TÉCNICO LISBOA

MEMORANDUM

From: Berke Duarte dos Santos

Student ID: ist195923

Degree: Master's in Biomedical Engineering

Thesis Title: Assessing *Escherichia coli* Motility as a Bioindicator for Heavy Metals and PFOA Contamination in Water: Optical Analysis of Biological and Chemical Responses to Contaminant Exposure

Last Update: 31 October 2024

2D pre-processing and post-processing

The processing of the 2D motility data was divided into four main steps: (step 1) pre-processing of files; (step 2) creation of Motion History Images (MHIs); (step 3) post-processing of files; and (step 4) kinematic data extraction from MHIs. The python scripts developed for steps 1, 2, and 4 are explained in detail in this document.

Step 1

Step 1 of the processing pipeline involved the pre-processing of the original recordings, each saved as a series of multiple tif-images (i.e., frames). The python script for this step (`pre_processing.py`) was developed with the help of AI (i.e., ChatGPT and Perplexity AI). Its purpose is to resize, convert, as well as re-organise the original images recorded. The script entails several distinct functions, but only requires a parent directory as its main input. The parent directory selected should always refer to a main folder containing subfolders for each recording set. For example, one recording set could contain the recordings of the samples with 100 and 50 mg/L of zinc chloride, after 0.1, 1, and 4 h (with each recording saved as tif-images within a corresponding subfolder, together with the tif-metadata XML files). The folder structure required for a succesful implementation of the script is summarised in Figure 1(a).

Within the pre-processing script, the `resize_and_convert_tif_images` function iterates through the main parent directory and its subfolders to identify tif-images. The function then resizes these images to 2048 x 2048 pixel-sized images, converts them to grayscale, and saves a copy of each image into a new folder saved within the recording set directory (**change** folder). Inside the change folder, the images are saved in subfolders corresponding to their respective recordings (inside a raw folder). During the process of copying the images, each file is renamed while retaining essential information provided by its original name. The information retained includes: the

type of sample (e.g., 100 mg/L zinc chloride), its incubation time (0.1, 1, 4, or 24 h), the number of the sample (triplicate 1, 2, or 3), as well as the number of the recording (recording 1, 2, 3, or 4). Subsequently, the `analyze_subfolders` function examines each tif-metadata XML file, extracting acquisition times to calculate the frame rate of each recording. At the same time, the code also computes the number of images required for specific timestamps. The timestamps can be altered by the user, but the defaults were set to 2, 5, and 10 seconds. The output from this function for each recording folder is logged into a text file, which is ultimately saved within the recording set directory after all of the recording subfolders have been analysed. Finally, the `process_image_copying_and_renaming` function uses the saved text files, in particular the information regarding the number of images required for each timestamp, and copies that specified amount (of each recording) into three new folders (one for each timestamp), which are once again saved within the recording set directory. The final three folders are named shortened_2, shortened_5, and shortened_10, and contain the subfolders for each one of the recordings. The tif-images in these subfolders are also saved in a raw subfolder, since these are all copied from the previously mentioned change folder. Finally, the `pre_processing` function is responsible for integrating all of the aforementioned processes. It should be mentioned that the final order of the images/frames in each recording folder remains unaltered throughout all of the pre-processing steps to avoid compromising the ensuing motility analysis. The final folder structure (after all the pre-processing steps are applied) is provided in Figure 1(b).

Step 2

The creation of MHIs was performed with the use of the Holographic Examination for Lifelike Motility (HELM) package. The MHIs were created for each one of the subfolders saved within the shortened (_2, _5, and _10) directories. A simplified look of the folder structure following MHI creation is depicted in Figure (the MHIs were always saved in the validate folders, within each recording subfolder in the shortened directories).

Step 3

After the MHIs were created, the following step was to post-process the output files. Once again with the help of AI, a python script (`post_processing.py`) was developed to organise, analyse, and detect features in the images stored in the validation directories created by the MHI creation process (step 2). Several functions are part of this script, which requires a parent directory as a mandatory input.

The script starts with the `process_blob_detection` function, which is responsible for enhancing image contrast using CLAHE (Contrast Limited Adaptive Histogram Equalization), as well as performing blob detection. The blob detection process is able to detect bright and dark blobs found in the two images saved within the validate subfolders (i.e., the first and median images). The main parameters of detection of the blobs can be easily changed by the user, and includes: the minimum and maximum intensity threshold for both bright and dark blobs, as well as the minimum and maximum area of both sets of blobs. In addition, the function also generates a results summary for each processed file, detailing the number of detected dark and bright blobs, as well as the total blob count (bright + dark blobs). These results are logged into

a text file that is saved together with the annotated copies of the original images in the blob-detection folder, within each shortened directory. The `copy_mhi_files` function copies all of the MHI files from the validate subfolders into a new folder, which is also saved within the shortened directory. Finally, the `organize_analysis` is responsible for creating a dedicated analysis folder for each original shortened directory (i.e., shortened_(2,5, or 10)seconds_analysis folders) within the main parent directory. To these new directories, the function copies the content from both the blob-detection and MHIs folders (annotated images and MHIs, respectively). The function is also responsible for extracting the contents of the text files in the blob-detection folders (i.e., blob counting results), and converting them into a csv file. This csv file only contains the information regarding the filename, and the corresponding total blob count (of the first images). At the end, the `run_all_functions` function iterates through the main directory structure, sequentially integrating all of the aforementioned functions ensuring the correct processing of the results. The final folder structure (after all the post-processing steps are applied) is shown in Figure .

Step 4

The final phase in processing the 2D motility data involved extracting kinematic information from the MHIs previously created. After the segmentation of the tracks in the MHIs of interest, the following steps were sequentially performed (the ds python files described below are available at `https://depositonce.tu-berlin.de/items/8eacee77-81ef-4e99-a650-927063d61b36`):

1. Converting and saving each one of the segmented tracks as npy files, using the ds2 python file. The new npy files are all saved within the tracks folder, which itself is found in the validate folder;

2. Optimization of the parameters necessary for subsequent blob detection, using the ds3 python file. The blobs are only checked in a single frame of the original pre-processed recordings (found in the raw subfolders, within the shortened directories);

3. Blob detection in all of the frames of a given recording applying the previously optimized parameters of detection. The ds4 python file was employed for this purpose. The script uses the aforementioned optimized parameters to detect blobs across multiple frames of a recording. The blobs are then saved (i.e., their x,y-coordinates, size, and timepoint) as an npy file inside a new folder, created within each recording's raw directory.

4. Alignment of the MHI tracks with the detected blobs (across all frames of the recording) utilising the ds5 python file. The script checks the spatial and temporal alignment (for each recording) by using the information found in three different npy files: (1) the original MHI npy, which was obtained when the MHI itself was created; (2) the npy file resulting from the blob detection addressed in the previous item of the kinematic processing step; and (3) the npy files of the track segments, which was the first stage of the kinematic processing step (described above). The script then saves the validated (i.e., aligned) blobs for each track in new npy files, within a new subfolder in the validate directory of the recording (MHI_Tracks);

5. Extraction (i.e., copy) of all the tracks from all of the analysed recordings to a new folder location, by implementing the ds6 python file. After all of the desired recordings have been analysed following the steps described above, this script is responsible for compiling the npy files in the MHI_Tracks folders, into a single main folder (all_npy_tracks_NEW) saved in the shortened directory. It should be mentioned that multiple subfolders are created within the all_npy_tracks_NEW folder location (one for each recording of a particular recording set);

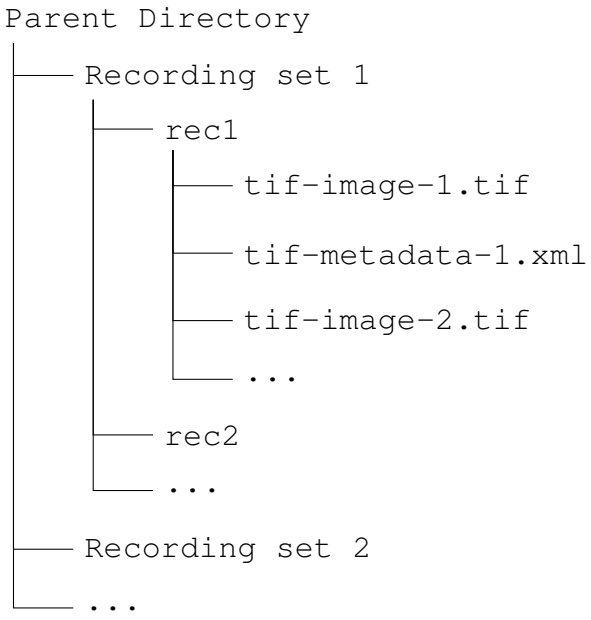
6. Extraction of the acquisition frequencies using the ds7 python script. Two directories are provided to this script. One of the directories refers to all of the npy track data (all_npy_tracks_NEW folder), and the other refers to the main recording set directory (containing the recordings being analysed). Within each recording of the latter directory, the script extracts the acquisition times from the first and last XML files, and calculates the time interval between both. Through the division of the total number of XML files by the calculated time interval, the script is able to provide the frequency (in Hz) of that same recording. The other directory provided comes into play to ensure that only the XML files from recordings with corresponding npy files are parsed by the script. The results from this script are saved as a text file (average_frequency.txt) in the corresponding recording subfolder within the all_npy_tracks_NEW folder directory;

7. Filtering of the tracks, conversion of the time intervals between blobs into seconds, and conversion of the xy-coordinates of the pixels into μm. These three objectives were achieved using the ds8 python script. The script receives one directory as input that corresponds to the all_npy_tracks_NEW folder directory. It uses the directory to extract the acquisition frequencies (previously saved), in order to filter the npy files saved in each of the recording subfolders (saved in that same directory). The idea is that the filter retains only the largest blob at each timepoint, which directly correlates to the most likely position of the cell (i.e., the blob) along a track. The filtered data is then saved as a new npy file in the same directory as the unfiltered data. Additionally, the script is also responsible for converting the time intervals (between blobs) to seconds, using the previously calculated acquisition frequencies. The coordinates (x and y) are also converted to μm based on the dimensions of the original image files. The converted data is saved into a new and final npy file in the same directory as mentioned before;

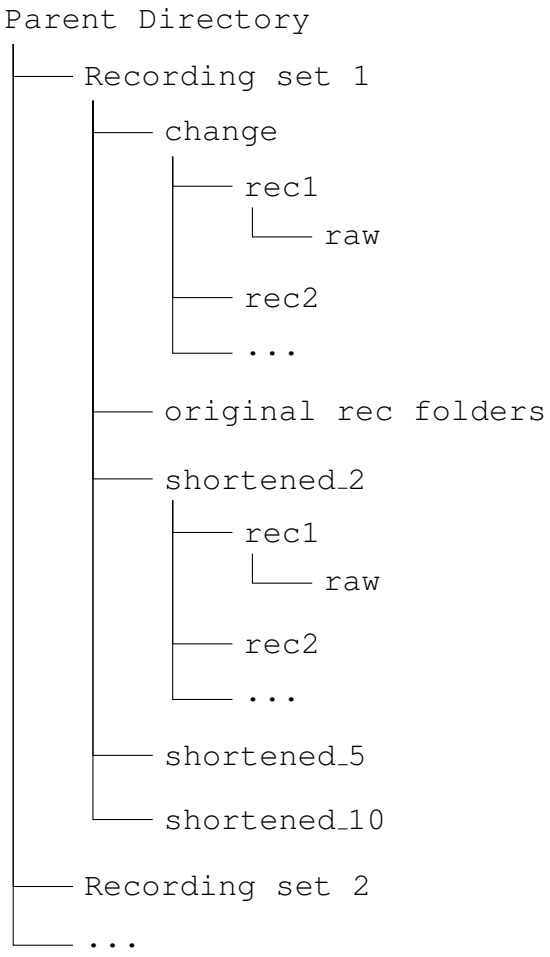
8. Analysis of the filtered and converted tracks regarding all motility parameters of interest, with the use of the ds9 python script. The script essentially processes the filtered track data, in the aforementioned npy files of each recording of interest, by extracting and analysing key kinematic features. The script first filters noise by applying a moving average filter, followed by the calculation of the velocities between consecutive timepoints (distance over time). Speeds/velocities above 60 μm/s are filtered out by the script. The script also computes the ratio of the standard deviation to the average of calculated velocities (**speed dynamic**). Another relevant feature extracted by the script is the **straightness index**. The displacement of the blobs along its track is another key parameter analysed, which is accomplished through the detection of significant variations between consecutive displacement vectors. Essentially, the displacement analysis makes use of the dot product between two consecutive vectors to calculate the angle estab-

lished between them. Variations are only recorded if the angle measured exceeds 30°. In addition, the script filters out displacement vectors that are shorter than 0.01 μm (i.e., norm ¡ 0.01), and calculated angles (between two vectors) that are less than 0.1 seconds apart from each other, with the exception of the first change in direction. The total number of direction changes is divided by the duration of the corresponding track, and the result is saved as the **direction change rate**. At the end, all of the mentioned kinematic features associated with the movement of individual blobs are saved in a text file within each recording subfolder in the all_npy_tracks_NEW directory provided. Speed plots for each one of the tracks are also saved within the same directory, as png images.

A simplified tree structure of the output directory, following all of the described kinematic analysis steps, is provided in Figure 4.



(a)



(b)

Figure 1: Simplified folder structure before (a) and after (b) pre-processing.

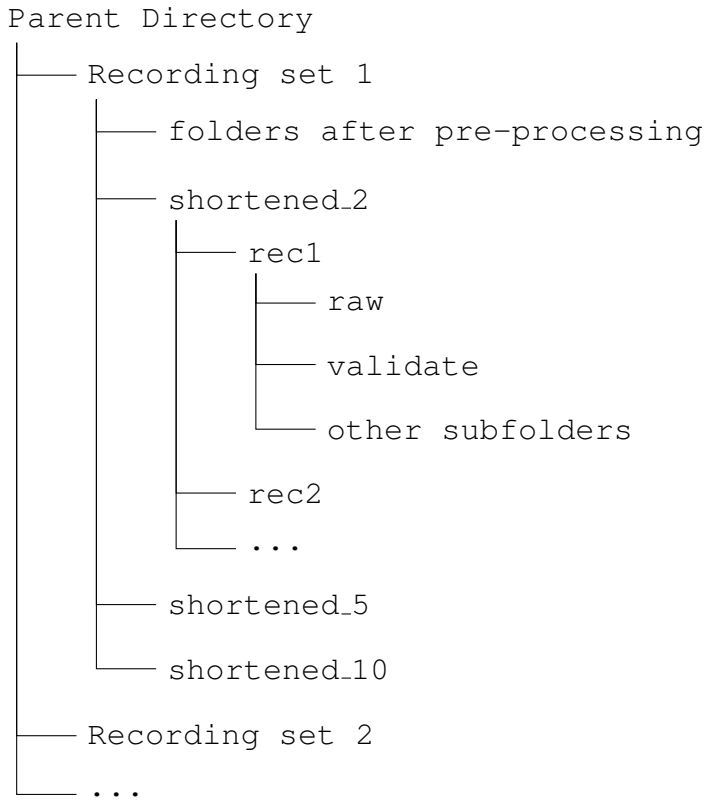


Figure 2: Simplified folder structure after MHIs creation (before post-processing step).

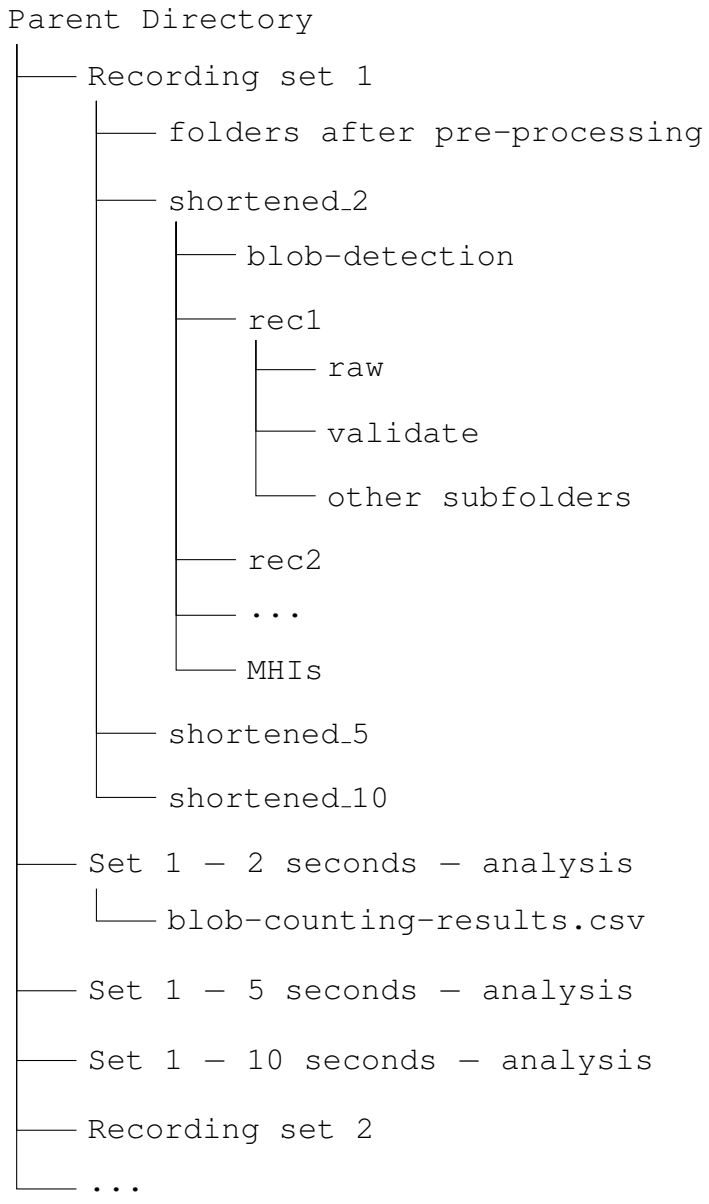


Figure 3: Simplified folder structure after post-processing.

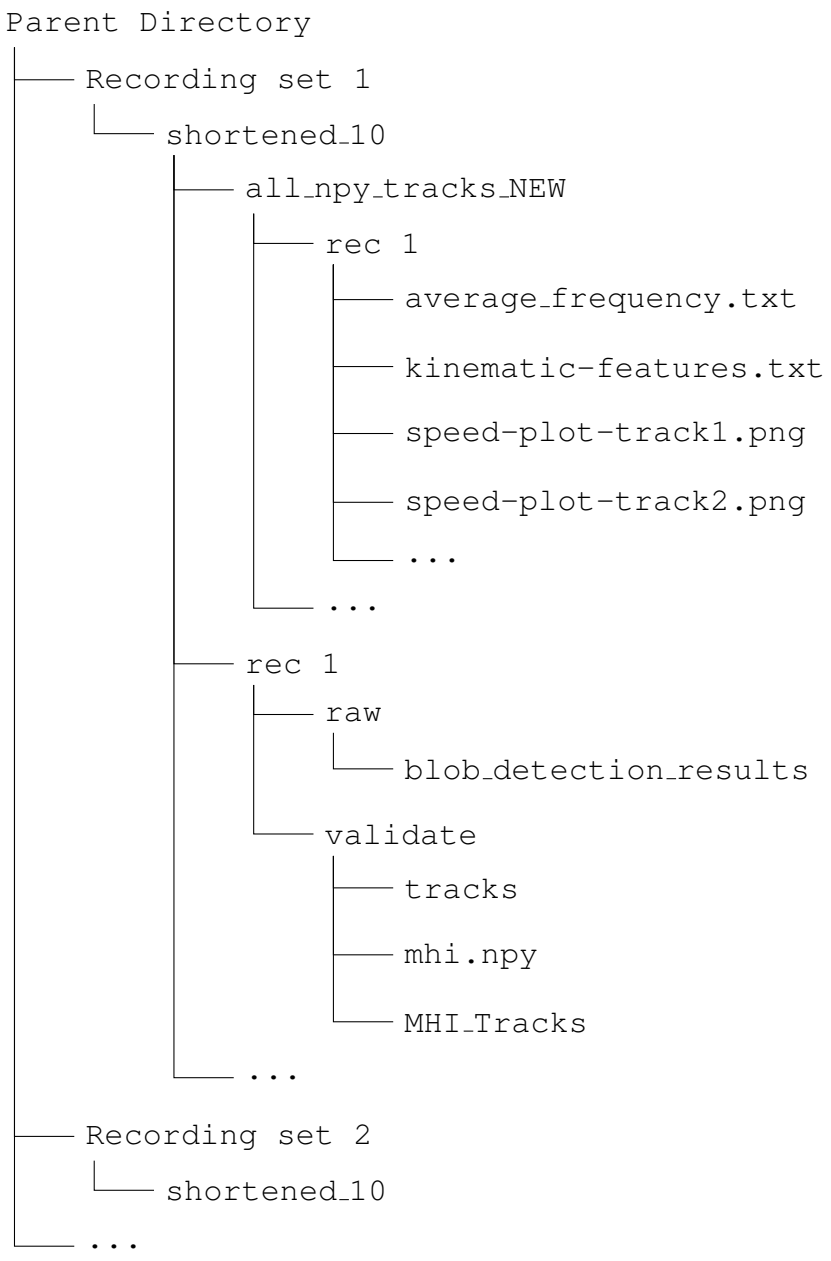


Figure 4: Simplified output directory structure following kinematic data extraction and analysis from processed data. The shortened_5 directory was chosen rather than the other two shortened folders, since the tracks in these recordings tend to be more suitable for analysis.