Project 1

ME-480, Autumn 2023

Malo Lemmel, Arthur Eglin, Victoire Tchatat, Dillon Zadoks, Jolan Fath

1 Introduction

The goal of this miniproject is to segment and track bacteria on microscopy movie files to analyse their collective motion and speculate on the functions of mechanosensing in group behavior.

The data provided are **.tiff** movie files of Pseudomonas aeruginosa (PAO1 ΔfliC) strains that have been genetically modified. Each of these strains lack the fliC gene that allows the synthesis of the flagellae to allow the bacteria to swimm. That way the bacteria cannot swim anymore and can only move by twitching motion via their type IV pili. The different strains are the following:

1. PAO1 ΔfliC: these cells can twitch in all directions
2. PAO1 ΔfliC ΔpilH: these cells lack the pilH gene, which controlls the polarization of the extension motor pilB of the pili by allowing reverse motion, the cells constantly move forward
3. PAO1 ΔfliC ΔpilG ΔcpdA: these cells lack the pilG gene that controlls the extension motor pilB of the pili by favoring forward motion, and lack the cpdA gene that controlls the degradation of cAMP to AMP, the cells constantly reverse their twitching direction.

Each strain is recorded in dilute or dense conditions, for each condition, the image parameters are the following:

1. dilute:
   (a) 1 s frame interval
   (b) 333 μm x 333 μm
   (c) pixel size 0.1625 μm
2. dense:
   (a) 3 s frame interval
   (b) 374 μm x 374 μm
   (c) pixel size 0.1625 μm

In both cases, cells were grown at the agarose-glass interface for 2 h starting from high cell density. These images were all acquired by phase contrast microscopy.

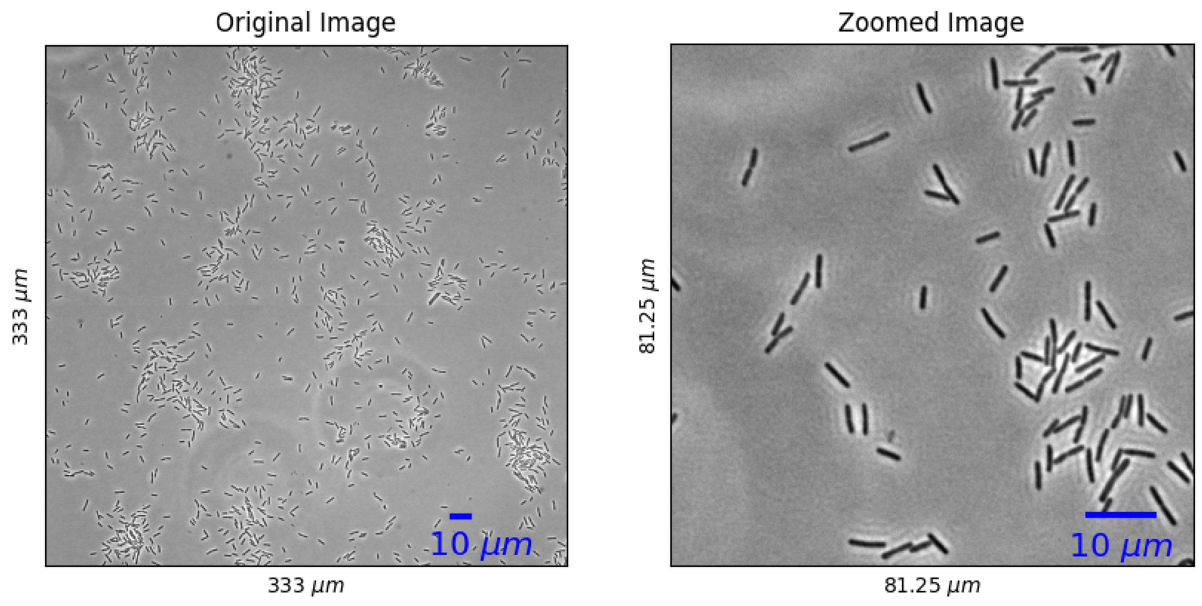

Figure 1: Raw data from the PAO1 ΔfliC ΔpilG ΔcpdA strain

2 Segmentation

Segmentation helps in distinguishing individual cells from the background and from neighboring cells. This is essential for accurately tracking the movement, division, and other dynamic behaviors of cells over time. Once cells are segmented, various quantitative features such as cell area, shape, and intensity can be measured for each identified cell. These features provide information about cell morphology and behavior. We attempted to segment the cells in the provided movies in three different ways to obtain an optimal result.

2.1 Basic segmentation

The algorithm begins by applying an intensity threshold to the image. This step involves setting a pixel intensity value above/below which pixels are considered part of the object of interest (e.g., cells) and below/above which pixels are considered background. This will give a binary mask of the image that can be represented as a "black and white" image. The threshold can be determined by looking at the distribution of the intensity values. The high peak contains all the background values, the values corresponding to the bacteria are the ones that come before this peak since the bacteria appear darker than the background on the image.

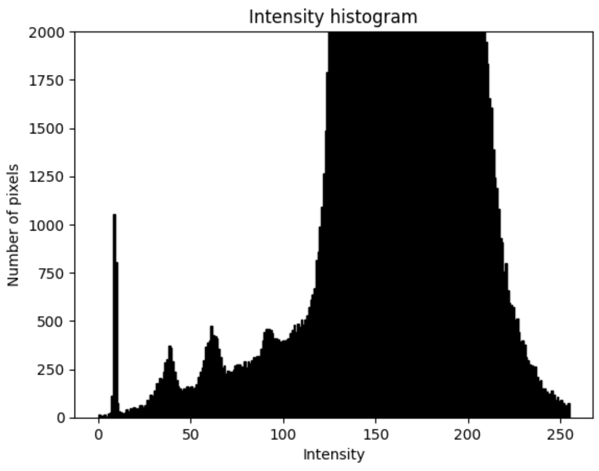

Figure 2: Histogram of the averaged values of all frames in the WT dilute movie

We will thus select 100 as the intensity threshold, since we need to consider all the small bumbs in the distribution. Indeed, the movie is affected by an intensity drift and taking a lower threshold value would get rid of some bacteria in frames at the end of the movie. In order to improve the segmentation, the intensity drift could be taken into account and the intensity threshold could be adapted according to the drift.

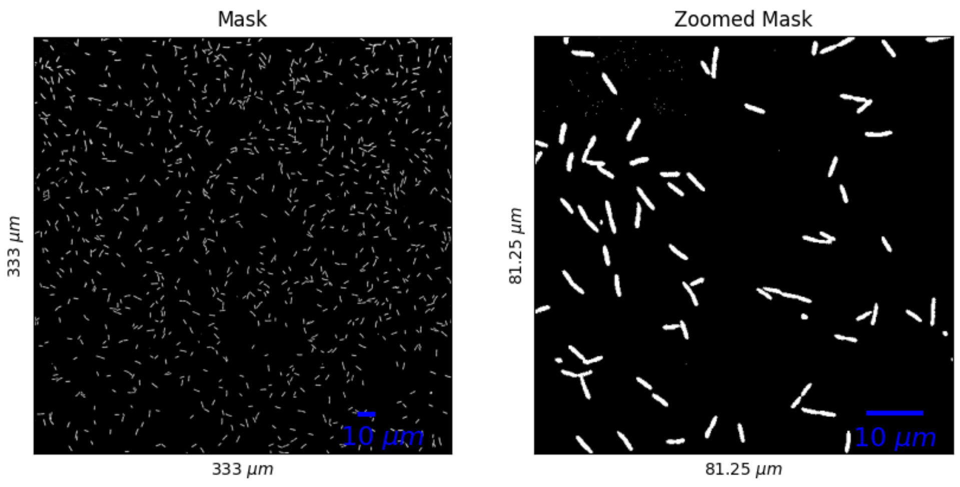

Figure 3: Binary mask from the PAO1 ΔfliC ΔpilG ΔcpdA strain, after intensity thresholding step

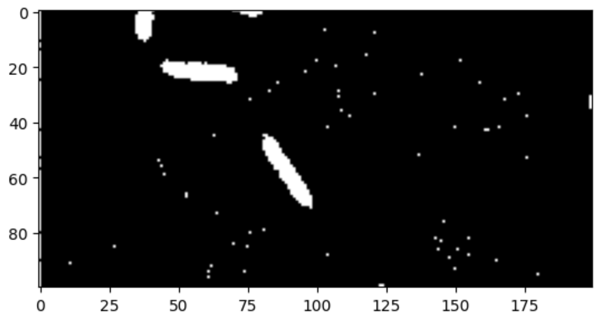

Figure 4: Artifacts in the mask after intensity thresholding

As we can see on the zoomed image on the right, the thresholding step keeps some pixels within what should be considered as background. This is due to white noise in the background signal and to the chosen threshold. To remove this we can apply a morphological erosion operation to the mask. This operation will as it name suggests, erode the shapes on the masks using a structuring element known as a kernel. This 'erodes' or shrinks the boundaries of the objects in the image and removes the previously mentioned noise.

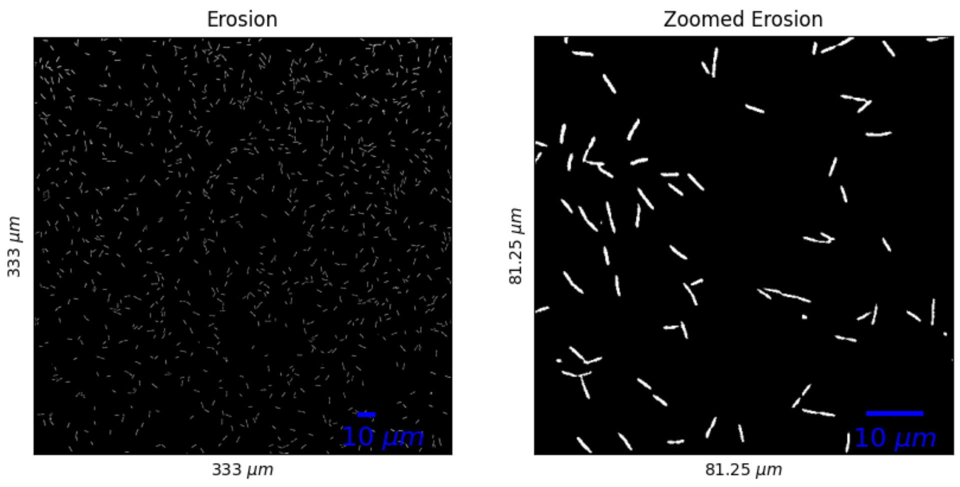

Figure 5: Binary mask from the PAO1 ΔfliC ΔpilG ΔcpdA strain, after intensity thresholding and erosion steps

We now have a mask free of residual noise, but the bacteria shapes are now too small, this is why a third operation is needed to reconstruct the original shapes. For this we use the morphological dilation operation that is the opposite operation of erosion, and will dilate the shapes on the image by the kernel size. If we take the same kernel as for erosion, we will end up with an almost perfect reconstitution of the image without noise.

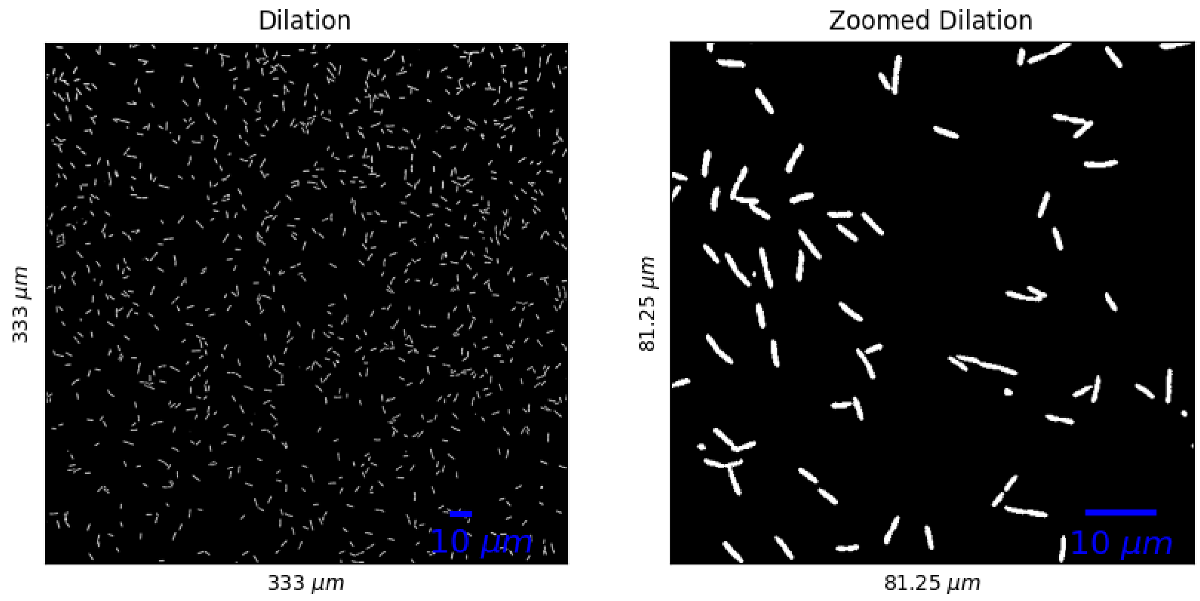

Figure 6: Binary mask from the PAO1 ΔfliC ΔpilG ΔcpdA strain, after intensity thresholding, erosion and dilation steps

The advantage of this method is its simplicity and low computation cost if not many morphological steps are required.
The downside is its resilience to intensity drift, and incapacity to separate objects that are very close to each other. This being said, this method performs quite well even on the dense files.

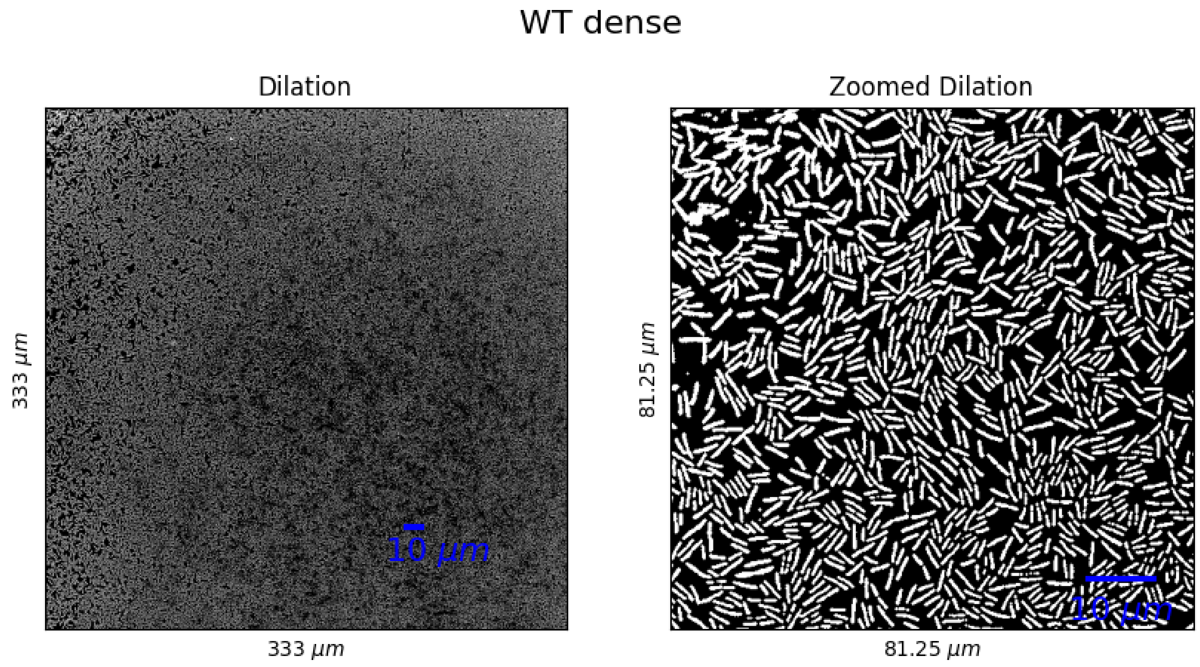

Figure 7: Binary mask from the PAO1 ΔfliC ΔpilG ΔcpdA strain, after intensity thresholding, erosion and dilation steps

2.2 Watershed algorithm

2.2.1 Watershed algorithm positive part

How does it work :

- Gradient Calculation: The algorithm starts by calculating the gradient of the image.
- Marker Placement: Based on the gradient information, "markers" are placed in the image.For cells segmentation, these markers are generally placed near the centers of the cells.
- Flood Fill: The algorithm then treats the gradient image as a topographic map, where the intensity values represent elevations. Starting from the markers, a virtual "flooding" process occurs. Pixels are flooded from the markers, and as the "water" (segmentation) rises, it fills the basins formed by the topographic gradients.
- Segmentation: As the flooding process progresses, the merging of the basins defines the segmentation boundaries. The result is a set of segmented regions, each associated with a specific marker. It's the restarting point of the segmentation for the new image for example

The watershed algorithm is useful with objects that are close together or touching. By simulating a flooding process, it assigns pixels to specific regions based on their proximity to markers, segmenting the image into distinct areas.

2.2.2 Watershed algorithm: Optimizing possibilities

- For Gradient Calculation: As we said the algorithm will place the markers on the center of the cell. It might not always be the best to represent the cell location because they are not symmetric. Furthermore, too many markers slow down the code and can result into over-segmentation
- For Gradient Computation: The noise in the segmentation has a huge impact on the calculation of the gradient. So the less noise, the better the gradient.
- For Preprocessing: This is the part the segmentation is going on. We are using filtering (threshold) and smoothing (erosion) during the segmentation for the preprocessing. Nonetheless, we can't apply them too much or else some cells simply disappear if the erosion is too high for example.

2.3 Results from the Segmentation : Question 1

2.3.1 Discussion for the segmentation part

Most of the problems encountered during the implementation of our algorithm are around these 3 factors :

- Overlapping Cells: Sometimes, the algorithm may struggle to accurately delineate individual cells within densely populated regions. Addressing this challenge is crucial for tracking and analyzing cell behaviors accurately and could be improved.
- Varied Cell Appearances:, The diversity in cell shapes, sizes, and intensity variations within an image complicate the segmentation.In our code it sometimes leads to some cells disappearing because their orientation give them a small area and are then eroded. It leads to having some sorts of negative positive (Machine Learning) that could be better treated with an AI probably.
- Dynamic Cell Morphology: Cells are undergoing dynamic morphological changes(here mostly migration). The segmentation algorithm similarly to what has been said before has some slight problems when the cross area section of cells are too small. As we are treating videos, sometimes some cells are disappearing after the segmentation or on the contrary appearing suddenly because of this.

3 Tracking

The second part of this project is the tracking of the bacteria in time, across the movie. This will provide useful information about the movement and orientation of the bacteria and will allow to analyze their motion behaviour.

The algorithm first computes the centroid of each detected shape in each frame. The function *cv2.findContours()* returns a list of the points drawing the contour of each shape. By iterating on this list, we can compute the area of each shape using the function *cv2.contourArea()*, and from this select the shapes that correspond to the size of a bacterium. After these steps, we compute the centroid coordinate via the function *cv2.moments()*. We now have a list of all the track particles at each frame of the video.

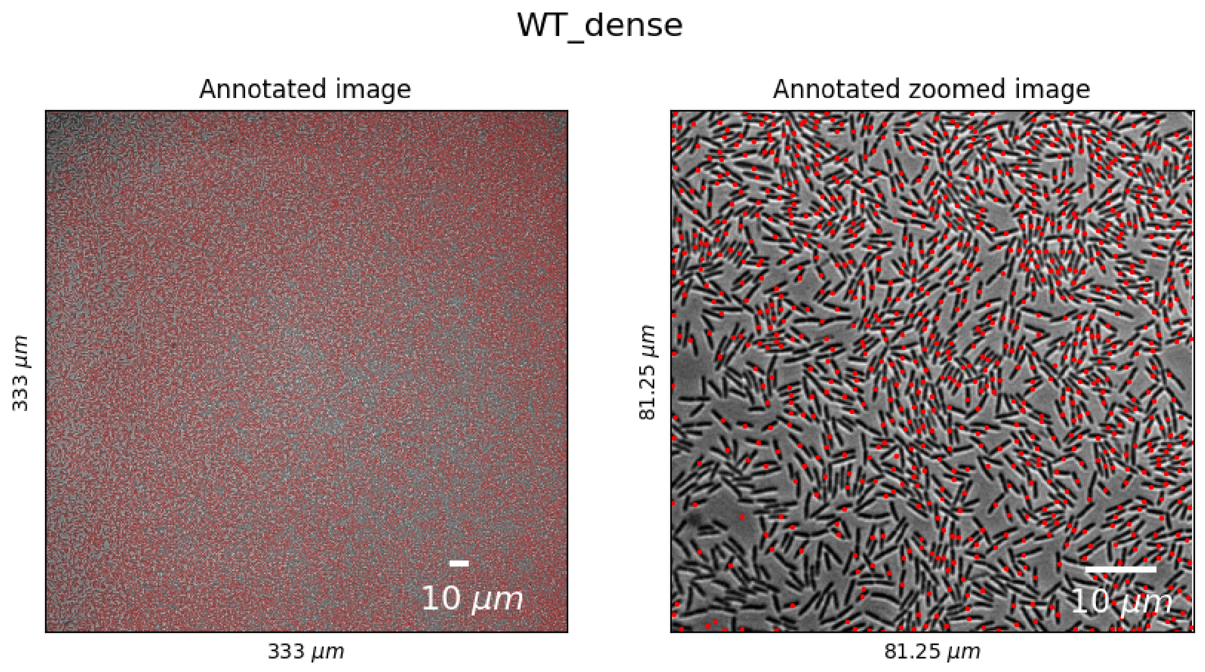

Figure 8: Raw image from the PAO1 ΔfliC strain, with positions of each tracked bacterium in the frame

We now need to match the position of each bacterium through the entire movie. This will allow to extract the positions of the bacteria over time and thus their motion, velocity and direction. To do so, we use the *tp.link()* function from the trackpy library that matches positions across the video. This function allows a high level of tuning to improve the results, which was not explore in depth here but could highly improve the overall result. However to reduce the error and the number of artifacts tracks, we can filter the output of the *tp.link()* function by frame length since we want to keep trajectories that ideally last for the entire movie. Due to the segmentation and matching error, chosing a too high frame window will omit a lot of interesting tracks so a good frame number is 20. This step greatly reduces the number of tracks (from more than 25'000 to less than 2000 for the dilute file) which is much more adequate for visualization and for extracting meaningful statistics.

We can plot the different trajectories over the first frame of the video. This is however quite unreadable. We will then concentrate on specific bateria.

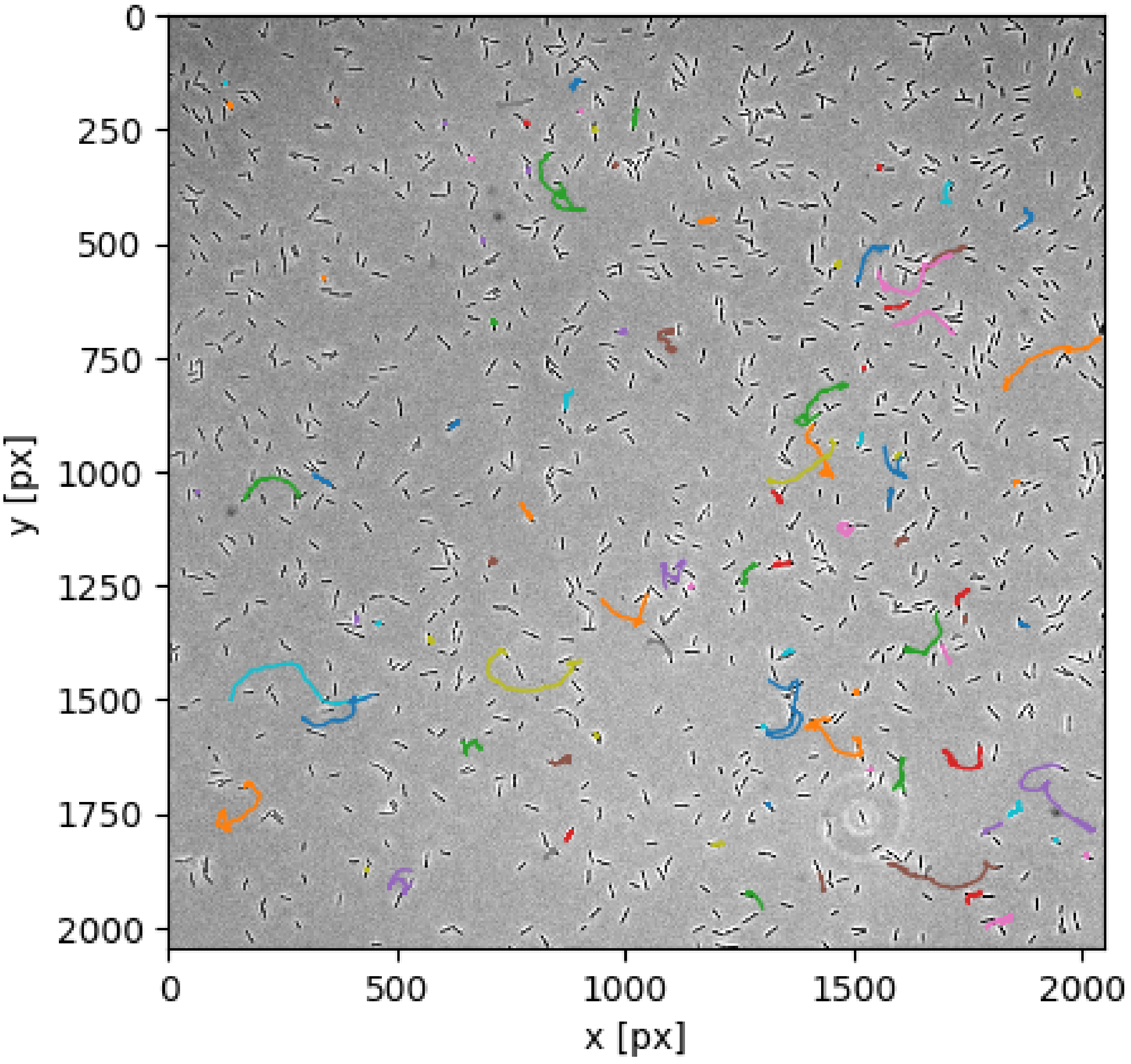

Figure 9: Filtered tracks from the PAO1 ΔfliC strain after basic segmentation method

To further correct the trajectories, we can look at the drift of the cells along the x and y axis. This drift can be induced by many factors, e.g. inclined surface, moving microscope, etc... In this example, we can detect a consequent drift along the x-axis that we will subtract to the trajectories to correct them.

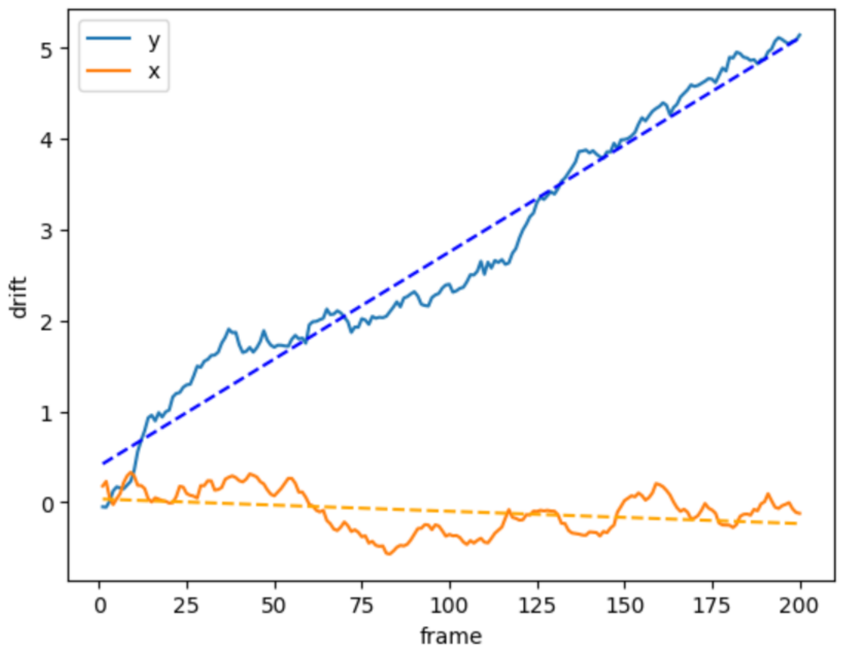

Figure 10: WT dense drift along the x and y axis

Trackpy also have already built-in tracking function, we tried the *tp.batch()* function. This tracking algorithm has the same procedure than the previous one. The goal is to get a centroid of the cell, then plot the different tracked cells with the drift.

After the classic tracking part we do a scatter plot (Fig.17) to allow us to visually inspect the distribution of masses and sizes among particles. Here we observe that there is a correlation between the 2 and it also indicates how the cells have a Gaussian distribution of sizes in the video. From this data the ML of the Trackpy algorithm chooses the cell that it won't take into account when doing the filter.

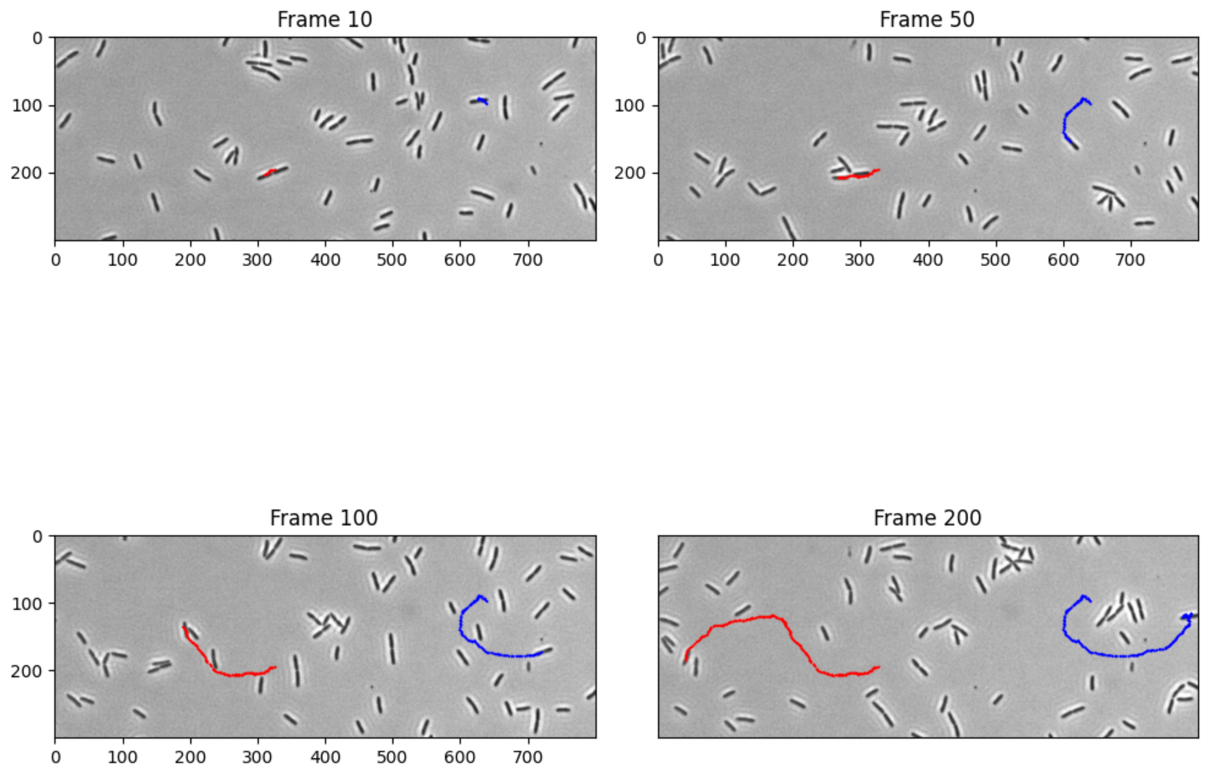

Figure 11: Selected tracks for visualization of the tracking in the WT dilute movie.

4 Analysis

4.1 Nematic Order

The means of the nematic orders in the different movies are the following:

- WT dilute: 0.26546246673120943
- pilH dilute: 0.24698068275596802
- pilG dilute: 0.2203001845327909

The nematic order varies from 0 for fully isotropic (no preferred direction) environment to 1 for perfectly ordered environment. Here we can see that the values are pretty low (between 0.2 and 0.3), which suggests a low level of structured orientation. However we can see a difference between the pilG sample and the WT and pilH samples. The pilG sample has a lower nematic order than the two others, which suggest a lower structured orientation. This makes sense since we know that the pilG lacking cells reverse their motions very frequently making them unable to move in a straight line with a preferred orientation.

4.2 Ensemble mean square displacement

The values for the ensemble mean square displacement (emsd) of each strain are the following:

- WT dilute: n = 1.703179, A = 4.654144
- pilH dilute: n = 1.919838, A = 5.225071
- pilG dilute: n = 1.54246, A = 0.833352

We see that for the pilG strain, the emsd is much lower than for the two other strain, which is understandable since pilG bacteria reverse their motion very often and thus do not make great distances.

5 Conclusion

With the simple implementation that we did and the results that we got, we can support the assumptions made in the introduction about the bacteria strains motion behavior.

However, to be able to confirm those assumptions, the segmentation and tracking algorithms, would need to be much more accurate. To do so, we could implement pre-trained machine learning models that would segment bacteria much more accurately. The best options would be to train a model on segmented data corresponding to the shape and size of our data. For this, we could create a U-Net model that would learn segmentation according to self made segmentation masks. To multiply the training data, we could use data augmentation steps such as rotations, mirroring, bluring, noise induction, etc...

Also, other statistics could be interesting to measure, such as the bacteria's velocity. To do so, we could compute the instantaneous velocity at each frame by the following formula:

$v_x = \frac{dx}{dt} = \frac{x_i - x_{i-1}}{t_i - t_{i-1}} = x_i - x_{i-1}$ $v_y = \frac{dy}{dt} = \frac{y_i - y_{i-1}}{t_i - t_{i-1}} = y_i - y_{i-1}$

The total velocity of the bacterium would be computed by taking the norm of the x- and y-velocities.

$v = \sqrt{v_x^2 + v_y^2}$