PoL Physics of Life TU Dresden

CENTER FOR SYSTEMS BIOLOGY DRESDEN

CBG Max Planck Institute of Molecular Cell Biology and Genetics

CLIJ: GPU-accelerated image processing for everyone

Robert Haase

CASUS, Feb 4th 2021

@haesleinhuepf
@PoLDresden

TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

Acknowledgements

Akanksha Jain (Treutlein lab) @jain_akanksha_

Dani Vorkel (Myers lab) @happifocus

Theresa Suckert (Krause lab, Oncoray)

Pradeep Rajasekhar (Poole lab, Monash U) @pr4deepr

Talley J. Lambert (HMS) @TalleyJLambert

Juan Nunez-Iglesias (Monash) @jnuneziglesias

Pavel Tomancak (CSBD / MPI-CBG) @PavelTomancak

Gene Myers (CSBD / MPI-CBG) @TheGeneMyers

Stéphane Rigaud (Institut Pasteur) @StRigaud

Jean-Yves Tinevez (Institut Pasteur) @jytinevez

Florian Jug (CSBD / MPI-CBG) @florianjug

Szabolcs Horvát (Modes lab) @szhorvat

Johannes Girstmair (Tomancak lab) @jogirstmair

Brian Northon @truenorth_ia

Matthias Arzt (Myers/Jug lab)

https://fiji.sc

https://napari.dev

https://image.sc

http://eubias.org/NEUBIAS/

Alex Herbert (University of Sussex)
Alexandr Dibrov (MPI CBG)
Bertrand Vernay (IGBMC, Strasbourg)
Bert Nitzsche (PoL TU Dresden)
Bradley Lowekamp (NIAID Washington)
Bram van den Broek (Netherlands Cancer Institute)
Brenton Cavanagh (RCSI)
Bruno C. Vellutini (MPI CBG)
Carl D. Modes (CSBD/MPI-CBG)
Curtis Rueden (UW-Madison LOCI)
Damir Krunic (DKFZ)
Daniel J. White (GE)
Deborah Schmidt (CSBD/MPI CBG)
Douglas P. Shepard (Univ Arizona)
Eduardo Conde-Sousa (Univ. of Porto)
Erick Ratamero (The Jackson Laboratory)
Gaby G. Martins (IGC)
Gayathri Nadar (MPI CBG)
Guillaume Witz (Bern University)
Giovanni Cardone (MPI Biochem)
Irene Seijo Barandiaran (MPI CBG Dresden)
Jan Brocher (Biovoxxel)
Juergen Gluch (Fraunhofer IKTS)
Kisha Sivanathan (Harvard Medical School)
Kota Miura
Laurent Thomas (Acquifer)
Lior Pytowski (University of Oxford)
Loic A. Royer (CZ Biohub)
Marion Louveaux (Institut Pasteur Paris)
Martin Weigert (EPFL Lausanne)
Matthew Foley (University of Sydney)
Nico Stuurman (UCSF)
Nicola Maghelli (MPI CBG)
Nicolas Brouilly (IBDM Marseille)
Nik Cordes (Los Alamos National Laboratory)
Noreen Walker (MPI CBG Dresden)
Ofra Golani (Weizmann Institute of Science)
Patrick Dummer
Peter Haub
Pete Bankhead (University of Edinburgh)
Peter Steinbach (HZDR Dresden)
Pit Kludig
Rita Fernandes (University of Porto)
Romain Guiet (BIOP-EPFL)
Ruth Whelan-Jeans
Sebastian Munck (VIB Leuven)
Siân Culley (MRC LMCB)
Stein Rørvik
Stéphane Dallongeville (Pasteur)
Tanner Fadero (UNC-Chapel Hill)
Thomas Irmer (Zeiss)
Tobias Pietzsch (MPI-CBG)
Uwe Schmidt (MPI CBG)
Wilson Adams (VU Biophotonics)

MPI CBG Core Facilities

- Advanced Imaging Facility
- Light Microscopy Facility
- Scientific Computing
- IT Department
- Fly lab

Funding

DFG public – DFG 2020 FÜR DAS WISSEN ENTSCHEIDEN

Federal Ministry of Education and Research

R.H. was supported by the German Federal Ministry of Research and Education (BMBF) under the code 031L0044 (Sysbio II) and by the Deutsche Forschungsgemeinschaft under Germany's Excellence Strategy – EXC2068 – Cluster of Excellence Physics of Life of TU Dresden.

@haesleinhuepf
@PoLDresden

CENTER FOR SYSTEMS BIOLOGY DRESDEN

CBG Max Planck Institute of Molecular Cell Biology and Genetics

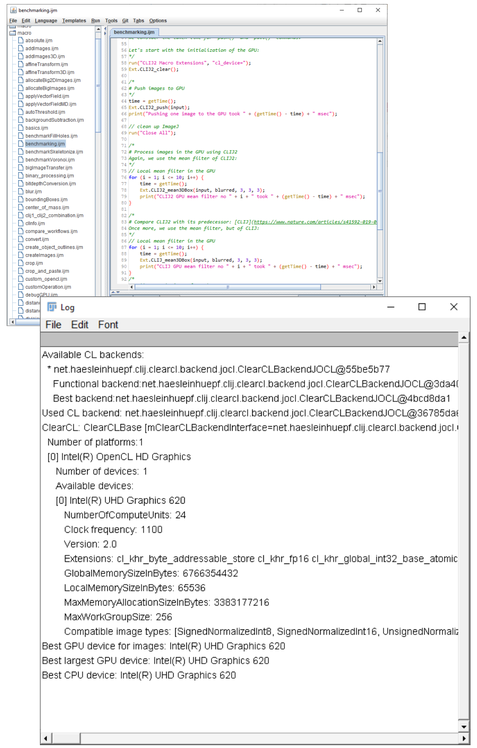
TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

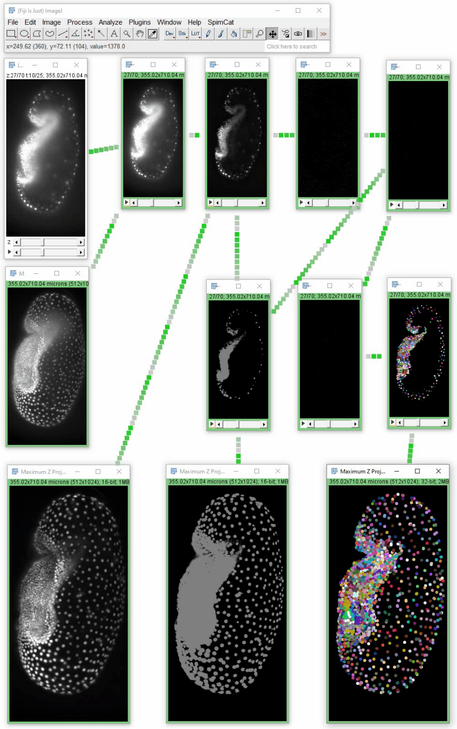
Overview

PoL Physics of Life TU Dresden

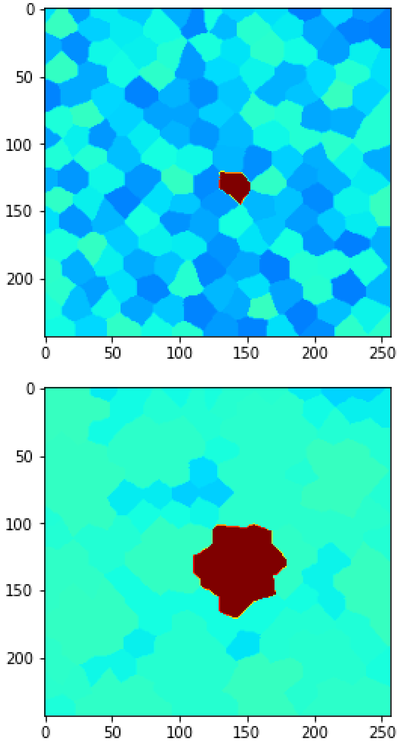
Part I: Basics



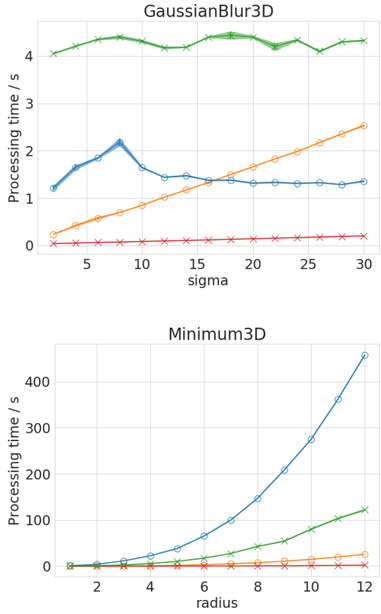
Part II: Interactive workflow design



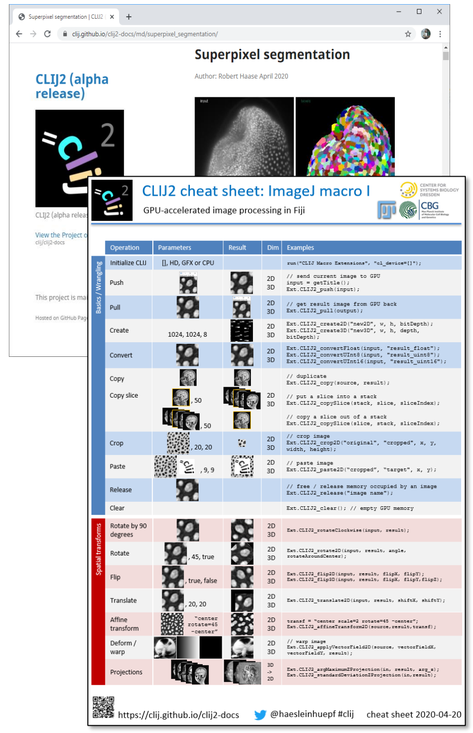
Part III: Life-sciences specific image processing



Part IV: Benchmarking



Part V: Accessibility

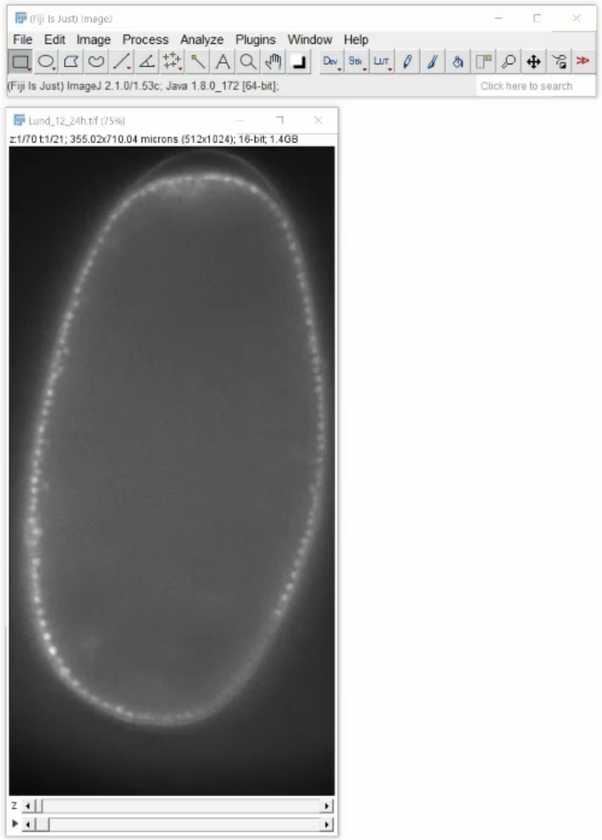


Part I: Basics

Image processing in life-sciences

PoL Physics of Life TU Dresden

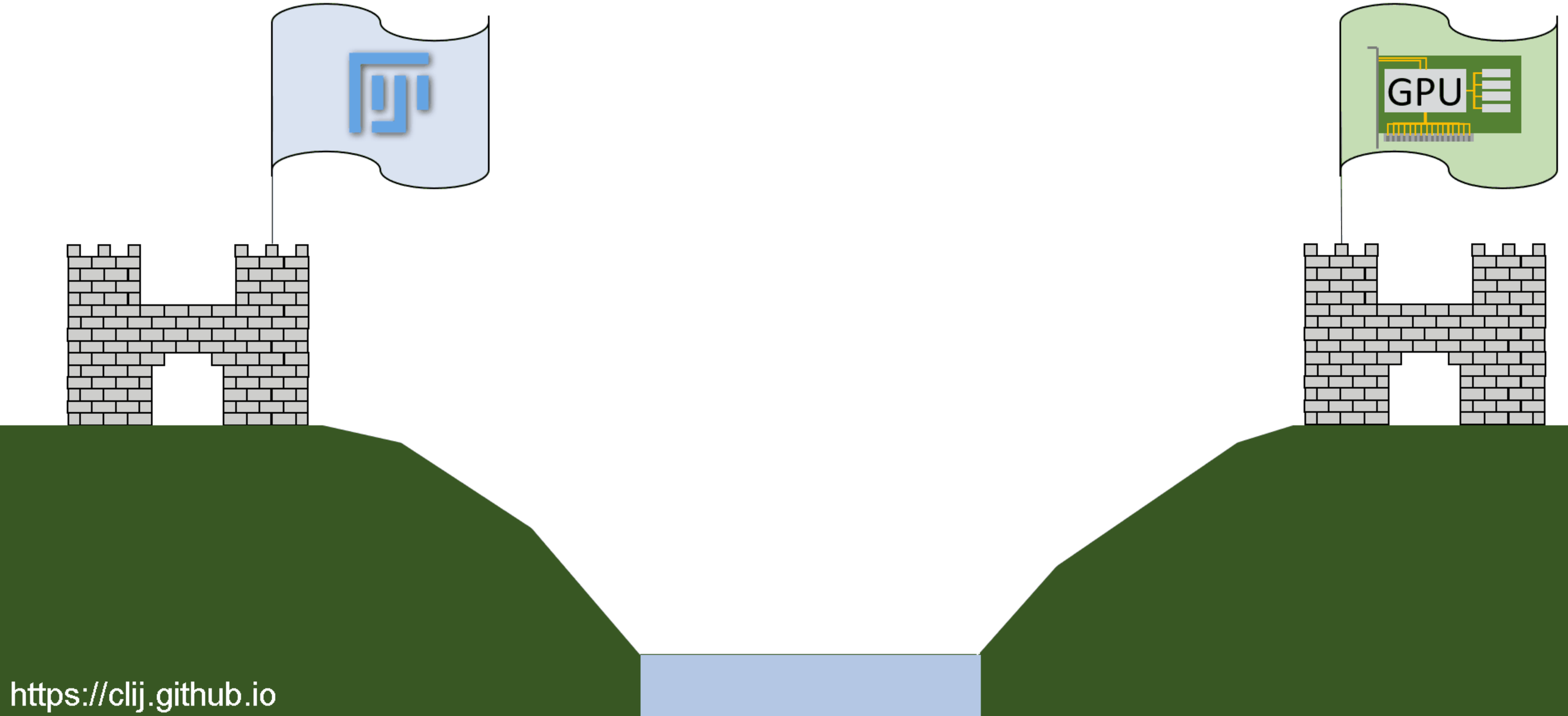
- State-of-the-art software for more than 20 years: ImageJ / Fiji



2x

ImageJ / Fiji integration for OpenCL

PoL
Physics of Life
TU Dresden

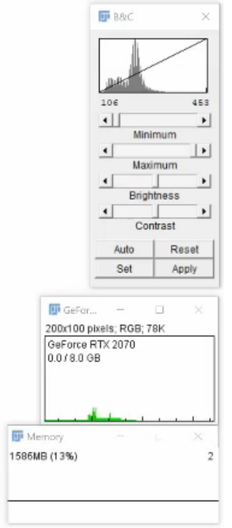


https://clij.github.io

How image processing is supposed to be

(… in my honest opinion)

2x



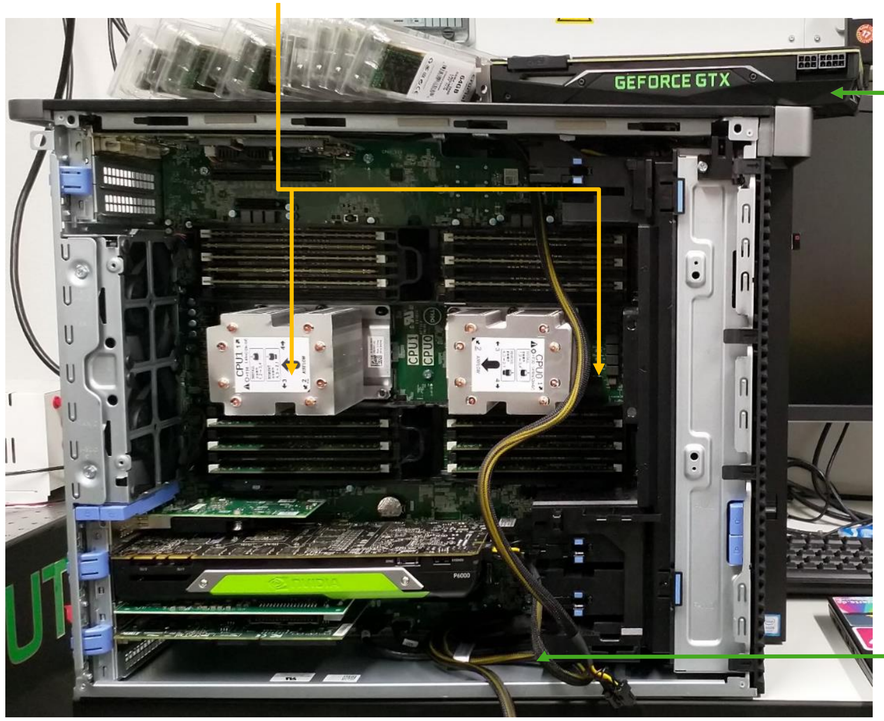
Graphics Processing Units (GPUs)

PoL
Physics of Life
TU Dresden

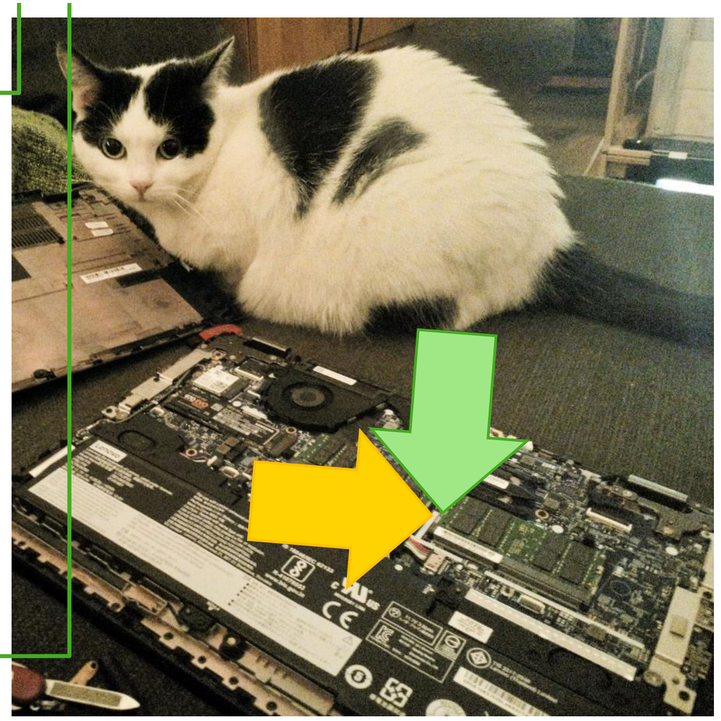
Every computer that has a screen also has a GPU.

Central Processing Unit (CPU)

Graphics Processing Unit (GPU)



dedicated GPUs



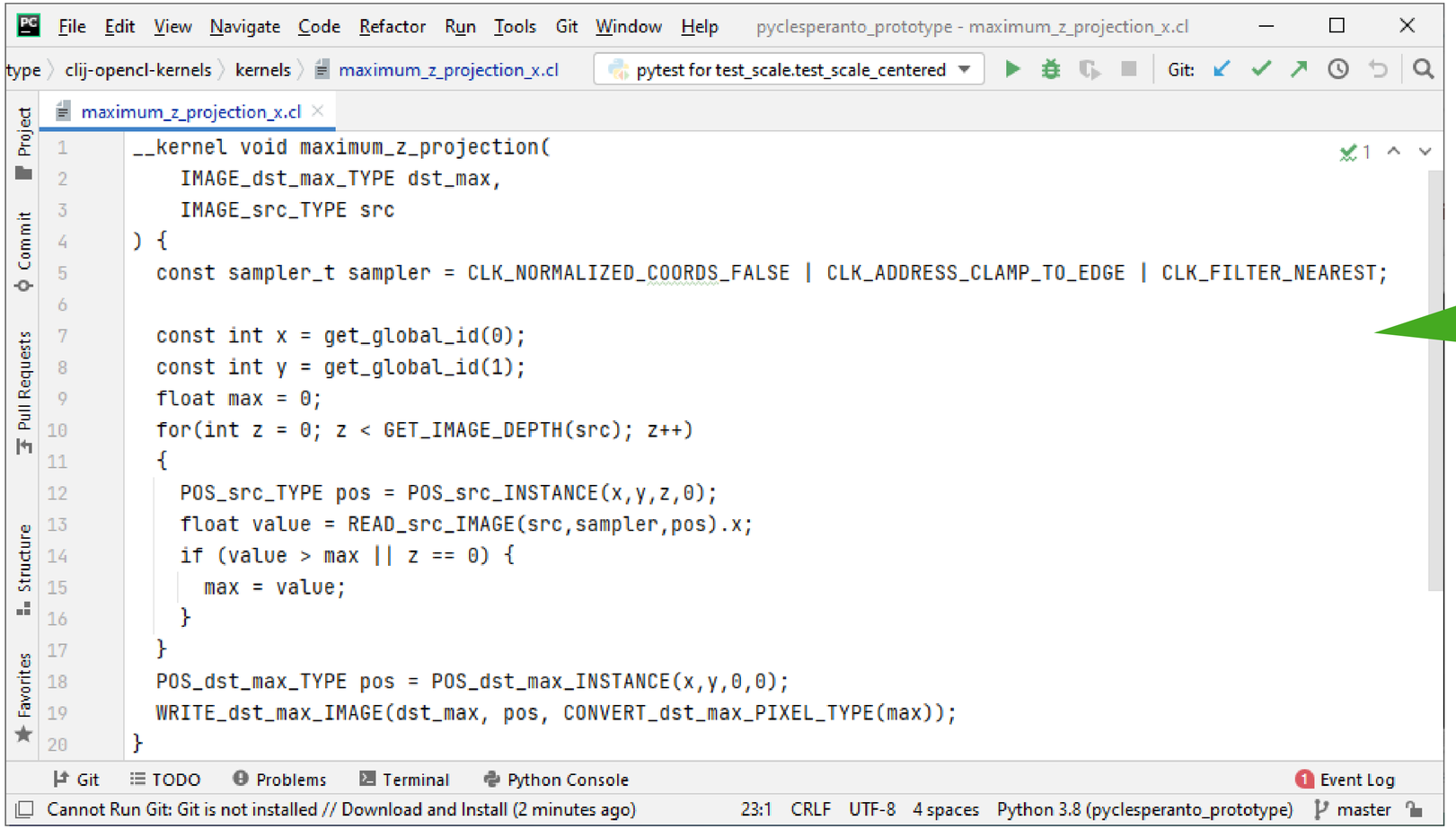
integrated GPUs

external “eGPUs”

OpenCL-based GPU-acceleration

PoL Physics of Life TU Dresden

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



Maximum intensity projection along Z

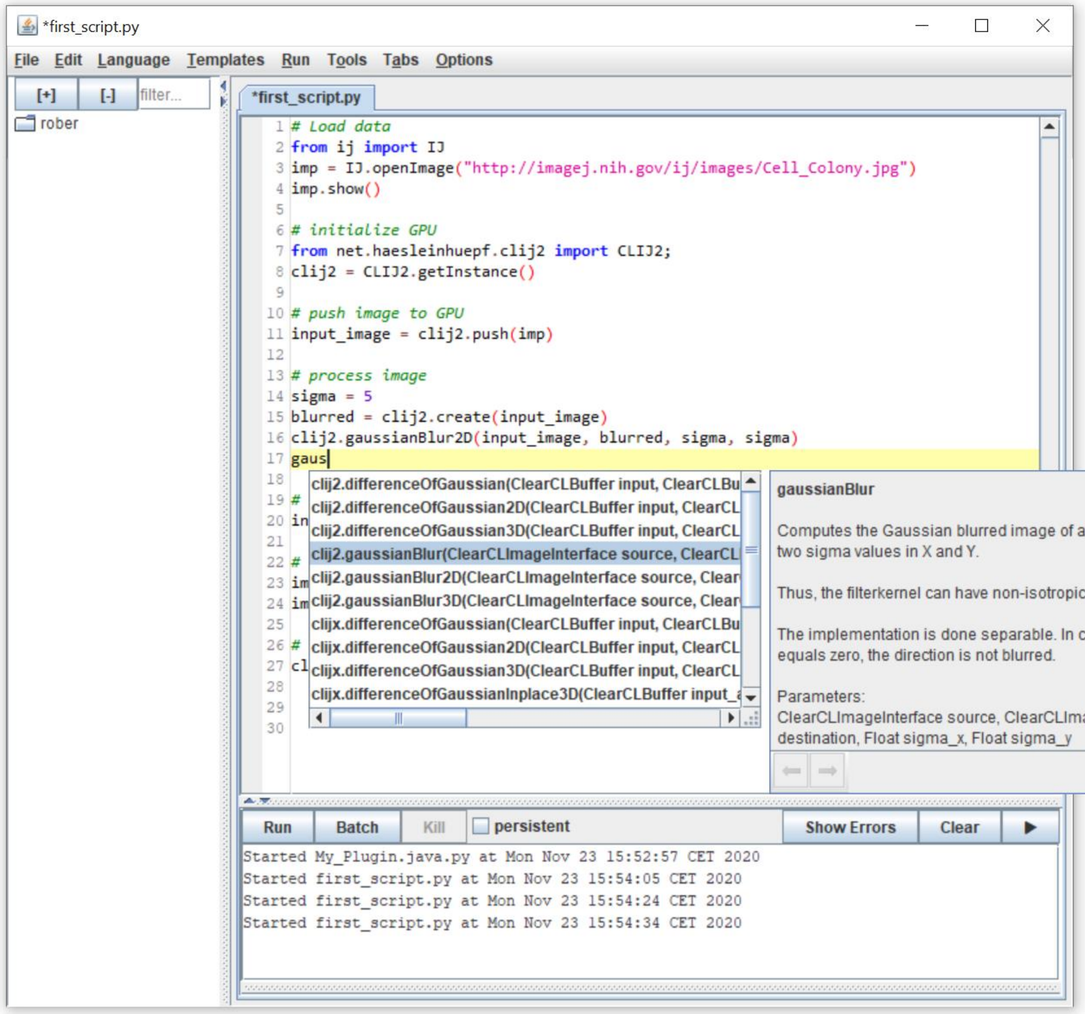
ImageJ / Fiji integration

PoL Physics of Life TU Dresden

- We integrated GPU-acceleration into ImageJ / Fiji using ImageJ Macro “instead” of OpenCL.

“CLIJ” was born.

OpenCL ImageJ



CLIJ2: What every ImageJ Macro script must have

PoL Physics of Life TU Dresden

Load data

Initialize GPU

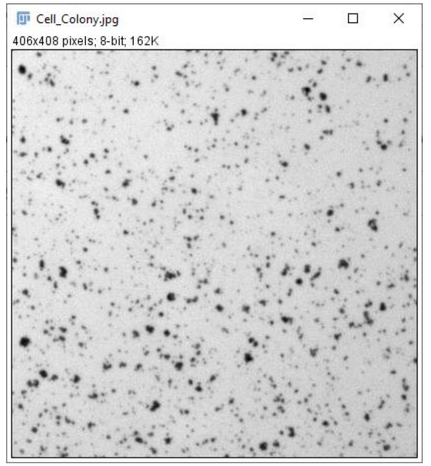
Push

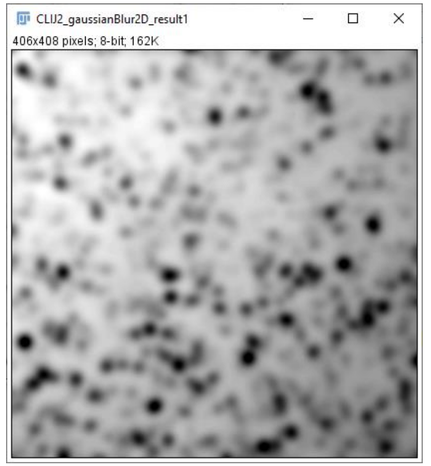
Process images

Pull

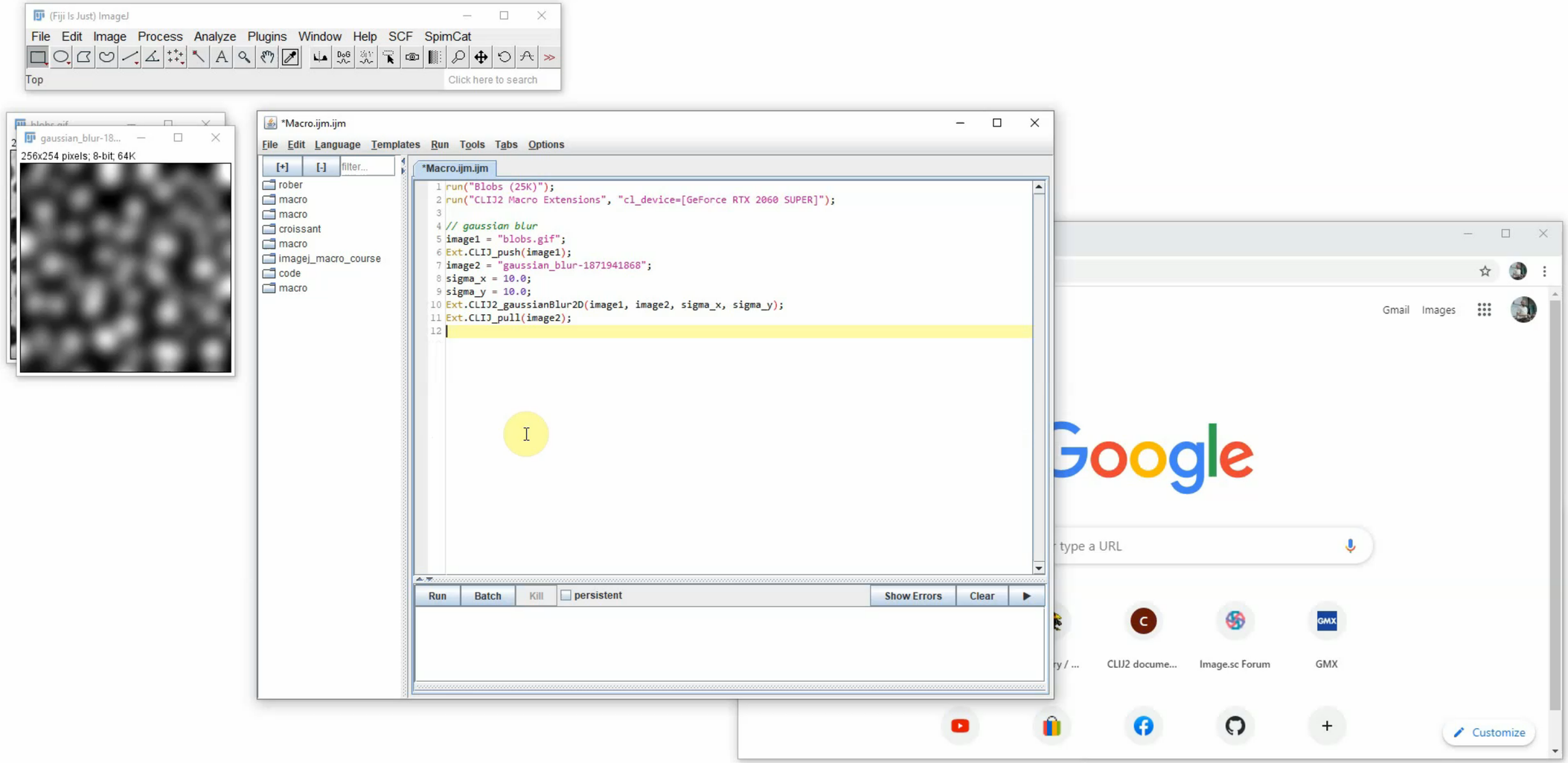
Cleanup

```
// load data
run("Cell Colony (31K)");

// initialize GPU
run("CLIJ2 Macro Extensions", "cl_device=");
Ext.CLIJ2_clear();

// push image to GPU
input_image = getTitle();
Ext.CLIJ2_push(input_image);

// process image
sigma = 5;
Ext.CLIJ2_gaussianBlur2D(input_image, result_image, sigma, sigma);

// optional: release input data
Ext.CLIJ2_release(input_image);

// pull result back from GPU
Ext.CLIJ2_pull(result_image);

// clean up by the end
Ext.CLIJ2_clear();
```





CLIJ2: Macro editing

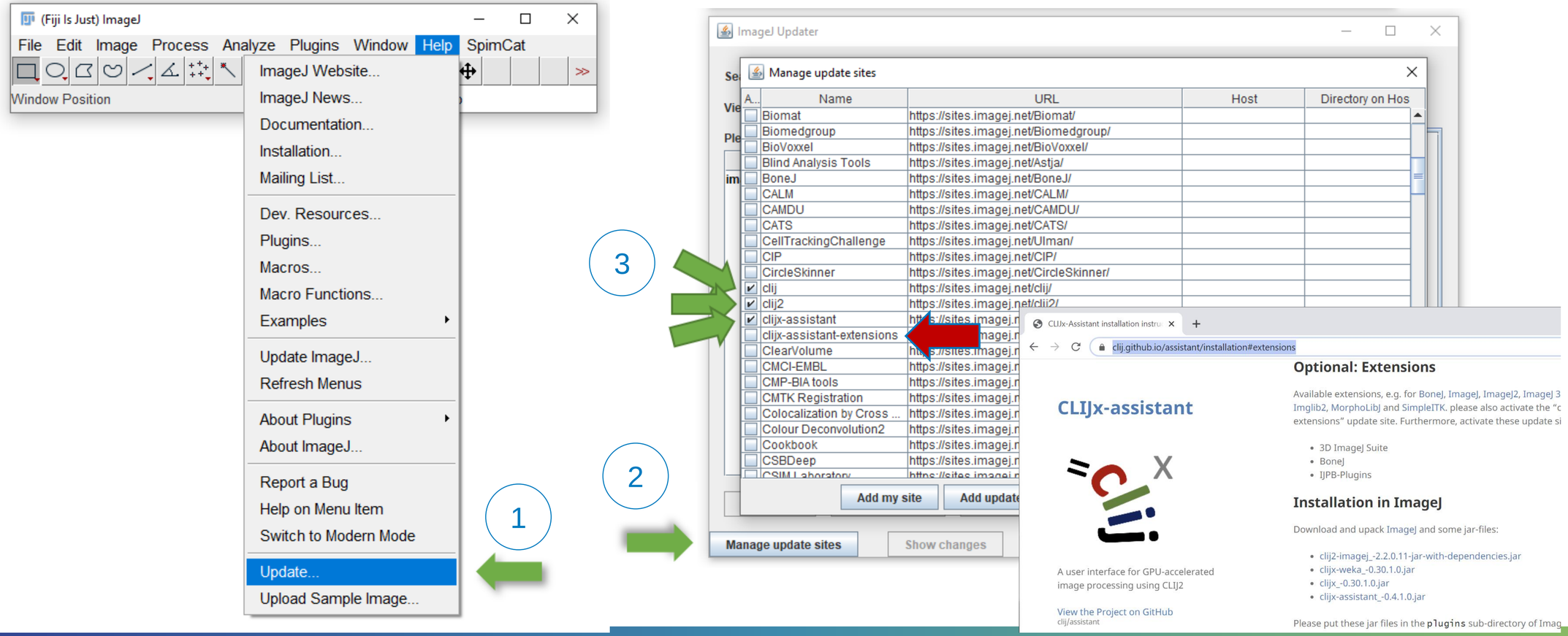


Exercises

Exercise 0: Installation

Just activate the CLIJ update sites, in menu Help > Update...

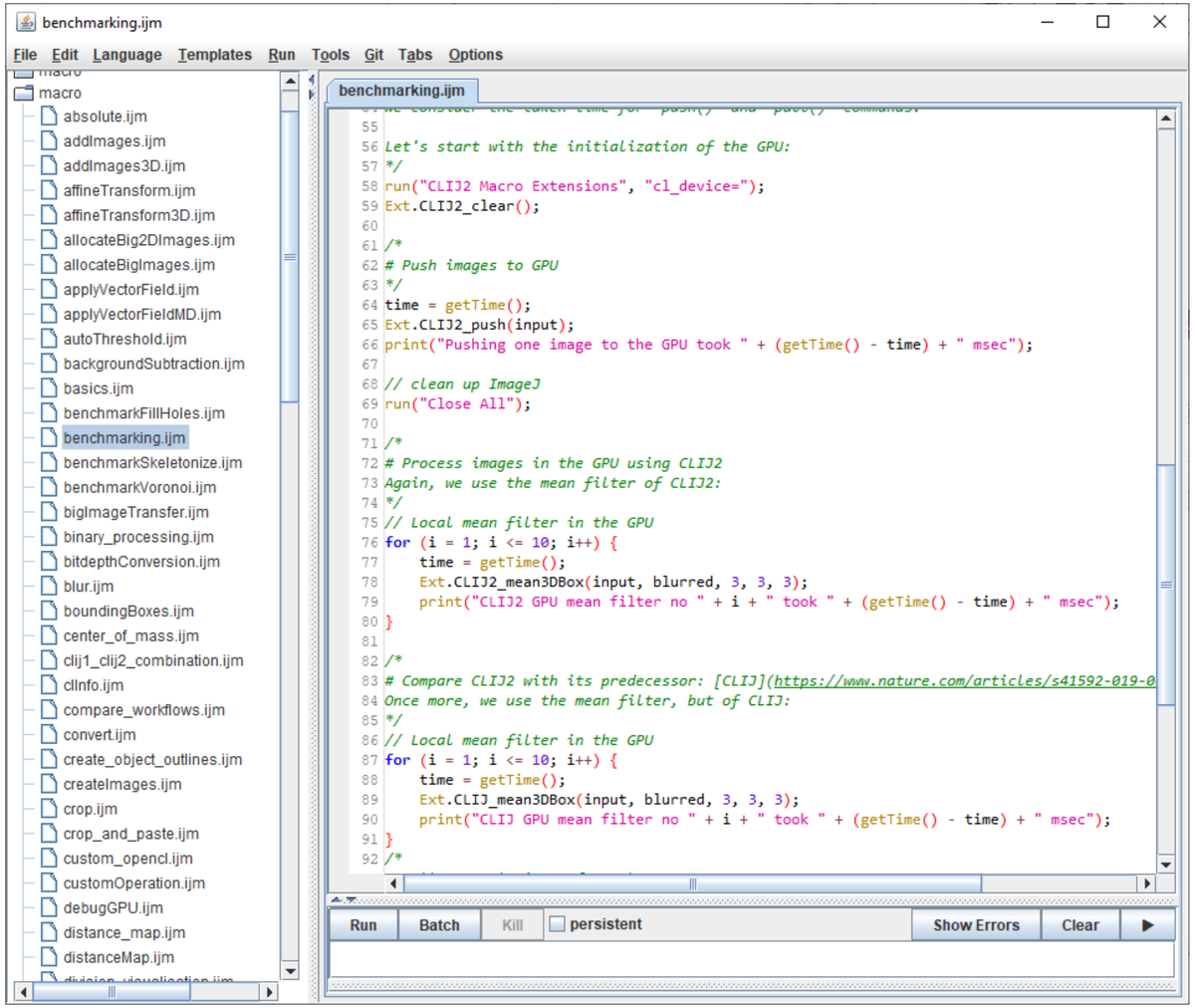
Linux users: you may need to install OpenCL, e.g. `apt-get install ocl-icd-devel`



Exercise 1: Testing if CLIJ2 works

Get an example macro from
http://clij.github.io/clij2-docs/

> Code examples
>> ImageJ Macro
>>> benchmarking.ijm

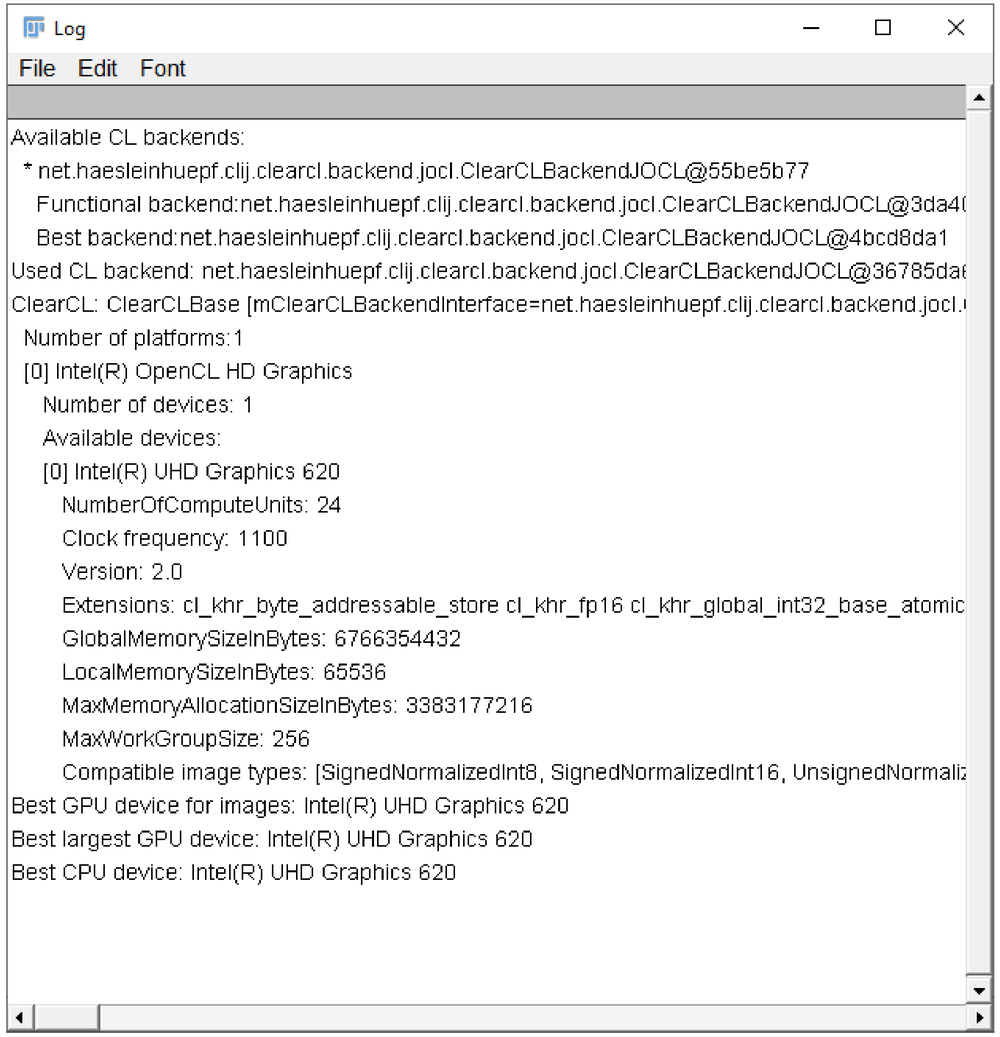


Exercise 2: Study your hardware properties

PoL Physics of Life TU Dresden

Execute this script:
https://github.com/clij/clij2-docs/blob/master/src/main/macro/clInfo.ijm

- What GPU is used per default?
- Home much memory does it have available?
- How large can a single image be when being processed in the GPU?
- Assume your image stack is 1024x1024 pixels large and of type 32-bit float, how many planes can it have so that it still fits in GPU memory?

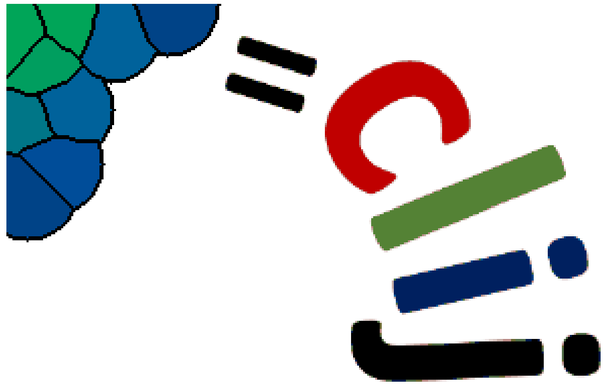


Part II: Interactive workflow design

CLIJ2: More functions, wider applicability

PoL
Physics of Life
TU Dresden

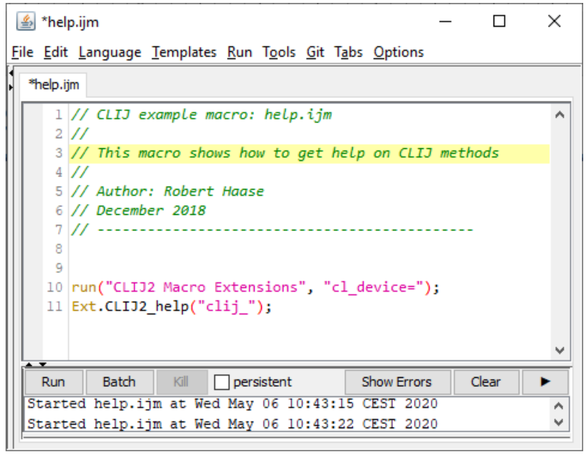
- Functions for working with neighborhood-relationships (between cells)

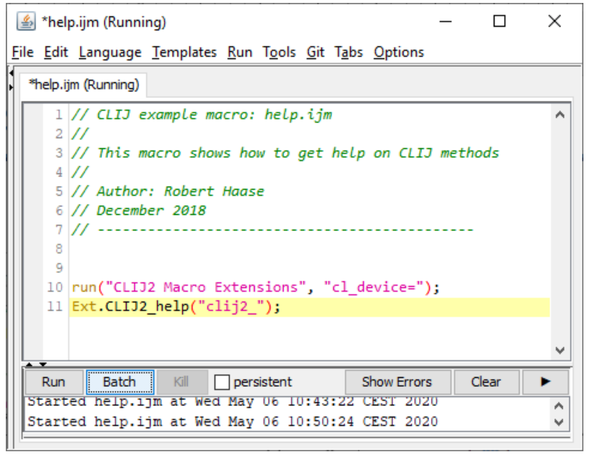


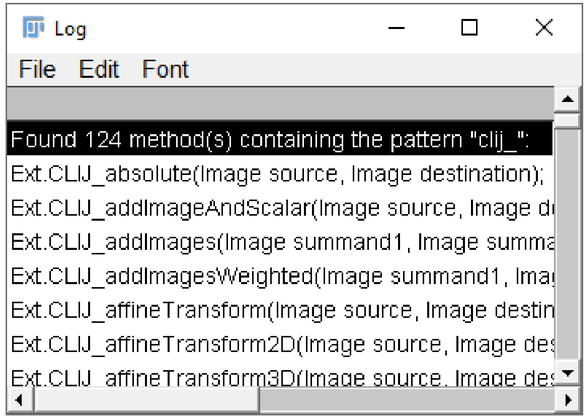
CLIJ2: More functions, wider applicability

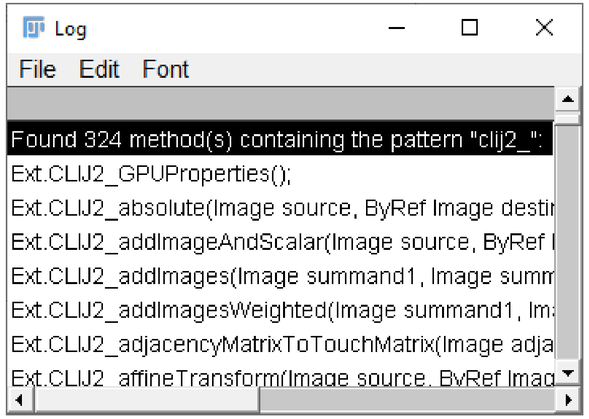
PoL Physics of Life TU Dresden

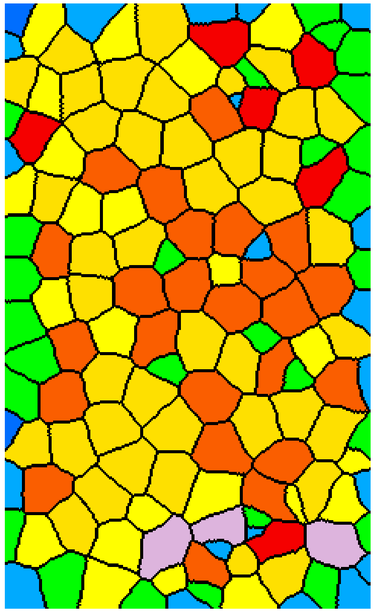
- Functions for working with neighborhood-relationships (between cells)









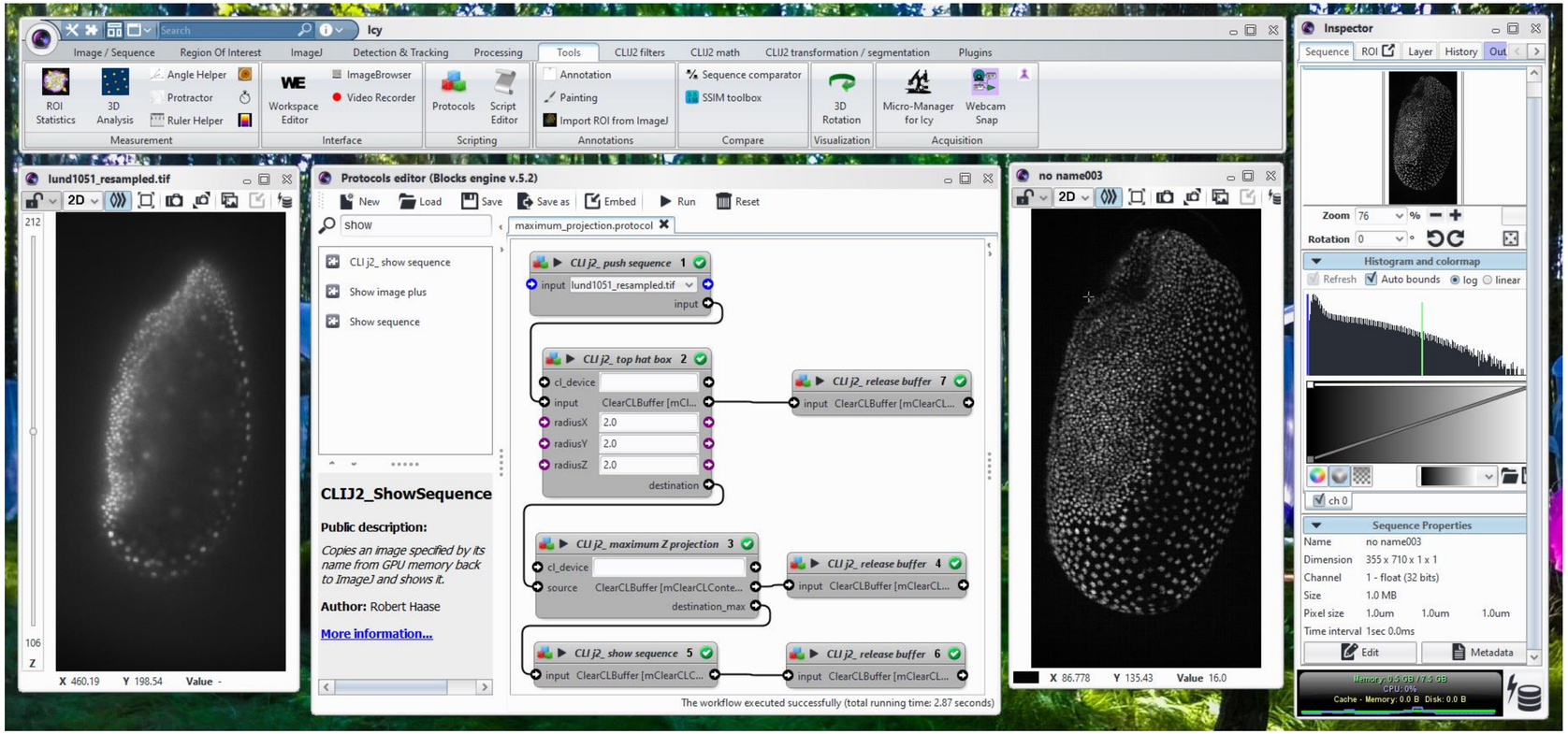


Number of neighbors

CLIJ2: More functions, wider applicability

PoL Physics of Life TU Dresden

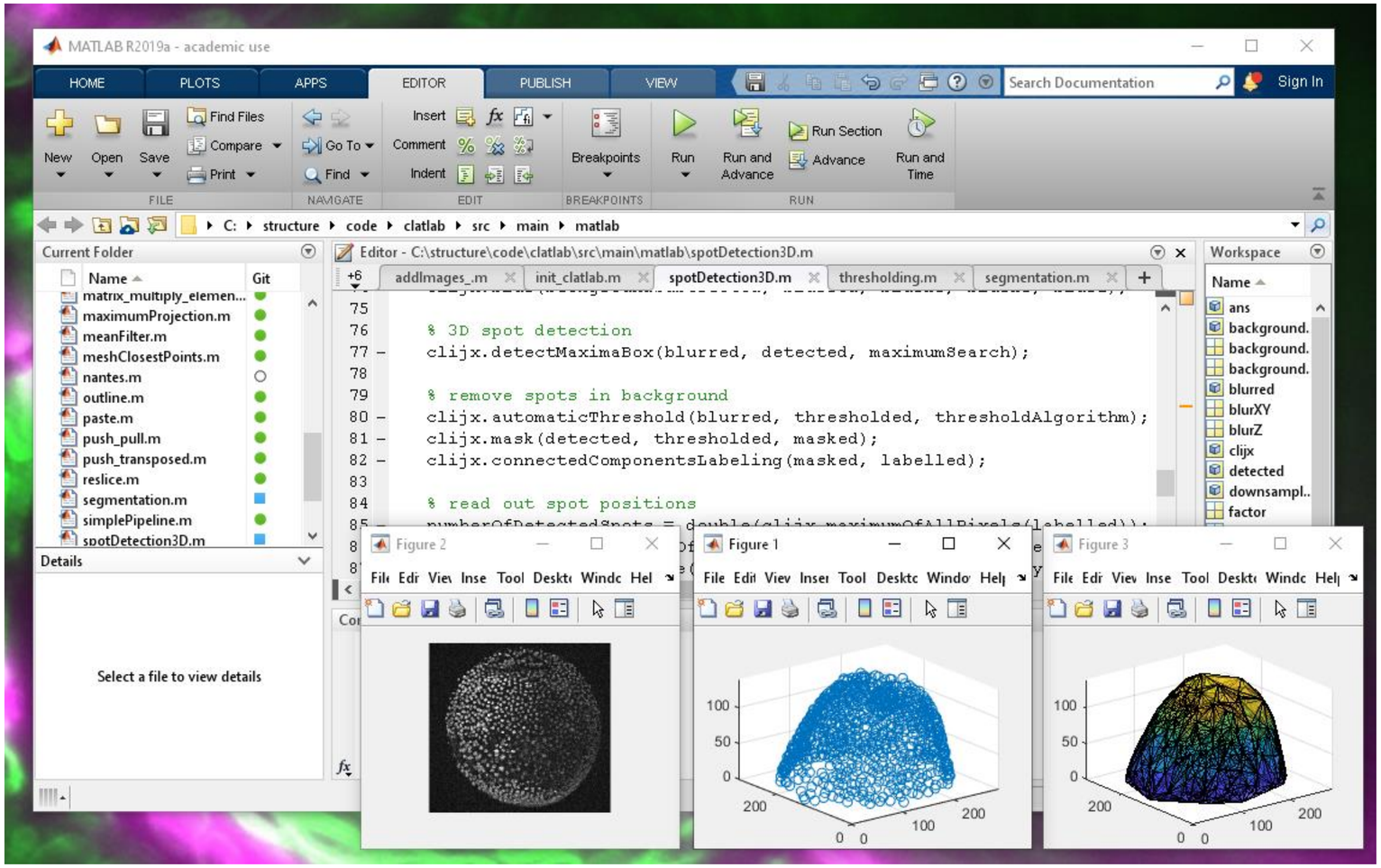
- Functions for working with neighborhood-relationships (between cells)
- Integration for
  - Icy



CLIJ2: More functions, wider applicability

PoL Physics of Life TU Dresden

- Functions for working with neighborhood-relationships (between cells)
- Integration for
  - Icy
  - Matlab



CLIJ2: What every Matlab script must have

PoL Physics of Life TU Dresden

Load data

Initialize GPU

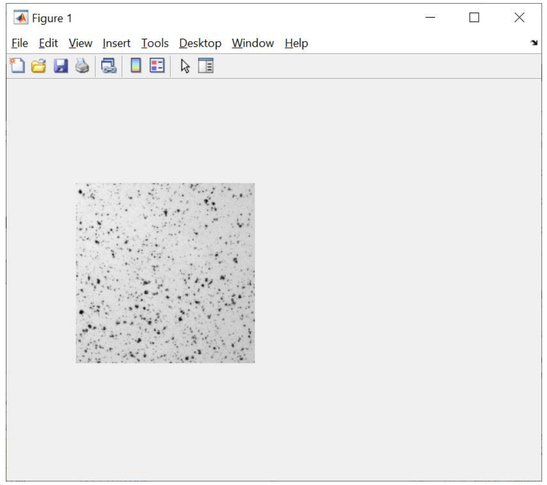
Push

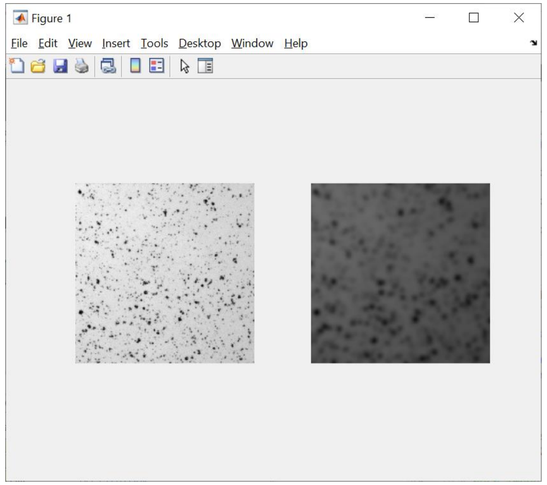
Process images

Pull

Cleanup

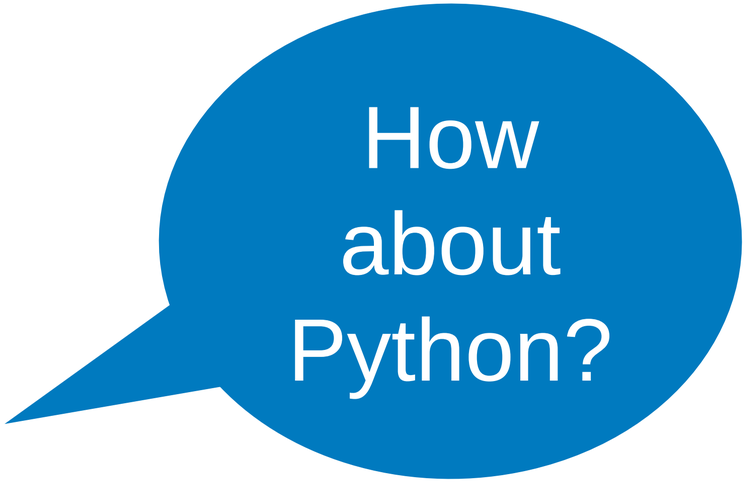
```
% load data
image = imread("http://imagej.nih.gov/ij/images/Cell_Colony.jpg");
subplot(1, 2, 1), imshow(image);

% initialize GPU
clij2 = init_clatlab();

% Push Crop_ResSin1CGFP_016-1.tif to GPU memory
input_image = clij2.pushMat(image);

% process image
sigma = 5;
blurred = clij2.create(input_image);
clij2.gaussianBlur2D(input_image, blurred, sigma, sigma)

% optional: release input data
input_image.close()

% pull result back from GPU
result = clij2.pullMat(blurred);
%subplot(1, 2, 2), imshow(result);

% clean up by the end
clij2.clear()
```

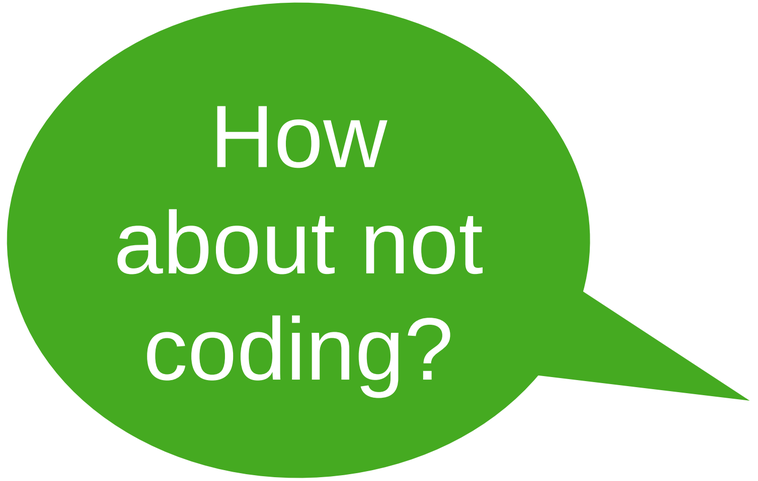




CLIJ3?

- Planned release: Summer 2021





~~CLIJ3~~? -> clEsperanto!

PoL Physics of Life TU Dresden

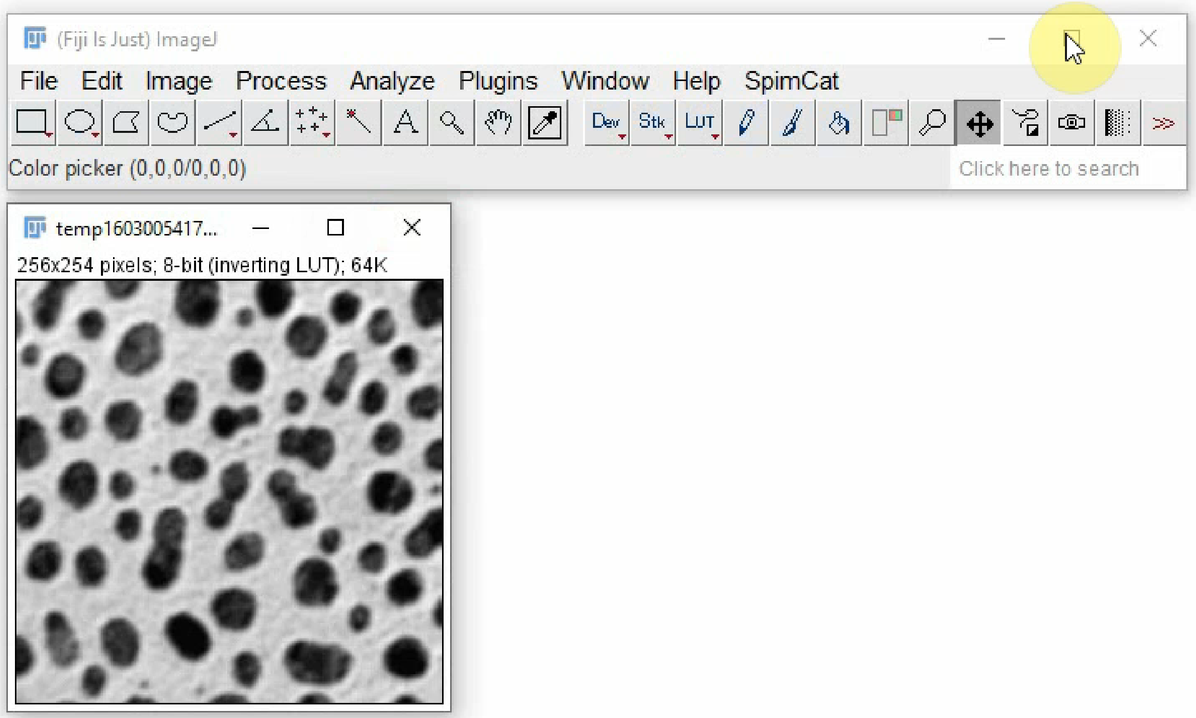
- Planned release: Summer 2021

How about Python?

How about not coding?

cle._

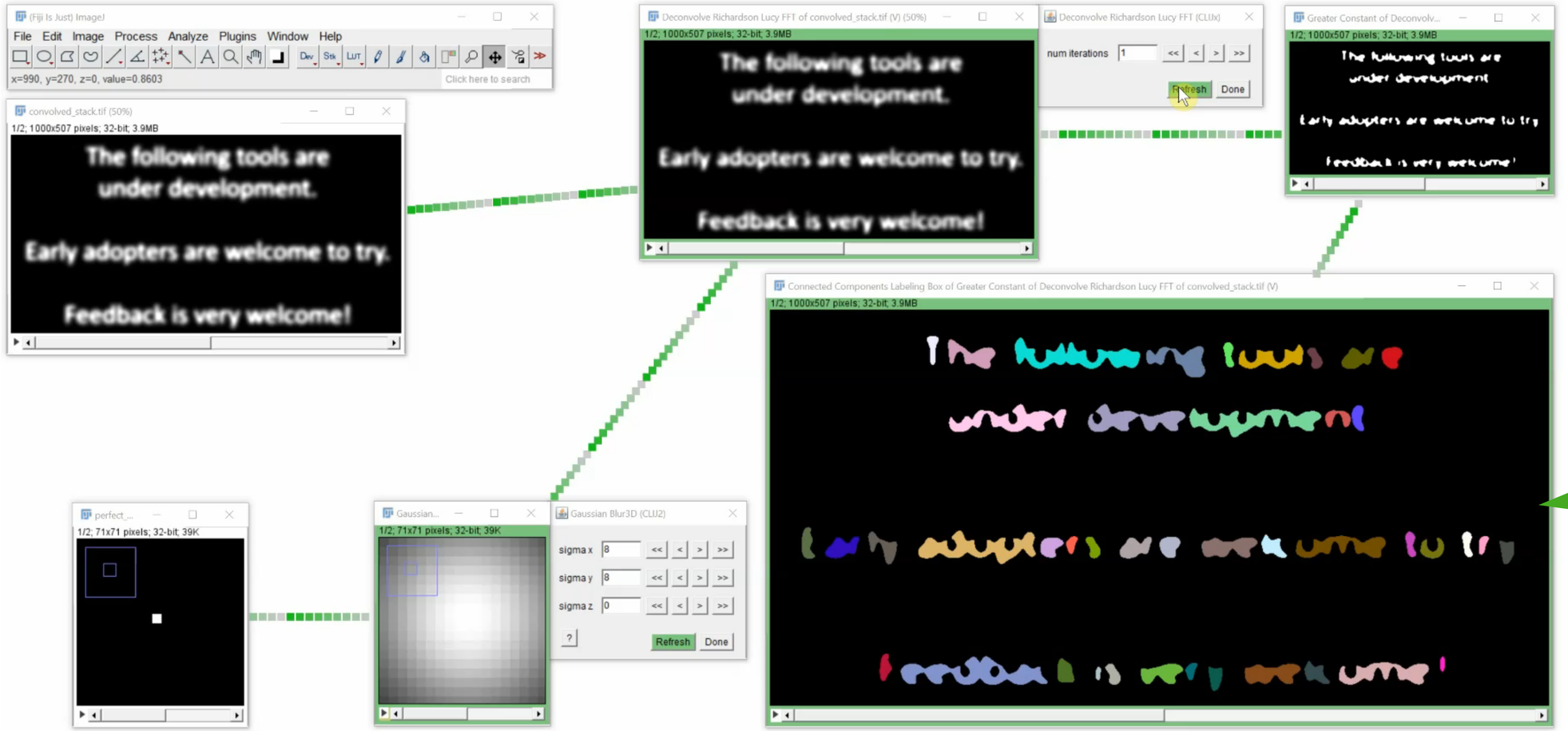
GPU acceleration – without coding



GPU acceleration – without coding

PoL Physics of Life TU Dresden

- “Image Data Flow Graphs” allow better understanding of connections between operations

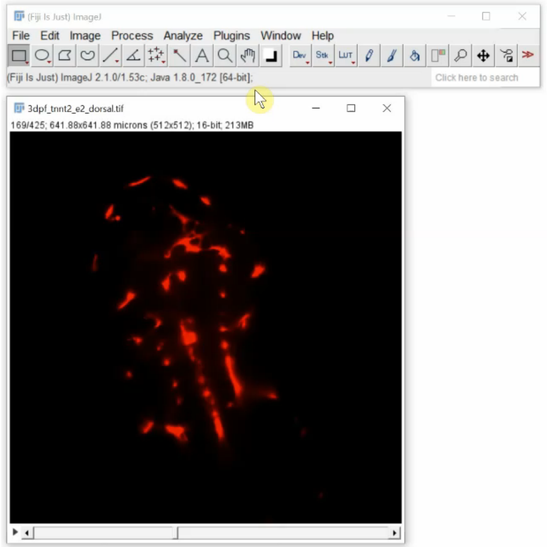


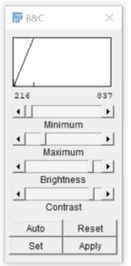
Special thanks to Brian Northon!

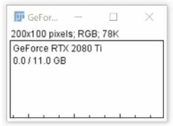
Brian Northon
@truenorth_ia

GPU acceleration + code generation

- After setting up the workflow, generate code!







Special thanks to Elisabeth Kugler!

Elisabeth Kugle
@KuglerElisabeth

Image data source: Elisabeth Kugler; labs of Tim Chico and Paul Armitage, The University of Sheffield (UK)" https://zenodo.org/record/4204839#.X8DCRGhKQ2w

Image Data Flow Graphs & clEsperanto

PoL Physics of Life TU Dresden

- Study relationships between operations and compare different strategies

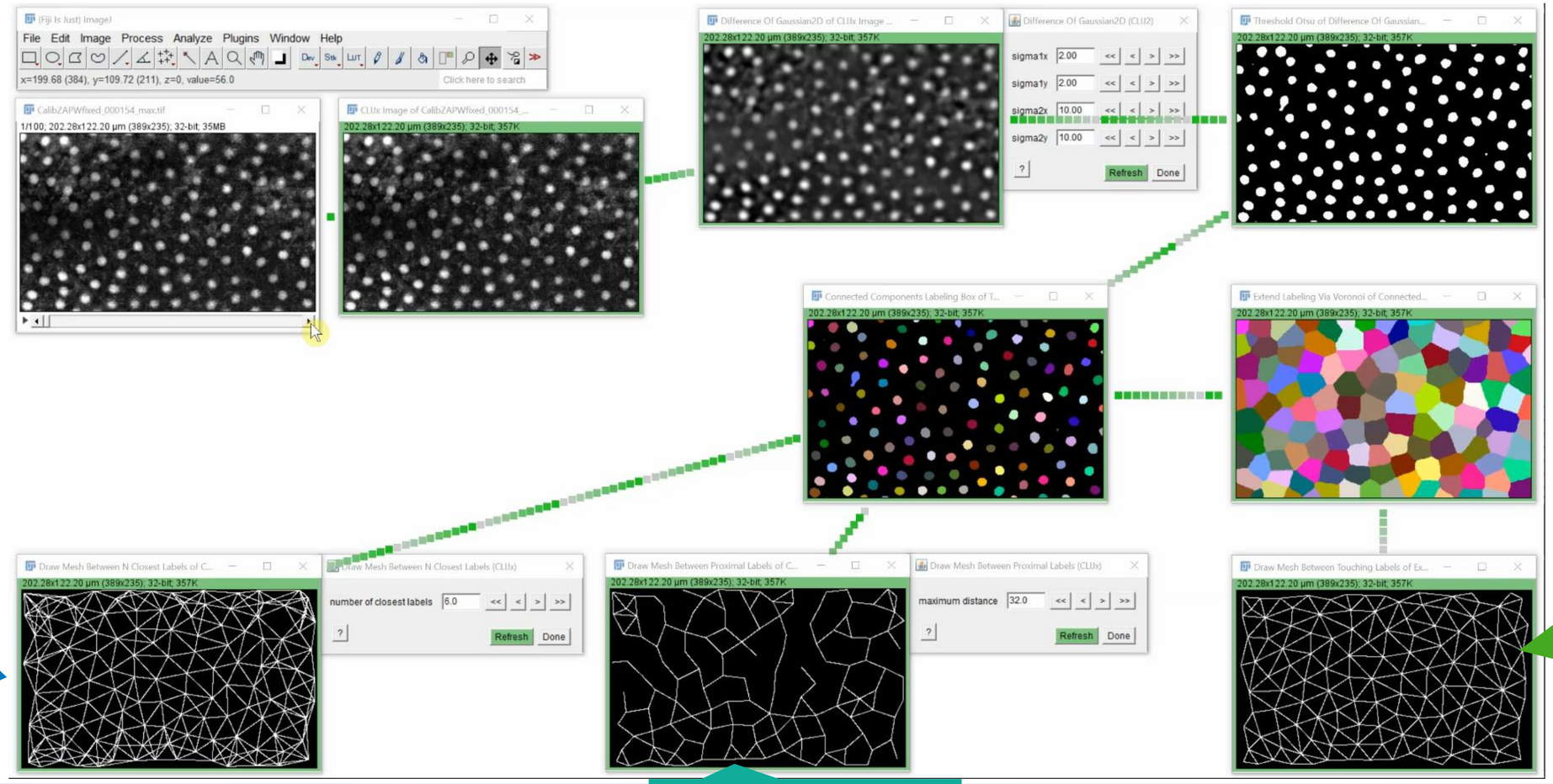
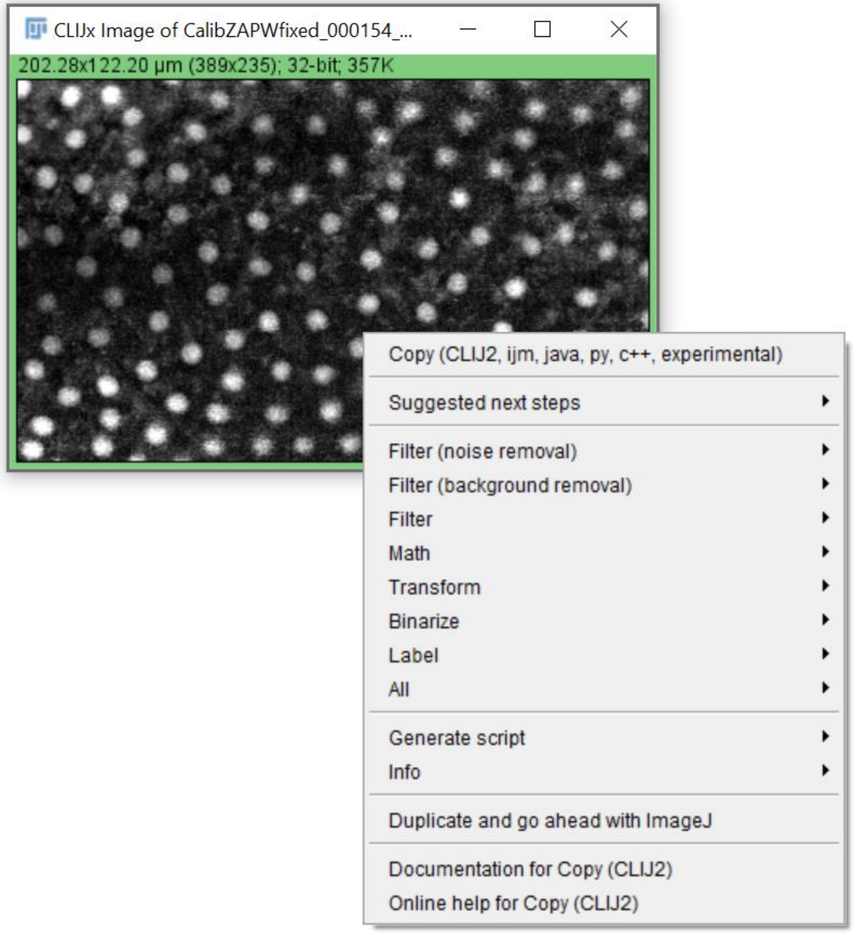
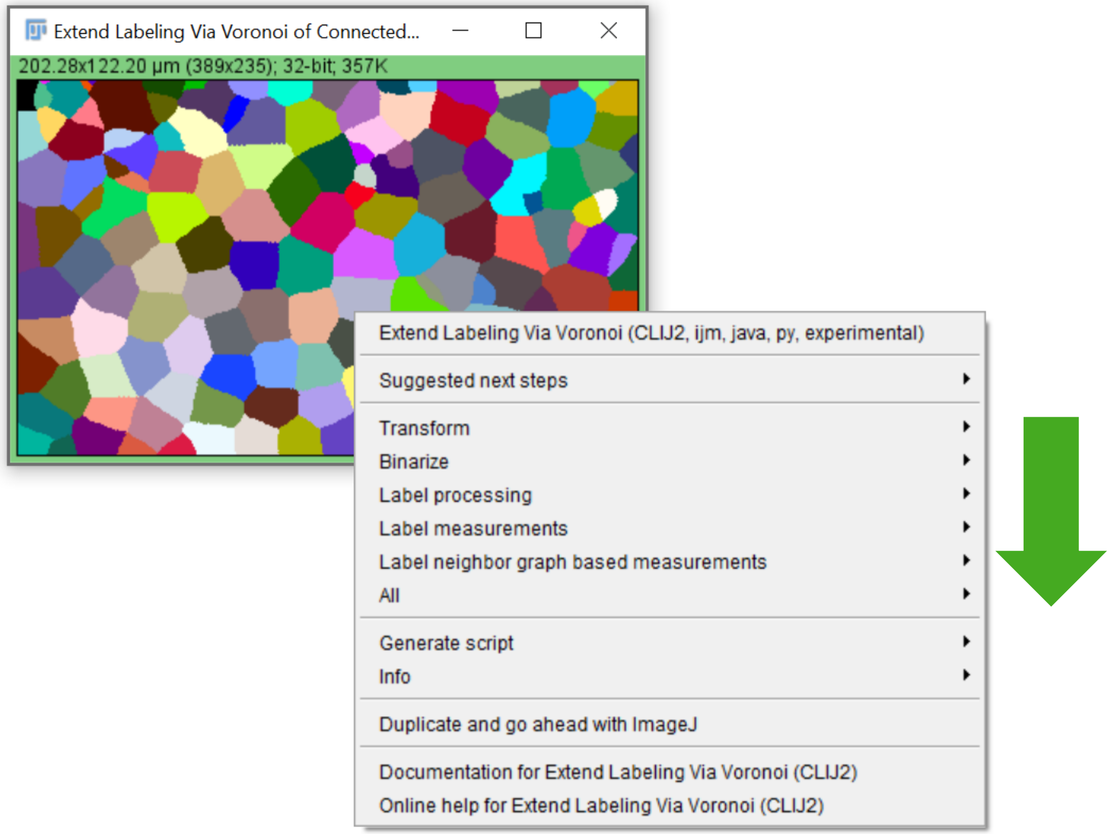


Image Data Flow Graph design

PoL Physics of Life TU Dresden

The menu order is intentional: From preprocessing to analysis

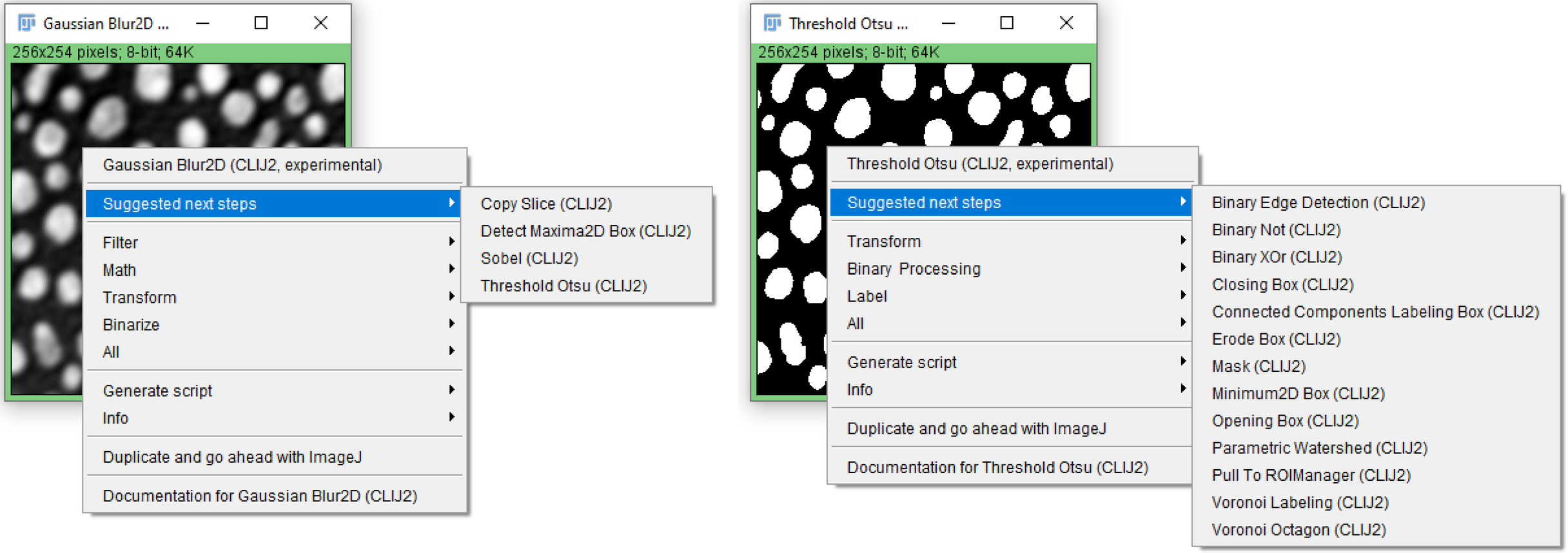




Expert system: Suggestions

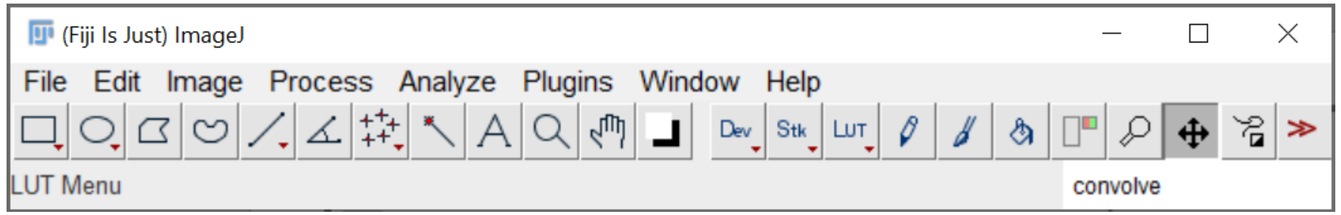
PoL Physics of Life TU Dresden

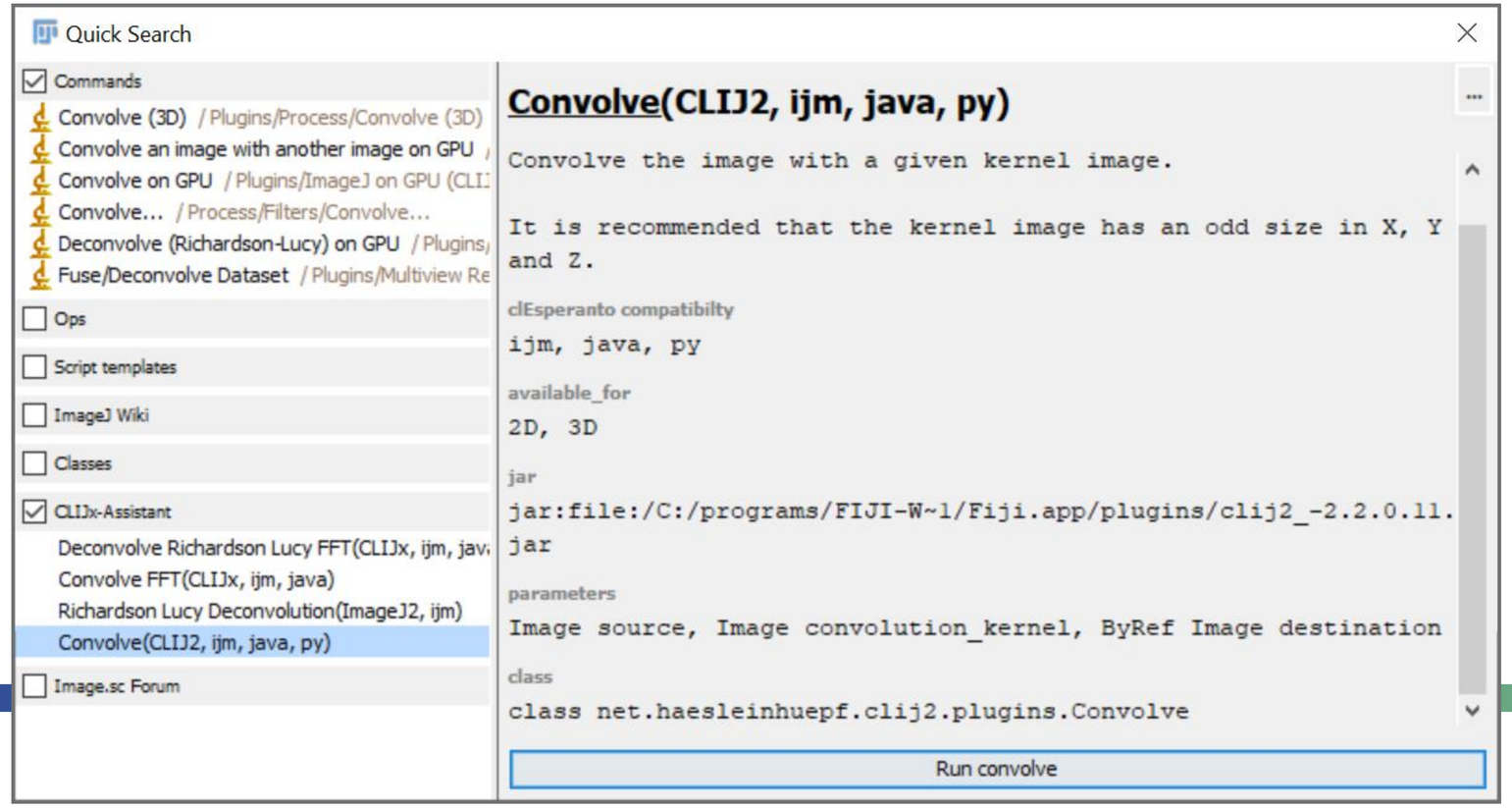
Explore suggestions!



Fiji search bar

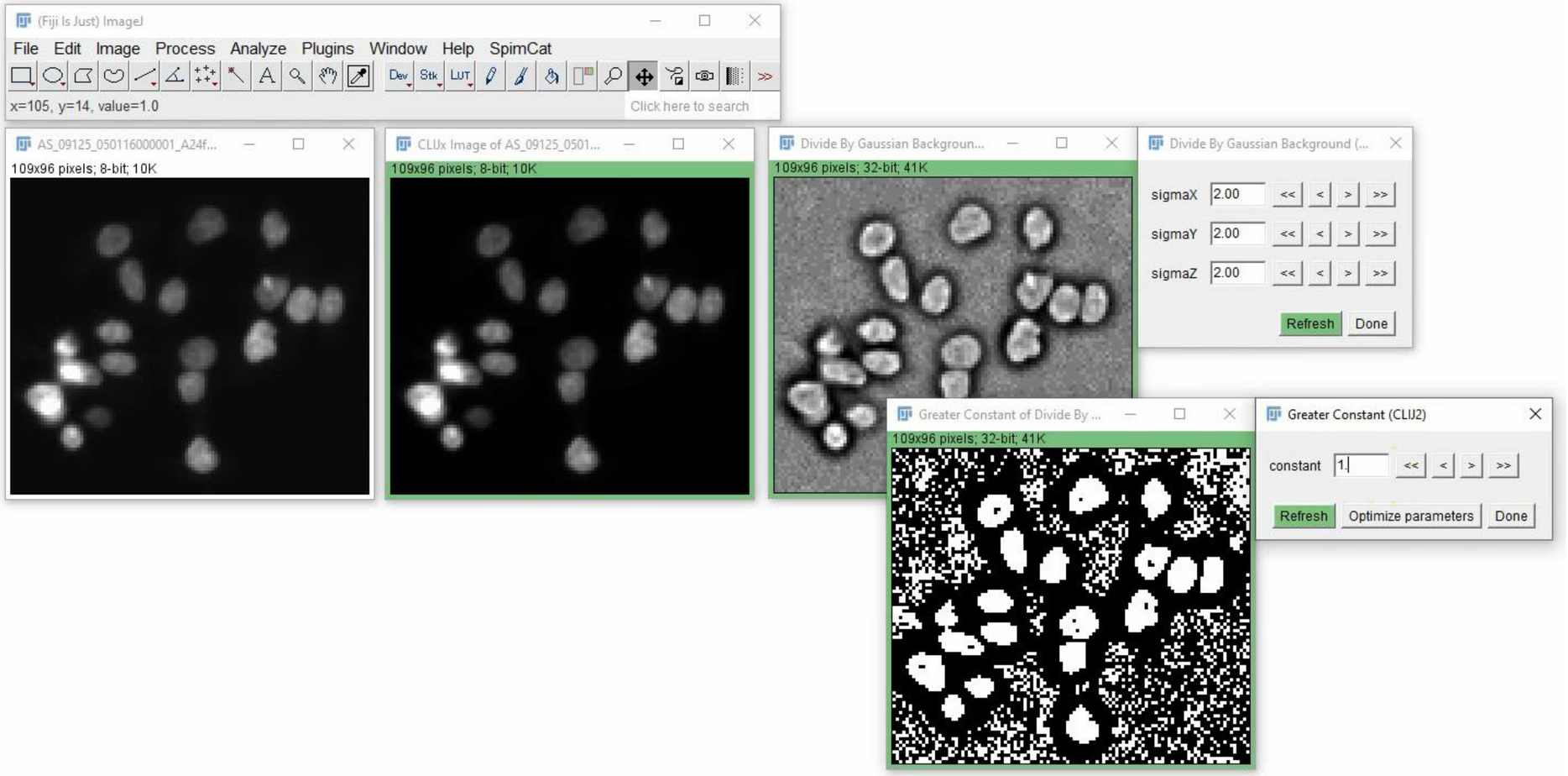
In Fijis search bar result, there is a new category: CLIJx-assistant which offers IDFG operations





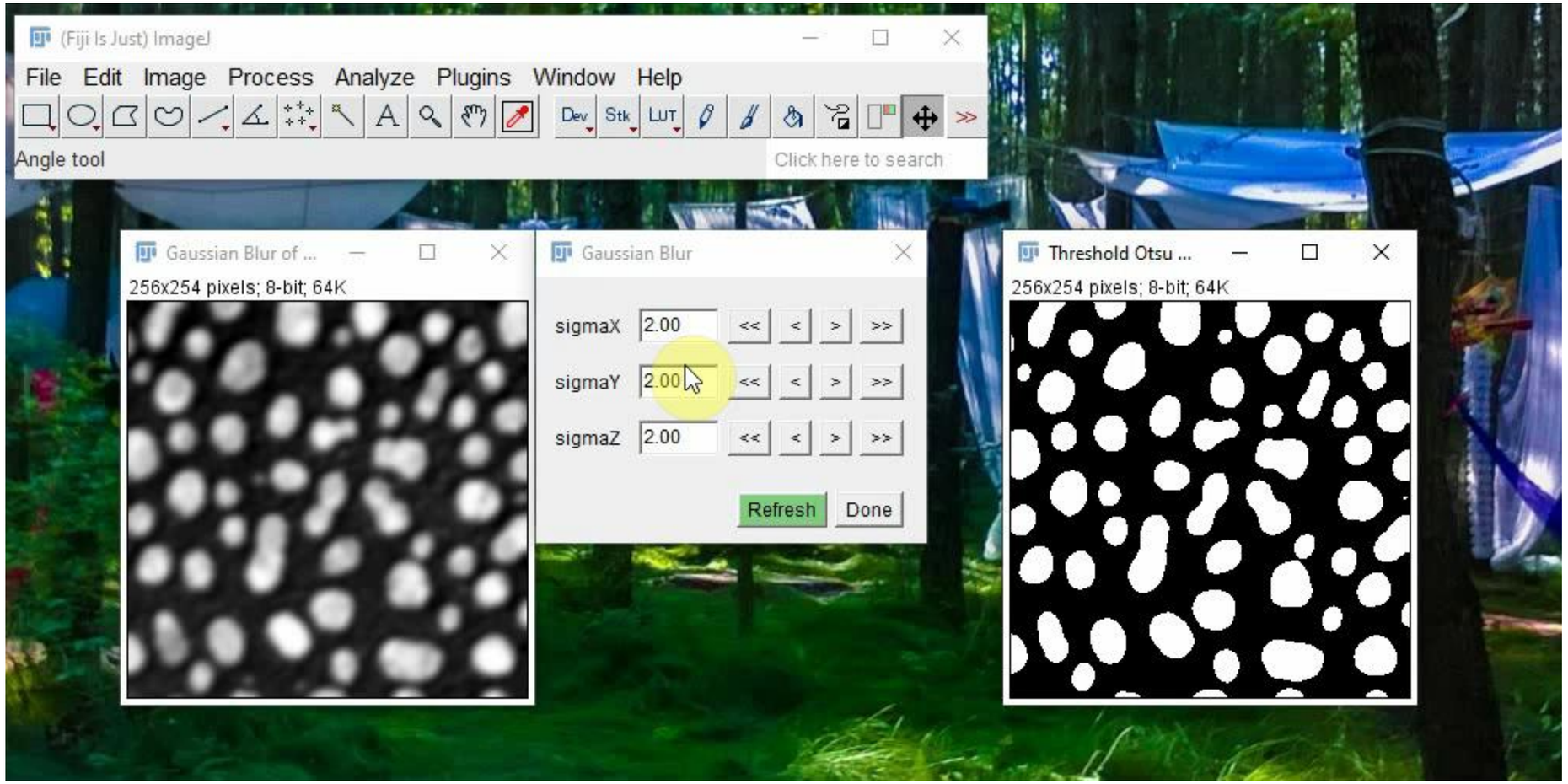
Parameter optimization

If your image data flow graph ends in a binary image, you can optimize numeric parameters automatically.



Reset parameters

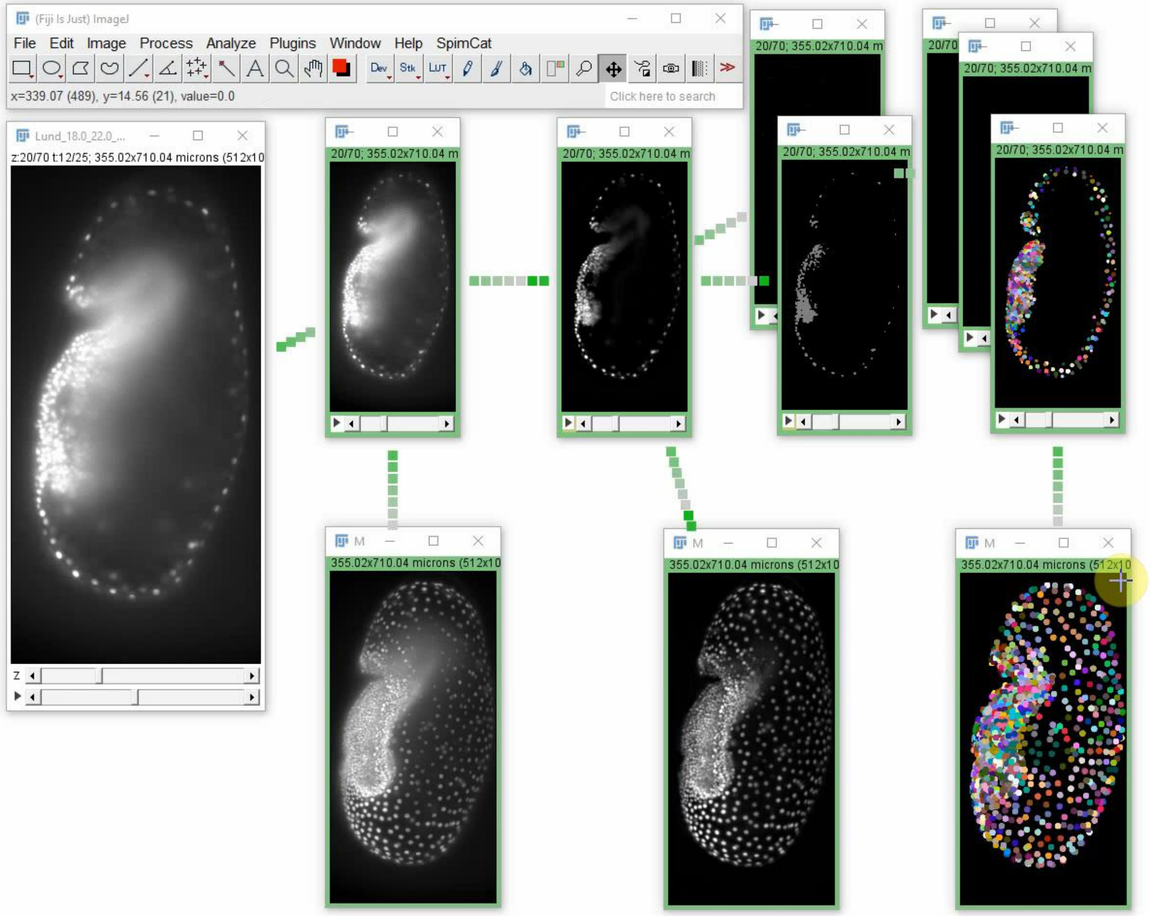
If you or the optimizer screwed up parameter settings, you can get them back from the history!



Export to scripts

PoL
Physics of Life
TU Dresden

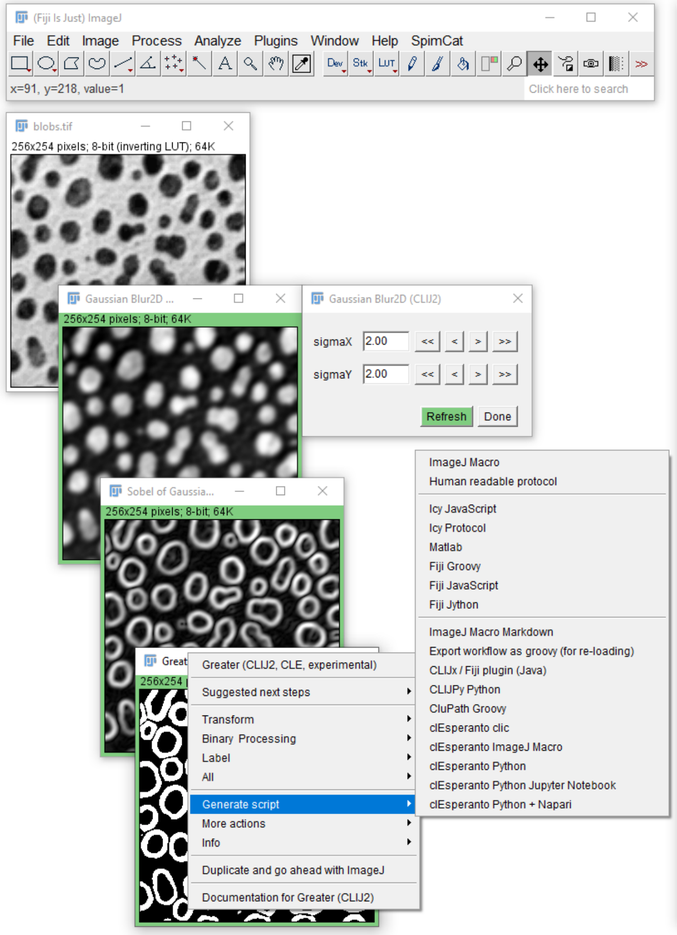
Export to multiple programming languages

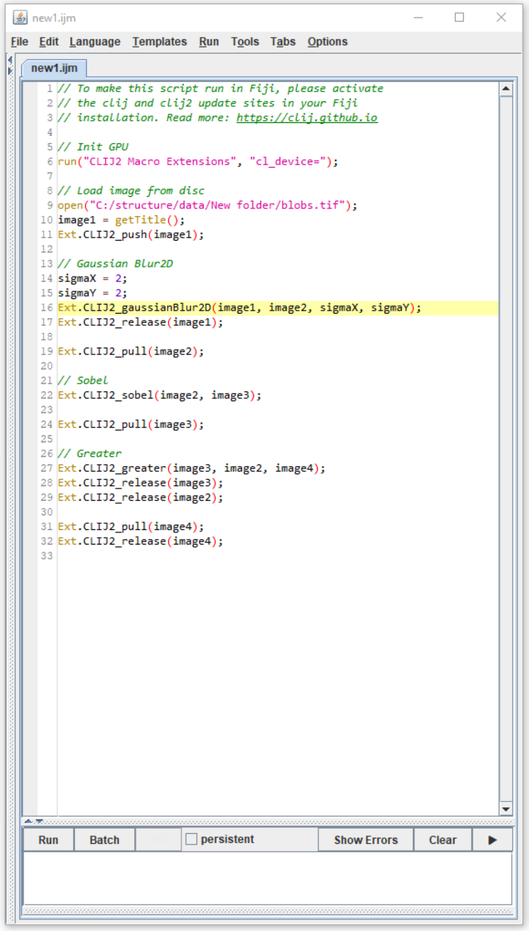


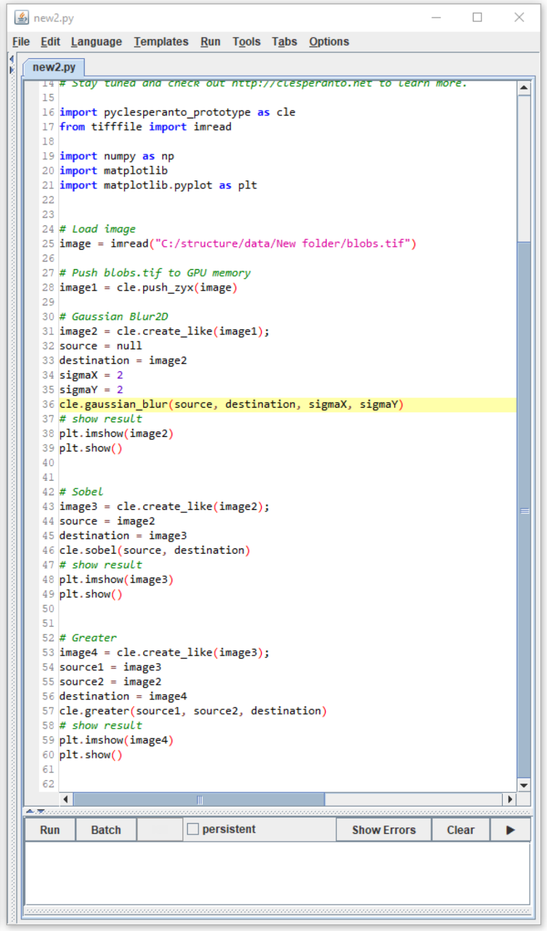
clEsperanto script language comparison

PoL Physics of Life TU Dresden

Generate multiple scripts in multiple languages from a given Image Data Flow Graph

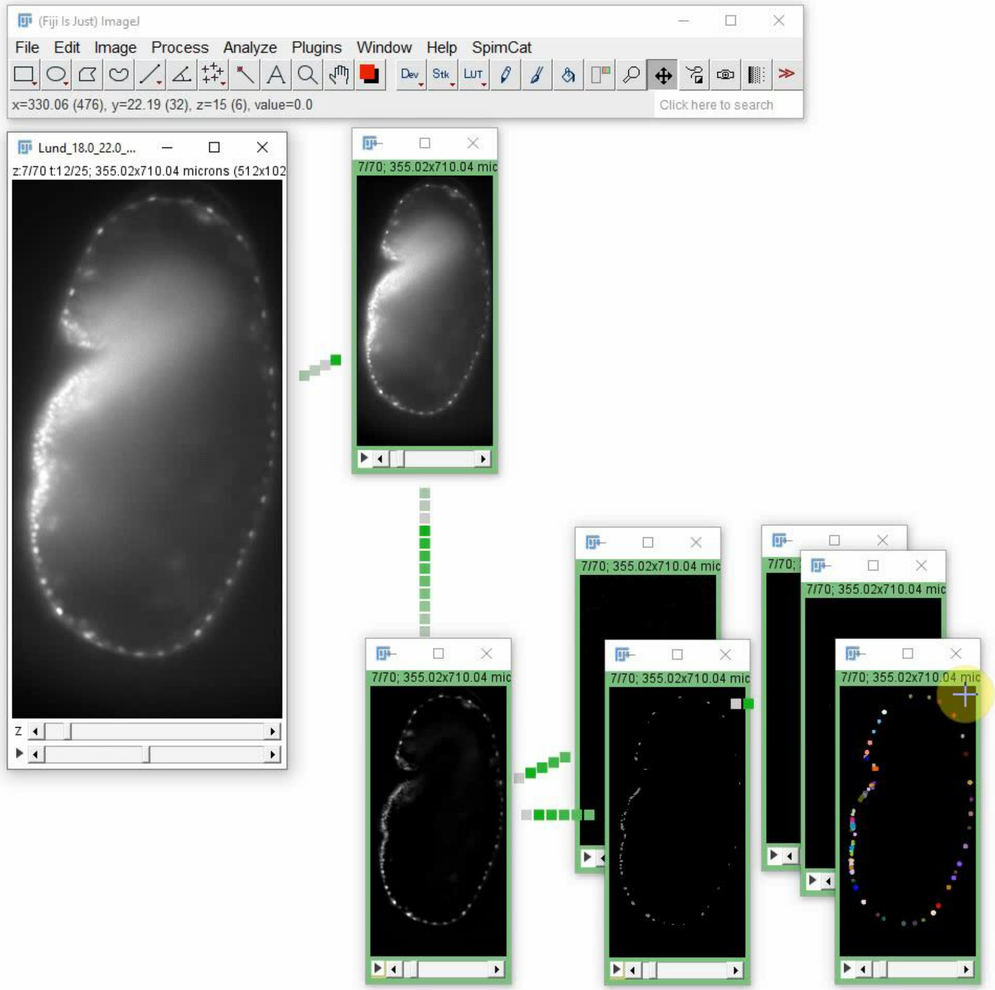






Export from Fiji to napari

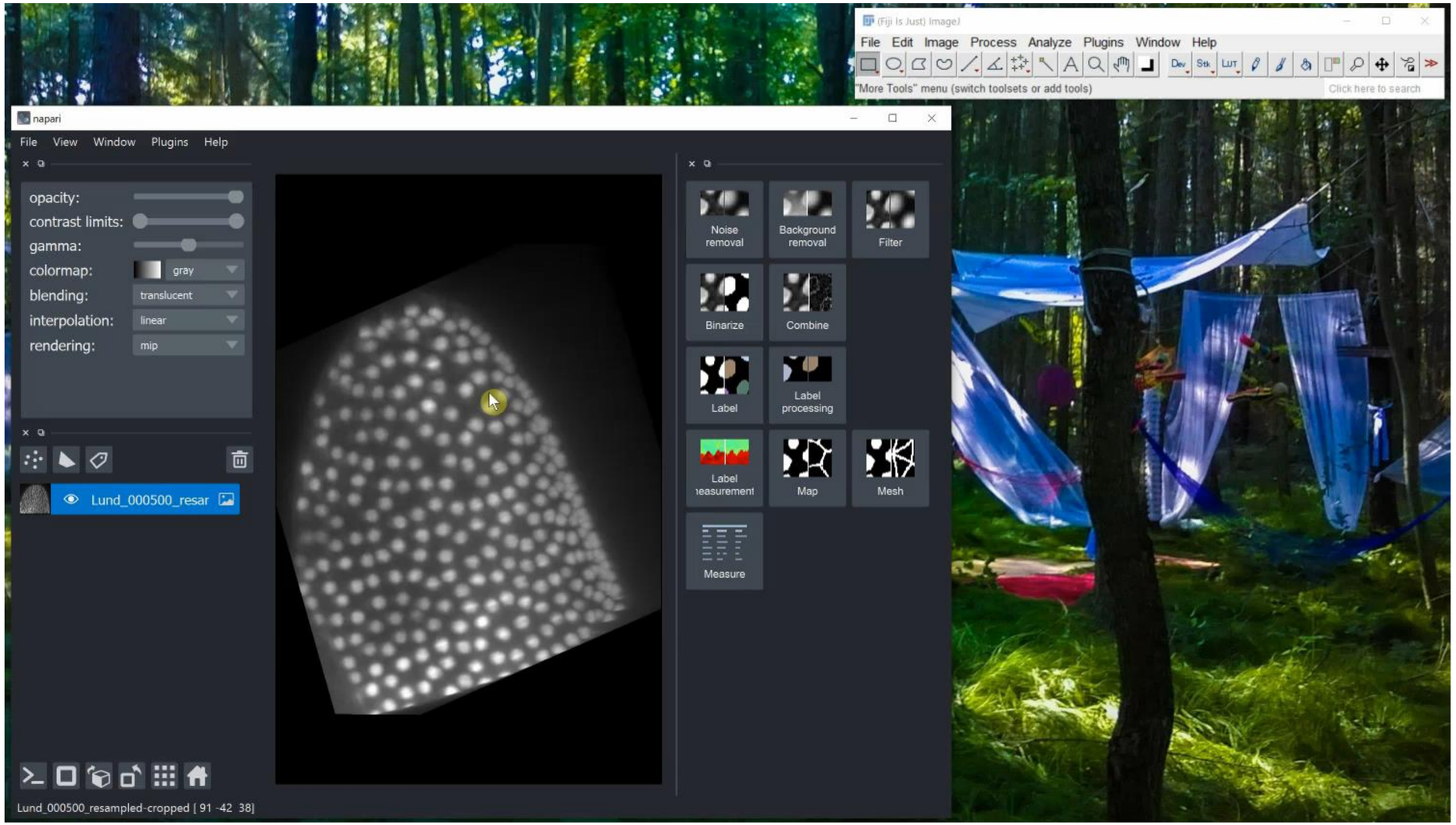
napari is a multidimensional image viewer in the Python ecosystem



Export from napari to Fiji

PoL Physics of Life TU Dresden

Using the napari-pyclesperanto-assistant



py-clEsperanto: What every Python script must have

PoL Physics of Life TU Dresden

Load data

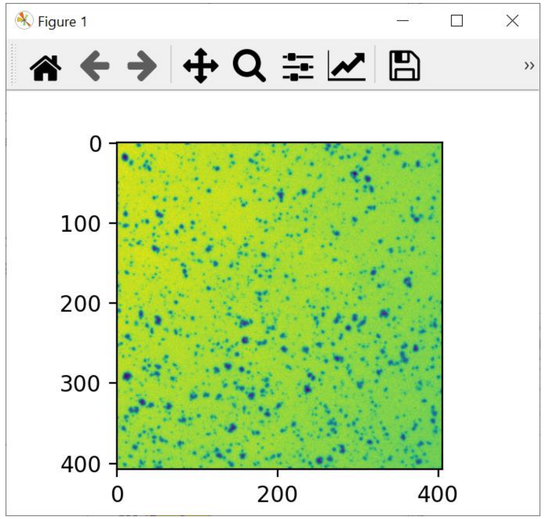
Initialize GPU

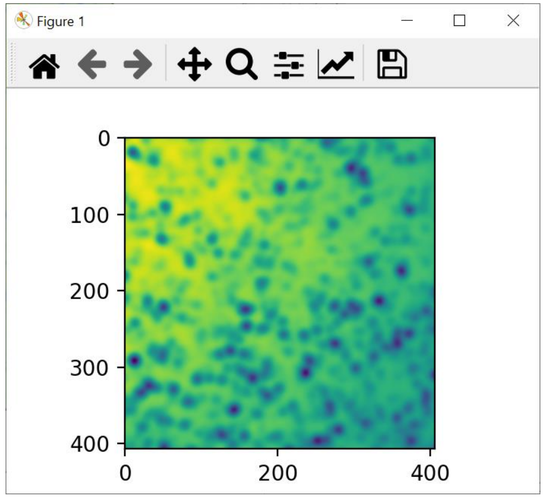
Push

Process images

Pull

```python
from skimage.io import imread
import matplotlib.pyplot as plt

# Load image
image = imread("https://imagej.nih.gov/ij/images/Cell_Colony.jpg")
plt.imshow(image)
plt.show()

# initialize GPU
import pyclesperanto_prototype as cle
# optional:
cle.select_device("GPU name")

# Push blobs.tif to GPU memory
image1 = cle.push_zyx(image)

# Threshold Otsu
sigma = 5
blurred = cle.create_like(image1)
cle.gaussian_blur(image1, blurred, sigma, sigma)

# pull result back from GPU
result = cle.pull_zyx(blurred)
plt.imshow(result)
plt.show()
```



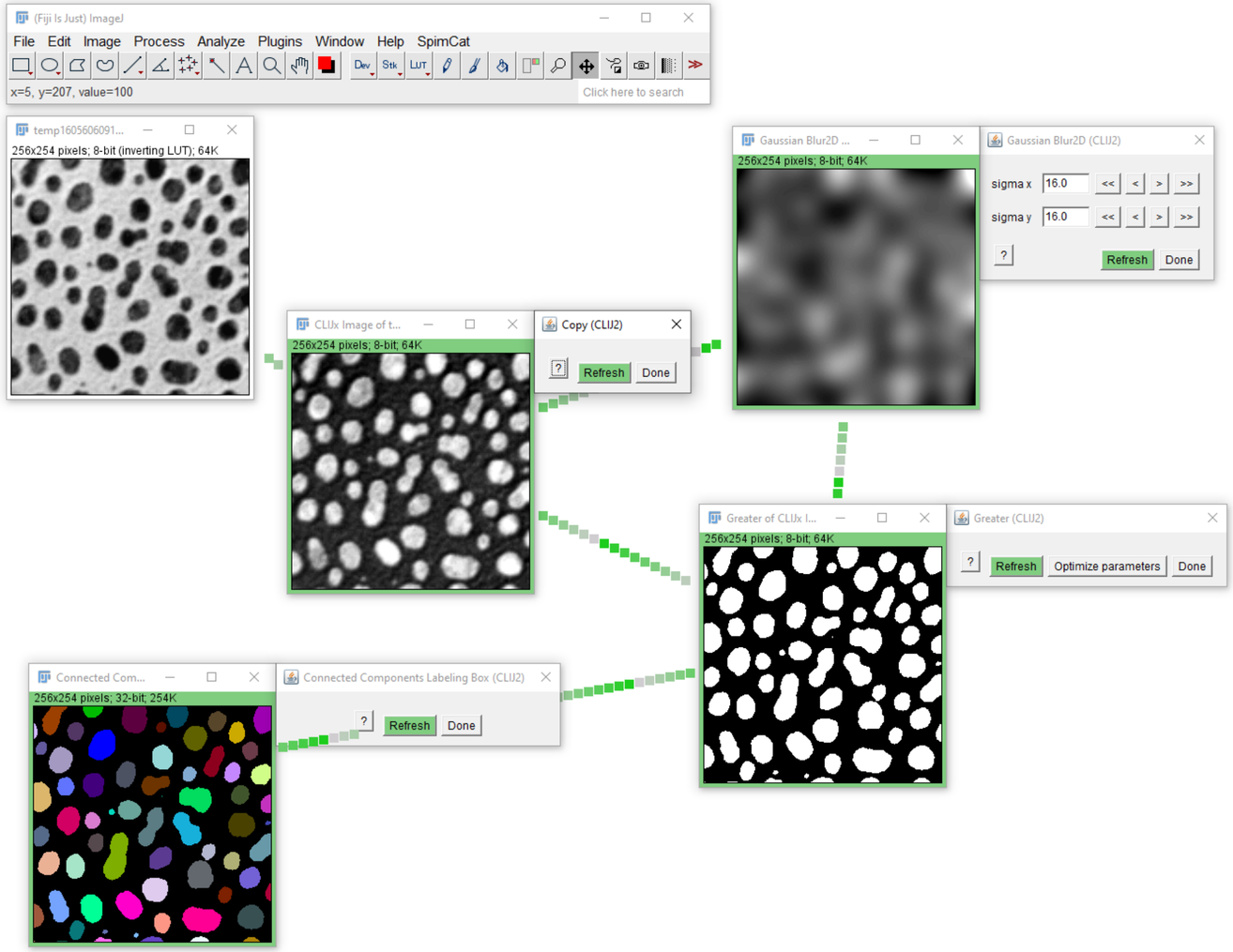


Exercises

Exercise 3: Segment blobs.gif

Design a workflow for segmenting blobs.gif (File > Open Samples...)

- Export the workflow as ImageJ Macro script for and Fiji Jython.
- Optional: Export the workflow as Icy Javascript, as Icy protocol and for QuPath as groovy script. Feel free to generate a Fiji plugin.
- Furthermore, if you only used operations, which are "py" compatbile, export a clEsperanto-based Jupyter notebook and a Python script that uses Napari.



Exercise 4: Install & test pyclesperanto

PoL Physics of Life TU Dresden

Install pyclesperanto

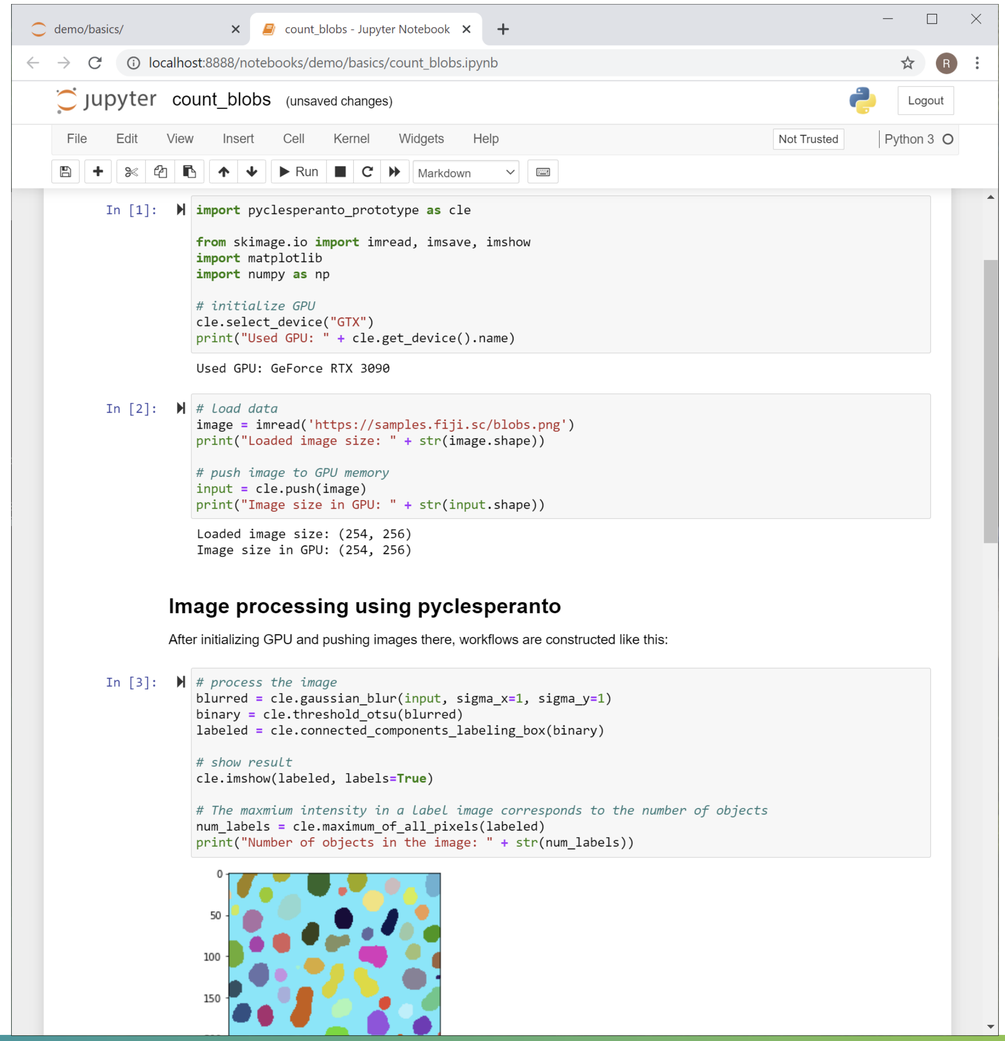
```
pip install pyclesperanto_prototype
```

And download & execute this notebook:

https://github.com/clEsperanto/pyclesperanto_prototype/blob/master/demo/basics/count_blobs.ipynb

Linux users: you need to install OpenCL, e.g.

```
apt-get install ocl-icd-devel
```

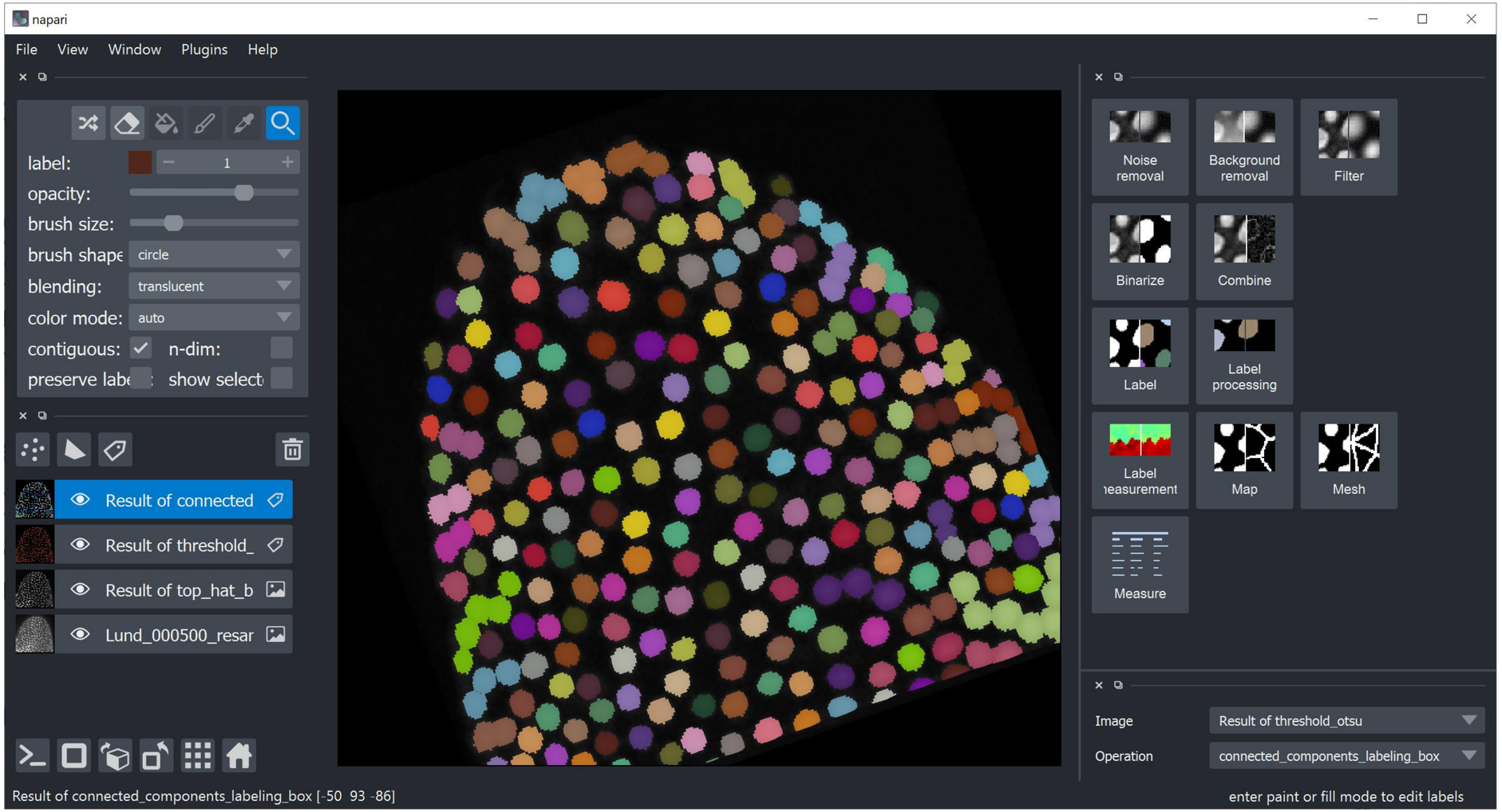


Exercise 5: Napari-pyclesperanto-assistant

PoL Physics of Life TU Dresden

Install and try the assistant for napari

https://clesperanto.github.io/napari_pyclesperanto_assistant/



Exercise 6: Benchmarking: reuse memory

PoL Physics of Life TU Dresden

Compare reusing memory and re-allocating memory. Execute some operations in a loop and measure processing time.

Reuse memory

```
for i in range(0, num_iterations):
    cle.maximum_sphere(flip, flop, 10, 10, 0)
    cle.minimum_sphere(flop, flip, 10, 10, 0)
```

Re-allocate memory

```
for i in range(0, num_iterations):
    flop = cle.maximum_sphere(flip, radius_x=10, radius_y=10, radius_z=0)
    flip = cle.minimum_sphere(flop, radius_x=10, radius_y=10, radius_z=0)
```

Command Prompt

```
(beetlesafari) C:\structure\code\i2k2020_tutor
Used GPU: GeForce RTX 2080 Ti
Image size: (50, 1024, 1024)
Flip-flop took 0.3108978271484375s
Flip-flop took 0.218419551849365 23s
Flip-flop took 0.219372749328613 28s
Flip-flop took 0.219413280487060 55s
Flip-flop took 0.220448255538940 43s
Flip-flop took 0.22041201591491 7s
Flip-flop took 0.219418764114379 88s
Flip-flop took 0.22140479087829 59s
Flip-flop took 0.220408201217651 37s
Flip-flop took 0.220408916473388 67s
Re-alloc took 0.3904910087585449s
Re-alloc took 0.35600066184997 56s
Re-alloc took 0.361290931701660 16s
Re-alloc took 0.362677097320556 64s
Re-alloc took 0.36241197586059 57s
Re-alloc took 0.361839056015014 65s
Re-alloc took 0.36122393608093 26s
Re-alloc took 0.363983154296875s
Re-alloc took 0.367833852767944 34s
Re-alloc took 0.352453708648681 64s

(beetlesafari) C:\structure\code\i2k2020_tutor
```

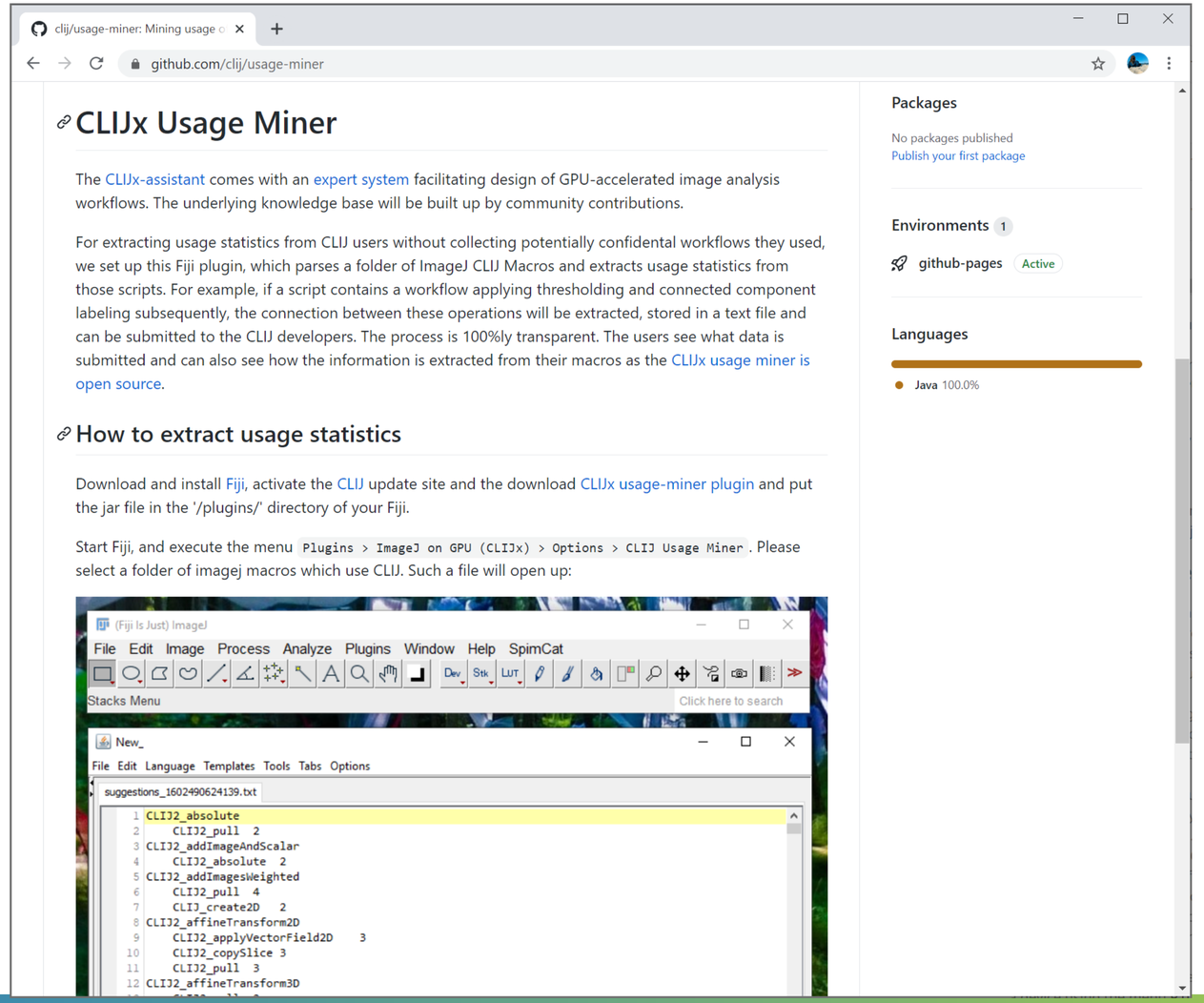
Exercise 7: Usage miner

Homework: After you implemented some ImageJ Macros at home, let us analyse your command usage statistics. Visit the usage-miner website to learn how.

https://github.com/clij/usage-miner

With your contribution, we can make suggestions better!

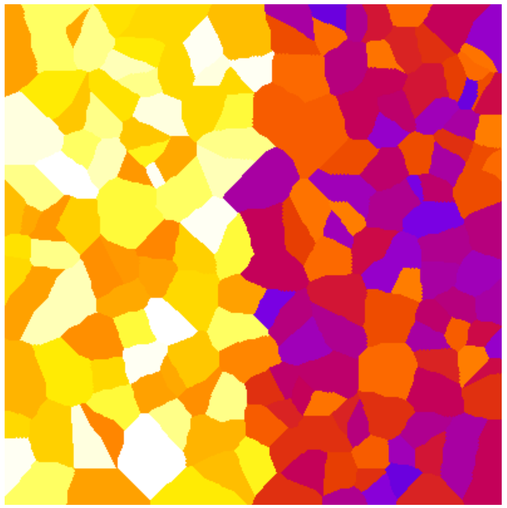
Thanks ☺



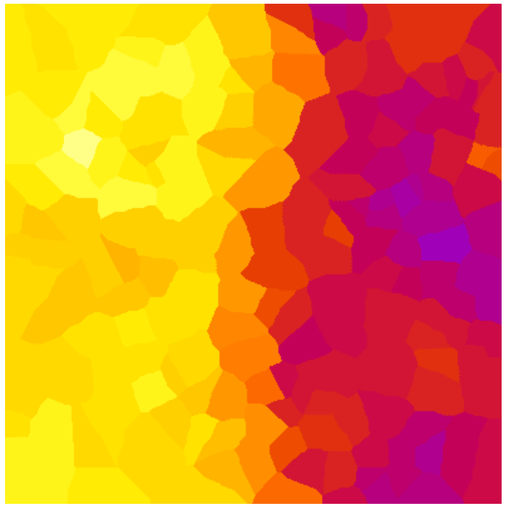
Part III: Life-sciences specific image processing

Operations on adjacency graphs

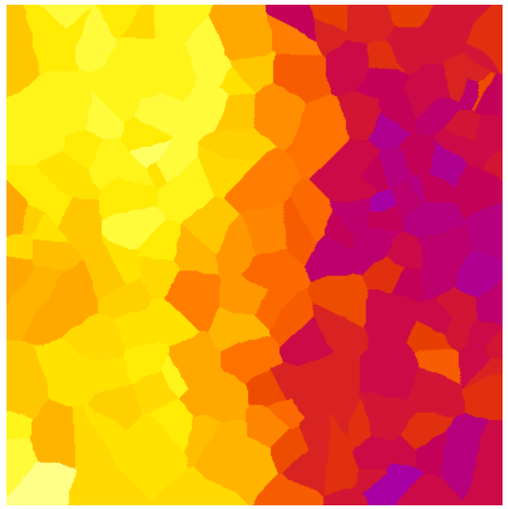
- “Pixels aren’t necessarily squares. They can have any shape.”



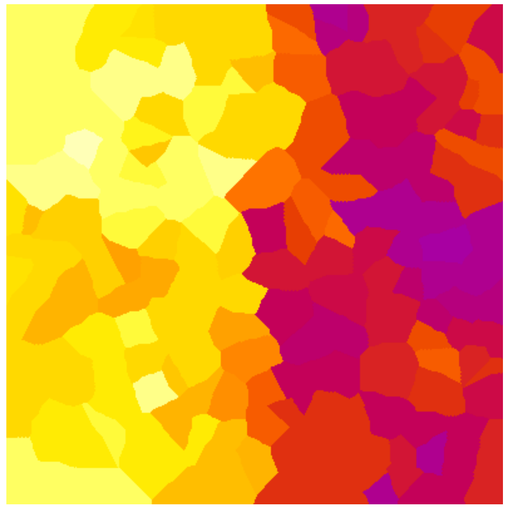
Original



Mean filtered



Mean filtered
(wider radius)



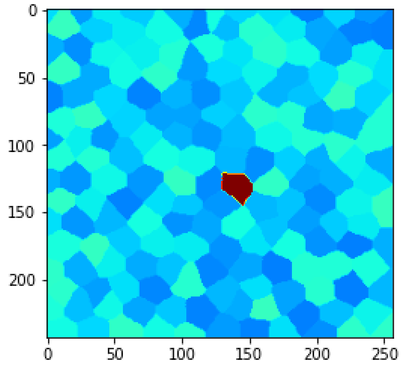
Median filtered

I wasn’t aware of any software where you can just do these operations.
Thus, with CLIJ2 you can.

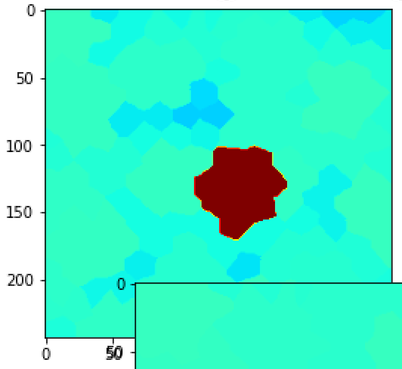
Neighborhood definitions

PoL Physics of Life TU Dresden

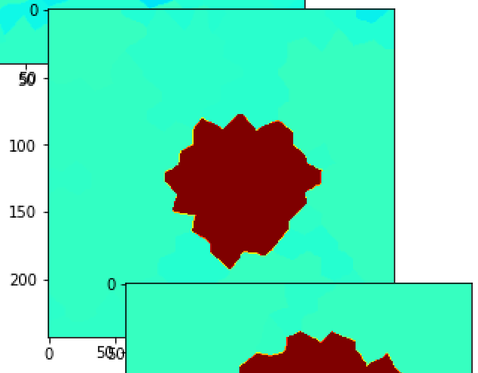
"Cells"



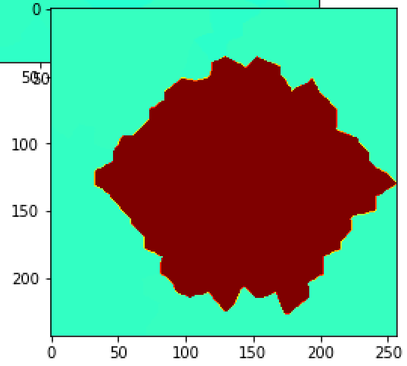
Touching neighbors



Neighbors of neighbors

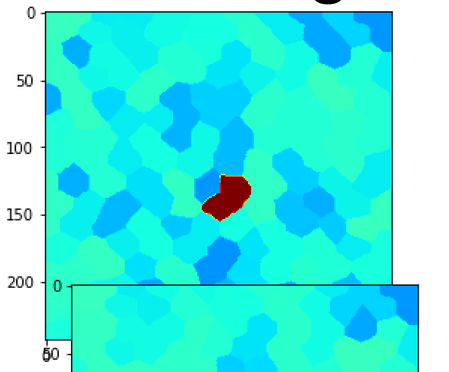


of neighbors

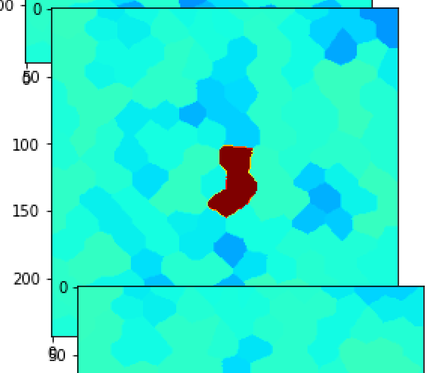


N nearest neighbors

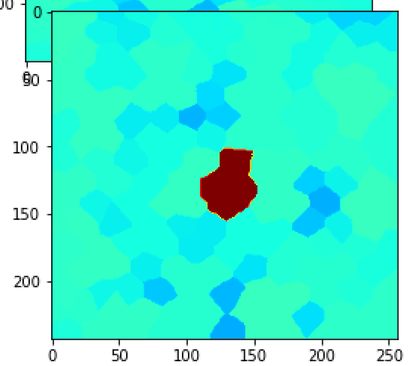
n=2



n=3

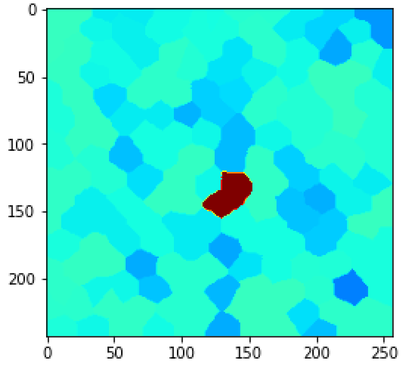


n=4

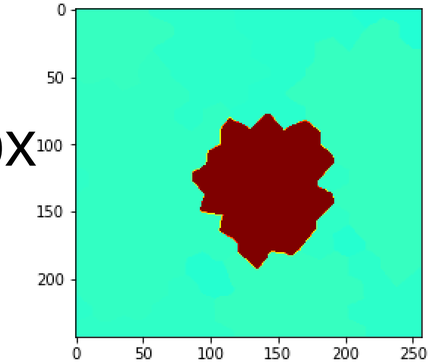


Proximal neighbors

d < 20 px



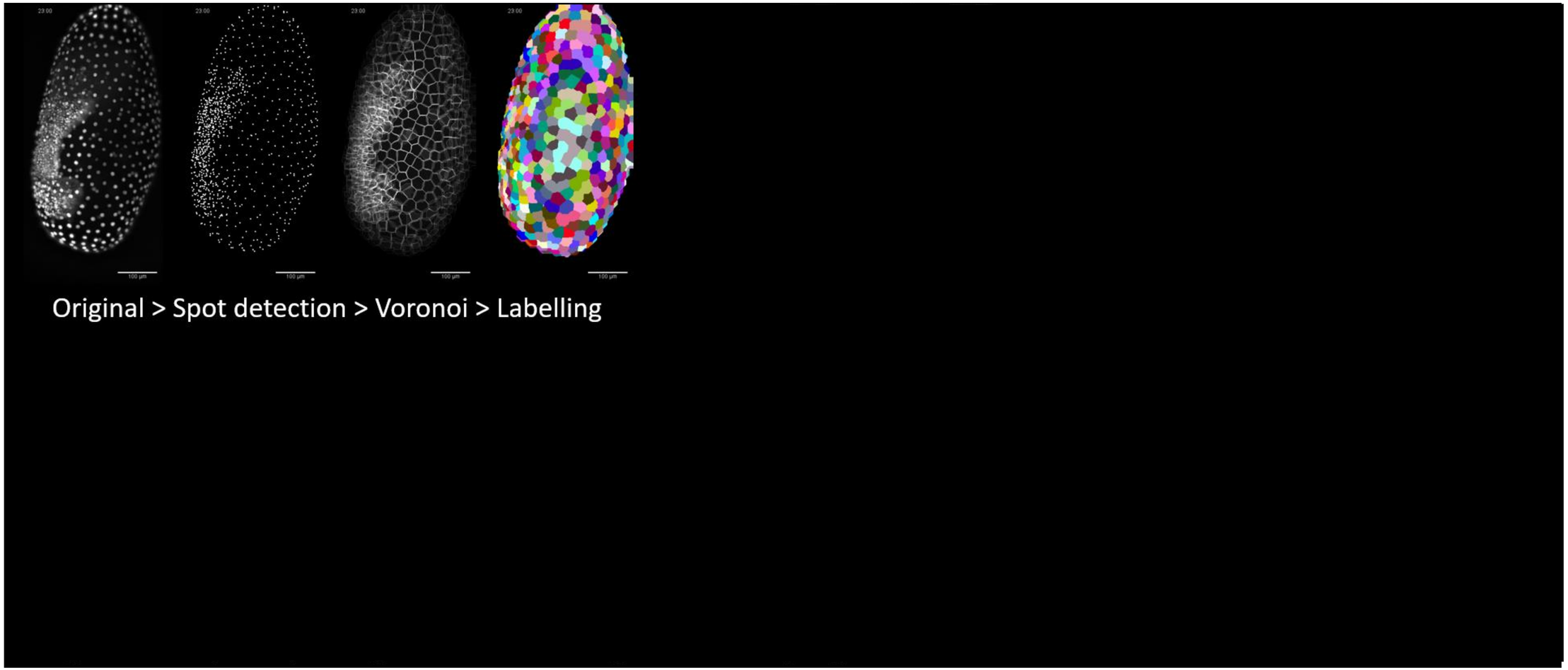
d < 120 px



Operations on adjacency graphs

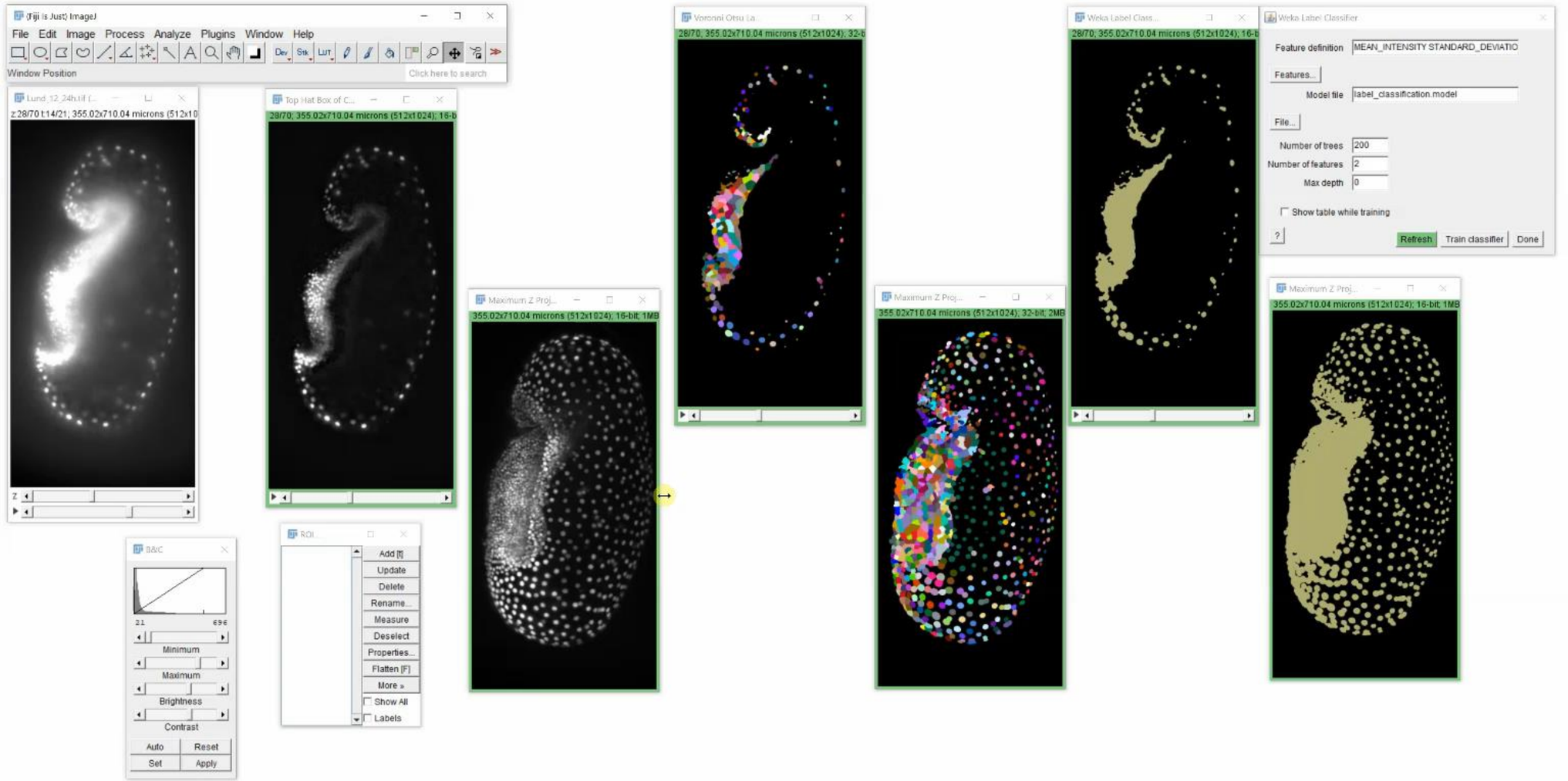
PoL
Physics of Life
TU Dresden

Convolution and machine learning for cell classification



Machine learning for cell & tissue classification

PoL
Physics of Life
TU Dresden

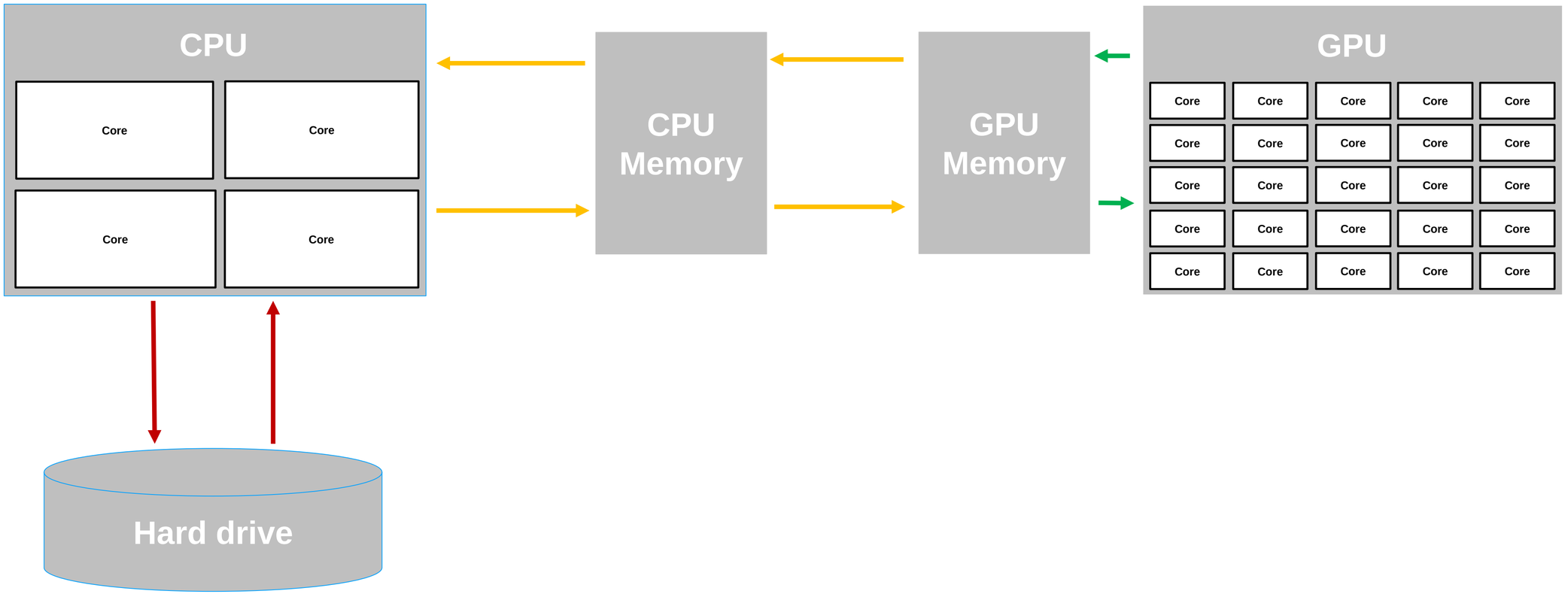


Part IV: Benchmarking

GPUs allow real-time image processing

PoL
Physics of Life
TU Dresden

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



GPU-accelerated image processing

PoL Physics of Life TU Dresden

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

vs.

Workstation CPU

2x Intel Xeon Silver 4110

Workstation GPU

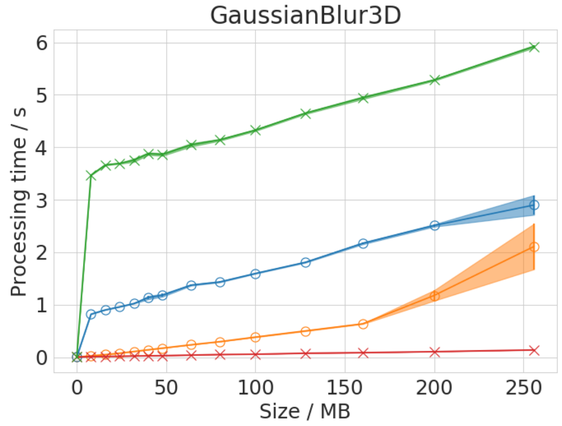
Nvidia Quadro P6000

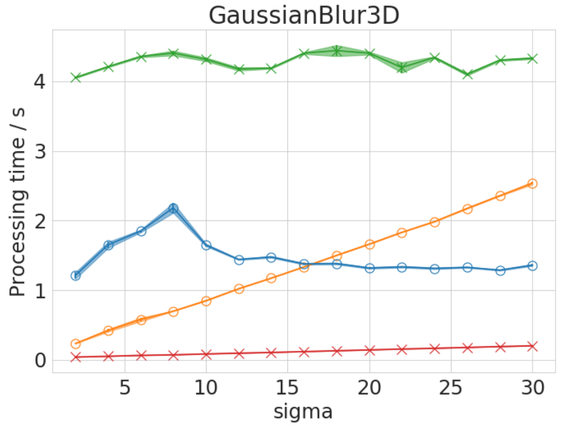
Laptop CPU

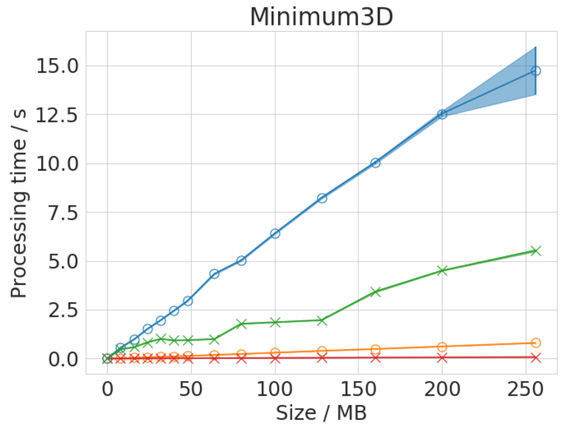
Intel Core i7-8650U

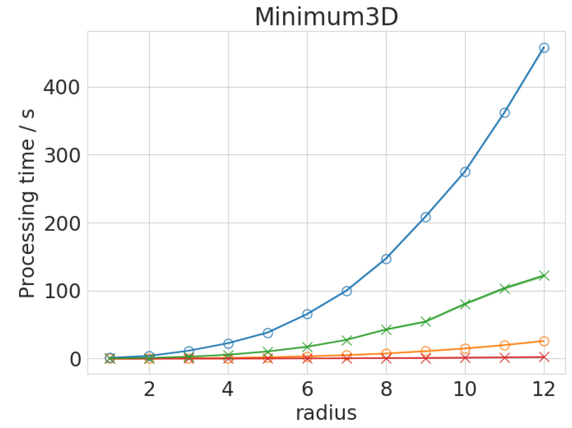
Laptop GPU

Intel UHD 620 GPU







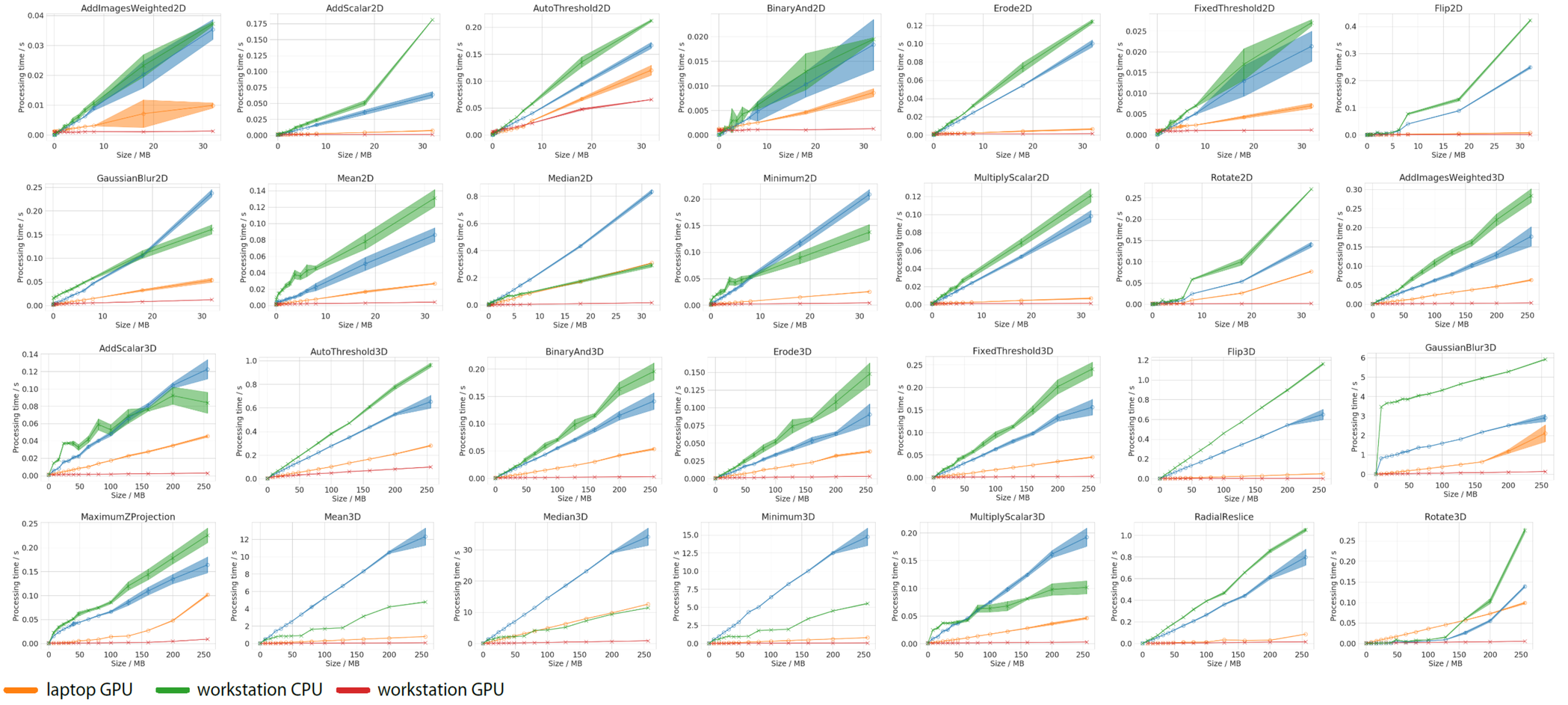


Haase et al. Nat Methods (2020), https://www.nature.com/articles/s41592-019-0650-1

GPU-accelerated image processing

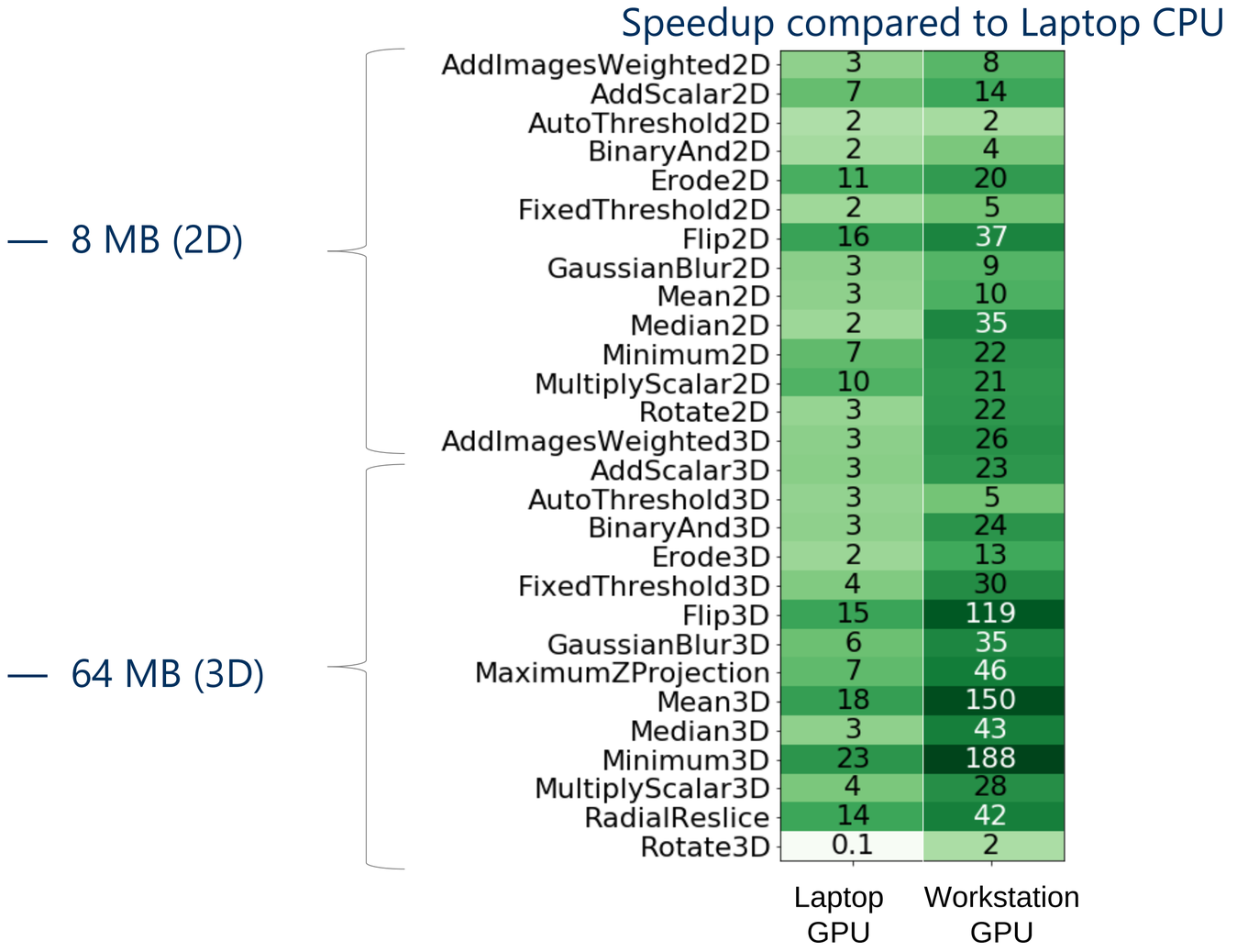
PoL Physics of Life TU Dresden

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



GPU-accelerated image processing

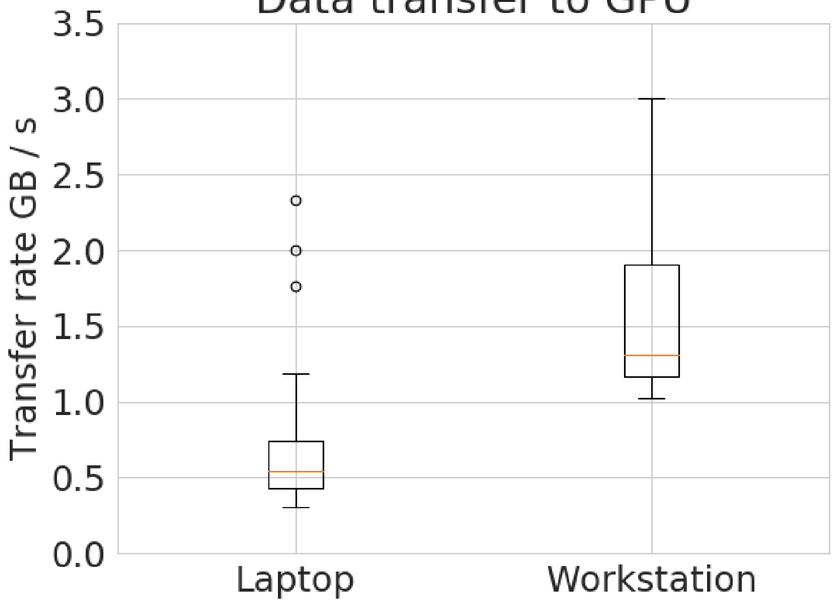
PoL Physics of Life TU Dresden



Data transfer takes time

BUT: Push() and Pull() take additional time!

push()

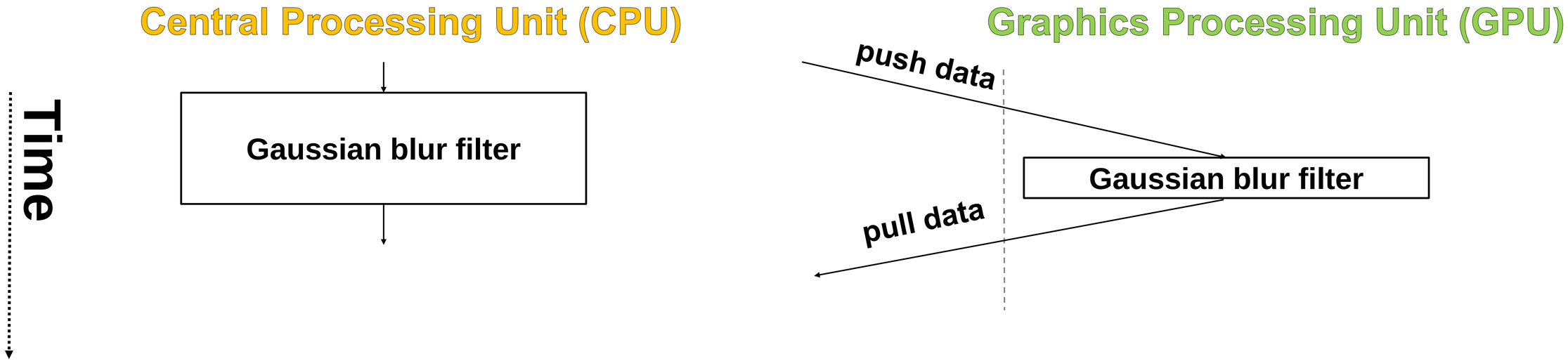


pull()

Build workflows consisting of many operations

PoL Physics of Life TU Dresden

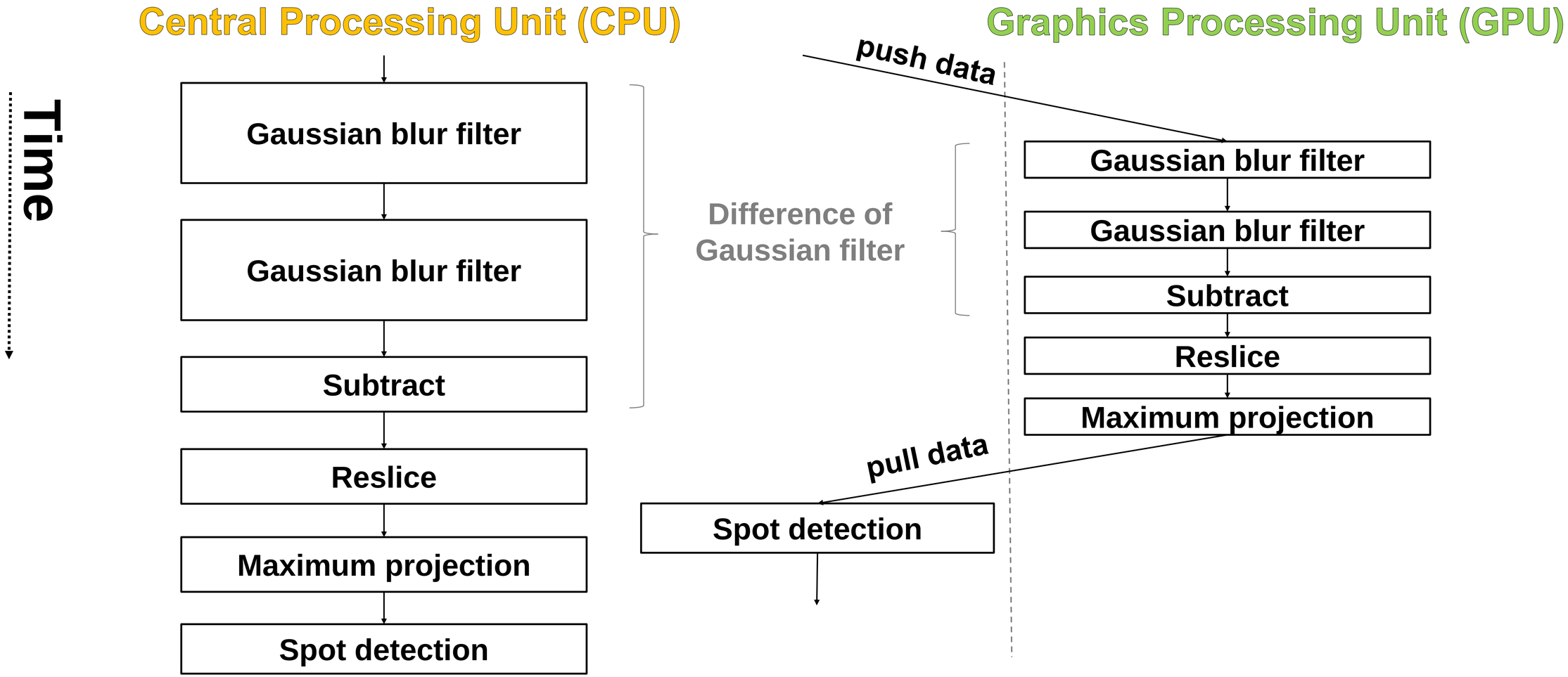
GPU acceleration may suffer from data transfer between CPU and GPU



Build workflows consisting of many operations

PoL Physics of Life TU Dresden

GPU acceleration may suffer from data transfer between CPU and GPU

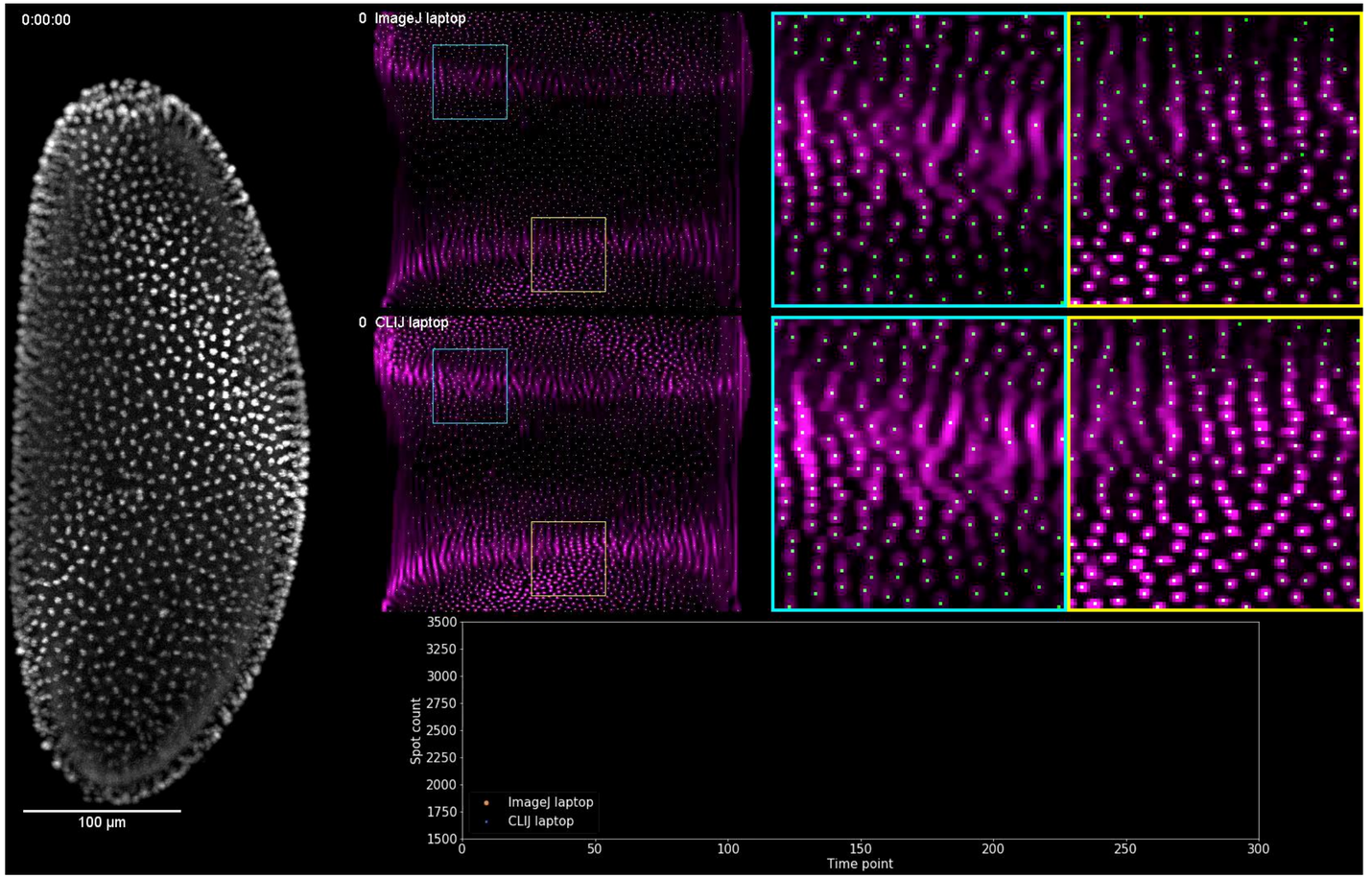


Practical use-case: Cell counting

PoL Physics of Life TU Dresden

Counting spots in 300 frames of light sheet data (including I/O)

Drosophila melanogaster, histone-RFP



ImageJ on CPU (laptop)
33 seconds per frame

2:44 h (timelapse)

ImageJ using the GPU (laptop)
2.2 seconds per frame

11 min (timelapse)

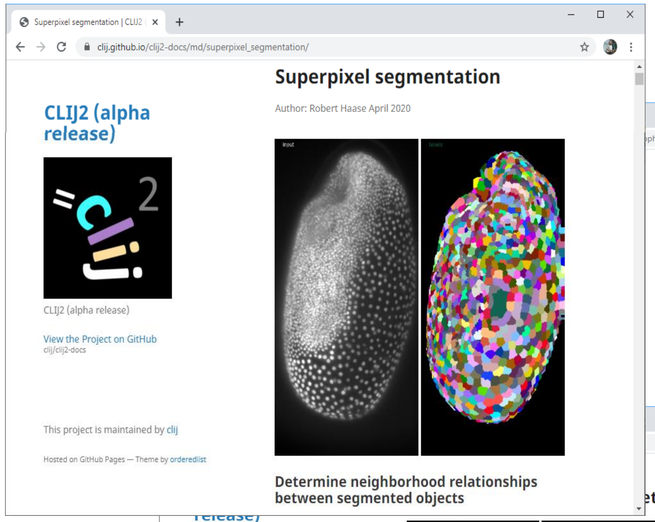
ImageJ using a dedicated GPU (workstation)
1 second per frame

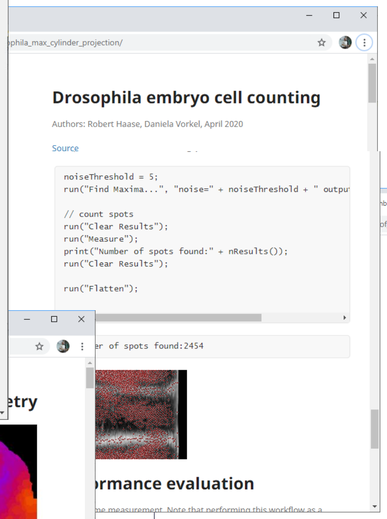
5 min (timelapse)

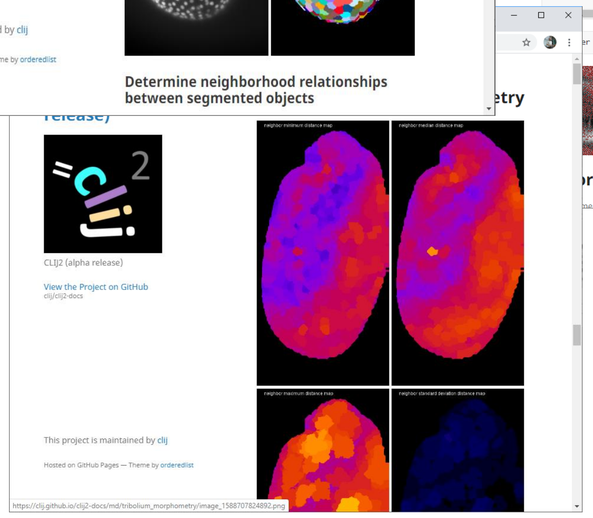
Part V: Accessibility

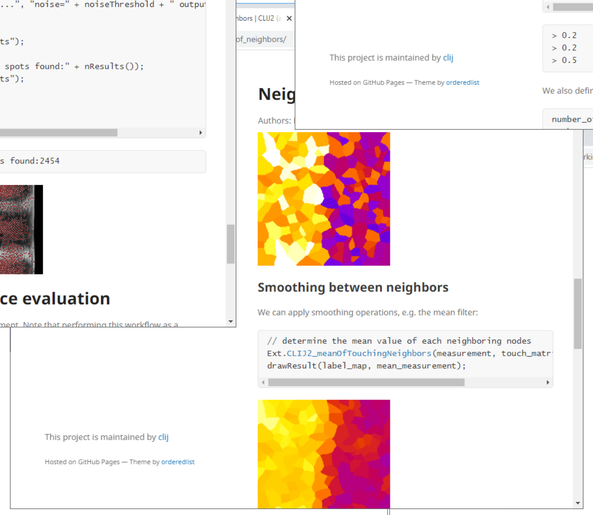
Online documentation: tutorials

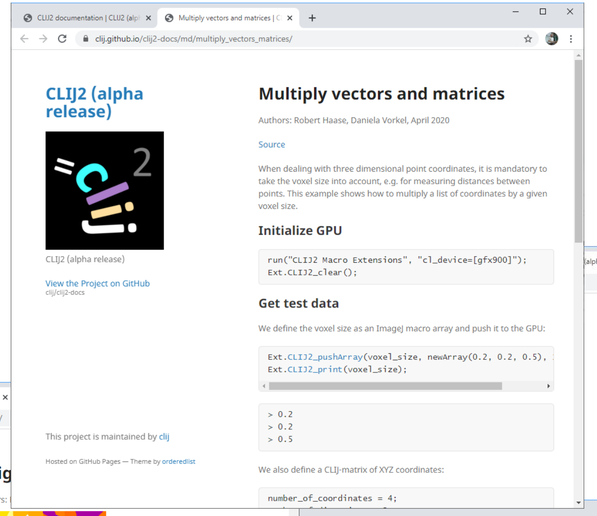
PoL Physics of Life TU Dresden

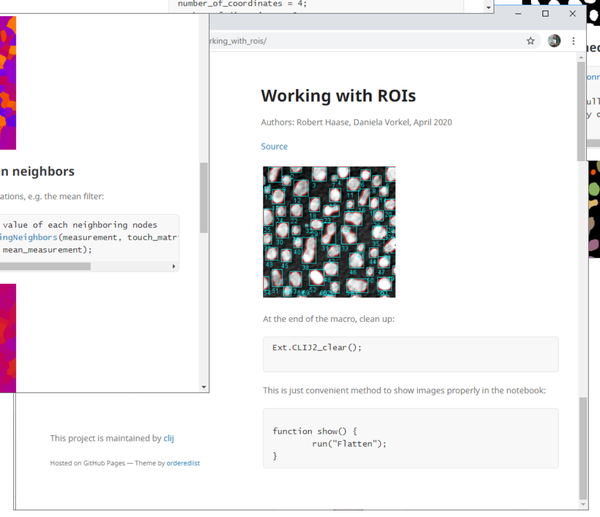


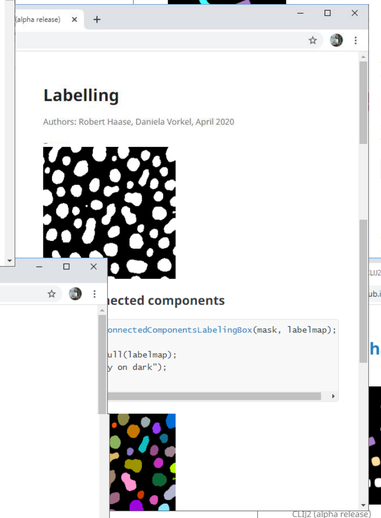


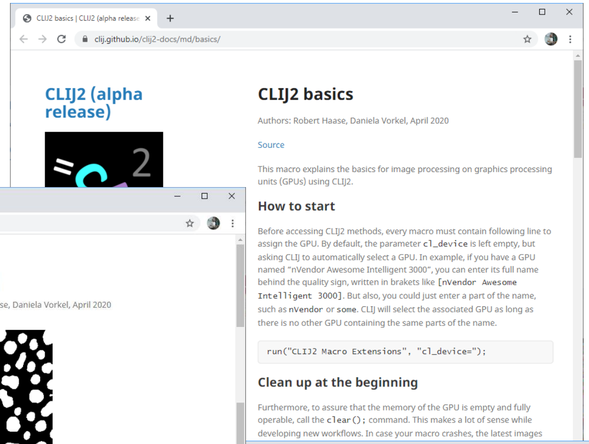


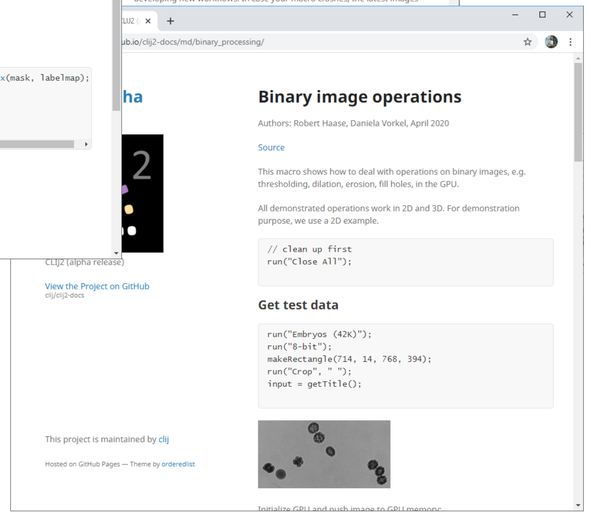












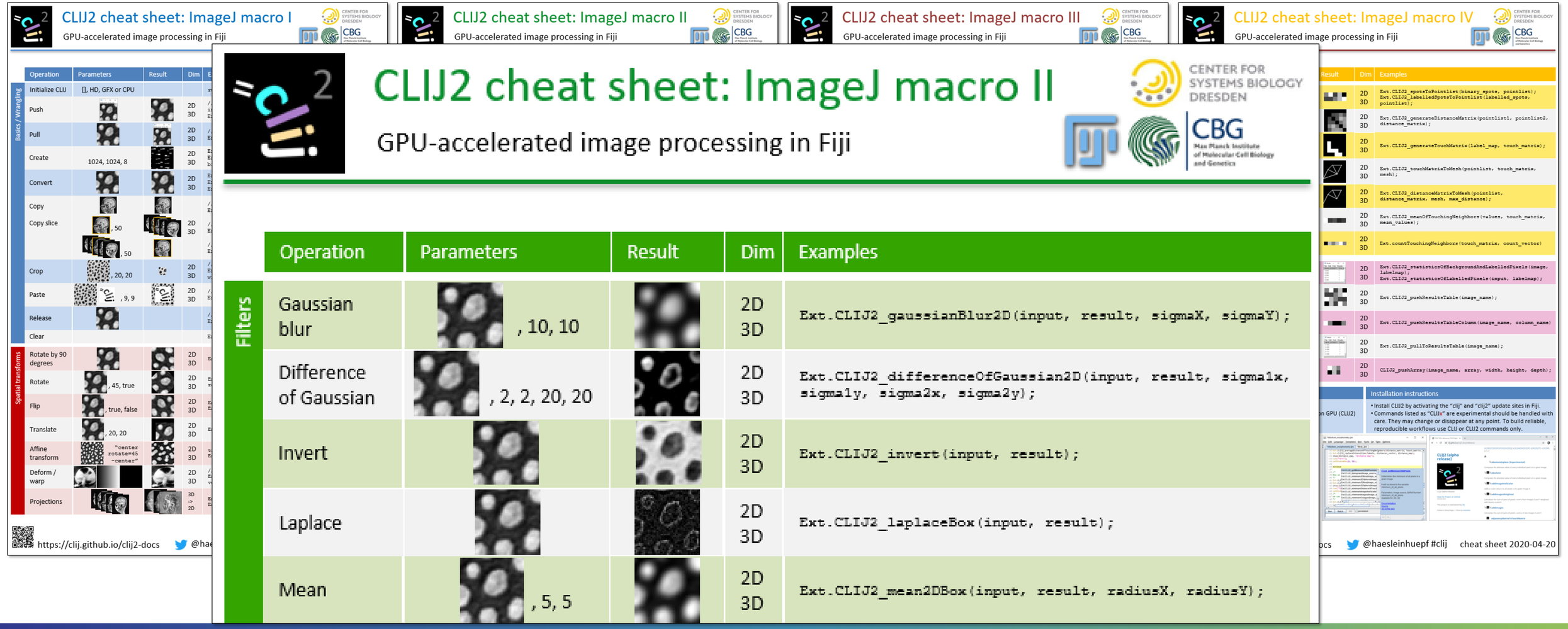
Big thanks to

Dani Vorkel (Myers lab) @happifocus

Online documentation: Cheat sheets

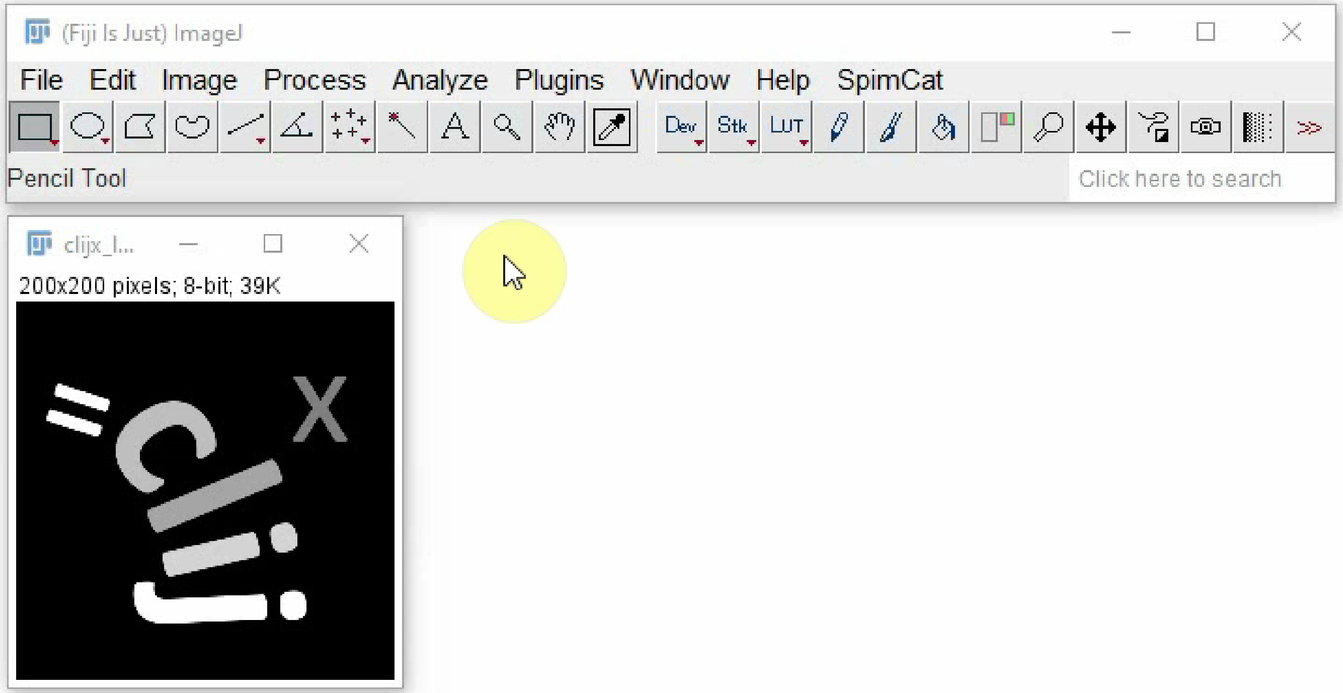
PoL Physics of Life TU Dresden

Cheat sheets show the most important methods with input and output parameters visually.



Video tutorials

For example: The Fiji plugin generator



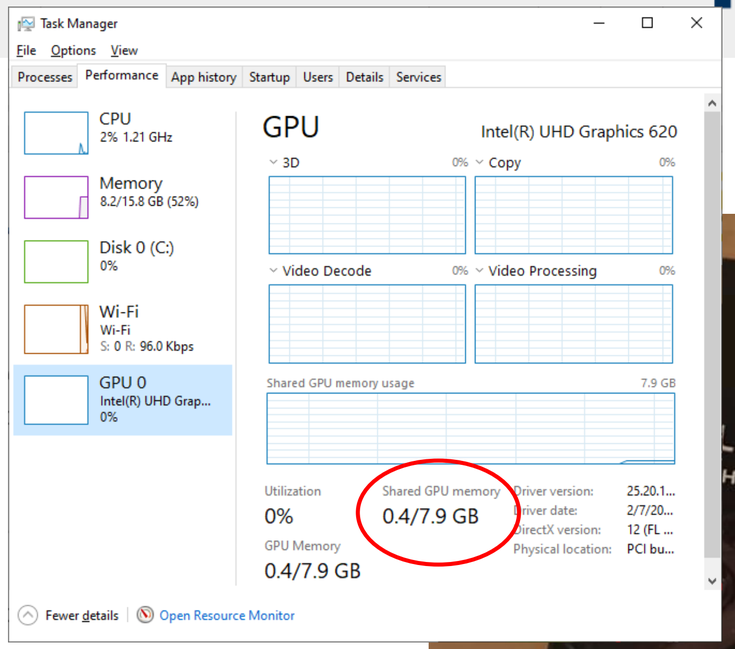
Checklist

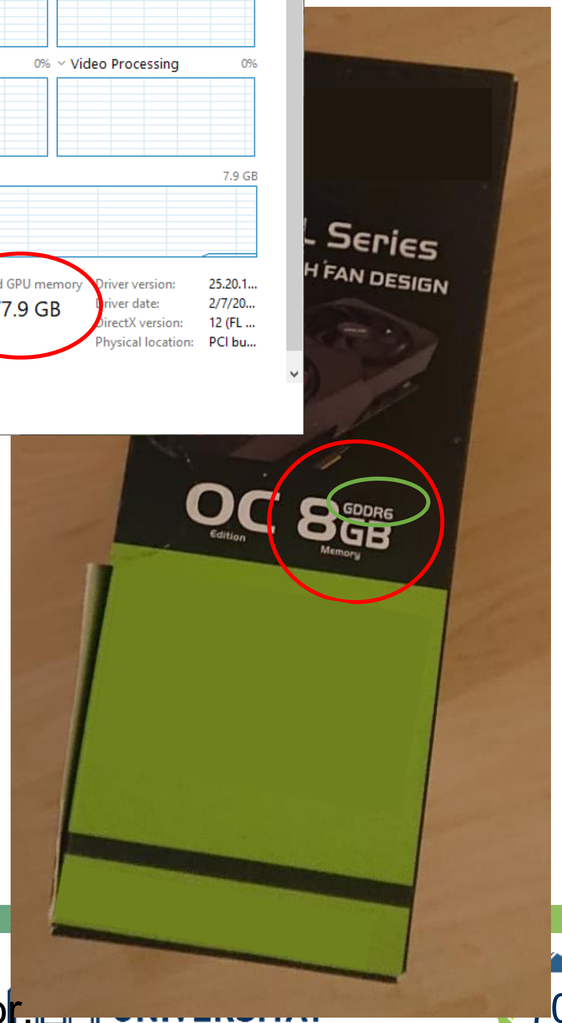
When does GPU-accelerated image processing make sense?

In order to accelerate your image analysis workflow

- you need an image analysis workflow in the first place,
- it should be slow; ideally:
  (image processing time / image loading time) > 10 to 100,
- a single processing step (rule of thumb: image size in GB x4) should fit in your graphics card memory,
- you need to re-write your workflow at least partly and
- you should have experience with ImageJ macro.

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).



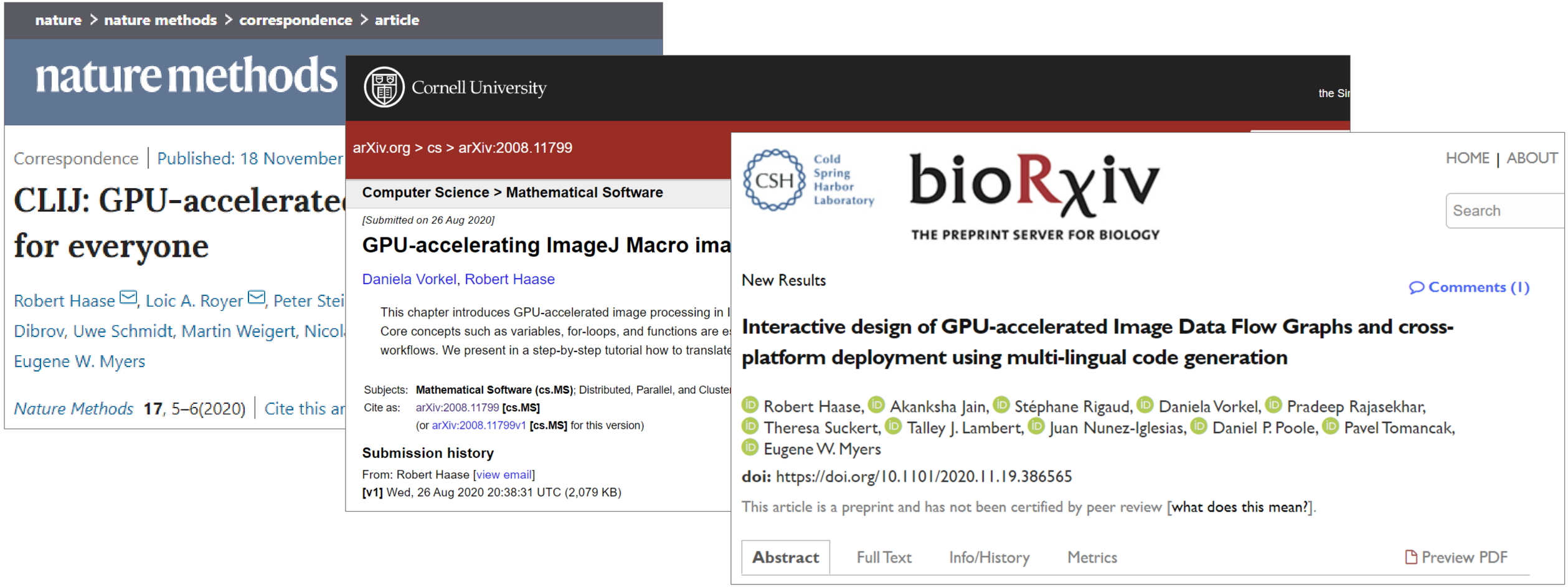


Disclosure: I don't receive any money or anything from any GPU vendor.

Last but not least: citability

PoL Physics of Life TU Dresden

- If you work with CLIJ and friends, please cite the paper(s). It'll be hard to apply for grants otherwise.



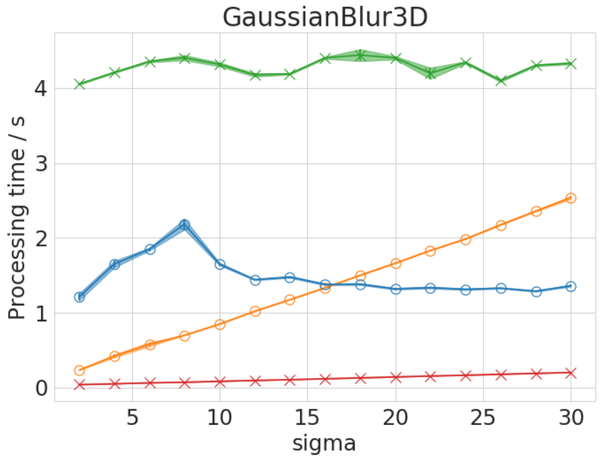
Summary

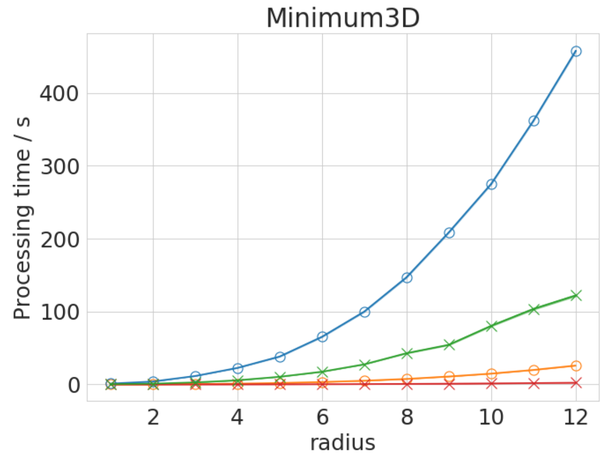
GPU-accelerated image processing works best with

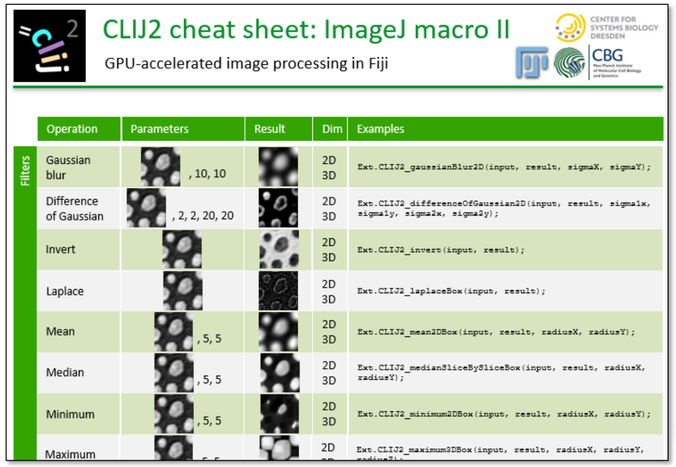
- workflows with many steps between push and pull,
- processing many images,
- on dedicated GPUs.

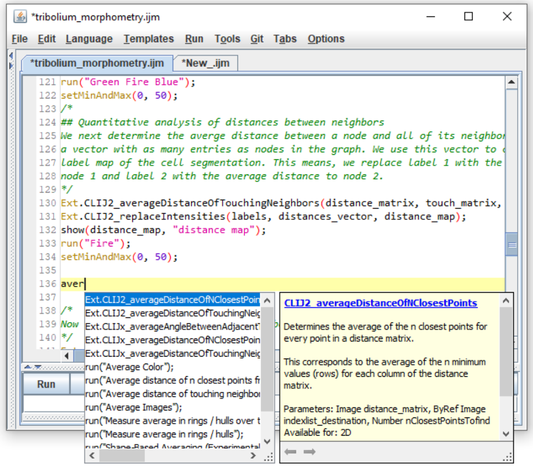
Human users like most

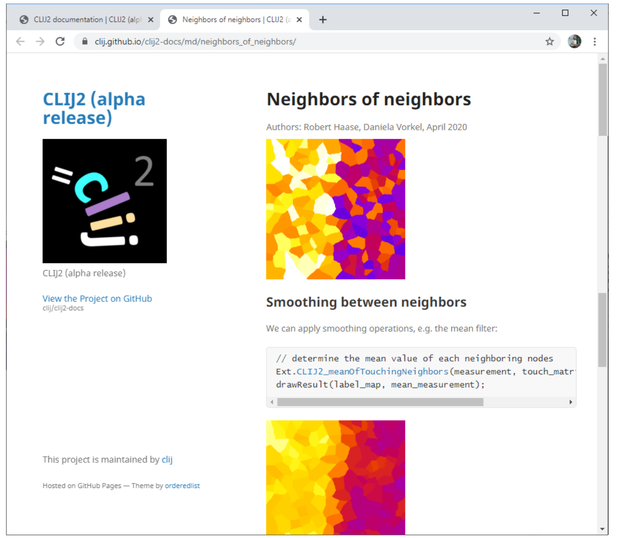
- documentation by example,
- integrated documentation and
- interactive user interfaces.

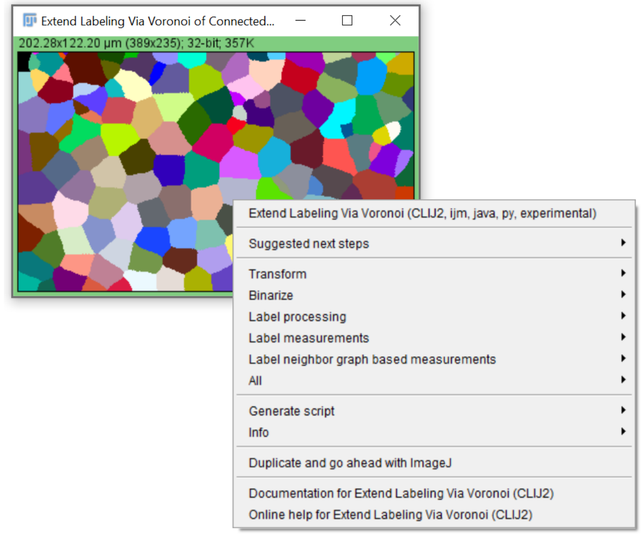












Acknowledgements

Akanksha Jain (Treutlein lab) @jain_akanksha_

Dani Vorkel (Myers lab) @happifocus

Theresa Suckert (Krause lab, Oncoray)

Pradeep Rajasekhar (Poole lab, Monash U) @pr4deepr

Talley J. Lambert (HMS) @TalleyJLambert

Juan Nunez-Iglesias (Monash) @jnuneziglesias

Pavel Tomancak (CSBD / MPI-CBG) @PavelTomancak

Gene Myers (CSBD / MPI-CBG) @TheGeneMyers

Stéphane Rigaud (Institut Pasteur) @StRigaud

Jean-Yves Tinevez (Institut Pasteur) @jytinevez

Florian Jug (CSBD / MPI-CBG) @florianjug

Szabolcs Horvát (Modes lab) @szhorvat

Johannes Girstmair (Tomancak lab) @jogirstmair

Brian Northon @truenorth_ia

Matthias Arzt (Myers/Jug lab)

Alex Herbert (University of Sussex)
Alexandr Dibrov (MPI CBG)
Bertrand Vernay (IGBMC, Strasbourg)
Bert Nitzsche (PoL TU Dresden)
Bradley Lowekamp (NIAID Washington)
Bram van den Broek (Netherlands Cancer Institute)
Brenton Cavanagh (RCSI)
Bruno C. Vellutini (MPI CBG)
Carl D. Modes (CSBD/MPI-CBG)
Curtis Rueden (UW-Madison LOCI)
Damir Krunic (DKFZ)
Daniel J. White (GE)
Deborah Schmidt (CSBD/MPI CBG)
Douglas P. Shepard (Univ Arizona)
Eduardo Conde-Sousa (Univ. of Porto)
Erick Ratamero (The Jackson Laboratory)
Gaby G. Martins (IGC)
Gayathri Nadar (MPI CBG)
Guillaume Witz (Bern University)
Giovanni Cardone (MPI Biochem)
Irene Seijo Barandiaran (MPI CBG Dresden)
Jan Brocher (Biovoxxel)
Juergen Gluch (Fraunhofer IKTS)
Kisha Sivanathan (Harvard Medical School)
Kota Miura
Laurent Thomas (Acquifer)
Lior Pytowski (University of Oxford)
Loic A. Royer (CZ Biohub)
Marion Louveaux (Institut Pasteur Paris)
Martin Weigert (EPFL Lausanne)
Matthew Foley (University of Sydney)
Nico Stuurman (UCSF)
Nicola Maghelli (MPI CBG)
Nicolas Brouilly (IBDM Marseille)
Nik Cordes (Los Alamos National Laboratory)
Noreen Walker (MPI CBG Dresden)
Ofra Golani (Weizmann Institute of Science)
Patrick Dummer
Peter Haub
Pete Bankhead (University of Edinburgh)
Peter Steinbach (HZDR Dresden)
Pit Kludig
Rita Fernandes (University of Porto)
Romain Guiet (BIOP-EPFL)
Ruth Whelan-Jeans
Sebastian Munck (VIB Leuven)
Siân Culley (MRC LMCB)
Stein Rørvik
Stéphane Dallongeville (Pasteur)
Tanner Fadero (UNC-Chapel Hill)
Thomas Irmer (Zeiss)
Tobias Pietzsch (MPI-CBG)
Uwe Schmidt (MPI CBG)
Wilson Adams (VU Biophotonics)

https://fiji.sc

https://napari.dev

https://image.sc

http://eubias.org/NEUBIAS/

MPI CBG Core Facilities

- Advanced Imaging Facility
- Light Microscopy Facility
- Scientific Computing
- IT Department
- Fly lab

Funding

DFG public

Federal Ministry of Education and Research

R.H. was supported by the German Federal Ministry of Research and Education (BMBF) under the code 031L0044 (Sysbio II) and by the Deutsche Forschungsgemeinschaft under Germany's Excellence Strategy – EXC2068 – Cluster of Excellence Physics of Life of TU Dresden.

@haesleinhuepf
@PoLDresden

CENTER FOR SYSTEMS BIOLOGY DRESDEN

CBG Max Planck Institute of Molecular Cell Biology and Genetics

https://clij.github.io/

TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept