciences

- Raytracing enables differentiating surface and sub-surface mesh nodes

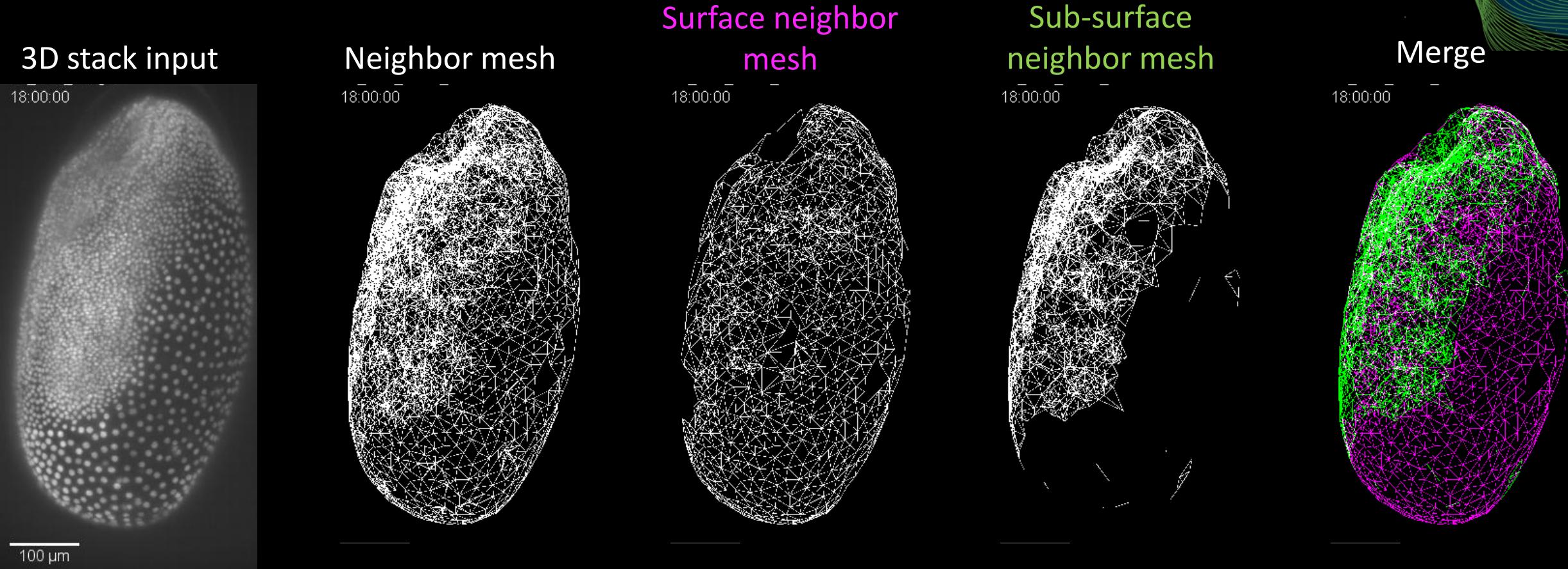


Image processing in life-sciences

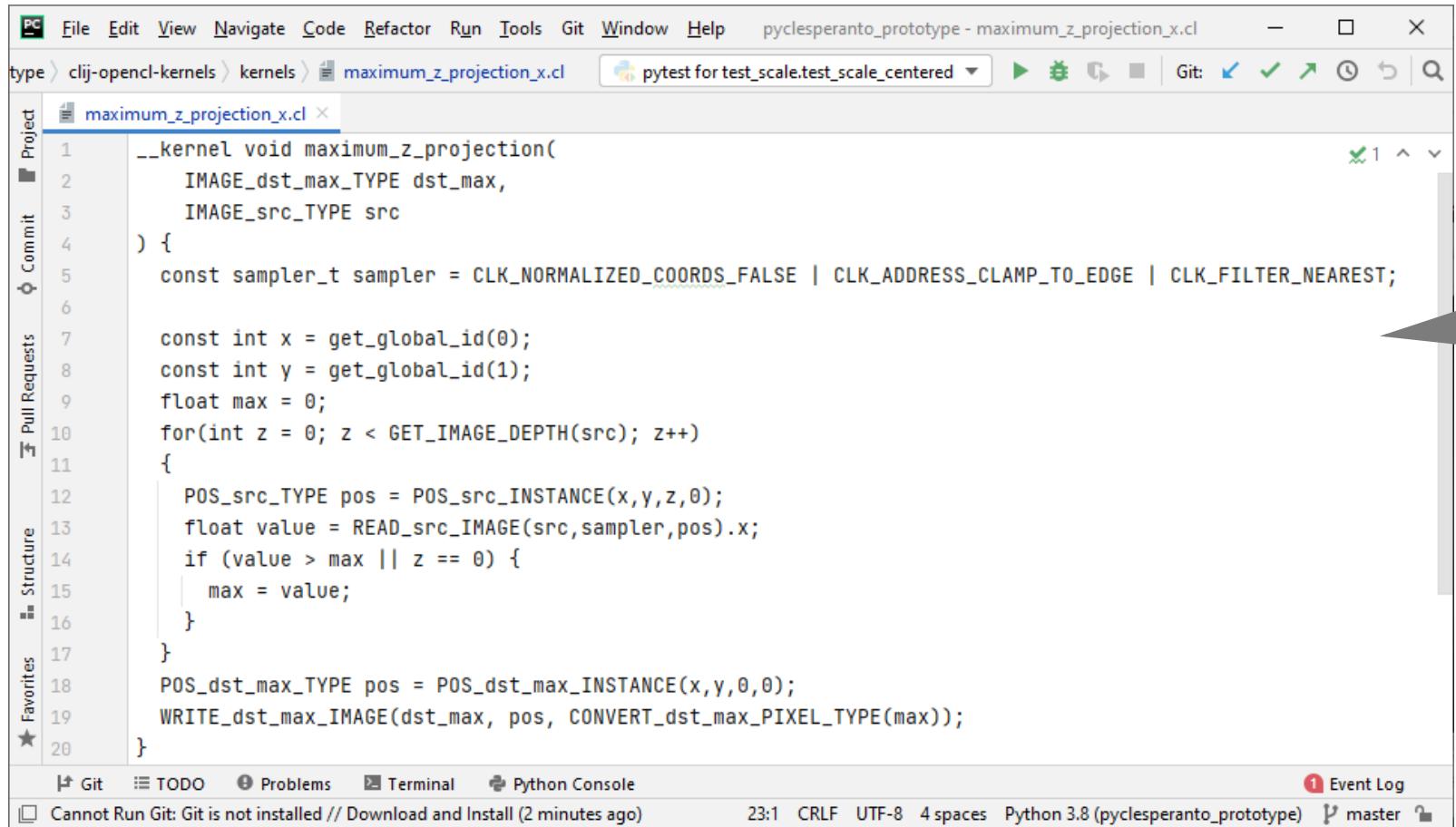
- State-of-the-art software for more than 20 years: ImageJ / Fiji



2x

OpenCL-based GPU-acceleration

- GPU-acceleration? Learn the Open Computing Language (OpenCL)!



The screenshot shows an IDE interface with the following details:

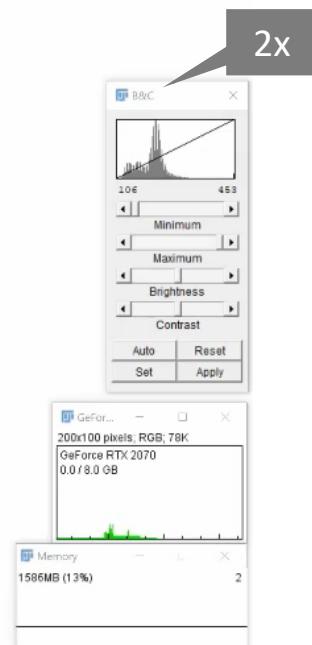
- Title Bar:** pyclesperanto_prototype - maximum_z_projection_x.cl
- Toolbar:** File, Edit, View, Navigate, Code, Refactor, Run, Tools, Git, Window, Help.
- Project Explorer:** Shows a project named "clij-opencl-kernels" with a file "maximum_z_projection_x.cl" selected.
- Code Editor:** Displays the following OpenCL kernel code:

```
1  __kernel void maximum_z_projection(
2      IMAGE_dst_max_TYPE dst_max,
3      IMAGE_src_TYPE src
4  ) {
5      const sampler_t sampler = CLK_NORMALIZED_COORDS_FALSE | CLK_ADDRESS_CLAMP_TO_EDGE | CLK_FILTER_NEAREST;
6
7      const int x = get_global_id(0);
8      const int y = get_global_id(1);
9      float max = 0;
10     for(int z = 0; z < GET_IMAGE_DEPTH(src); z++)
11     {
12         POS_src_TYPE pos = POS_src_INSTANCE(x,y,z,0);
13         float value = READ_src_IMAGE(src,sampler,pos).x;
14         if (value > max || z == 0) {
15             max = value;
16         }
17     }
18     POS_dst_max_TYPE pos = POS_dst_max_INSTANCE(x,y,0,0);
19     WRITE_dst_max_IMAGE(dst_max, pos, CONVERT_dst_max_PIXEL_TYPE(max));
20 }
```

- Bottom Status Bar:** Git: (with icons), Event Log (1 event), Python 3.8 (pyclesperanto_prototype), master branch, 23:1 CRLF, UTF-8, 4 spaces.
- Message Bar:** Cannot Run Git: Git is not installed // Download and Install (2 minutes ago)

Maximum
intensity
projection
along Z

User-friendly GPU-acceleration



GPU-accelerated image processing

- Performance depends on operation, image size, parameters, hardware,

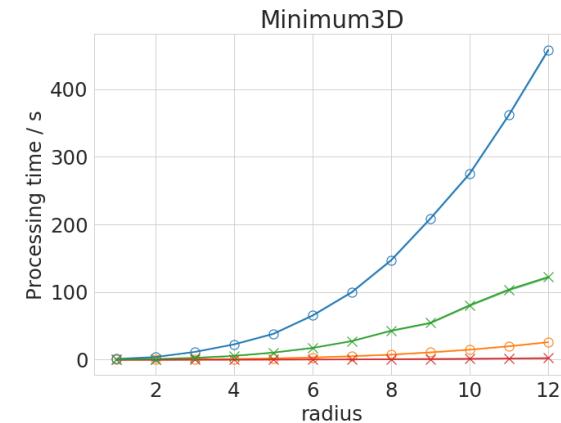
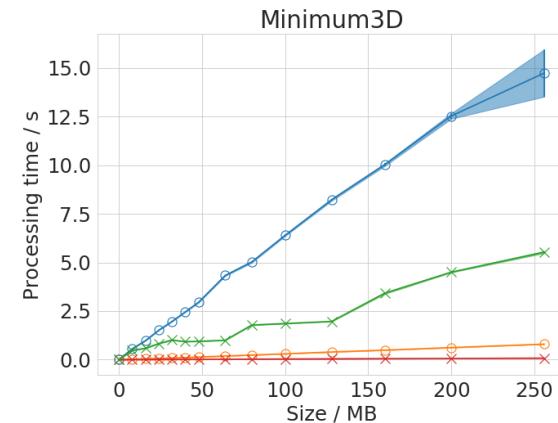
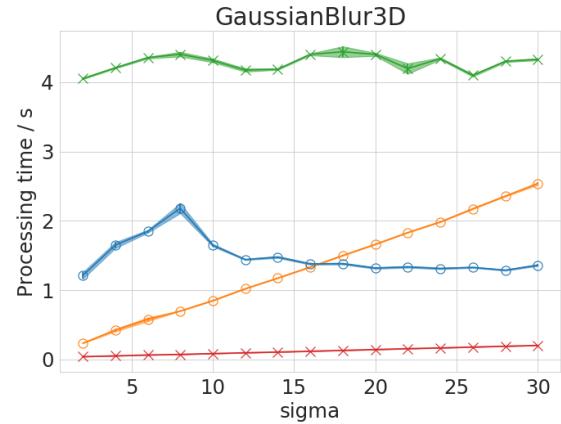
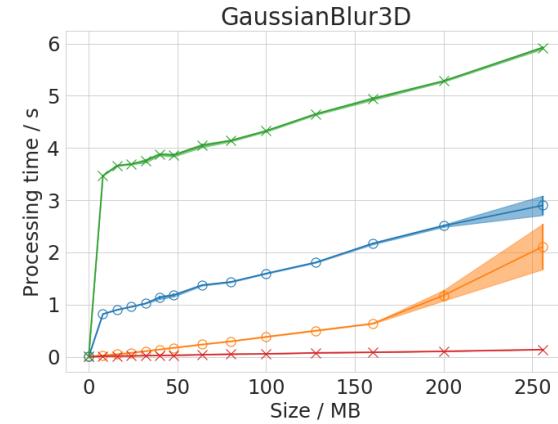


vs.



Workstation CPU
2x Intel Xeon Silver
4110
Laptop CPU
Intel Core i7-8650U

Workstation GPU
Nvidia Quadro
P6000
Laptop GPU
Intel UHD 620 GPU



GPU-acceleration only makes sense if images are large

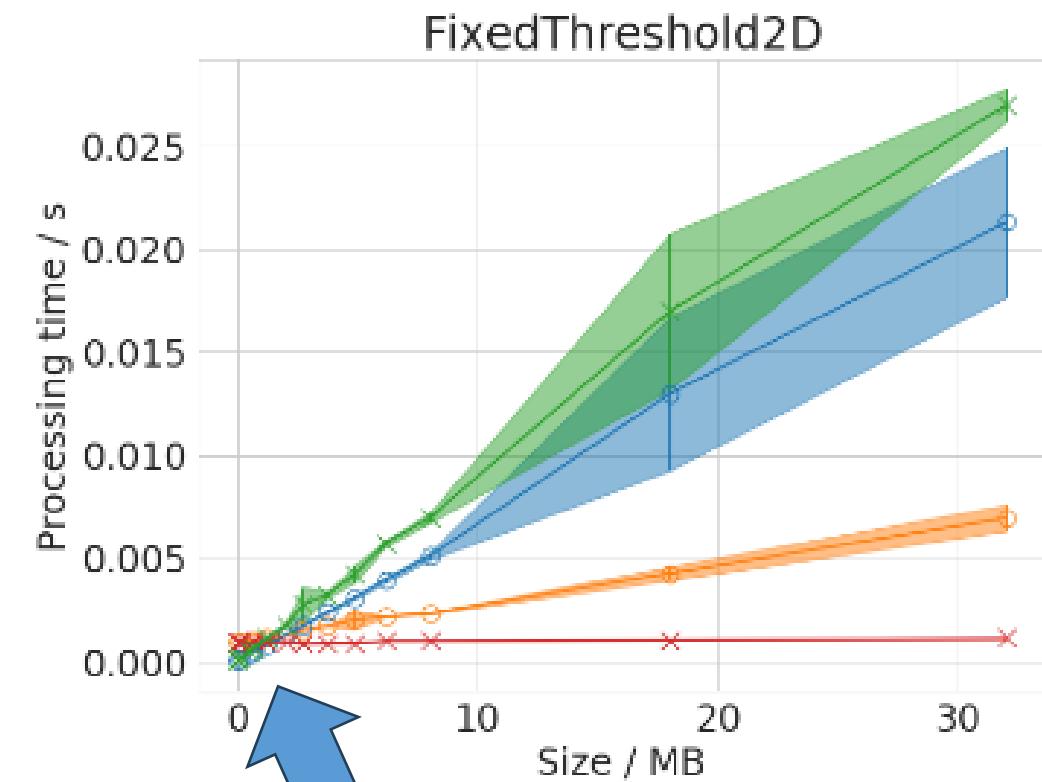
- Because the CPU cache is faster than GPU-memory.
- CPU cache is also very small though.
- Example:

Intel Core i7-8650u*

Cache	
Cache L1:	64 KB (per core)
Cache L2:	256 KB (per core)
Cache L3:	8 MB (shared)

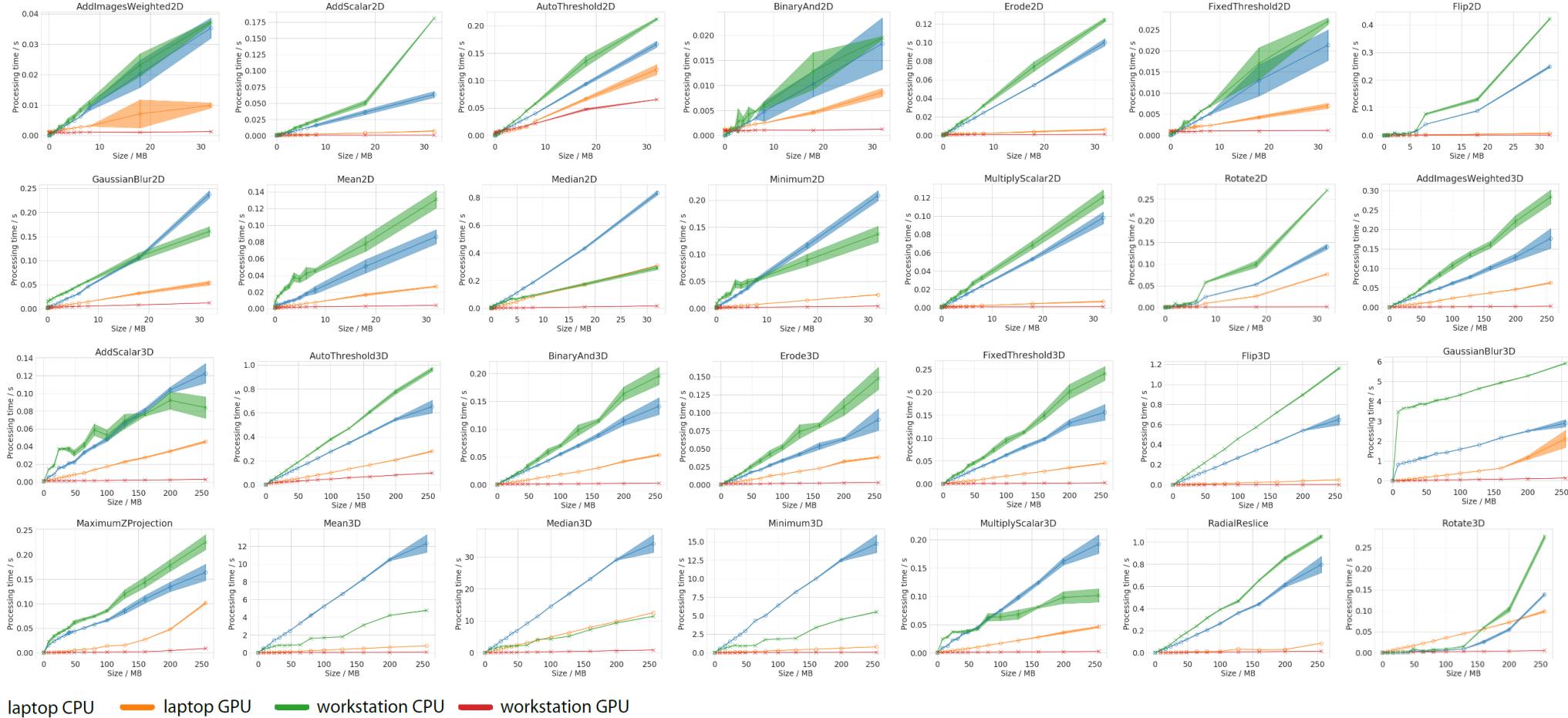
Workstation CPU
2x Intel Xeon Silver
4110
Laptop CPU
Intel Core i7-8650U

Workstation GPU
Nvidia Quadro
P6000
Laptop GPU
Intel UHD 620 GPU

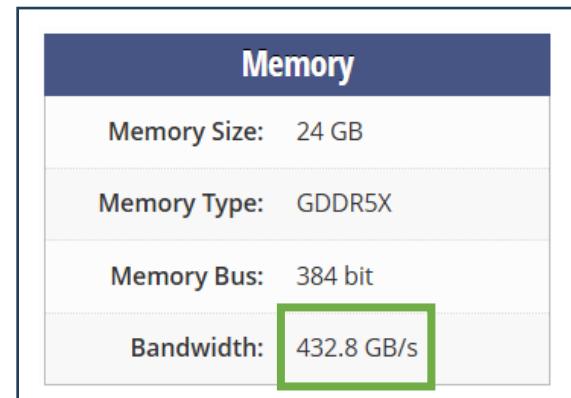
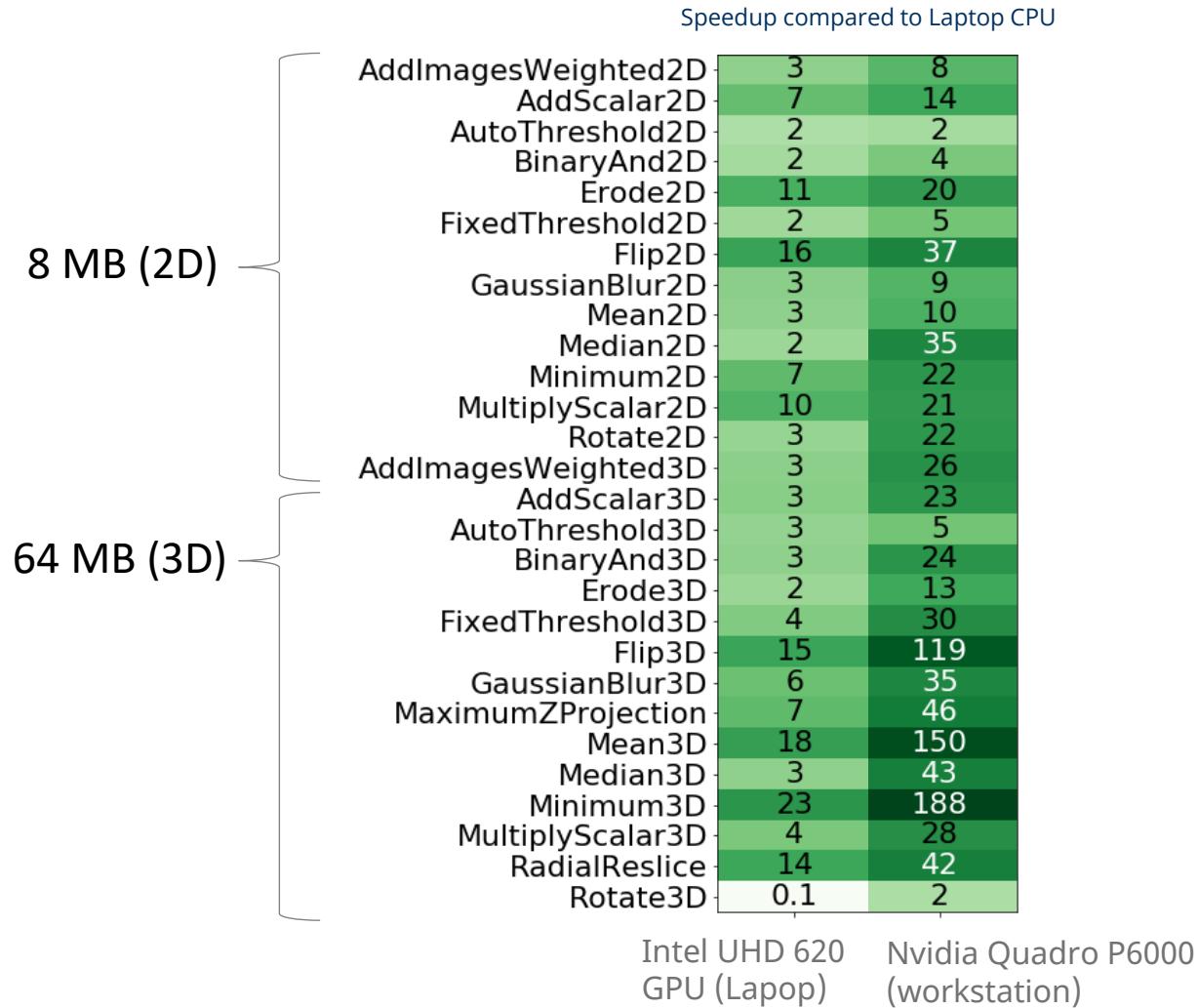


GPU-accelerated image processing

- Performance depends on operation, image size, parameters, hardware,



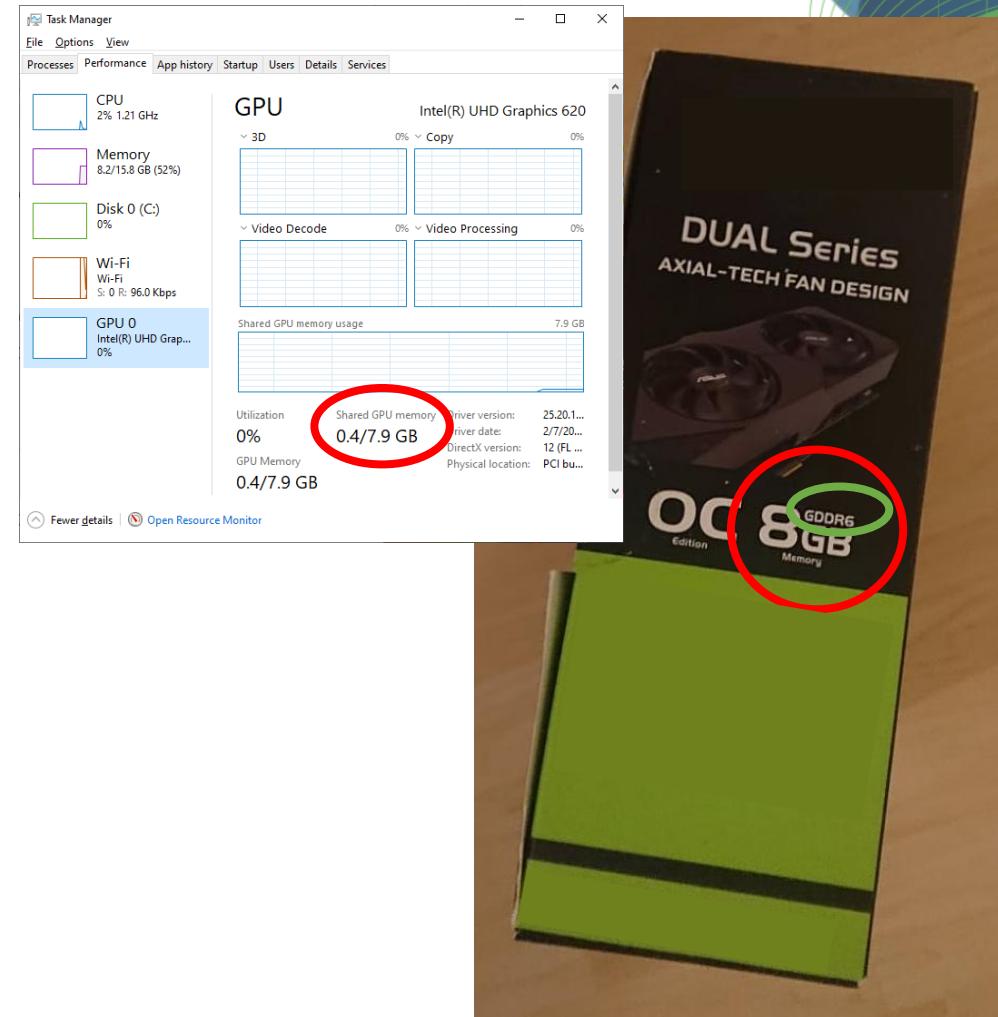
GPU-accelerated image processing



Checklist: When does GPU-accelerated image processing make sense?

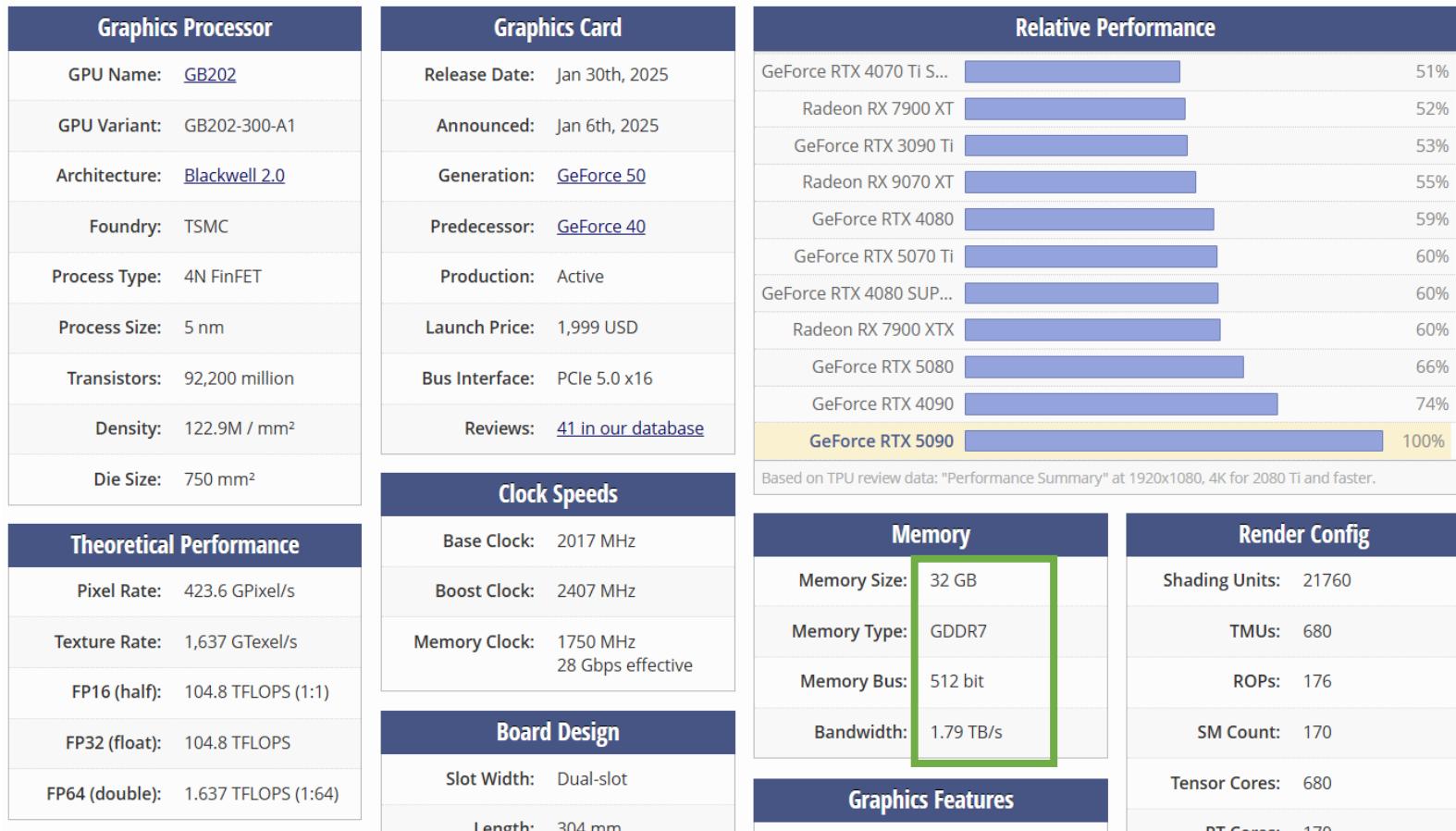
- In order to accelerate your image analysis workflow
- The pre-existing workflow should be slow; ideally: $(\text{processing time} / \text{loading time}) > 10$,
- Memory requirements: rule of thumb: image size in GB \times number of processing steps + 1) should **fit in your graphics card memory**,

- If you really want to get the most out of it, you should own a graphics card with **GDDR6** memory (memory bandwidth $> 400 \text{ Gb/s}$).
- Comparison: common DDR4 memory has a bandwidth of about 40 GB/s



Nvidia RTX 5090 specs

- Image processing is “memory bound”; important specs:



Quiz: What does “memory bound” mean?

Nvidia RTX 4070 versus 4070 mobile

- Vendors name GPUs similarly, despite their configuration is quite different.

RTX 4070

Graphics Processor	
GPU Name:	AD104
GPU Variant:	AD104-250-A1
Architecture:	Ada Lovelace
Foundry:	TSMC
Process Size:	5 nm
Transistors:	35,800 million
Density:	121.8M / mm ²
Die Size:	294 mm ²

RTX 4070 mobile

Graphics Processor	
GPU Name:	AD106
GPU Variant:	GN21-X6
Architecture:	Ada Lovelace
Foundry:	TSMC
Process Size:	5 nm
Transistors:	22,900 million
Density:	121.8M / mm ²
Die Size:	188 mm ²

Nvidia RTX 4070 versus 4070 mobile

- Vendors name GPUs similarly, despite their config is quite different.

Quiz: Why are mobile GPUs throttled?

RTX 4070

Clock Speeds	
Base Clock:	1920 MHz
Boost Clock:	2475 MHz
Memory Clock:	1313 MHz 21 Gbps effective
Memory	
Memory Size:	12 GB
Memory Type:	GDDR6X
Memory Bus:	192 bit
Bandwidth:	504.2 GB/s

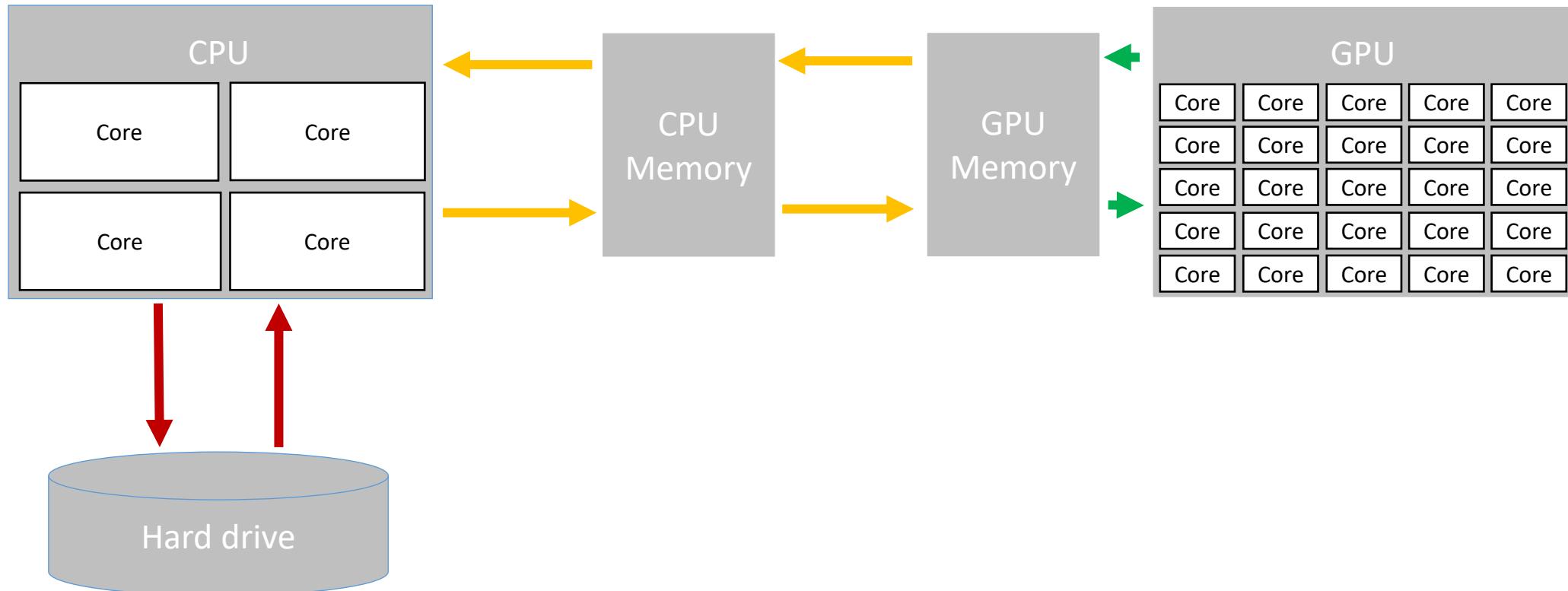
Board Design	
Slot Width:	Dual-slot
Length:	240 mm 9.4 inches
Width:	110 mm 4.3 inches
Height:	40 mm 1.6 inches
TDP:	200 W
Suggested PSU:	550 W

Clock Speeds	
Base Clock:	1395 MHz
Boost Clock:	1695 MHz
Memory Clock:	2000 MHz 16 Gbps effective
Memory	
Memory Size:	8 GB
Memory Type:	GDDR6
Memory Bus:	128 bit
Bandwidth:	256.0 GB/s

Board Design	
Slot Width:	IGP
TDP:	115 W
Outputs:	Portable Device Dependent
Power Connectors:	None

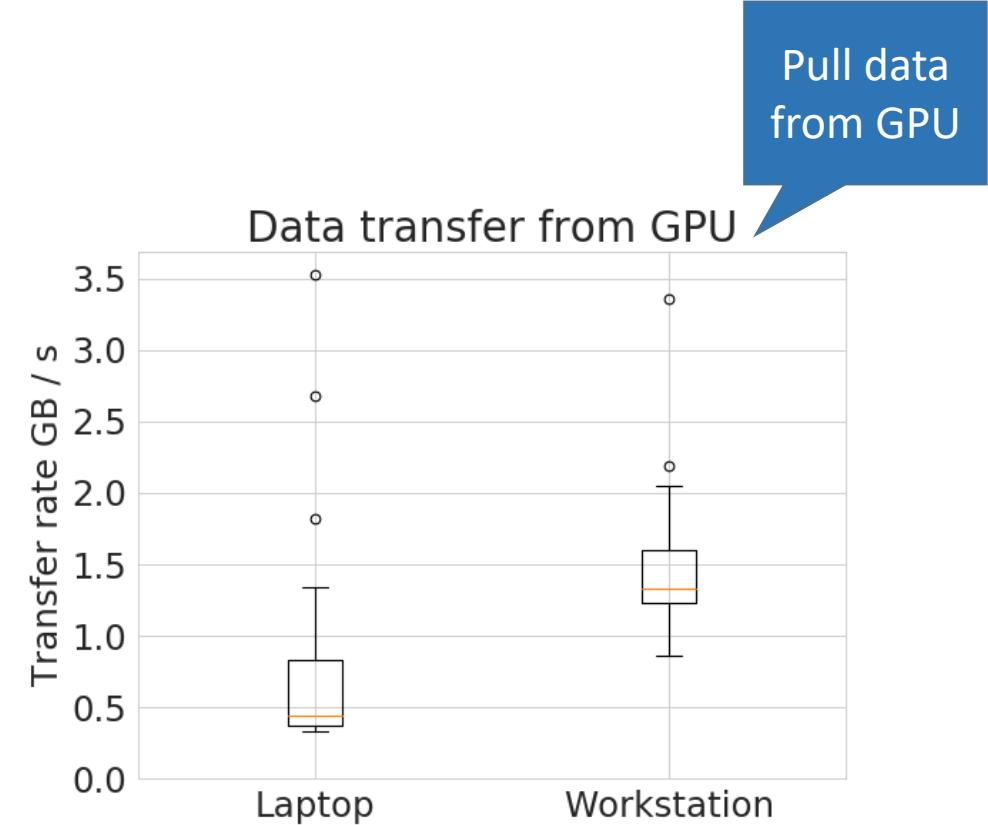
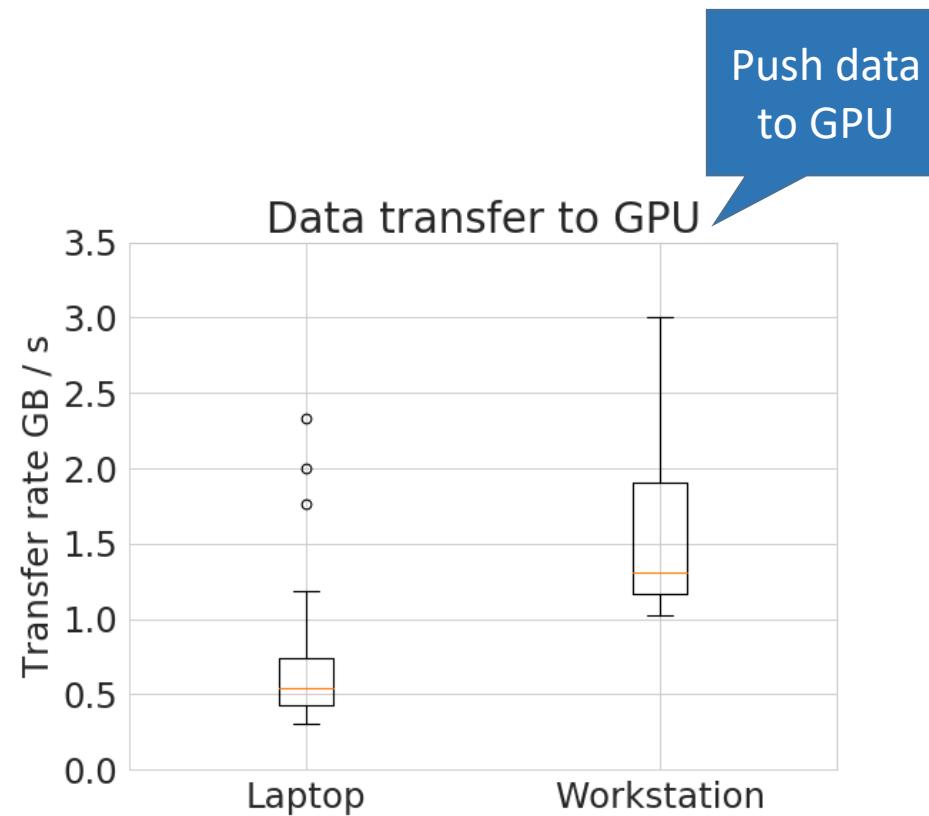
GPUs allow real-time image processing

- GPUs are specialised in processing, very fast thanks to many cores and fast memory access



Data transfer takes time

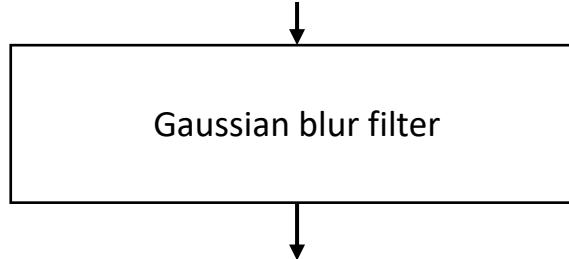
- Data transfer is the bottle neck



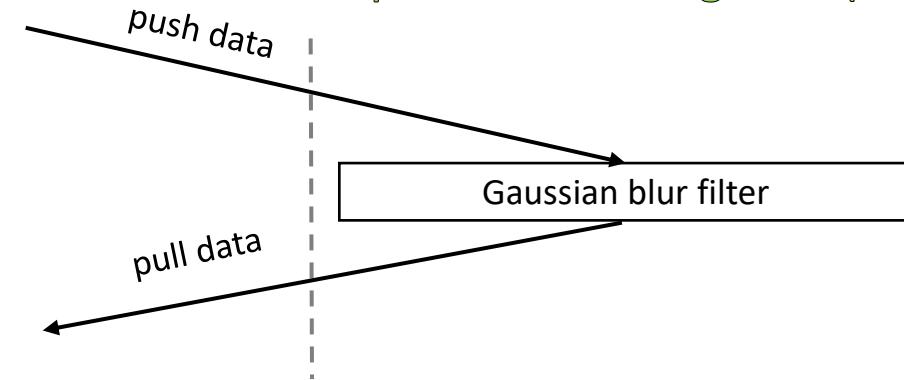
Build workflows consisting of many operations

GPU acceleration may suffer from data transfer between CPU and GPU

Central Processing Unit (CPU)



Graphics Processing Unit (GPU)



Optimal performance through smart memory management

- Example workflow processing a *Drosophila melanogaster* embryo, histone-GFP

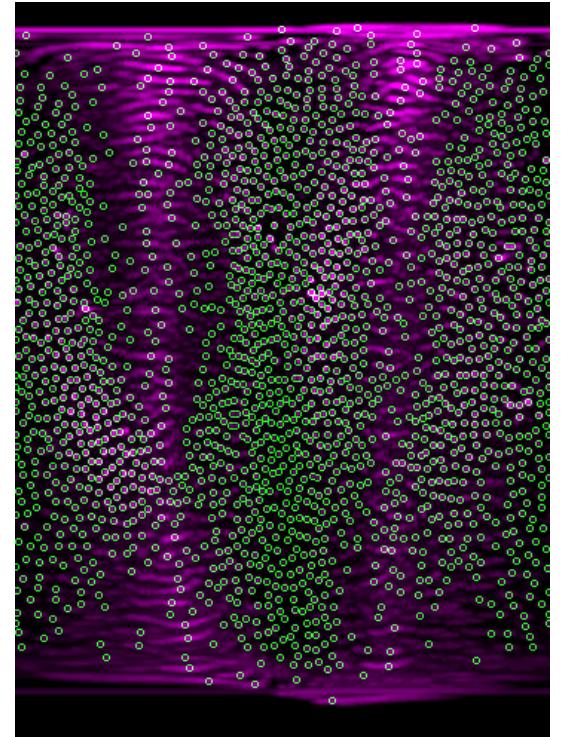
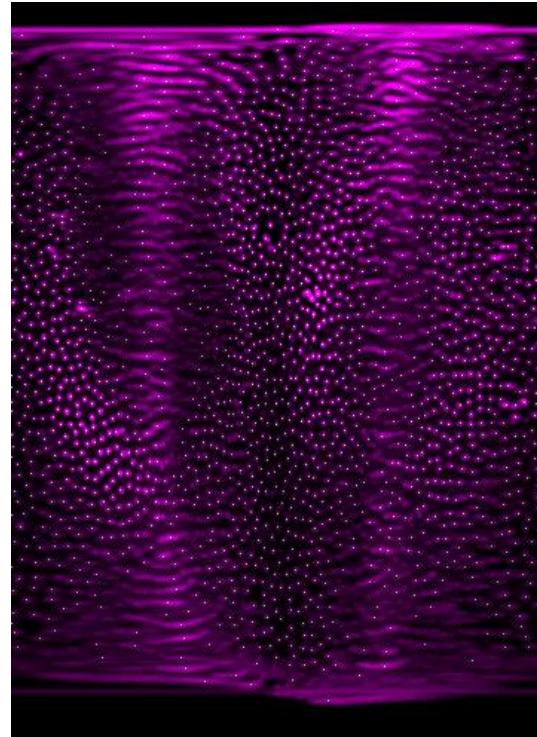
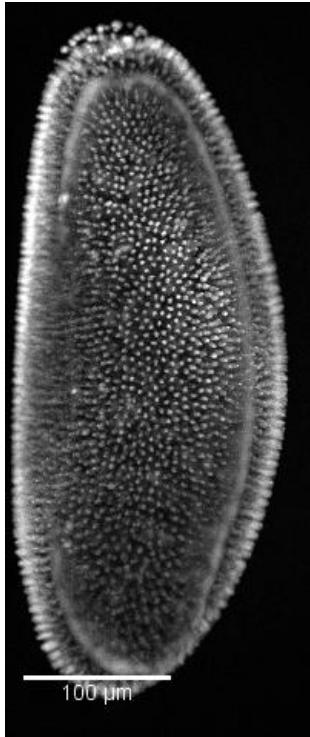
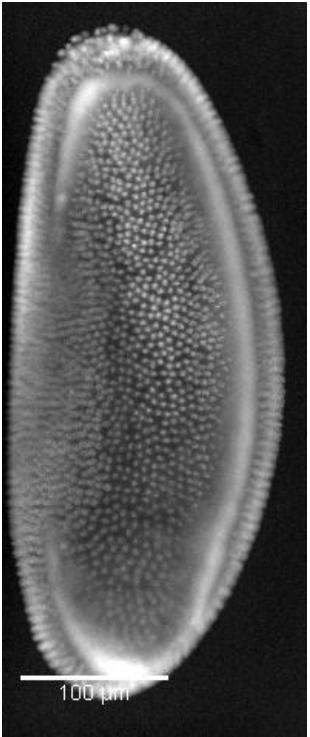
Load data

Preprocessing

Transformation

Segmentation

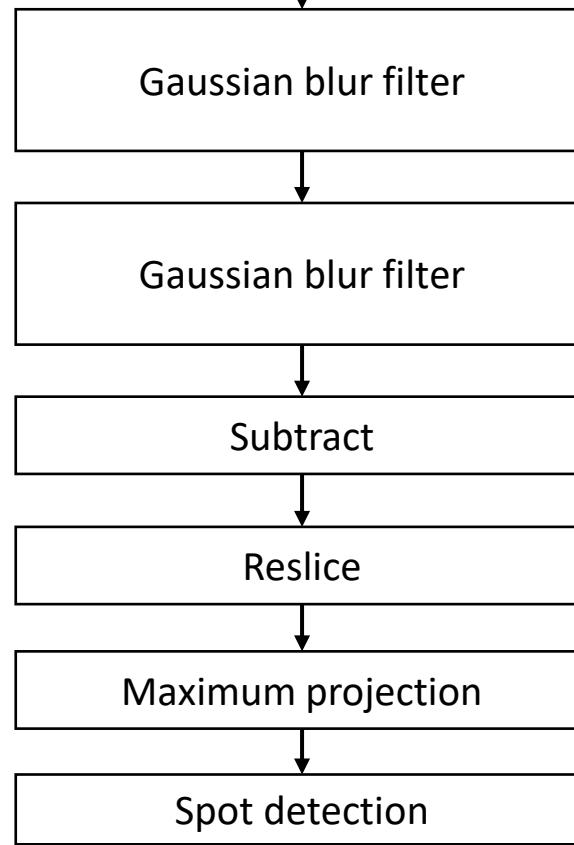
Save data



Build workflows consisting of many operations

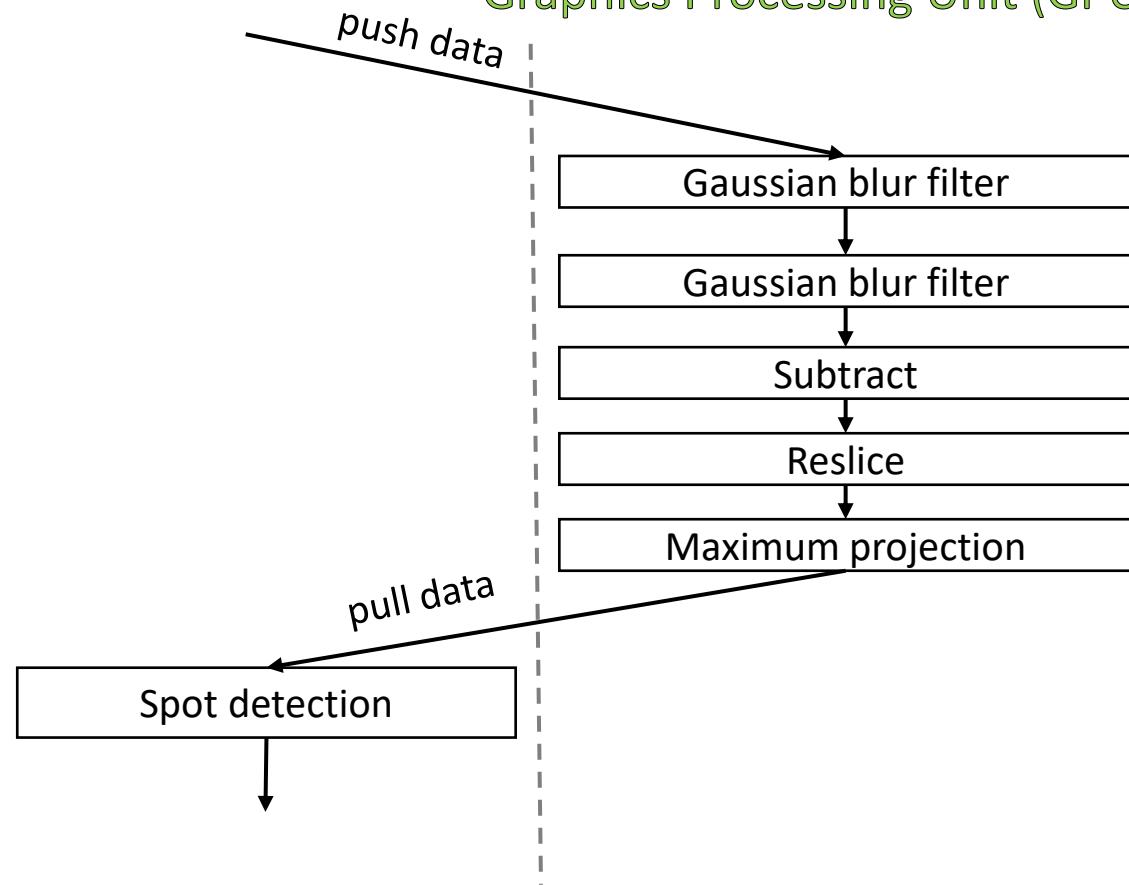
GPU acceleration may suffer from data transfer between CPU and GPU

Central Processing Unit (CPU)



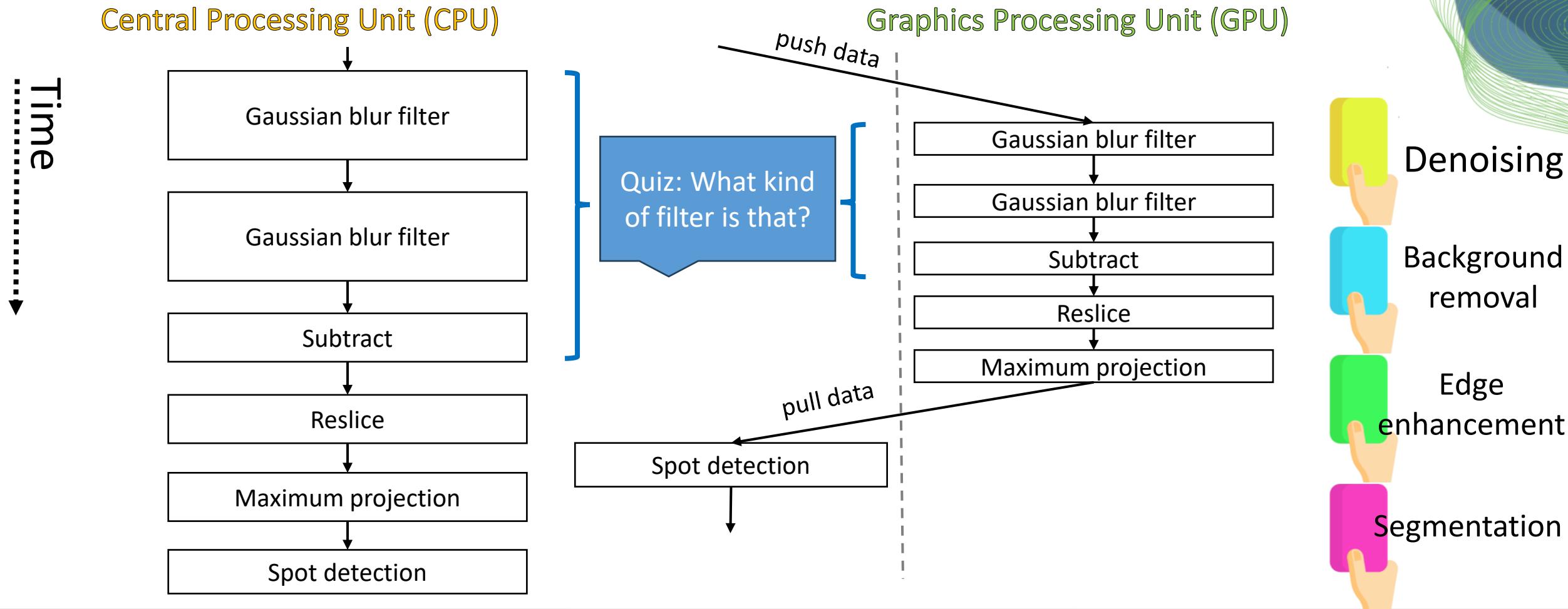
Time ↓

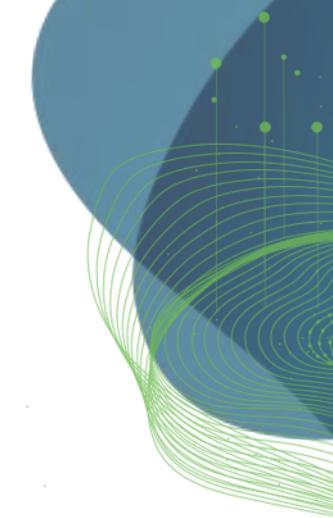
Graphics Processing Unit (GPU)



Build workflows consisting of many operations

GPU acceleration may suffer from data transfer between CPU and GPU





GPU-accelerated Image Processing in Python: OpenCL / clesperanto

Robert Haase

Funded by



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.

GPU-accelerated image processing

- Listing available GPUs

```
[2]: import pyclesperanto as cle  
  
# List available devices  
cle.available_device_names()
```

```
[2]: ['NVIDIA GeForce RTX 4070 Laptop GPU', 'gfx1103']
```

Integrated
AMD GPU

GPU-accelerated image processing

- Choosing a GPU

```
[2]: import pyclesperanto as cle  
  
# List available devices  
cle.available_device_names()
```

```
[2]: ['NVIDIA GeForce RTX 4070 Laptop GPU', 'gfx1103']
```

```
[3]: # select a specific device with only a part of its name  
cle.select_device("TX")
```

```
[3]: (OpenCL) NVIDIA GeForce RTX 4070 Laptop GPU (OpenCL 3.0 CUDA)  
      Vendor: NVIDIA Corporation  
      Driver Version: 560.94  
      Device Type: GPU  
      Compute Units: 36  
      Global Memory Size: 8187 MB  
      Local Memory Size: 0 MB  
      Maximum Buffer Size: 2046 MB  
      Max Clock Frequency: 1230 MHz  
      Image Support: Yes
```

Quiz: Why does it make sense to limit the maximum buffer size to about $\frac{1}{4}$ of the global memory size

GPU-accelerated image processing

- Memory constraints

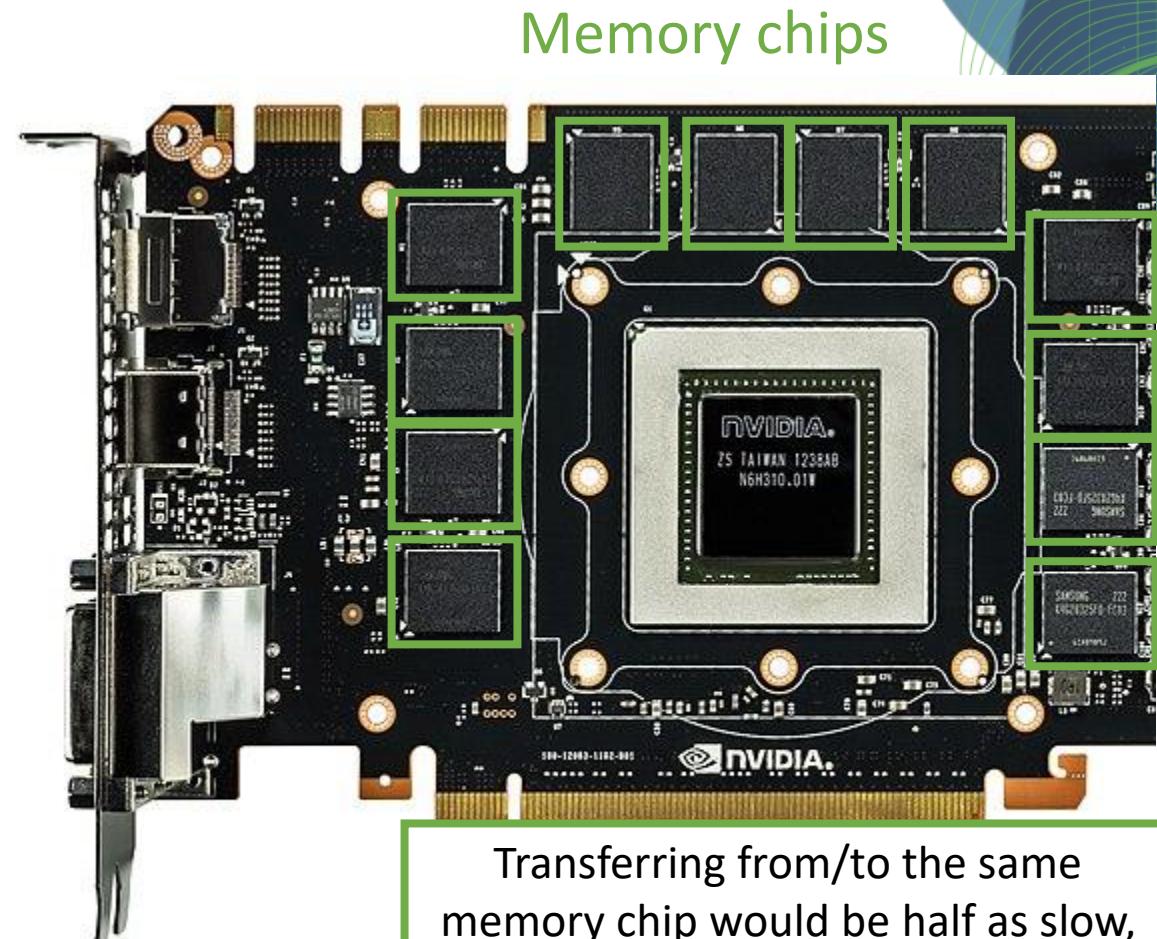
```
[2]: import pyclesperanto as cle

# List available devices
cle.available_device_names()

[2]: ['NVIDIA GeForce RTX 4070 Laptop GPU', 'gfx1103']

[3]: # select a specific device with only a part of its name
cle.select_device("TX")

[3]: (OpenCL) NVIDIA GeForce RTX 4070 Laptop GPU (OpenCL 3.0 CUDA)
      Vendor: NVIDIA Corporation
      Driver Version: 560.94
      Device Type: GPU
      Compute Units: 36
      Global Memory Size: 8187 MB
      Local Memory Size: 0 MB
      Maximum Buffer Size: 2046 MB
      Max Clock Frequency: 1230 MHz
      Image Support: Yes
```



Memory chips

GPU-accelerated image processing

- Memory constraints

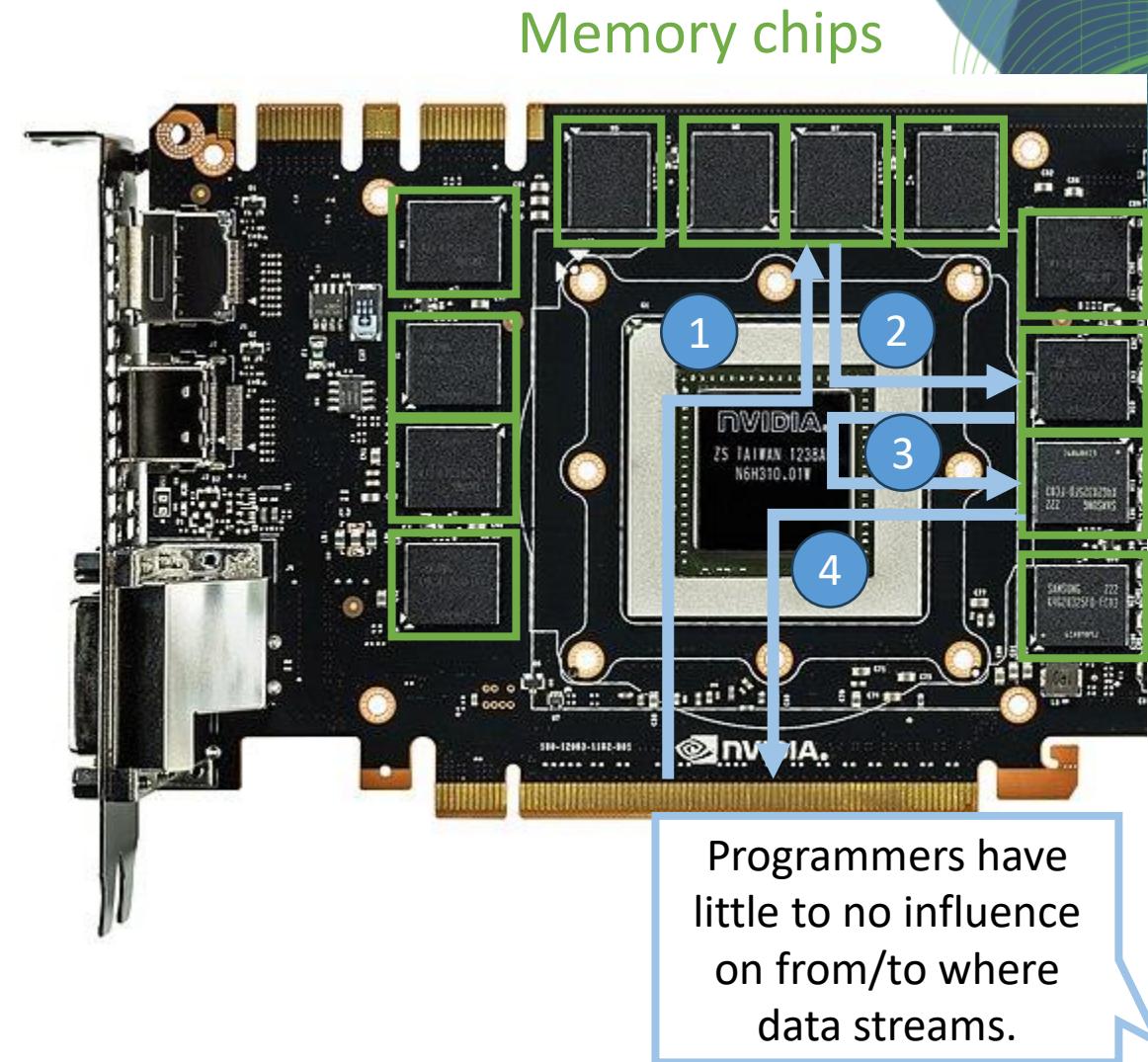
```
[2]: import pyclesperanto as cle

# List available devices
cle.available_device_names()

[2]: ['NVIDIA GeForce RTX 4070 Laptop GPU', 'gfx1103']

[3]: # select a specific device with only a part of its name
cle.select_device("TX")

[3]: (OpenCL) NVIDIA GeForce RTX 4070 Laptop GPU (OpenCL 3.0 CUDA)
      Vendor: NVIDIA Corporation
      Driver Version: 560.94
      Device Type: GPU
      Compute Units: 36
      Global Memory Size: 8187 MB
      Local Memory Size: 0 MB
      Maximum Buffer Size: 2046 MB
      Max Clock Frequency: 1230 MHz
      Image Support: Yes
```



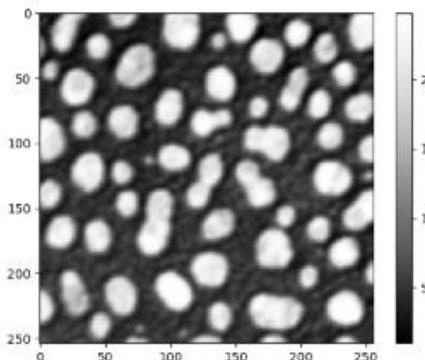
GPU-accelerated image processing

- GPU-accelerated image processing also just uses functions.
- You used it in the recent weeks already.

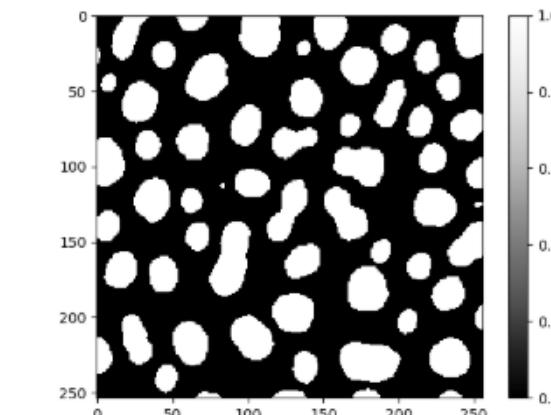
```
[6]: # noise removal
```

```
blurred = cle.gaussian_blur(image, sigma_x=1, sigma_y=1)  
blurred
```

```
[6]:
```

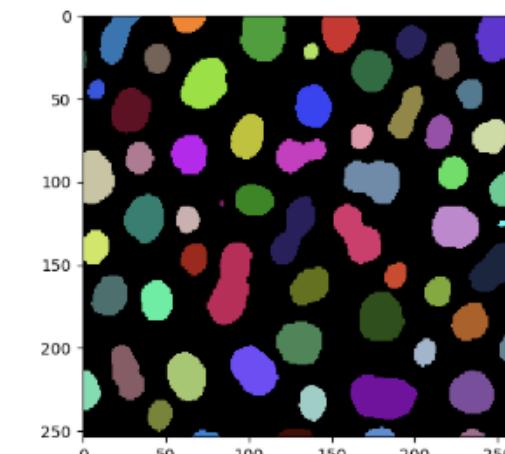


```
[7]:
```



```
[8]:
```

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



cle_.image

shape (254, 256)

dtype uint32

size 254.0 kB

min 0.0

max 62.0

GPU-accelerated image processing

- Hint: Use SHIFT-Tab to discover functions in clesperanto. You can find them all in the main module. There are no sub-modules like in scikit-image.

```
[ ]: cle.gauss
f gauss_otsu_labeling          function
f gaussian_blur                function
f subtract_gaussian_background function
f divide_by_gaussian_background function
f difference_of_gaussian       function ▾
```

```
[ ]: cle.otsu
f gauss_otsu_labeling          function
f eroded_otsu_labeling         function
f voronoi_otsu_labeling        function
f threshold_otsu               function
f remove_labels_with_map_values_out_of_range function ▾
```

Benchmarking

- The timeit library makes it straightforward to benchmark compute time in Python / Jupyter.

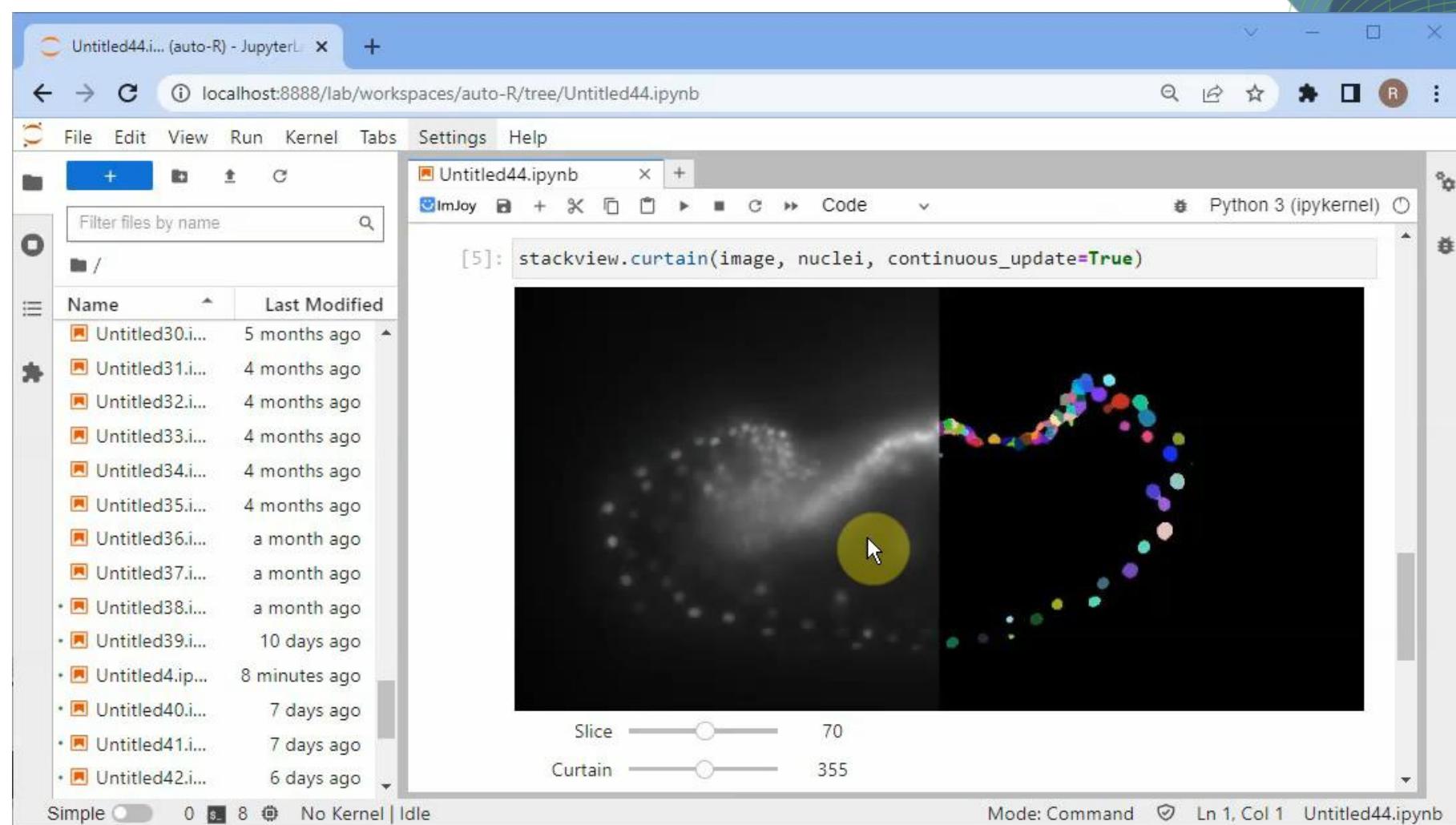
```
[7]: %timeit  
skimage.filters.gaussian(image, sigma=5, out=blurred_image, preserve_range=True)  
2.53 s ± 21.8 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

```
[16]: %timeit  
cle.gaussian_blur(ocl_image, output_image=ocl_blurred, sigma_x=5, sigma_y=5, sigma_z=5)  
19.2 ms ± 465 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

Quiz: How much faster is cle compared to the scikit-image filter in this case?

3D visualization on the cluster

- When working on the cluster / Jupyter Hub, consider using stackview instead of napari for inspecting images in 3D.





GPU-accelerated Image Processing in Python: CUDA / cupy

Robert Haase

Funded by



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.

cupy

- CUDA-based GPU-accelerated [image] data processing in Python

The image displays two side-by-side browser windows. The left window shows the official CuPy website at cupy.dev. It features a dark background with a green and white geometric cube logo and the word "CuPy". A green button labeled "NumPy/SciPy-compatible Array Library for GPU-accelerated Computing with Python" is highlighted with a red box. Below the button are three buttons: "GET STARTED" (red), "API REFERENCE" (teal), and "GITHUB" (teal). The right window shows the GitHub repository for CuPy at github.com/cupy/cupy/. The repository page includes a summary card with "Issues 451", "Pull requests 63", and "Fork 707". Below this are sections for "main", "About", "Branches", and "Tags". A list of recent pull requests is shown, with the first one by "takagi" being a merge pull request. The right sidebar contains a "NumPy & SciPy for GPU" section, a "cupy.dev" link, and a tag cloud with terms like "python", "gpu", "numpy", "cuda", etc.

drop-in replacement for numpy and scipy

- The API of some cupy packages is close to the scipy/numpy API.
- This allows *easy* switching from scipy to cupy.

```
import scipy.ndimage as ndi
```

```
ndi.gaussian_filter(image, sigma=5)
```

```
import cupyx.scipy.ndimage as xdi
```

- However, image data still needs to be pushed to GPU memory.

```
xp_image = xp.asarray(image)
```

```
xdi.gaussian_filter(xp_image, sigma=5)
```

Common patterns

- To make code independent from cupy availability, while minimizing if-else blocks, some common design patterns emerged:

```
try:  
    import cupy as xp  
except:  
    import numpy as xp  
  
import numpy as np
```

If this fails because cupy is not installed,
This will execute and xp will be available.

We can still use np anyway.

The same pattern works with scipy.ndimage

```
try:  
    import cupyx.scipy.ndimage as xdi  
except:  
    import scipy.ndimage as xdi  
  
import scipy.ndimage as ndi
```

Common patterns

- You can then call magic code like this, which will do different things depending on cupy-availability.

- If cupy is available:

```
[3]: image = imread("../data/blobs.tif")
      xp_image = xp.asarray(image)
      type(xp_image)

[3]: cupy.ndarray
```

- If cupy is not available:

```
[3]: image = imread("../data/blobs.tif")
      xp_image = xp.asarray(image)
      type(xp_image)

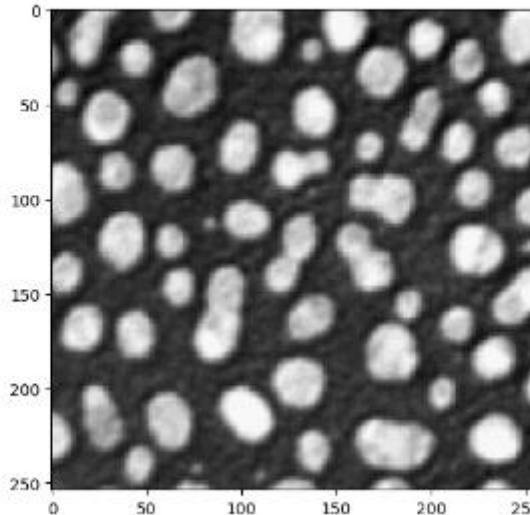
[3]: numpy.ndarray
```

Common patterns

- Some if-else blocks are hard to avoid

```
[7]: if np == xp:  
    np_image = xp_image  
else:  
    np_image = xp.asarray(xp_image)  
  
imshow(np_image)
```

```
[7]: <matplotlib.image.AxesImage at  
0x24692ef85e0>
```

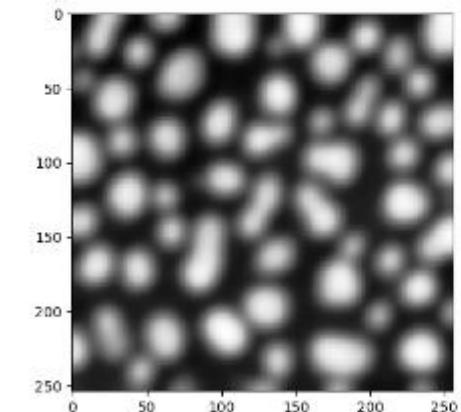


- Stackview aims to be cupy/numpy agnostic

```
[4]: xp_blurred = xdi.gaussian_filter(xp_image, sigma=5)
```

```
[5]: stackview.insight(xp_blurred)
```

```
[5]:
```



shape	(254, 256)
dtype	uint8
size	63.5 kB
min	35
max	237

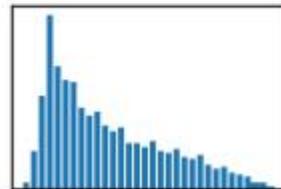
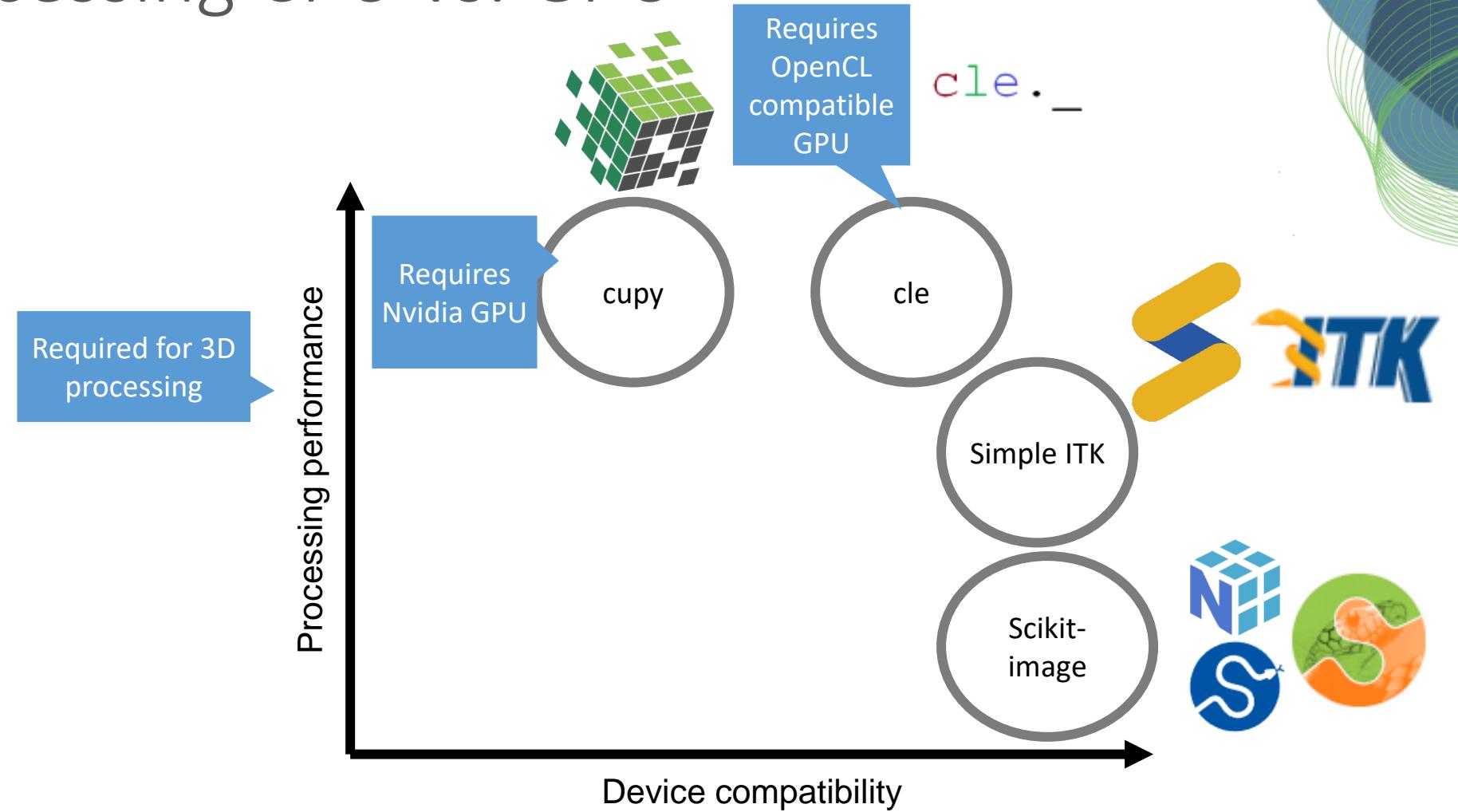
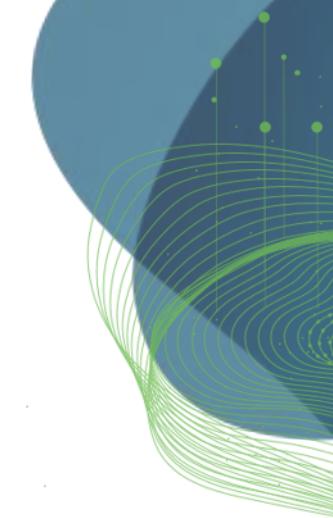


Image Processing CPU vs. GPU

- Performance versus compatibility





Tiled image processing

Robert Haase

Funded by



Bundesministerium
für Bildung
und Forschung

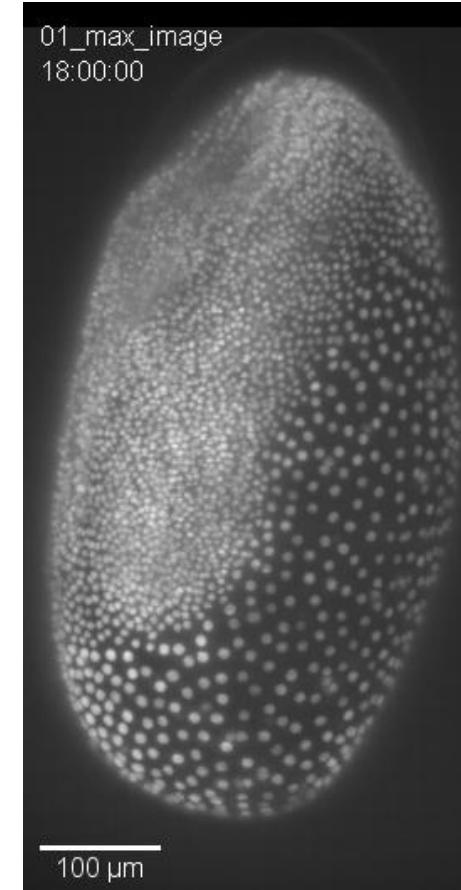


Diese Maßnahme wird gefördert durch die Bundesregierung
aufgrund eines Beschlusses des Deutschen Bundestages.
Diese Maßnahme wird mitfinanziert durch Steuermittel auf
der Grundlage des von den Abgeordneten des Sächsischen
Landtags beschlossenen Haushaltes.

Quiz: Memory constraints

- Assume your computer runs out of memory while processing this image dataset.
What can you do to avoid this?

- 1.
- 2.
- 3.
- 4.
5. Tiled image processing



The full data set is about
1024 (width)
2048 (height)
100 (depth)
3700 (frames)
large

Optimal performance through smart memory management

- The classical way of dealing with large image stacks...

Load data → Preprocessing

Load data

Load data

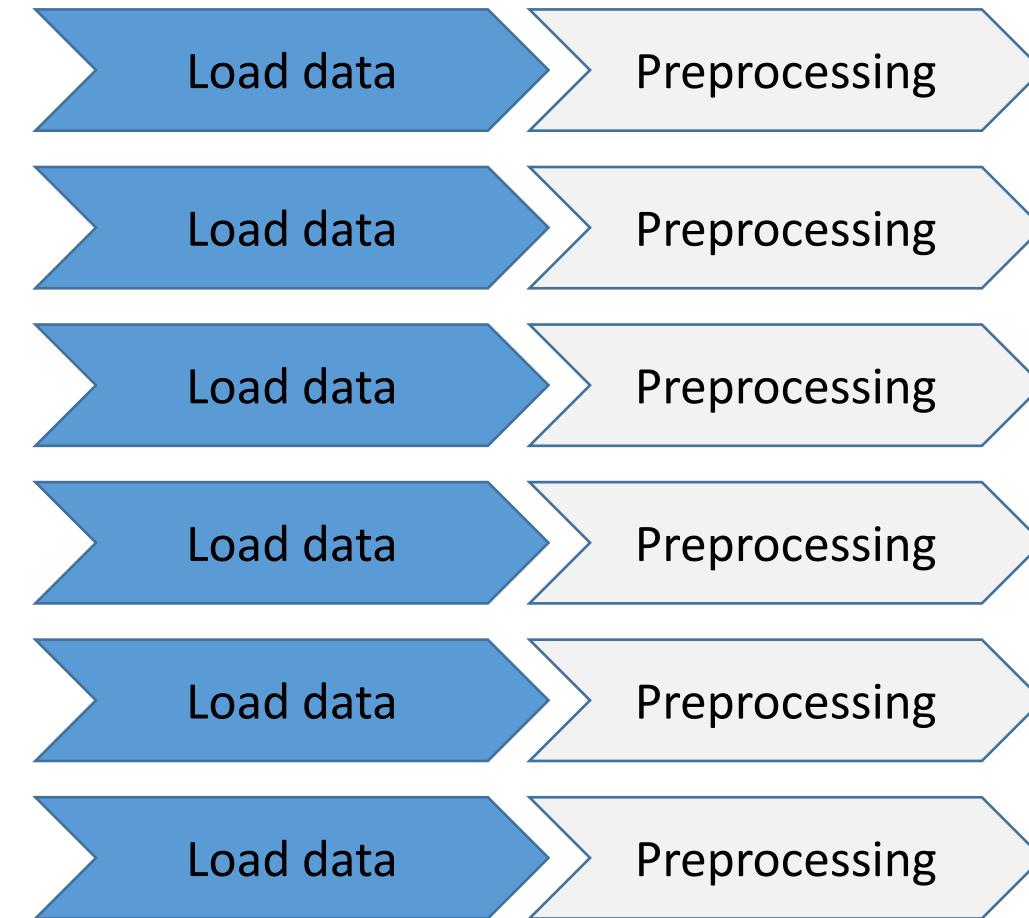
Load data

Load data

Load data

Optimal performance through smart memory management

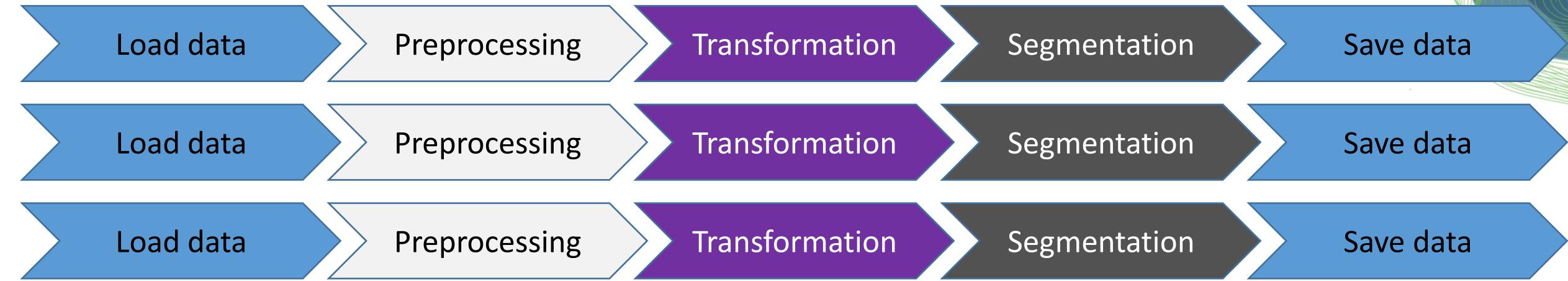
- The classical way of dealing with large image stacks... is suboptimal



This strategy does not just take long; it also costs a lot of memory!

Optimal performance through smart memory management

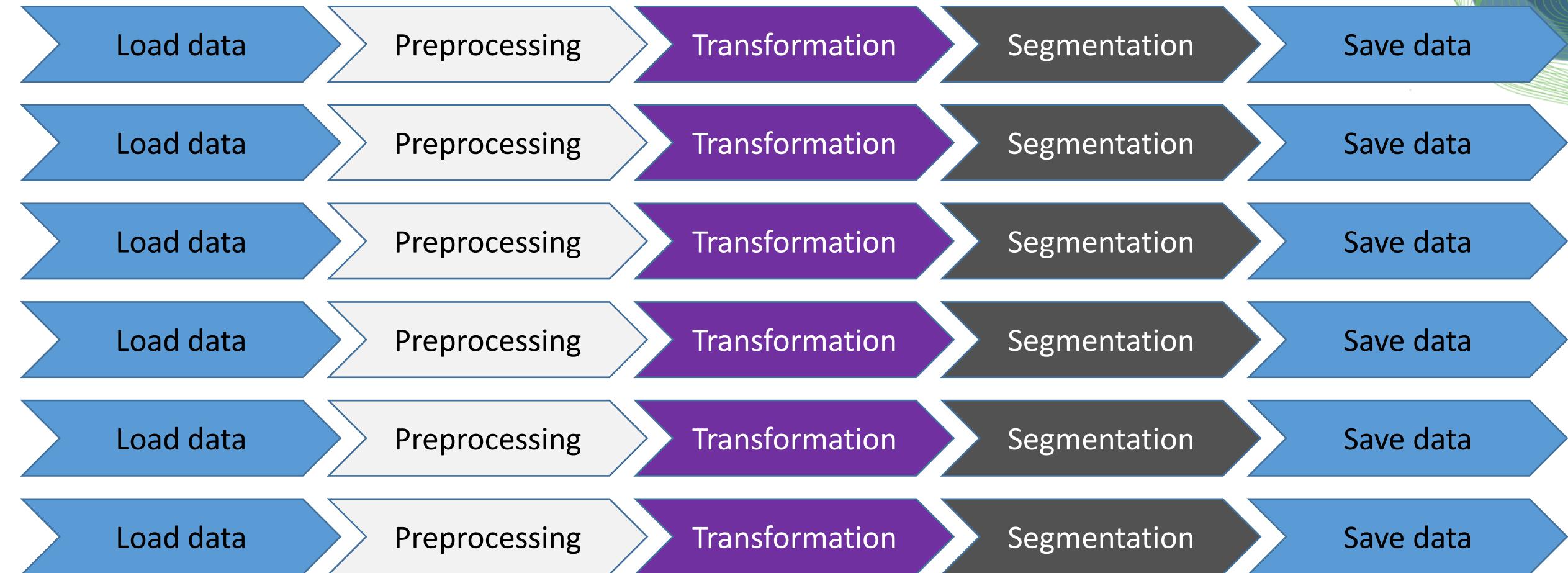
- Processing time-point by time-point is more efficient!



This strategy also works
tile-by-tile on large 3D
stacks and timelapse data!

Optimal performance through smart memory management

- Even better: Distribute tasks between parallelized computation systems

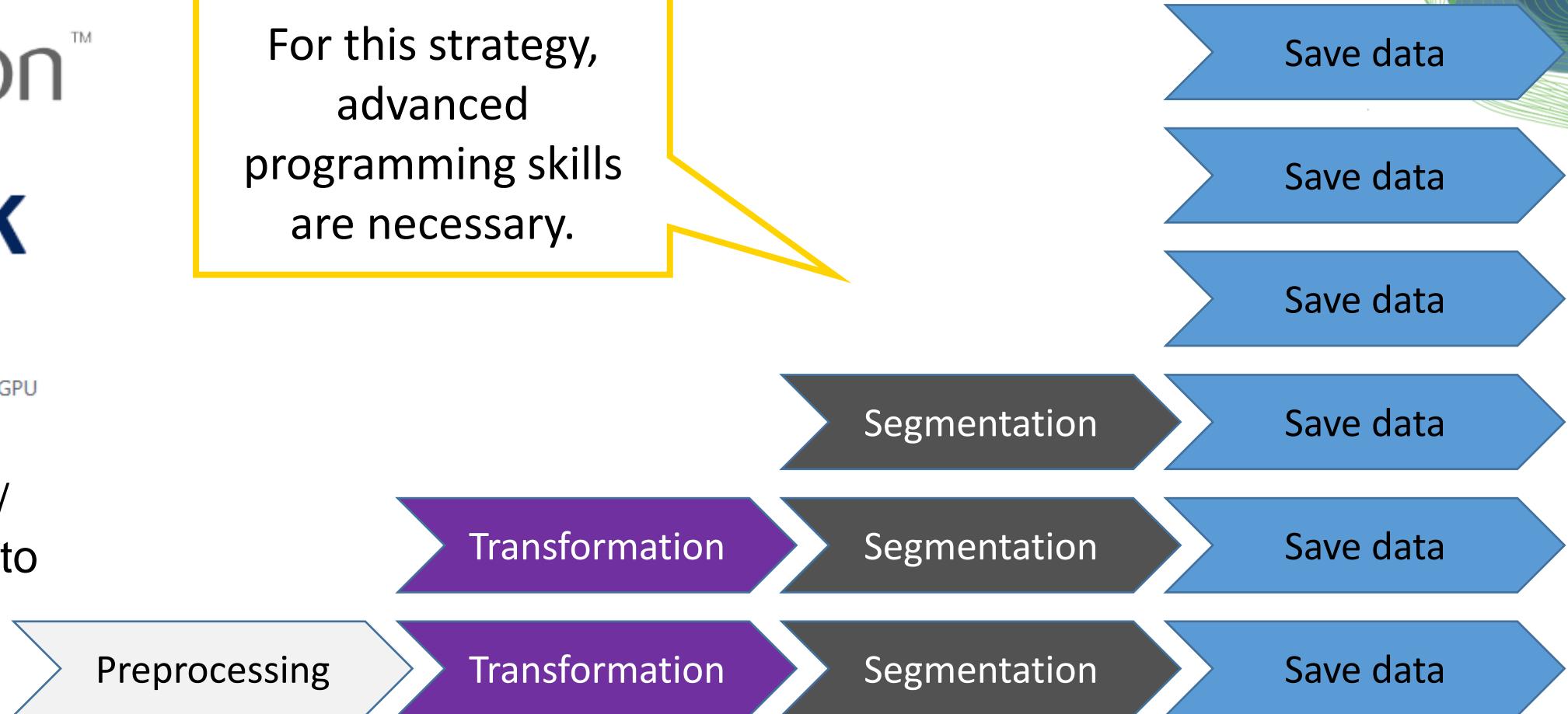


Optimal performance through smart memory management

- Even better: Distribute tasks between parallelized computation systems

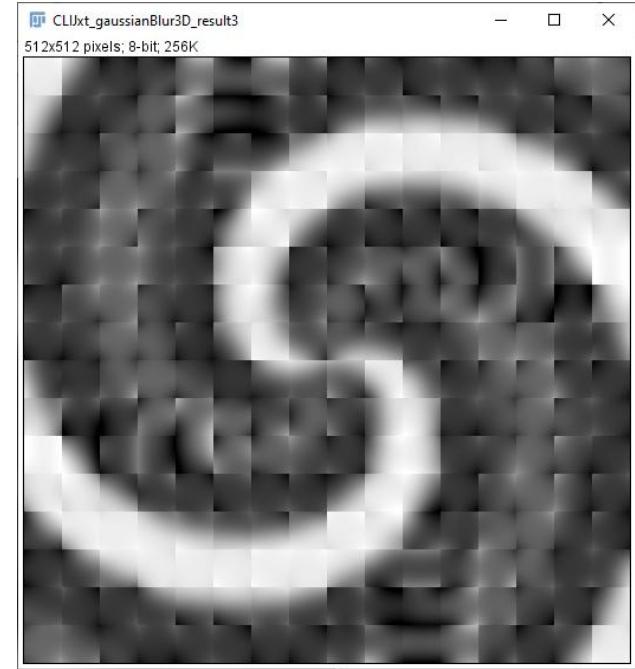
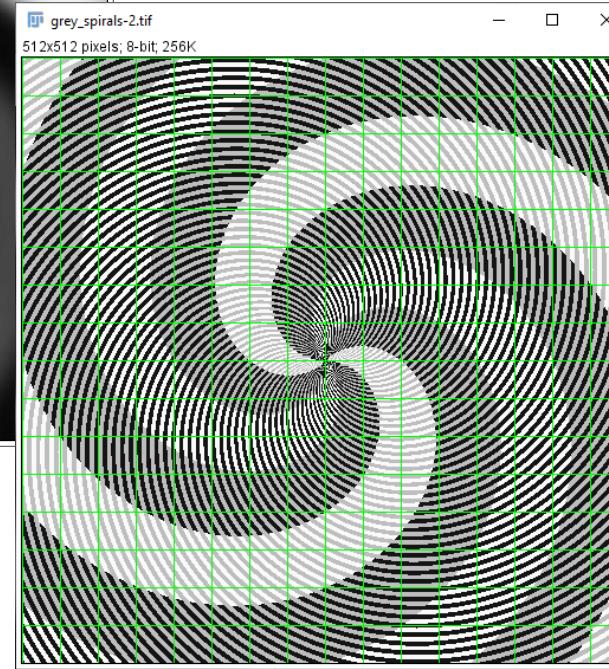
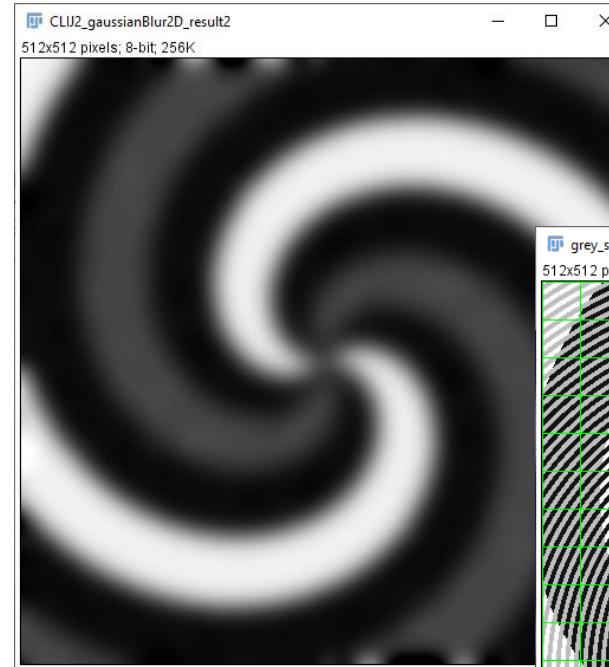
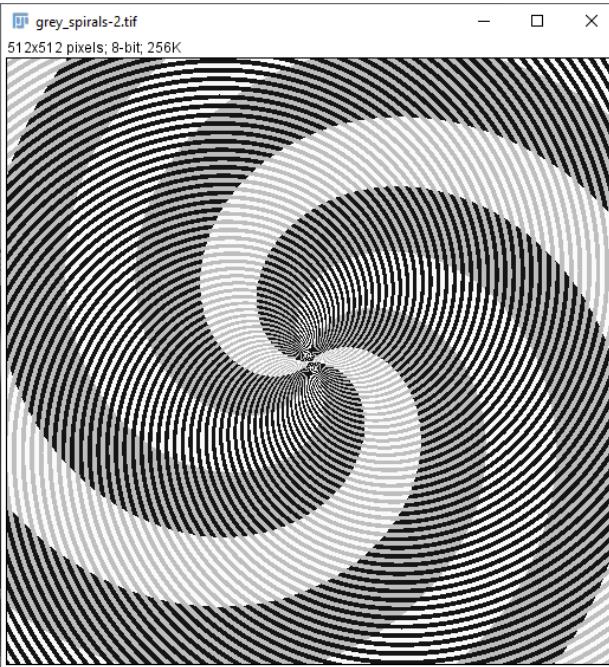


For this strategy,
advanced
programming skills
are necessary.



Tiling

- The **last** perimeter against big data



If the image is too large for the computer memory,
image processing as a whole is *not possible*.

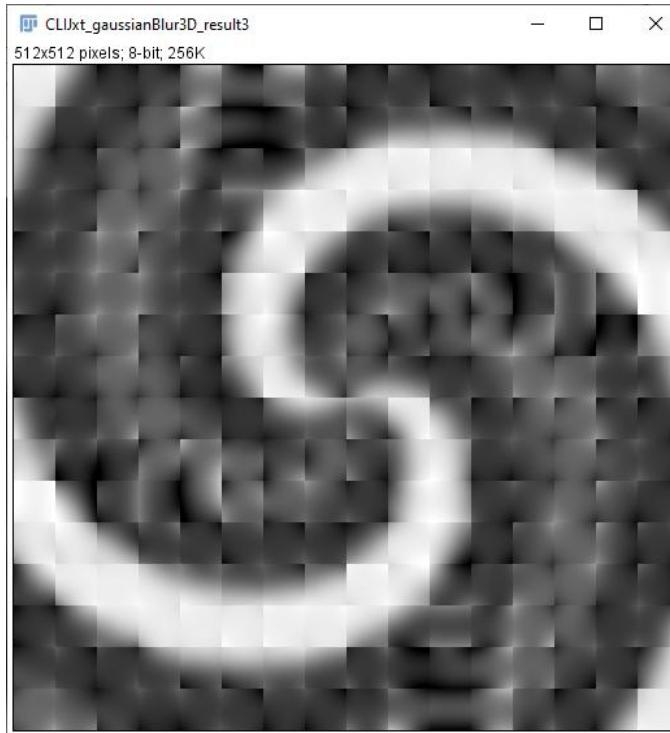
Processing tile-by-tile poses new challenges

Tiling

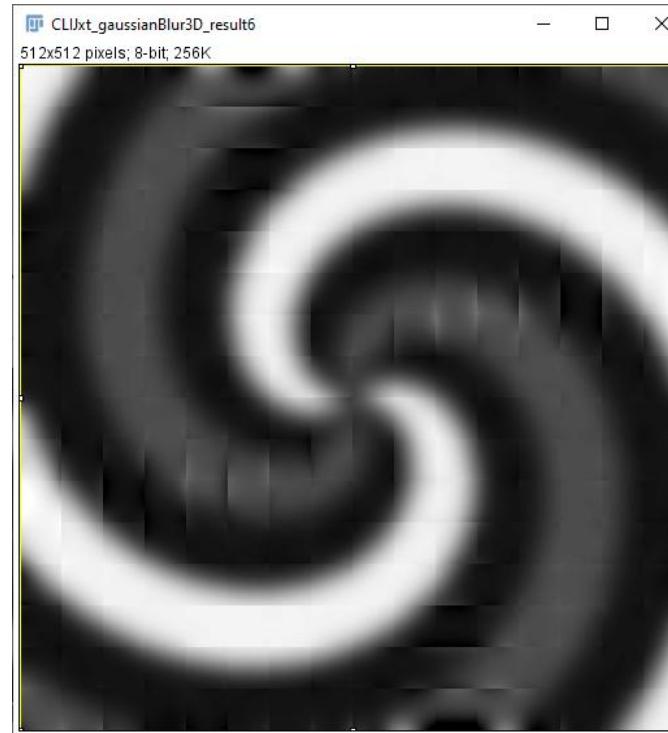
- Example: Gaussian blur (sigma = 20)
- Solution: Process with overlapping tiles (size + margin)

Optimal margin size depends
on algorithm and its
parameters

Margin: 0 pixels



Margin: 10 pixels



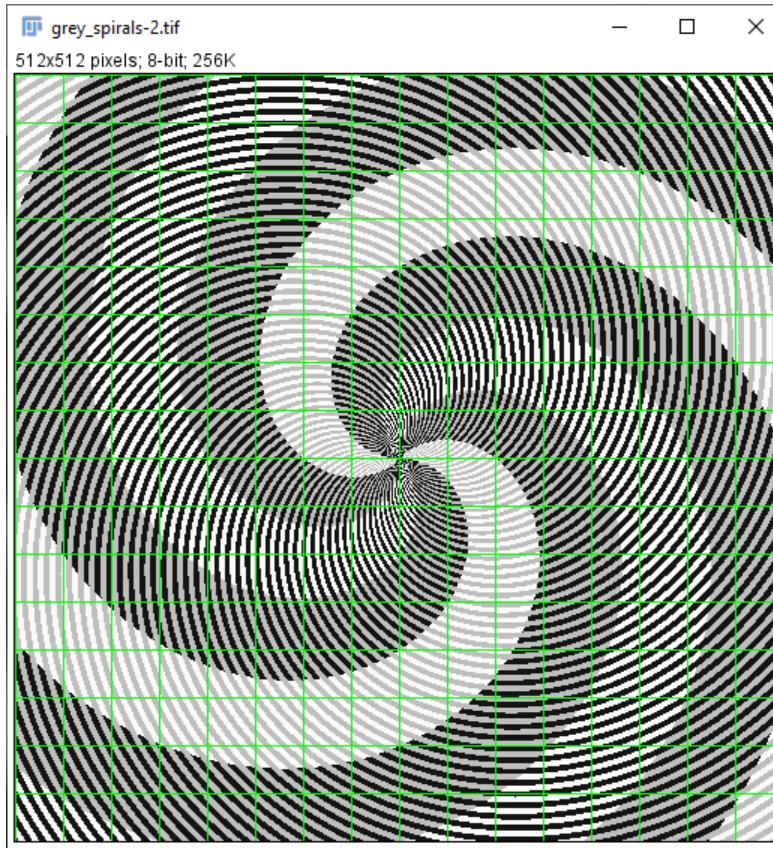
Margin: 20 pixels



Tiling

- Example: Gaussian blur (sigma = 20 pixels)
- Solution: Process with overlapping tiles (size + margin)

Computation time depends
on tile size and margin width



Margin: 20 pixels
Size: 5x original

Margin: 10 pixels
Size: 2.7x original

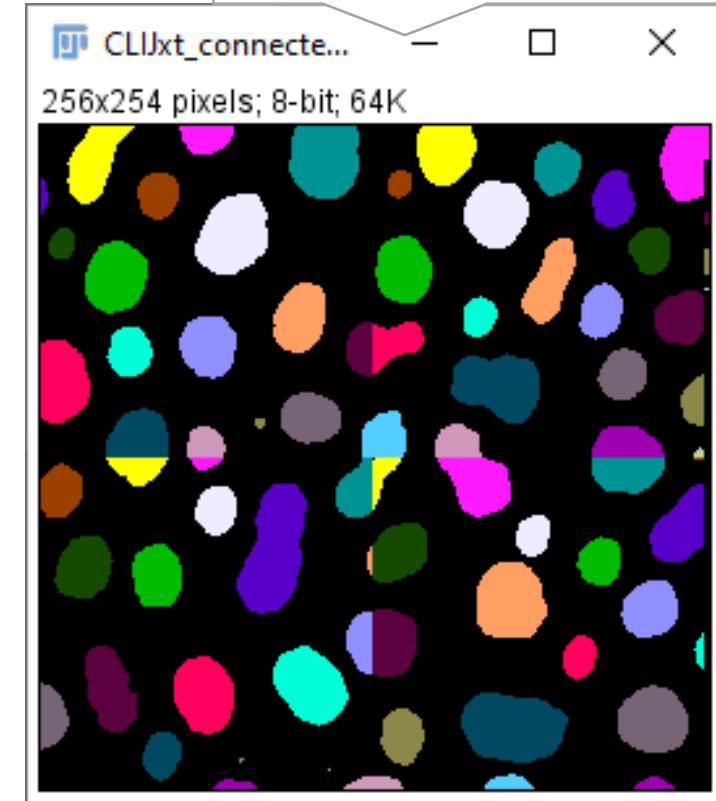
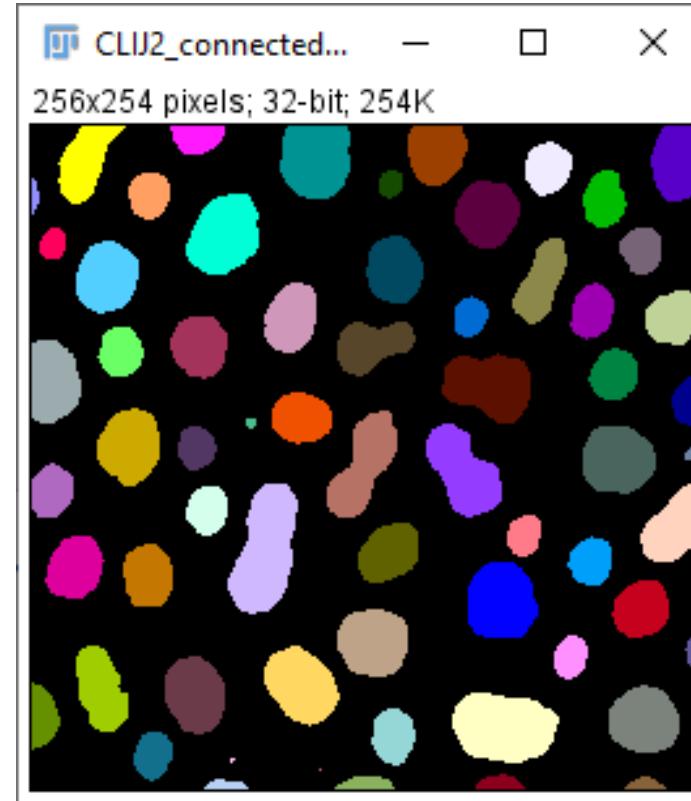
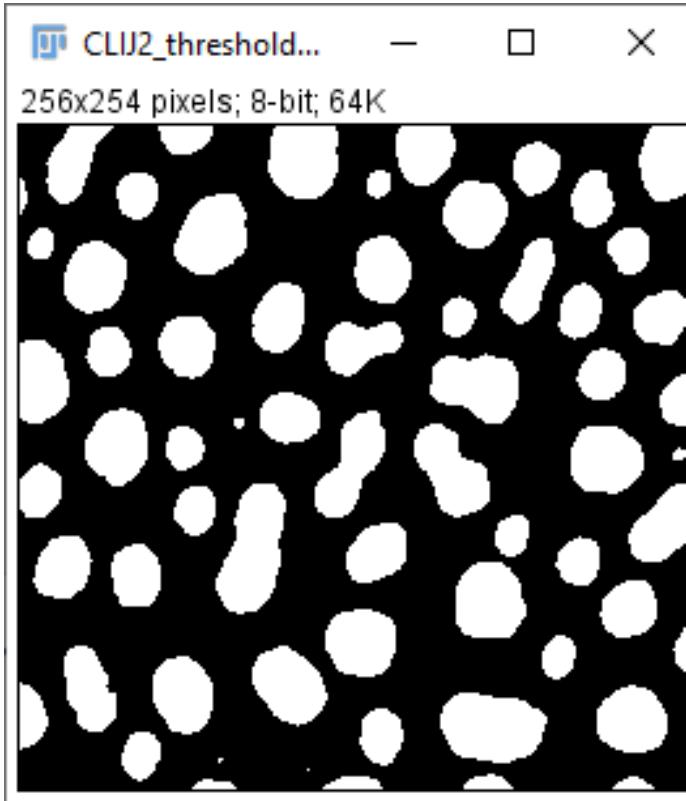
Tile
32x32 pixels

52x52 pixels

72x72 pixels

Tiling

- Some algorithms are hard to solve by processing tiles
- Example: Connected component analysis



Checking which labels touch and combine them is feasible.

Tiling

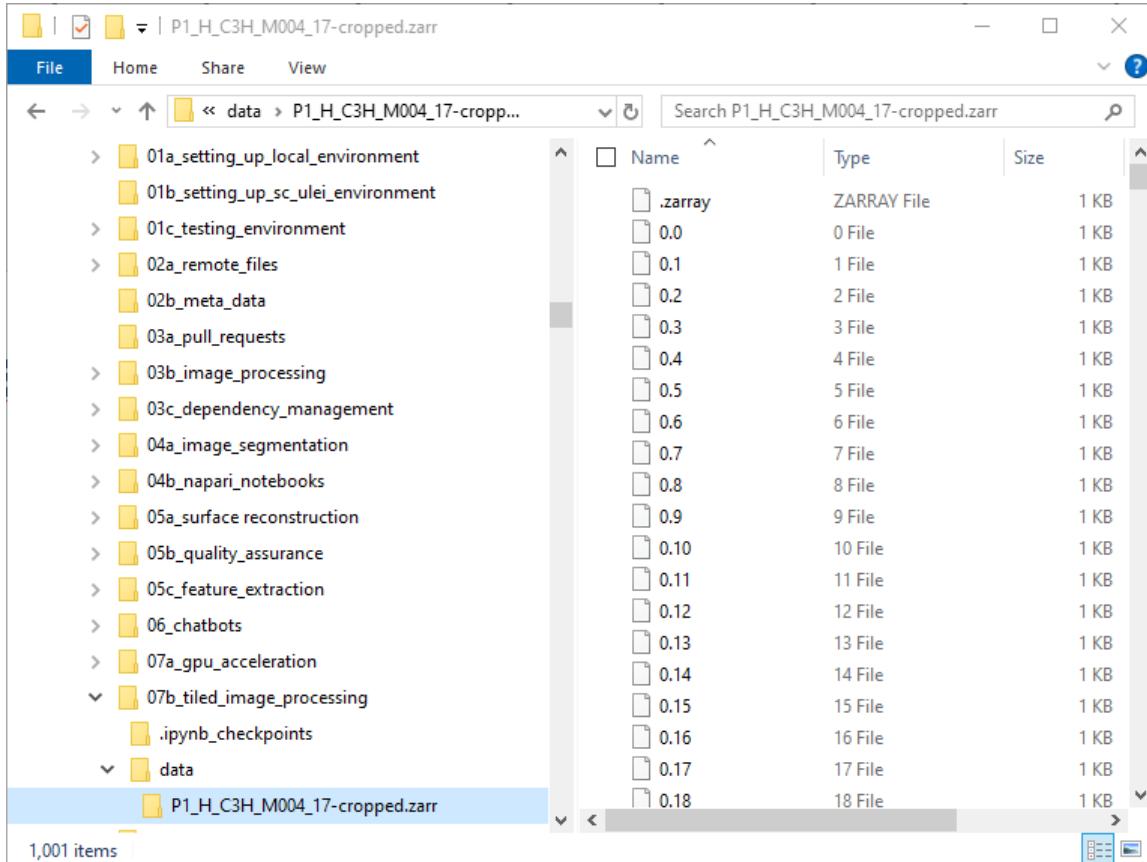
- Some algorithms are hard to solve by processing tiles
- Example: Connected component analysis

There are algorithms for that, but hardly available tools.



Tiled image processing in Python

- Key: tiled file formats, for parallel, distributed, lazy loading



After executing this,
no pixel has been
read yet.

```
zarr_image = da.from_zarr(zarr_filename)  
zarr_image
```

	Array	Chunk
Bytes	9.54 MiB	9.77 kB
Shape	(2000, 5000)	(100, 100)
Dask graph	1000 chunks in 2 graph layers	
Data type	uint8 numpy.ndarray	

5000 2000

Tiled image processing in Python

- Lazy processing

```
[5]: tile_map = da.map_blocks(count_nuclei, zarr_image)  
  
tile_map
```

```
Processing image of size (0, 0)  
(1, 1)  
Processing image of size (1, 1)  
(1, 1)
```

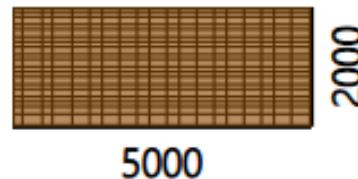
```
[5]:
```

	Array	Chunk
Bytes	76.29 MiB	78.12 kB
Shape	(2000, 5000)	(100, 100)

Task graph 1000 chunks in 3 graph layers

Data type float64 numpy.ndarray

After executing this, no pixel has been read or processed yet.



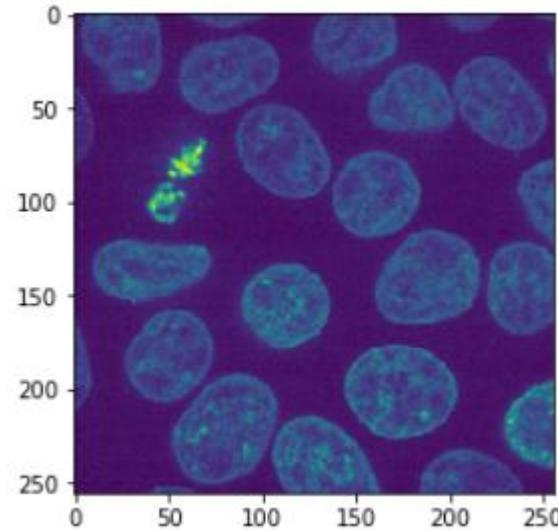
After that, results are available

```
result = tile_map.compute()
```

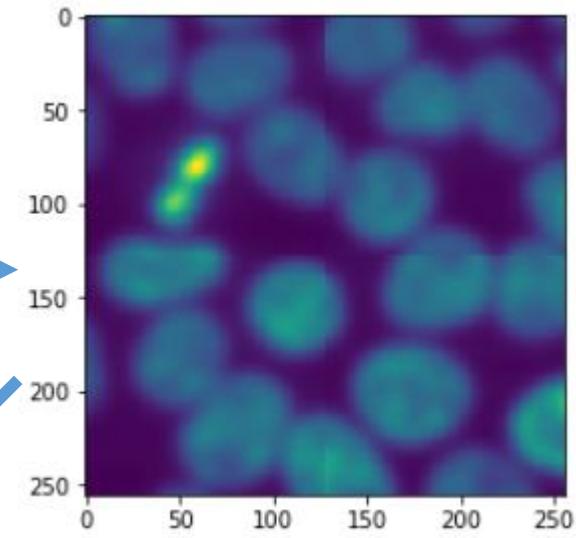
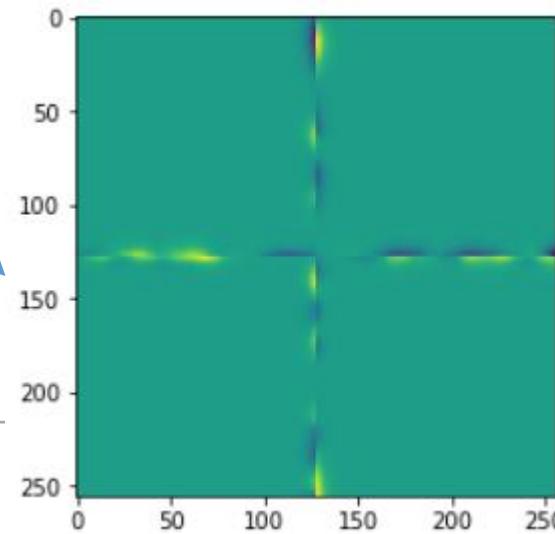
```
Processing image of size (100, 100)  
Processing image of size (100, 100)
```

Tiling with/out overlap

- Processing of images in tiles: artifacts ad tile borders

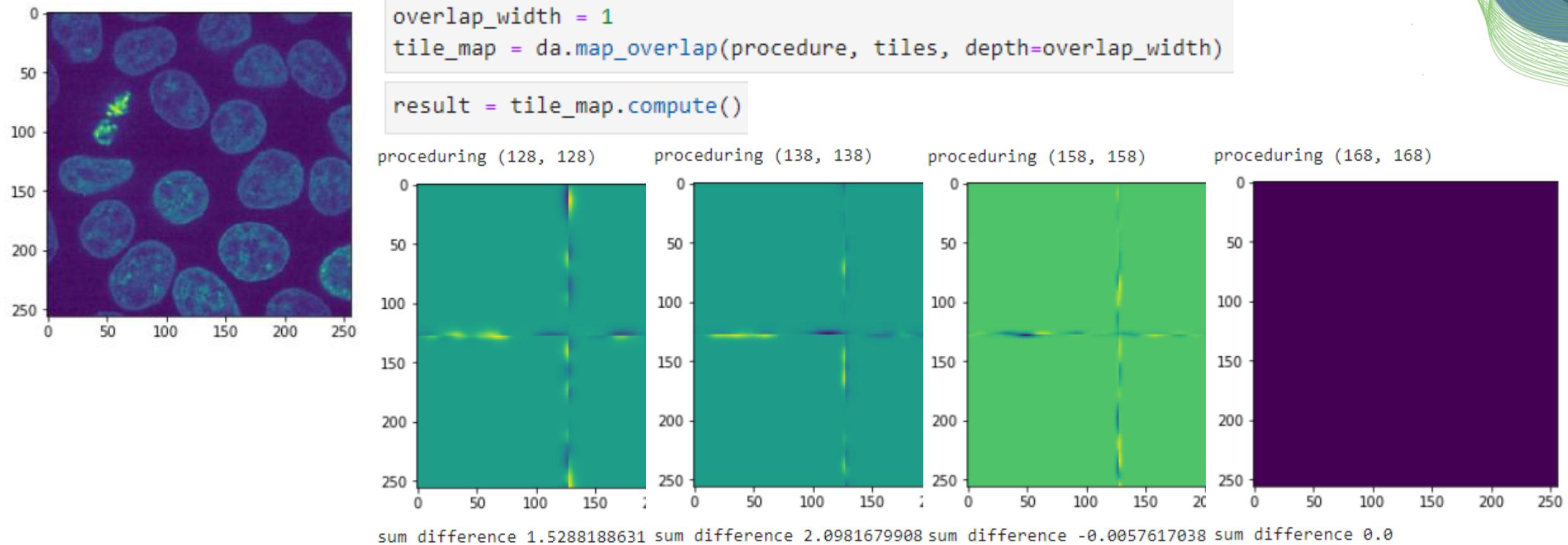


```
tile_map = da.map_blocks(procedure, tiles)  
  
result = tile_map.compute()
```



Tiling with/out overlap

- Processing of images in tiles: artifacts at tile borders





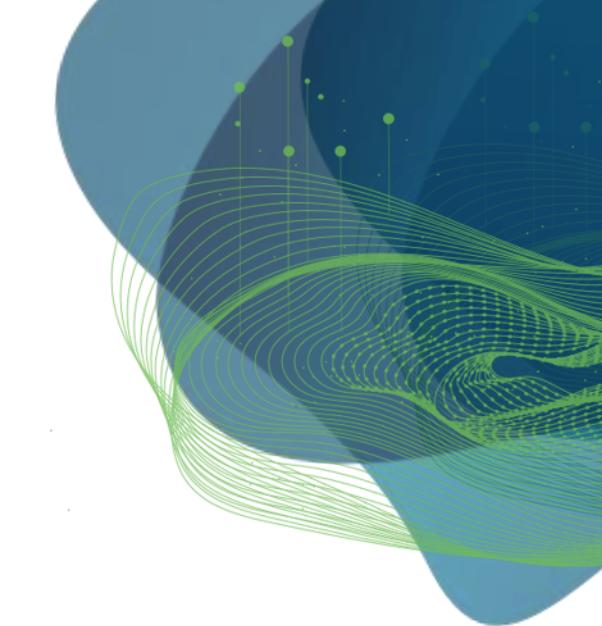
DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS
AND ARTIFICIAL INTELLIGENCE

Segmentation quality estimation

Robert Haase

Reusing materials from Lena Maier-Hein, Annika Reinke (DKFZ) et al.
and Martin Schätz (Charles Uni Prague)



GEFÖRDERT VOM



Bundesministerium
für Bildung
und Forschung



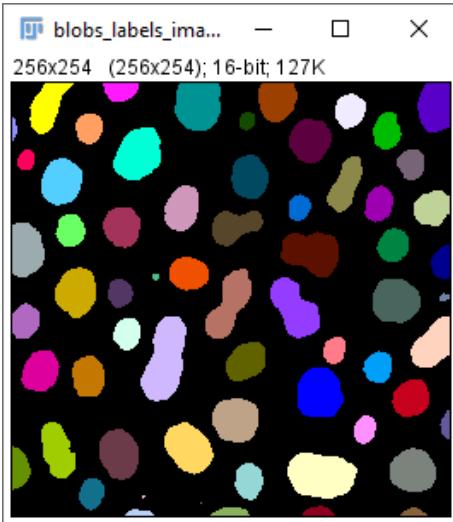
Diese Maßnahme wird gefördert durch die Bundesregierung
aufgrund eines Beschlusses des Deutschen Bundestages.
Diese Maßnahme wird mitfinanziert durch Steuermittel auf
der Grundlage des von den Abgeordneten des Sächsischen
Landtags beschlossenen Haushaltes.

Goal

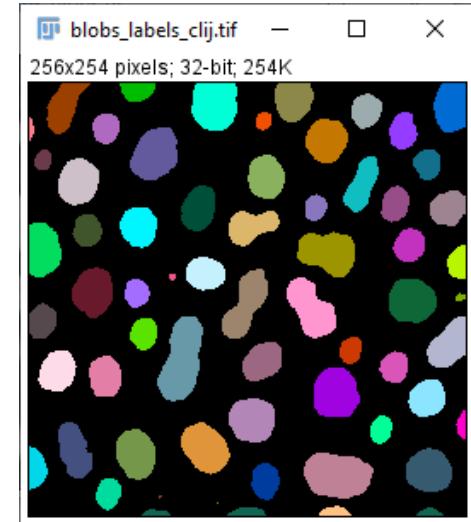
- Compare label images quantitatively, to know
 - how “good” a segmentation algorithms is and/or
 - how “variable” segmentations (from humans or computers) are

How can we know if
these results are the
same?

Human
annotation
("ground truth")



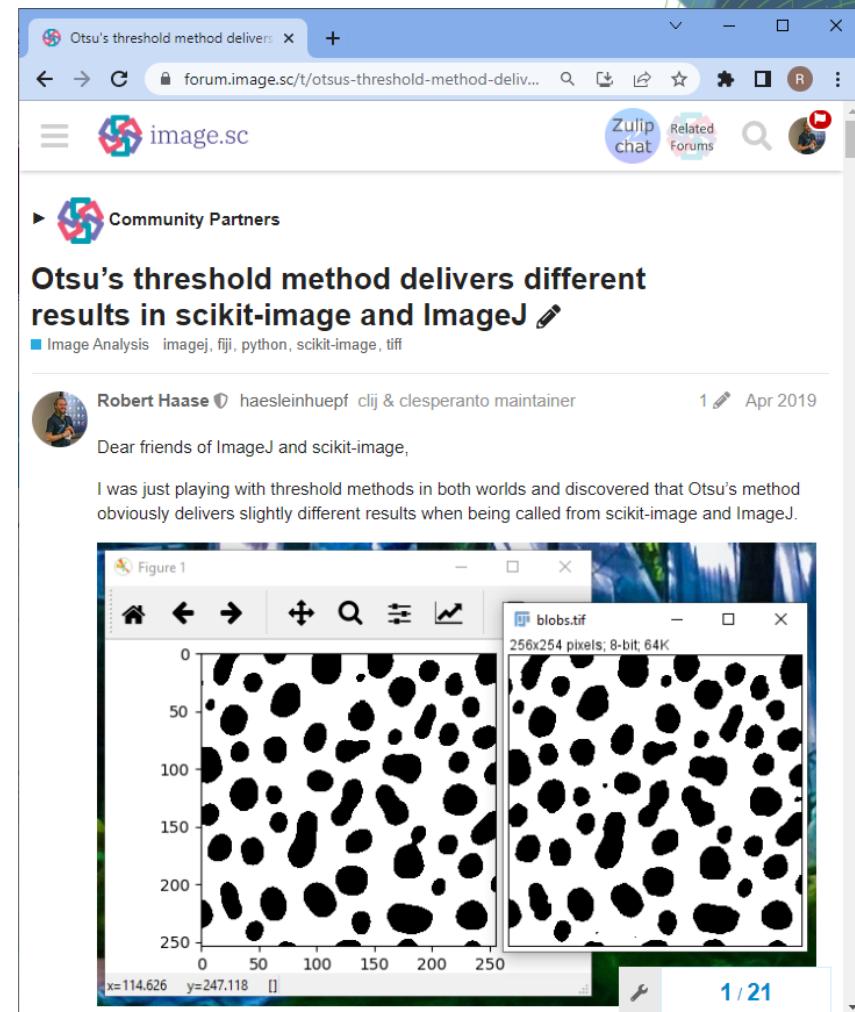
Algorithm result



Why do results vary?

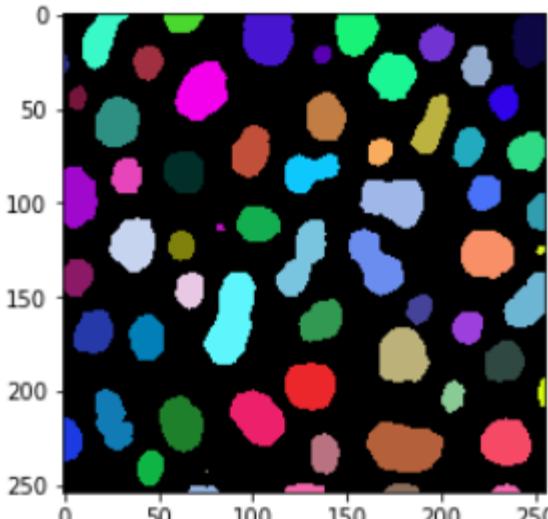
Potential reasons of same workflows delivering different results:

- Image data type (8/16/32-bit float/int)
- Workflow implementation
- How histograms are determined
- How the threshold is determined from the histogram
- Compute architecture (CPU, GPU, TPU, ...)
- Hardware vendor
- Software / driver versions

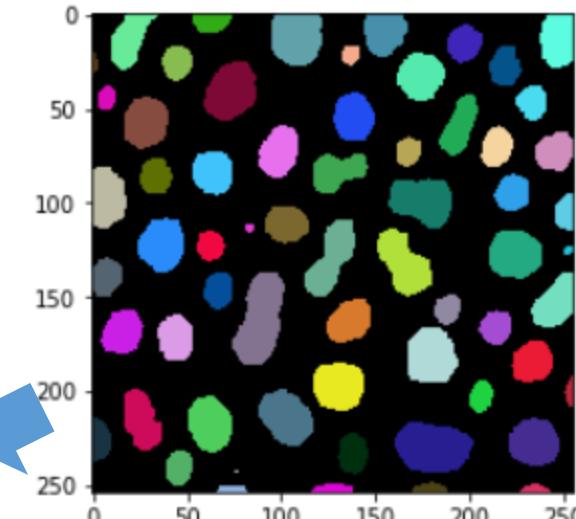
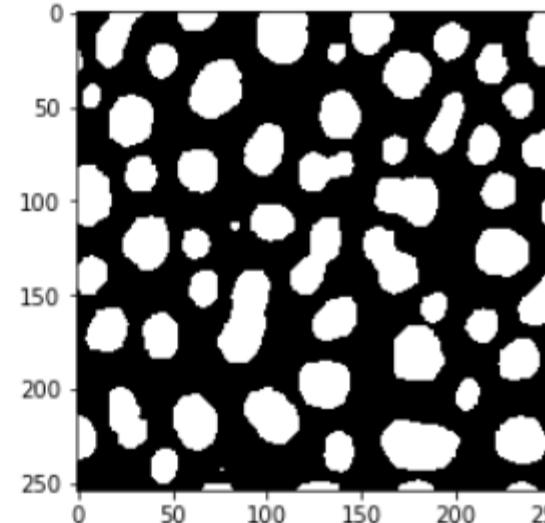
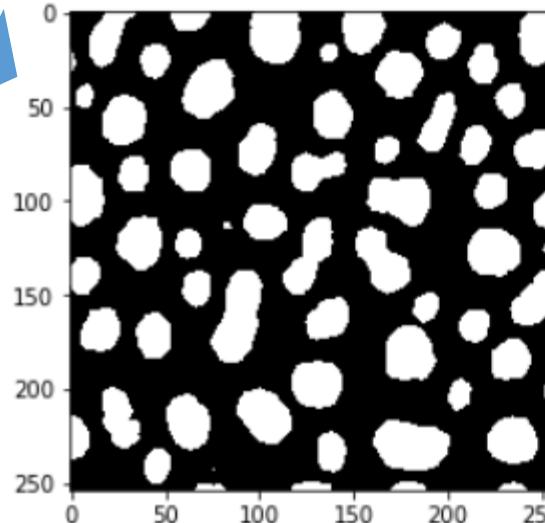


Visual comparison

- The order in label images may be different. To compare them visually, we need to turn them into binary images first.



```
binary_blobs_imagej = imread(filenames[0]) > 0  
binary_blobs_skimage = imread(filenames[1]) > 0  
imshow(binary_blobs_imagej)  
imshow(binary_blobs_skimage)
```



Quantitative comparison

- Voxel-wise Youden-Index

$$YI = p_{TP} + p_{TN} - 1$$

- Volume error

$$\Delta_V = V_A - V_B$$

$$\delta_V = \frac{\Delta_V}{V_B}$$

- Dice Index

$$DI(A, B) = \frac{2 |A \cap B|}{|A| + |B|}$$

- Jaccard Index

$$JI(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{DI}{2 - DI}$$

- Contour distance

$$d_{e,min}(a, B) = \min(d_e(a, b) | b \in B)$$

$$\bar{d}_c(A, B) = \frac{\sum_{\forall a \in C(A)} d_{e,min}(a, C(B))}{|C(A)|}$$

$$\bar{d}_{bil,c}(A, B) = \frac{\bar{d}_c(A, B) + \bar{d}_c(B, A)}{2}$$

- Hausdorff distance

$$d_H(A, B) = \max(d_{e,min}(a, B) | a \in A)$$

$$d_{bil,H}(A, B) = \max(d_H(A, B), d_H(B, A))$$

- Simplified Hausdorff distance

$$d_H(A, B) = \max(d_{e,min}(a, C(B)) | a \in C(A))$$

- Volume standard deviation

$$\delta_{\bar{V}} = 2 \frac{|V_A - V_B|}{|V_A + V_B|}$$

- Classification error

$$e_{Class} = \frac{H}{|TP| + |FN|}$$

- Hamming distance

$$d_h = |A \cup B| - |A \cap B| \\ = |FP| + |FN|$$

Choosing the right metric is key

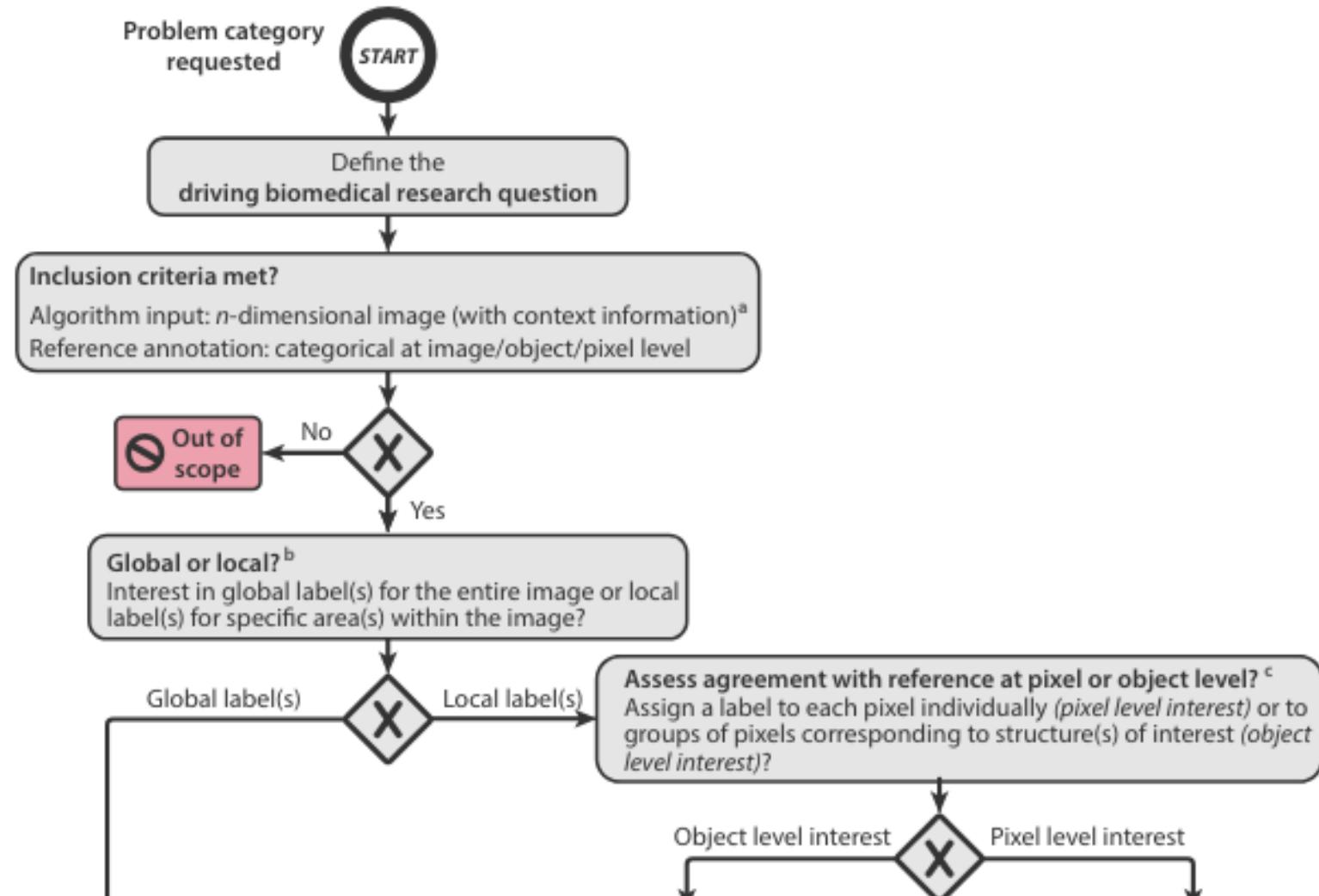
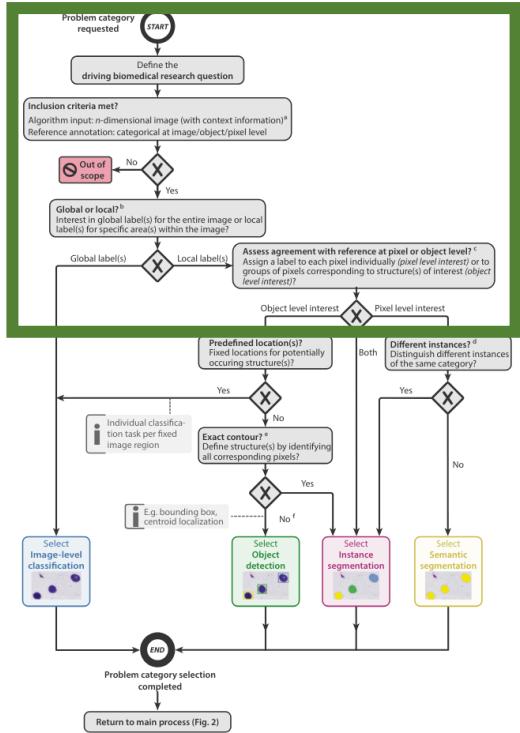
- Systematic overview by Maier-Hein, Reinke et al.

The screenshot shows a browser window for arXiv.org. The URL is arxiv.org/abs/2206.01653. The page title is "Metrics reloaded: Recommendations for image analysis validation". It's a Computer Science paper in Computer Vision and Pattern Recognition. The authors listed are Lena Maier-Hein, Annika Reinke, Patrick Godau, Minu D. Tizabi, Florian Buettner, Evangelia Christodoulou, Ben Glocker, Fabian Isensee, Jens Kleesiek, Michal Kozubek, Mauricio Reyes, Michael A. Riegler, Manuel Wiesenfarth, A. Emre Kavur, Carole H. Sudre, Michael Baumgartner, Matthias Eisenmann, Doreen Heckmann-Nötz, Tim Rätsch, Laura Acion, Michela Antonelli, Tal Arbel, Spyridon Bakas, Arriel Benis, Matthew Blaschko, M. Jorge Cardoso, Veronika Cheplygina, Beth A. Cimini, Gary S. Collins, Keyvan Farahani, Luciana Ferrer, Adrian Galdran, Bram van Ginneken, Robert Haase, Daniel A. Hashimoto, Michael M. Hoffman, Merel Huisman, Pierre Jannin, Charles E. Kahn, Dagmar Kainmueller, Bernhard Kainz, Alexandros Karargyris, Alan Karthikesalingam, Hannes Kenngott, Florian Kofler, Annette Kopp-Schneider, Anna Kreshuk, Tahsin Kurc, Bennett A. Landman, Geert Litjens, Amin Madani, Klaus Maier-Hein, Anne L. Martel, Peter Mattson, Erik Meijering, Bjoern Menze, Karel G.M. Moons, Henning Müller, Brennan Nichyporuk, Felix Nickel, Jens Petersen, Nasir Rajpoot, Nicola Rieke, Julio Saez-Rodriguez, Clara I. Sánchez, Shravya Shetty, Maarten van Smeden, Ronald M. Summers, Abdel A. Taha, Aleksei Tiulpin, Sotiris A. Tsafaris, Ben Van Calster, Gaël Varoquaux, Paul F. Jäger. The abstract discusses flaws in machine learning (ML) algorithm validation and the need for better metrics. The page includes a sidebar with download options (PDF, HTML, Tex Source, Other Formats), a CC-BY license, and citation links to NASAADS, Google Scholar, and Semantic Scholar.

The screenshot shows a browser window for nature.com. The URL is nature.com/articles/s41592-023-02151-z. The page title is "Metrics reloaded: recommendations for image analysis validation". It's published in Nature Methods, Perspectives section, article. The authors listed are Lena Maier-Hein, Annika Reinke, Patrick Godau, Minu D. Tizabi, Florian Buettner, Evangelia Christodoulou, Ben Glocker, Fabian Isensee, Jens Kleesiek, Michal Kozubek, Mauricio Reyes, Michael A. Riegler, Manuel Wiesenfarth, A. Emre Kavur, Carole H. Sudre, Michael Baumgartner, Matthias Eisenmann, Doreen Heckmann-Nötz, Tim Rätsch, Laura Acion, Michela Antonelli, Tal Arbel, Spyridon Bakas, Arriel Benis, ... Paul F. Jäger. The page includes a sidebar with a download button for PDF, associated content links for "The future of bioimage analysis" and "Understanding metric-related pitfalls in image analysis validation", and a focus section.

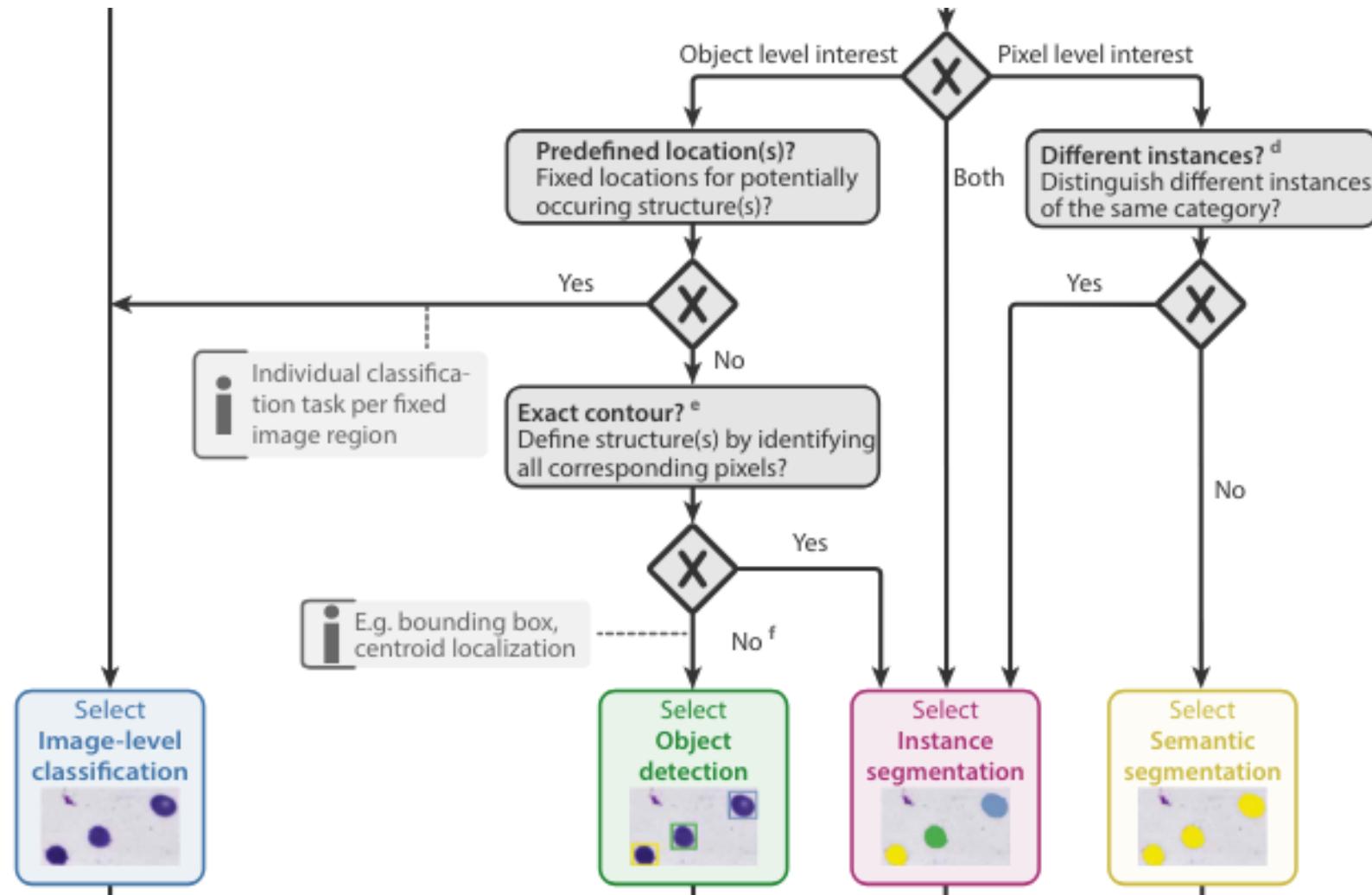
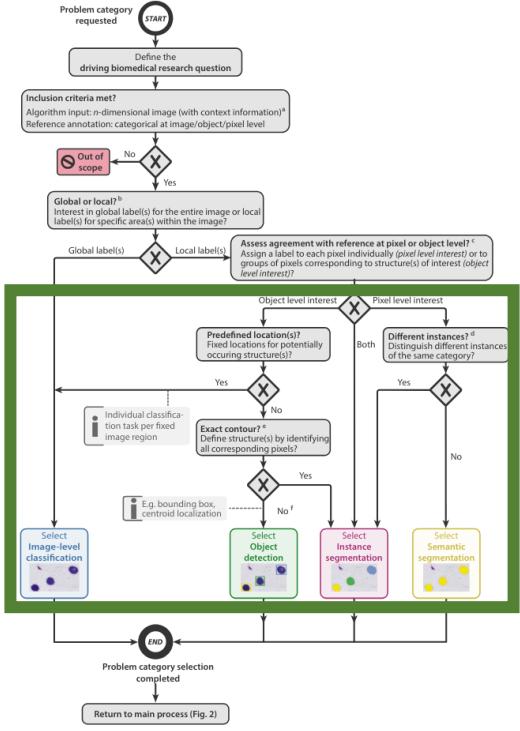
Choosing the right metric is key

+ S1



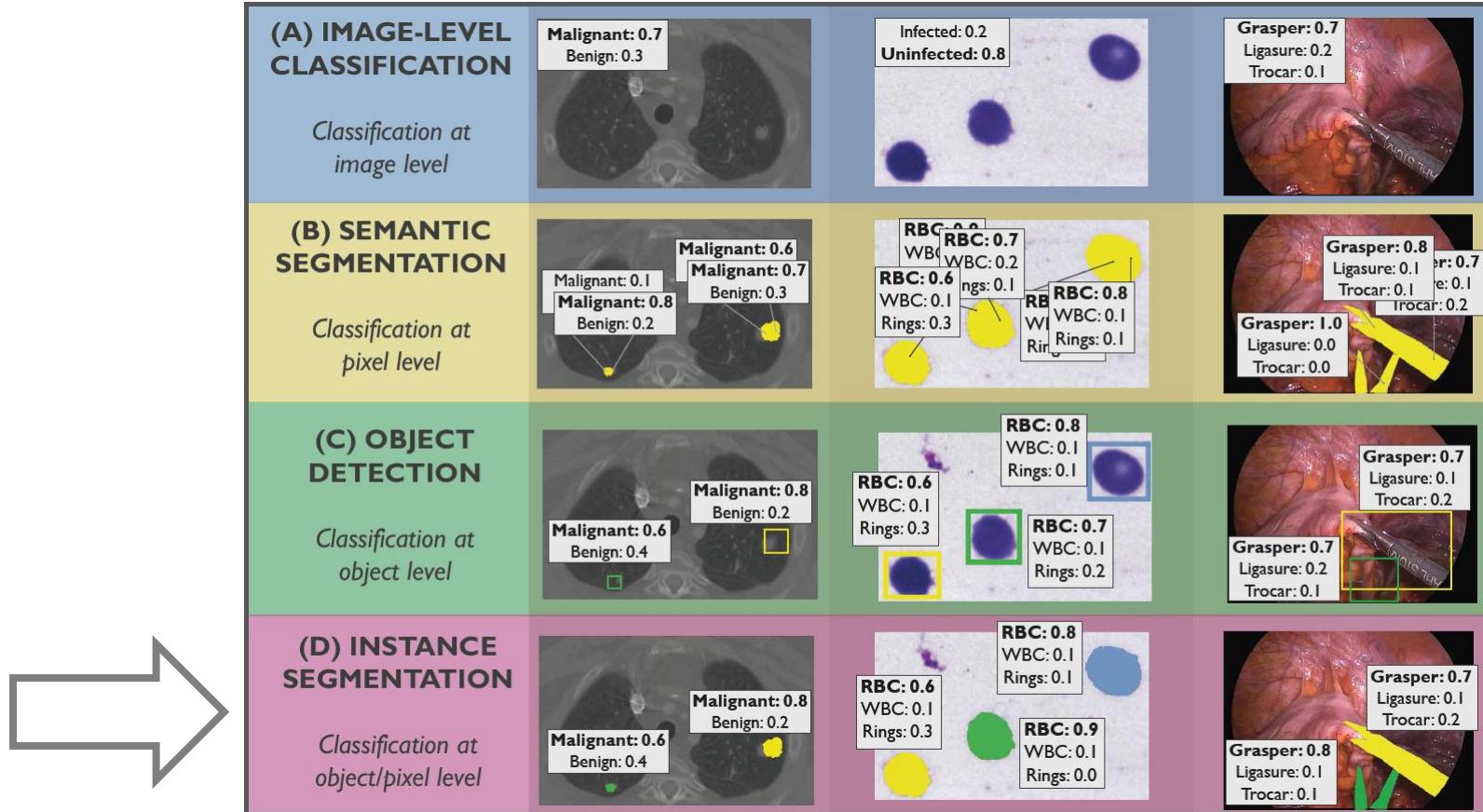
Choosing the right metric is key

+ S1



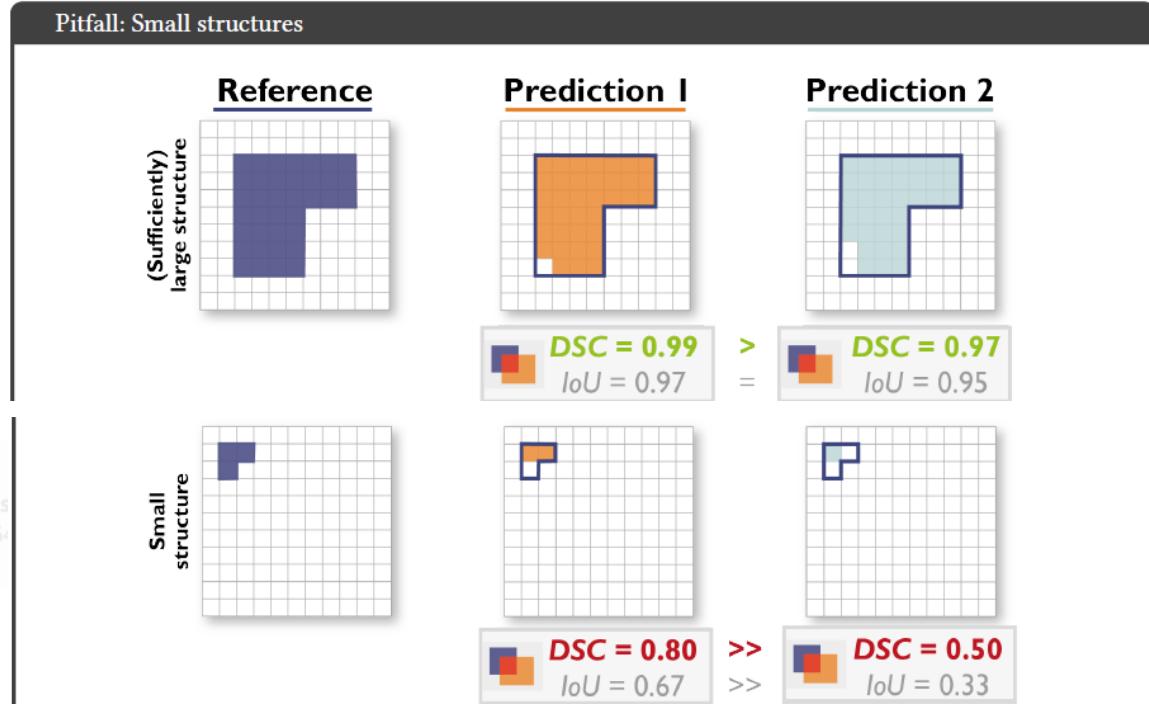
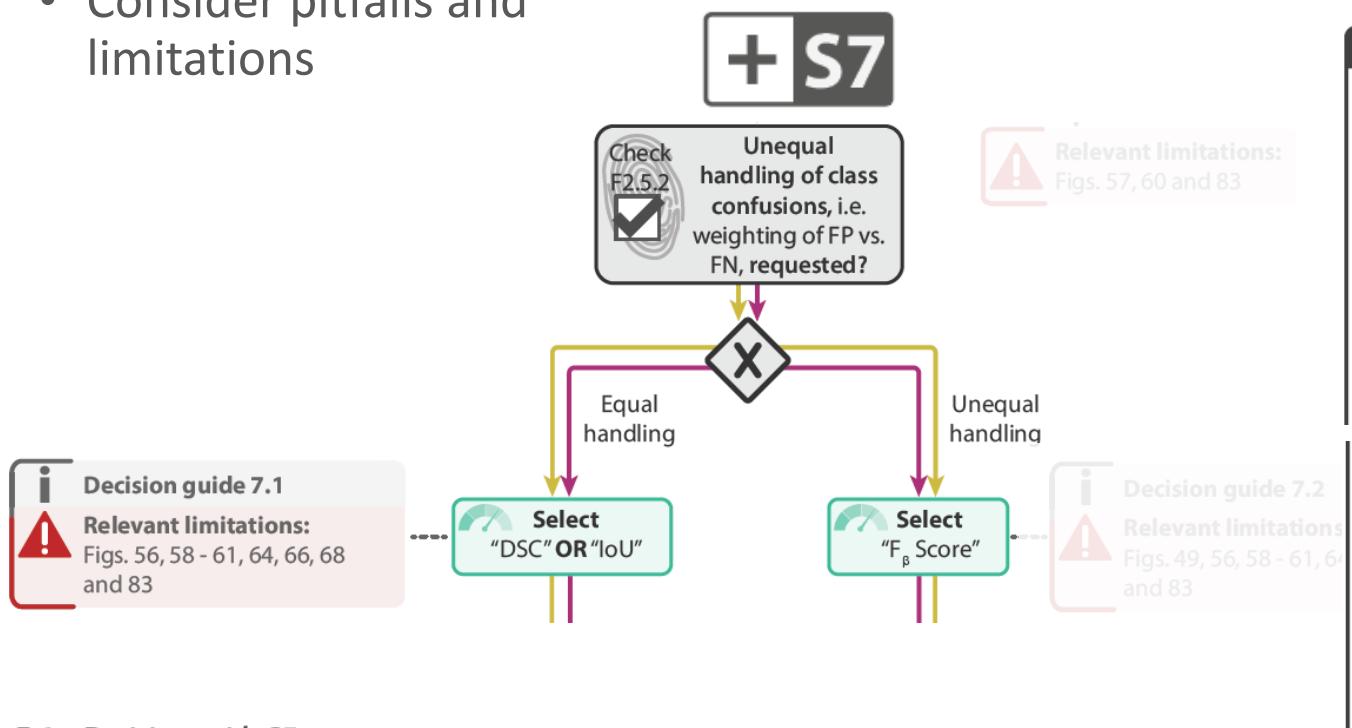
Choosing the right metric is key

- Define your question



Overlap metrics

- Consider pitfalls and limitations



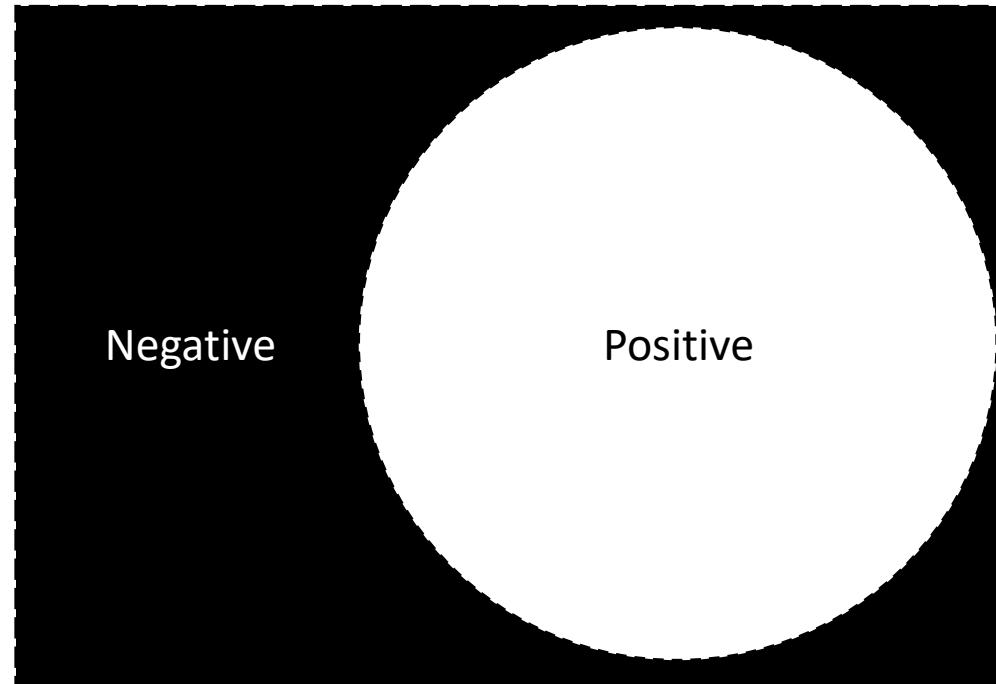
E.6 Decision guide S7.

D7.1: Dice vs IoU

The DSC is identical to the F_1 Score on pixel level and closely related to the IoU, which, in turn, is identical to the Jaccard Index (see equations 5 and 6). The two metrics will yield the same ranking (of aggregated metric values) in most applications (theoretically, deviations are possible), such that there is no value in combining them. Commonly, the computer vision community prefers the IoU, while the medical image community favors the DSC.

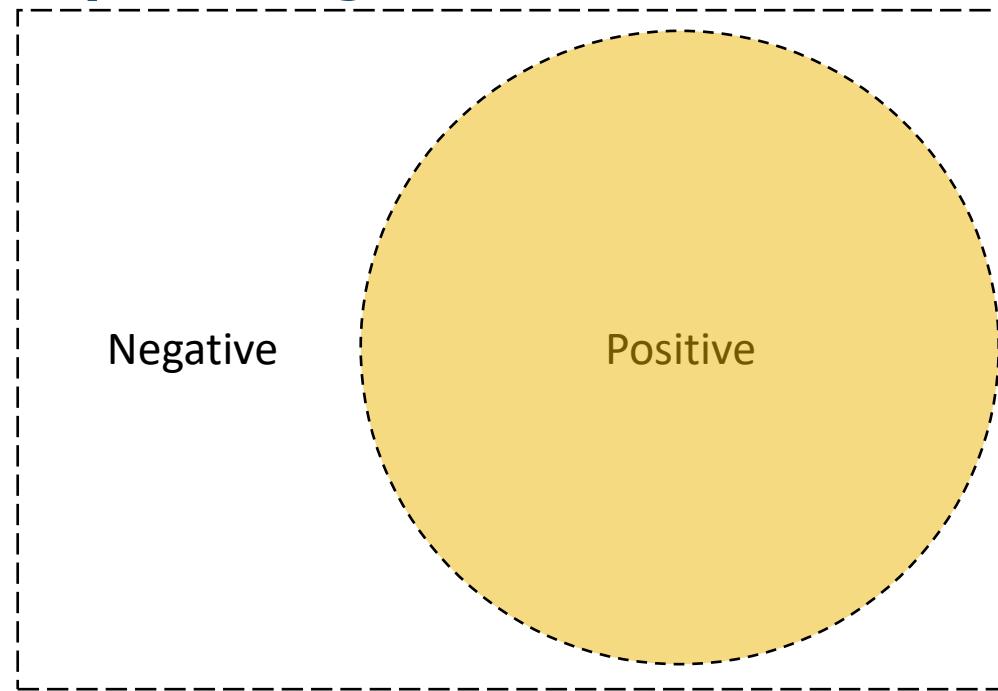
Segmentation quality estimation

- In general
 - Define what's positive and what's negative.



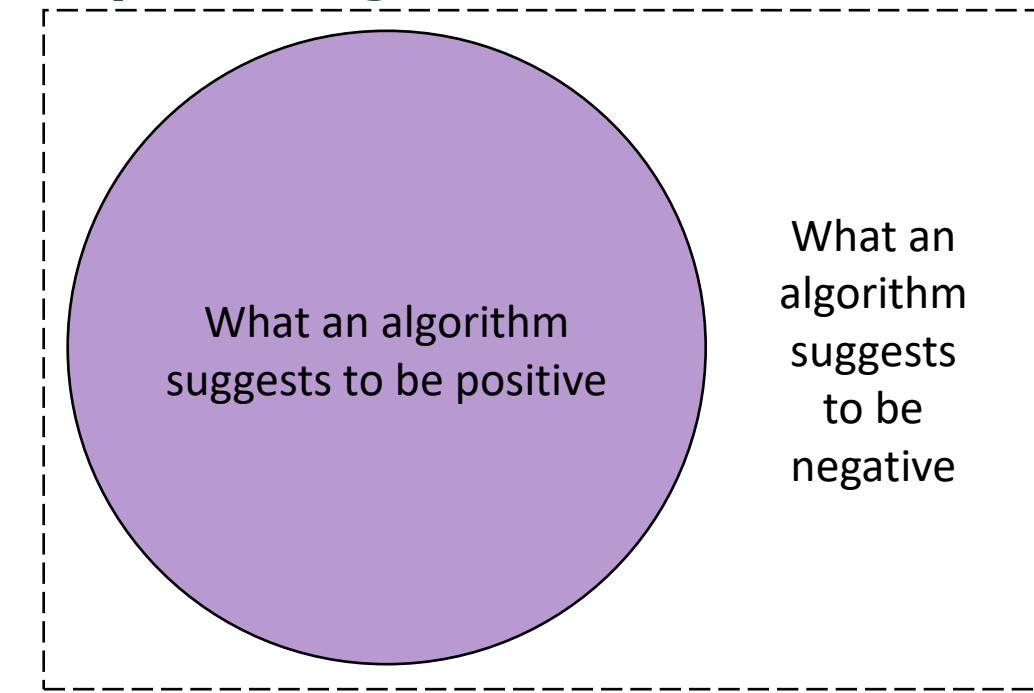
Segmentation quality estimation

- In general
 - Define what's positive and what's negative.



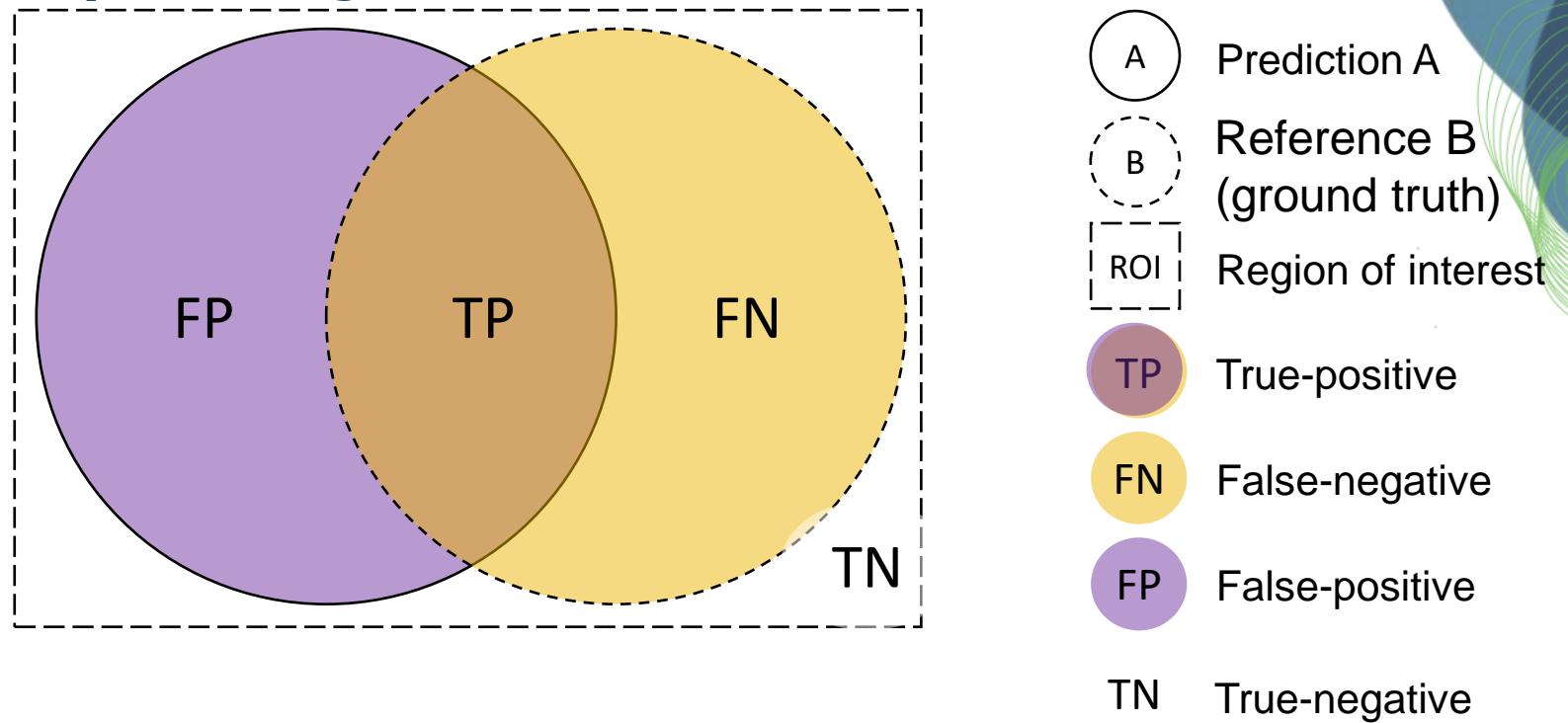
Segmentation quality estimation

- In general
 - Define what's positive and what's negative.



Segmentation quality estimation

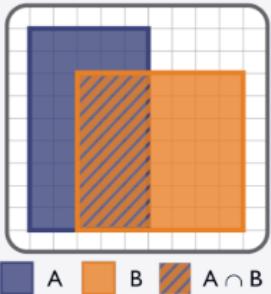
- In general
 - Define what's positive and what's negative.
 - Compare with a reference to figure out what was true and false
- Welcome to the Theory of Sets



Dice versus Jaccard Index (IoU)

DICE SIMILARITY COEFFICIENT (DSC)

Synonyms: Dice, Dice Coefficient, Sørensen–Dice Coefficient, F_1 Score, Balanced F Score



$$\text{DSC}(A, B) = \frac{2 |A \cap B|}{|A| + |B|} = \frac{2 \text{PPV} \cdot \text{Sensitivity}}{\text{PPV} + \text{Sensitivity}}$$

VALUE RANGE: [0, 1] ↑

DESCRIPTION

DSC measures the overlap between two structures.

DEFINITION

[Dice, 1945]

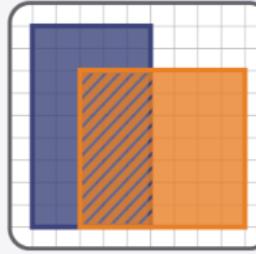
	RECOMMENDED FOR
ImLC	<input checked="" type="radio"/>
SemS	<input checked="" type="radio"/>
ObD	<input checked="" type="radio"/>
InS	<input checked="" type="radio"/>

RECOMMENDATIONS

- An overlap-based metric (by default the DSC or IoU) should be used in most cases of segmentation assessment. An exception is the case of consistently tiny structures along with a noisy reference.
- DSC should generally be used in combination with a boundary-based metric if boundaries are of interest.
- DSC should generally not be considered if...
 - ... there is a high variability of structure sizes within an image or across images.
 - ... inter-rater variability is requested to be compensated.
 - ... over- and undersegmentation should be treated similarly.
- DSC should be considered as a metric in the medical community rather than in the computer vision and biology communities (where the almost identical IoU is preferred).

INTERSECTION OVER UNION (IoU)

Synonyms: Jaccard Index, Tanimoto Coefficient



$$\begin{aligned}\text{IoU}(A, B) &= \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \\ &= \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \\ &= \frac{\text{PPV} \cdot \text{Sensitivity}}{\text{PPV} + \text{Sensitivity} - \text{PPV} \cdot \text{Sensitivity}}\end{aligned}$$

VALUE RANGE: [0, 1] ↑

DESCRIPTION

IoU measures the overlap between two structures. It is often referred to as **Box IoU** when comparing bounding boxes, **Mask IoU** when comparing segmentation masks, or **Approx IoU** when comparing approximations of objects beyond bounding boxes.

DEFINITION

[Jaccard, 1912]

RECOMMENDED FOR

ImLC	SemS	ObD	InS
<input checked="" type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>

IMPORTANT RELATIONS

$$\text{IoU} = \frac{\text{DSC}}{2 - \text{DSC}} \quad \text{IoU} = \frac{F_\beta}{2 - F_\beta} \quad \text{for } \beta = 1$$

RECOMMENDATIONS

- An overlap-based metric (by default DSC or IoU) should be used in most cases for segmentation assessment. An exception is the case of consistently tiny structures along with a noisy reference.
- IoU should generally be used in combination with a boundary-based metric if boundaries are of interest.
- IoU should generally not be considered if...
 - ... there is a high variability of structure sizes within an image or across images.
 - ... inter-rater variability is requested to be compensated.
 - ... over- and undersegmentation should be treated similarly.
- IoU should be considered as a metric in the computer vision and biology communities rather than in the medical community (which prefers the almost identical DSC).

Dice versus Jaccard Index (IoU)

2.7.5 Decision guide S6.

DG6.1: Dice Similarity Coefficient (DSC) versus Intersection over Union (IoU)

Summary of DG6.1: DSC versus IoU

DSC

- ⇒ Identical to F_1 Score
- ⇒ Close relation to IoU (see Eq. 5)
- ⇒ Preference in medical community

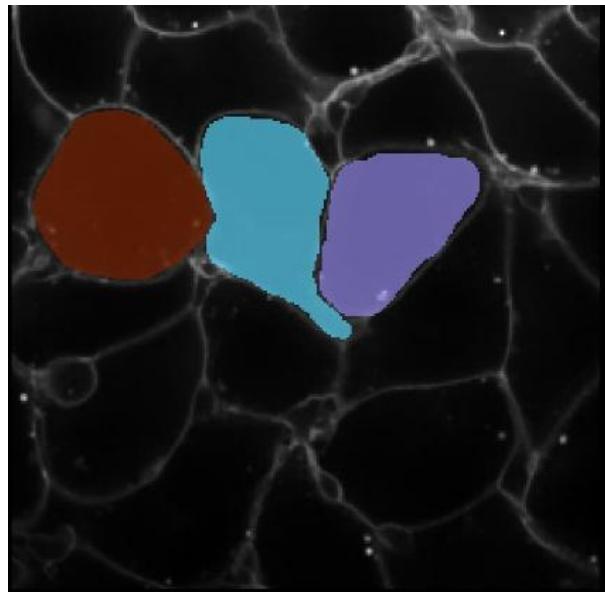
IoU

- ⇒ Identical to Jaccard Index
- ⇒ Close relation to DSC (see Eq. 4)
- ⇒ Preference in computer vision community

Extended Data Tab. SN 2.18. Comparison of Dice Similarity Coefficient (DSC) and Intersection over Union (IoU) in the context of the decision guide DG6.1 for Subprocess S6. Context: no exclusive interest in the center line of structures (FP2.3 = FALSE, FP3.3 = FALSE) and equal severity of class confusions (FP2.5.2 = FALSE).

Sparse Jaccard Index

- For every annotated object, we compute the maximum IoU with any segmented object.
- We average this value over all annotated objects



Sparse
instance annotation



IoU = 0.35



IoU = 0.66

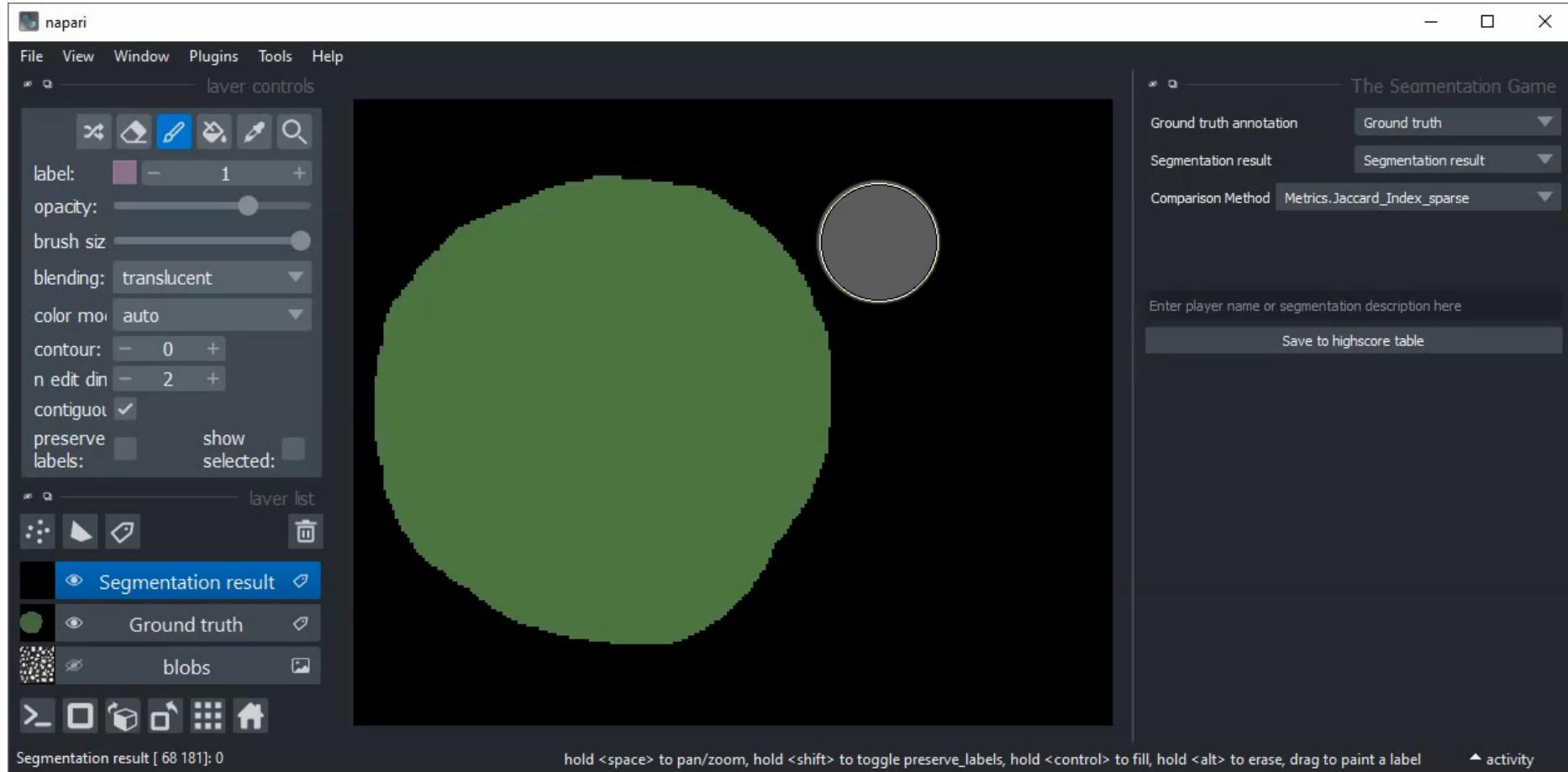


IoU = 0.69

Instance segmentation candidates

Sparse Jaccard Index

- Play with metrics to gain understanding

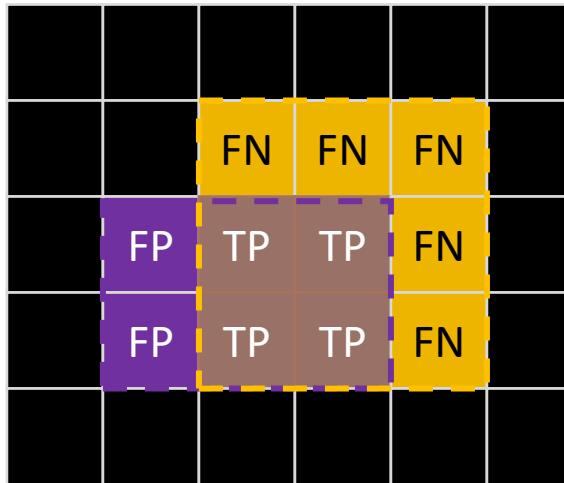
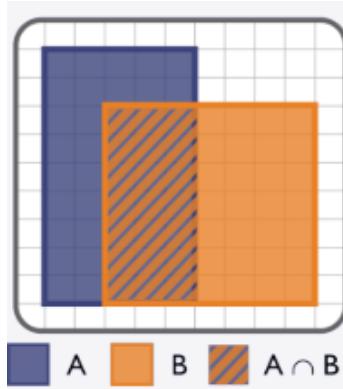
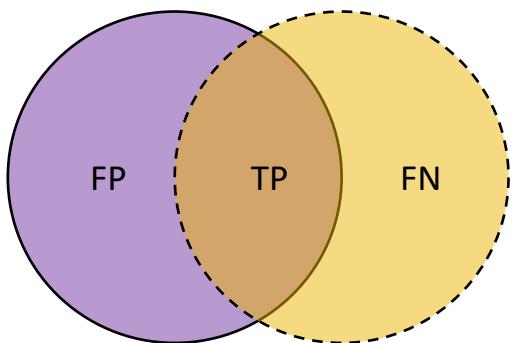


With Martin Schätz
(Charles Uni, Prague)
@schatzcz

Pixel-wise versus Object-wise evaluation

- Pixel wise: Segmentation quality

Prediction Ground truth



True-positive: 4

False-negative: 5

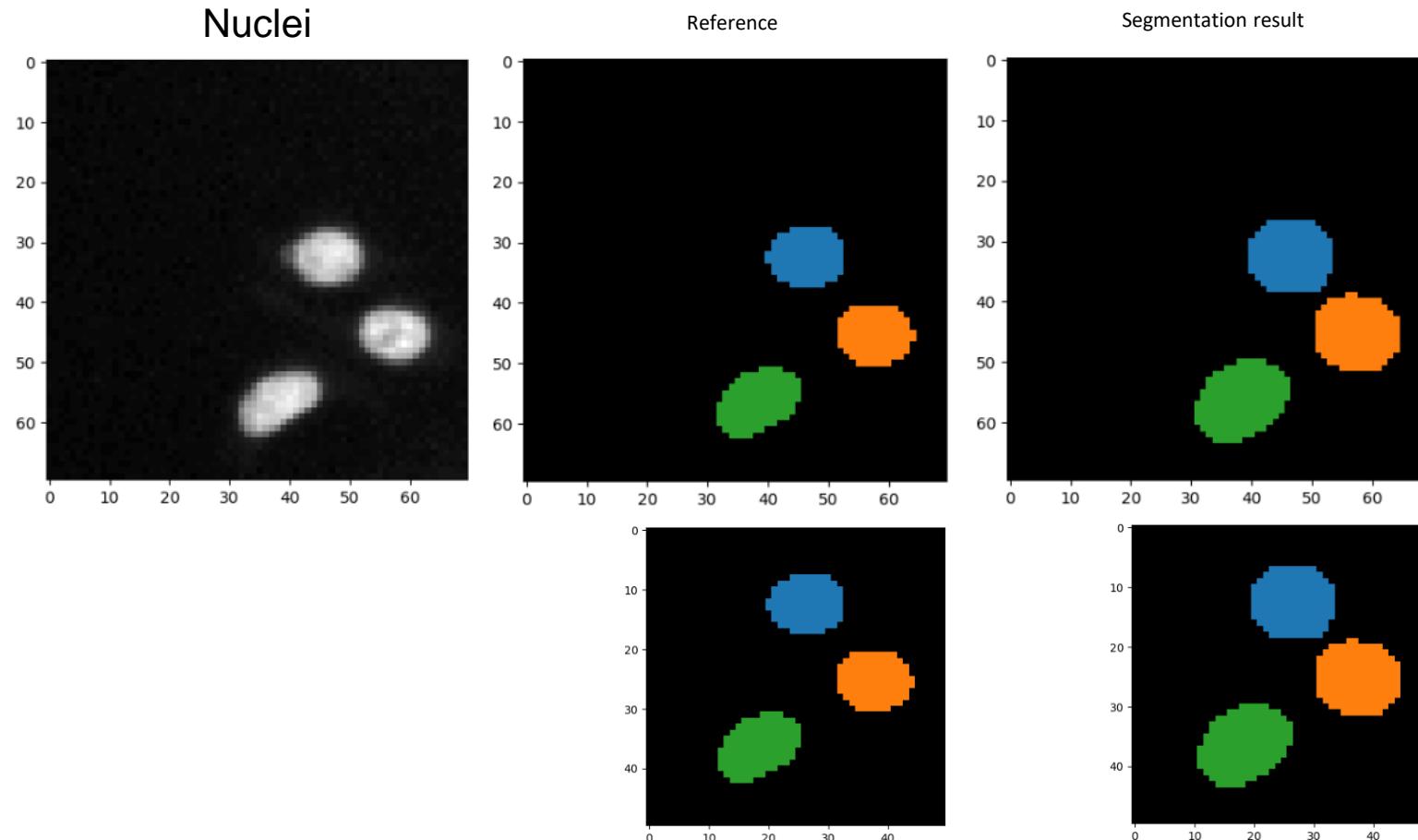
False-positive: 2

$$\begin{aligned} \text{IoU}(A,B) &= \frac{\text{A} \cap \text{B}}{\text{A} + \text{B} - \text{A} \cap \text{B}} \\ &= \frac{|\text{A} \cap \text{B}|}{|\text{A}| + |\text{B}| - |\text{A} \cap \text{B}|} = \frac{|\text{A} \cap \text{B}|}{|\text{A} \cup \text{B}|} \end{aligned}$$

IoU = 4 / 11

Accuracy versus Jaccard Index (IoU)

- Side-effects of image size and number of nuclei



$$A = \frac{TP + TN}{FN + FP + TP + TN}$$

$$J = \frac{TP}{FN + FP + TP}$$

Accuracy: 0.97

Jaccard Index: 0.73

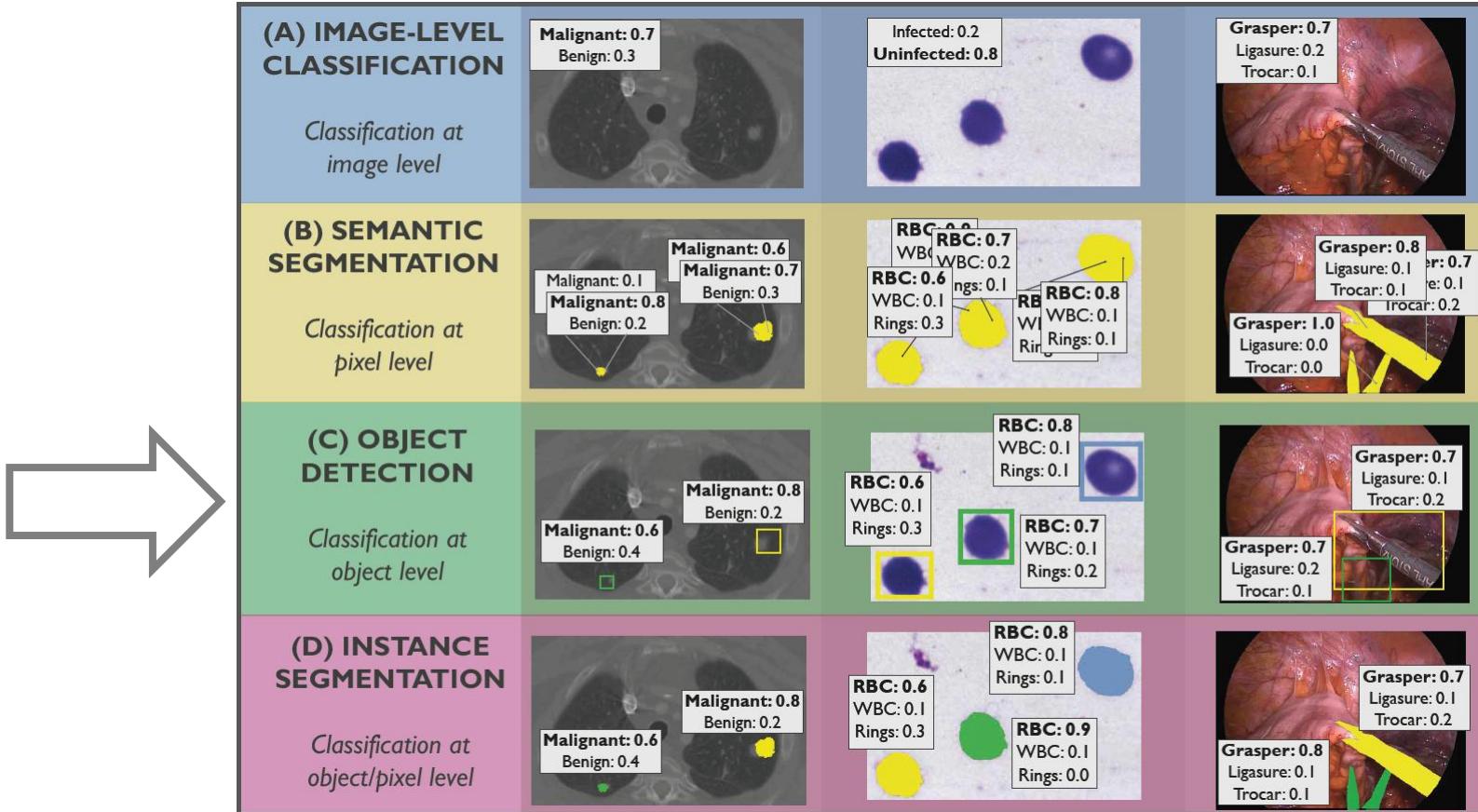
Accuracy decreases because
there are less correct black
pixels (TN)

Accuracy: 0.95

Jaccard Index: 0.73

Choosing the right metric is key

- Define your question



Pixel-wise versus Object-wise evaluation

- Are these objects overlapping?

		FN	FN			
FP	TP	FN				
FP	FP					

$$\text{IoU} = 1 / 7$$

		FP	TP	FN		
		FP	TP	FN		

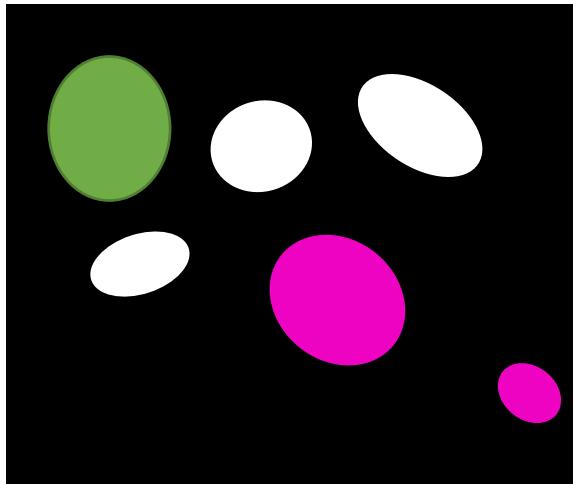
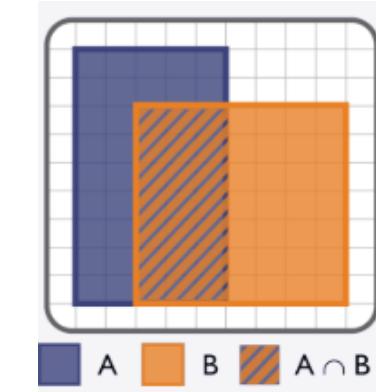
$$\text{IoU} = 1 / 3$$

		FP	TP	TP	FN	
		FP	TP	TP	FN	

$$\text{IoU} = 1 / 2$$

Pixel-wise versus Object-wise evaluation

- Object wise: Detection quality



True-positive: 3

False-negative: 1

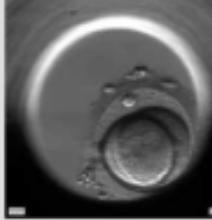
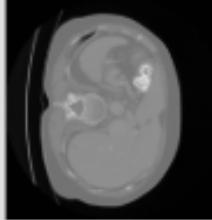
False-positive: 2

$$\begin{aligned} \text{IoU}(A, B) &= \frac{\text{A} \cap \text{B}}{\text{A} + \text{B} - \text{A} \cap \text{B}} \\ &= \frac{|\text{A} \cap \text{B}|}{|\text{A}| + |\text{B}| - |\text{A} \cap \text{B}|} = \frac{|\text{A} \cap \text{B}|}{|\text{A} \cup \text{B}|} \end{aligned}$$

$\text{IoU} = 1 / 2$

Special case in medical imaging

- We elaborate segmentation quality of one / large object:

PROBLEM DESCRIPTION	ID	SCENARIO	SAMPLE INPUT IMAGE	RECOMMENDED OUPUT	RECOMMENDATION
Segmentation of large objects	SemS-1	Embryo segmentation from microscopy images			<p>Problem category: Semantic segmentation</p> <p>Overlap-based metric (S6): Dice Similarity Coefficient (DSC)</p> <p>Boundary-based metric (S7): Normalized Surface Distance (NSD)</p> <p>Specific property-related metric: Liver segmentation: Absolute Volume Difference</p>
	SemS-2	Liver segmentation in computed tomography (CT) images			

Further reading

The screenshot shows a web browser with two tabs open. The left tab displays a blog post from the FocalPlane website. The right tab provides a detailed explanation of the Jaccard index.

Left Tab: Quality assurance of segmentation results

- Posted by [Mara Lampert](#), on 13 April 2023
- This blog post revolves around determining and improving the quality of segmentation results. A common problem is that this step is often omitted and done rather than by actually quantifying it. The blogpost achieves this quantification as this leads to reproducibility.

Right Tab: The Jaccard index

The **Jaccard Index** is a measure to investigate the similarity or difference of sample sets. Hereby, the so called **sparse Jaccard Index** measures the overlap lying between 0 (no overlap) and 1 (perfect overlap). Therefore, the ground truth label is compared to the segmented label by determination of the maximum overlap. The metric result is the mean overlap of all investigated labels (Jaccard, 1902) ([see this jupyter notebook](#)).

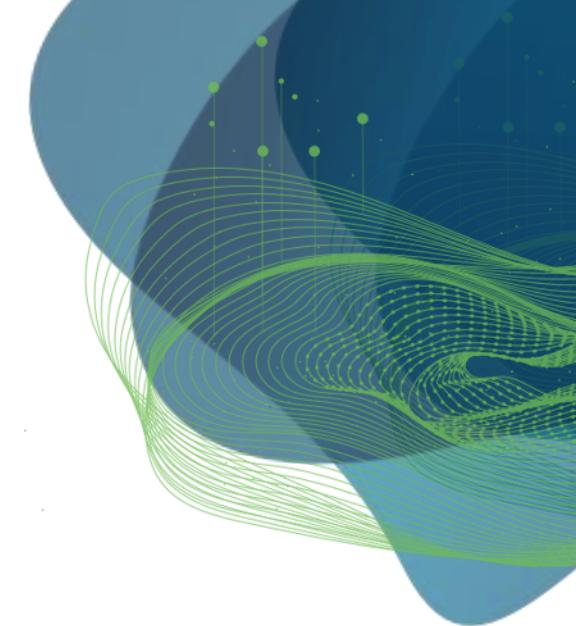


Mara Lampert
@marabuuuu.bsky.social



DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS
AND ARTIFICIAL INTELLIGENCE



Exercises

Robert Haase

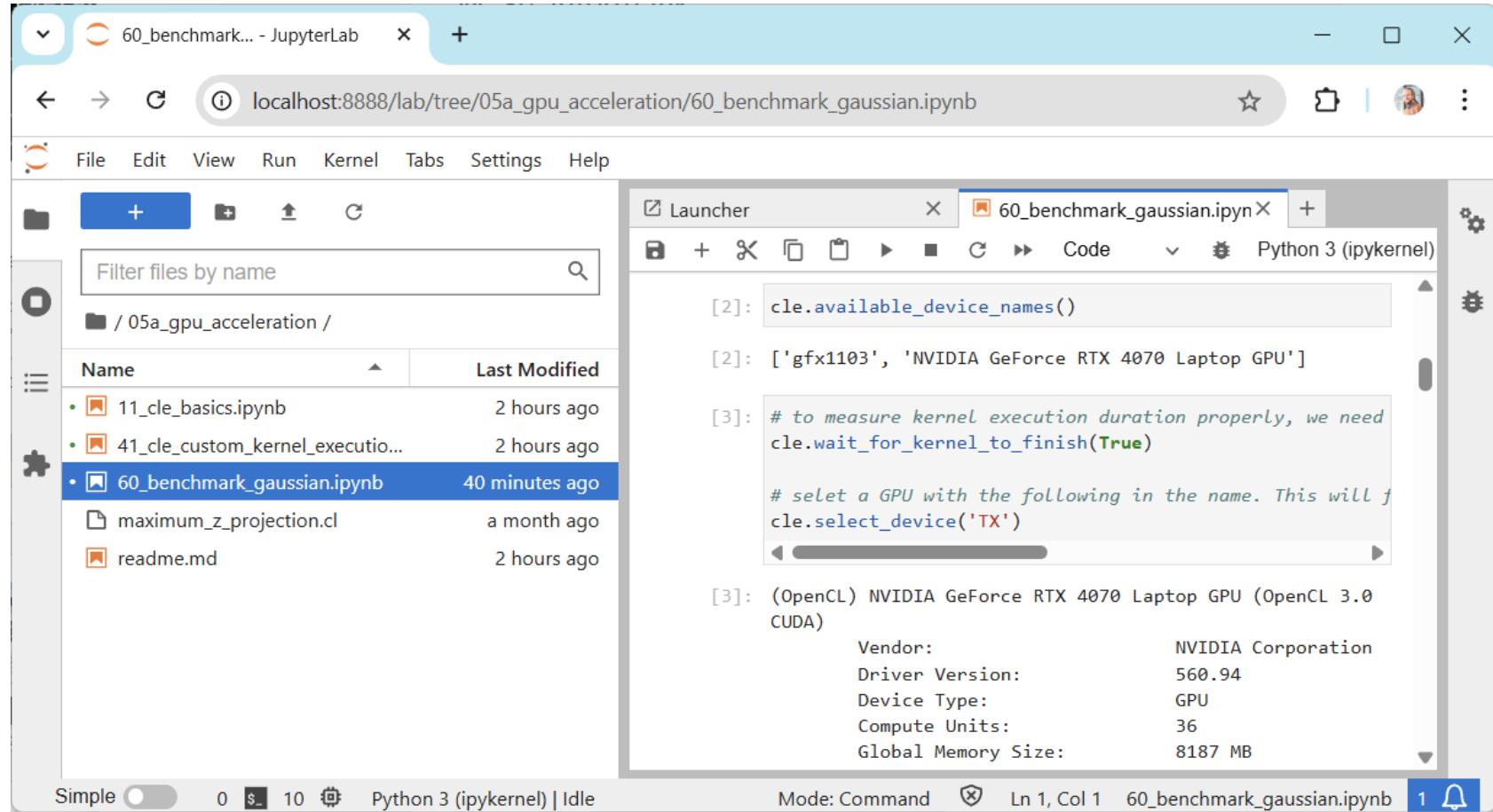
GEFÖRDERT VOM



Bundesministerium
für Bildung
und Forschung

Exercise: GPU-accelerated image processing

- Compare CPU processing speed with a GPU, consider using the cluster for this



The screenshot shows a JupyterLab interface. On the left, a file tree displays a directory structure under '/05a_gpu_acceleration/'. The file '60_benchmark_gaussian.ipynb' is selected. In the main area, a code editor shows Python code using the cle library to interact with a GPU. The code includes:

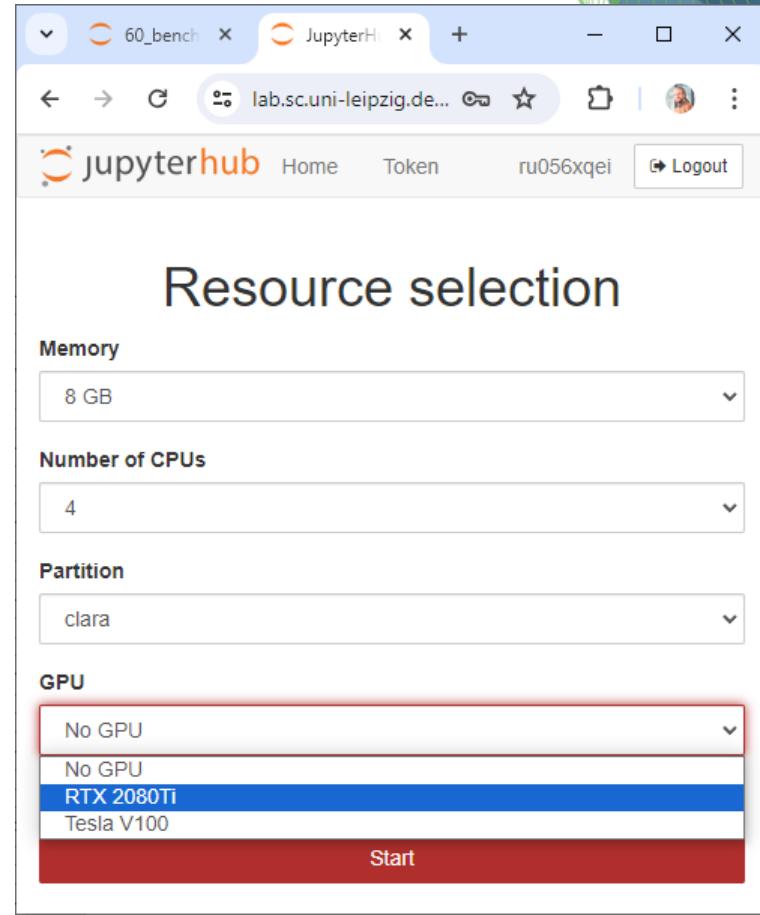
```
cle.available_device_names()
['gfx1103', 'NVIDIA GeForce RTX 4070 Laptop GPU']

# to measure kernel execution duration properly, we need
cle.wait_for_kernel_to_finish(True)

# select a GPU with the following in the name. This will f
cle.select_device('TX')

(OpenCL) NVIDIA GeForce RTX 4070 Laptop GPU (OpenCL 3.0
CUDA)
Vendor: NVIDIA Corporation
Driver Version: 560.94
Device Type: GPU
Compute Units: 36
Global Memory Size: 8187 MB
```

The status bar at the bottom indicates 'Python 3 (ipykernel) | Idle'.



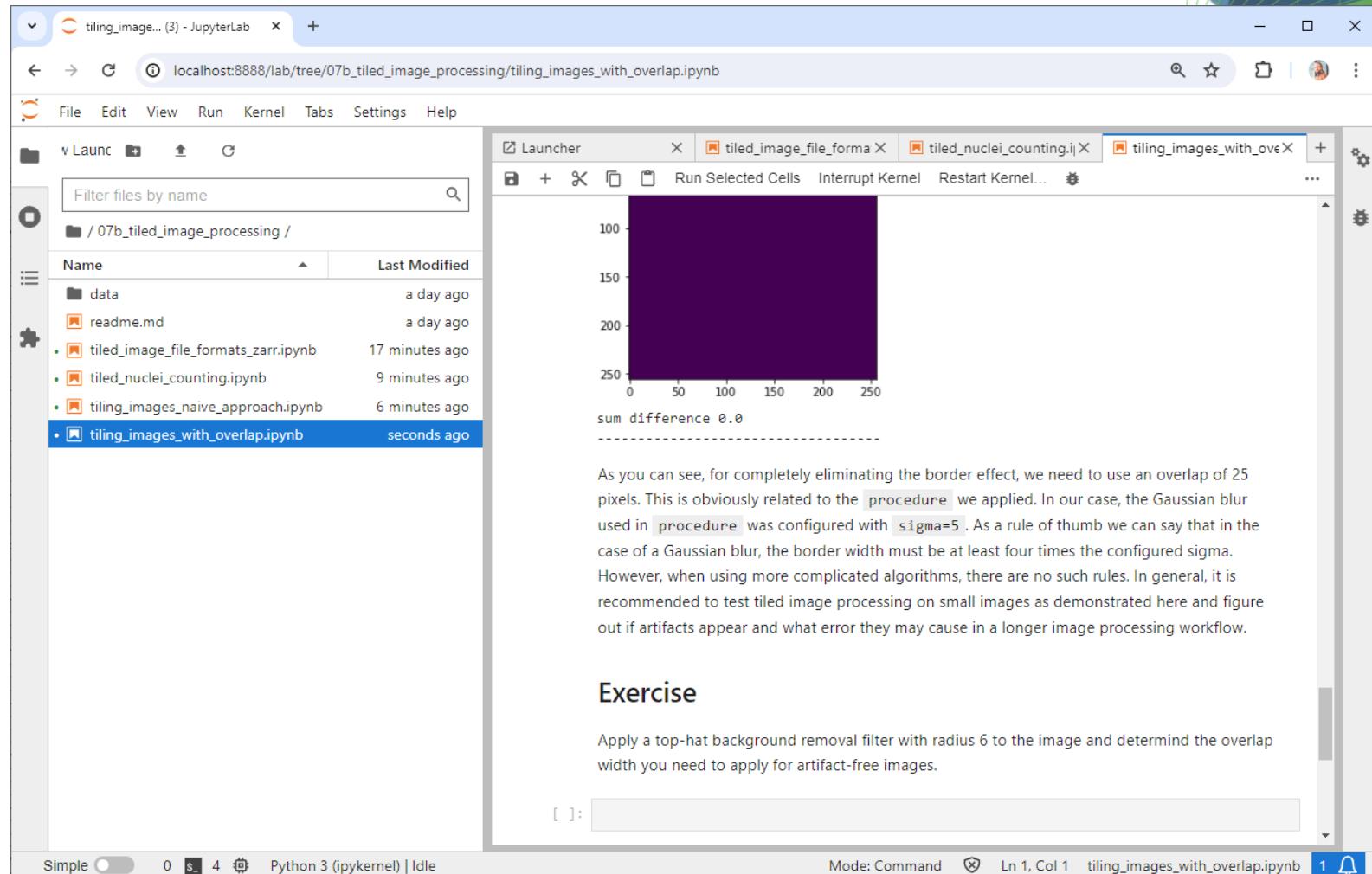
The screenshot shows a JupyterHub interface. A 'Resource selection' dialog is open, showing configuration for a job. The 'GPU' dropdown menu is highlighted and contains the following options:

- No GPU
- No GPU
- RTX 2080Ti** (selected)
- Tesla V100

Below the dropdown is a red 'Start' button.

Exercise: Tiled image processing

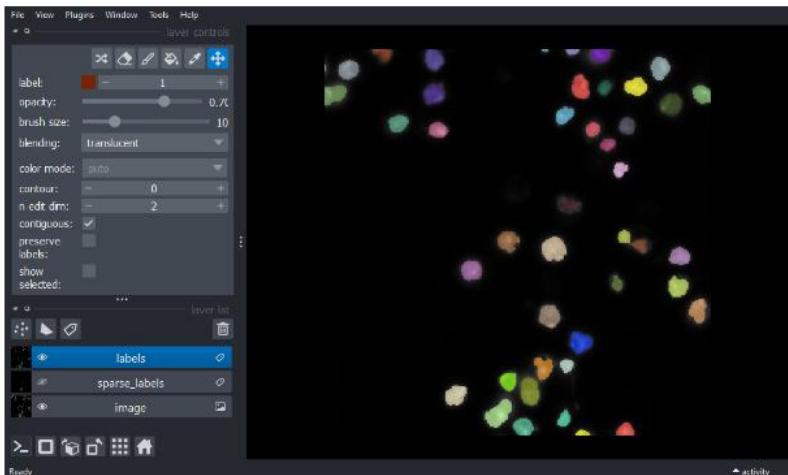
- Apply background-removal to an image in tiles. Determine the overlap width that's necessary to have artifact-free results.



Segmentation quality

- Measure segmentation quality of a given algorithm applied to a folder of images.

```
[7]: def my_segmentation_algorithm(input_image):  
  
    # background subtraction  
    background_subtracted = nsbatwm.white_tophat(input_image, radius = 10)  
  
    # instance segmentation / labeling  
    labels_result = nsbatwm.voronoi_otsu_labeling(background_subtracted,  
  
    return labels_result
```



Quality estimation: Sparse Jaccard Index

From the two label images loaded and produced above we can compute the sparse Jaccard Index.

```
[9]: metrics.jaccard_index_sparse(sparse_labels, labels)  
[9]: 0.8357392602053431
```

Exercise

Use the following for-loop and code snippets from above to compute the segmentation quality of all images in the folder. Provide the average quality over all images.

```
[10]: for image_filename in os.listdir(image_folder):  
    print(image_folder + image_filename)  
  
data/BBBC007_batch/17P1_POS0013_D_1UL.tif  
data/BBBC007_batch/20P1_POS0005_D_1UL.tif  
data/BBBC007_batch/20P1_POS0007_D_1UL.tif  
data/BBBC007_batch/20P1_POS0010_D_1UL.tif  
data/BBBC007_batch/A9_p7d.tif  
data/BBBC007_batch/AS_09125_040701150004_A02f00d0.tif
```

Optional exercise

- Play Bio-image Analysis Bubbles and test your skills: Can you build a workflow that reproduces the results?



Major parts of this game
were written by the AI in
cursor.
<https://www.cursor.com/>