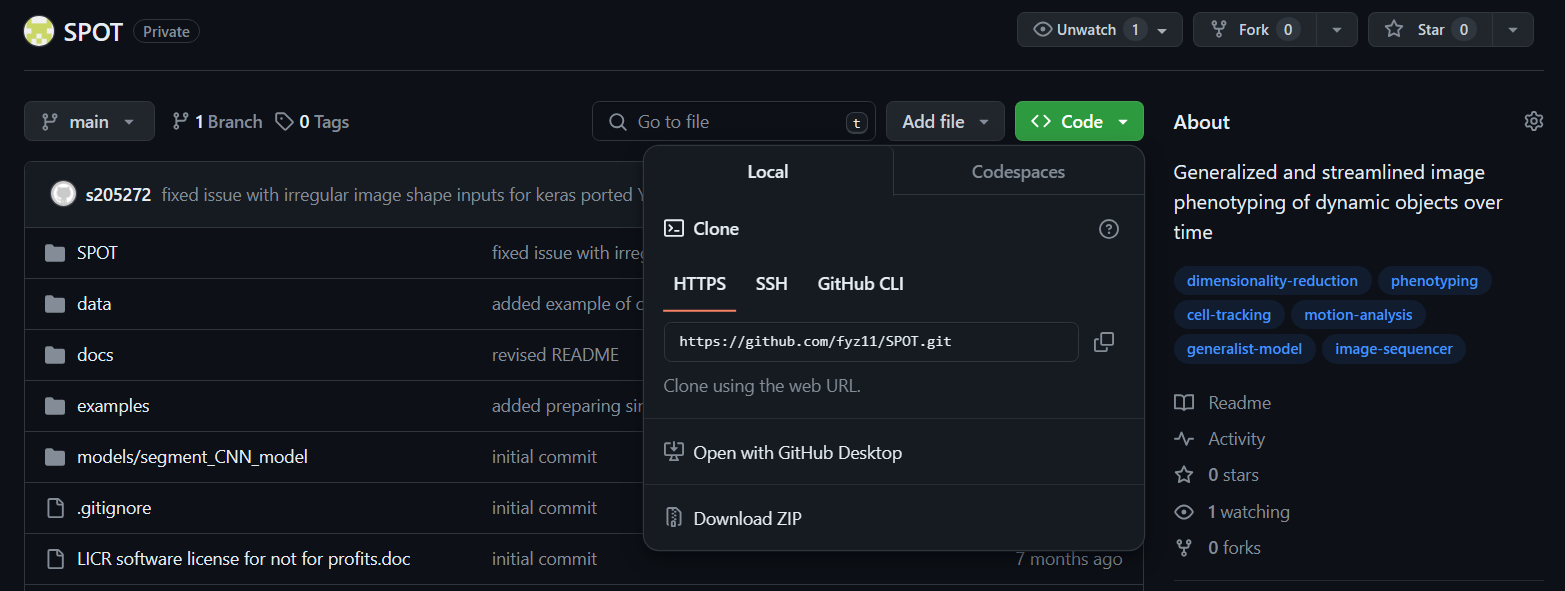
**Getting Started with SPOT (Python Library):**

**a Step-by-Step Tutorial on running the example scripts**

1. **Basic Installation of Repository:**



1. Go to <https://github.com/fyz11/SPOT/tree/main>, click the green ‘code’ button and either ‘download ZIP’ or clone the repository onto your PC using Git and the terminal using the command: git clone <https://github.com/fyz11/SPOT.git>
2. We recommend setting up a new Python Anaconda environment. Download and install miniconda from https://docs.anaconda.com/free/miniconda/miniconda-install/ following the instructions for your operating system. Then in a command prompt execute the command conda create –n SPOT\_env python=3.9 where SPOT\_env is the name of the environment to install SPOT into and python=3.9 specifies the desired Python version to install.
3. Activate the conda environment in the terminal, conda activate SPOT\_env
4. Navigate to the root of the Github repository and run pip to install SPOT to the environment, pip install .



You should now be set up so that you can import SPOT using Python as if it was a native library from anywhere on your system.

1. **Downloading and setting up neural network weights if using our custom pretrained organoid bounding box detection and segmentation:**
2. (For using our custom organoid detector). Please download and save the pretrained weights from the link: [keras\_YOLOv3\_organoid\_detector\_weights](https://www.dropbox.com/scl/fi/qzowc9s9n30zh6qdyzeqw/keras_YOLOv3_organoid_detector2.h5?rlkey=6deiqemsmcz3yin9b5dnz0e6y&dl=0) to the desired folder on your system. We suggest keeping this in the same location as the segmentation weights below. This detector is used to find all organoids, returning in terms of bounding box in YOLO format i.e. each box is given as (class, x, y, w, h) where (x,y) is the centroid, and (w,h) is the width and height of the box.

1. (For using our custom organoid segmentation). This is included within the SPOT Github, located in models/segment\_CNN\_model/. This model has been trained with resized 64x64 crops of individual organoids. The way to use it is to crop detected individual organoids and resize to 64x64 images. The output is a probability map 0-1 which can be thresholded and binarized to give the organoid binary segmentation.

You should now have download and saved the saved models locally on your system.

*The remainder tutorial steps assume you have placed the models under SPOT/models/ of the Github repository and is running the examples scripts in-place i.e. from the examples folder*

1. **Video Dataset preparation before running SPOT (Preprocessing):**
2. (Extended focus depth projection for ‘flattening’ 3D stack to a single 2D projection image, **Optional**).

The following examples can be executed on the example organoid video provided in the repository at SPOT/data/organoids/fluorescent\_murine\_color/KRAS G12D EYFP 2.wmv

1. (Detection of global blurry/out-of-focus frames and temporal registration for multi-part acquisitions). See SPOT/examples/SPOT\_Stage1\_Step0\_translation-register-RGB-confocal\_video.py to see an example of how to use registration to remove stage shift in acquisition.
2. (Spectral unmixing for multi-color videos, **Optional**). There may be spectral bleed-through between different color channels due to the fluorophores used and the microscope filters used. This may cause some objects to ‘bleed-through’ into other channels. One way to correct this to improve downstream processing is to apply spectral unmixing using non-negative matrix factorization. See SPOT/examples/SPOT\_Stage1\_Step0\_unmix-RGB-confocal\_video.py for example of how to run the function.
3. **Running individual SPOT steps on the videos:**

The following examples can be executed on the example organoid video provided in the repository at SPOT/data/organoids/fluorescent\_murine\_color/KRAS G12D EYFP 2.wmv. This section walks through how to run our custom organoid detector, tracker and segmentation to obtain organoid boundaries per timepoint, per video prior to computation of the SAM phenome.

These scripts demonstrates working with RGB videos. To see example of how to adapt the steps to single channel grayscale videos (i.e. by faking an RGB channel) see XXX.

1. (Organoid bounding box detection). This uses the YOLOv3 architecture to detect organoids. Each detected organoid in a frame is saved in the format (class, x, y, w, h) where ‘class’ is the type of organoid, here just ‘organoid’; (x,y) is the centroid of the bounding box; and (w,h) is its width and height. See SPOT/examples/SPOT\_Stage1\_Step1\_detect\_bbox-RGB-confocal\_video.py for example of how to run the function. **N.B.** you must check the weightsfile parameter is set to where you have downloaded the pretrained organoid detector weights.
2. (Organoid bounding box tracking). This uses bipartite matching using 1. - intersection-over-union as a cost function to optimally match detected bounding boxes between consecutive frames. Optical flow is further used to improve matching. The detections at frame *t* is transformed using the optical flow to its predicted position at frame *t+1*. This predicted position is used to match with the detected bounding boxes frame *t+1*. If a match is found, the predicted is replaced with that detected. If a match is not found, the predicted is used as the detection. If an organoid uses predicted detection for more frames than a specified ‘wait time’, here 5 frames, then this track is terminated and all previous imputed 5 detections removed. See SPOT/examples/SPOT\_Stage1\_Step2\_track\_detect\_bbox-RGB-confocal\_video.py for example of how to apply.
3. (Organoid segmentation from bounding box crop). Organoid tracks are filtered by minimum lifetime. The tracked bounding boxes are used to crop the image and obtain image patches containing single organoids. The single organoids are binary segmented using a pretrained convolutional network after thresholding on the output cell probabilities. The neural network is trained on 64 x 64 pixel images. Patches are resized to run the network. The organoid boundary is computed from the output and is rescaled in x- and y- directions to recover the real dimensions. See SPOT/examples/SPOT\_Stage1\_Step3\_segment\_tracked\_bbox-RGB-confocal\_video.py for example of how to apply.
4. (Postprocessing organoid segmentation including using temporal tracking to improve temporal consistency). We use the segmentations to detect and remove potential duplicated segmentations, to remove border segmentations of partial organoids, to temporally smooth segmentations to combat discontinuous segmentation errors, and to produce temporal tracks with a minimum temporal segmentation consistency (i.e. frame-to-frame intersection-over-union > (we use 0.25)). See SPOT/examples/SPOT\_Stage1\_Step4\_postprocess\_segment\_tracked\_bbox-RGB-confocal\_video.py for example of how to apply.
5. **Compute SAM phenome**
6. (Compute SAM phenome for each segmented organoid, with motion features computed using consecutive frames). We compute the SAM phenome for each final segmented organoid for every timepoint. Motion features for frame *t-1* are computed using frame *t*. Therefore terminal organoid instances of tracks do not have motion features. These will be removed during analysis. See SPOT/examples/SPOT\_Stage2\_Step1\_compute\_SAM\_phenomes.py
7. **Generating metadata for video files**
8. (Automatically generate a metadata table for all analyzed videos). The previous steps process the organoids without using any metadata information of the experiment such as genetics, treatment condition etc. For analysis, we need to match the video metadata to each segmented organoid instance. To do this, we generate a template table for all videos in an experiment, which automatically populates some salient information such as filepath etc. See *SPOT/examples/SPOT\_Stage2\_Step2\_generate\_metadata\_table.py*. The generated table contains columns which are blank which can be manually filled in. If not filled in, they will be treated as NaNs. This is information such as channel number corresponding to each genetics/condition, if using multi-color videos.
9. By default, Step 15 will generate a ‘metadata.csv’ file in the rootfolder of the experiment. See the example at SPOT/data/organoids/fluorescent\_murine\_colon/metadata\_expt.csv. You will see that whilst we pre-filled in the pixel resolution and temporal sampling through the code, additional metadata such as the img\_channel (which image channel 1=red, 2=green, 3=blue) the organoid is in. If there are multiple channels, each is to be separated by : . The order should be the same as that for Genetics. Since here there is no patients or other conditions, they are left blank, else they should be filled in similarly to genetics column. See SPOT/data/organoids/fluorescent\_murine\_colon/metadata\_expt\_filled.csv for an example of how the information should be filled in for this example video.
10. **Exporting computed metrics with metadata for each video dataset (experiment)**
11. (Compiles all SAM phenomes from all videos in an experiment to produce a single .csv for Shape, Appearance and Motion features and each with video metadata and a single .mat object file with all compiled organoid segmentations as 200 equispaced point boundaries). See *SPOT/examples/SPOT\_Stage2\_Step3\_compile\_and\_export\_SAM\_phenomes.py*.
12. (Crops individual organoid image patches using the segmentation boundaries, compiling them into a single .mat object according to the compiled boundaries and metadata from Step 16). See *SPOT/examples/SPOT\_Stage2\_Step4\_compile\_and\_export\_object\_patch\_imgs.py*.
13. **SPOT analysis of all experiments**

We use the publicly available Single-cell tracking challenge datasets available for download free at <https://celltrackingchallenge.net/2d-datasets/> and have frame-by-frame segmentation and tracking data to demonstrate how to use SPOT to compute SAM phenomes and analysis the dynamic phenotypes. This also illustrates usage of SPOT on single-channel videos.

1. (Prepare single cell tracking challenge dataset: extract segmentations as contours) see SPOT/examples/SPOT\_demo\_single-cell-tracking-challenge\_Step0\_Prepare\_Single\_Cell\_Tracking\_Challenge\_Dataset\_All.py
2. (Compute SAM phenomes for all valid tracked segmentations). See SPOT/examples/SPOT\_demo\_single-cell-tracking-challenge\_Step1\_Compute\_SAM\_phenome\_Single\_Cell\_Tracking\_Challenge\_Dataset\_All.py
3. (Compile all SAM phenomes from both available videos). We demonstrate how it could be easier to bypass the official generating video metadata step. See SPOT/examples/SPOT\_demo\_single-cell-tracking-challenge\_Step2\_Compile\_All\_SAM\_phenomes\_cell-tracking-challenge.py
4. (Conduct all analytical steps of SPOT stage 3). This example works through the unsupervised feature selection, identifying feature contribution, SAM modules and the phenotype clustering, derivation of phenotype trajectories, stacked temporal barplots of phenotype clusters and HMM transition analysis. See SPOT/examples/SPOT\_demo\_single-cell-tracking-challenge\_Step3\_SPOT\_analyze\_all\_SAM\_phenomes\_cell\_tracking\_challenge.py