ScaDS.AI
DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS AND ARTIFICIAL INTELLIGENCE

Funded by

Bundesministerium für Bildung und Forschung

SACHSEN

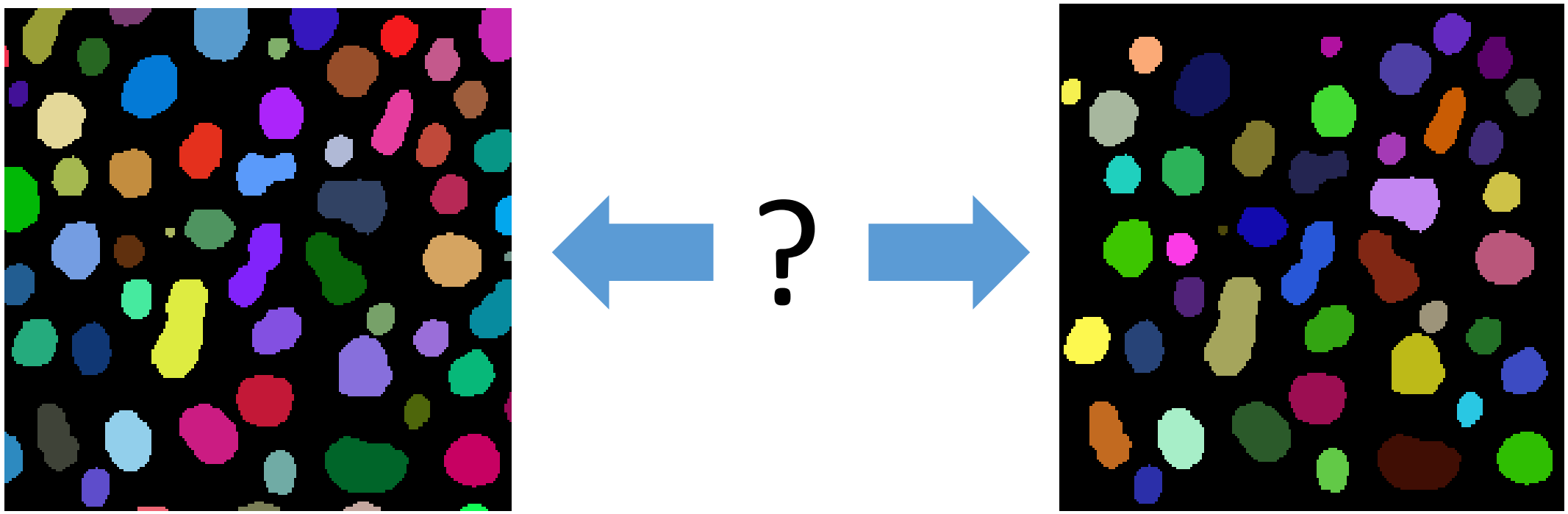
Diese Maßnahme wird gefördert durch die Bundesregierung aufgrund eines Beschlusses des Deutschen Bundestages. Diese Maßnahme wird mitfinanziert durch Steuermittel auf der Grundlage des von den Abgeordneten des Sächsischen Landtags beschlossenen Haushaltes.

Tiled & GPU-accelerated Image Processing

Robert Haase

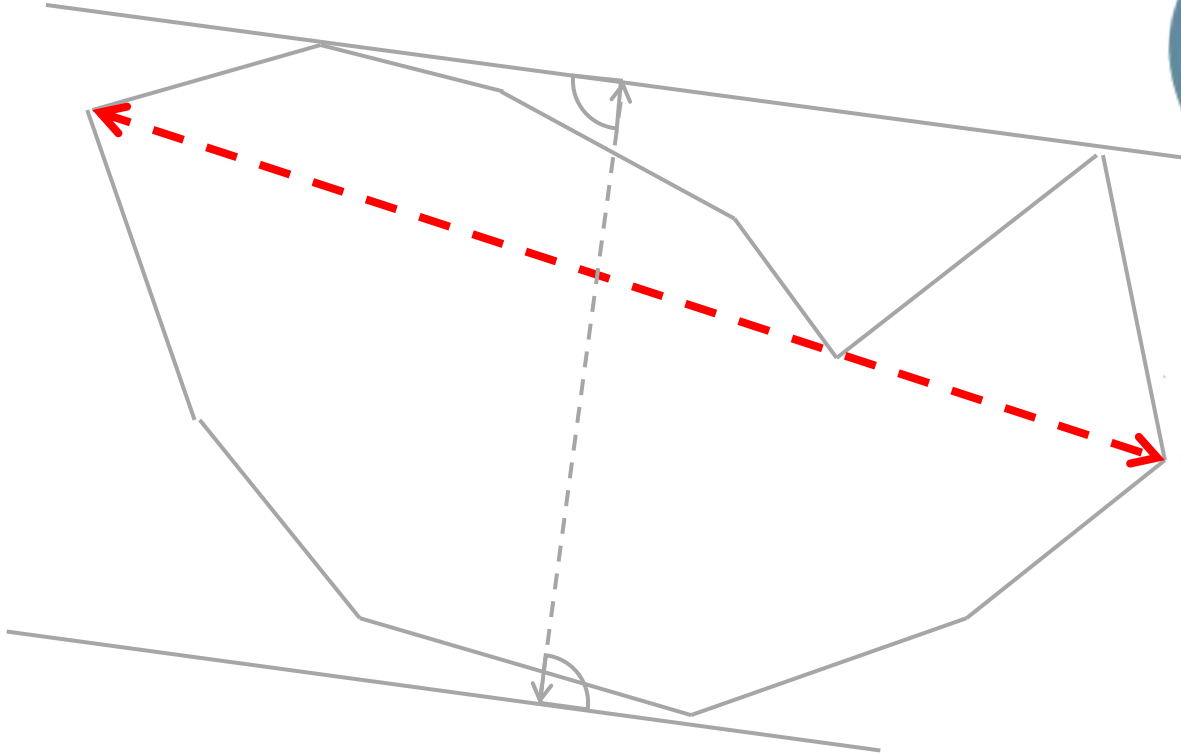
Quiz:

- 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

ScaDS.AI
DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS AND ARTIFICIAL INTELLIGENCE

GPU-accelerated Image Processing

Robert Haase

Funded by

Bundesministerium für Bildung und Forschung

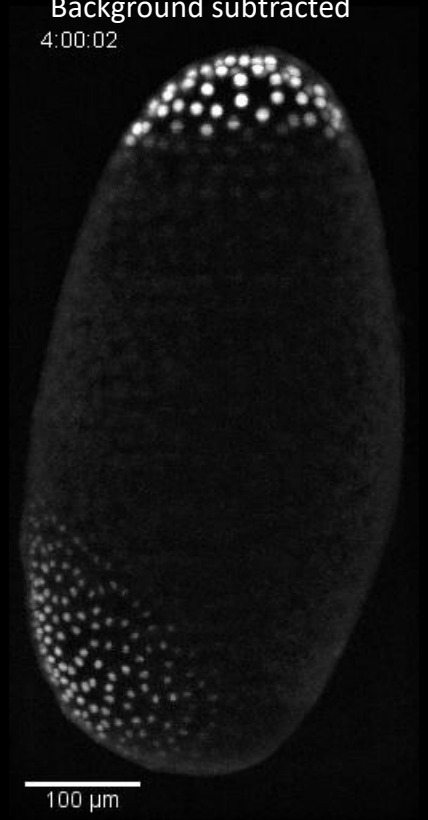
SACHSEN

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.

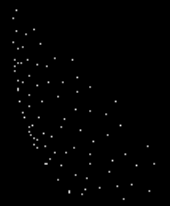
GPU accelerated image processing in life sciences

- … to study embryo development

Tribolium castaneum nuclei-GFP,
Background subtracted

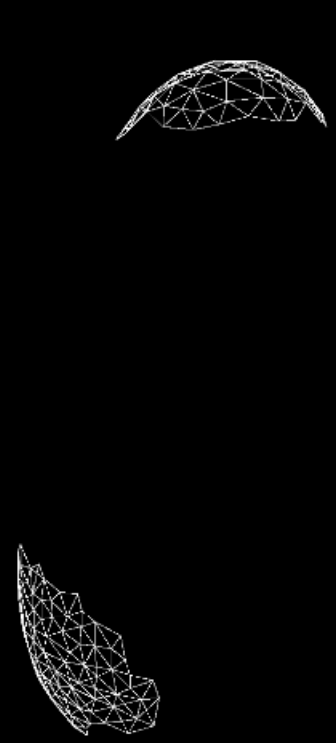


Spot detection (3D)

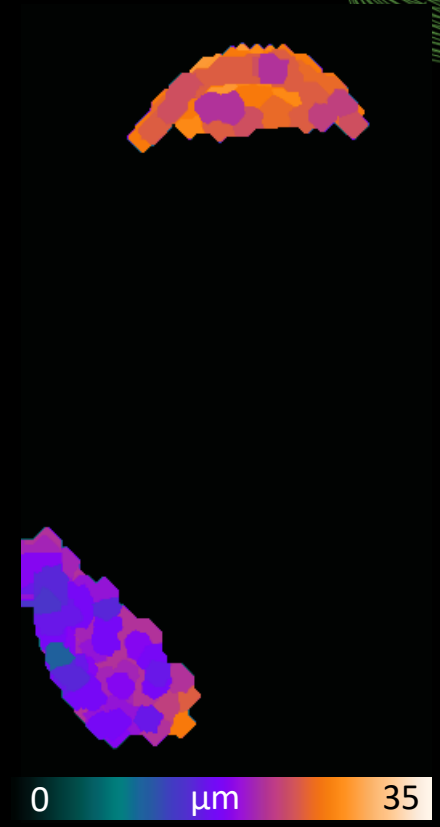


Theoretical membranes
(pseudo Voronoi map)

Neighbor mesh



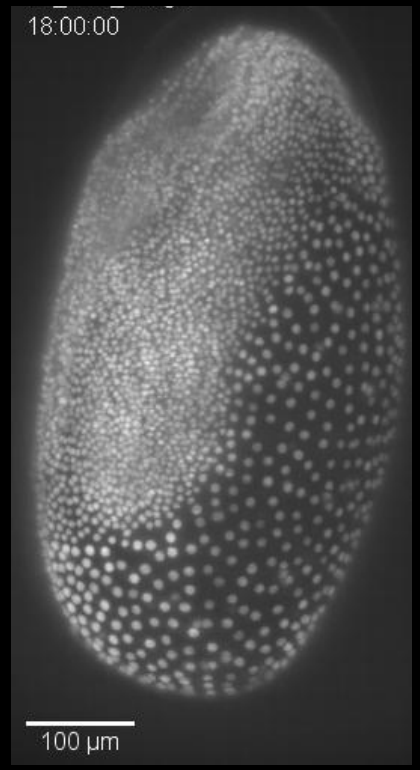
Average centroid distance of neighbors



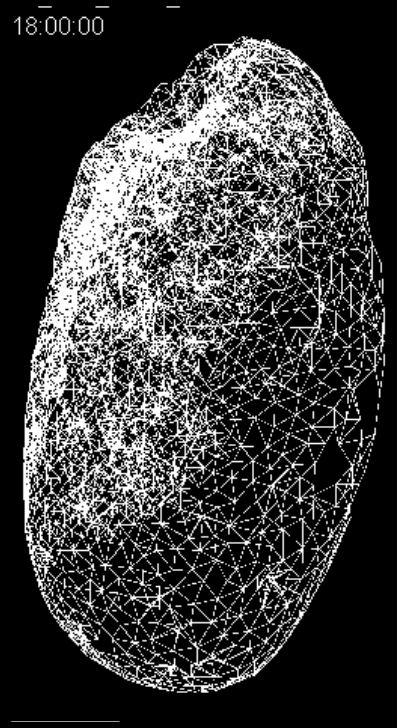
GPU accelerated image processing in life sciences

- Raytracing enables differentiating surface and sub-surface mesh nodes

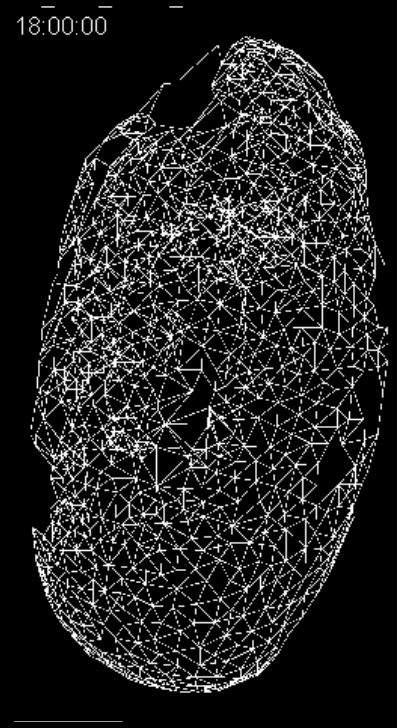
3D stack input



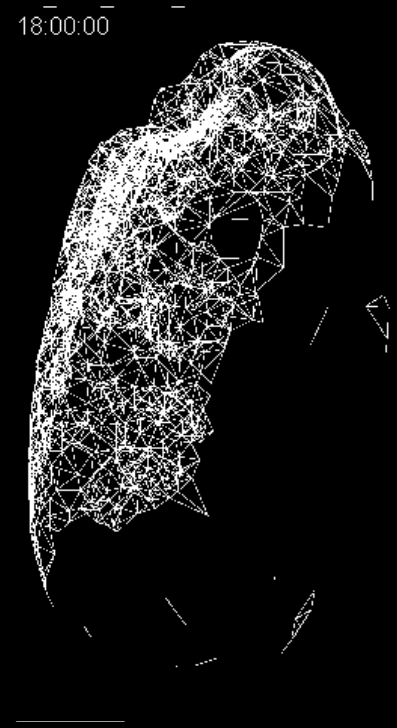
Neighbor mesh



Surface neighbor mesh



Sub-surface neighbor mesh



Merge

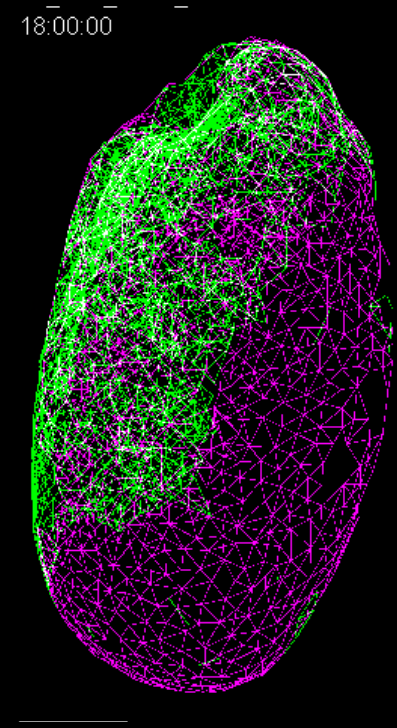


Image processing in life-sciences

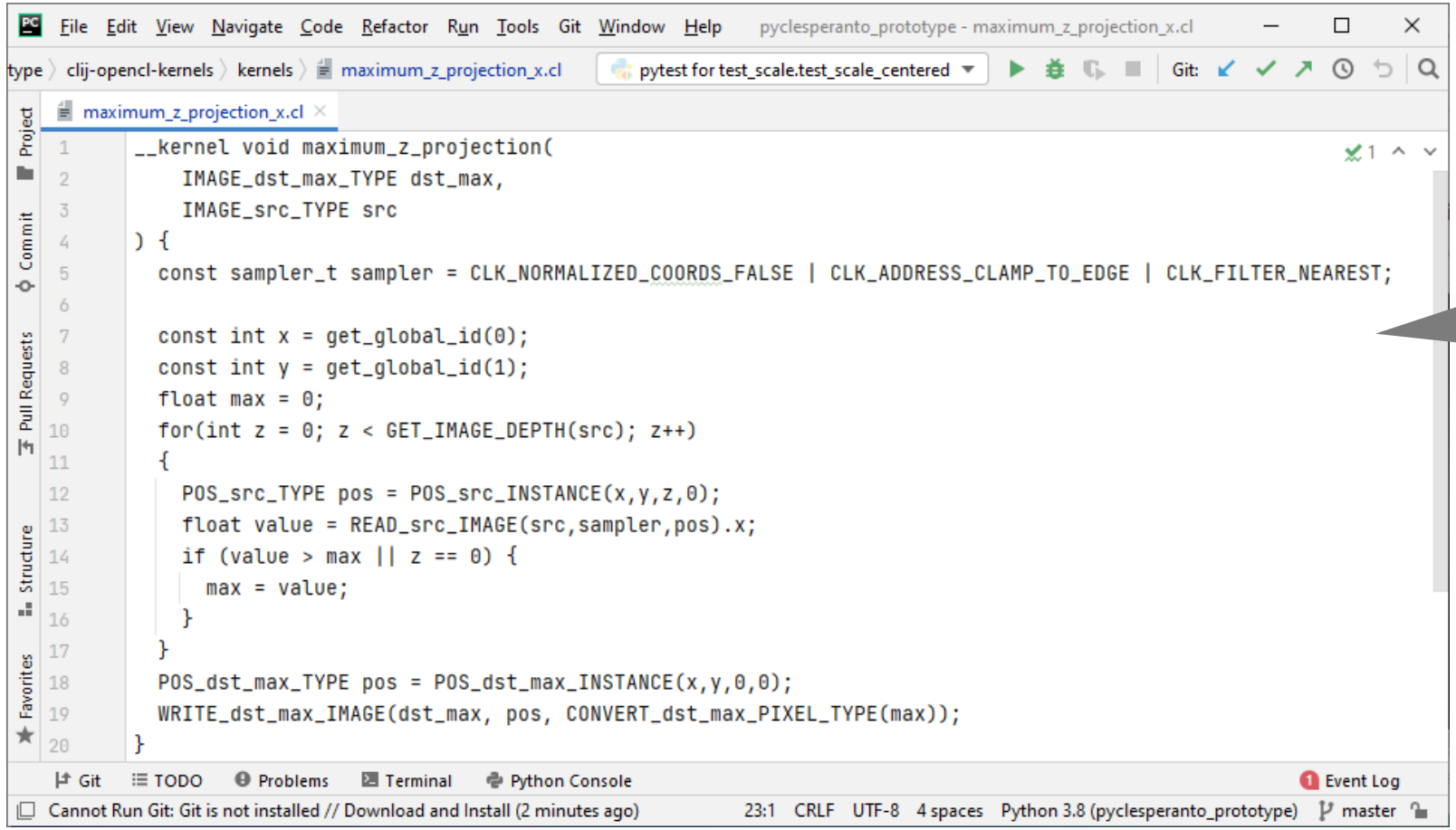
- State-of-the-art software for more than 20 years: ImageJ / Fiji

2x

Image data source: Daniela Vorkel, Myers lab, MPI-CBG/CSBD

OpenCL-based GPU-acceleration

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



Maximum intensity projection along Z

OpenCL: https://www.khronos.org/opencl/

Kernel collection: https://github.com/clEsperanto/clij-opencl-kernels

User-friendly GPU-acceleration

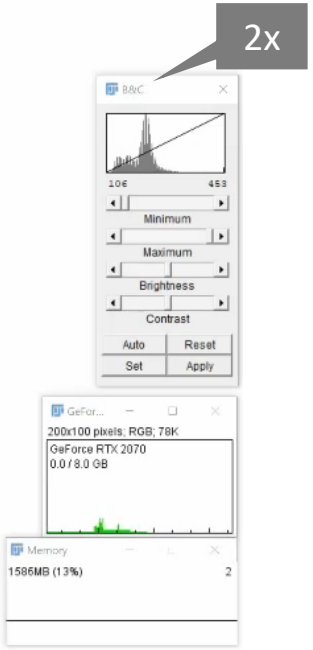
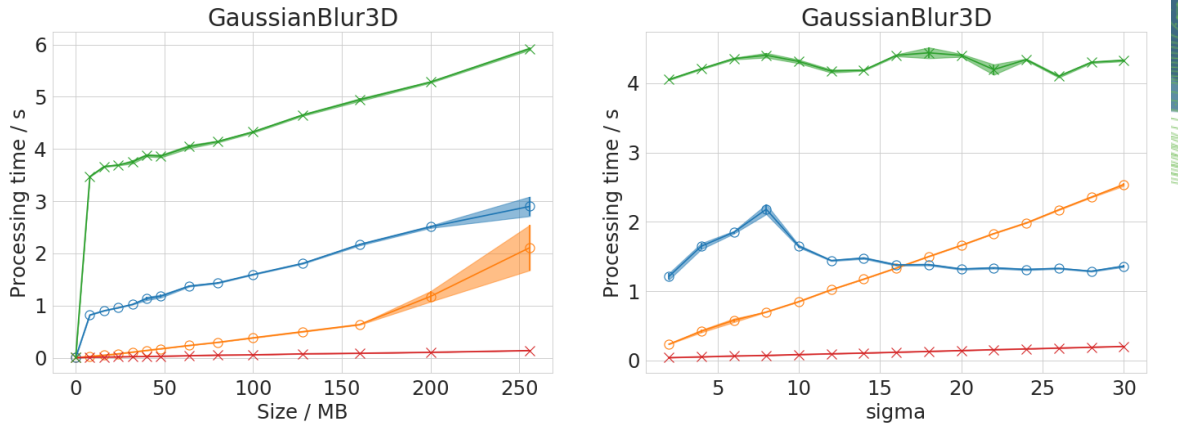


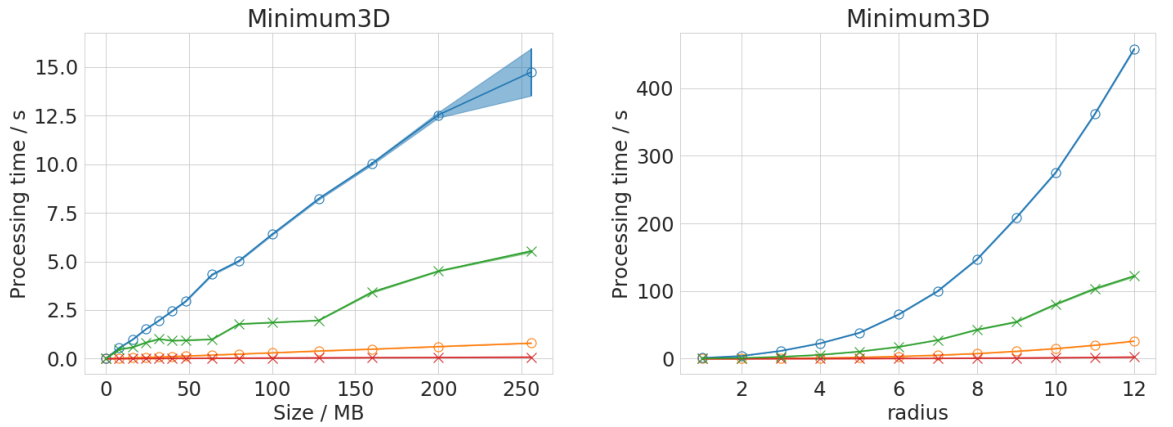
Image data source: Daniela Vorkel, Myers lab, MPI-CBG/CSBD

GPU-accelerated image processing

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

vs.





Workstation CPU

2x Intel Xeon Silver 4110

Workstation GPU

Nvidia Quadro P6000

Laptop CPU

Intel Core i7-8650U

Laptop GPU

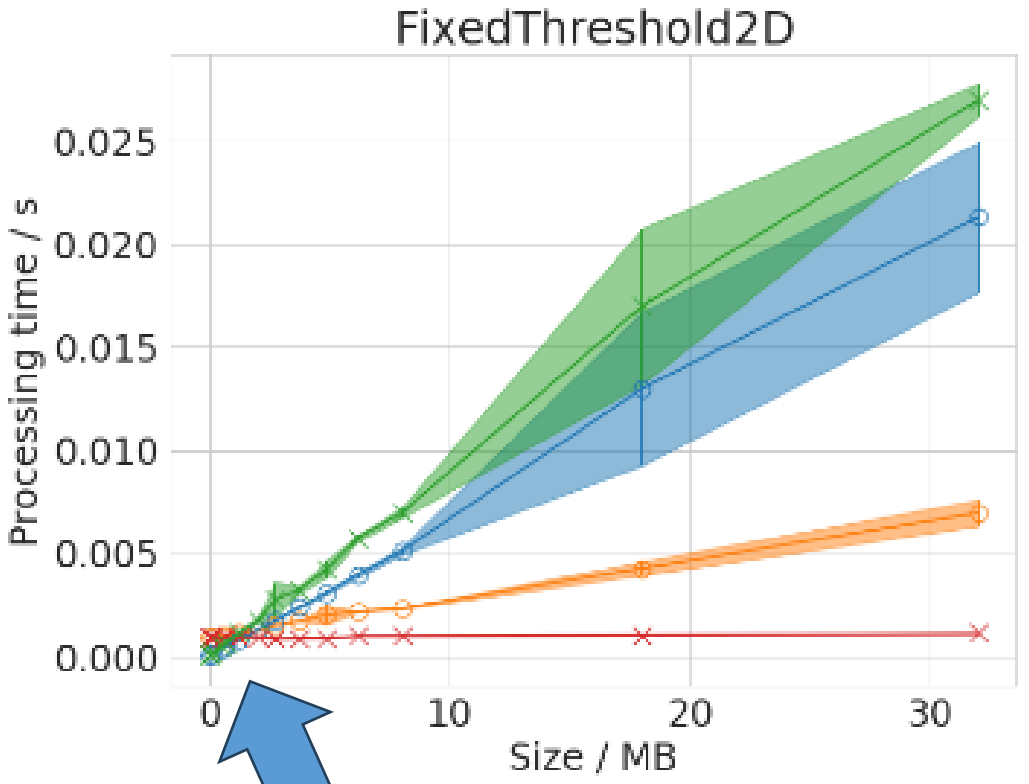
Intel UHD 620 GPU

Haase et al. Nat Methods (2020),
https://www.nature.com/articles/s41592-019-0650-1

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

Workstation GPU

Nvidia Quadro P6000

Laptop CPU

Intel Core i7-8650U

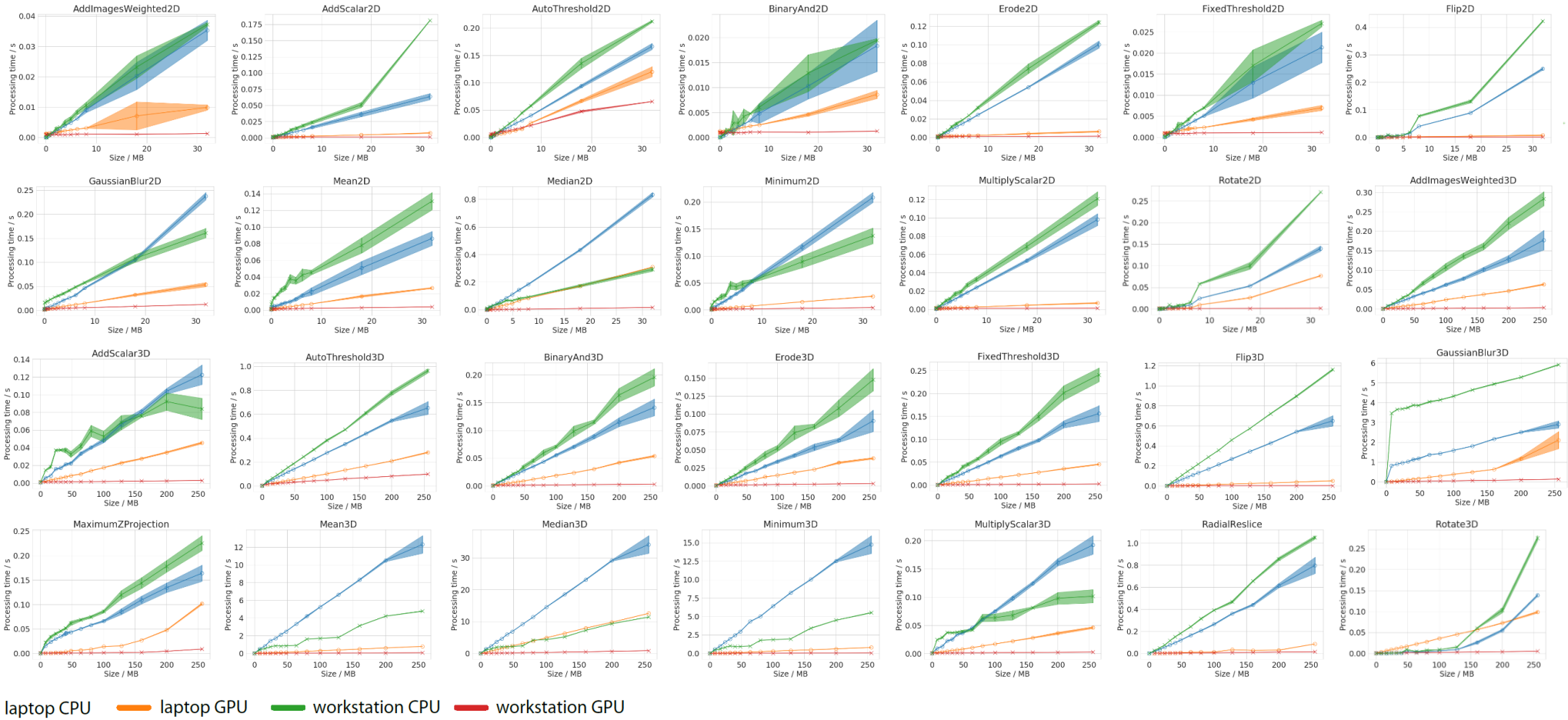
Laptop GPU

Intel UHD 620 GPU

* https://www.techpowerup.com/cpu-specs/core-i7-8650u.c1975

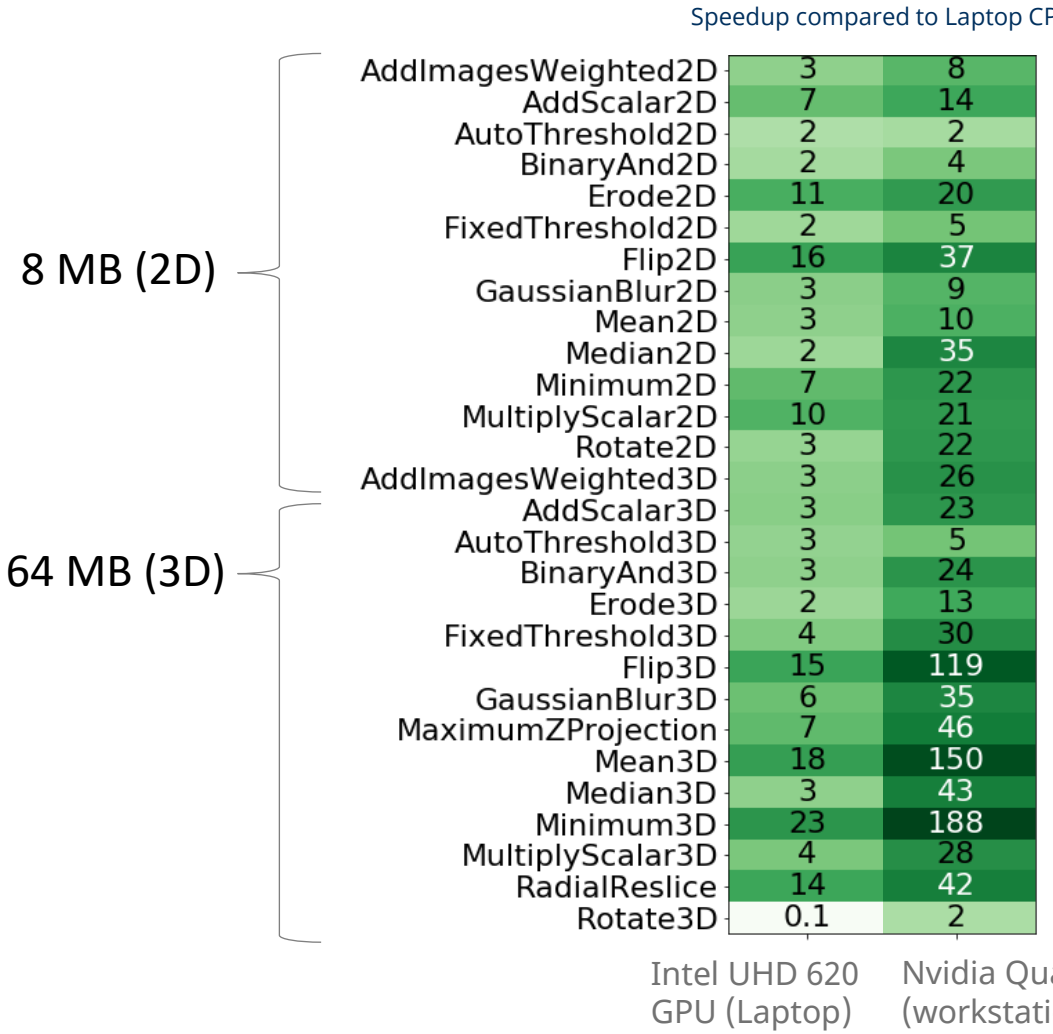
GPU-accelerated image processing

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



Haase et al. Nat Methods (2020),
https://www.nature.com/articles/s41592-019-0650-1

GPU-accelerated image processing

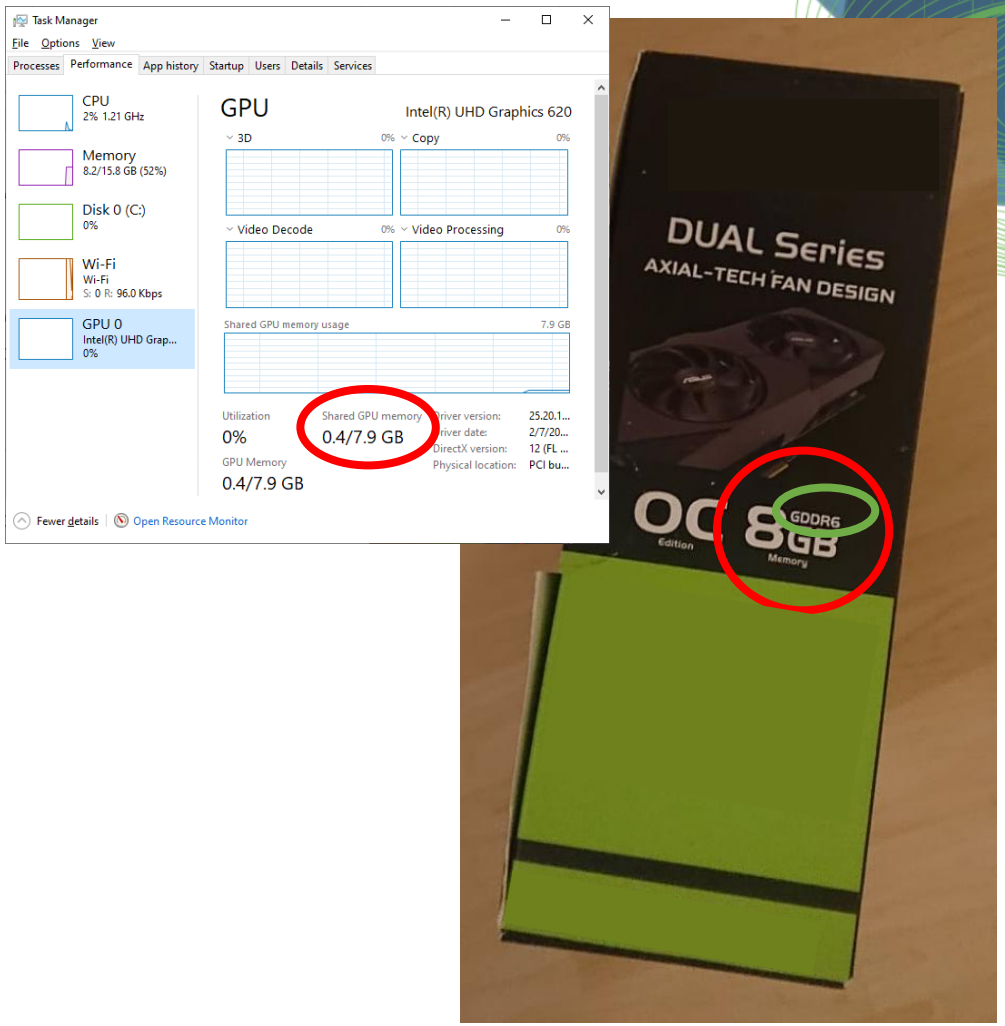


Memory	
Memory Size:	24 GB
Memory Type:	GDDR5X
Memory Bus:	384 bit
Bandwidth:	432.8 GB/s

Haase et al. Nat Methods (2020),
https://www.nature.com/articles/s41592-019-0650-1
https://www.techpowerup.com/gpu-specs/quadro-p6000.c2865

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: (processing time / loading time) > 10,
- Memory requirements: rule of thumb: image size in GB x 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 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:

Graphics Processor	
GPU Name:	GB202
GPU Variant:	GB202-300-A1
Architecture:	Blackwell 2.0
Foundry:	TSMC
Process Type:	4N FinFET
Process Size:	5 nm
Transistors:	92,200 million
Density:	122.9M / mm²
Die Size:	750 mm²

Graphics Card	
Release Date:	Jan 30th, 2025
Announced:	Jan 6th, 2025
Generation:	GeForce 50
Predecessor:	GeForce 40
Production:	Active
Launch Price:	1,999 USD
Bus Interface:	PCIe 5.0 x16
Reviews:	41 in our database

Relative Performance	
GeForce RTX 4070 Ti S...	51%
Radeon RX 7900 XT	52%
GeForce RTX 3090 Ti	53%
Radeon RX 9070 XT	55%
GeForce RTX 4080	59%
GeForce RTX 5070 Ti	60%
GeForce RTX 4080 SUP...	60%
Radeon RX 7900 XTX	60%
GeForce RTX 5080	66%
GeForce RTX 4090	74%
GeForce RTX 5090	100%

Based on TPU review data: "Performance Summary" at 1920x1080, 4K for 2080 Ti and faster.

Theoretical Performance	
Pixel Rate:	423.6 GPixel/s
Texture Rate:	1,637 GTexel/s
FP16 (half):	104.8 TFLOPS (1:1)
FP32 (float):	104.8 TFLOPS
FP64 (double):	1.637 TFLOPS (1:64)

Clock Speeds	
Base Clock:	2017 MHz
Boost Clock:	2407 MHz
Memory Clock:	1750 MHz 28 Gbps effective

Board Design	
Slot Width:	Dual-slot
Length:	304 mm

Memory	
Memory Size:	32 GB
Memory Type:	GDDR7
Memory Bus:	512 bit
Bandwidth:	1.79 TB/s

Render Config	
Shading Units:	21760
TMUs:	680
ROPs:	176
SM Count:	170
Tensor Cores:	680
RT Cores:	170

Graphics Features

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 / mm²
Die Size:	294 mm²

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 / mm²
Die Size:	188 mm²

Nvidia RTX 4070 versus 4070 mobile

- Vendors name GPUs similarly, despite their configuration is quite diffe

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

RTX 4070 mobile

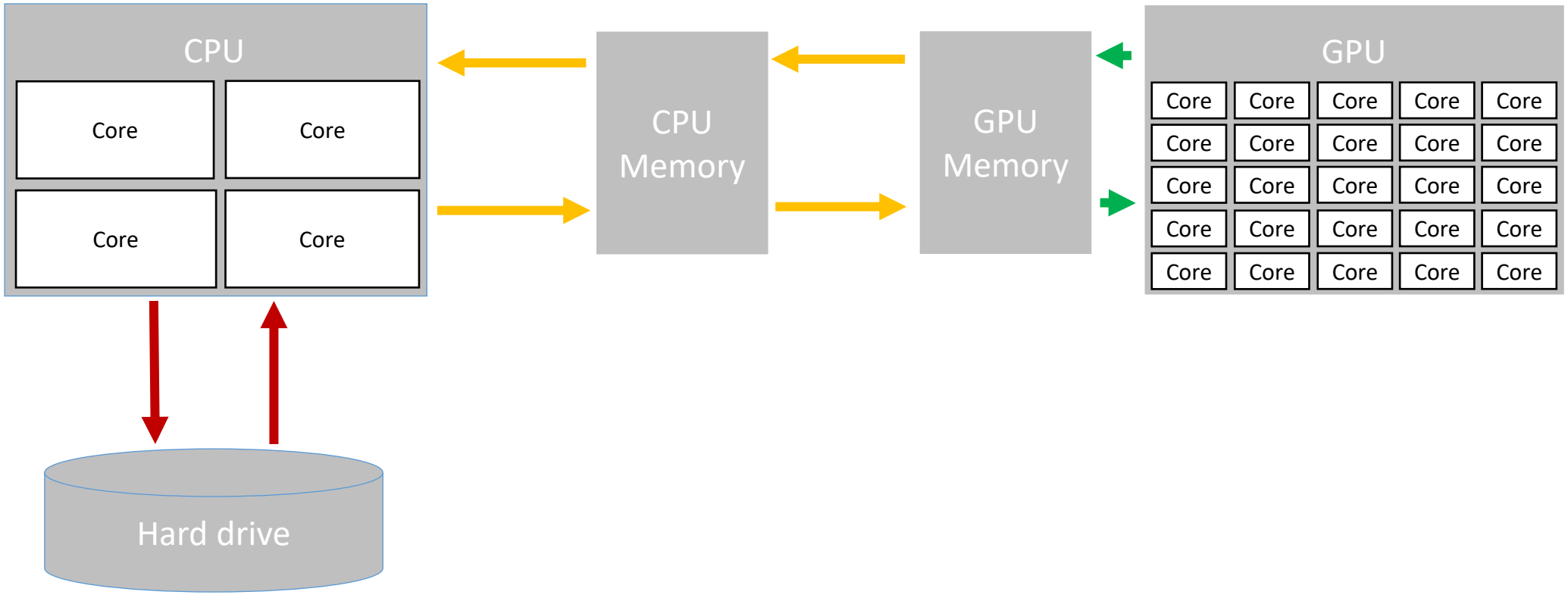
Clock Speeds	
Base Clock:	1395 MHz
Boost Clock:	1695 MHz
Memory Clock:	2000 MHz 16 Gbps effective

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

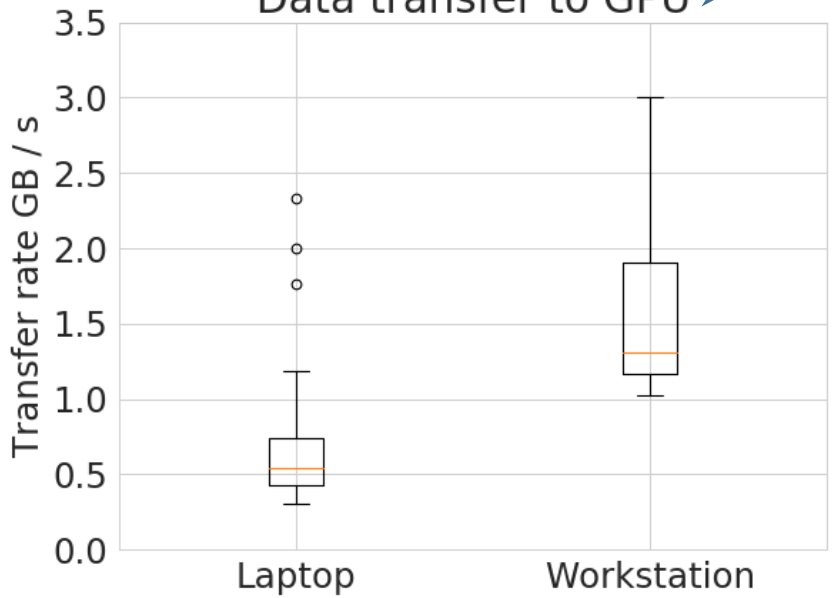
- GPUs are specialised in processing, very fast thanks to many cores and fast memory access



Data transfer takes time

- Data transfer is the bottle neck

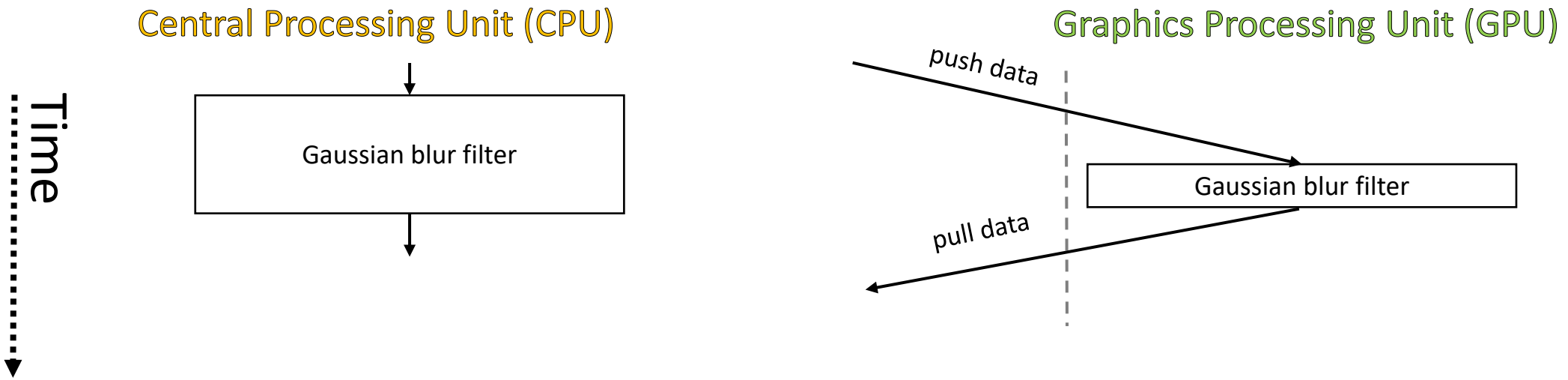
Push data to GPU



Pull data from GPU

Build workflows consisting of many operations

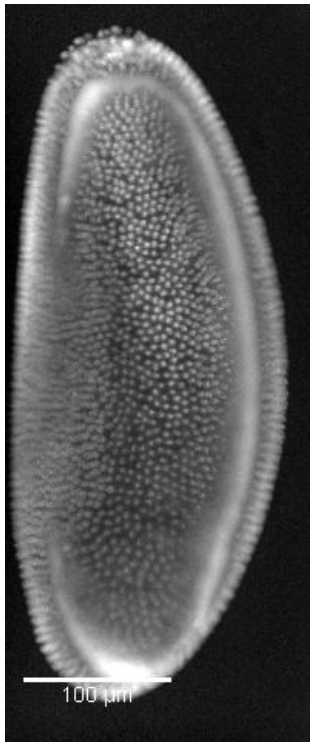
GPU acceleration may suffer from data transfer between CPU and GPU

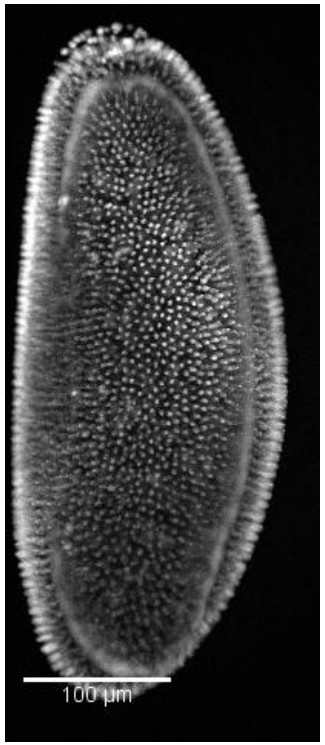


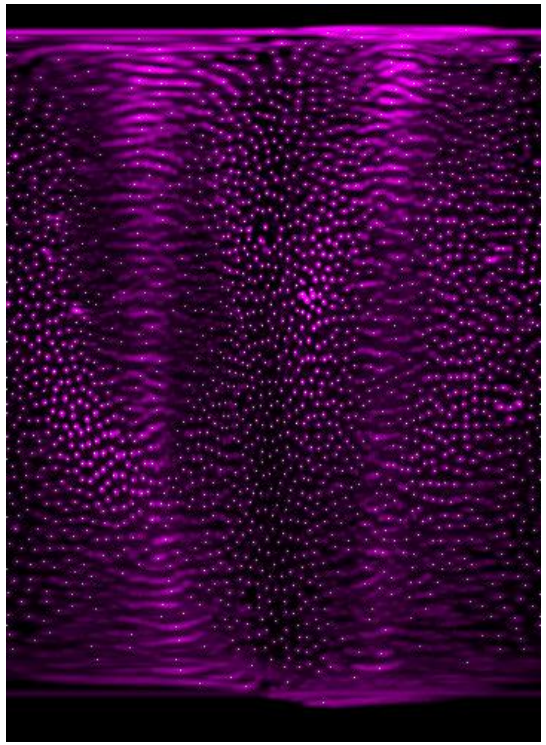
Optimal performance through smart memory management

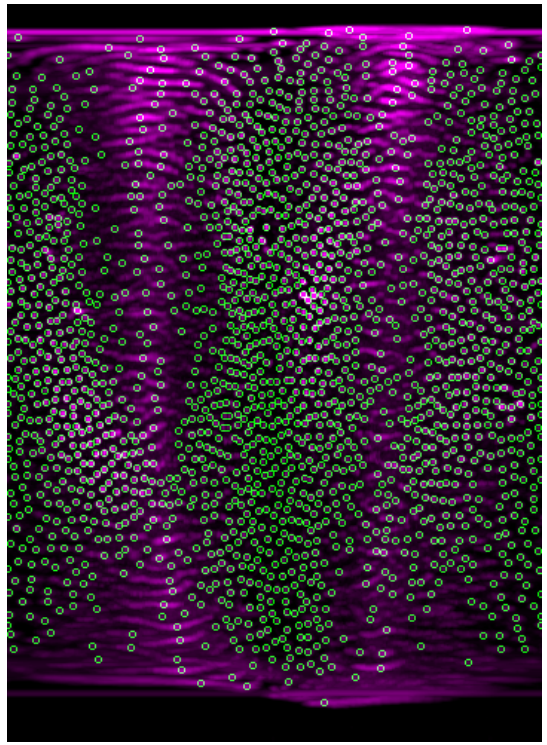
- Example workflow processing a Drosophila melanogaster embryo, histone-GFP

Load data → Preprocessing → Transformation → Segmentation → Save data





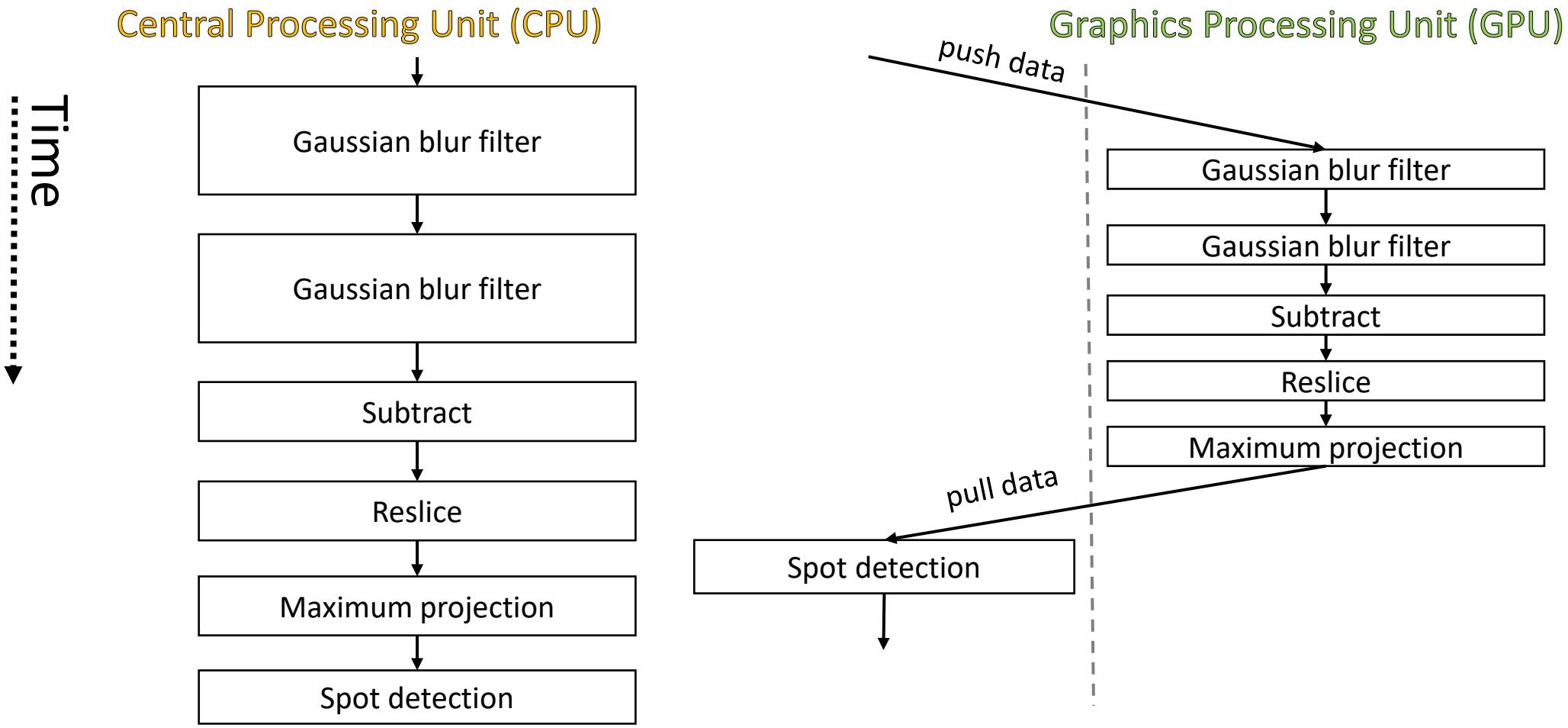




Haase et al. Nat Methods (2020),
https://www.nature.com/articles/s41592-019-0650-1
https://clij.github.io/

Build workflows consisting of many operations

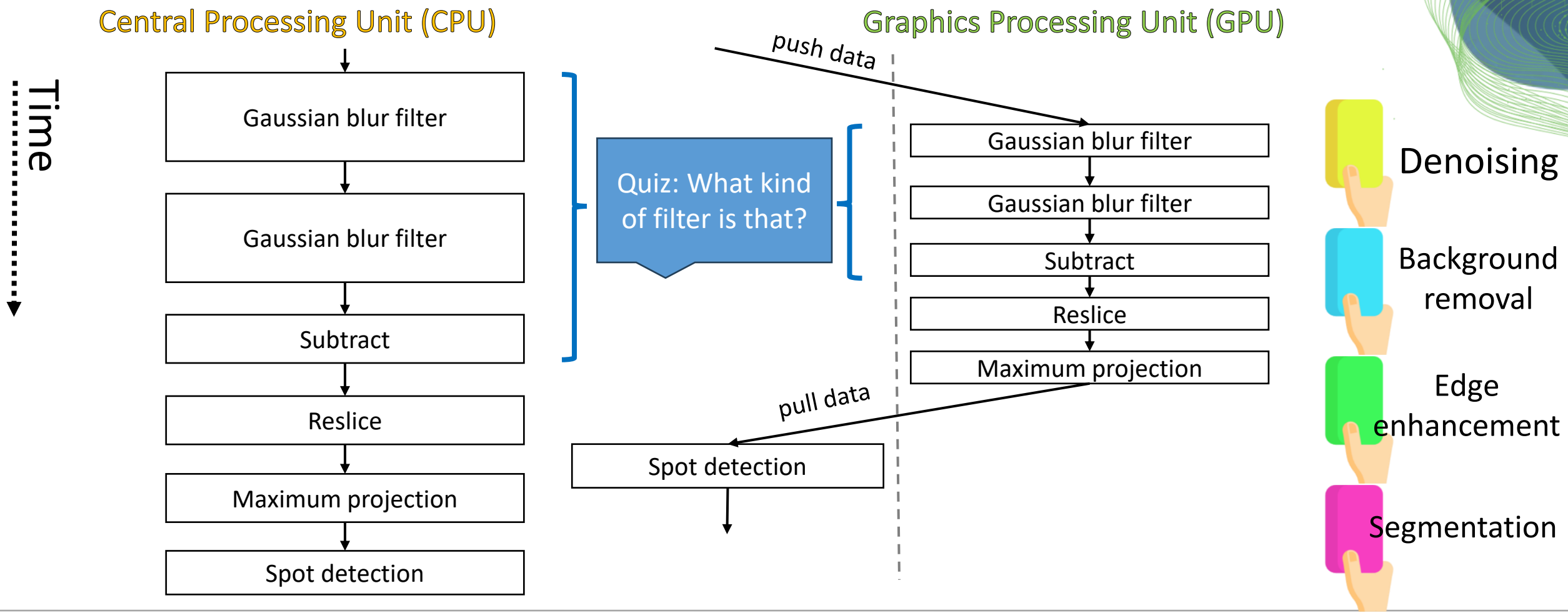
GPU acceleration may suffer from data transfer between CPU and GPU



Haase et al. Nat Methods (2020),
https://www.nature.com/articles/s41592-019-0650-1
https://clij.github.io/

Build workflows consisting of many operations

GPU acceleration may suffer from data transfer between CPU and GPU



Haase et al. Nat Methods (2020),
https://www.nature.com/articles/s41592-019-0650-1
https://clij.github.io/

ScaDS.AI
DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS AND ARTIFICIAL INTELLIGENCE

Funded by

Bundesministerium für Bildung und Forschung

SACHSEN

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 Python: OpenCL / clesperanto

Robert Haase

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

https://github.com/clEsperanto/pyclesperanto
Picture cropped from
https://commons.wikimedia.org/wiki/File:NVIDIA_GeForce_GTX_780_PCB-Front.jpg

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 ¼ of the global memory size?

https://github.com/clEsperanto/pyclesperanto

Picture cropped from
https://commons.wikimedia.org/wiki/File:NVIDIA_GeForce_GTX_780_PCB-Front.jpg
licensed CC-BY 2.0 by GBPublic_PR

GPU-accelerated image processing

- Memory constraints

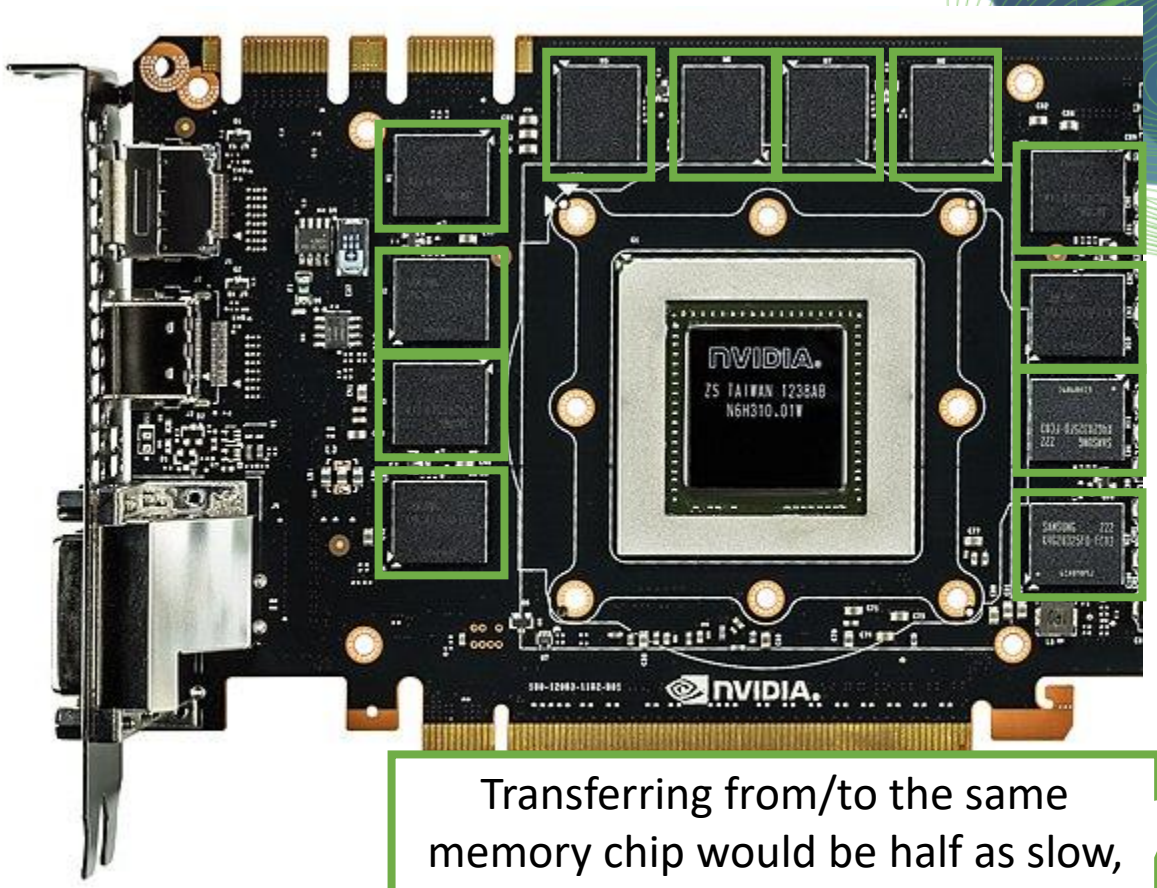
```
[2]: import pyclesperanto as cle

     # 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



Transferring from/to the same memory chip would be half as slow, compared to streaming data from one chip through another.

https://github.com/clEsperanto/pyclesperanto

Picture cropped from
https://commons.wikimedia.org/wiki/File:NVIDIA_GeForce_GTX_780_PCB-Front.jpg
licensed CC-BY 2.0 by GBPublic_PR

GPU-accelerated image processing

- Memory constraints

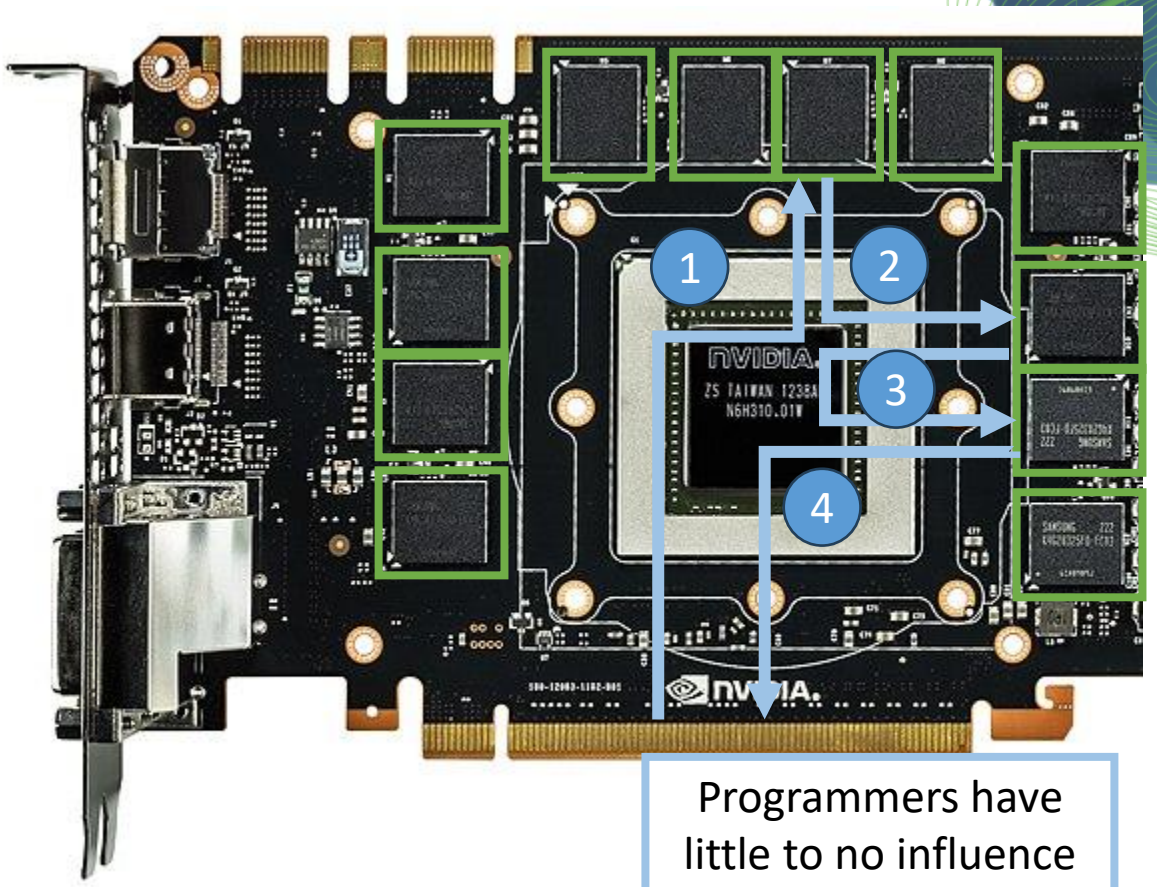
```
[2]: import pyclesperanto as cle

     # 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



Programmers have little to no influence on from/to where data streams.

https://github.com/clEsperanto/pyclesperanto

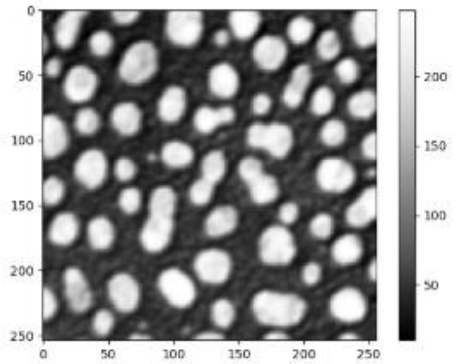
Picture cropped from
https://commons.wikimedia.org/wiki/File:NVIDIA_GeForce_GTX_780_PCB-Front.jpg
licensed CC-BY 2.0 by GBPublic_PR

GPU-accelerated image processing

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

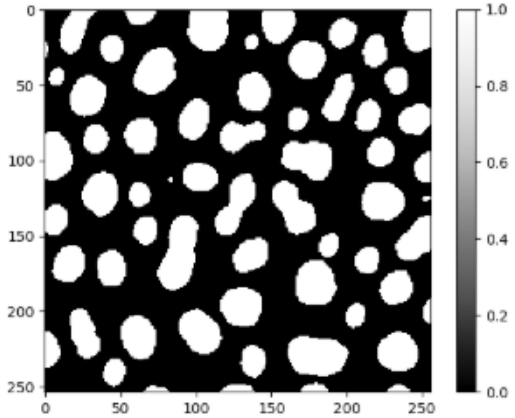
[6]:
```
# noise removal
blurred = cle.gaussian_blur(image, sigma_x=1, sigma_y=1)
blurred
```

[6]:



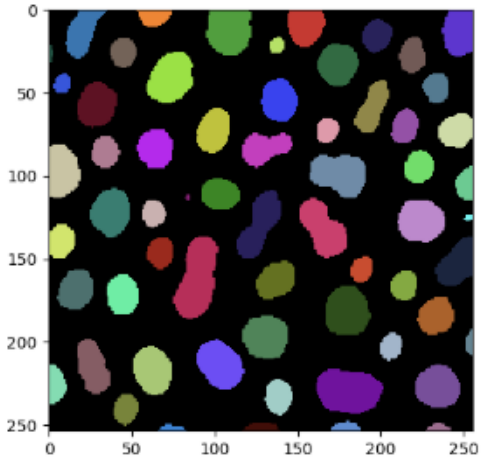
[7]:
```
# binarization
binary = cle.threshold_otsu(blurred)
binary
```

[7]:



[8]:
```
labels = cle.label(binary)
labels
```

[8]:

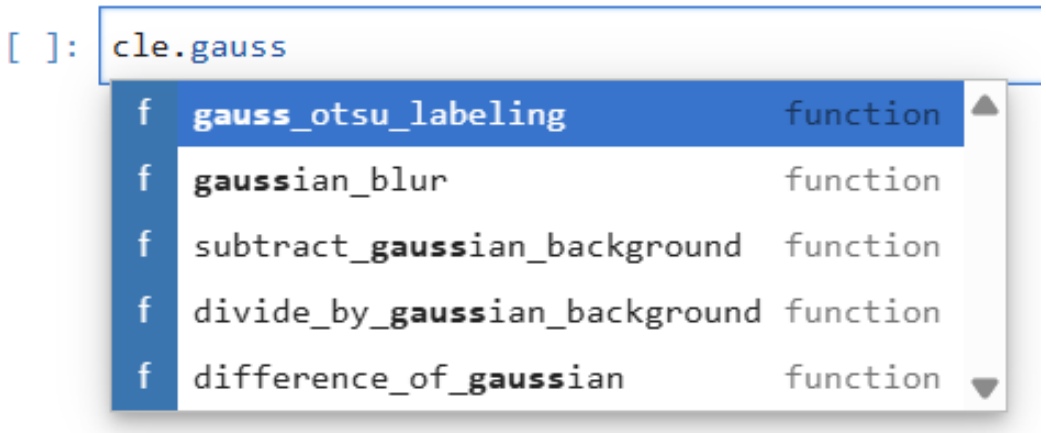


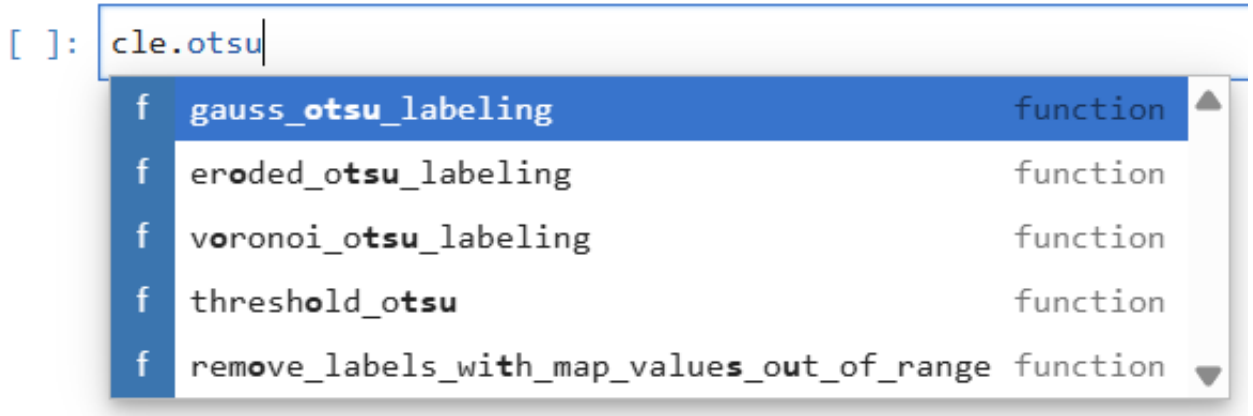
cle._ image

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.





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 `sckikit-image` filter in this case?

Benchmarking

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

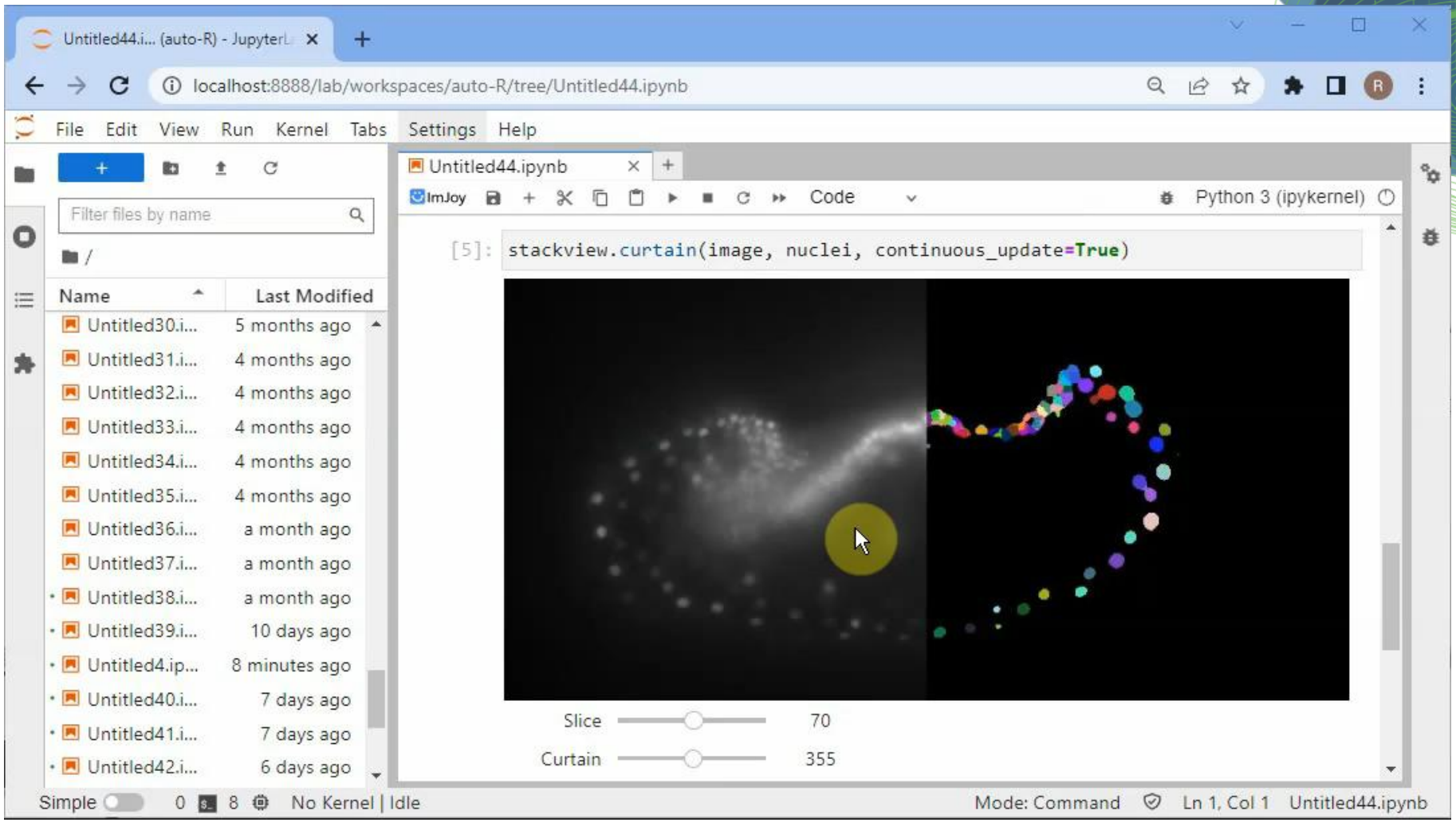
```
[7]: %%timeit
     skimage.filters.gaussian(image, sigma=5, out=blurred_image, preserve_range=True)

     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: Why do we pass the output-image here?

3D visualization on the cluster

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



ScaDS.AI
DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS AND ARTIFICIAL INTELLIGENCE

Funded by

Bundesministerium für Bildung und Forschung

SACHSEN

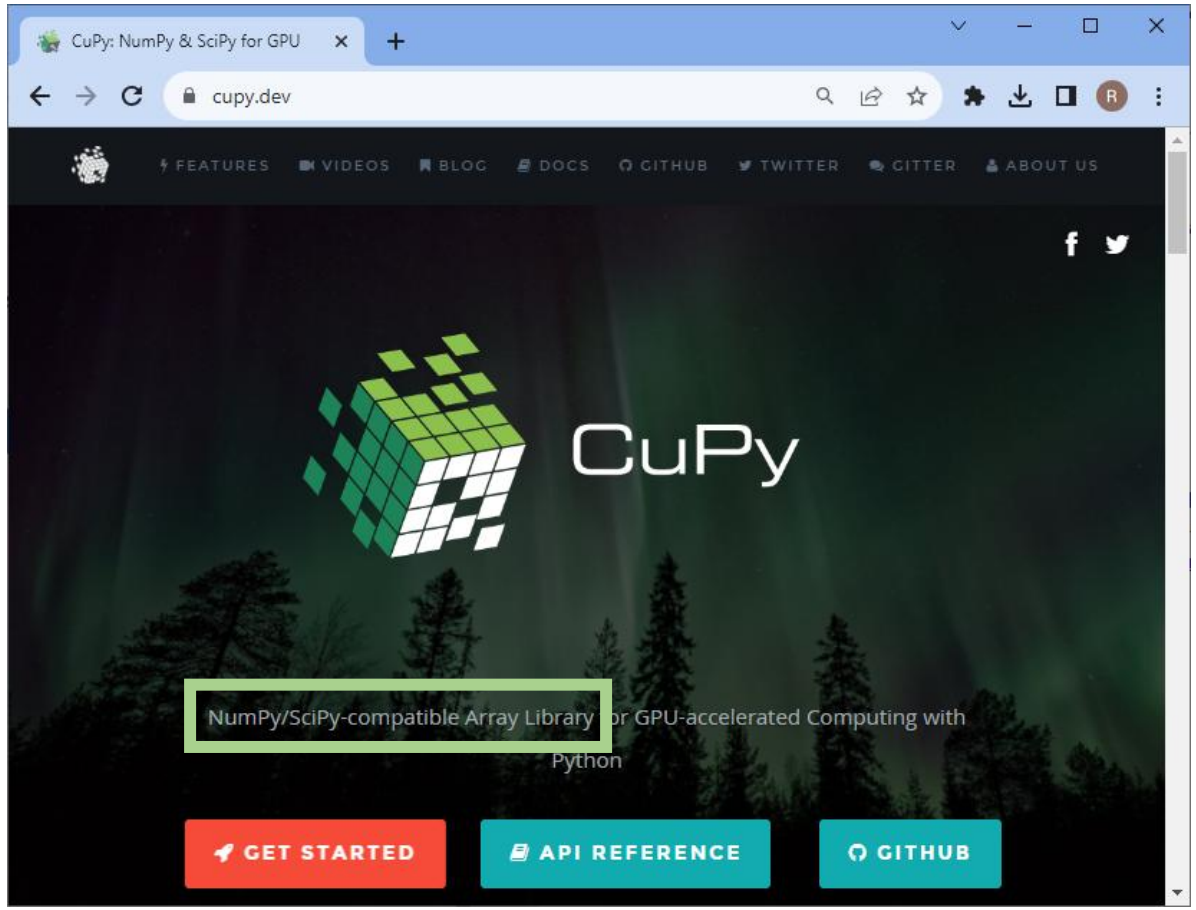
Diese Maßnahme wird gefördert durch die Bundesregierung aufgrund eines Beschlusses des Deutschen Bundestages. Diese Maßnahme wird mitfinanziert durch Steuermittel auf der Grundlage des von den Abgeordneten des Sächsischen Landtags beschlossenen Haushaltes.

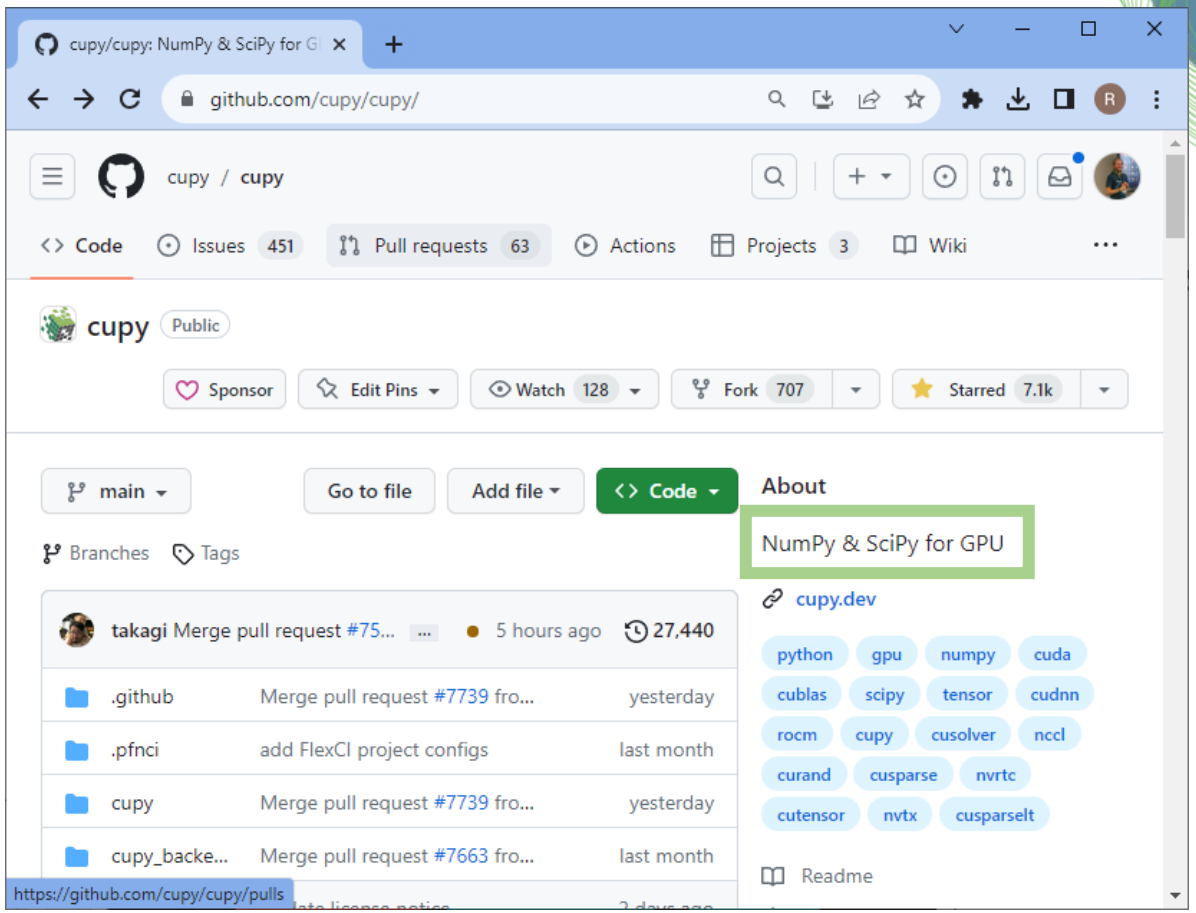
GPU-accelerated Image Processing in Python: CUDA / cupy

Robert Haase

cupy

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





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.

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

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

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

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

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

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

Common patterns

- To make code independent from cupy availablity, 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)
```

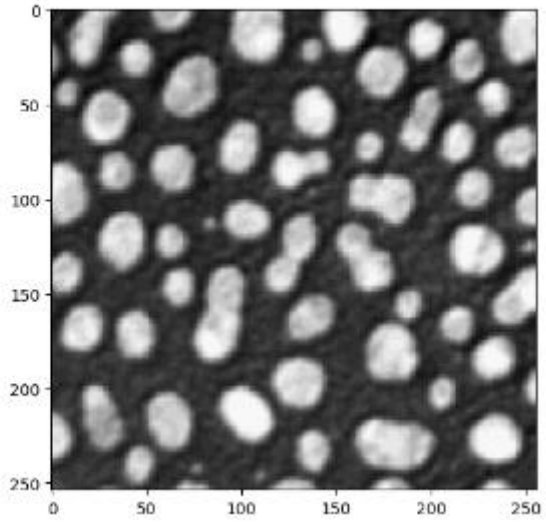
```
[3]: numpy.ndarray
```

Common patterns

- Some if-else blocks are hard to avoid

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

     imshow(np_image)
```

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

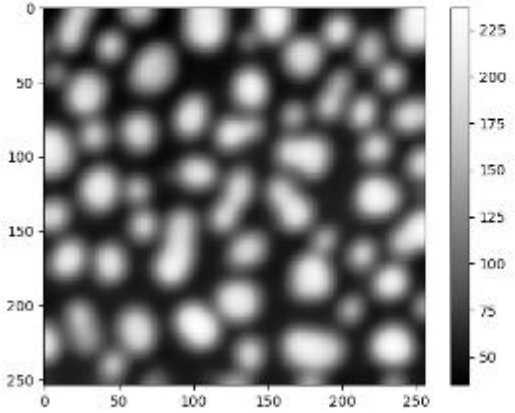


- Stackview aims to be cupy/numpy agnostic

```
[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

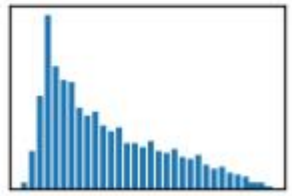
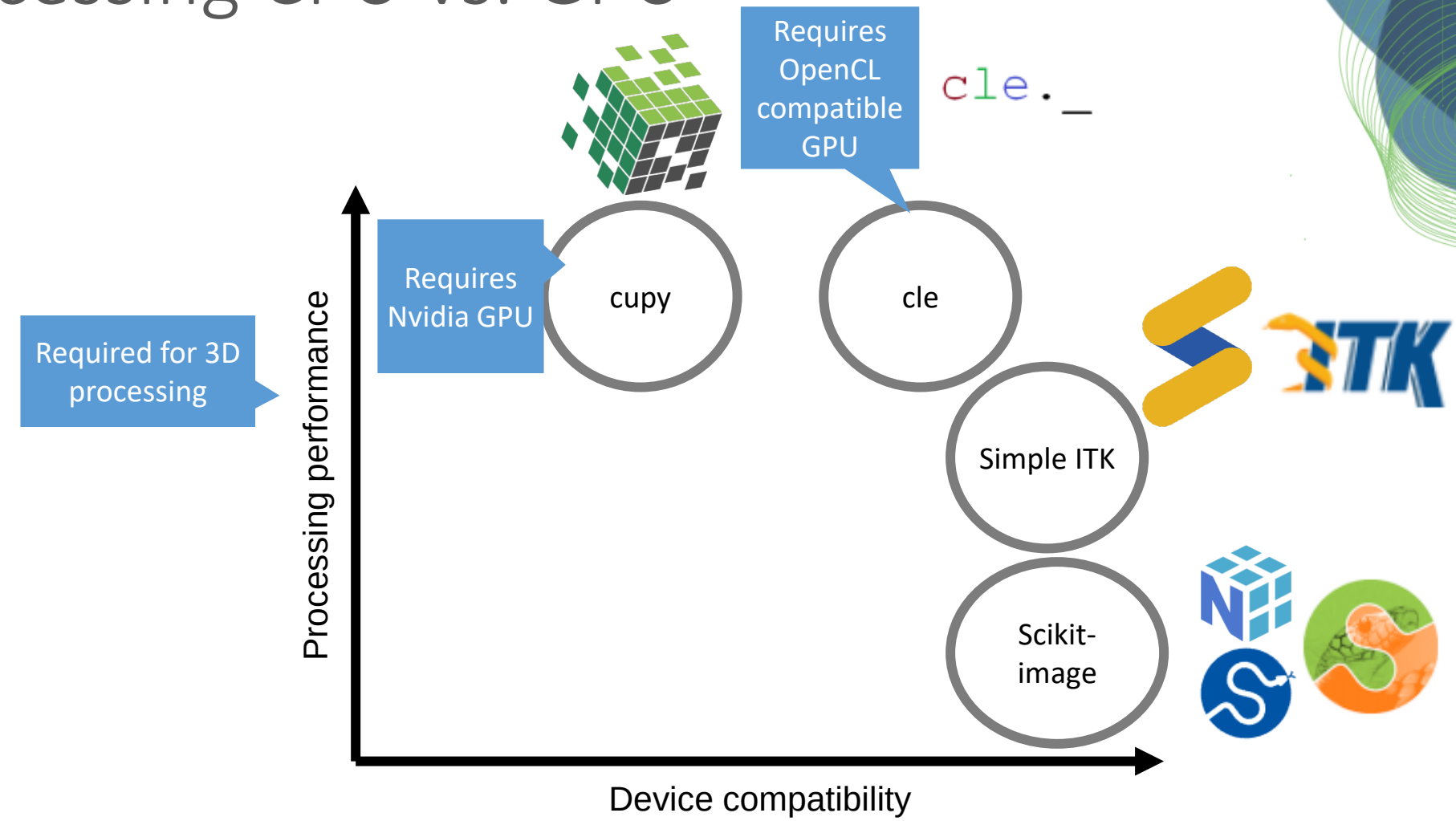


Image Processing CPU vs. GPU

- Performance versus compatibility



ScaDS.AI
DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS AND ARTIFICIAL INTELLIGENCE

Funded by

Bundesministerium für Bildung und Forschung

SACHSEN

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.

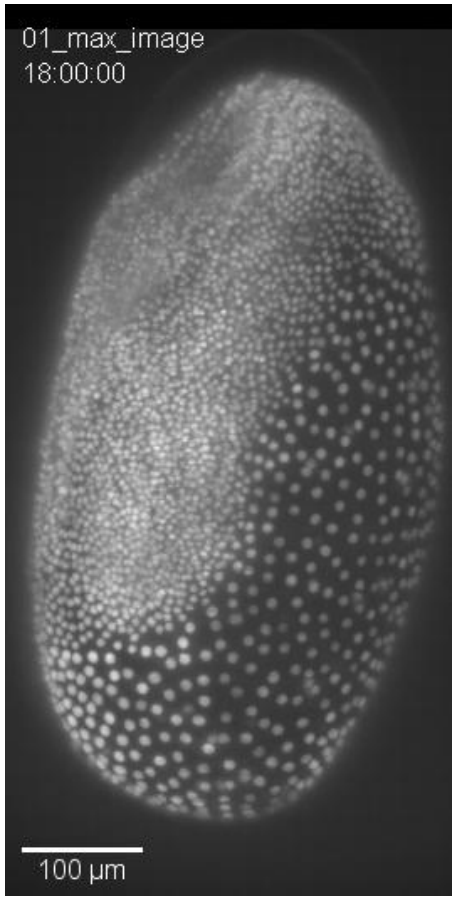
Tiled image processing

Robert Haase

Quiz: Memory constraints

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

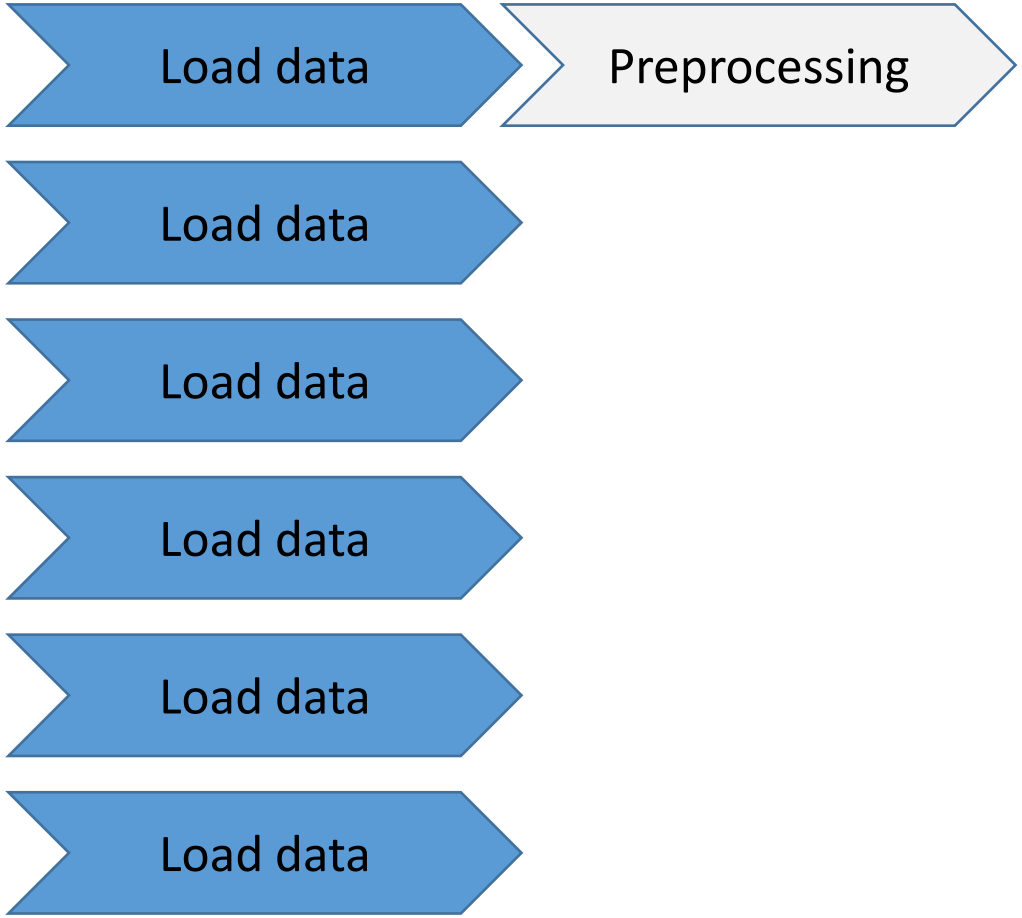
1.
2.
3.
4.
5. Tiled image processing



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

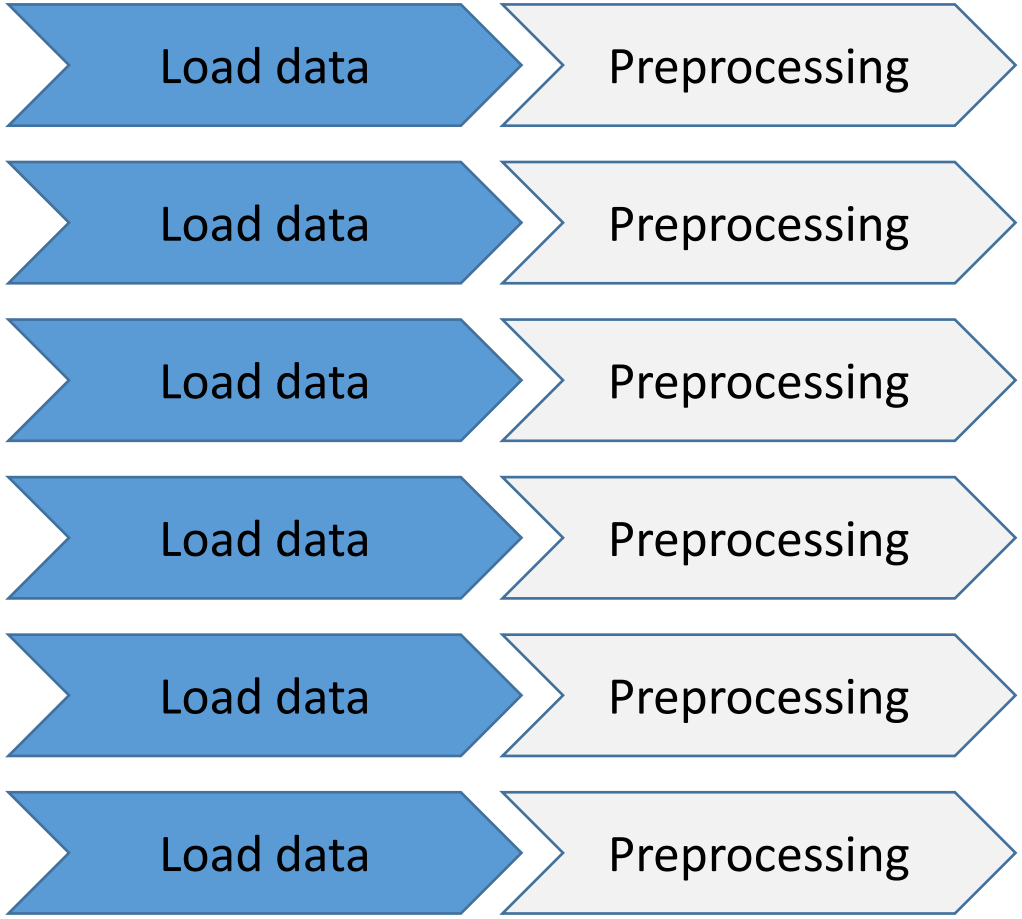
Optimal performance through smart memory management

- The classical way of dealing with large image stacks…



Optimal performance through smart memory management

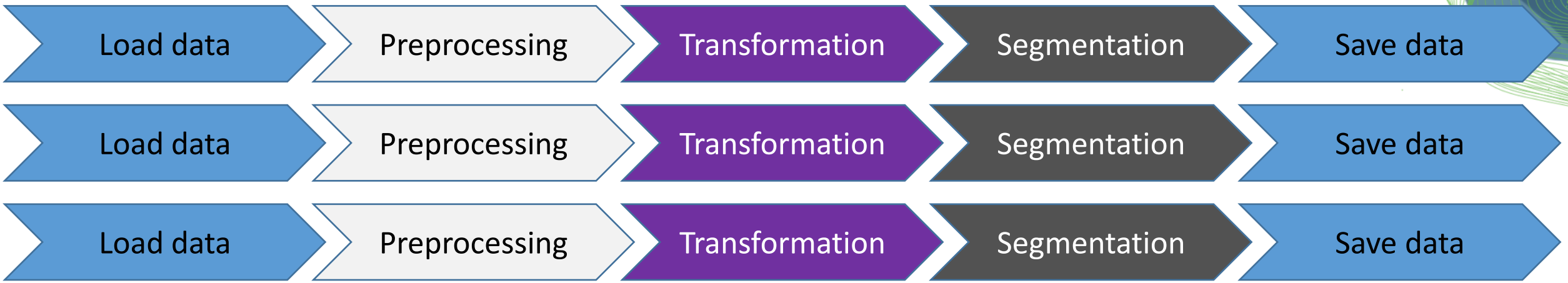
- The classical way of dealing with large image stacks… is suboptimal



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

Optimal performance through smart memory management

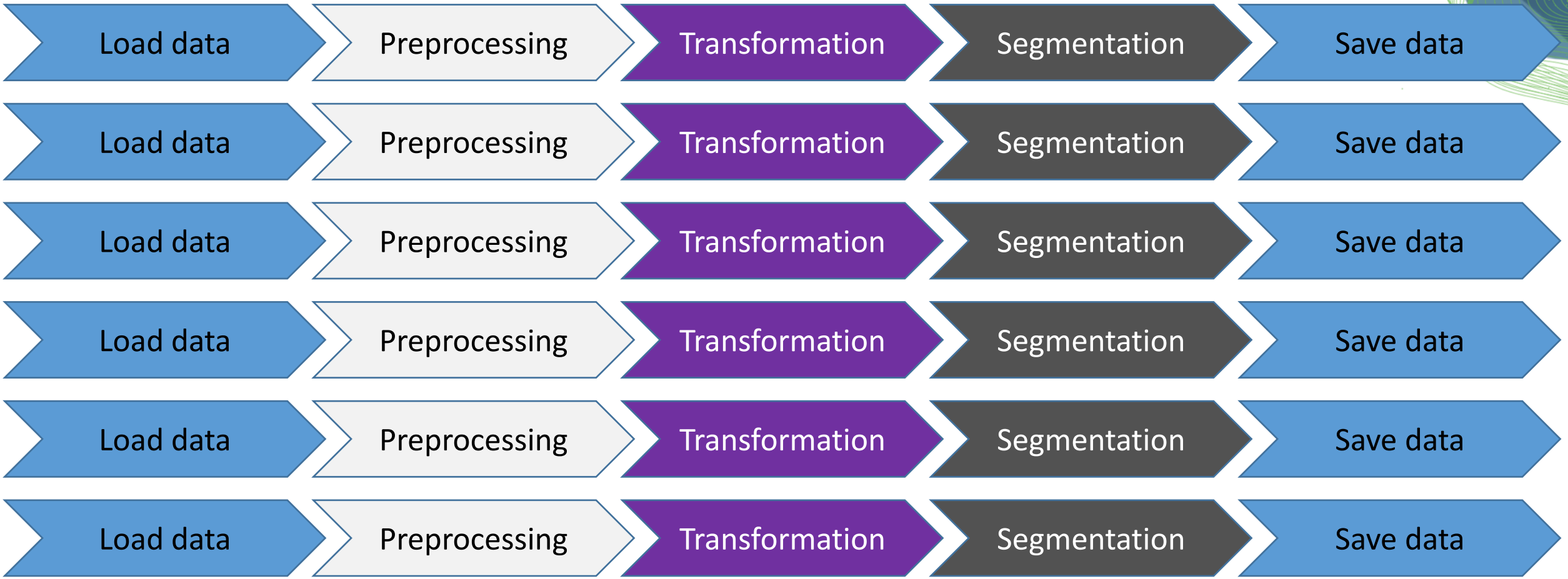
- Processing time-point by time-point is more efficient!



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

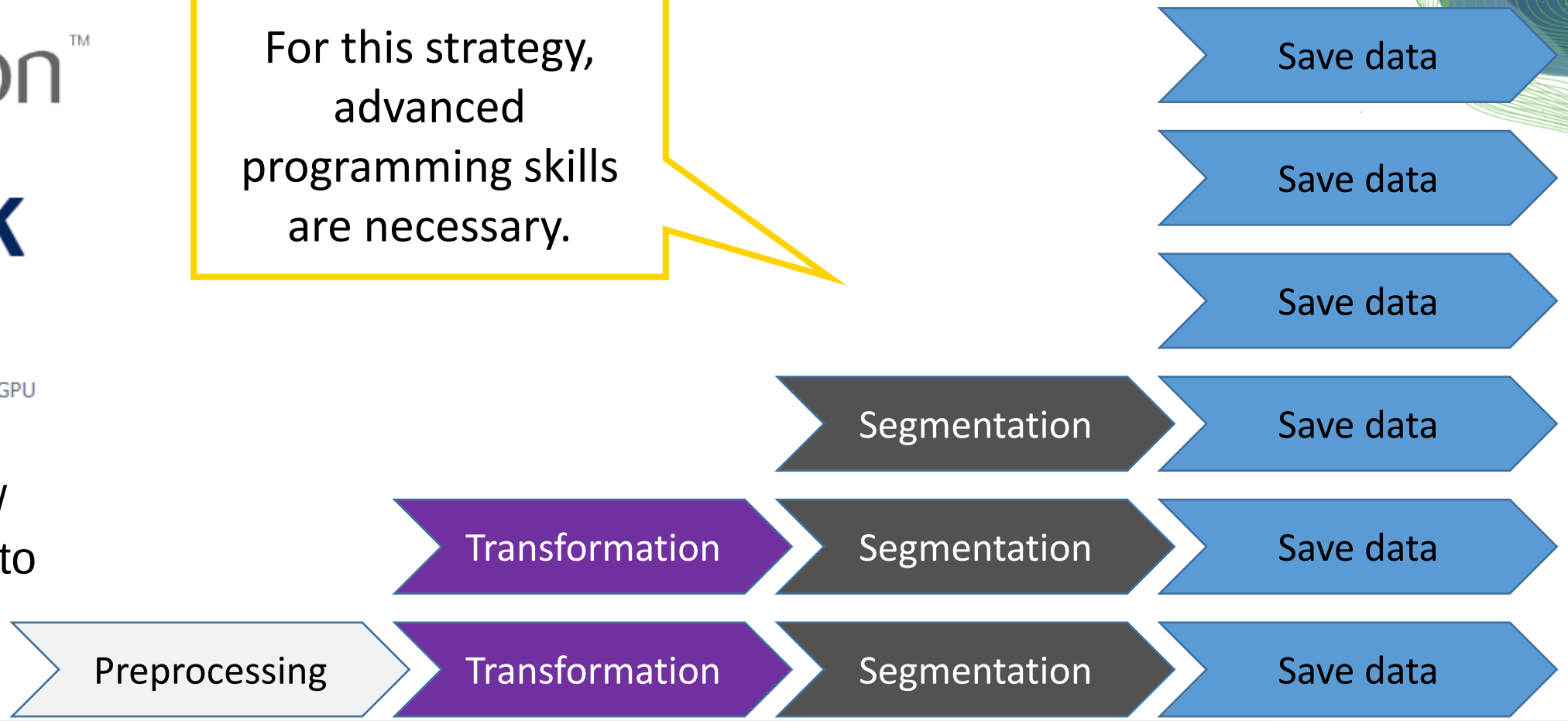
python™

dask

CuPy
NumPy & SciPy for GPU

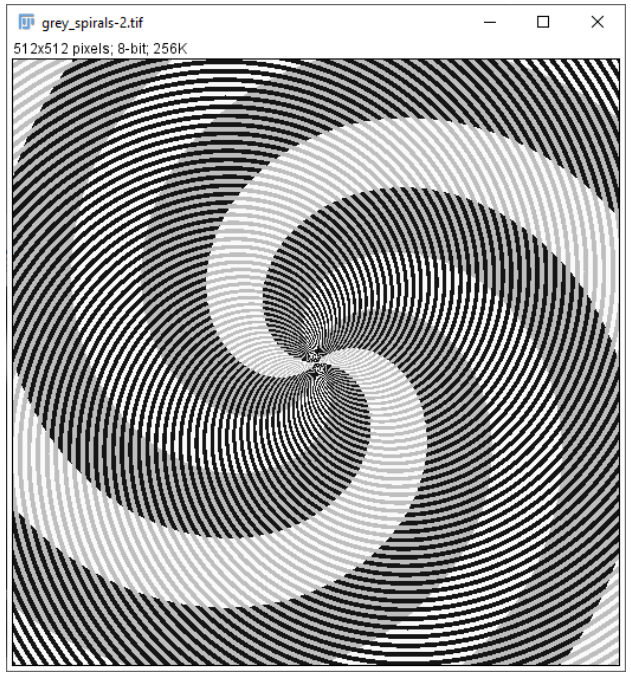
cle._ pyopencl / clesperanto

For this strategy, advanced programming skills are necessary.

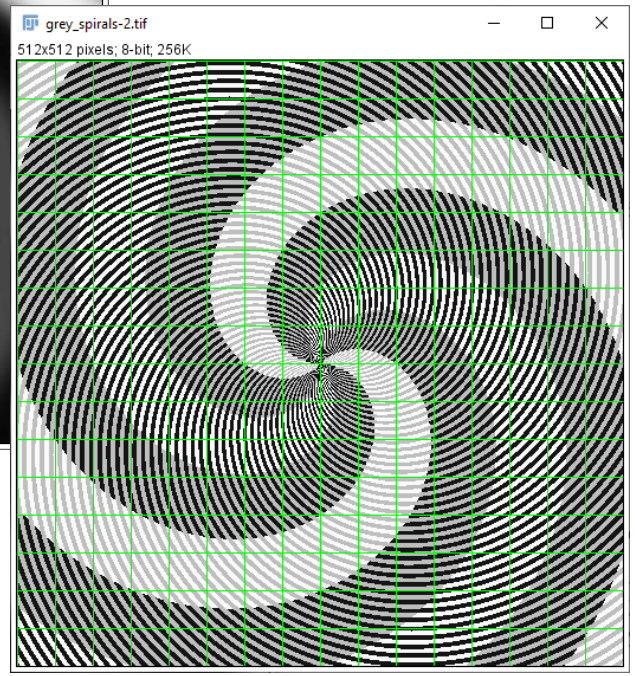


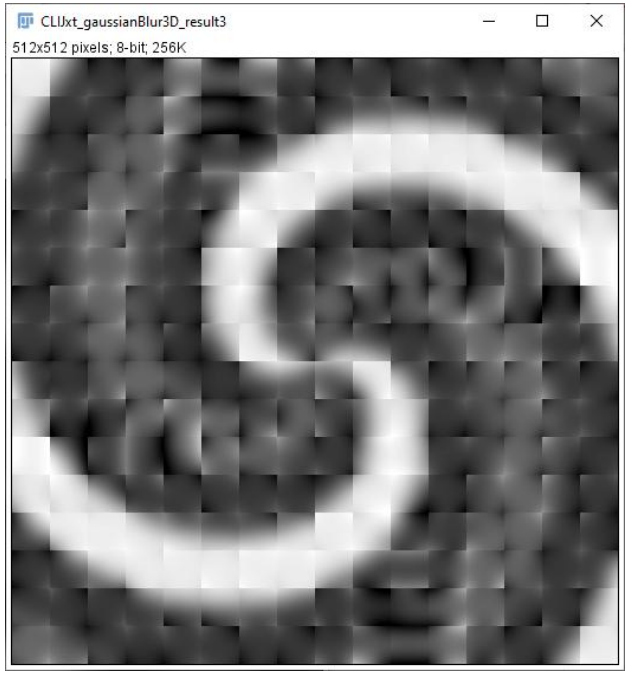
Tiling

- The last perimeter against big data



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





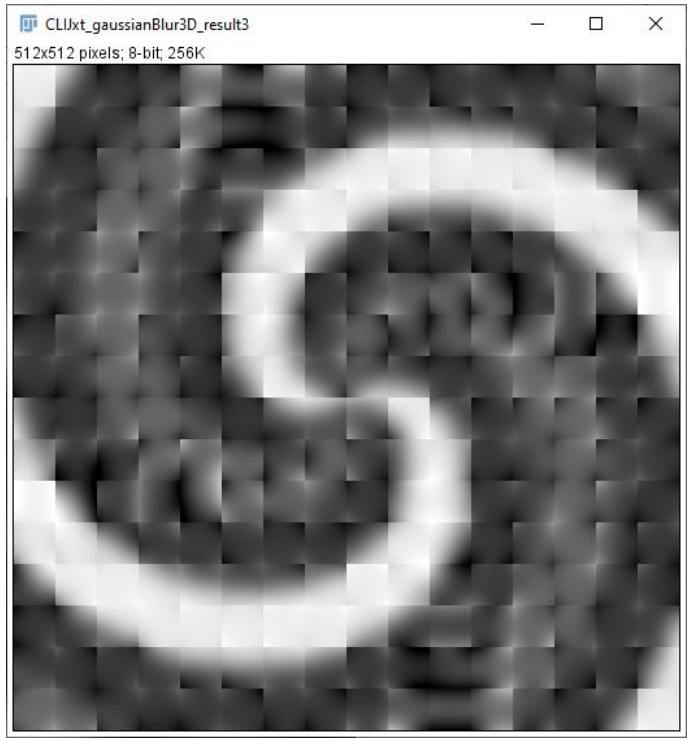
Processing tile-by-tile poses new challenges

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

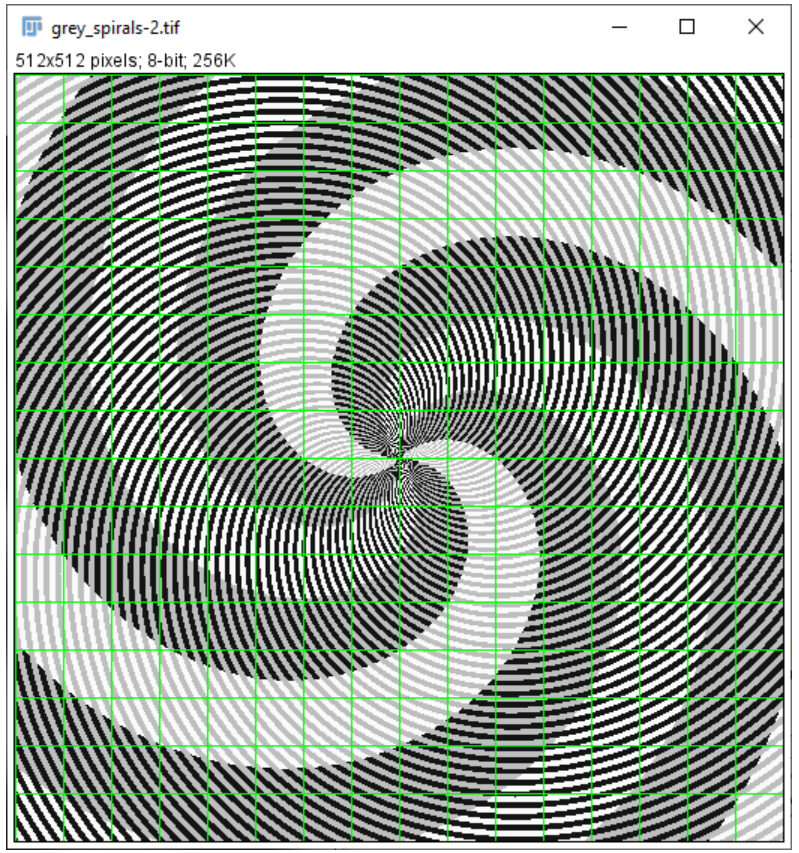


Margin: 10 pixels

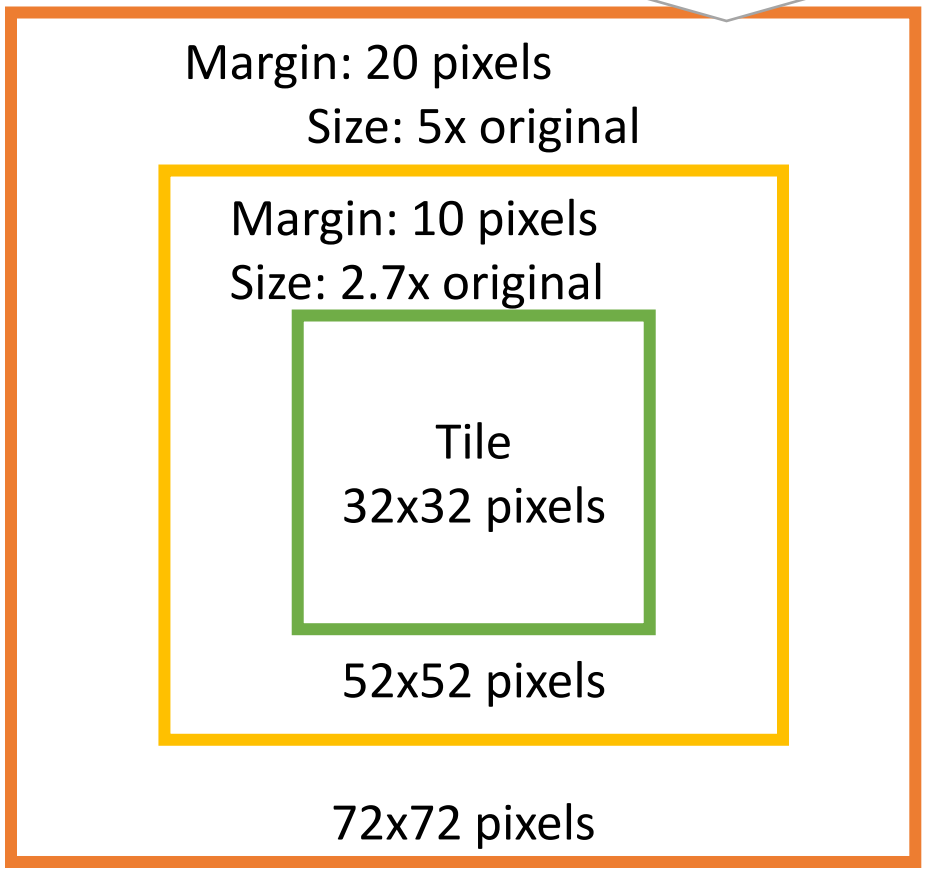
Margin: 20 pixels

Tiling

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



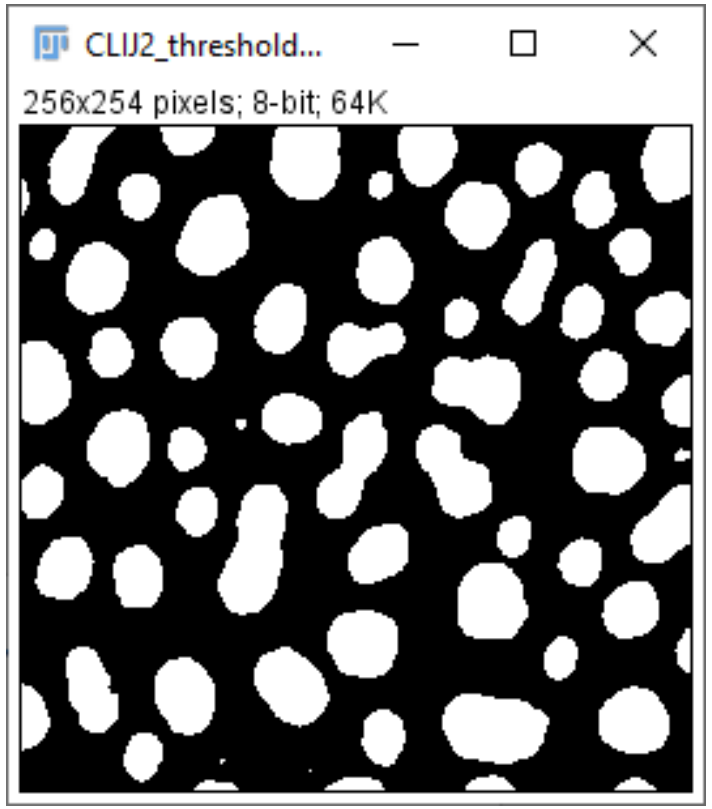
Computation time depends on tile size and margin width

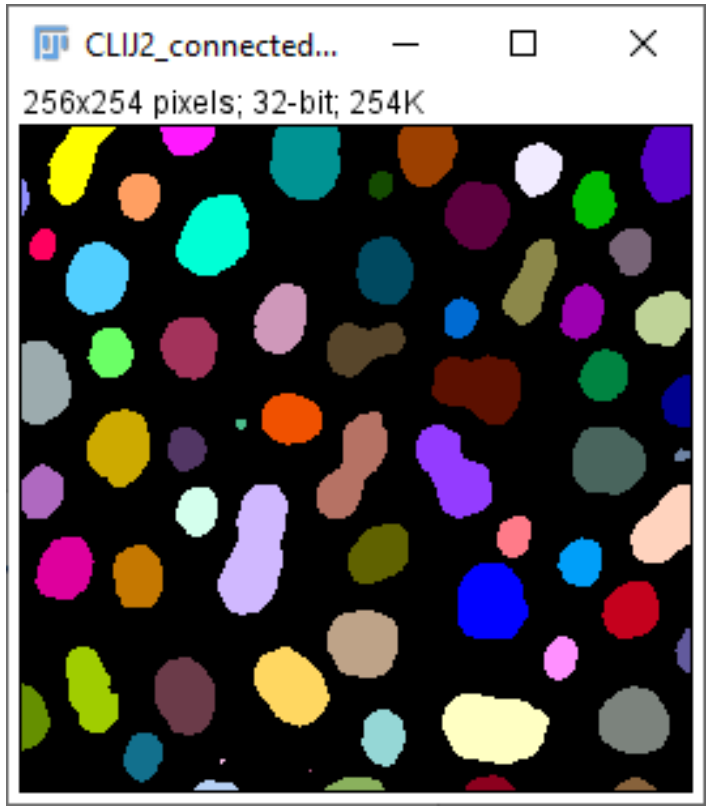


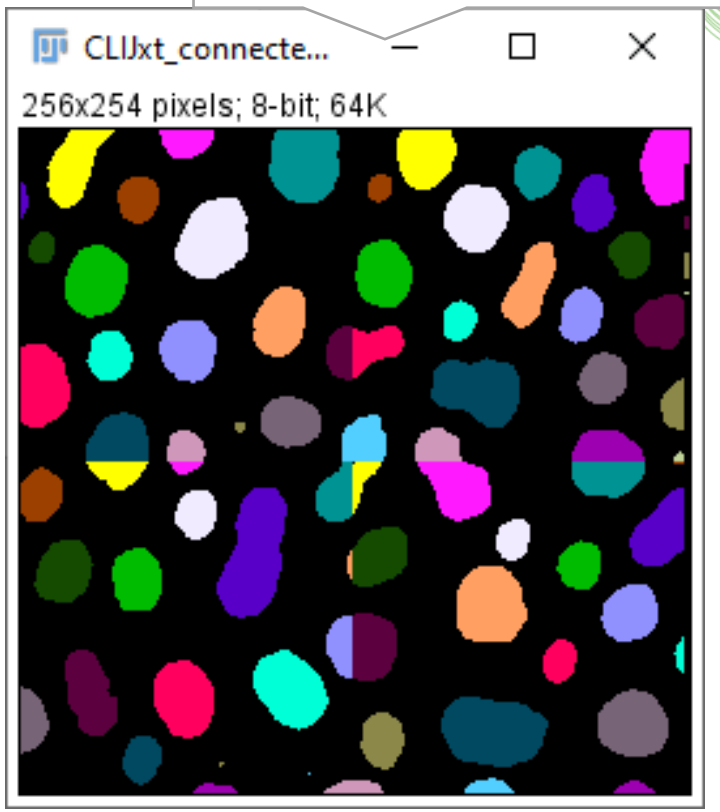
Tiling

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

Checking which labels touch and combine them is feasible.







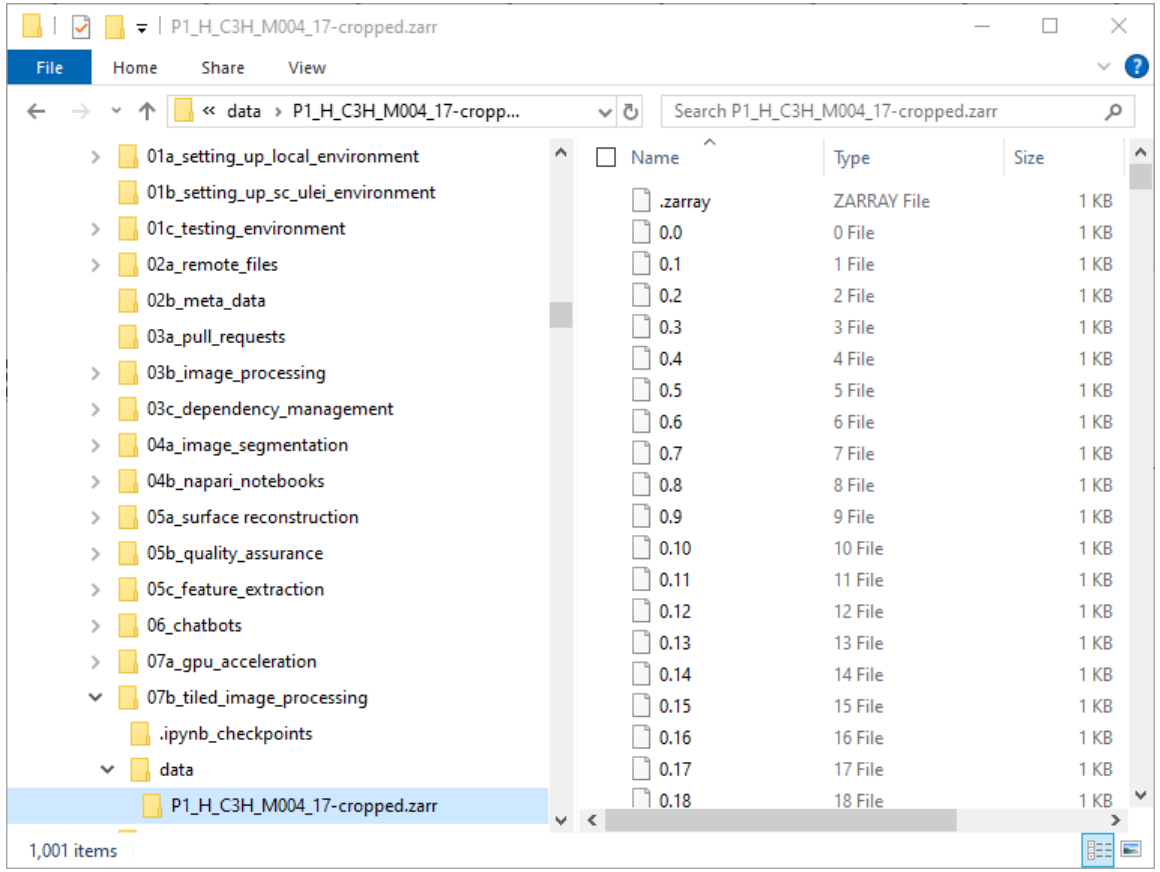
Tiling

- 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

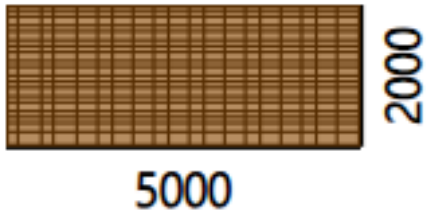
- Key: tiled file formats, for parallel, distributed, lazy loading



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 kiB
Shape	(2000, 5000)	(100, 100)
Dask graph	1000 chunks in 2 graph layers	
Data type	uint8 numpy.ndarray	



Tiled image processing in Python

- Lazy processing

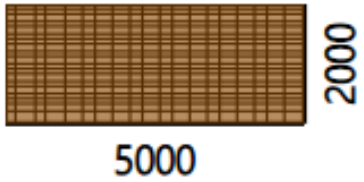
After executing this, no pixel has been read or processed yet.

```
[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 kiB
Shape	(2000, 5000)	(100, 100)
Dask graph	1000 chunks in 3 graph layers	
Data type	float64 numpy.ndarray	



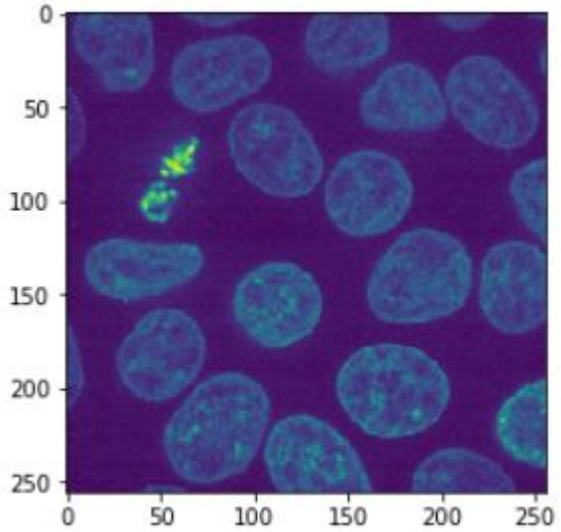
After that, results are avaialble

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

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

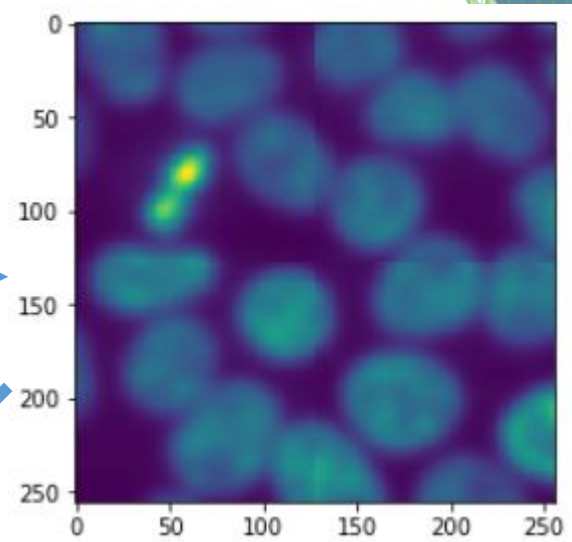
Tiling with/out overlap

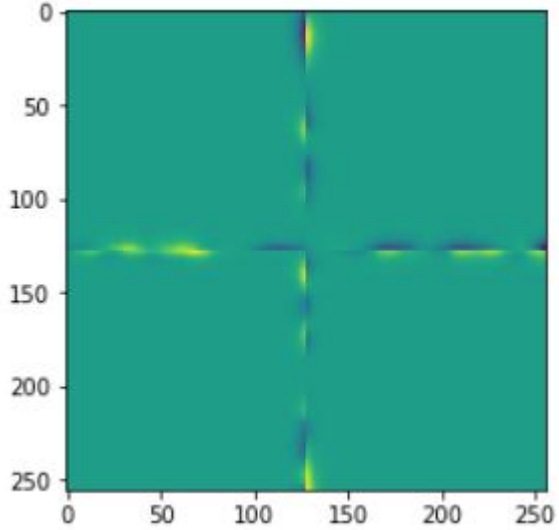
- Processing of images in tiles: artifacts ad tile borders




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

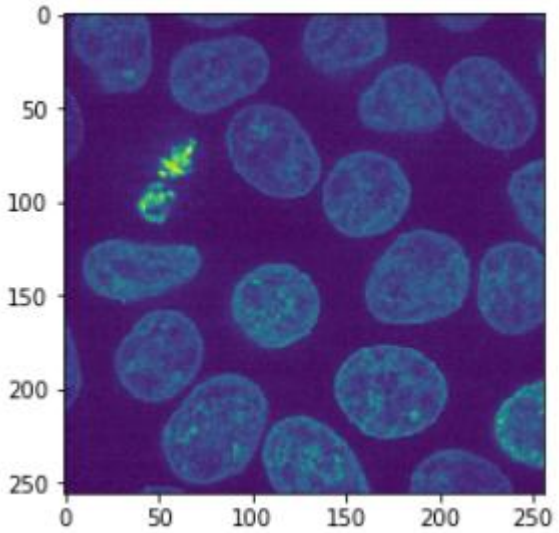
```
result = tile_map.compute()
```





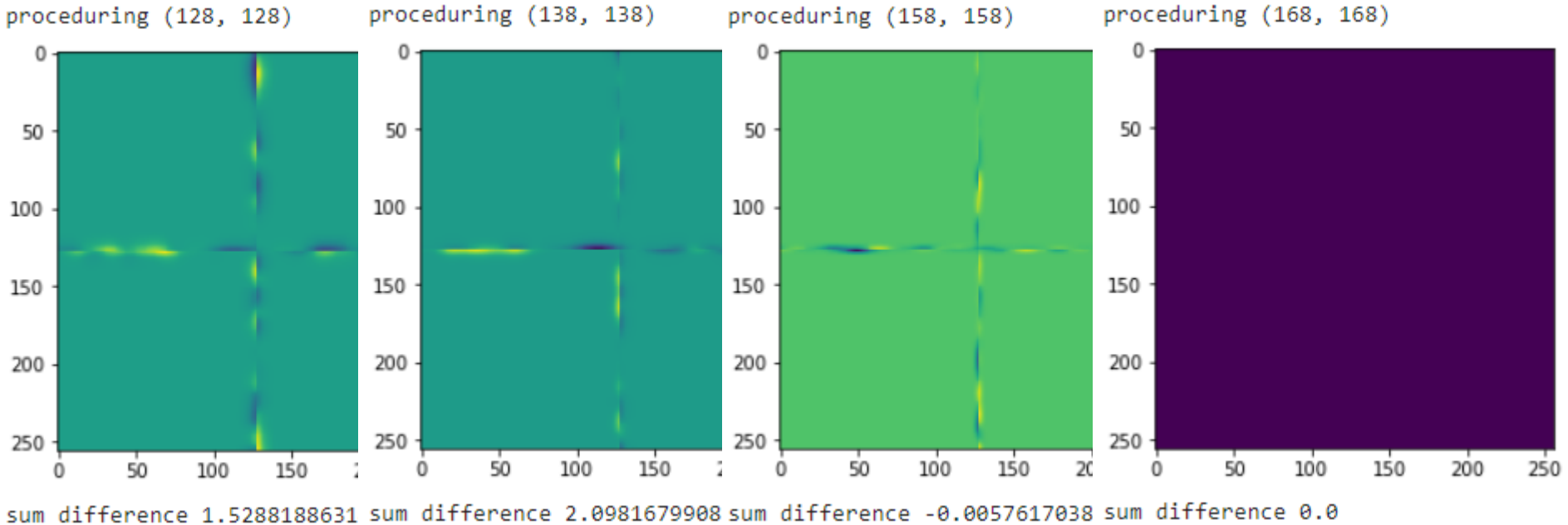
Tiling with/out overlap

- Processing of images in tiles: artifacts at tile borders



```
overlap_width = 1
tile_map = da.map_overlap(procedure, tiles, depth=overlap_width)
```

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



ScaDS.AI
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

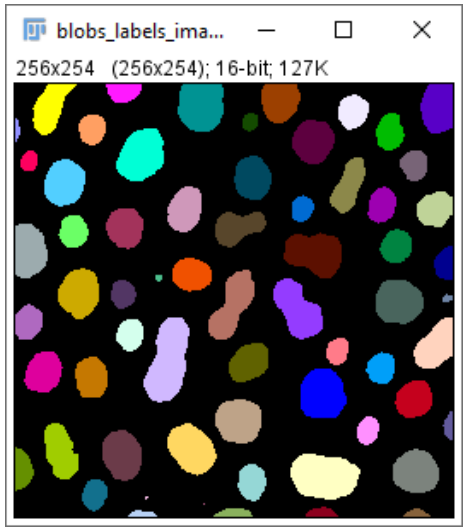
SACHSEN
Diese Maßnahme wird gefördert durch die Bundesregierung aufgrund eines Beschlusses des Deutschen Bundestages.
Diese Maßnahme wird mitfinanziert durch Steuermittel auf der Grundlage des von den Abgeordneten des Sächsischen Landtags beschlossenen Haushaltes.

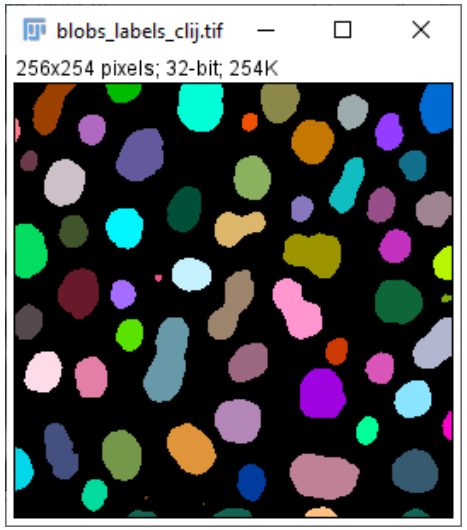
Goal

- 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”)





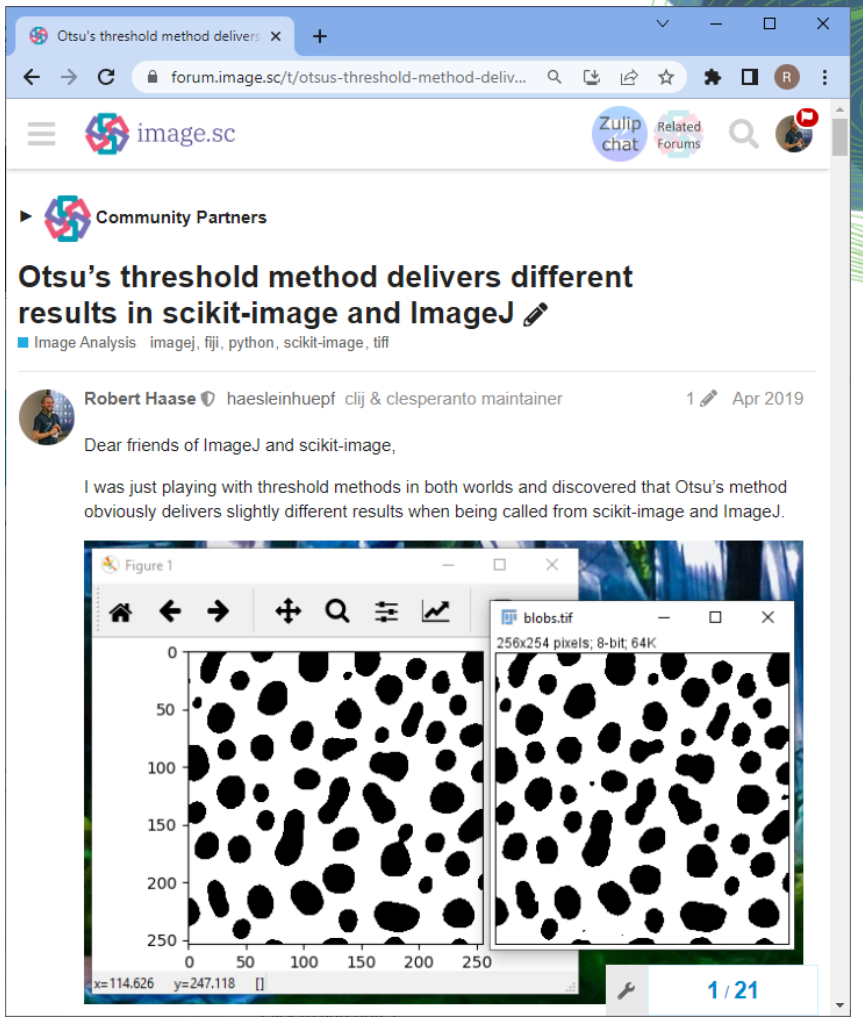
Algorithm result

Why do results vary?

Potential reasons of same workflows delivering different results:

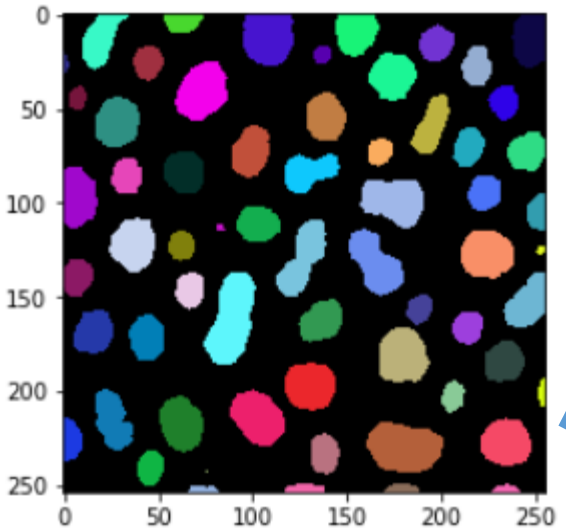
- Image data type (8/16/32-bit float/int)
- Workflow implementation
- How histograms are determined
- How the threshold is determined from the histogram

- Compute architecture (CPU, GPU, TPU, …)
- Hardware vendor
- Software / driver versions

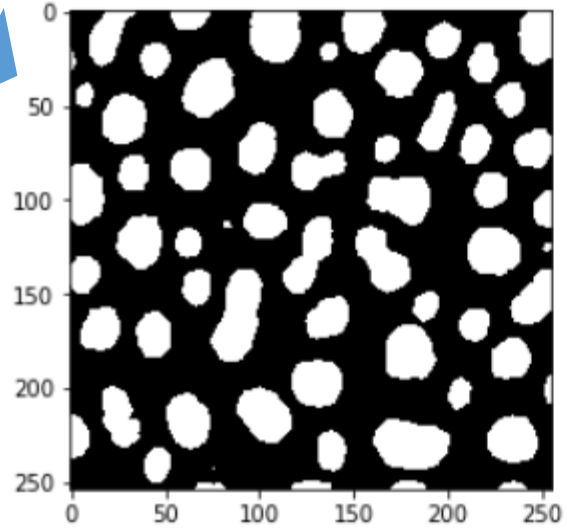


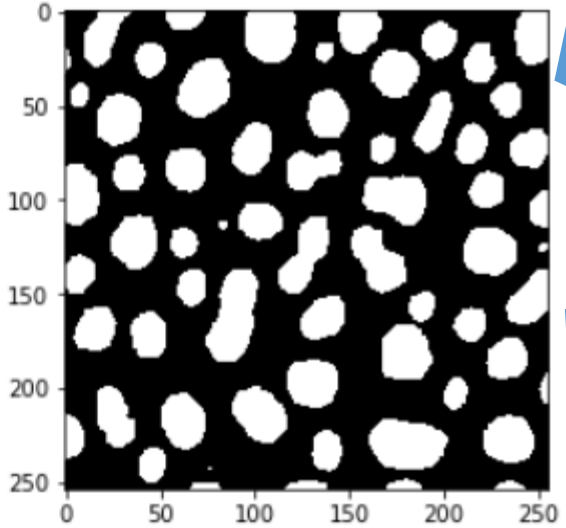
Visual comparison

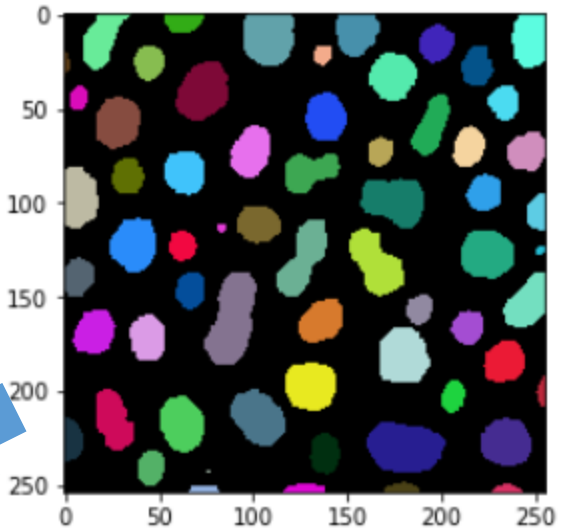
- The order in label images may be different. To compare them visually, we need to turn them into binary images first.

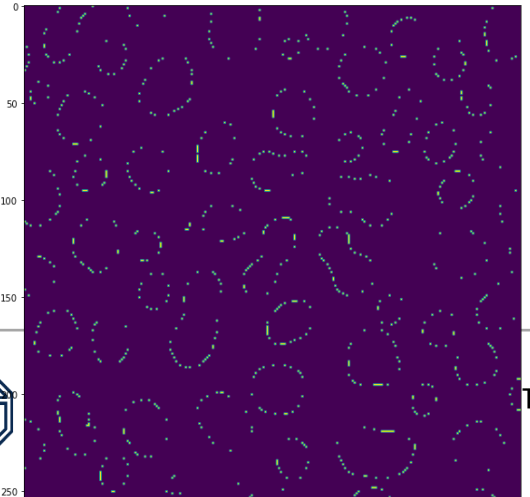


```
binary_blobs_imagej = imread(filenames[0]) > 0
binary_blobs_skimage = imread(filenames[1]) > 0
imshow(binary_blobs_imagej)
imshow(binary_blobs_skimage)
```









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

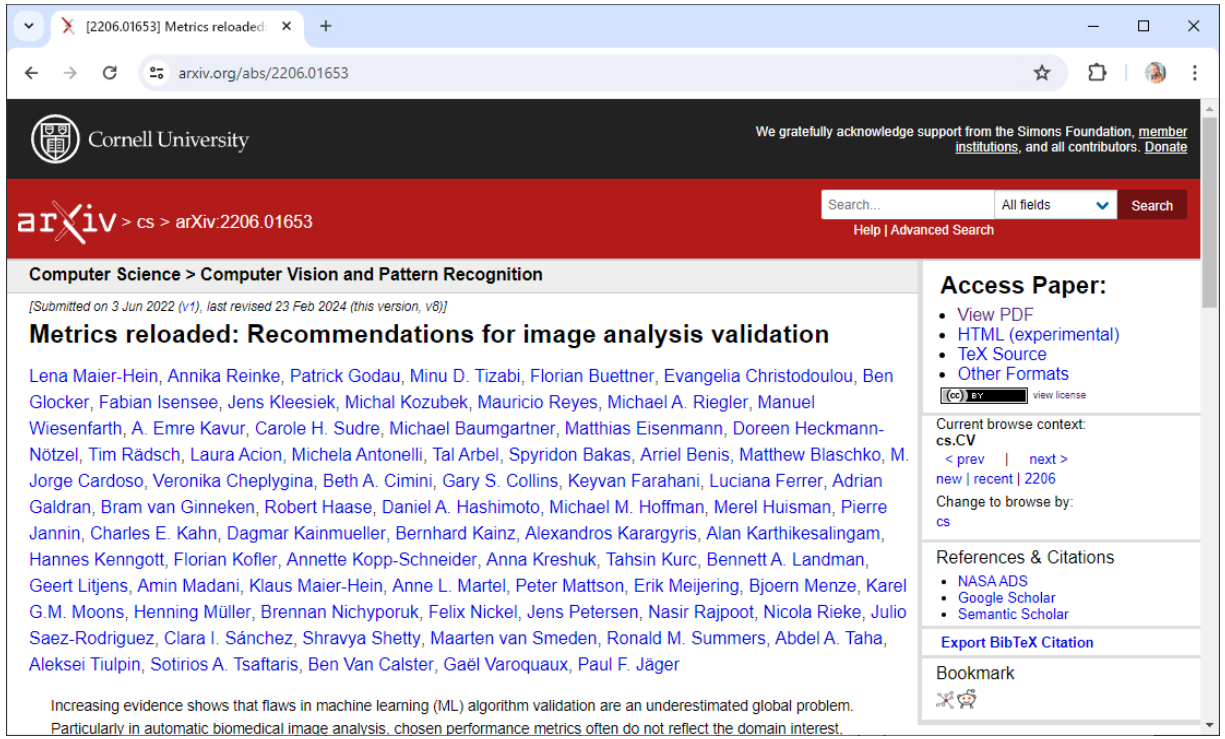
$$e_{Class} = \frac{H}{|TP| + |FN|}$$

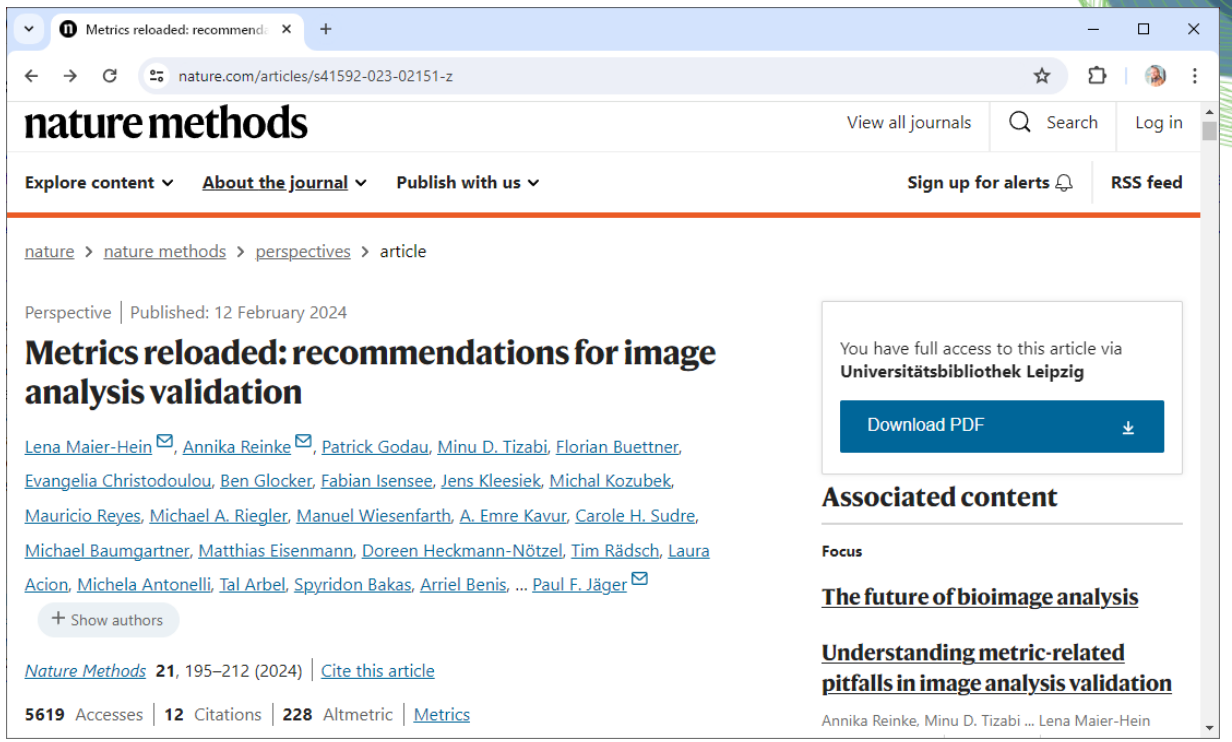
- Hamming distance

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

Choosing the right metric is key

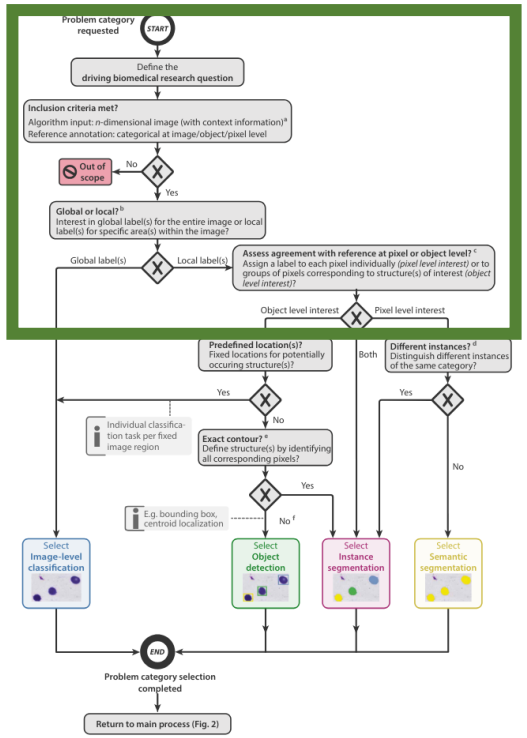
- Systematic overview by Maier-Hein, Reinke at al.





Choosing the right metric is key

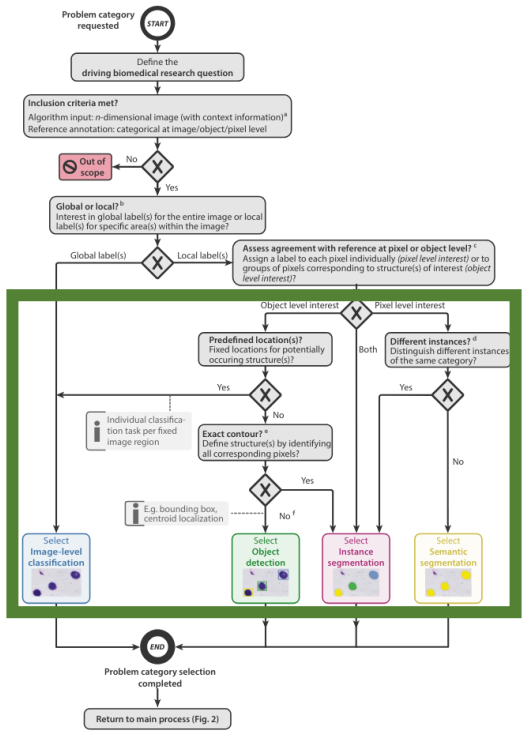
+S1

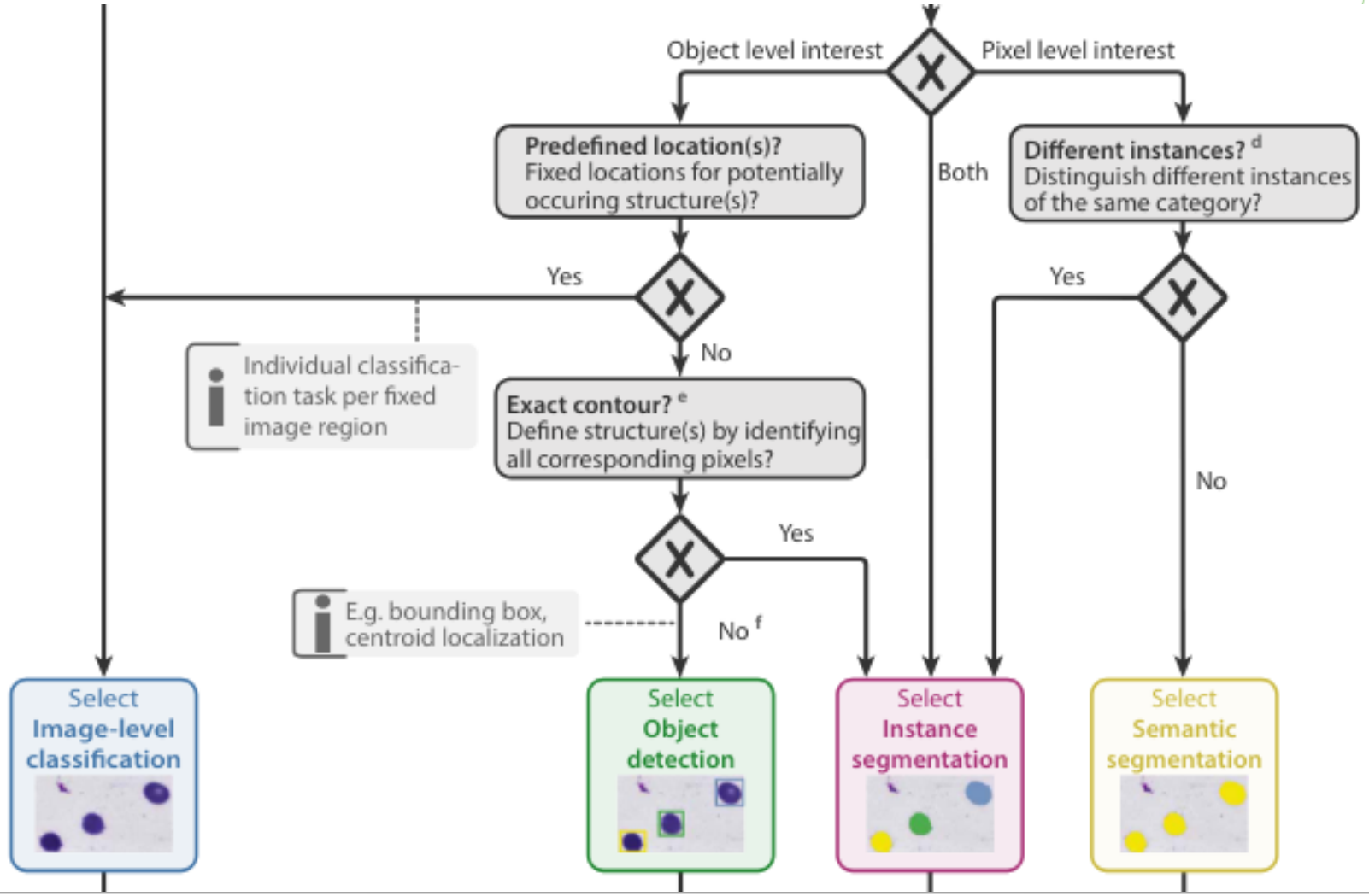


Source: Cropped from Ext. Data Fig 1 in Maier-Hein, Reinke, et al. licensed CC-BY 4.0 https://arxiv.org/pdf/2206.01653.pdf

Choosing the right metric is key

+S1

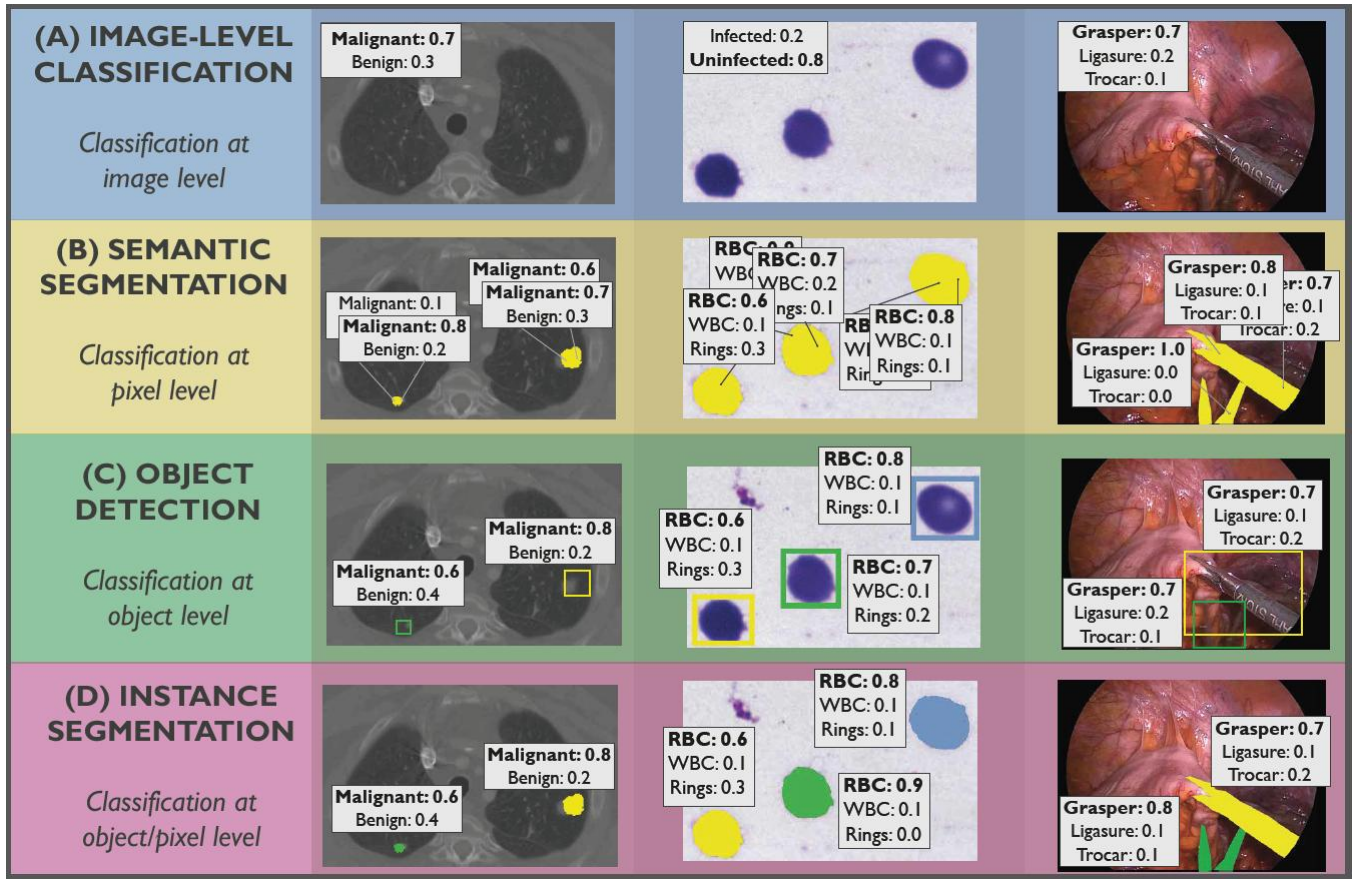




Source: Cropped from Ext. Data Fig 1 in Maier-Hein, Reinke, et al. licensed CC-BY 4.0 https://arxiv.org/pdf/2206.01653.pdf

Choosing the right metric is key

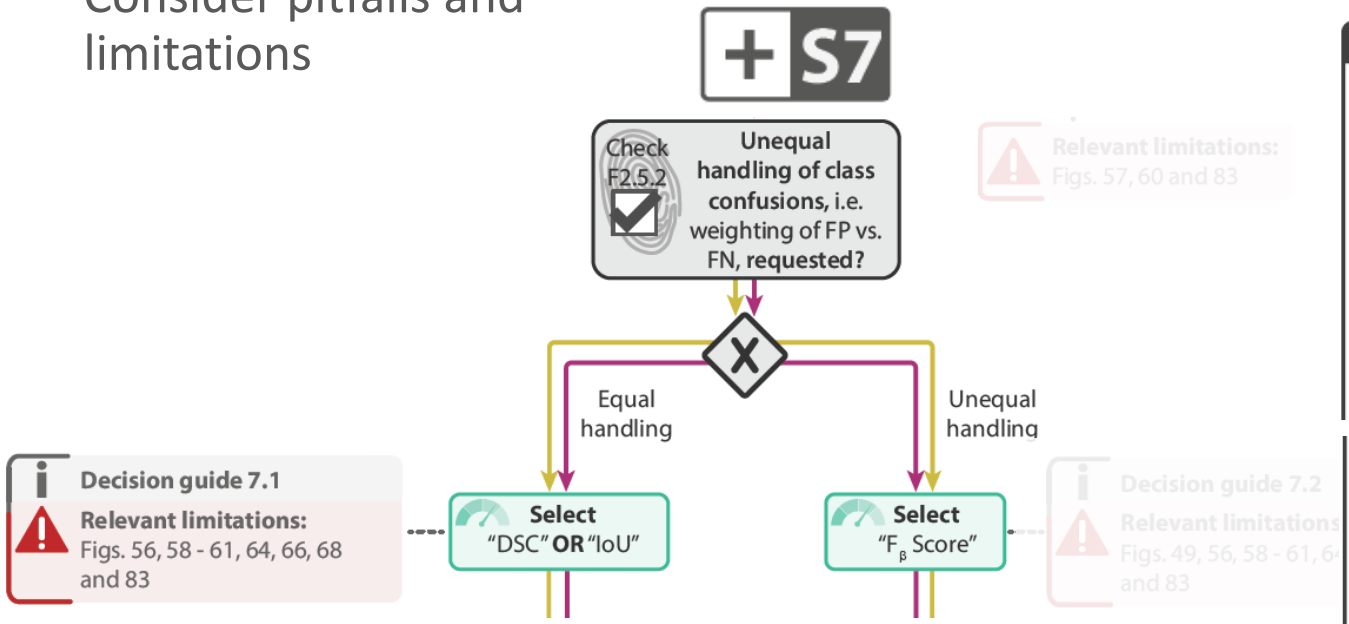
- Define your question

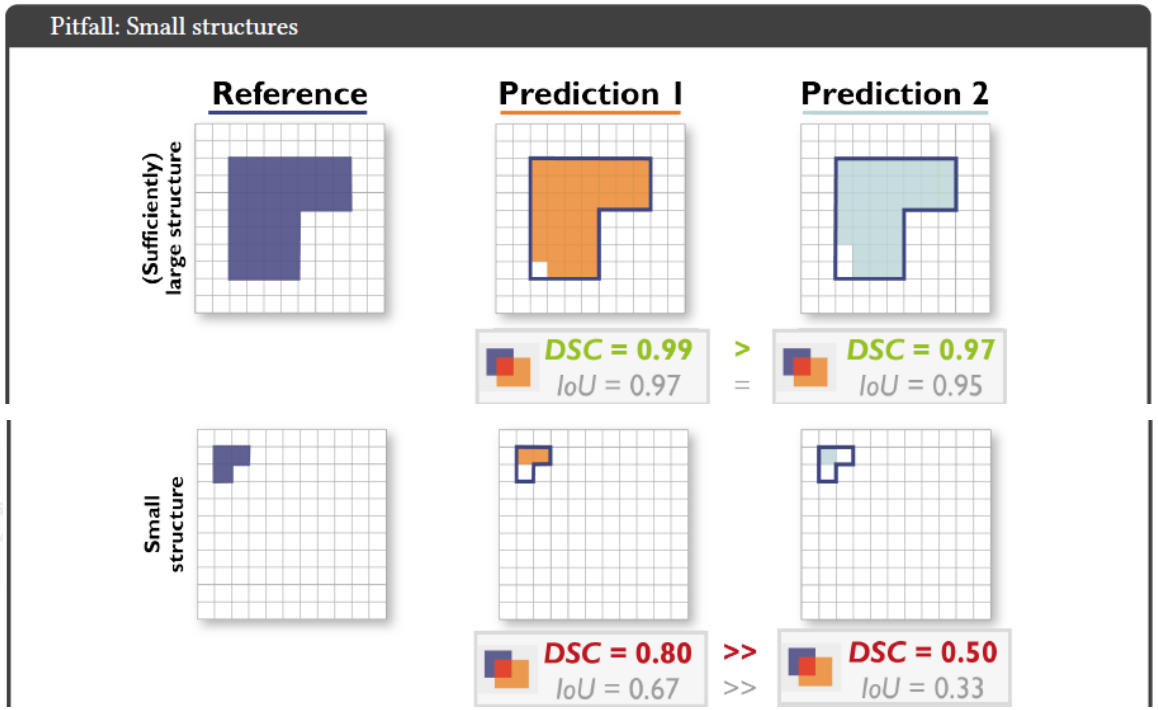


Source: Cropped from Fig 4 in Maier-Hein, Reinke, et al. licensed CC-BY 4.0 https://arxiv.org/pdf/2206.01653.pdf

Overlap metrics

- Consider pitfalls and limitations





(From Figure 56)

E.6 Decision guide S7.

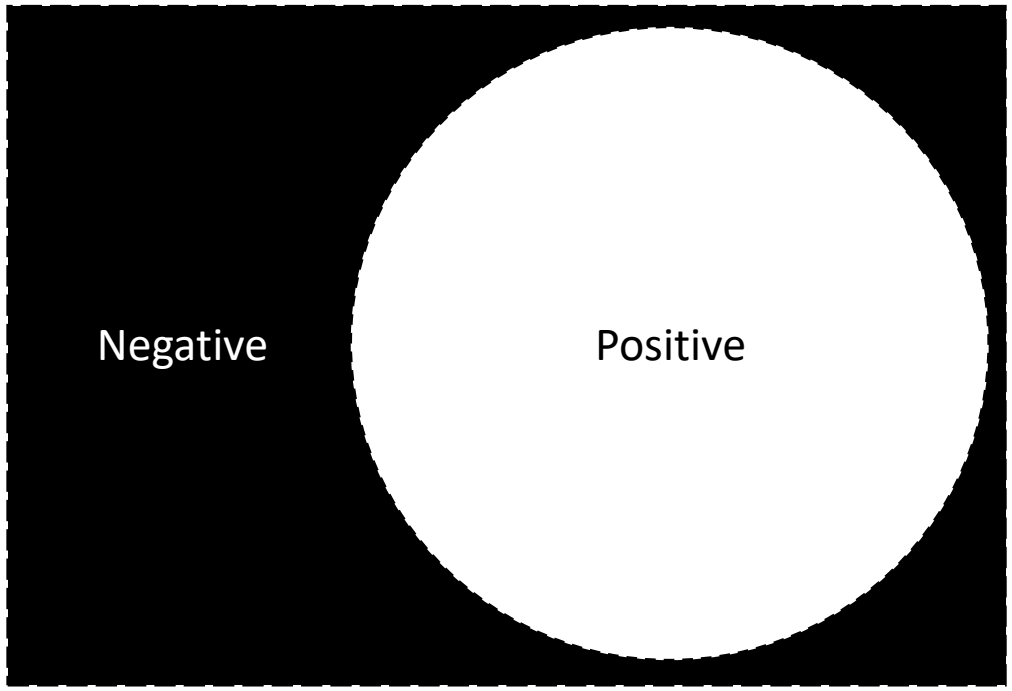
D7.1: Dice vs IoU

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.

Modified (cropped for space reasons) from Maier-Hein, Reinke et al. Arxiv (2023) licensed CC-BY 4.0 https://arxiv.org/pdf/2206.01653.pdf

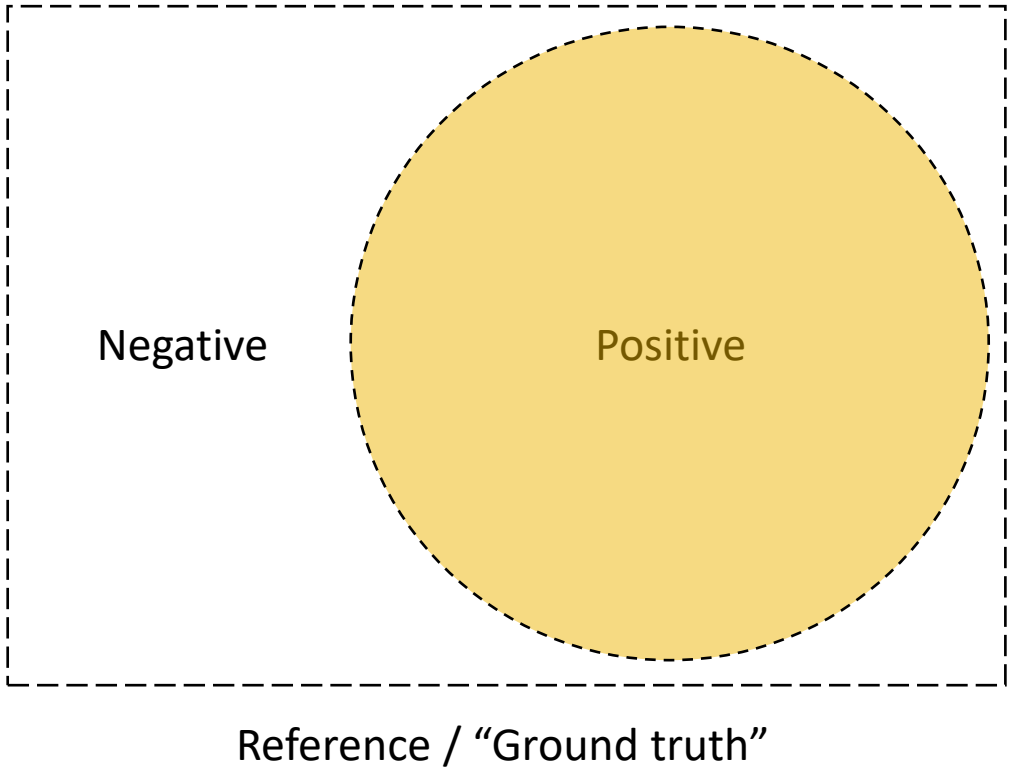
Segmentation quality estimation

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



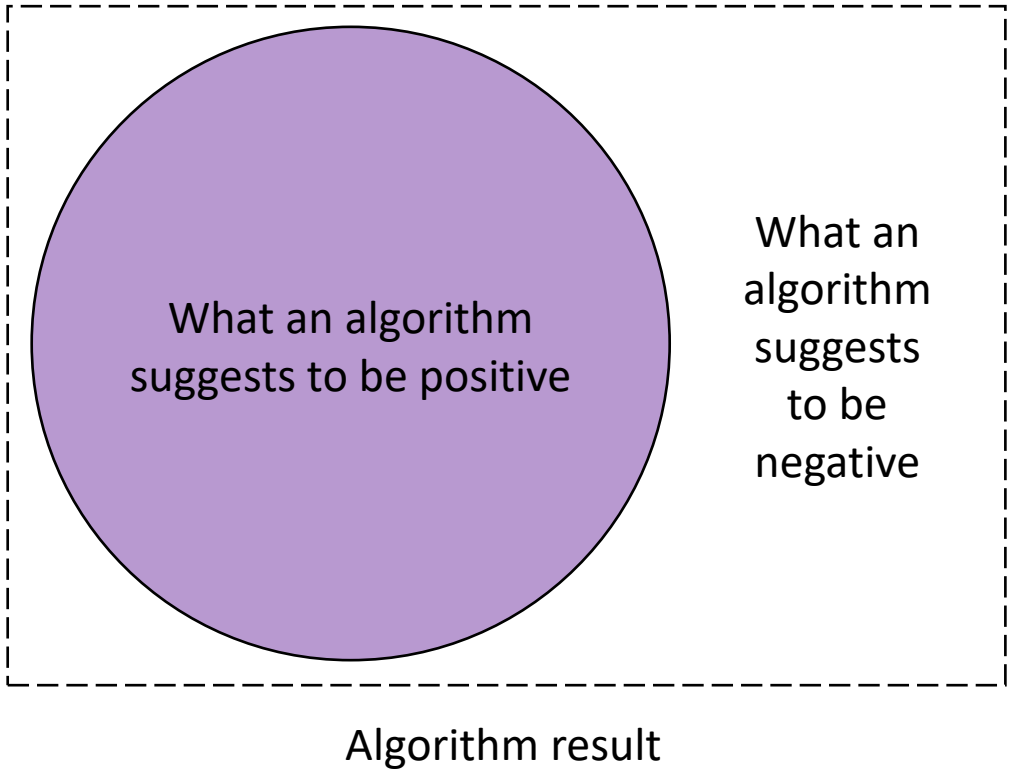
Segmentation quality estimation

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



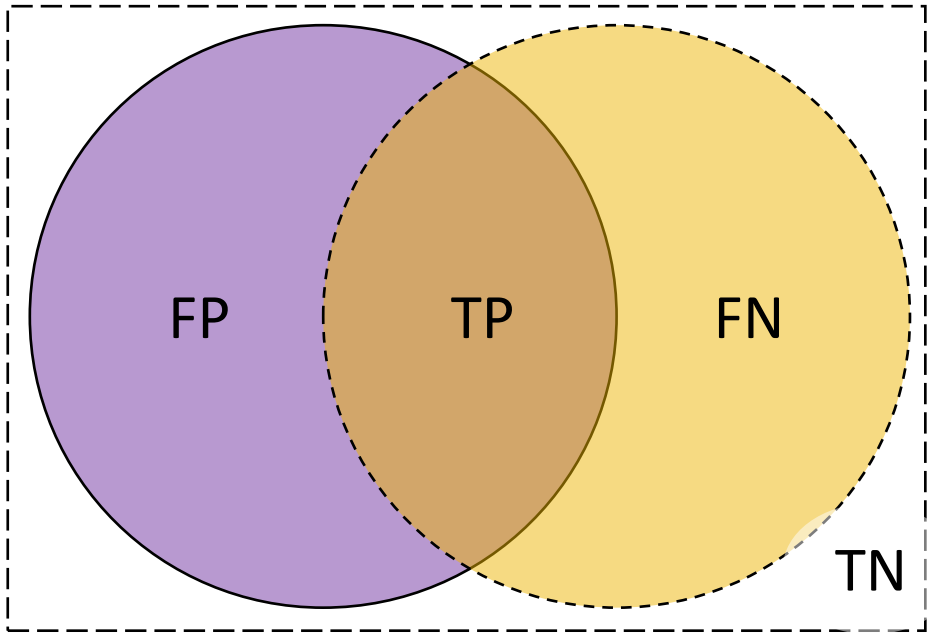
Segmentation quality estimation

- 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



A — Prediction A

B — Reference B (ground truth)

ROI — Region of interest

TP — True-positive

FN — False-negative

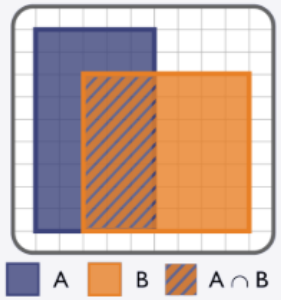
FP — False-positive

TN — True-negative

Dice versus Jaccard Index (IoU)

DICE SIMILARITY COEFFICIENT (DSC)

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



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

VALUE RANGE: [0, 1] ↑

DESCRIPTION
DSC measures the overlap between two structures.

DEFINITION
[Dice, 1945]

RECOMMENDED FOR

ImLC	SemS	ObD	InS
●	●	●	●

IMPORTANT RELATIONS

DSC is closely related to the IoU = Jaccard index:

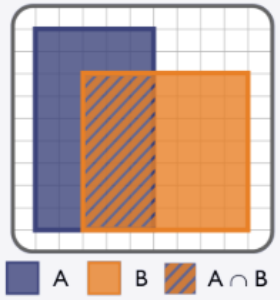
$$DSC = \frac{2 \cdot IoU}{1 + IoU}$$

DSC is equal to the F_1 Score ($\beta = 1$ in F_β Score) at pixel level.

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



$$IoU(A,B) = \frac{}{} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} = \frac{|A \cap B|}{|A \cup B|} = \frac{PPV \cdot Sensitivity}{PPV + Sensitvity - PPV \cdot Sensitivity}$$

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
○	●	○	●

IMPORTANT RELATIONS

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

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).

Source: Cropped from Ext. Data Fig SN 3.5 and 3.9 in Maier-Hein, Reinke, et al. licensed CC-BY 4.0 https://arxiv.org/pdf/2206.01653.pdf

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

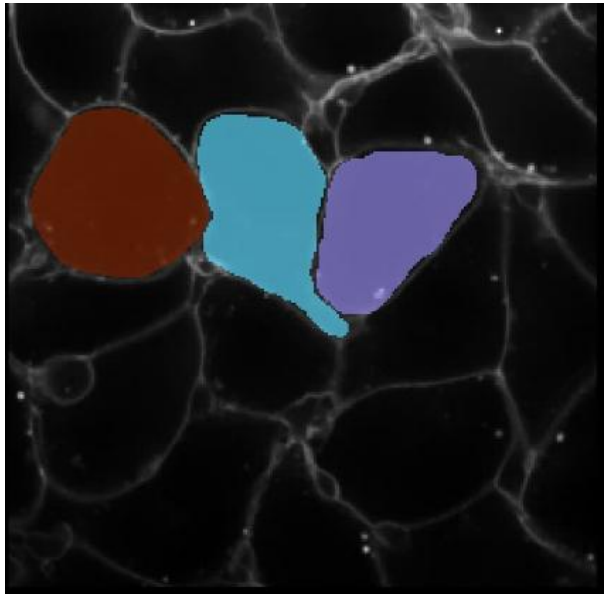
IoU

- 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

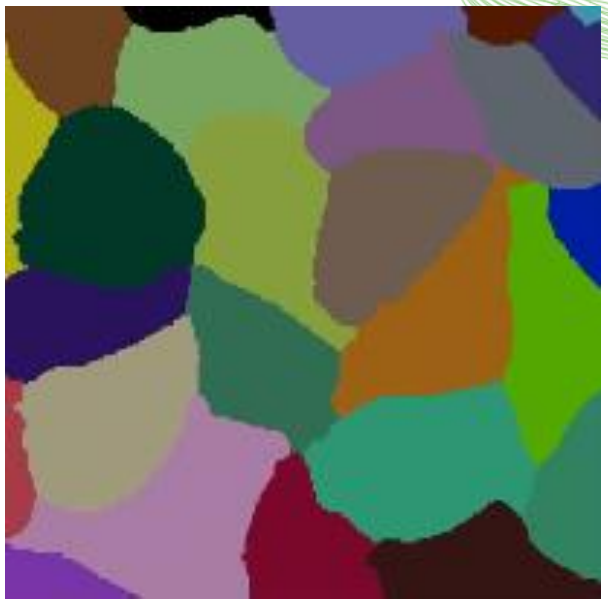
- For every annotated object, we compute the maximum IoU with any segmented object.
- We average this value over all annotated objects



Sparse
instance annotation

IoU = 0.35

IoU = 0.66

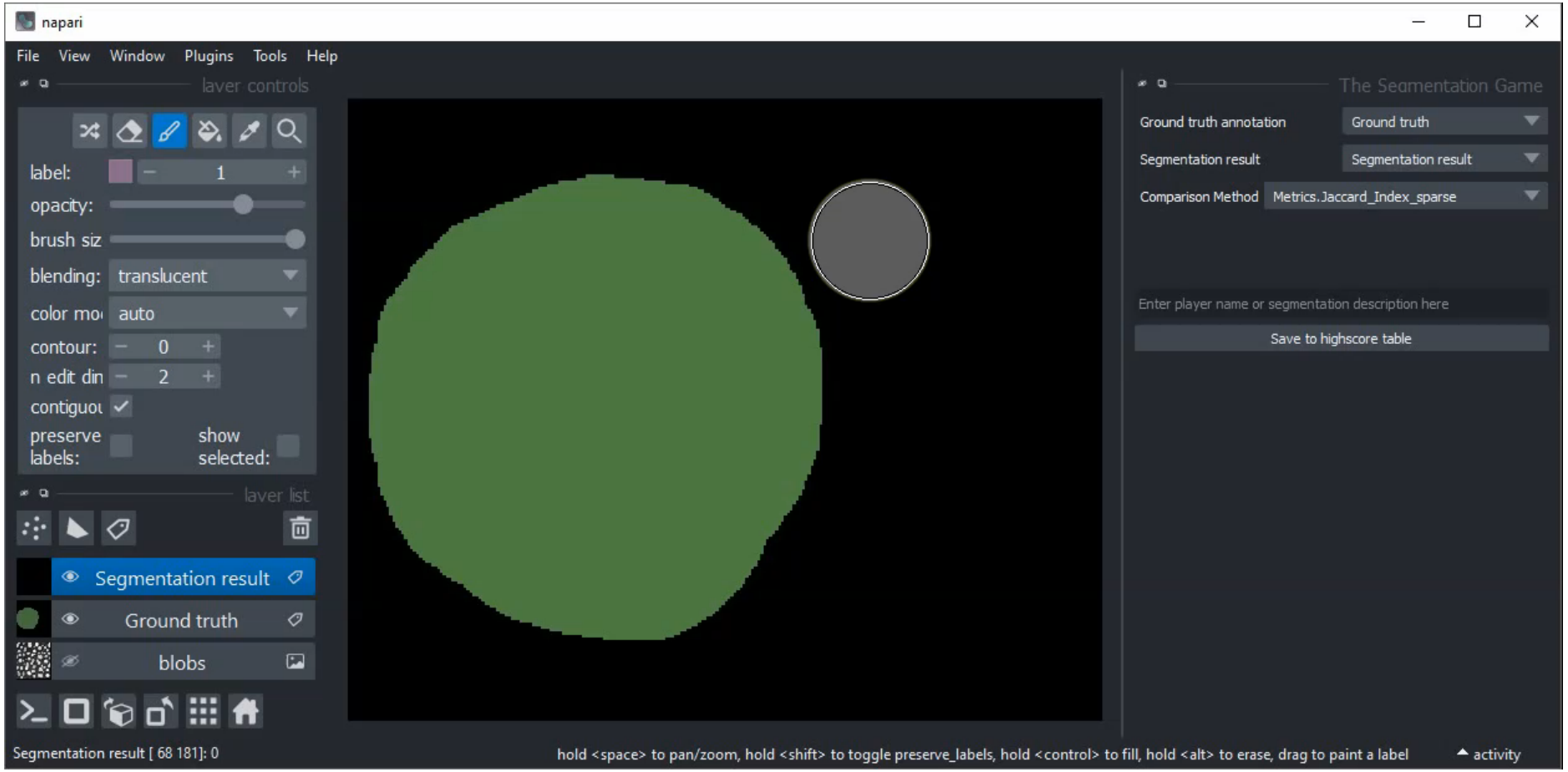


IoU = 0.69

Instance segmentation candidates

Sparse Jaccard Index

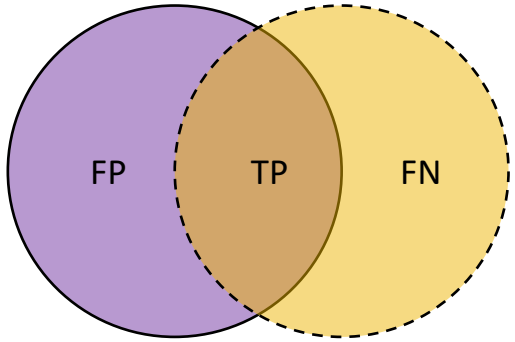
- Play with metrics to gain understanding

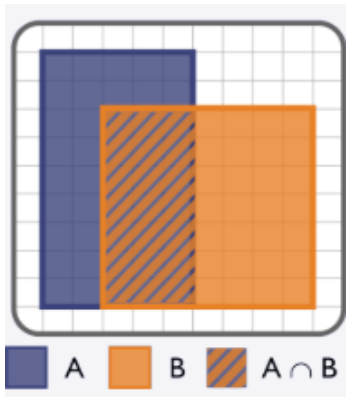


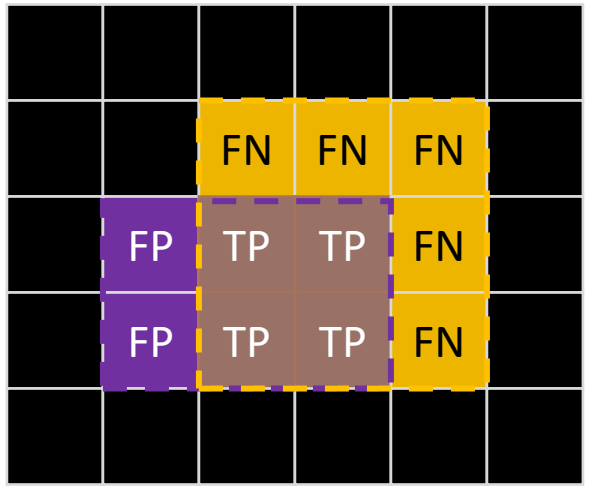
With Martin Schätz
(Charles Uni, Prague)
@schatzcz

Pixel-wise versus Object-wise evaluation

- Pixel wise: Segmentation quality







True-positive: 4

False-negative: 5

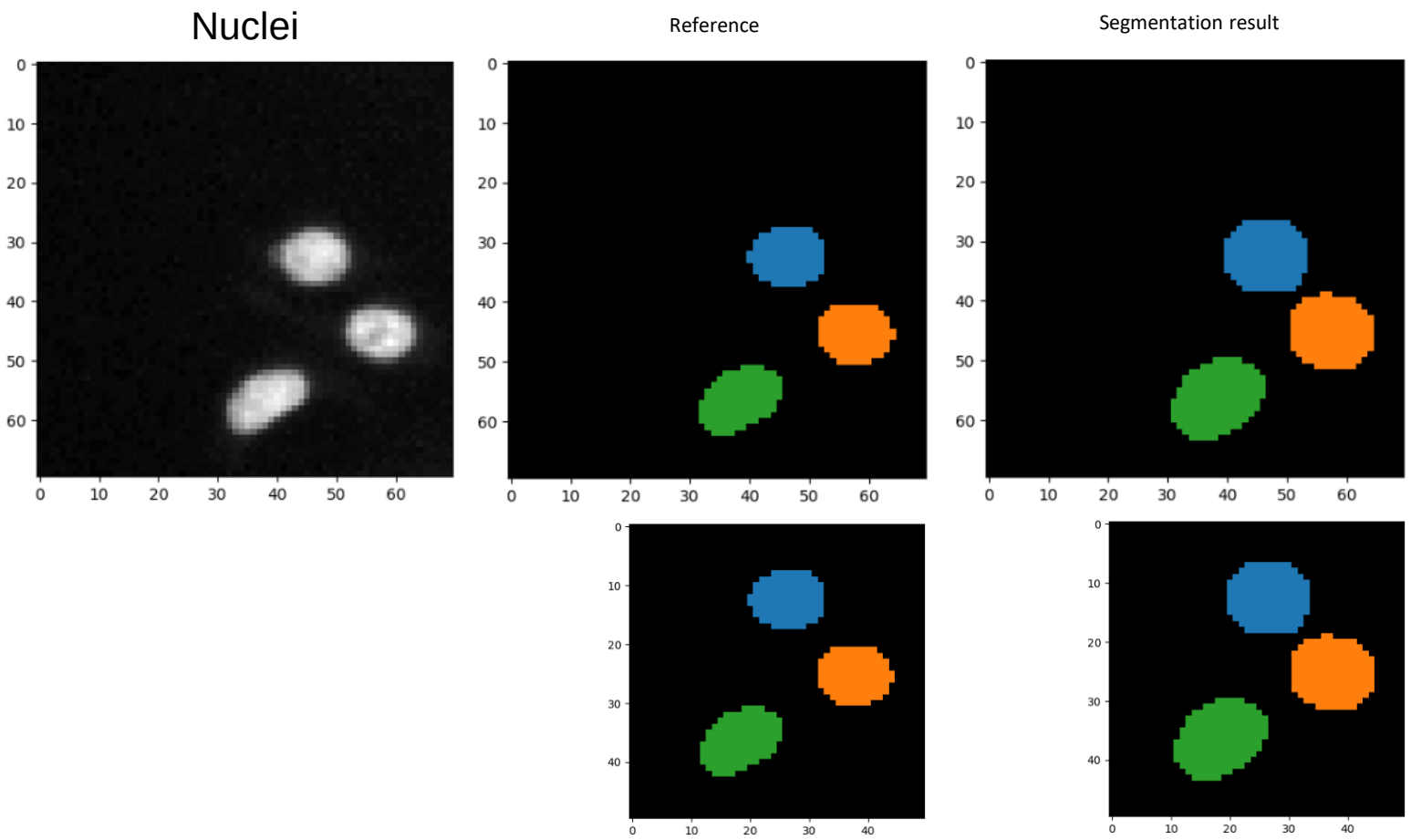
False-positive: 2

$$IoU(A,B) = \frac{\square}{\square + \square - \square} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} = \frac{|A \cap B|}{|A \cup B|}$$

IoU = 4 / 11

Accuracy versus Jaccard Index (IoU)

- Side-effects of image size and number of nuclei



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

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

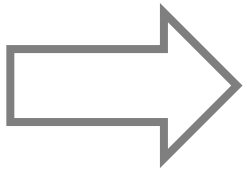
Accuracy: 0.97
Jaccard Index: 0.73

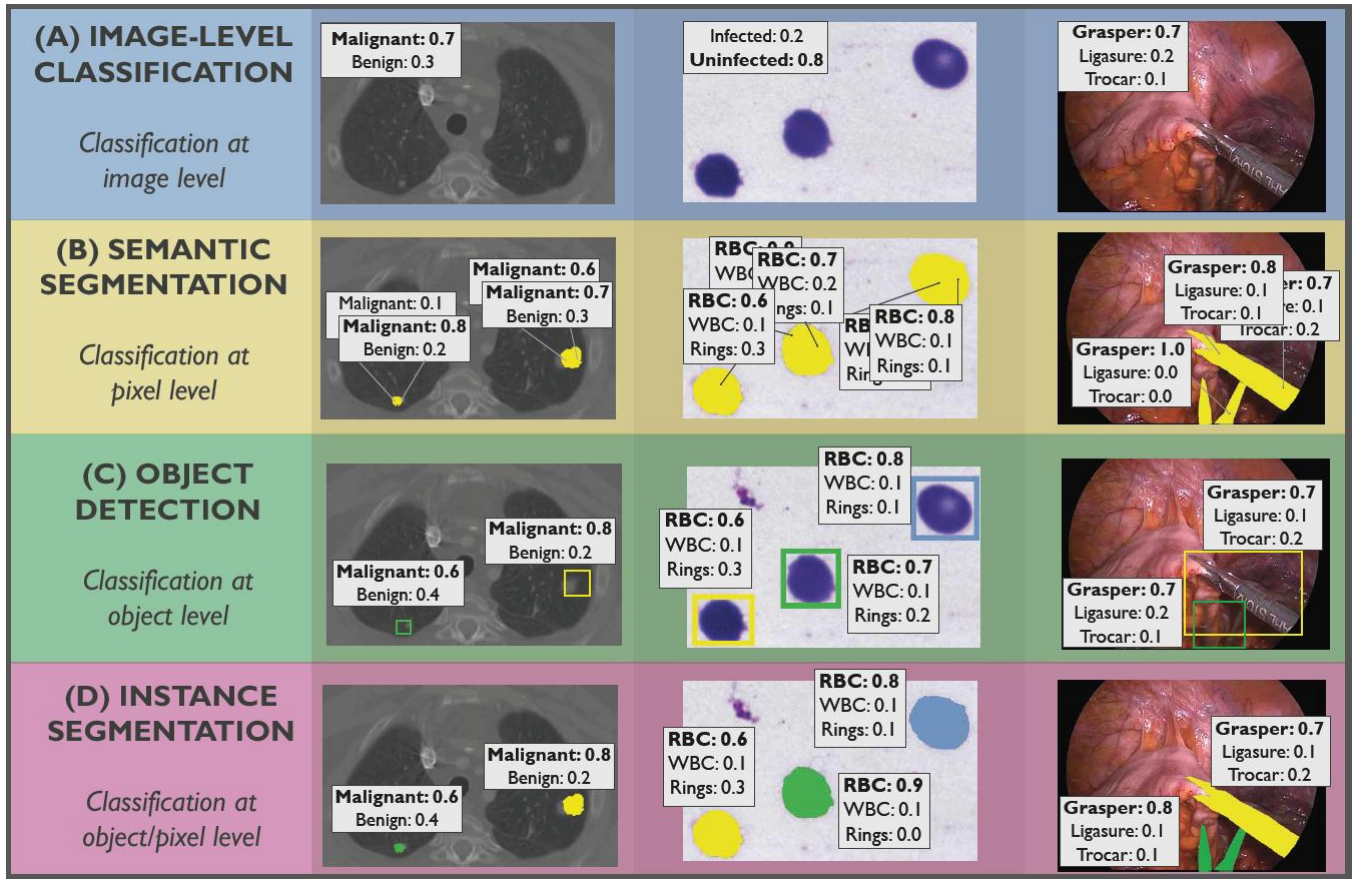
Accuracy decreases because there are less correct black pixels (TN)

Accuracy: 0.95
Jaccard Index: 0.73

Choosing the right metric is key

- Define your question

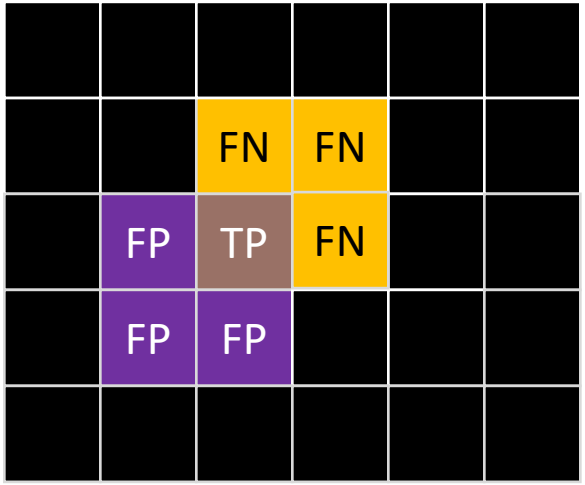




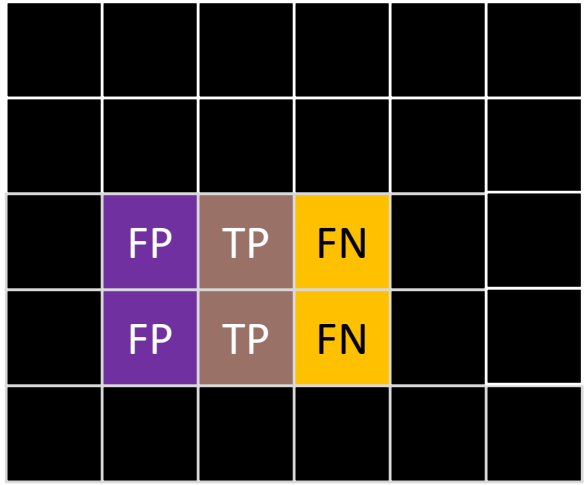
Source: Cropped from Fig 4 in Maier-Hein, Reinke, et al. licensed CC-BY 4.0 https://arxiv.org/pdf/2206.01653.pdf

Pixel-wise versus Object-wise evaluation

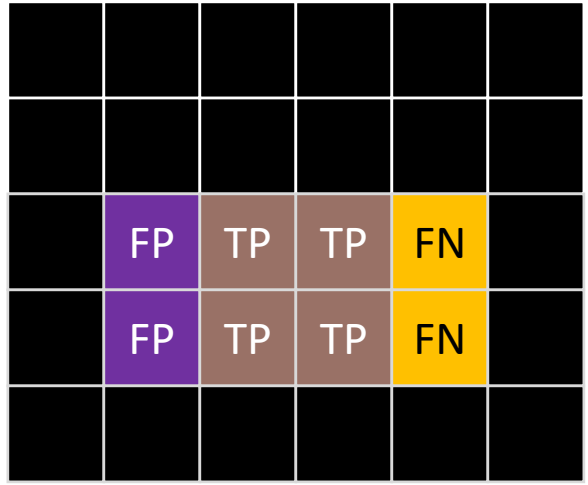
- Are these objects overlapping?



IoU = 1 / 7



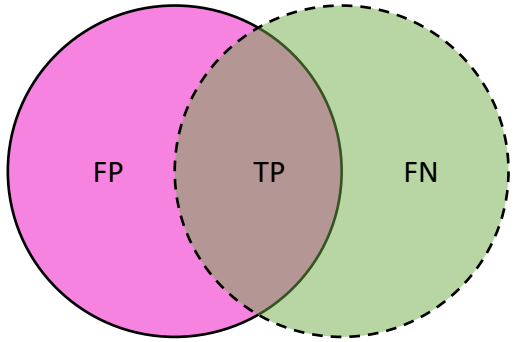
IoU = 1 / 3

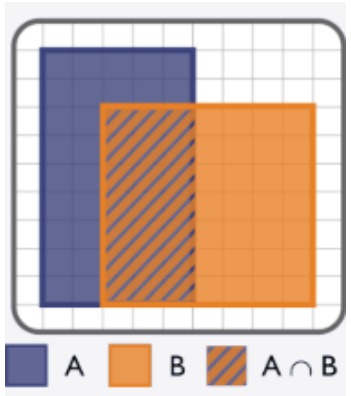


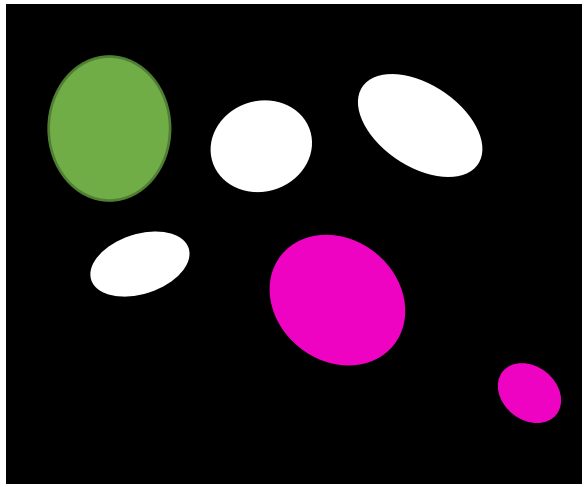
IoU = 1 / 2

Pixel-wise versus Object-wise evaluation

- Object wise: Detection quality







True-positive: 3

False-negative: 1

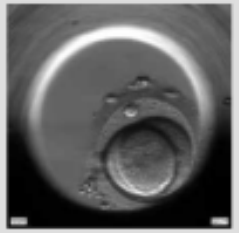
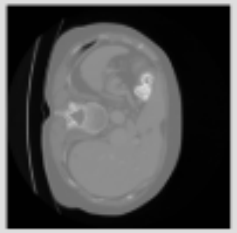
False-positive: 2

$$IoU(A,B) = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} = \frac{|A \cap B|}{|A \cup B|}$$

IoU = 1 / 2

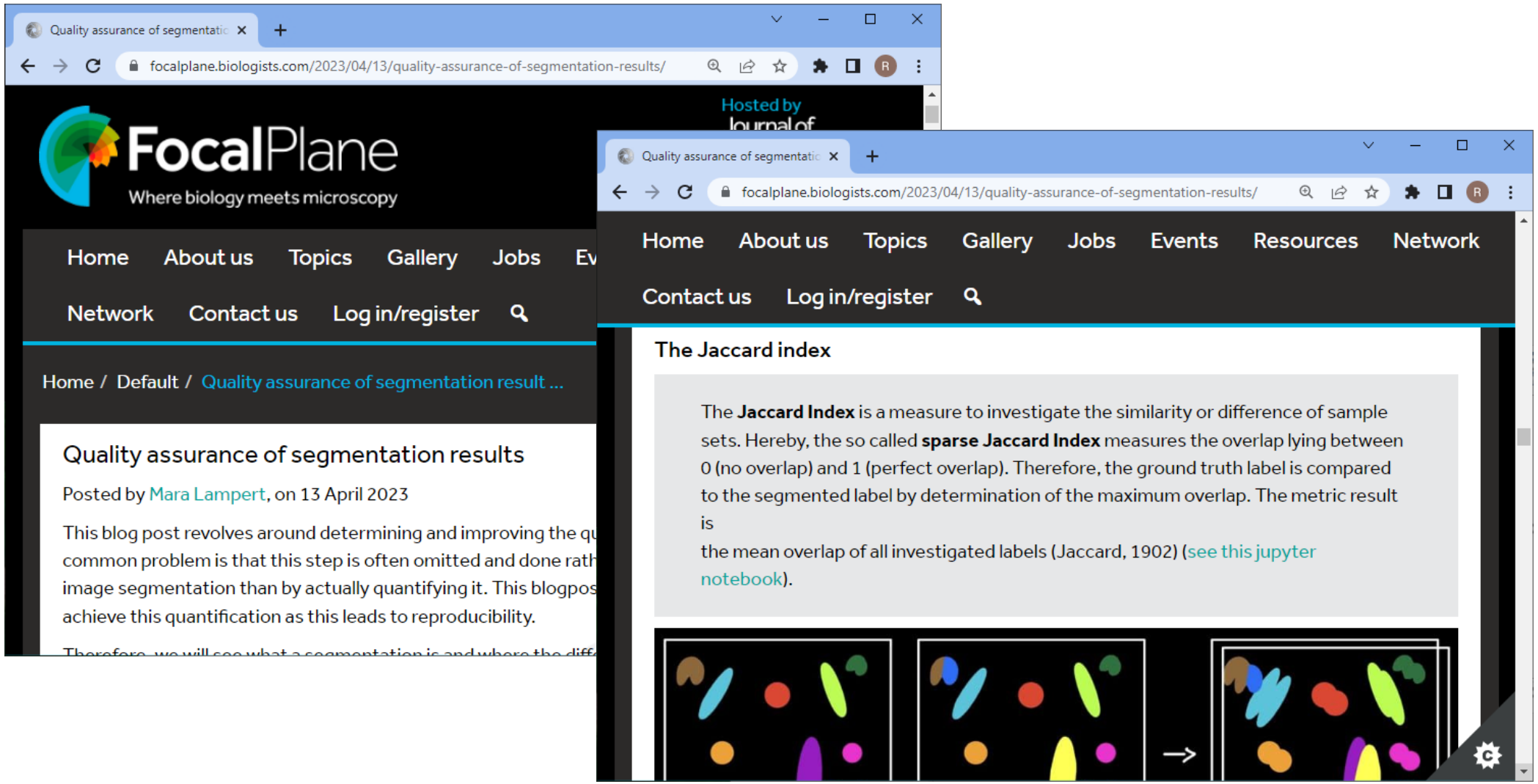
Special case in medical imaging

- 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			**Problem category:** Semantic segmentation **Overlap-based metric (S6):** Dice Similarity Coefficient (DSC) **Boundary-based metric (S7):** Normalized Surface Distance (NSD) **Specific property-related metric:** Liver segmentation: Absolute Volume Difference
	SemS-2	Liver segmentation in computed tomography (CT) images			

Source: Cropped from Fig 5 in Maier-Hein, Reinke, et al. licensed CC-BY 4.0 https://arxiv.org/pdf/2206.01653.pdf

Further reading



Mara Lampert
@marabuuuu.bsky.social

ScaDS.AI
DRESDEN LEIPZIG
CENTER FOR SCALABLE DATA ANALYTICS
AND ARTIFICIAL INTELLIGENCE

Exercises

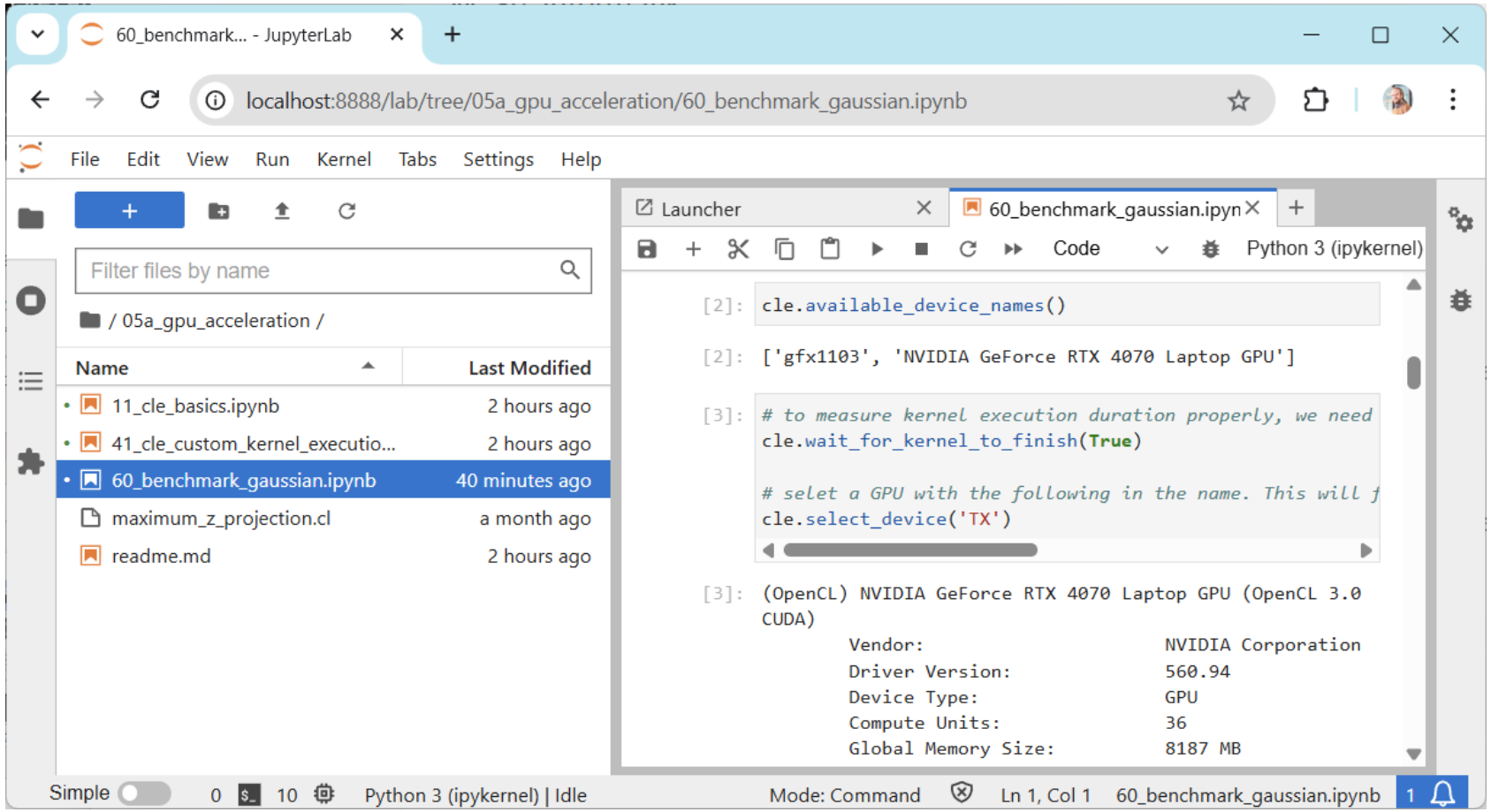
Robert Haase

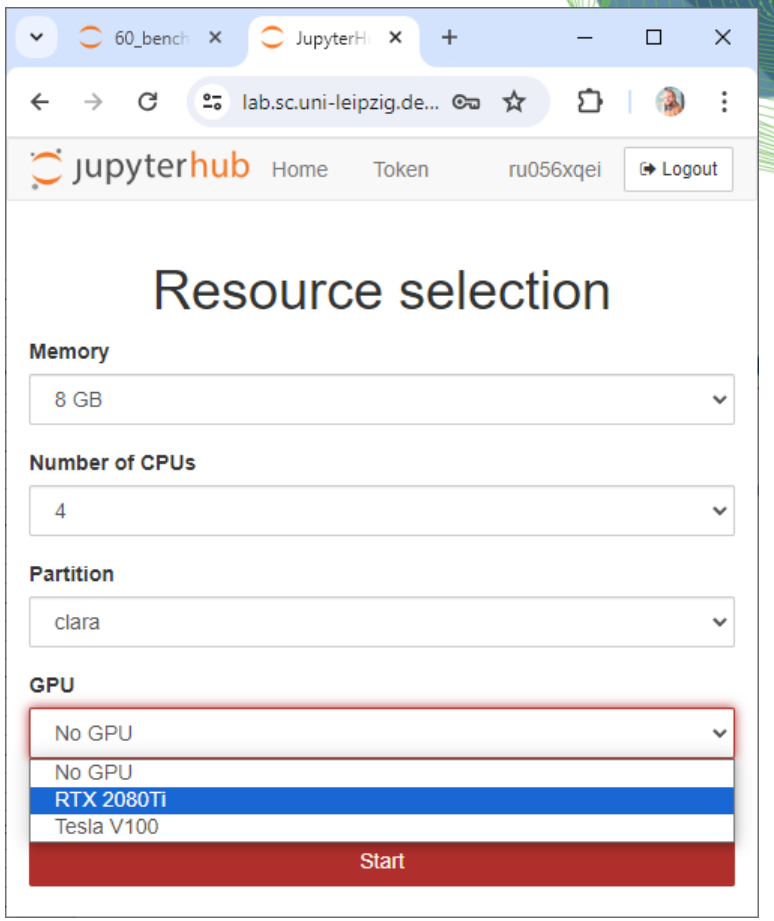
GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

SACHSEN
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.

Exercise: GPU-accelerated image processing

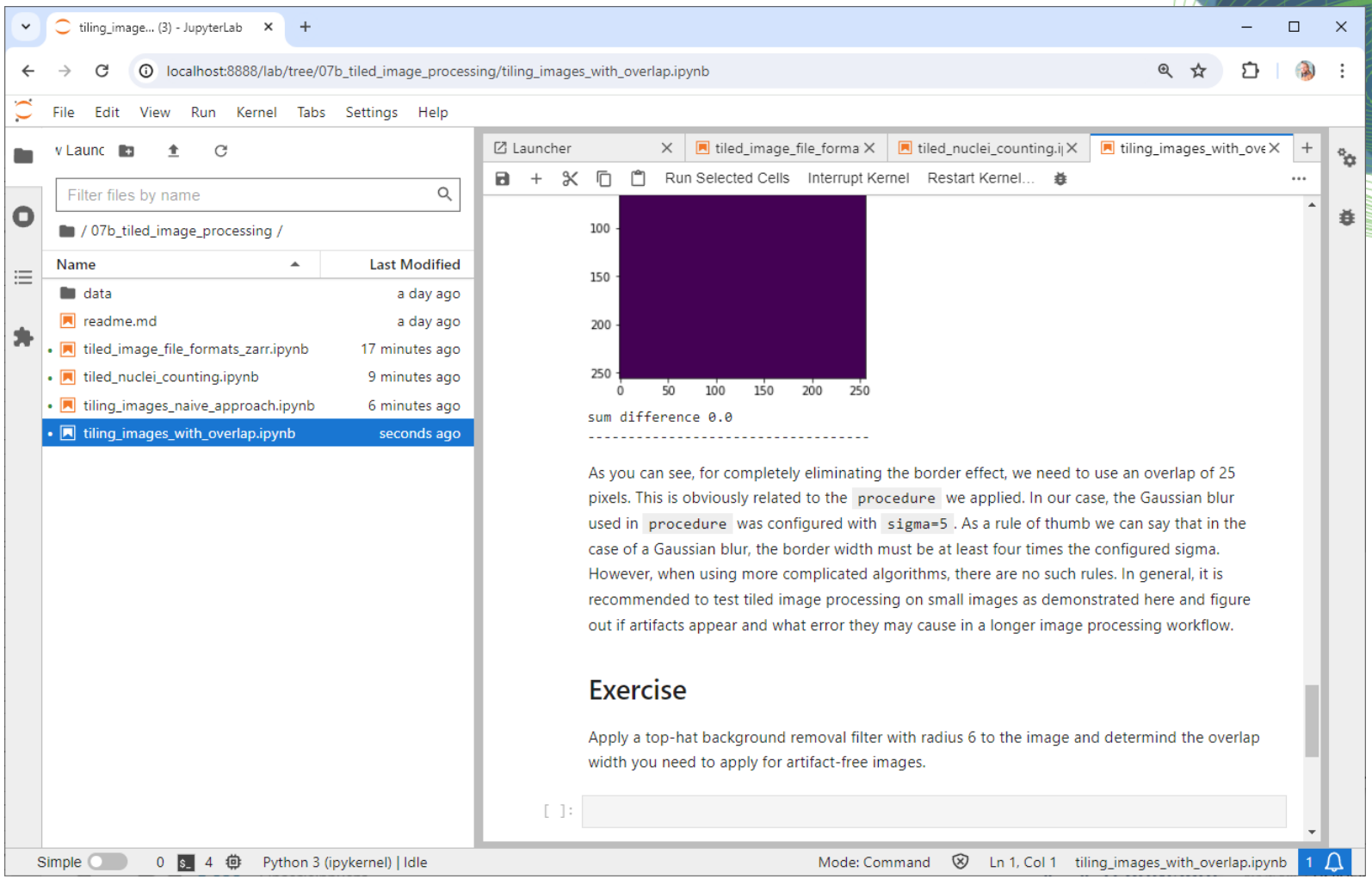
- Compare CPU processing speed with a GPU, consider using the cluster for this





Exercise: Tiled image processing

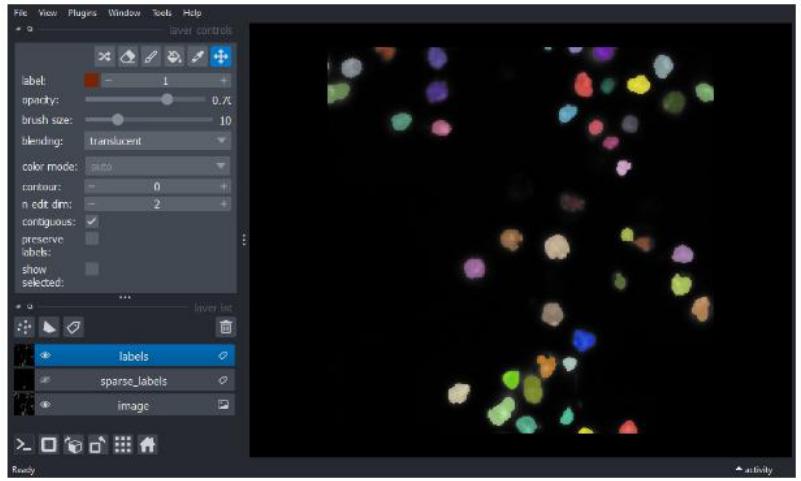
- Apply background-removal to an image in tiles. Determine the overlap width that‘s necessary to have artifact-free results.



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 segmenation / 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

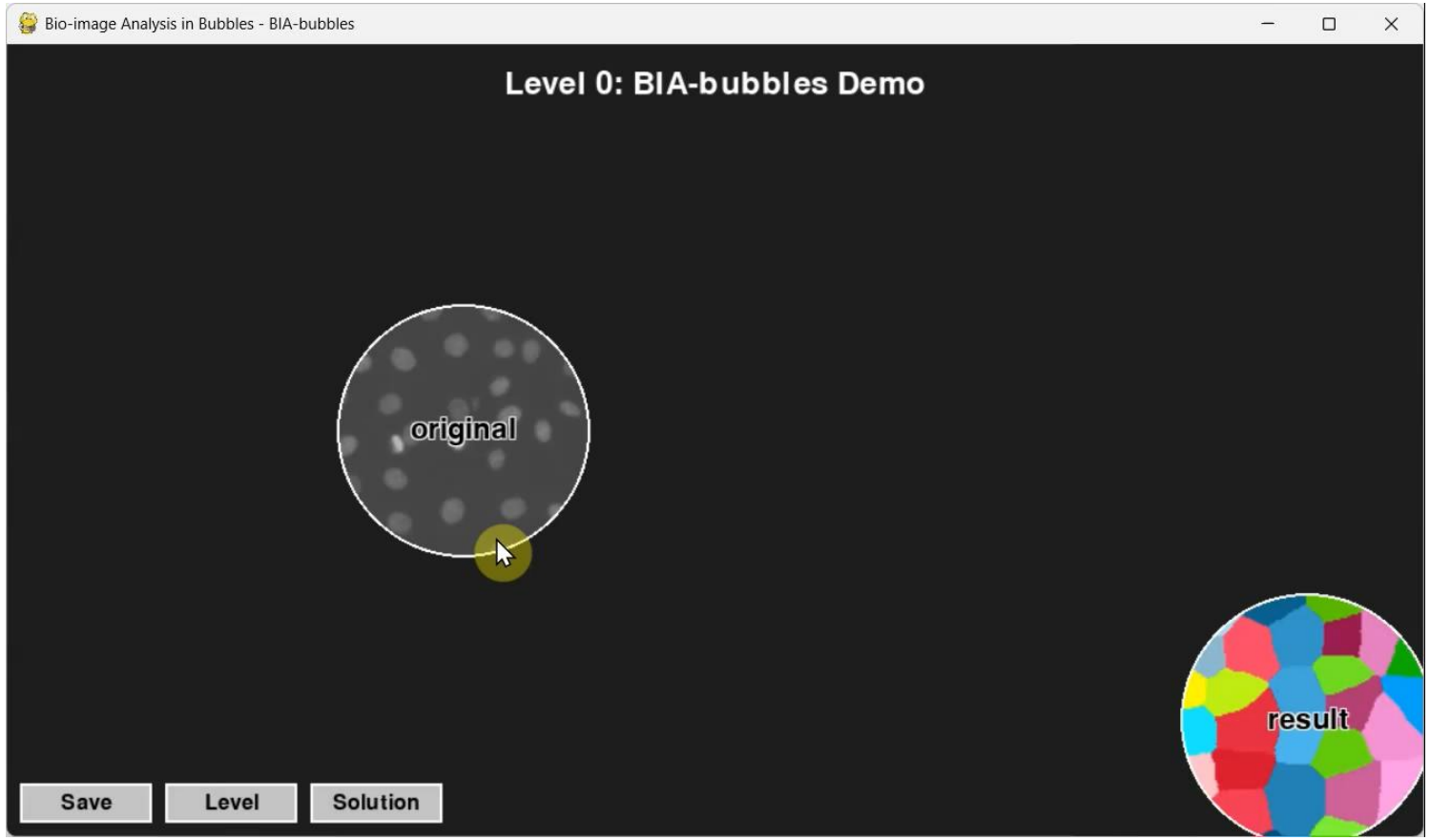
Use the following for-loop and code snippets from above to compute the segmentation quality of all images in the folder. Provide the average quality over all images.

```
[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/