Analysis of cell morphology and fluorescence / microfluidics

Alexandros Papagiannakis, Christine Jacobs-Wagner lab, Sarafan ChEM-H, Stanford University, 2023

This notebook provides an analysis example of the cell morphology and fluorescence statistics from the segmented and tracked cell labels in microfluidics. The functions applied in this analysis are included in the *microfluidics_analysis_functions_ghv.py*.

In [ ]:
```
import pandas as pd
import pickle
import os
import sys
import warnings
warnings.filterwarnings('ignore')
```

1. Adding the functions/classes to the path

In [ ]:
```
# includes all functions used for the analysis of the Brownian statistics
module_path = os.path.abspath(os.path.join('..'))
if module_path not in sys.path:
    sys.path.append(".../microfluidics_analysis_functions_ghv.py")

# Check the path to confirm the analysis functions are included
print(sys.path)

%run "....\microfluidics_analysis_functions_ghv.py"
```

2. Variable definition

In [ ]:
```
results_path = ".../results" # the results folder to store the variables
images_path = ".../images" # the folder with all the images for all the fluorescence channels, microfluidic channels and xy positions
fluorescent_channels=[1,2,3]  # examples of the labels of the analyzed channel
                              # e.g. 1 for phase, 2 for GFP, 3 for mCherry
half_window=5   # length of the vector used for the cross product estimation to determine the sign of the cell width position
edge_width=8   # the edge width of the cropped microdluidic channel image used for background estimation
every_nth=3  # the frame interval of the fluorescence images (e.g. 3 for every 3rd phase contrast image)
```

In [ ]:
```
cell_dataframe = 'dataframe that includes the tracked cell labels'
```

3. Get complete cell division cycle trajectories

A complete cell division cycle trajectory corresponds to the trajectory of a cell that was born by another trajectory and gives birth to another trajectory after cell division.

The cell division cycle phase will also be determined, with 0 corresponding to birth and 1 to division.

In [ ]:
```
# determine the mother cell for each trajectory
cell_dataframe, mother_cell_dict = get_mother_cells(cell_dataframe, results_path, area_percentage=(1.5, 2.5))
# determine the cell division cycle phase
cell_dataframe = get_cell_cycle_phases(cell_dataframe)
# get the complete cell division cycle trajectories
mothers_with_mothers = get_mothers_with_mothers(cell_dataframe)
```

4. Draw the medial axis for each single cell

In [ ]:
```
# number_of_microfluidic_channels is the total number of channels across all xy positions
# The algorithm can identify the different channels by their position and experiment labels in the folder
for index in range(0,number_of_microfluidic_channels):.
    all_medial_axis(results_path, index, crop_frame=3, resize_factor=10, order=2, smoothing=3000, end_cap=3, show=False)
# This function can also be ran in parallel and saves the medial axis dictionaries in the results folder
```

5. Object (e.g. nucleoid) segmentation

The fluorescence in any of the fluorescent channels (e.g. channel 3 in this example) can be used to segment fluorescently labeld objects within the cell masks (e.g. the nucleoid in this example).

In [ ]:
```
hu_masks_dict = segment_all_nucleoids(images_path, channel=3, cell_dataframe=cell_dataframe,
                                      log_adaptive_params=[2,1000,80,2,9,-2,0], min_mask_size=15, hole_fill=5, edge_width=8,
                                      save_path=results_path)
```

6. Polarity calculation

The polarity (1 or -1) of the segmented and tracked cell division cycles can be calculated from the polisitions of the new and the old poles relative to the cell center. This polarity can be used to orient all the relatice cell coordinates such that negative cell lengths are oriented towards the old pole and positive cell lengths towards the new pole, with zero indicating the cell center.

In [ ]:
```
# the medial axis dict is a collection of the medial axes defitinions for all the segmented cells
medial_axis_dict = load_medial_axes(results_path)
cell_dataframe, polarity_dict = apply_polarity(cell_dataframe, medial_axis_dict)
```

7. Map pixels on the medial axis

Knowing the medial axis and the polarity of each single cell, all cell pixels can be mapped on the central line, along the cell length and across the cell width.

In [ ]:
```
for index in range(0,number_of_microfluidic_channels):
    run_oned_estimation_single(results_path, images_path, half_window, edge_width, fluorescent_channels, index, every_nth)
# This function can also be ran in parallel and saves dictionaries with the relative pixel coordinates in the results folder
```

8.Calculation of whole cell fluorescence statistics for the fluorescent channels

The average cell fluorescence (concentration) is returned for each single cell, as well as the cell division cycle statistics (mean and standard deviation).

In [ ]:
```
cell_dataframe = apply_all_fluorescent_stats(images_path, results_path, channels=[2,3], every_nth=3, edge_width=8)
```

9. Calculation of the fluorescence statistics within the segmented cell objects (e.g. within the nucleoids)

In [ ]:
```
cell_dataframe = apply_object_fluorescence(results_path, images_path, channels=[3], every_nth=3, edge_width=8,
                                           min_mask_size=5, object_mask_dict_suffix='nucleoid_masks_dict')
```

10. Calculation of the fluorescence statistics within cell length sectors

The tuples include cell length fraction ranges within which the mean fluorescence (concentration) is quantified. Zero corresponds to the cell center and 1 to the poles.

The asymmetric_list variable includes the cell length ranges, taking polarity into account (two mean fuorescence values are returned, one for each cell half).
The symmetric_list does not consider polairty (one mean fluorescence value for both cell length ranges adhacent the cell center).

In [ ]:
```
cell_dataframe = get_single_cell_asymmetries(cell_dataframe, results_path,
                                             asymmetry_list=[(0,0.1),(0,0.2),(0,0.25), (0.25,0.5), (0.5,0.75), (0.75,1), (0.5,1), (0,0.5), (0,1), (0.45,0.55), (0.4,0.6)],
                                             symmetry_list=[(0.05),(0.1),(0.2),(0.25),(0.5),(0.75),(0.25,0.5),(0.5,0.75),(0.75,1),(0.5,1),(0.25,1),(0.25,0.75),(0.45,0.55),(0.4,0.6)])
```

11. Map the positions of diffraction limited spots on the medial axis

In [ ]:
```
# the patrticle_df is a Pandas dataframe which includes the detected particle positions. It is the result of applying the particle_tracking_microfluidics class
cell_particle_df = map_particle_positions(cell_dataframe, particle_df, half_window, results_path)
```