Bio-image Analysis using cupy

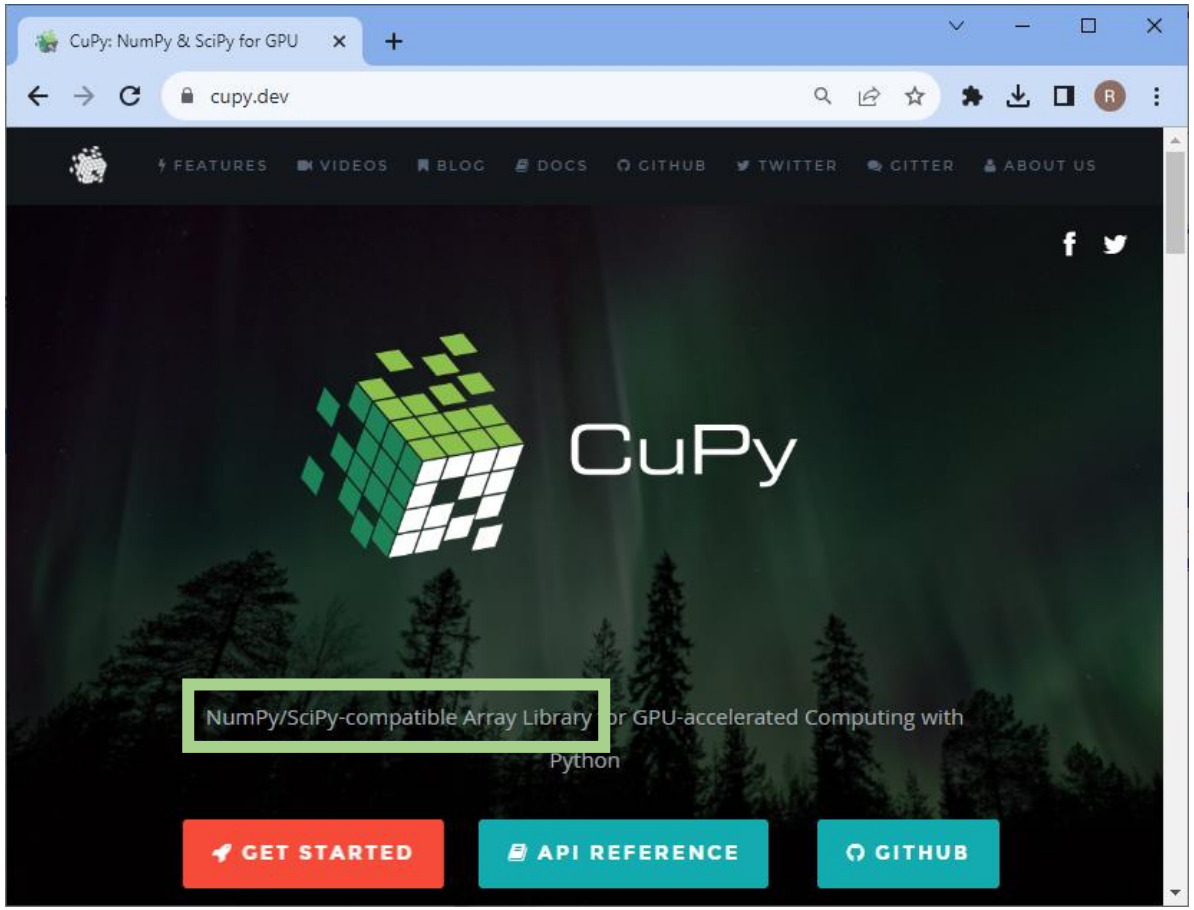
Robert Haase

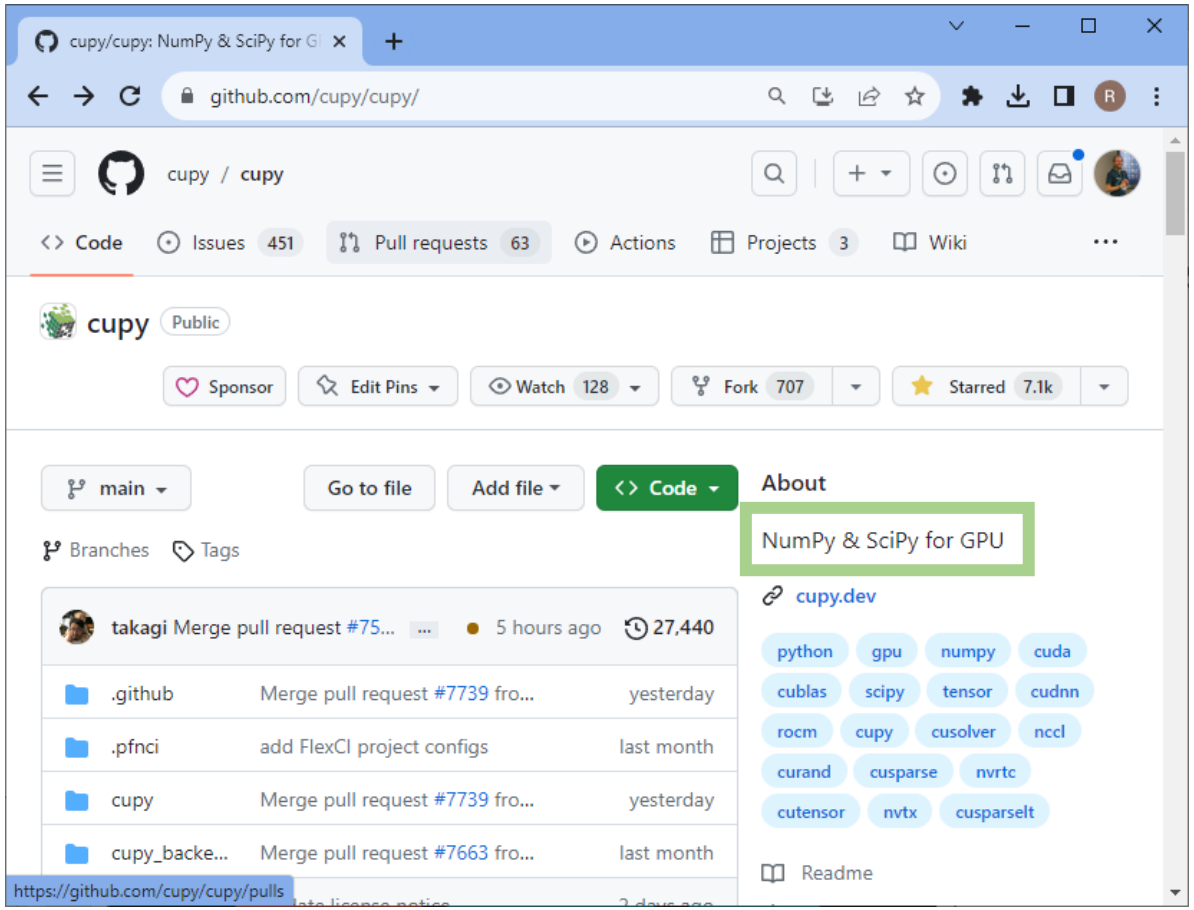
August 2023

@haesleinhuepf

cupy

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





drop-in replacement for numpy and scipy

PoL Physics of Life TU Dresden

- 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
```

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

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

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

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

```
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:

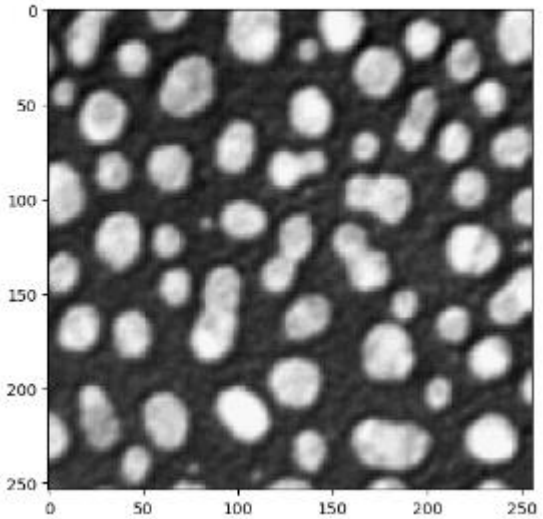
```
[3]: image = imread("../../data/blobs.tif")

     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>
```

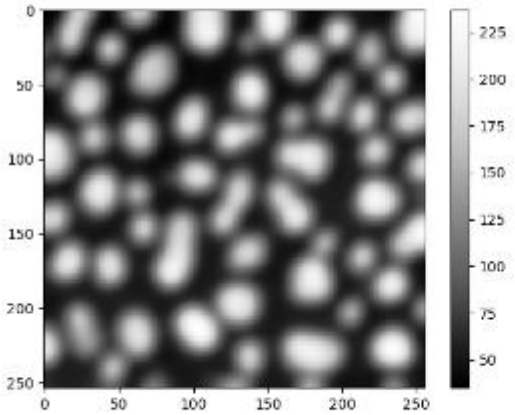


- *Our* libraries aim to be cupy/numpy agnostic

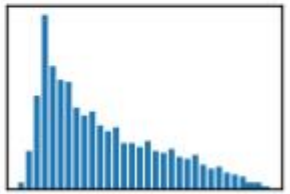
```
[4]: xp_blurred = xdi.gaussian_filter(xp_image, sigma=5)
```

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

[5]:



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



Custom kernels

- CUDA is also just C. You can write custom cupy kernels using simple syntax.
- T represents the image type

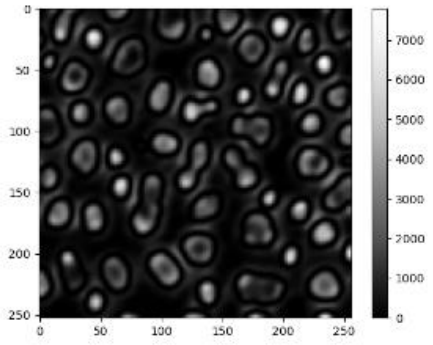
```
squared_difference = cp.ElementwiseKernel(
    'T x, T y',
    'T z',
    'z = (x - y) * (x - y)',
    'squared_difference')
```

Input, output parameters

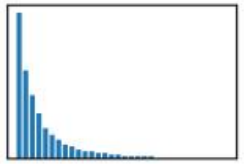
Math / CUDA code

Function call using cupy-arrays as parameters

```
[3]: cp_image = cp.asarray(image[1:], dtype=np.float32)
     image1 = gaussian_filter(cp_image, sigma=3)
     image2 = gaussian_filter(cp_image, sigma=7)

[5]: sqdiff = squared_difference(image1, image2)

[6]: stackview.insight(sqdiff)

[6]:
```



shape	(253, 256)
dtype	float32
size	253.0 kB
min	3.0791853e-08
max	7801.2026

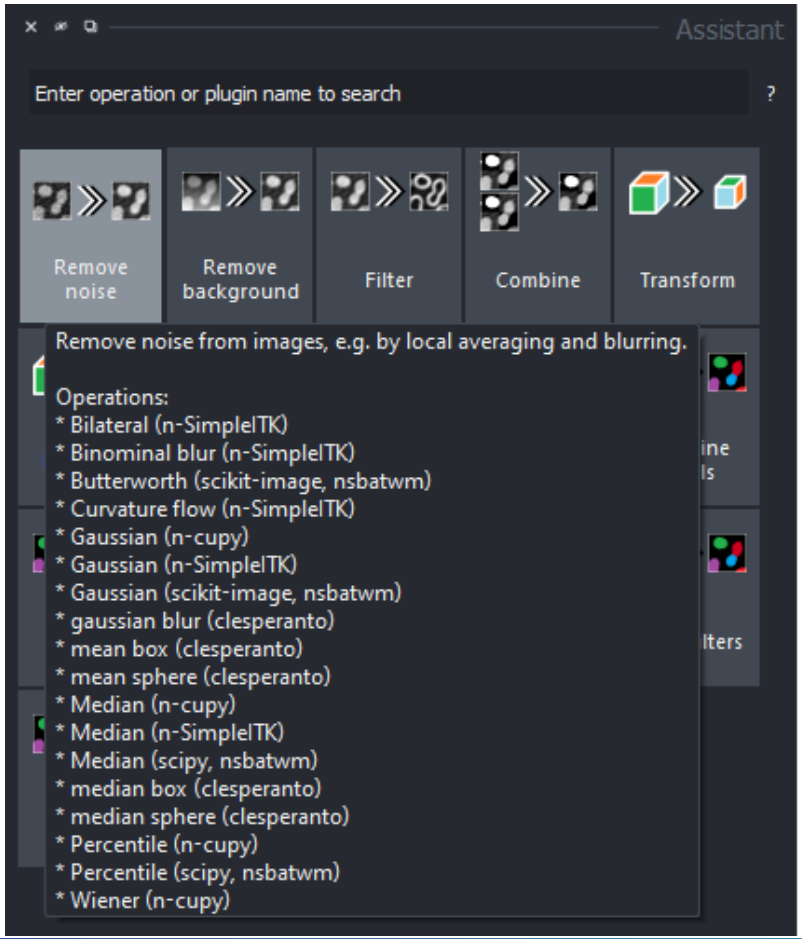


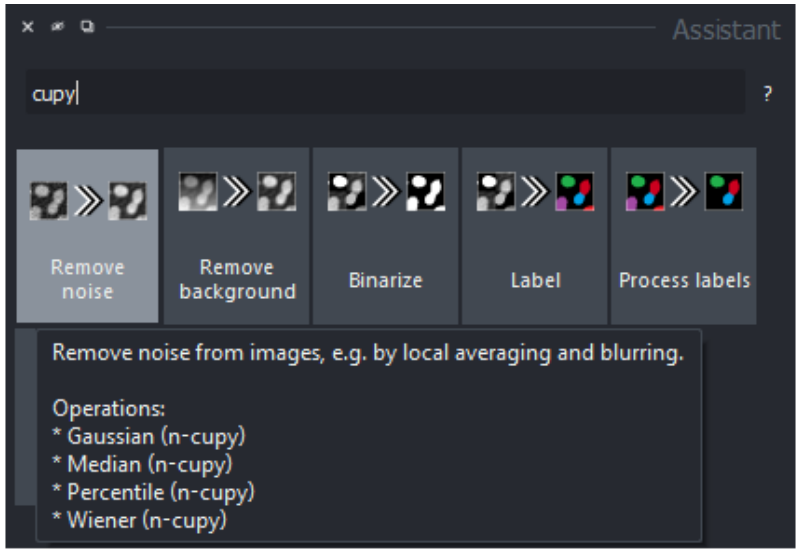
Read more: https://docs.cupy.dev/en/stable/user_guide/kernel.html

Convenience for Napari users

PoL Physics of Life TU Dresden

- To design cupy-based workflows using the Napari-assistant, enter “cupy” in the search field





Convenience for Napari users

PoL Physics of Life TU Dresden

- Using the plugin from Python works as with other napari-assistant compatible plugins
- It aims avoiding memory transfer, if used the right way.

```
import napari_cupy_image_processing as ncupy
```

```
[2]: image = imread("../../data/blobs.tif")
```

If you pass a numpy array as input, …

```
[3]: blurred = ncupy.gaussian_filter(image, sigma=5)

     isinstance(blurred, np.ndarray)

[3]: True
```

the output will by numpy as well

```
[5]: cp_image = cp.asarray(image)

     isinstance(cp_image, np.ndarray)

[5]: False
```

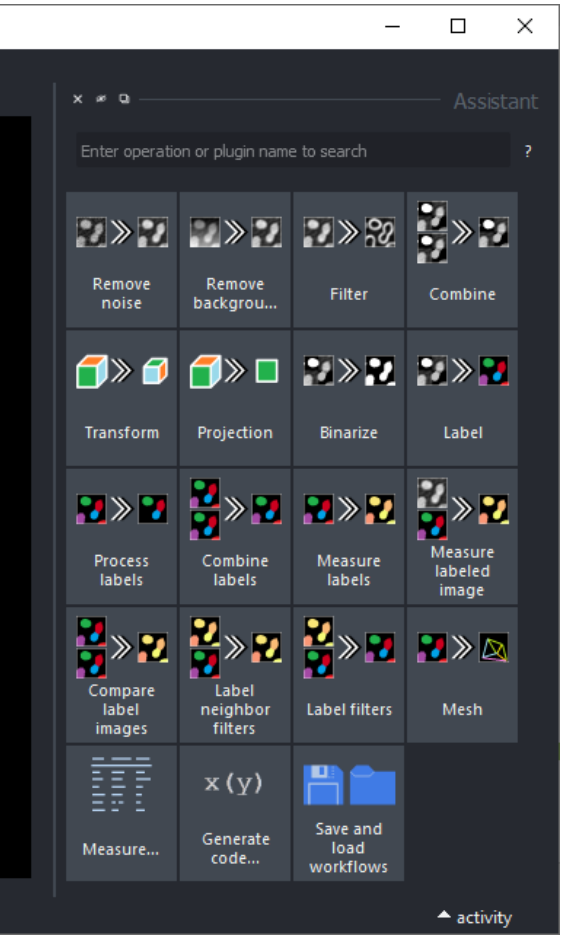
If you pass a cupy-array, …

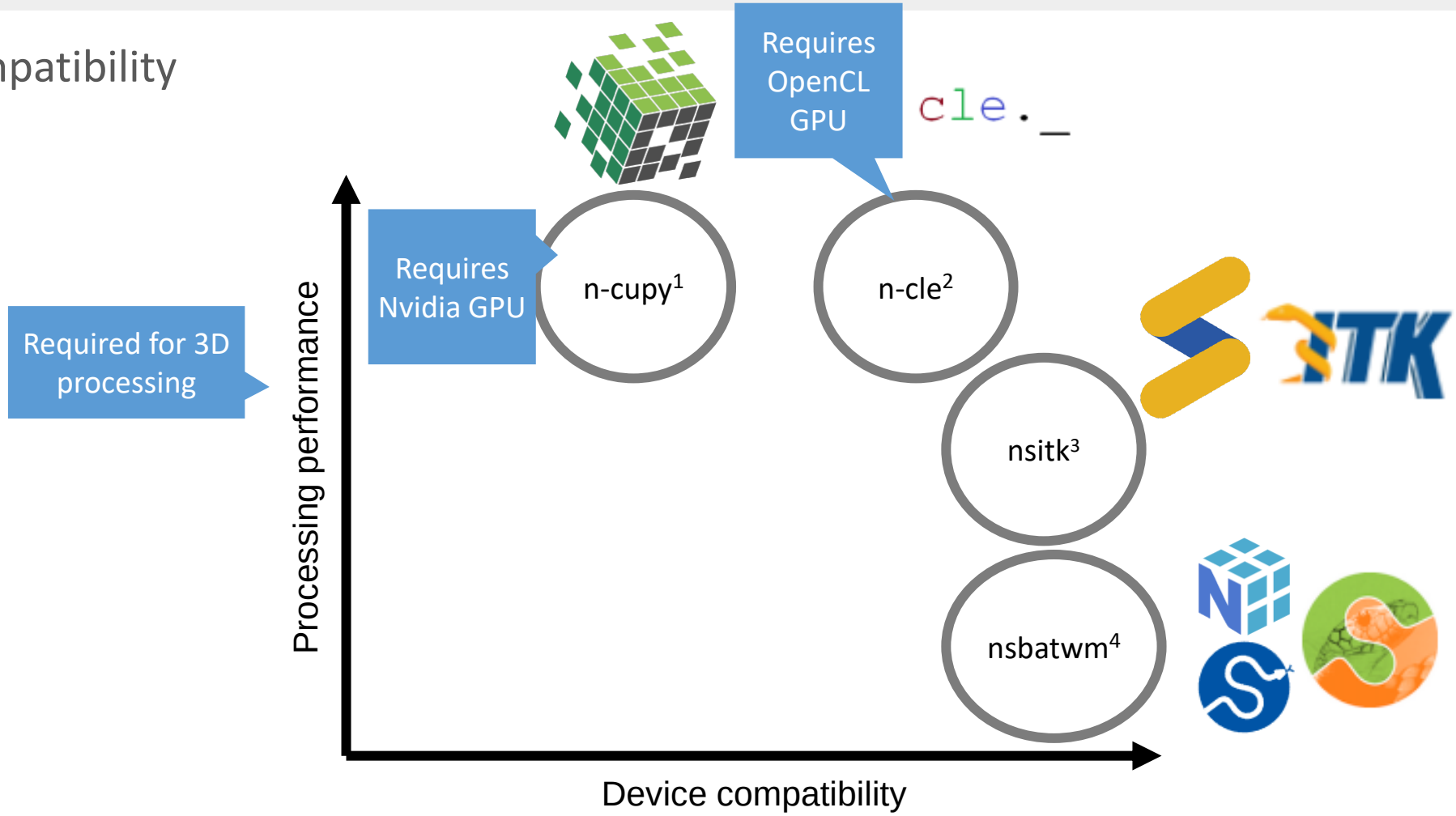
```
[6]: cp_blurred = ncupy.gaussian_filter(cp_image, sigma=5)

     isinstance(cp_image, np.ndarray)

[6]: False

[7]: type(cp_blurred)

[7]: cupy.ndarray
```

the output will by a cupy-array, too

Napari-Assistant compatible plugins

- Performance versus compatibility

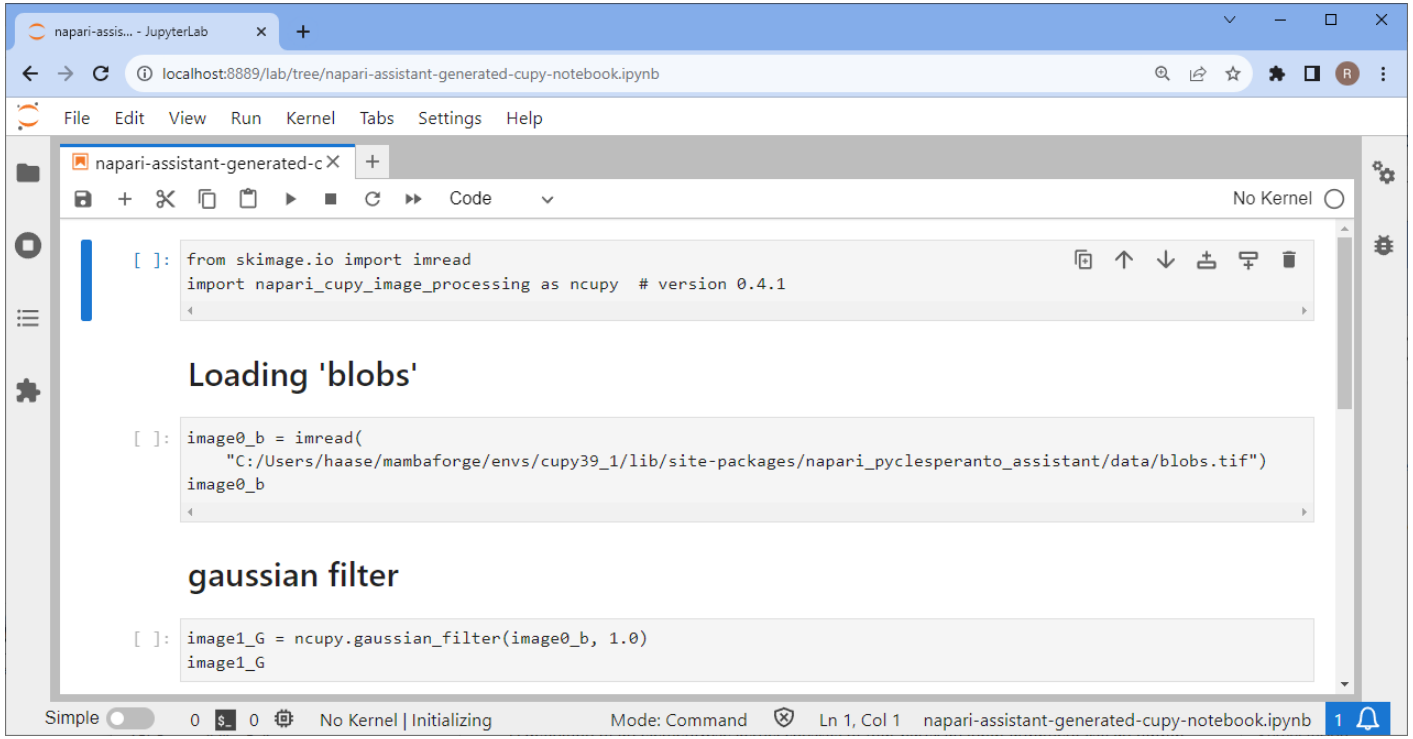




1 https://www.napari-hub.org/plugins/napari-cupy-image-processing
2 https://www.napari-hub.org/plugins/napari-pyclesperanto-assistant
3 https://www.napari-hub.org/plugins/napari-simpleitk-image-processing
4 https://www.napari-hub.org/plugins/napari-segment-blobs-and-things-with-membranes

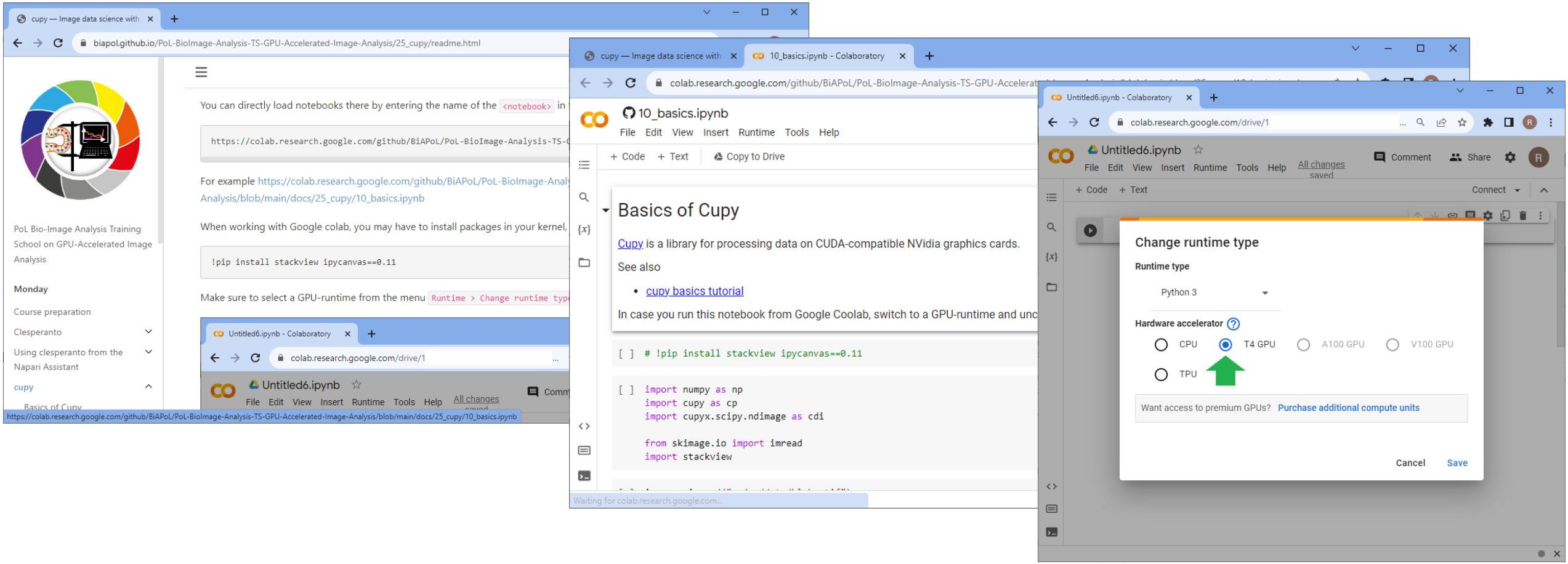
Exercises

- Design a workflow for segmenting blobs in napari using cupy operations only.
- Export a notebook.
- If cupy doesn't work on your laptop, there is a generated notebook available in the repository.



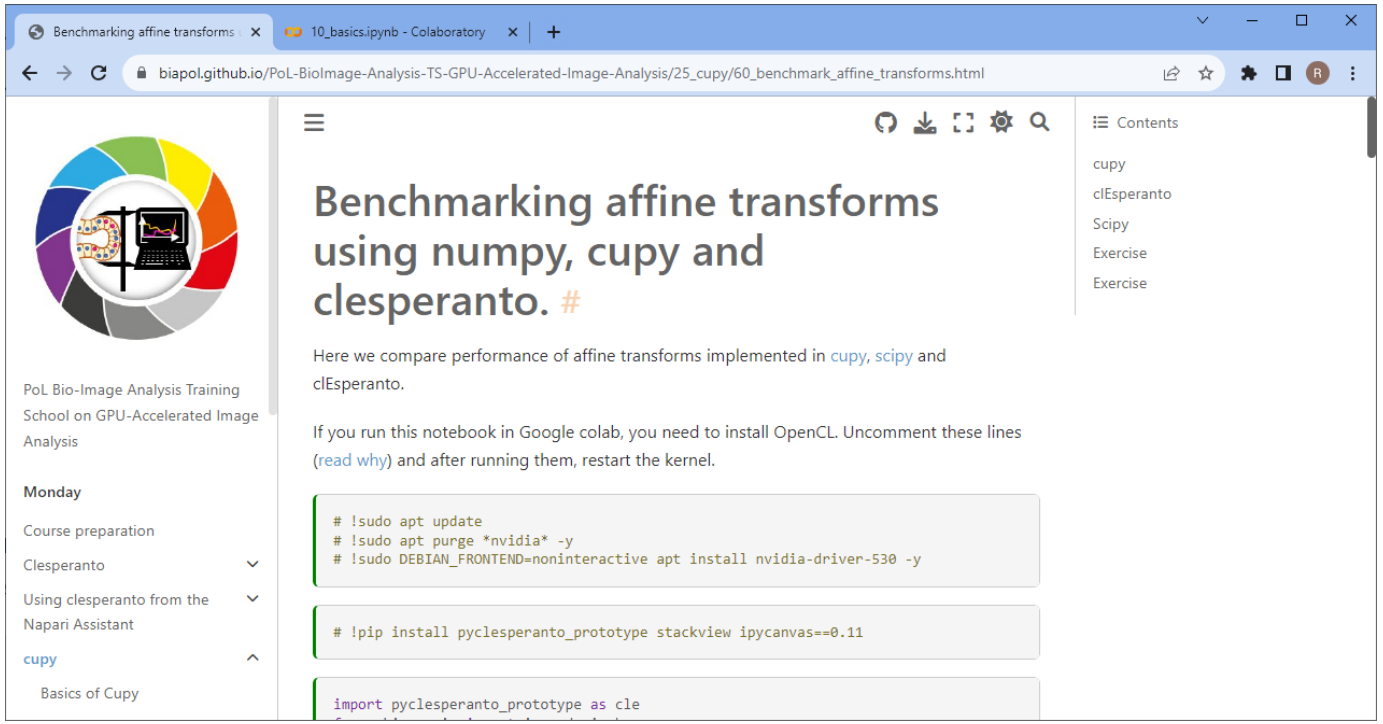
Exercises

- Run cupy-notebooks in google colab.



Exercises

- Benchmark cupy, clesperanto and numpy in comparison.
- How large do images have to be so that it makes sense to use a GPU?
- How different do operations behave differently?



https://biapol.github.io/PoL-BioImage-Analysis-TS-GPU-Accelerated-Image-Analysis/25_cupy/60_benchmark_affine_transforms.html

Acknowledgements

PoL Physics of Life TU Dresden

BiAPoL team

- Mara Lampert
- Marcelo Zoccoler
- Johannes Soltwedel
- Maleeha Hassan
- Allyson Ryan
- Till Korten
- Stefan Hahmann
- Somashekhar Kulkarni

Former lab members:

- Ryan George Savill
- Laura Zigutyte

Networks

BioDIP Biopolis Dresden Imaging Platform

CENTER FOR SYSTEMS BIOLOGY DRESDEN

NFDI4 BIOIMAGE

neubias network of european bioimage analysts

image.sc

German BioImaging Gesellschaft für Mikroskopie und Bildanalyse

Funding

DFG public

Federal Ministry of Education and Research

Chan Zuckerberg Initiative