Cell Tracking in 3D using deep learning segmentations

Varun Kapoor[‡§*], Claudia Carabana Garcia[‡§]

Abstract—Biological cells can be highly irregular in shape and move across planes making it difficult to be detect and track them in 3D with high level of accuracy. In order to solve the detection problem of such cells we developed a deep learning based segmentation technique which can reliably segment oddly and differently shaped cells in the same image in 3D. In biological experiments cells are sometimes brighter in the first few frames and are more faint later on or there could be bright and faint cells in the same image. A single deep learning network may not be able to segment such variable intensities present in the same image dataset. Hence we developed a technique that combines semantic and instance segmentation information coming from two different networks, U-net and stardist. U-net is used to perform foreground-background semantic segmentation whereas stardist is used to obtain a convex polygon representation of the overlapping cells which we use to obtain seeds for doing watershed on. The tools are available as open source python tools with front end user interface for training and applying model prediction as user friendly jupyter notebooks. Post segmentation we also developed a tool to track such cells using customised cost function to solve linear assignment problem and Jaqman linker for linking the tracks of dividing and merging cells. The tool is developed using widely used tool for tracking in Fiji, Trackmate. We perform the post analysis of tracks is Napari, which is an Euler angle based viewer providing user friendly track view of the obtained tracks along with analysis of the obtained trajectories. Napari tool for visualizing and analyzing the tracks is open source python based tool with front end jupyter notebooks for launching the customised widget.

Index Terms—segmentation, tracking, deep learning, irregular shaped cells

Introduction

Studying the dynamics of biological cells is key to understanding key biological questions. Such questions involve imaging the cells under different conditions, tracking their development over time and extracting relevant dynamical parameters such as cell intensity variation, cell size change, cell velocity, cell to tissue boundary distance change over time etc. Imaging conditions can be highly variable and have different sources of noise which degrades the quality of the image and with the increasing size of the data acquired using these microscopes it is imperative to have automated algorithms to enable their quantification. Such analysis requires reliable segmentation of cells followed by tracking software to track their motion and finally a track analysis software to extract the relevant information. In out work we develop a technique to segment cells of irregular shape in 3D including the cells that are very faint in their intensity signal. We trained a U-net :cite: Unet network for doing foreground background segmentation, we trained it on cells of highly variable intensity and with filament shaped and irregular shaped cells in the same training. We also trained a stardist network :cite: Stardist to do instance segmentation of cells, stardist learns the concept of a cell by learning a distance map of the cell, it then approximates the cell shape by a convex polygon. Such an approach works well for roundish shaped cells but often leads to boundary reconstruction errors for cells that have irregular shape. Hence we developed an approach that combines the U-net and stardist results to segment cells of irregular shape. This approach is available as pip package vollseg.

After segmentation of such cells often the bio image analysis task requires tracking of such cells. These cells can move several pixels from one time frame to the next in XYZ planes and linking the cells to each other often requires more than just the centroid information as the shape and the intensity of the cells contains information that can help in their reliable tracking. Such a tool is available in Fiji as a scijava plugin Trackmate. We use the codebase of Trackmate to build our customised tool called bTrackmate. It is freely available as a Fiji plugin too. Trackmate provides an interactive track editing interface and exports the tracks as xml files containing the track information. Since the default viewer of Fiji may not always be an optimal choice of the viewer for tracks in XYZ and time we provide a possibility to export the xml file to be viewed in Euler angle based viewer in python called Napari. Napari is used not just to view the tracks but also to extract track information from the obtained tracks that can be extended based on use cases. For our use cases we provide front end jupyter notebooks and the package is freely available as pip package napatrackmater.

Segmentation

Our segmentation task required segmentation of cells coming from developing mouse embryo in 3D. These cells can be elongated (fibroblasts) or have an irregular shape (luminal). Any bio image analysis task starts with segmentation of such cells coming out of a microscope. In order to avoid phototoxicity that leads to cell death the imaging conditions have to be modulated to not have too high laser intensity under which the cells are imaged in. This leads to a low signal to noise ratio image dataset. Segmentation of such cells could be tedious with the conventional computer vision based techniques alone which almost always will lead to

* Corresponding author: varun.kapoor@curie.fr
‡ Institut Curie
§ Paris, France

Copyright © 2021 Varun Kapoor et al. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

DRAFT

over segmentation in such images :cite:Tobias . However given enough training data, deep learning networks can be trained to achieve the same task with high degree of accuracy. Segmentation tasks can broadly be divided into semantic or instance segmentation methods. In the semantic segmentation approach only background-foreground segmentation is performed where the pixels are classified either belonging to an object class or not, in the instance segmentation approach the object pixels are classified as belonging to object A or B. In our case we use U-net to perform semantic segmentation of the cells in 3D. U-net is independent of shape of the cell hence can do a good semantic segmentation task, if the cells do not overlap connected component analysis alone is enough to segment the cells. But often in timelapses the cells often overlap and this requires a network that can do instance segmentation. Stardist has proven to be network that performs well in such segmentation tasks compared to other available networks for biological cells. Stardist is an N + 1 channel U-net network where N output channels are distance from the center of the cell to the boundary over a range of angles and a single channel for foreground-background pixel probability map. Using this distance information a mathematically abstract representation of a cell can be learnt by the network. The limitation of this network is that it works reliably for star-convex shapes and does not perform well if the shape of the cells is irregular. We combine the strengths of both the networks in the following way: From stardist we obtain convex polygons after doing the non maximal suppression, from these convex polygons we obtain their centroid that serve as a starting seed for the watershed process. By keeping the probability threshold high we only keep the seeds of relatively bright cells at a given timepoint in 3D. We then use the U-net segmentation to find the seeds that were missed by stardist, if stardist had those seeds we do not put new seeds but if these seeds were missed for the faint cells in the timepoint we accept the U-net seeds and add it to the seed pool to start the watershedding process. In such a combination we produce an energy map where the distance transform was learned by the network. The object instances are basis of the energy map.

To train U-net and stardist networks for the segmentation task we created labelled training dataset of mouse basal and luminal cells. There are several network hyperparameters that have to be chosen to ensure that the model is not over or under fitting to the data. Such hyperparameters include the network depth, the starting number of convolutional filters that double with depth thereby increasing the number of optimization parameters of the network. For a network to generalize well on unseen data we need to fine tune these parameters.

We trained several networks, compared their training and validation losses and also measured their performance on ground truth data the networks to asses their performance. In order to assess the performance of the segmentation we use object level metric proposed by DSB18 for evaluating the segmentation results. We compute true positive (TP) as intersection over union of the predicted and the ground truth being greater than a given threshold,

$$\tau \in [0,1]$$

. Unmatched objects are false positives (FP) and unmatched ground truth objects are false negatives (FN). We then compute average precision

$$AP_\tau = \frac{TP_\tau}{TP_\tau + FP_\tau + FN_\tau}$$

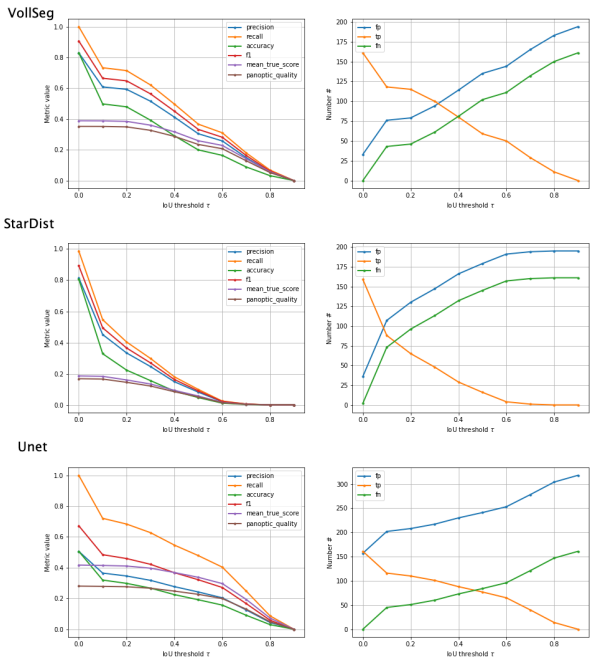

Fig. 1: *Metric of comparision between 1) VollSeg, 2) Stardist, 3) Unet.*

evaluated across 7 Z stacks. We also compute mean squared error between the ground truth and the predicted results. In Fig. we show the stardist, unet and results from our approach (vollseg). We also show the results as plots in Fig.:ref:metrics U-net has low performance when it comes to object level segmentation as two channel unet can not do instance segmentation and hence shows poor object level detection scores but good true positive rate. But at a semantic level U-net is better than stardist at resolving the shape of the objects, vollseg even has a better performance than Unet as it discards objects below a certain size which are unlikely to be biological cells Fig.:ref:mse. It was shown in :cite: stardist that their network has better performance compared to multi channel unet results hence we compared our segmentation directly with stardist and have lower mean squared error and better shape prediction as compared to stardist. We also show a visual comparision of the GT and VollSeg segmentation in Fig., comparision of GT and U-net segmentation in Fig. and of GT and stardist segmentation in Fig. From these metrics and visual segmentation comparision we see that vollseg has overall best performance in terms of segmentation of such irregular shaped cells in 3D.

The code for the seed criteria is shown below

```
def iou3D(boxA, centroid):

  ndim = len(centroid)
  inside = False

  Condition = [Conditioncheck(centroid, boxA, p, ndim)
   for p in range(0,ndim)]

  inside = all(Condition)

  return inside

def Conditioncheck(centroid, boxA, p, ndim):

    condition = False

    if centroid[p] >= boxA[p]
```

DRAFT

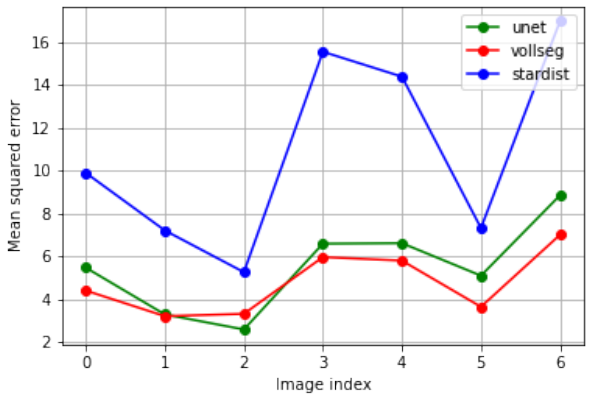

Fig. 2: *Mean Squared error comparision between VollSeg, Stardist, Unet.*

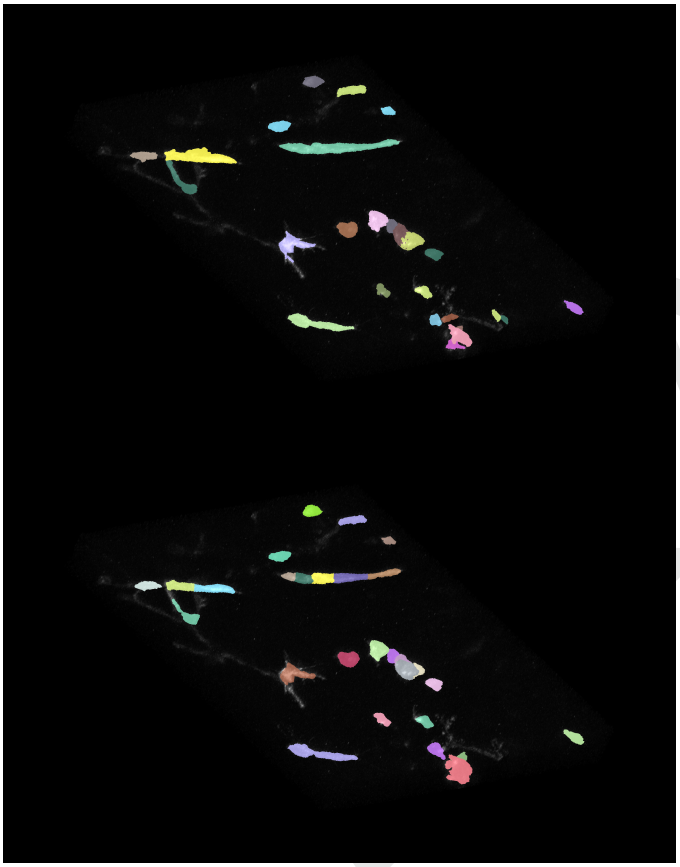

Fig. 3: *Visual 3D segmentation comparision between 1) GT segmentation (top) and 2) VollSeg segmentation (bottom).*

```
    and centroid[p] <= boxA[p + ndim]:

        condition = True

    return condition
```

After obtaining the pool of seeds we can perform watershedding on either the distance map coming from stardist or the pixel probability map that is also an output of the stardist algorithm. We use U-net semantic segmentation as a mask in the watershedding process. The code for doing so is shown below

```
def WatershedwithMask3D(Image, Label,mask, grid):
```

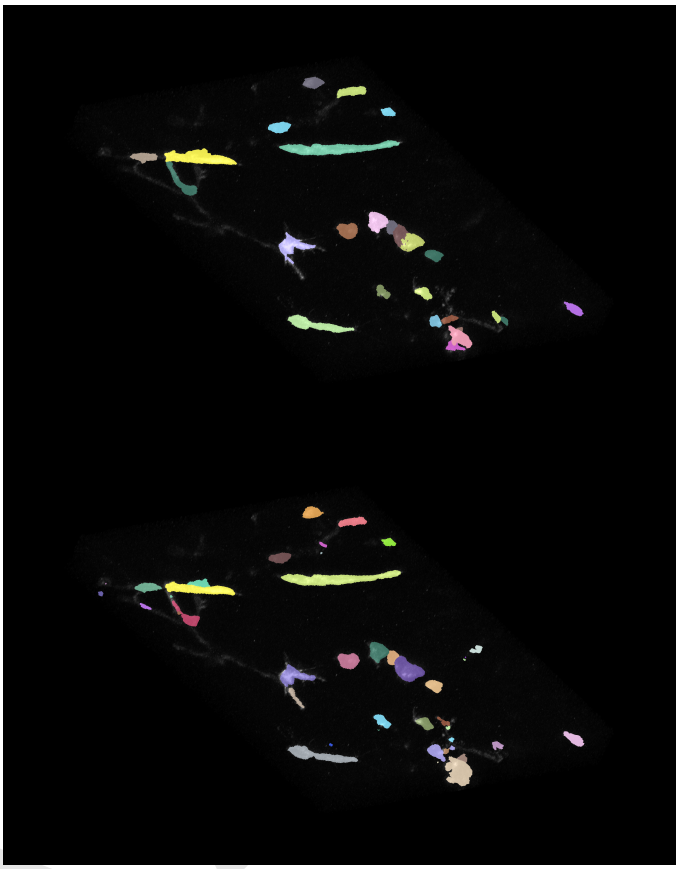

Fig. 4: *Visual 3D segmentation comparision between 1) GT segmentation (top) and 2) Unet segmentation (bottom).*

```
properties = measure.regionprops(Label, Image)
binaryproperties =
measure.regionprops(label(mask), Image)
cord =
[prop.centroid for prop in properties]
bin_cord =
[prop.centroid for prop in binaryproperties]
Binarybbox =
[prop.bbox for prop in binaryproperties]
cord = sorted(cord ,
key=lambda k: [k[0], k[1], k[2]])
if len(Binarybbox) > 0:
        for i in range(0, len(Binarybbox)):

            box = Binarybbox[i]
            inside =
            [iou3D(box, star) for star in cord]

            if not any(inside) :
                    cord.append(bin_cord[i])


cord.append((0,0,0))
cord = np.asarray(cord)
cord_int = np.round(cord).astype(int)

markers_raw = np.zeros_like(Image)
markers_raw[tuple(cord_int.T)] =
1 + np.arange(len(cord))
markers =
morphology.dilation(markers_raw,
morphology.ball(2))

watershedImage =
watershed(-Image, markers, mask)
```

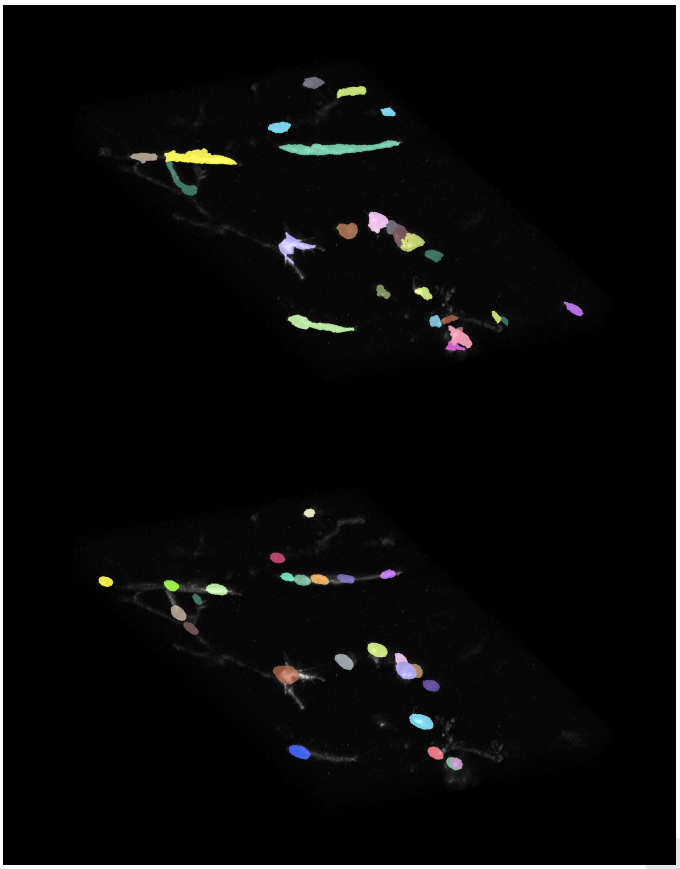

Fig. 5: *Visual 3D segmentation comparision between 1) GT segmentation (top) and 2) Stardist segmentation (bottom).*

```
    return watershedImage, markers
```

Here the Label comes from stardist prediction and mask comes from the U-net prediction. The result of this approach is a 3D instance segmentation which we obtain for the luminal cells as shown in Fig.{1}. In the software package we provide training and prediction notebooks for training the base U-net and stardist networks on your own dataset. The package comes with jupyter notebooks for training and prediction on local GPU servers and also on Google Colab.

Interactive codebase

To train your networks using vollseg, install the code via pip install vollseg in your tensorflow enviornment with python > 3.7 and < 3.9. The first notebook needed is the one with takes the training dataset as input and creates npz file for U-net training, specify the path to your data that contains the folder of Raw and Segmentation images with the same name of images to create training pairs. Also to be specified is the name of the generated npz file along with the model directory to store the h5 files of the trained model and the model name.

```
Data_dir = '/data/'
NPZ_filename = 'VolumeSeg'
Model_dir = '/data/'
Model_Name = 'VolumeSeg'
```

In the next cell specify the model parameters, these parameters are the patch size chosen for training in XYZ for making overlapping patches for training, the number of patches to make the training data are also to be specified. The network depth is an important hyperparameter, more the network depth more are the number of parameters in the network and the image patch size has to be big enough so that when downsampling happens with increasing depth the size of the image in the inner most layer is still greater than 1. Start number of convolutional filters is another crucial hyperparameter controlling the network learning capacity. The number of filters are double at each layer of the network and depending on the size of the training dataset and of the GPU memory capacity this parameter can be tuned when doing hyperparameter optimization to obtain the best model for the given dataset. In this cell as a first step we generate the npz file for U-net training by setting the boolean GenerateNPZ to be true. Then in the next cell we can either train U-net and stardist network sequentially by setting TrainUNET and TrainSTAR booleans to be true or the users can split the training task between two GPUs by making a copy of the notebook and training one network per notebook. The other parameters to be chosen are the number of epochs for training, kernel size of the convolutional filter, the number of rays for stardist network to create a distance map along these directions. Additionally some of the OpenCL computations can be perfromed on a GPU using gputools library and if that is installed in the enviornment you can set use_gpu_opencl to be true.

```
#Network training parameters
NetworkDepth = 5
Epochs = 100
LearningRate = 1.0E-4
batch_size = 1
PatchX = 256
PatchY = 256
PatchZ = 64
Kernel = 3
n_patches_per_image = 16
Rays = 128
startfilter = 48
use_gpu_opencl = True
GenerateNPZ = True
TrainUNET = False
TrainSTAR = False
```

After the network has been trained it will save the config files of the training configuration for both the networks along with the weight vector file as h5 files that will be used by the prediction notebook.

For running the network prediction on XYZ shape images use the prediction notebook either locally or on Colab. In this notebook you only have to specify the path to the image and the model directory. The only two parameters to be set here are the number of tiles (for creating image patches to fit in the GPU memory) and min_size in pixel units to discard segmented objects below that size. Since we perform watershed on either the probability map or the distance map coming out of stardist the users can choose the former by setting UseProbability variable to true or by default we use the distance map. The code below operates on a directory of XYZ shape images.

```
ImageDir = 'data/tiffiles/'
Model_Dir = 'data/'
SaveDir = ImageDir + 'Results/'
UNETModelName = 'UNETVolumeSeg'
StarModelName = 'VolumeSeg'

UnetModel = CARE(config = None,
name = UNETModelName,
basedir = Model_Dir)
StarModel = StarDist3D(config = None,
name = StarModelName,
```

DRAFT

```
basedir = Model_Dir)
Raw_path =
os.path.join(ImageDir, '*.tif')
filesRaw =
glob.glob(Raw_path)
filesRaw.sort
min_size = 5
n_tiles = (1,1,1)
for fname in filesRaw:

      SmartSeedPrediction3D(ImageDir,
      SaveDir, fname,
      UnetModel, StarModel,
      min_size = min_size,
      n_tiles = n_tiles,
      UseProbability = False)
```

Tracking

After we obtain the segmentation using our approach we create a csv file fo the cell attributes that include their location, size and volume of the segmented cells. We use this csv file of the cell attributes as input to the tracker along with the Raw image. The Raw image is used to measure the intensity signal of the segmented cels while the segmentation is used to do the localization of the cells which we want to track. We do the tracking in Fiji, which is a popular software among the biologists. We developed our code over the existing tracking solution called Trackmate cite{TM}. Trackmate uses linear assingment problem (LAP) algorithm to do linking of the cells and uses Jaqman linker for linking the segments for dividing and merging trajectories. We introduced a new parameter of minimum tracklet length to aid in the track editing tools also provided in the software. Hence by introducing a biological context of not having very short trajectories we reduce the track editing effort to correct for the linking mistakes made by the program. For testing our tracking program we used a freely available dataset from the cell tracking challenge of a developing C.elegans embryo. Using our software we can remove cells from tracking which do not fit certain criteria such as being too small (hence most likely a segmentation mistake) or being low in intensity or outside the region of interest such as when we want to track cells only inside a tissue. For this dataset we kept 12,000 cells and after filtering short tracks kept about 50 tracks with and without division events. The track information is saved as an XML file and can be re-opened to perform track editing from the last saved checkpoint. This is particularly useful when editing tracks coming from a huge dataset.

For this dataset the track scheme along with overlayed tracks in shown in Fig. The trackscheme is interactive as selecting a node in the trackscheme highlights the cell in Green and by selecting a cell in the image highlights its location in the trackscheme. Extensive manual for using the track editing is available on Fiji wiki.

Track Analysis

After obtaining the tracks from bTrackmate we save them as Trackmate XML file, this file contains the information about all the cells in a track. Since the cells can be highly erratic in their motions and move in not just the XY plane but also in Z we needed an Euler angle based viewer to view such tracks from different camera positions, recently a new and easy to use viewer based on python called Napari came into existence. Using this viewer we can easily navigate along multi dimensions, zoom and pan the view, toggle the visibility of image layers etc. We made a python package to bridge the gap between the Fiji and the Napari world

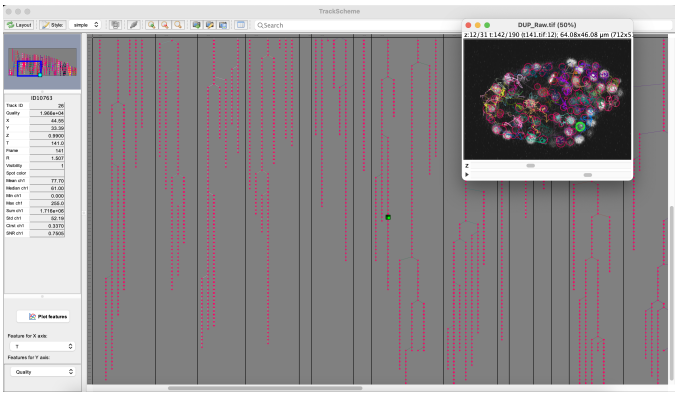

Fig. 6: *Trackscheme display for the C-elegans dataset.*

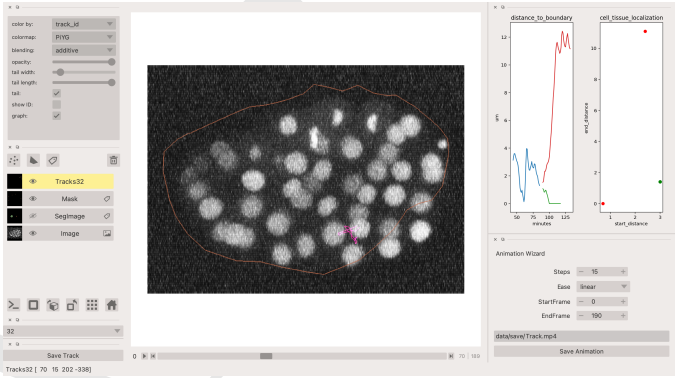

Fig. 7: *Parent cell before division.*

by providing a track exporter that can read in the track XML files coming from the Fiji world and convert them into the tracks layer coming form the Fiji world. We use this viewer not just to view the tracks but also to analyze and extract the track information. As a first step we separate the dividing trajectories from the non-dividing trajectories, then in one notebook we compute the distance of the cells in the track from the tissue boundary and record the starting and the end distance of the root tracks and the succeeding tracklets of the daughter cells post division for dividing trajectories and only the root track for the non-dividing trajectory. This information is used to determine how cell chooses its fate, does it start from inside the tissue and remain inside during the duration of the experiment or does it move closer to the tissue boundary. This information is crucial when studying the organism in the early stage of development where the cells are highly dynamic and their fate is not known a priori.

Also another quantity of interest that can be obtained from the tools is quantification of intensity oscillations over time. In certain conditions there could be an intensity oscillation in the cells due to certain protein expression that leads to such oscillations, the biological question of interest is if such oscillations are stable and if so what is the period of the oscillation [Ines]. Using our tool intensity of individual tracklet can be obtained which is then Fourier transformed to show the oscillation frequency if any. With this information we can see the contribution of each tracklet in the intensity oscillation and precisely associate the time when this oscillation began and ended.

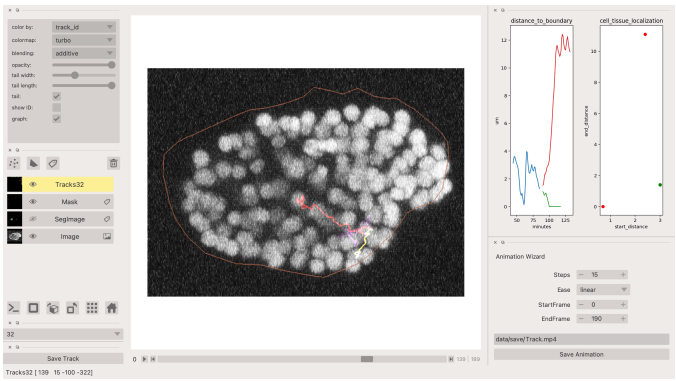

Fig. 8: *Parent cell after division, one daughter cells moves inside while other stays close to the boundary.*

REFERENCES

[Stardist] U. Schmidt, M. Weigert, C. Broaddus, and G. Myers,Cell detection with star-convex polygons, in Proceedings of MICCAI'18, 2018, pp. 265-273.

[Unet] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation, in Proceedings of MICCAI'15, 2015, pp. 234-241.

[Ines] Lahmann I, Brohl D, Zyrianova T, et al. Oscillations of MyoD and Hes1 proteins regulate the maintenance of activated muscle stem cells. Genes & Development. 2019 May;33(9-10):524-535. DOI: 10.1101/gad.322818.118.

[TM] Tinevez JY, Perry N, Schindelin J, Hoopes GM, Reynolds GD, Laplantine E, Bednarek SY, Shorte SL, Eliceiri KW. TrackMate: An open and extensible platform for single-particle tracking. Methods. 2017 Feb 15;115:80-90. doi: 10.1016 j.ymeth.2016.09.016. Epub 2016 Oct 3. PMID: 27713081.

DRAFT