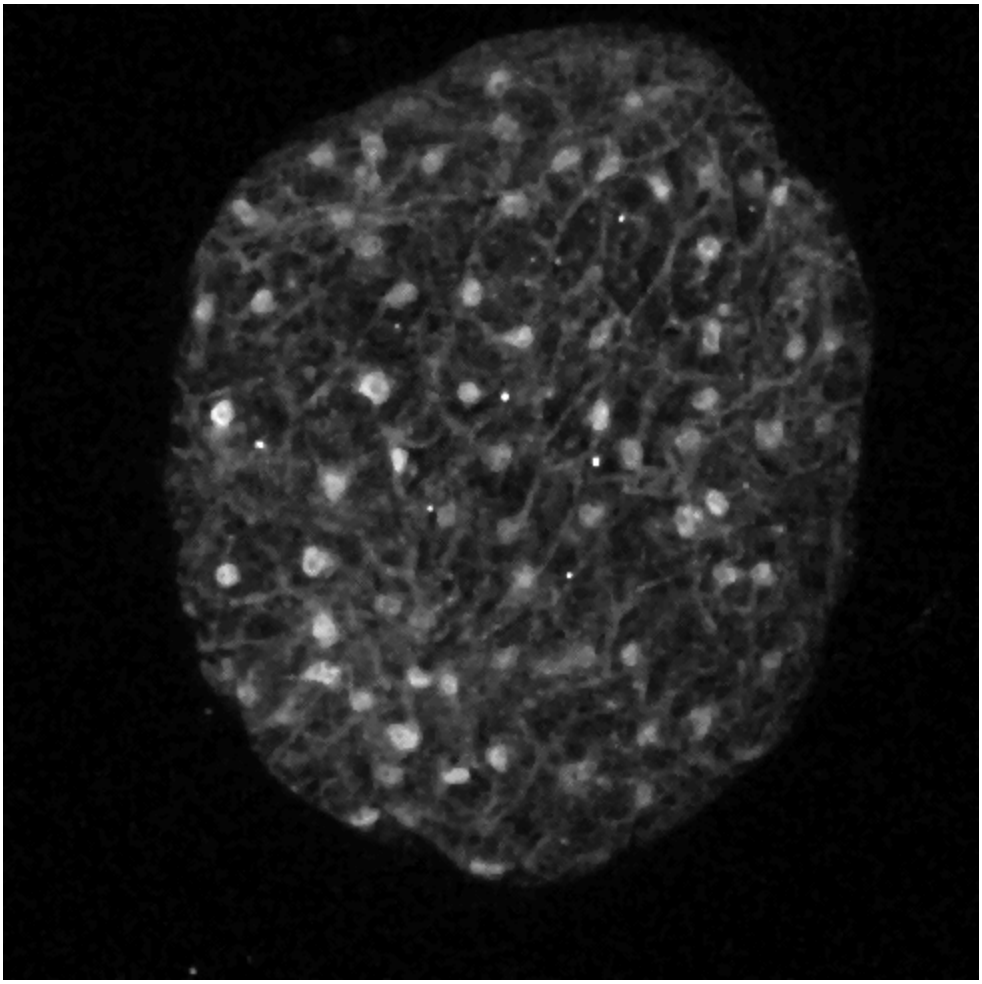
July, 2025. Liron Stettiner. lironst1@gmail.com.

Table of Contents

Introduction

The project is focused on identifying Hydra nuclei by measuring β-catenin signals using the Airyscan microscope. This
classification is then used in order to characterize changes in intensity, average size, distance and other

cellular-level statistics.



Cellpose

Cellpose (Nature article, GitHub and installation (make sure to install the GPU version of PyTorch), documentation, GUI documentation, YouTube tutorial)
is a neural network used for the segmentation of biological cells (BTW, the model also supports 3D images, so perhaps we
could extend this research to use the raw data without max projection).

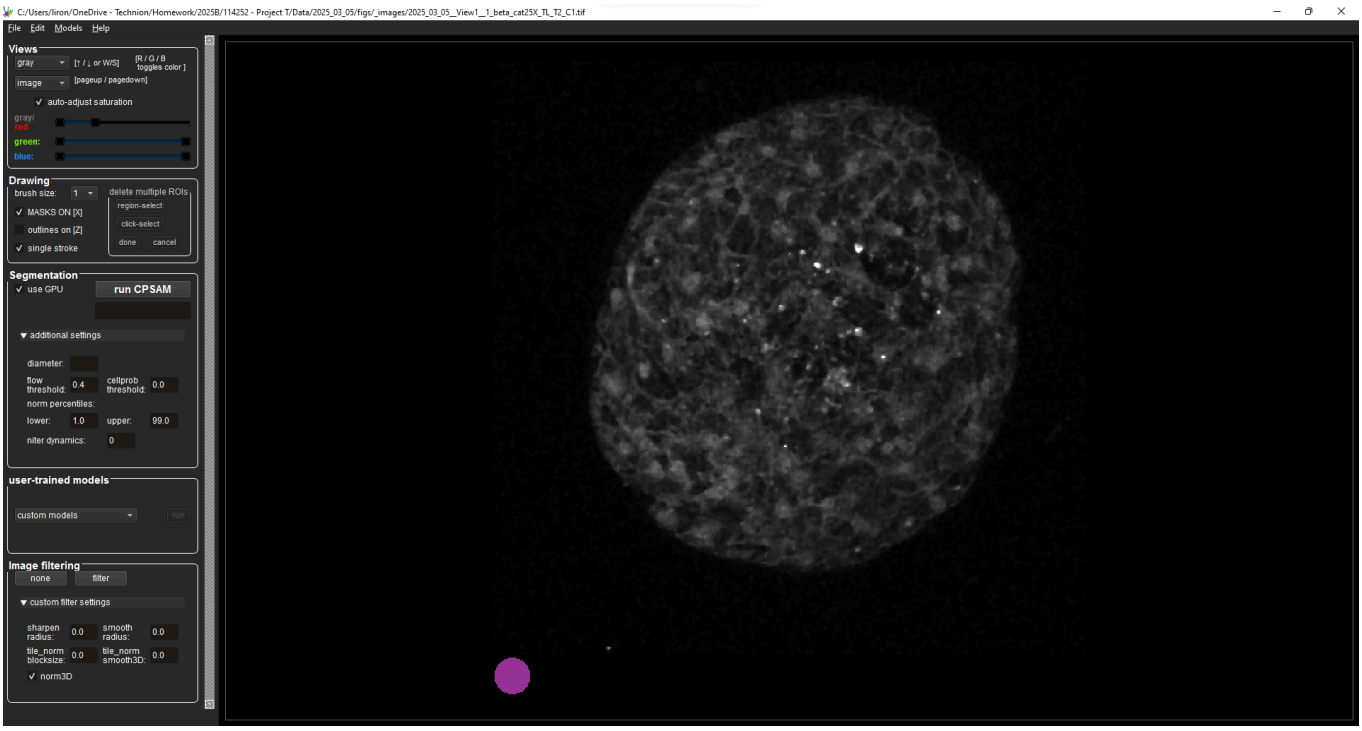
Cellpose has both a Python module and a GUI. I used the GUI to adjust model evaluation parameters by loading several
test images and fine-tuning the parameters (such that false-positives will be relatively small). Later, I used these
parameters in the Python module together with the rest of the code for automation.

To run the Cellpose GUI, follow the instructions in GitHub. Install it in your
Python environment:

```
pip install cellpose
```

Then run:

```
python -m cellpose
```

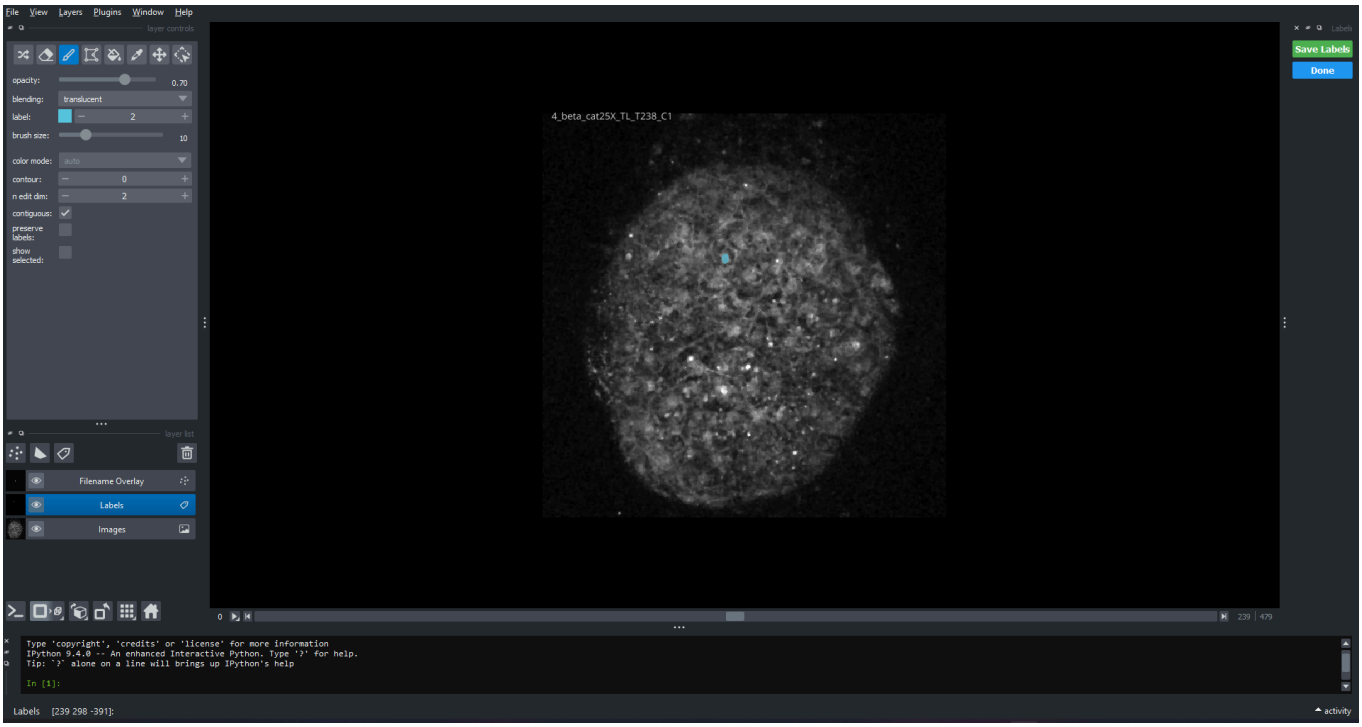


To load an image, click `Ctrl+L` or go to `File -> Load Image`.
You can change the parameters under the `Segmentaion -> additional settings` section and the
`Image filtering -> custom filter settings` section. Importantly, make sure to set the average cell diameter (in pixels).
In the past, there used to be multiple models you could choose between, but today there is only one available.
To run the model, click the `Segmentaion -> run CPSAM` button.

Napari

Napari is an interactive Python viewer for multi-dimensional images. We will use napari for manual image segmentation (see the `segment_manually` flag below).
To open napari (independently of this project), simply run:

```
napari
```



Code

I'm using Python version 3.11.9. Create a new virtual environment:

```
python -m venv "liron_project"
```

Activate the virtual environment:

```
source "myvenv/bin/activate"  # (Linux/Mac)
```

```
& "myvenv\Scripts\activate.bat"  # (Windows PowerShell)
```

Or, if using Anaconda:

```
conda create -y --name liron_project python=3.10.13
conda activate liron_project
```

Install the required dependencies:

```
python -m pip install python==3.11.9
python -m pip install -r requirements.txt -U --progress-bar on
```

To clear the cache, run:

```
conda clean --all -y
pip cache purge
```

Usage

To run the full code (cellpose segmentation and plots), run:

```
python main.py --dir DIR [--excel EXCEL] [--date DATE] [--pos POS]
```

For example, the following command will run the full code (segmentation and plotting) twice on experiments 2025-02-27 (pos 1) and 2025-03-05 (pos 1). The Excel file helps identify the experiment, its time interval, the first and final frames of β-catenin and plot the observed intensity (on a scale of {0,1,2,3} corresponding to {'none', 'low', 'mid', 'high'}):

```
python main.py --dir "path/to/all/data" --excel "betacatenin_head.xlsx" --date
"2025-02-27,2025-03-05" --pos "1,1"
```

The code assumes the parent directory contains `2025_02_27/View1` and `2025_03_05/View1` inside, and the outputs will be written into `2025_02_27/View1/output` and `2025_03_05/View1/output` accordingly.

To open napari for manual segmentation, run:

```
python main.py --dir DIR --segment_manually
```

Files

File Name	Description
`__init__.py`	The module initialization script.
`__cfg__.py`	A configuration file for all user-defined settings and parameters.
`main.py`	The main Python script to be called.
`main.ps1`	A PowerShell script that calls `main.py` (easier to use than calling `main.py` from the terminal each time).

`utils.py`
`utils_data_manager.py`
`utils_ilastik.py`
`utils_napari.py`
`utils_pixel_classifier.py`
	Python libraries with general utilities.

File Name	Description
`tests.py`	A few test functions used to check validity of inputs.

To run the code, open Terminal and run:

```
python main.py [...]
```

- Alternatively, run `main.ps1` from PowerShell.
- If one wishes to debug the code or have more flexibility, it is better to create a new python script. Here is an example:

```
from utils_data_manager import DataManager

DIR = "path/to/all/data"
EXCEL = "betacatenin_head.xlsx"
DATE = "2025_03_05"
POS = 1

dm = DataManager(dir_root=DIR,
        excel_data=EXCEL,
        date=DATE,
        pos=POS,
)
```

When calling with the `--help` flag, the output will be:

```
usage: main.py [-h] [-d DIR] [-p] [-dl DIRECTORIES_LIST] [-o OUTPUT] [-e
EXCEL] [--date DATE] [--pos POS] [--view VIEW] [--sample_size SAMPLE_SIZE] [--
labeled] [--unlabeled] [--no_cpsam_mask] [--no_plot] [--plot_only_stats] [--
segment_manually] [-v] [--debug]

options:

  -h, --help            show this help message and exit

  -d DIR, --dir DIR     Path to a directory containing the images. Default is
the current directory. Output will be saved in a subdirectory named 'output'.
If output directory already exists, values will be read from it for processing
(if they exist).

  -p, --print           Print image tree and exit.
```

```
  -dl DIRECTORIES_LIST, --directories_list DIRECTORIES_LIST
                        A .txt file with a list of directories to process. All
frames in each directory will be processed.

  -e EXCEL, --excel EXCEL
                        Path to an Excel file with experiment data. If
provided, the script will read the following columns: ['date', 'pos',
'time_after_cut', 'time_interval', 'main_orientation',
'initial_frame_beta_catenin', 'final_frame_beta_catenin',
'beta_catenin_intensity']. If `--date` and `--pos` are provided, the script
will only process data in `--dir` based on these values. Otherwise, it will
process all data in `--dir` (which should also appear in `--excel`).

  --date DATE           Date of the experiment in the format 'YYYY-MM-DD'. If
provided, the script will filter the data in `--excel` based on this date.
Multiple dates can be provided as a comma-separated list 'YYYY-MM-DD,YYYY-MM-
DD'. Note that this will only work if `--excel` and `--pos` are provided.

  --pos POS             View of the experiment, given as an integer. If
provided, the script will filter the data in `--excel` based on these
positions. Multiple positions can be provided as a comma-separated list
'1,2,3'. Note that this will only work if `--excel` and `--date` are provided.

  --view VIEW           An alias for `--pos`

  --sample_size SAMPLE_SIZE
                        Number of images to randomly sample from the
directory.If given in the range (0, 1], it is interpreted as a fraction (True
is the same as 1, i.e., use all data in random order. False will use all data
in the order discovered by os.path.walk). If None, all images are used.

  --labeled             Only process labeled images.

  --unlabeled           Only process unlabeled images.

  --no_cpsam_mask       Do not run Cellpose model (CPSAM) for segmentation. By
default, the script will run CPSAM for segmentation. If set, the script will
skip CPSAM segmentation and use existing masks (if available).

  --no_plot             Do not plot and do not save figures. By default, the
script will generate plots and save them in the output directory.If set, no
plots will be generated.

  --plot_only_stats     Plot only statistics. If set, the script will only
generate statistics plot, without frame-by-frame plots.
```

```
  --segment_manually    Segment images manually using napari. If set, the
script will open a napari viewer for manual segmentation.

  -v, --verbose         Enable verbose logging. If set, debug messages will be
printed to the logger and console.

  --debug               Enable debug mode. If set, debug messages will be
printed to the logger and console, and seed will be set to a fixed value for
reproducibility.
```

Output

```
└── <date>
    └── <pos>
        ├── image_1.tif (orignal images, could also be links image_1.tif.lnk)
        ├── ...
        └── labels
            ├── image_1.tif (user labels for training)
            ├── ...
        └── output
            └── cpsam_out (Cellpose model outputs)
                ├── image_1.pkl
                ├── ...
            └── figs
                └── classification (classification of each frame)
                    ├── image_1.tif
                    ├── ...
                ├── stats.tif (statistics of the entire experiment)
            └── random_forest_prob (output probs of RF pixel classifier)
                ├── image_1.pkl
                ├── ...
```

- The random forest pixel classifier is not currently implemented into the default run of `main.py`, but may be called through `utils_data_manager.DataManger.pixel_classifier_predict_prob()`.