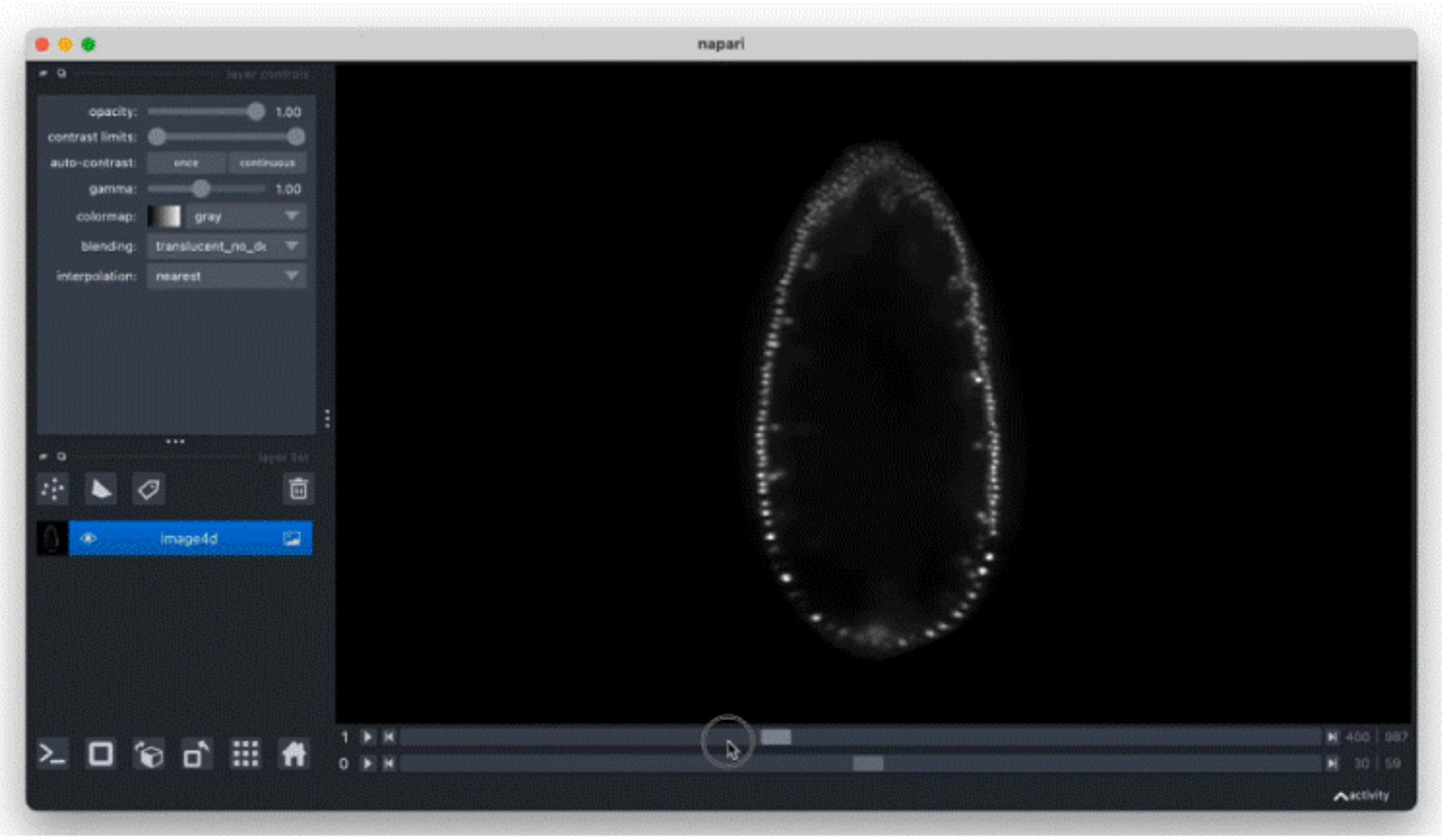
3D visualization and processing with Napari

Thierry Pécot
Research Engineer
Biosit SFR UMS CNRS 3480 – Inserm 018
CZI Imaging Scientist

search Ctrl K

napari: a fast, interactive viewer for multi-dimensional images in Python

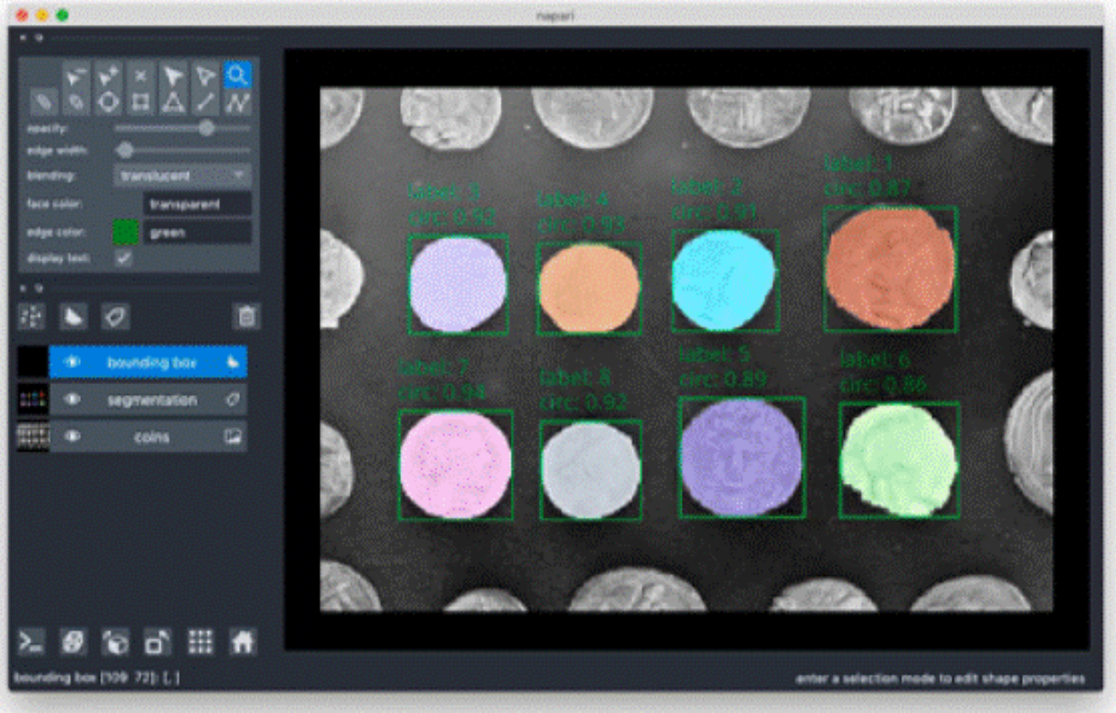


search Ctrl K

- Getting started
- napari tutorials
  - Annotation
  - Processing
  - Segmentation
    - Annotating segmentation with text and bounding boxes
  - Tracking
- How-to guides
- In-depth explanations
- Glossary
- napari workshops
- Sample databases
- Gallery

Annotating segmentation with text and bounding boxes

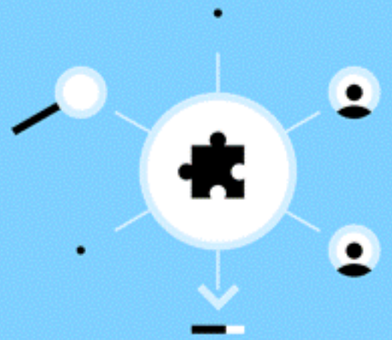
In this tutorial, we will use napari to view and annotate a segmentation with bounding boxes and text labels. Here we perform a segmentation by setting an intensity threshold with Otsu's method, but this same approach could also be used to visualize the results of other image processing algorithms such as object detection with neural networks.



napari **hub**

Plugins

Discover, install, and share napari plugins



- Discover plugins that solve your image analysis challenges
- Learn how to install into napari

Share your image analysis tools with napari's growing community

Discover Plugins Browse all

Search for a plugin by keyword or author

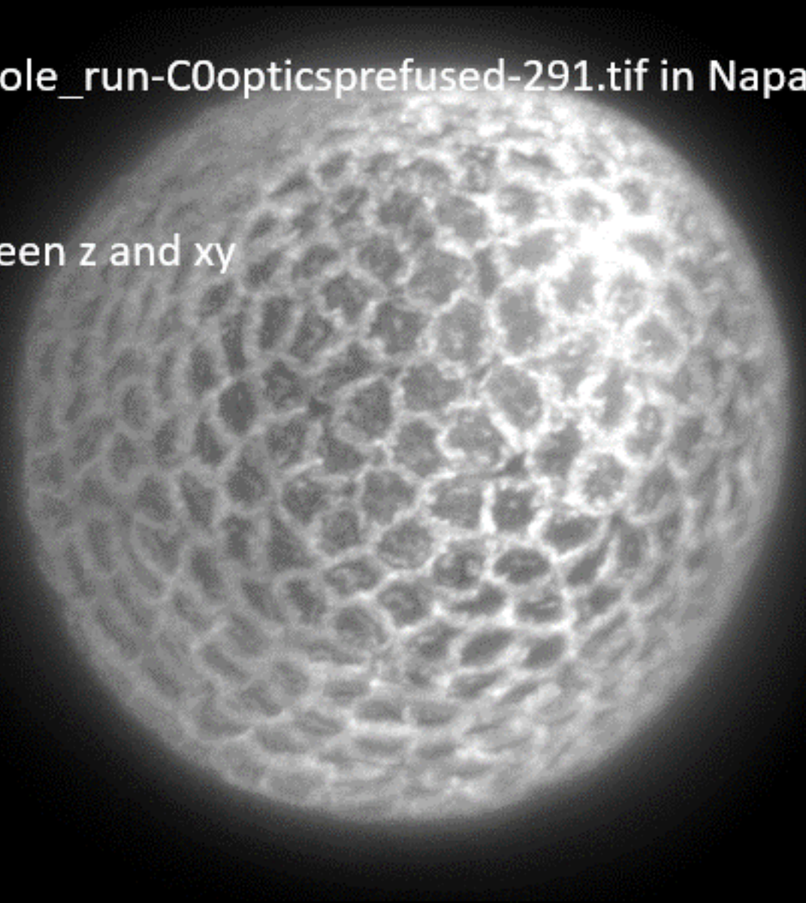
In an **Anaconda prompt**:

- conda create -n NapariEnv python=3.10 // create a virtual environment called NapariEnv
- conda activate NapariEnv // activate the virtual environment NapariEnv
- pip install napari[all] // install Napari
- Napari // open Napari

In an **Anaconda prompt**:

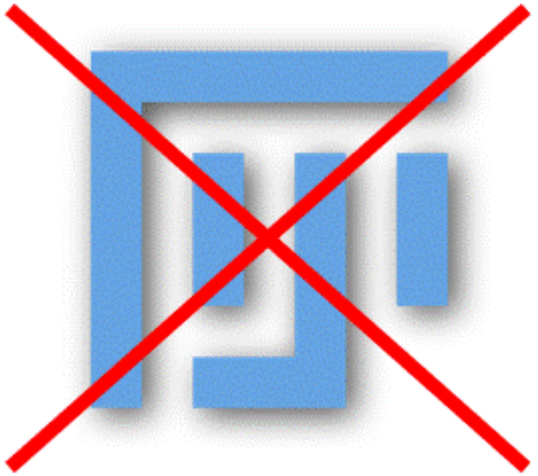
- conda create -n NapariEnv python=3.10 // create a virtual environment called NapariEnv
- conda activate NapariEnv // activate the virtual environment NapariEnv
- pip install napari[all] // install Napari
- Napari // open Napari

- Drag and drop Strausberg_Tribolium_LA-GFP_tailpole_run-C0opticsprefused-291.tif in Napari (available at https://zenodo.org/records/3981193)
- Visualize image in 2D and 3D, change scaling between z and xy
- Change contrast, change colormap, blending, ...
- Add Points layer
- Add shape layer
- Add a label layer, annotate 2 cells



Visualization

Visualization



- The **whole image** is loaded into the computer's **RAM** memory when opened with Fiji
- Stacks acquired with **lightsheet** microscopes = tens of GB

Visualization

- Open as virtual stack
- Save and visualize it with BigDataViewer

Visualization

- Open as virtual stack
- Save and visualize it with BigDataViewer

- Zarr format **splits data into chunks** so only **one or several chunks** are loaded **at once** into the computer's **RAM** memory
- **Napari**, a Python visualizer, allows to **visualize zarr data**

In an **Anaconda prompt**:

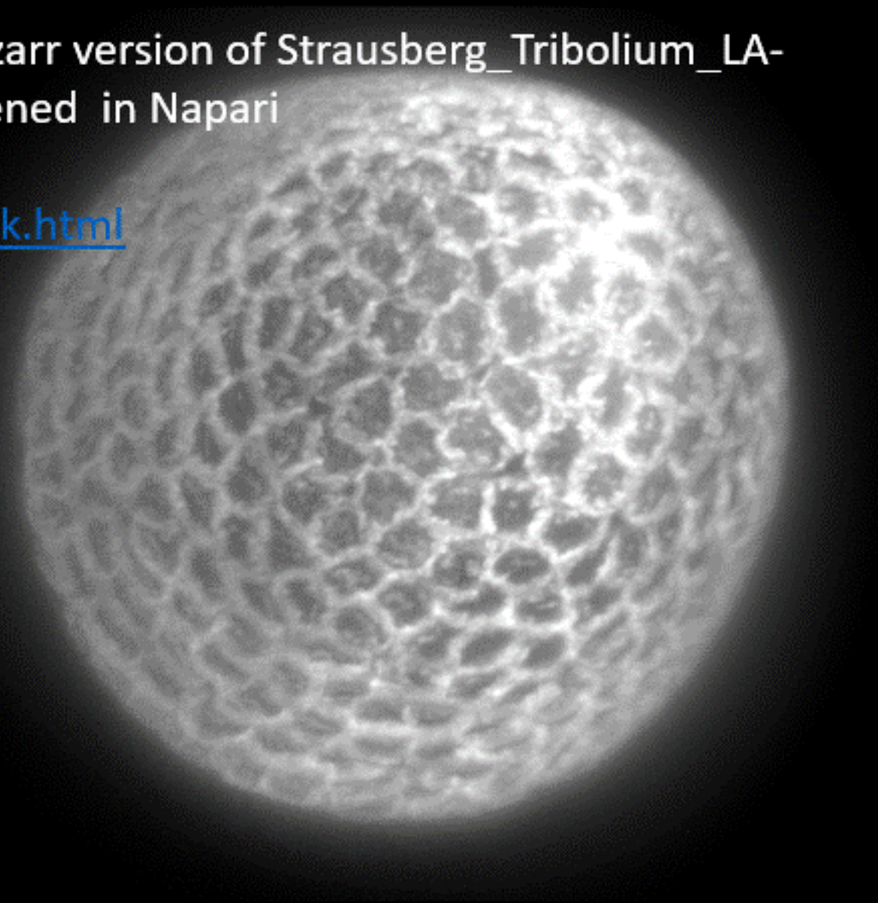
- conda install -c ome bioformats2raw // install Python package bioformats2raw
- bioformats2raw Strausberg_Tribolium_LA-GFP_tailpole_run-C0opticsprefused-291.tif Strausberg_Tribolium_LA-GFP_tailpole_run-C0opticsprefused-291.zarr // convert to zarr
- pip install napari-ome-zarr // install Napari plugin to open zarr images

In an **Anaconda prompt**:

- conda install -c ome bioformats2raw // install Python package bioformats2raw
- bioformats2raw Strausberg_Tribolium_LA-GFP_tailpole_run-C0opticsprefused-291.tif Strausberg_Tribolium_LA-GFP_tailpole_run-C0opticsprefused-291.zarr // convert to zarr
- pip install napari-ome-zarr // install Napari plugin to open zarr images

- Compare the RAM usage between the tif and the zarr version of Strausberg_Tribolium_LA-GFP_tailpole_run-C0opticsprefused-291 when opened in Napari

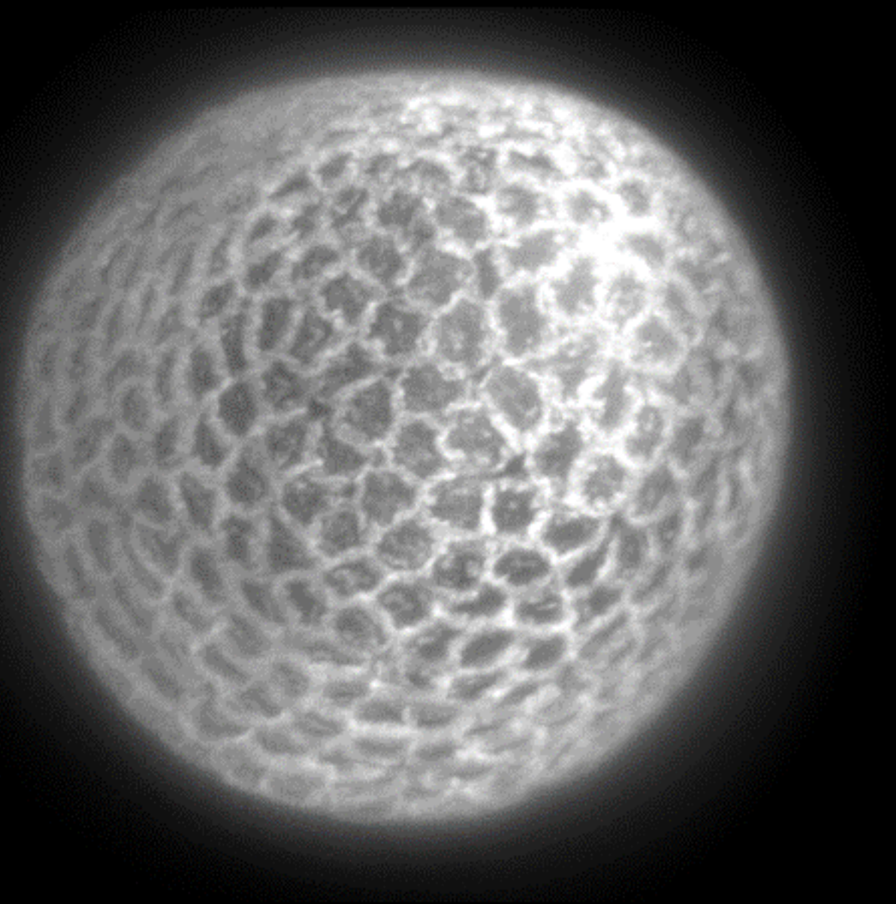
- https://napari.org/stable/tutorials/processing/dask.html



In an **Anaconda prompt**:

- conda install -c conda-forge openjdk

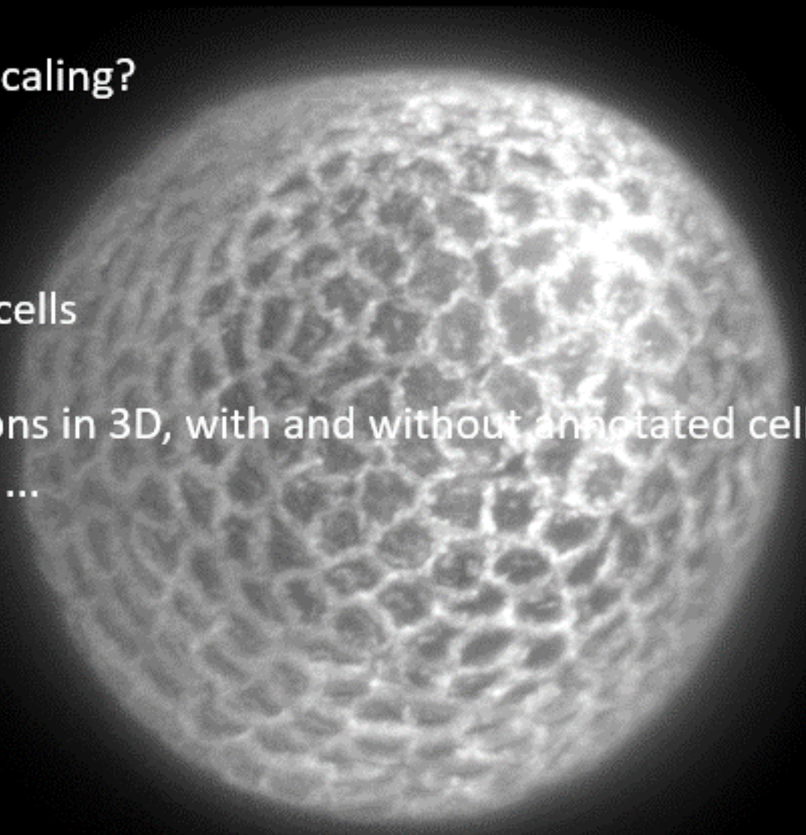
- pip install napari-bioformats napari-label-interpolator napari-skimage-regionprops napari-animation // install Napari plugin to open proprietary formats, to interpolate labels, to do measurements from regions and to create videos



In an **Anaconda prompt**:

- conda install -c conda-forge openjdk
- pip install napari-bioformats napari-label-interpolator napari-skimage-regionprops napari-animation // install Napari plugin to open proprietary formats, to interpolate labels, to do measurements from regions and to create videos

- Visualize image reconstruction in 3D, what about scaling?
- Annotate cells with the label interpolator plugin
- Extract measurements associated with annotated cells
- Create an animation with sections in 2D and sections in 3D, with and without annotated cells, with and without image, zooming in, zooming out, rotating, ...



Cell Detection with Star-convex Polygons

Uwe Schmidt[1,*], Martin Weigert[1,*], Coleman Broaddus[1], and Gene Myers[1,2]

[1] Max Planck Institute of Molecular Cell Biology and Genetics, Dresden, Germany
Center for Systems Biology Dresden, Germany
[2] Faculty of Computer Science, Technical University Dresden, Germany

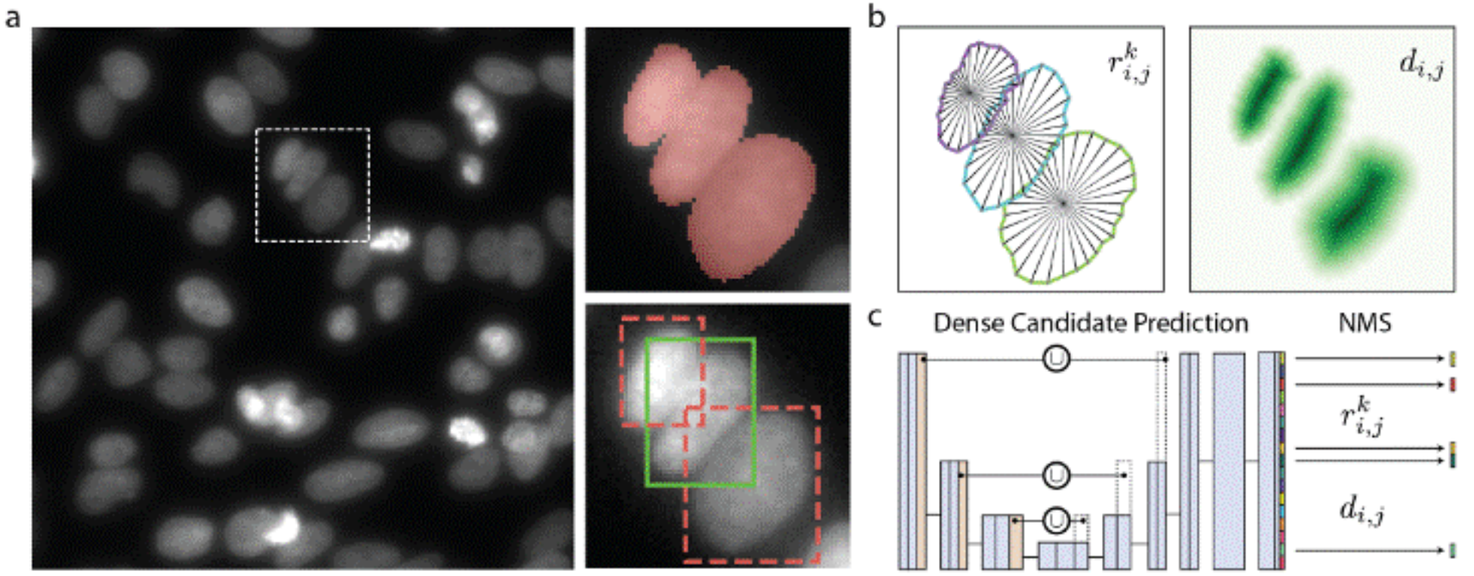


Fig. 1: *(a)* Potential segmentation errors for images with crowded nuclei: Merging of touching cells (upper right) or suppression of valid cell instances due to large overlap of bounding box localization (lower right). *(b)* The proposed StarDist method predicts object probabilities $d_{i,j}$ and star-convex polygons parameterized by the radial distances $r^k_{i,j}$. *(c)* We densely predict $r^k_{i,j}$ and $d_{i,j}$ using a simple U-Net architecture [15] and then select the final instances via non-maximum suppression (NMS).

Cell Detection with Star-convex Polygons

Uwe Schmidt[1,*], Martin Weigert[1,*], Coleman Broaddus[1], and Gene Myers[1,2]

[1] Max Planck Institute of Molecular Cell Biology and Genetics, Dresden, Germany
Center for Systems Biology Dresden, Germany
[2] Faculty of Computer Science, Technical University Dresden, Germany

For the workshop, a Stardist model was trained with data coming from 3 articles:

- **Whole-cell segmentation of tissue images with human-level performance using large-scale data annotation and deep learning.** *Nature Biotechnology (2022).*
- **A deep learning segmentation strategy that minimizes the amount of manually annotated images.** *F1000 Research (2022).*
- **Deep learning tools and modeling to estimate the temporal expression of cell cycle proteins from 2D still images.** *PLOS Computational Biology (2022).*

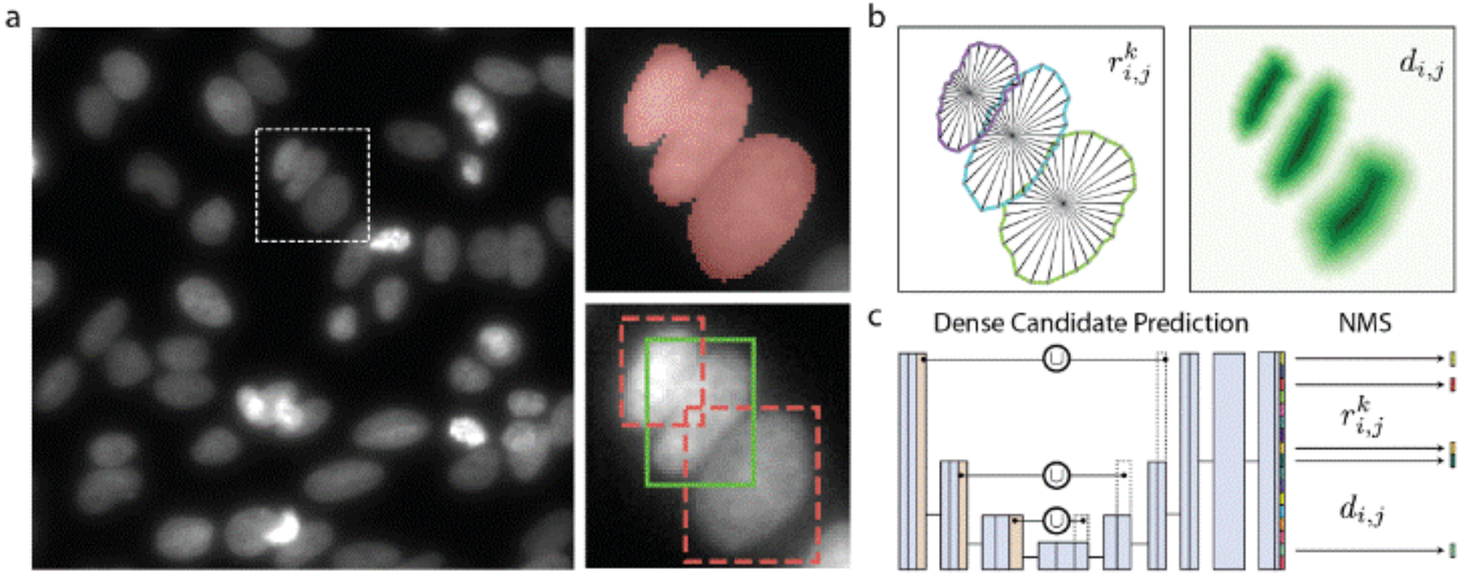
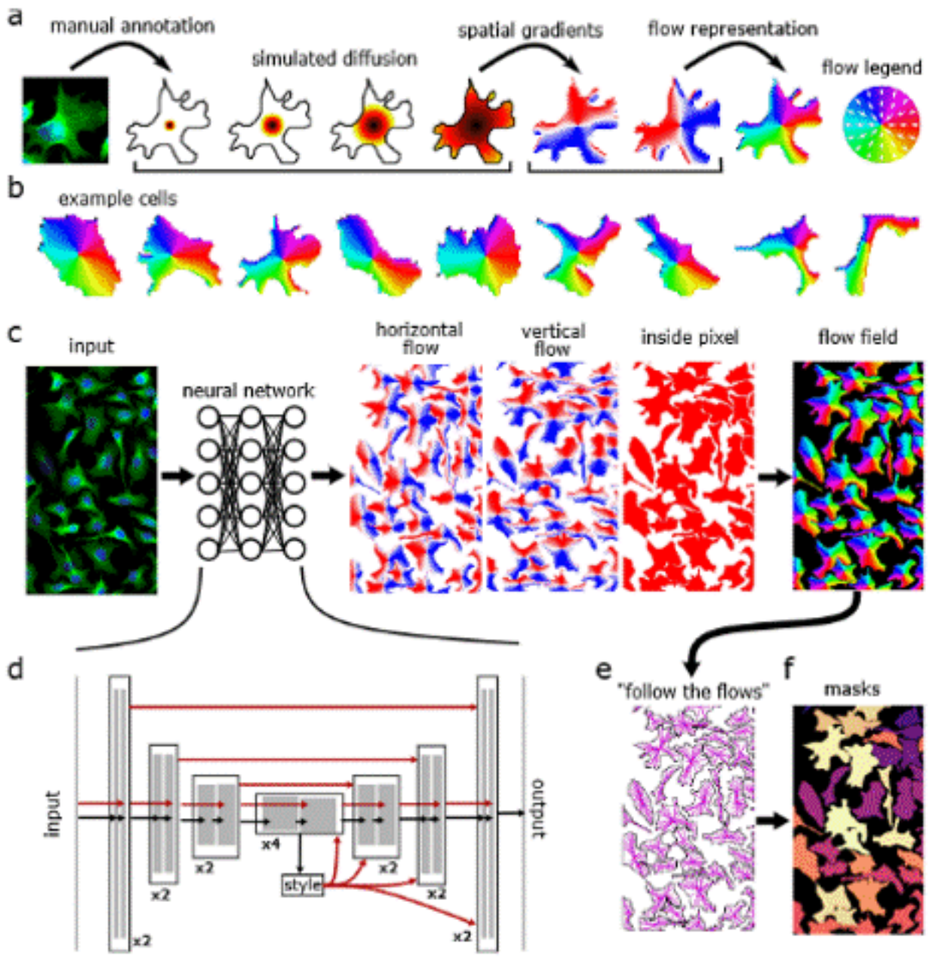
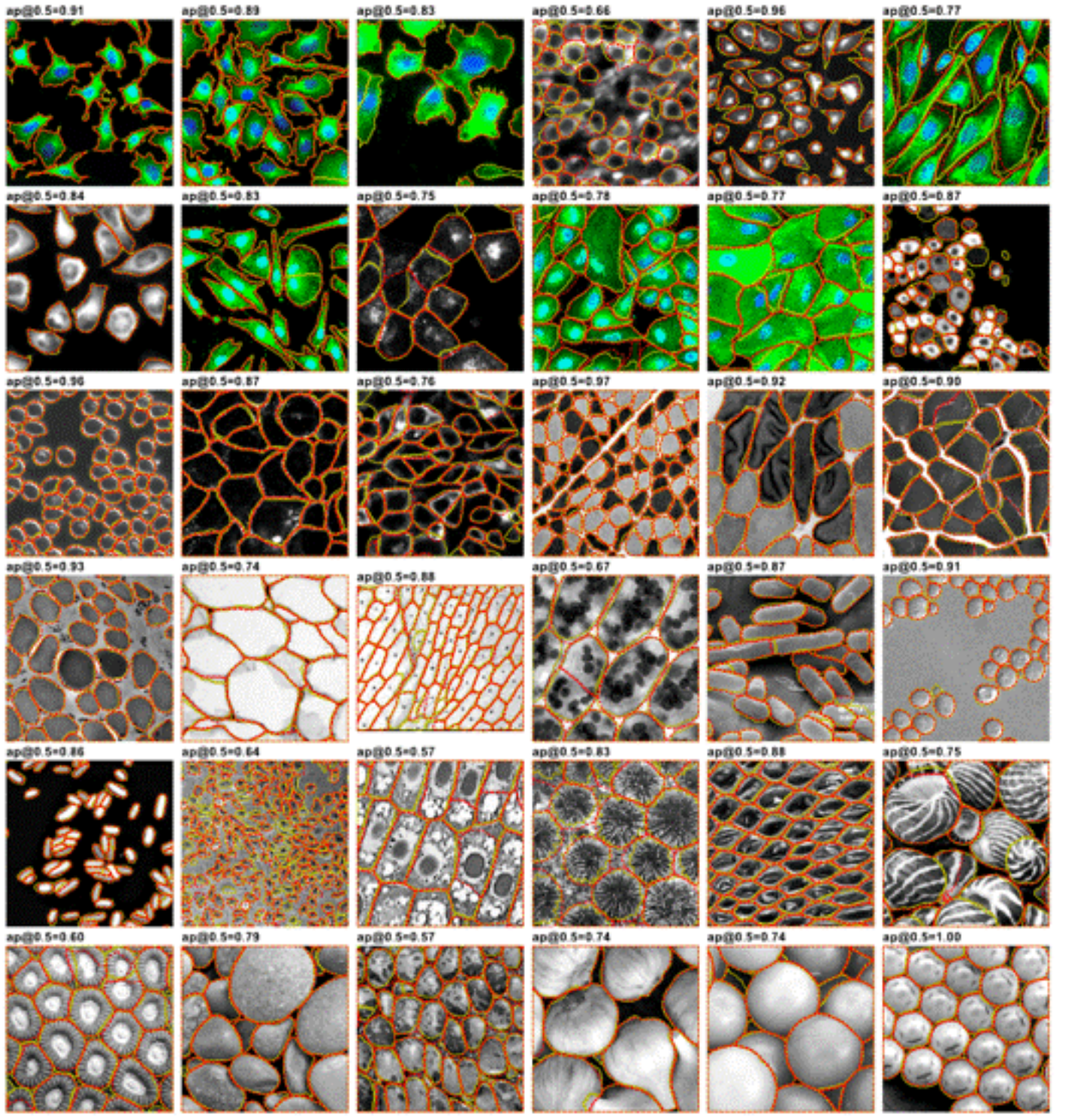


Fig. 1: *(a)* Potential segmentation errors for images with crowded nuclei: Merging of touching cells (upper right) or suppression of valid cell instances due to large overlap of bounding box localization (lower right). *(b)* The proposed StarDist method predicts object probabilities $d_{i,j}$ and star-convex polygons parameterized by the radial distances $r^k_{i,j}$. *(c)* We densely predict $r^k_{i,j}$ and $d_{i,j}$ using a simple U-Net architecture [15] and then select the final instances via non-maximum suppression (NMS).

CELLPOSE

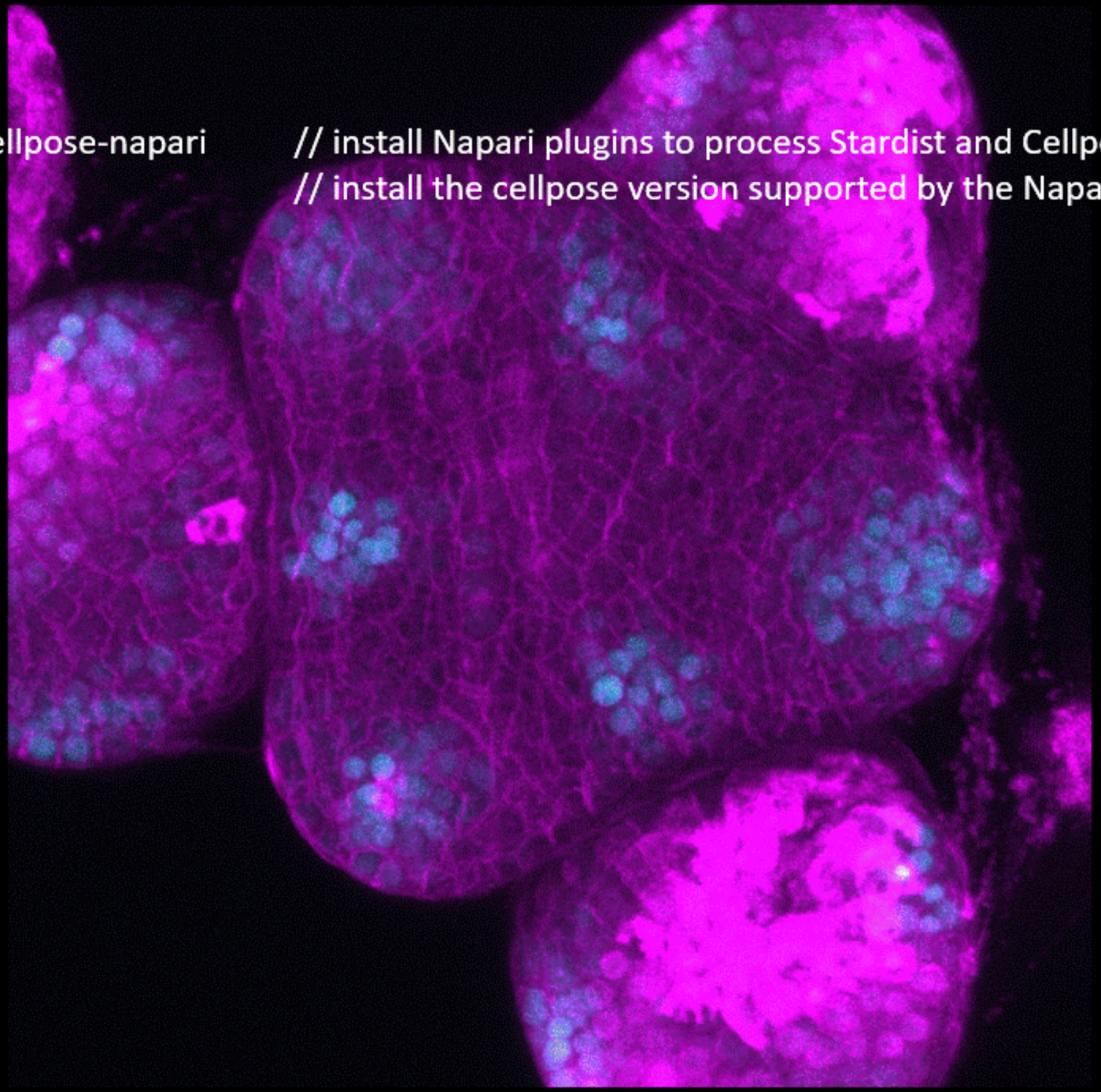




Stringer *et al.,* Cellpose: a generalist algorithm for cellular segmentation. *Nature Methods*, 2021

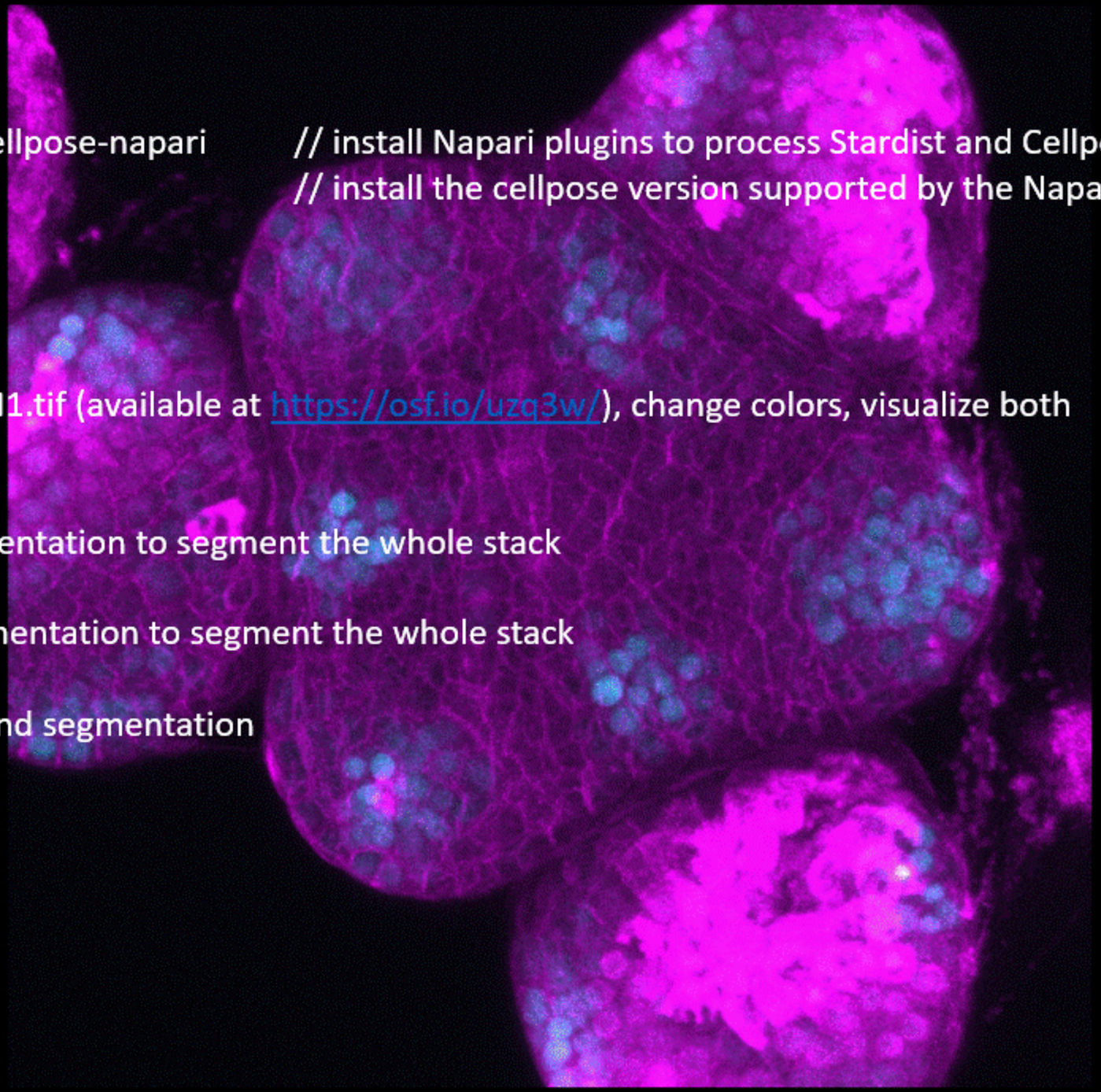
In an **Anaconda prompt**:

- pip install stardist-napari cellpose-napari // install Napari plugins to process Stardist and Cellpose
- pip install cellpose==2.0 // install the cellpose version supported by the Napari plugin



In an **Anaconda prompt**:

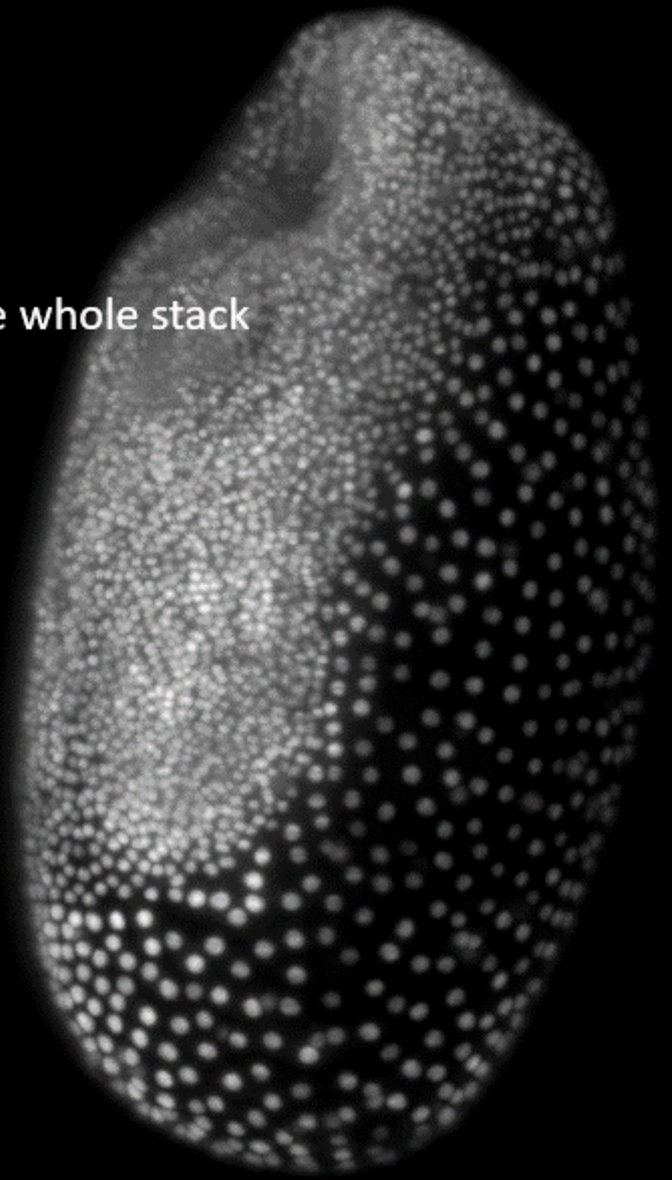
- pip install stardist-napari cellpose-napari // install Napari plugins to process Stardist and Cellpose
- pip install cellpose==2.0 // install the cellpose version supported by the Napari plugin



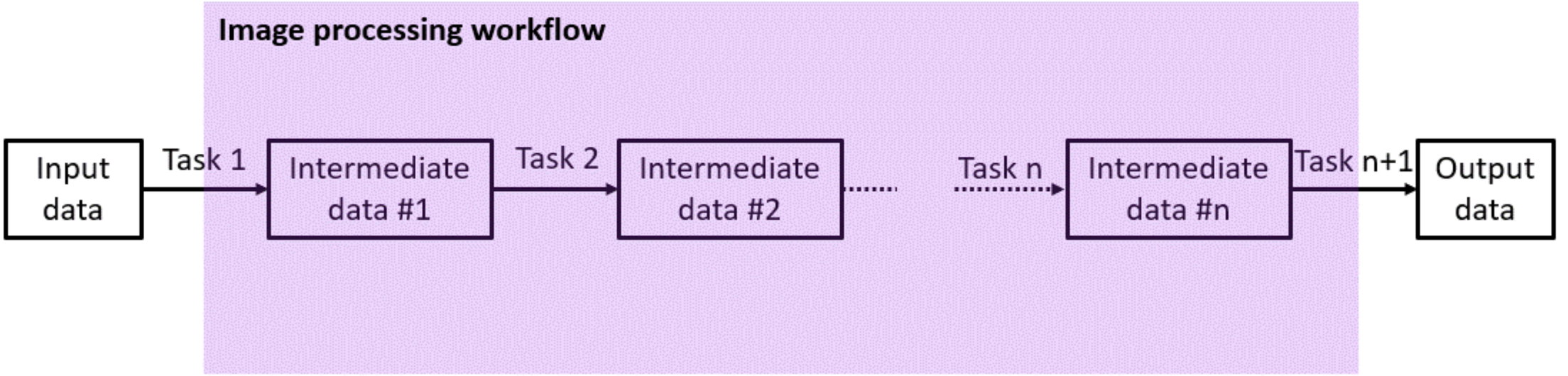
- Open MutX_DR5v2_6_4_M1.tif (available at https://osf.io/uzq3w/), change colors, visualize both channels
- Parameterize stardist segmentation to segment the whole stack
- Parameterize cellpose segmentation to segment the whole stack
- Make a video with image and segmentation

- Open lund1051_resampled.tif (available at https://git.mpi-cbg.de/rhaase/clij2_example_data/blob/master/lund1051_resampled.tif)

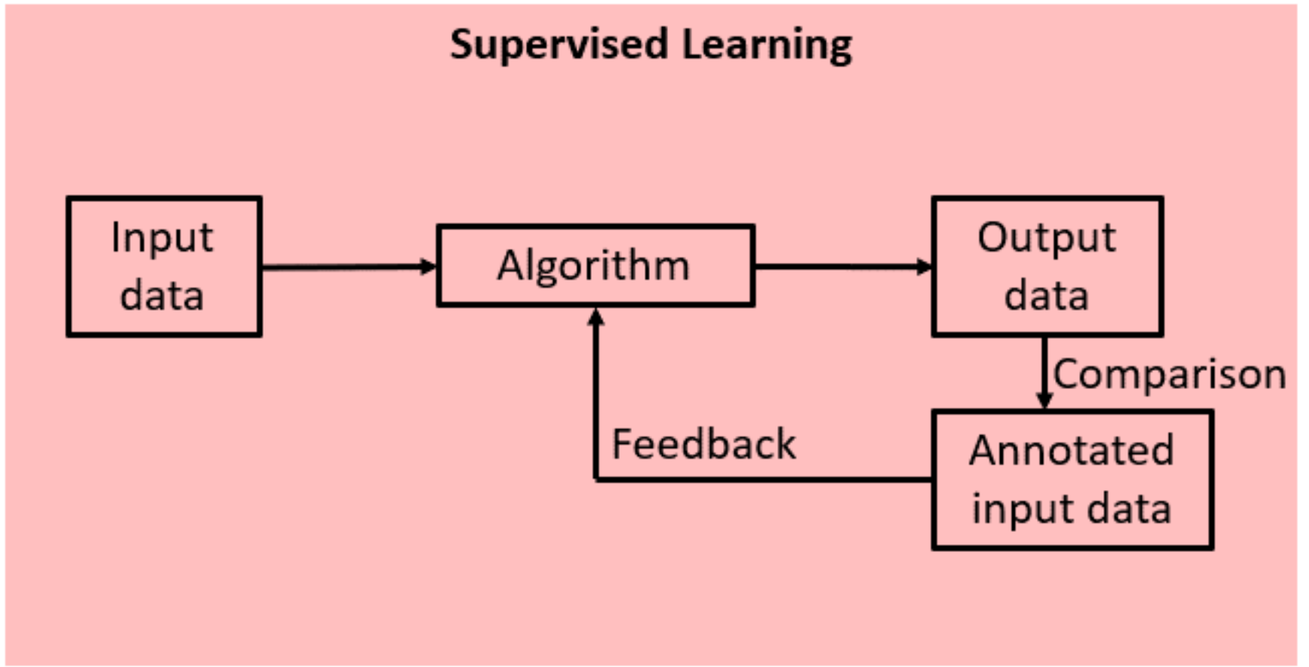
- Open lund1051_resampled_crop.tif to tune Stardist parameters and segment the whole stack



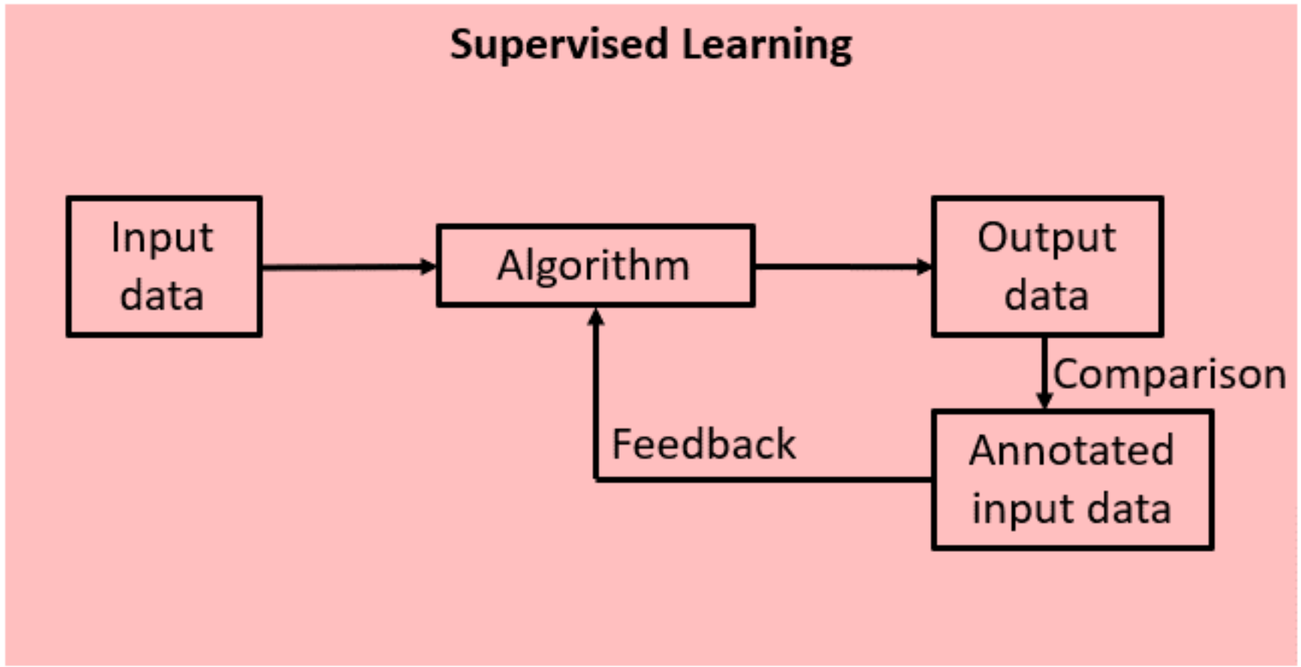
EXPLICIT PROGRAMMING

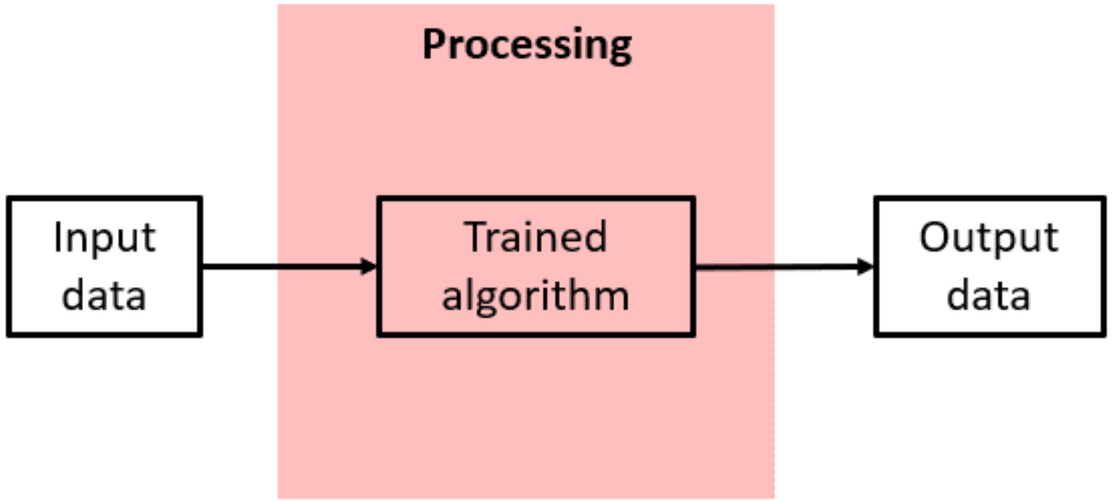


Supervised Machine Learning

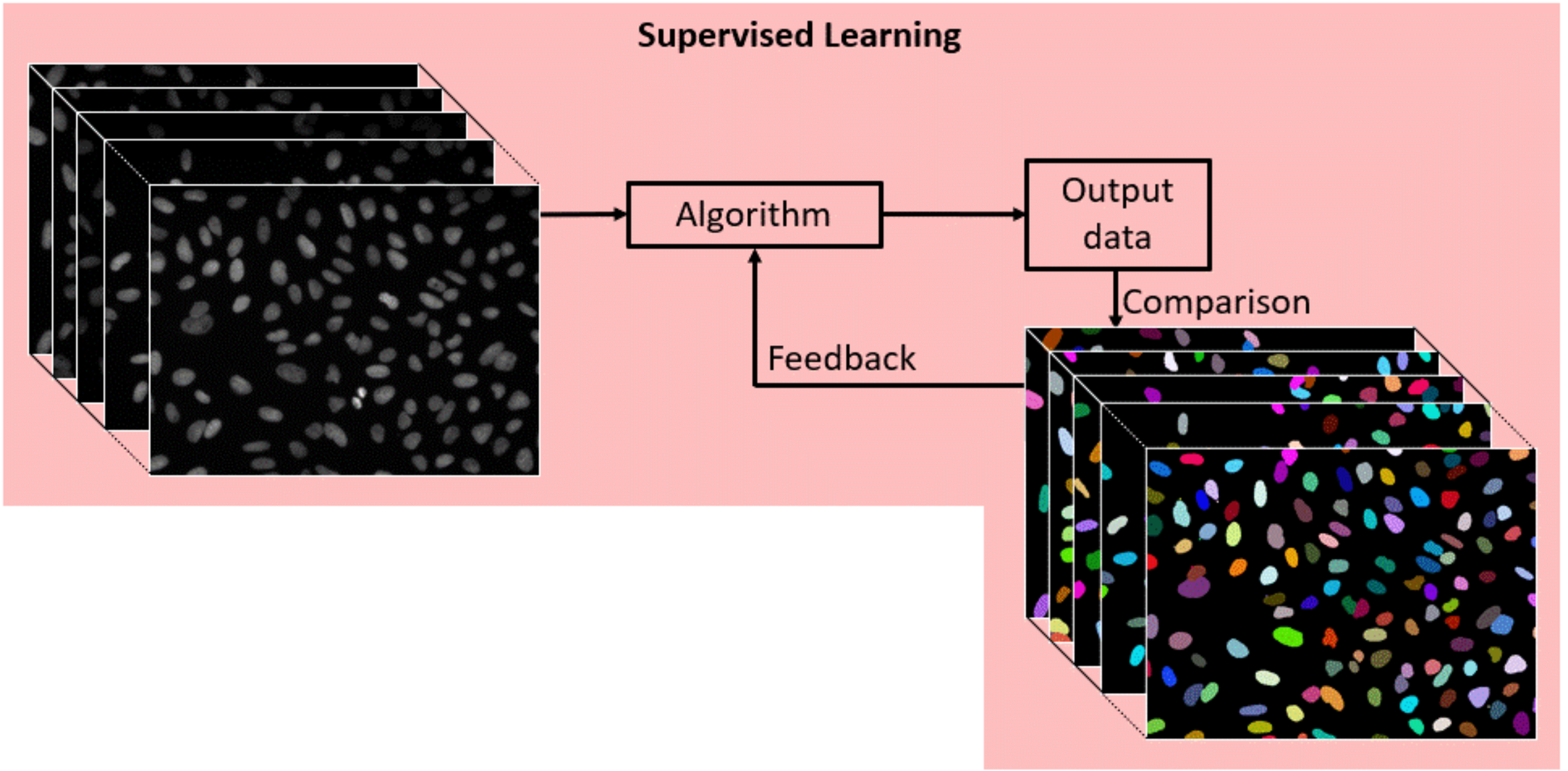


Supervised Machine Learning

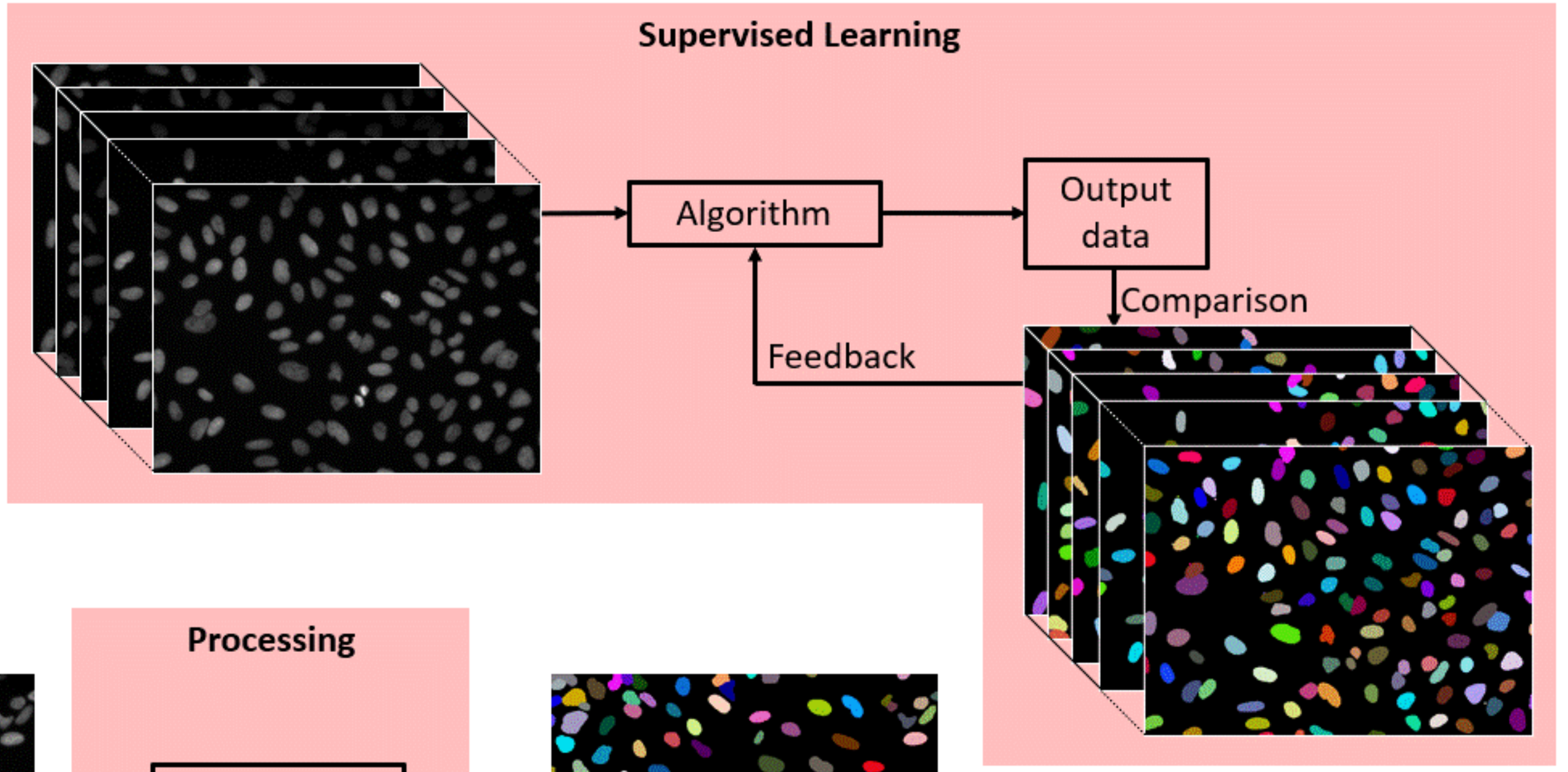


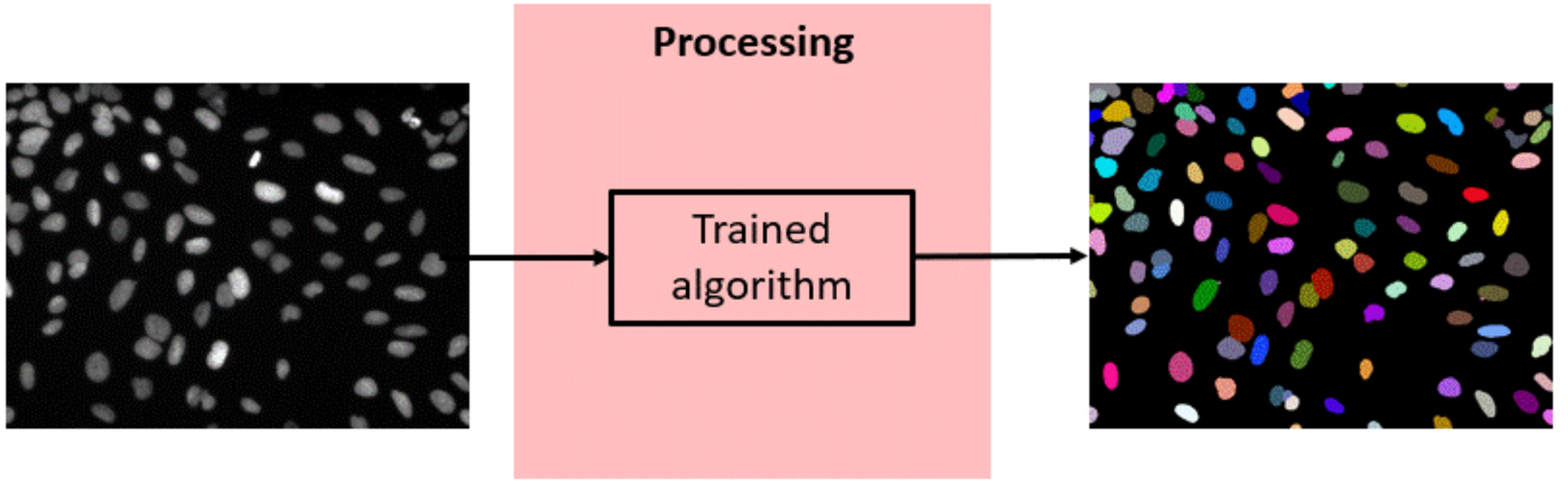


Supervised Machine Learning



Supervised Machine Learning





Shallow Machine Learning for Pixel Classification

Supervised classification:

- **Examples** of classes are **manually** defined by the user
- A **classifier** is **trained** by using **defined features** with these examples
- Data is then **automatically classified** by using the trained classifier

nature methods

Explore content ▾ About the journal ▾ Publish with us ▾ Subscribe

nature › nature methods › correspondence › article

Correspondence | Published: 18 November 2019

CLIJ: GPU-accelerated image processing for everyone

Robert Haase ✉, Loic A. Royer ✉, Peter Steinbach, Deborah Schmidt, Alexandr Dibrov, Uwe Schmidt, Martin Weigert, Nicola Maghelli, Pavel Tomancak, Florian Jug & Eugene W. Myers

Nature Methods **17**, 5–6 (2020) | Cite this article

5239 Accesses | **21** Citations | **95** Altmetric | Metrics

To the editor – Modern microscopy generates staggering amounts of multidimensional image data that place increasing demands on processing flexibility and efficiency. One way to speed up image processing is to exploit the parallel processing capabilities of graphics processing units (GPU).

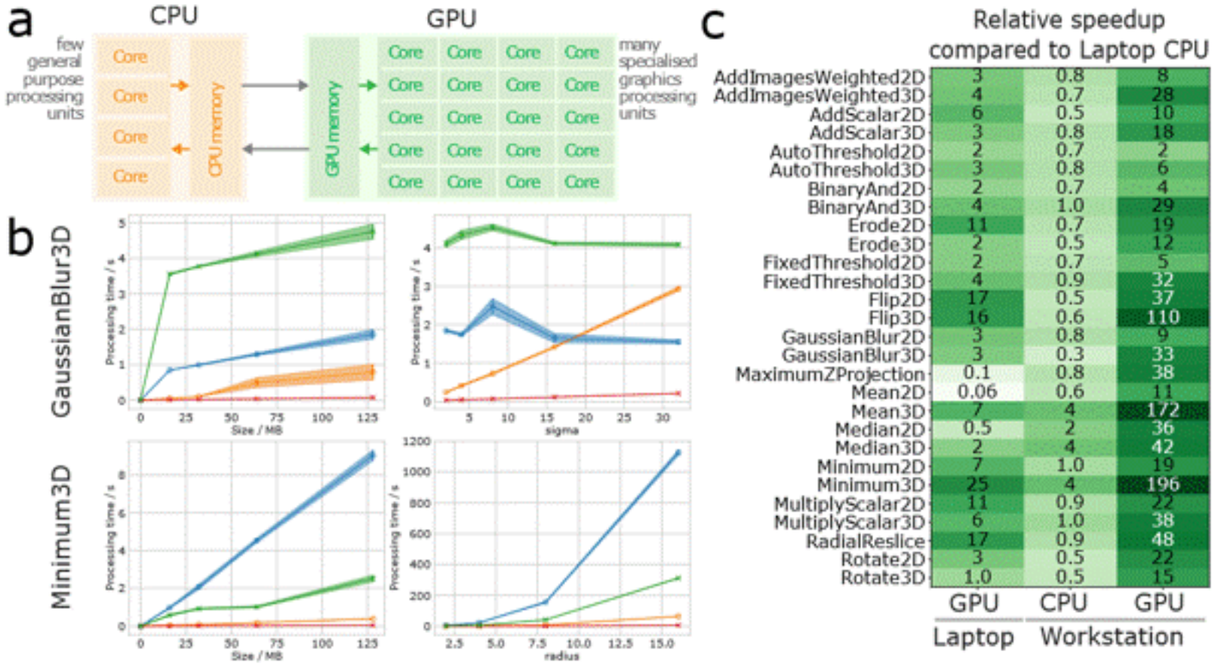


Figure 1: **(a)** GPU-acceleration schematically shows how many GPU cores potentially outperform a CPU with less cores. **(b)** Execution time of the Gaussian blur and minimum filter for different image sizes and parameters. Error bands denote the 99.9% confidence interval. **(c)** Overview of speed-up measurements when applied to 32 MB (2D) / 64 MB (3D) large 16-bit images with respect to computation times on a laptop CPU. Time measurements excluded memory transfer and compilation times.

Interactive design of GPU-accelerated Image Data Flow Graphs and cross-platform deployment using multi-lingual code generation

Robert Haase[1,2,3*], Akanksha Jain[4], Stéphane Rigaud[5], Daniela Vorkel[1,2], Pradeep Rajasekhar[6,7], Theresa Suckert[8], Talley J. Lambert[9], Juan Nunez-Iglesias[10], Daniel P. Poole[6,7], Pavel Tomancak[1], Eugene W. Myers[1,2]

Abstract Modern life science relies heavily on fluorescent microscopy and subsequent quantitative bio-image analysis. The current rise of graphics processing units (GPUs) in the context of image processing enables batch processing large amounts of image data at unprecedented speed. In order to facilitate adoption of this technology in daily practice, we present an expert system based on the GPU-accelerated image processing library CLIJ: The CLIJ-assistant keeps track of which operations formed an image and suggests subsequent operations. It enables new ways of interaction with image data and image processing operations because its underlying GPU-accelerated image data flow graphs (IDFGs) allow changes to parameters of early processing steps and instantaneous visualization of their final results. Operations, their parameters and connections in the IDFG are stored at any point in time enabling the CLIJ-assistant to offer an undo-function for virtually unlimited rewinding parameter changes. Furthermore, to improve reproducibility a broader audience boosts the need for accessible tools for building GPU-accelerated image analysis workflows; in the life sciences and in adjacent imaging-dependent research fields. Typically, designing data analysis procedures utilizing GPUs involves expertise in programming and knowledge of GPU-specific programming languages such as the Open Computing Language (OpenCL) [1]. We demonstrate how one can construct complete image analysis workflows without writing a single line of OpenCL by assembling workflows from operations provided by the CLIJ framework [2]. We called the user interface CLIJ-asssistant because it allows interactive design of image data flow graphs (IDFGs) in ImageJ [3] or Fiji [4], while guiding the user with automatic suggestions. Depending on previously executed operations, it only shows operations that are suitable to the currently selected image, as shown in Figure 1. Automatic suggestions, semi-automated parameter optimization, and the immediate view of results en-

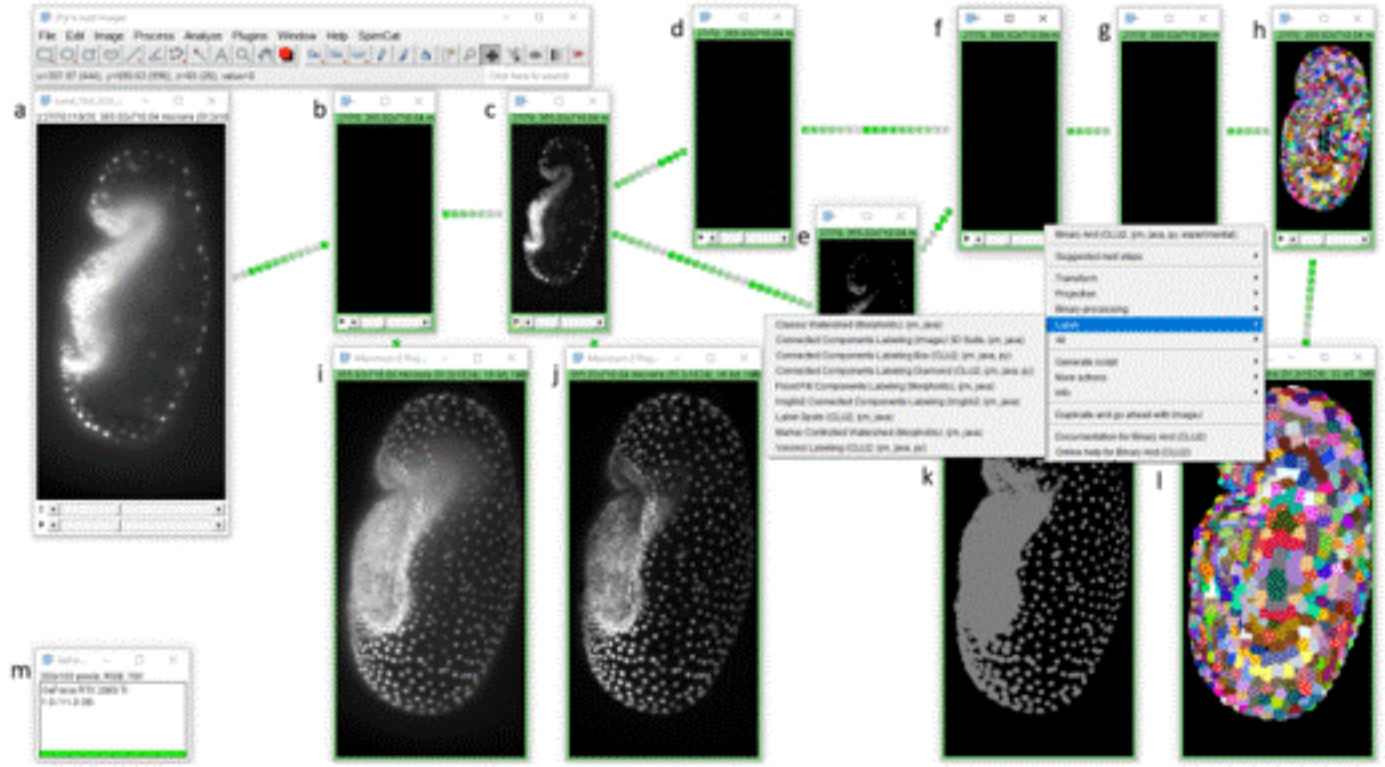
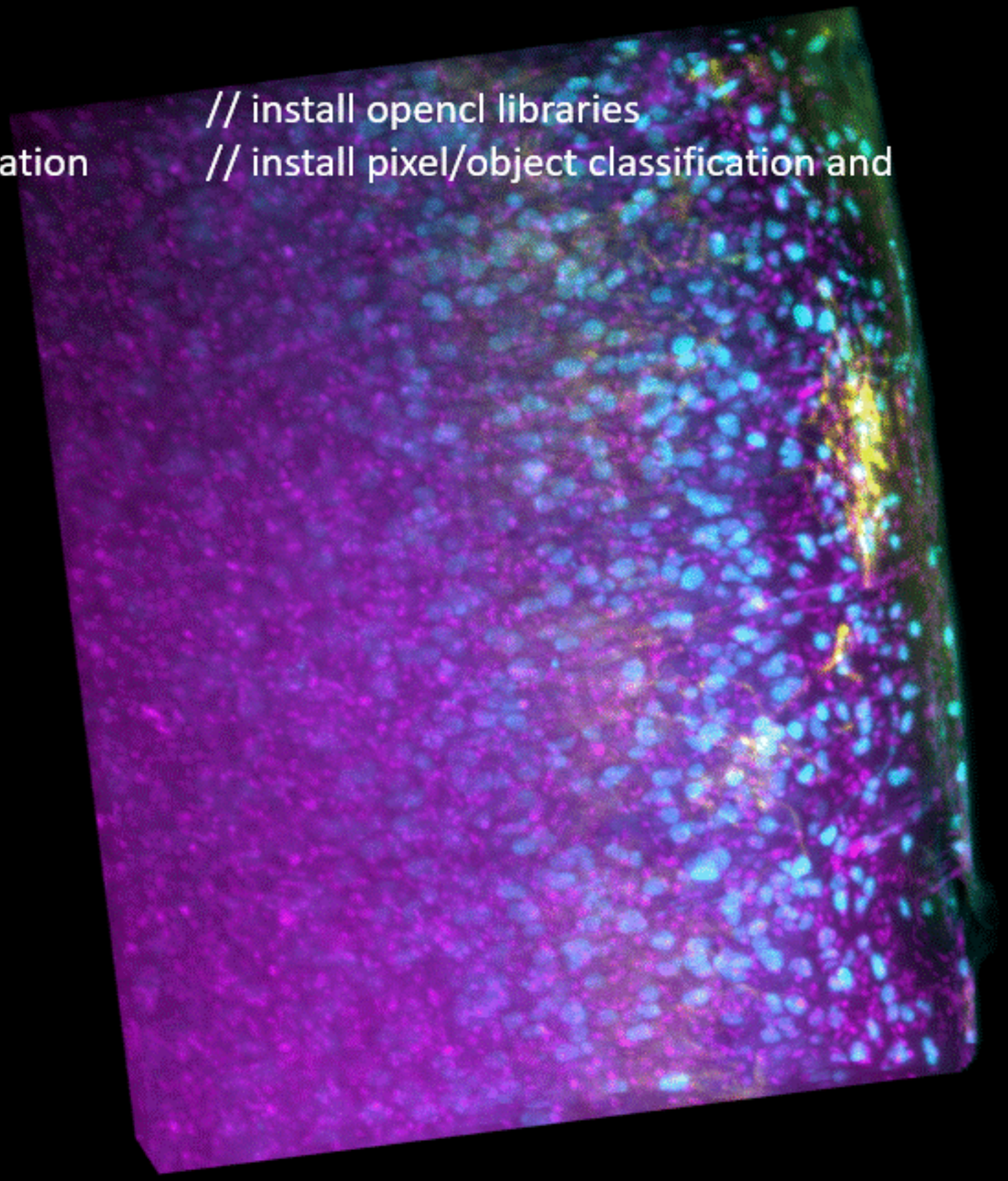


Figure 1: An IDFG allows assembly of image processing workflows and display of intermediate results in real-time. The presented graph takes a 4D image stack (a), pushes the current frame to the GPU memory (b), applies a top-hat filter for background subtraction (c), spot detection (d), thresholding (e), binary AND (f) spot labeling (g) and labeled spot extension (h). To facilitate examination of intermediate results, maximum intensity projections for intermediate results of pushing (i), background subtraction (j), thresholding (k) and label extension (l) are shown. The GPU memory display (m) allows to monitor available memory while setting up an IDFG. From any step in the graph, the user can select suitable subsequent operations using the shown right-click menu. The example shows the menu for the binary AND operation.

In an **Anaconda prompt**:

- conda install -c conda-forge pyopencl // install opencl libraries
- pip install napari-accelerated-pixel-and-object-classification // install pixel/object classification and Napari assistant

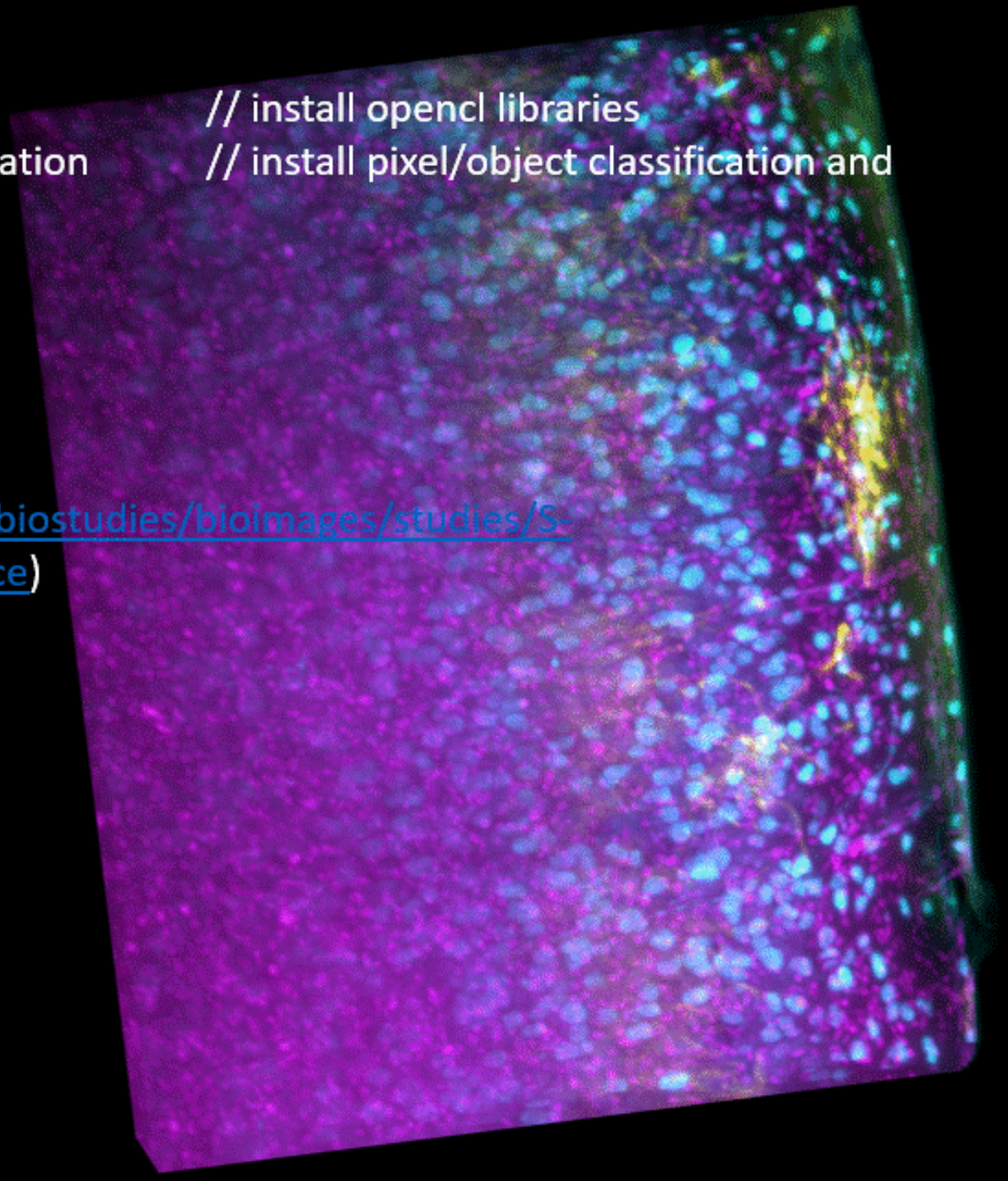


In an **Anaconda prompt**:

- conda install -c conda-forge pyopencl // install opencl libraries
- pip install napari-accelerated-pixel-and-object-classification // install pixel/object classification and Napari assistant

In Napari:

- Open PR012_L.tif (available at https://www.ebi.ac.uk/biostudies/bioimages/studies/S-BIAD1077?query=brain%20microscopy%20fluorescence)
- Train a pixel classifier to segment myelinated neurites
- Train a pixel classifier to segment neuronal cell bodies



In Napari:

- Create a workflow with the Napari assistant to load the pixel classifier to segment myelinated neurites, get the connected components, filter out the too small components and save the workflow
- Do the same thing for neuronal cell bodies
- Compare the segmentation obtained with the pixel classifier with Stardist and a more classical segmentation method with Napari assistant
- Compute the distance between neuronal cell bodies and myelinated neurites