Politecnico di Torino

SORBONNE UNIVERSITÉ

LABORATOIRE Jean Perrin

UNIVERSITÉ FRANCO ITALIENNE | UNIVERSITÀ ITALO FRANCESE

Thesis program in Physics of Complex Systems

Swimming behavior of *Paramecium*

By

Geremia Fracella

Supervisors:
Alexis Prevost, Supervisor
Léa-Laetitia Pontani, Supervisor
Andrea Antonio Gamba, Supervisor
Dario Dell'Arciprete, Co-Supervisor

Politecnico di Torino - Sorbonne Université
2025

Abstract

In this thesis, we present our work aiming at tracking and studying the swimming behavior of *Paramecium*, a unicellular eukaryotic microorganism, in a free environment over long time scales, reminiscent of the run-and-tumble motion of some flagellated bacteria, as it consists of long runs interrupted by shorter reorientation events. To achieve this, we started to develop an automatic tracking system using a translational stage mounted on a microscope. We recorded and analyzed the trajectories of four *Paramecia* tracked for different time durations and extrapolated mean velocity, mean square displacement and avoiding reactions frequency from the data. Lastly, we briefly discuss how - by analysing more trajectories and changing the physical and natural conditions of the observations - a better understanding of *Paramecium* dynamics can be achieved.

Contents

Chapter 1

Context and motivations

Paramecium, a unicellular eukaryotic microorganism, belonging to the phylum Ciliophora, is one of the most studied microorganisms in the fields of cell biology and zoology (Figure 1.1). Characterized by its elongated shape and motile cilia covering its entire surface, *Paramecium* exhibits remarkable agility and precision in navigating aquatic environments as a highly swift organism [1].

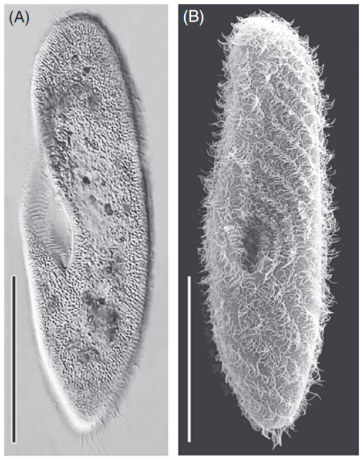


Fig. 1.1 (A) and (B) Light and scanning electron microscopic appearance of *Paramecium caudatum*. Scale bar in A and B = 100 µm.

1.1 *Paramecium* motion and avoiding reactions

In the absence of any obstacles, the motion of *Paramecium* is characterized by a helicoidal movement (it rotates along its major axis) interrupted by reorientation events called avoiding reactions (ARs). ARs can occur spontaneously or due to environmental stimulations (Figure 1.2, [2]).

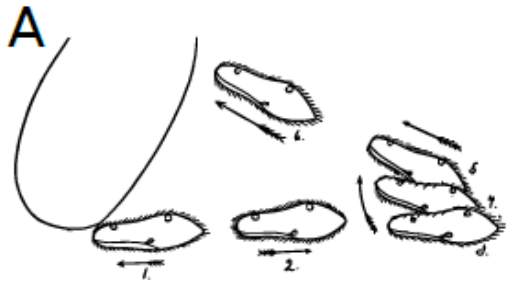


Fig. 1.2 (A) Diagram of a *Paramecium* performing an avoiding reaction against an obstacle, with successive positions marked from 1 to 6. The *Paramecium* interrupts its forward swimming when it collides with the obstacle (1) the cilia reorient, causing backward swimming (2) then an angular reorientation of its entire body occurs (3-5) when all the cilia have returned to their initial beating direction, the *Paramecium* swims forward again (6).

Initially, *Paramecium* detects various stimuli - chemical, mechanical, thermal and optical - thanks to specific receptors on its cell membrane. In response to these stimuli, the receptors generate a receptor potential, a graded depolarization proportional to the intensity of the stimulus.

When the receptor potential exceeds a critical threshold, calcium channels located on the cilia are activated, leading to a rapid influx of Ca^{2+} and the generation of an action potential. This action potential is characterized by an extremely rapid initial phase followed by potassium currents that restore the membrane potential to its resting level.

The localized influx of Ca^{2+} within the cilia results in an increased intraciliary calcium concentration, which reverses the direction of the ciliary beat. Under normal conditions, the cilia beat towards the posterior, facilitating forward movement; however, the Ca^{2+}-induced "ciliary reversal" causes the cilia to reverse their beat, propelling the *Paramecium* backward. This mechanism underlies the avoiding reactions, enabling

the organism to swiftly evade repellent or potentially harmful stimuli such as sudden mechanical contacts or adverse environmental changes [3].

1.2 Hydrodynamic regime

Our study focuses on a specific species of *Paramecia*: *Paramecium tetraurelia*, whose length and width are approximately 120-150 µm and 30 µm respectively.

The hydrodynamic regime in which *Paramecium* moves is characterized by viscous forces dominating over inertial forces, allowing its motion to be accurately described by low Reynolds numbers. In fact, the medium in which Paramecium is cultured -wheat grass powder solution- has density and viscosity nearly identical to water. When combined with the typical size and swimming speed of *Paramecium*, this results in a Reynolds number smaller than 1, of the order of 0.1, confirming that the system operates in the viscous regime [4].

This behavior is common not only to *Paramecium* but also to other microorganisms, such as bacteria, which similarly operate in a low-Reynolds-number environment.

1.3 History-dependent behavioral effects

This thesis aims at following a single *Paramecium* over long time scales to examine potential memory effects, i.e., whether the organism's behavior depends on its history.

A preliminary example of history-dependent behavior can be found in the study by Kunita et al., where measurements and modeling of the swimming patterns of *Tetrahymena* (a protist closely related to *Parame-*

cium) reveal that after being forced to swim in a circular arena for a period, the organism continues to exhibit a circular motion even when released into a larger arena. This observation suggests that the previous experience of the organism influences its subsequent behavior [5].

Previous studies have shown that *Paramecium* is capable of learning through classical conditioning. In a well-known experiment, *Paramecia*, which normally do not respond to a sound tone, began to trigger an escape response when the tone (e.g., 500 Hz) was repeatedly paired with an electric shock. This phenomenon indicates that the cell has modified its behavior based on experience, recording a form of "memory" that allows it to anticipate the noxious stimulus [6].

We have begun developing an automatic tracking system whose principle is to keep the organism centered in the field of view of a camera at all times, using an xy translational stage, since we observe that individual trajectories, velocities, and other features vary significantly, indicating different phenotypes.

Chapter 2

Materials and methods

2.1 Samples preparation

In Appendix A, detailed protocols can be found, outlining the preparation of the *Paramecia* medium, the bacteria medium (*Klebsiella pneumoniae*) on which *Paramecia* feed, and the protocols for culturing both *Paramecia* and bacteria. Once the culture of *Paramecia* has reached the stationary phase, we proceed with the sample preparation.

In the initial approach, the sample consisted of two rectangular glass slides attached together with double-sided scotch tape confining a drop of *Paramecia* suspension in a quasi-2d geometry (Figure 2.1). It is important to dilute the *Paramecia* solution to ensure that only a few of them are present in the drop, as a large number of *Paramecia* would make the tracking inefficient (see discussion in Section 3.2), as our objective is to follow them individually.

We began by recording videos of the drop of *Paramecia* with varying percentages of Protoslo, a viscous solution designed to slow down swimming protists such as our ciliate *Paramecium* (Appendix A), in order to facilitate the capture of the *Paramecia*.

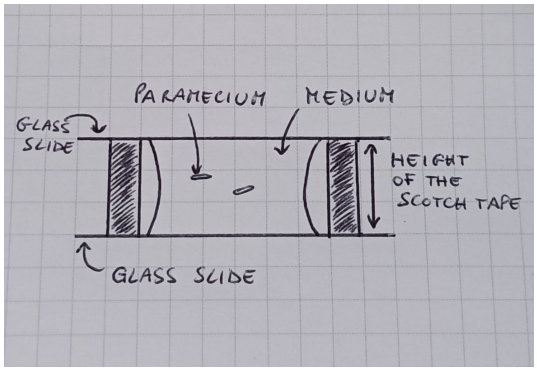


Fig. 2.1 Side view of the sample

2.2 Samples improvement

In the first approach, we used samples and droplets with small lateral dimensions, causing the *Paramecia* to reach the borders in a short time (Figure 2.2). Since we are using imagine analysis to locate the cells, detecting them could be rather difficult, which is why we aim to minimize contact with the borders by having much larger "pools". Additionally, with the scotch tape height being around 50 µm, the *Paramecium* was very confined in this region, inhibiting its natural helicoidal movement, and as a result, it stopped moving after a few minutes.

Fig. 2.2 Top view of the sample. A miscroscope glass slide (VWR) with dimensions 75 x 25 mm^2 and an cover slip (Epredia) with dimensions 55 x 25 mm^2 placed on top of it. A droplet of *Paramecia* between them. Scotch tape height of 50 µm.

We then modified the setup to create larger samples, using rectangular glass slides bigger than the previous ones. By testing the cells' behavior with different scotch tape heights, we found that an optimal height was

150 µm (Figure 2.3). This gives the cells more freedom of movement and extends their lifetime significantly.

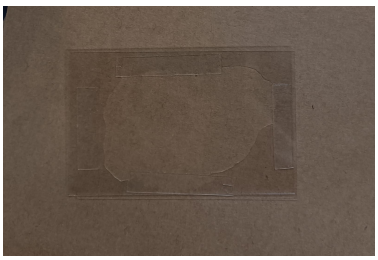


Fig. 2.3 Top view of the new sample. Two cover slips (Deckglaser) with dimensions 70 x 45 mm^2 stacked on top of each other. A droplet of *Paramecia* between them. Scotch tape height around of 150 µm.

Chapter 3

Automatic tracking setup

3.1 Preliminary version

Automated tracking systems for individual swimmers are being developed by a number of authors (see *e.g.* [7]).

For the automatic tracking of our *Paramecium*, we have been working on a system composed of a Windows 10 Pro based computer, a Hamamatsu ORCA Fusion BT camera, a Marzhauser translation stage and its controller, a National Instruments board (NI DAQ) that can generate TTL signals, and an Olympus BX51WI microscope (with 2X objective, yielding a spatial resolution of approximately 3 µm/px) (Figure 3.1).

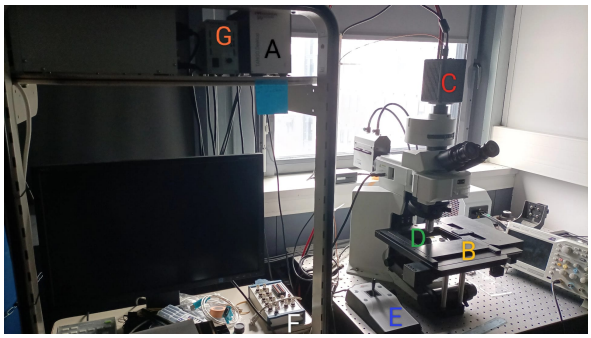


Fig. 3.1 Setup. (A) Marzhauser Tango controller (B) xy translational stage (C) Camera (D) Objective (E) Joystick (F) NI DAQ (G) Intensity light controller.

Communication between these elements is managed through a Matlab 2023a program, whose principle is to keep the organism centered in the field of view of the camera at all times. The operations executed in and the structure of the program are organized as follows:

1. Call the directories' paths for the functions used in the code, and create a folder to save the videos, data, and parameters of the experiment (on an SSD drive for fast saving).
2. Set the parameters: duration of the experiment, exposure time, frame rate, pixel/µm conversion.
3. Set the size of the square window used for the tracking.
4. Open and configure the stage controller using two functions: 'open_xystage.m' and 'config_xystage.m'.
5. Prepare the NI DAQ counter to generate a TTL output signal to trigger the camera.
6. Open and configure the camera parameters. Define a ROI (Region Of Interest) in which to track the *Paramecium*.
7. Open the preview window, and when a *Paramecium* enters the region (using a joystick to locate it), close the window and start the program.
8. The program calls a callback function (Appendix B) for each frame (in Matlab, a callback function is a function that is executed when a specific event or condition occurs, in our case the event is "image acquired"): it captures the image, creates a binary image by selecting an intensity threshold (chosen by maximizing the area of the mask superposed onto the brightfield image of the individual cell) where the *Paramecium* appears as a white object against a dark background, uses 'regionprops' (built-in Matlab function that performs morphological operations on binary images) to get the

centroids and areas of the regions, retains only regions with an area greater than a specified value (to eliminate "defects" like dust), selects the centroid nearest to the center of the ROI, calculates dx and dy (the distances from the center of the ROI in the x and y directions), and then uses the 'translate.m' function (written by us) to move the stage, keeping the *Paramecium* centered in the ROI.

9. Finally, it saves the data and parameters along with the corresponding video.

3.2 Problems encountered and solutions

This version of the program has several issues, ranging from the linearity and clarity of the code to the practicality of capturing "good" videos. Here, "good" means tracking the *Paramecium* for long periods without losing it from the ROI, losing it at the borders (*Paramecia* and borders would merge into a unified region, after thresholding the image), or mistakenly following another *Paramecium*.

Let's analyze the critical issues arising from the program by discussing some of the points given above in Section 3.1.

3. To improve the speed of the program, we can define a smaller window for tracking. Specifically, we reduced the window from a 576 x 576 px^2 square to a 288 x 288 px^2 one. This allows the 'regionprops' function to identify the centroids and areas of the regions within this smaller window, enhancing the code's performance by a few milliseconds.

4. Adjusting the 'config_xystage.m' parameters, particularly velocity and acceleration, can help in following the *Paramecium* without losing it (x and y stage velocities equals to 10 mm/s, x and y stage accelerations equals to 0.085 mm^2/s). Care must be taken with the acceleration value, as a high value can cause stage vibrations that affect the *Paramecium*'s motion. Indeed *Paramecia* are mechanosensitive ([3],[8]). The goal is

to find a balance between a high enough value to track the *Paramecium* effectively and a low enough value to avoid unwanted vibrations.

7. Actually, we don't need to initiate the *Paramecium* tracking using the manual joystick. We added a preliminary part to the callback function that automatically searches for the *Paramecia*. We need to account for the additional time the program requires for capture, but you can move the joystick after starting the program. When a *Paramecium* enters the region, tracking begins, and the joystick is disabled. Now, we can start tracking *Paramecia* moving at their typical velocities in aqueous solutions (maximum velocity $\sim$ 1 mm/s).

8. This is the core of the program, and it required some improvements. We divided this part into two subparts: one for catching the *Paramecium* and the other for the "centering problem." The program now waits for the *Paramecium* to enter a smaller region centered within ROI before starting tracking. Specifically, this new region is a square with a side length of 96 px. Due to the *Paramecia*'s velocity, starting tracking from a ROI corner might result in losing it after a few seconds.

Once a *Paramecium* is "caught", a new tracking section begins. This new version doesn't search for the *Paramecium* nearest to the center; instead, it looks for the one closest to the position of the tracked *Paramecium* in the previous frame. We thus no longer need to dilute the *Paramecia* solution very much, as the code is now clearer (it now prevents another *Paramecium* in its vicinity from being caught). Furthermore, it doesn't take additional processing time compared to the previous version.

An important technical aspect that cannot be overlooked is the 'translate.m' function's task. It includes the 'mor' command (provided by Marzhauser translational stage), which induces a relative movement from the actual position of the *Paramecium*. The 'mor' instruction must complete its movement before taking the new command. Moving the stage could take too long, causing us to lose the *Paramecium* or encounter unwanted oscillations (also due to the delay TTL pulse

at which it is sent). The solution is to reduce the real dx and dy by a factor of 10 (experimentally tested for 50 fps and these specific stage acceleration values). This approach achieves convergence similar to a PID (Proportional-Integral-Derivative) controller, but utilizes only the proportional term (a part of code of the new callback function code is provided in Appendix B).

Chapter 4

Data analysis and results

We experienced several difficulties in the observations: it is indeed challenging to keep the sample free of dust or other defects that can complicate tracking, and to achieve the right dilution to ensure an optimal number of *Paramecia* is also difficult. However, we managed to capture some videos to initiate the study of *Paramecium* trajectories.

Trajectories, velocity, mean square displacement, and avoiding reactions frequency have been calculated using a Python code that we wrote.

These videos are recorded at 50 frames per second. We'll report our first attempts to analyze four videos of different time durations of different *Paramecia*:

- Exp.0 —> 22 seconds
- Exp.1 —> 44 seconds
- Exp.2 —> 149 seconds
- Exp.3 —> 266 seconds (a bit more than 4 minutes) (click here for the video)

4.1 Trajectories and velocity

Let's start plotting the relative displacements dx and dy (for each frame) of the paramecium with respect to the center of the ROI.

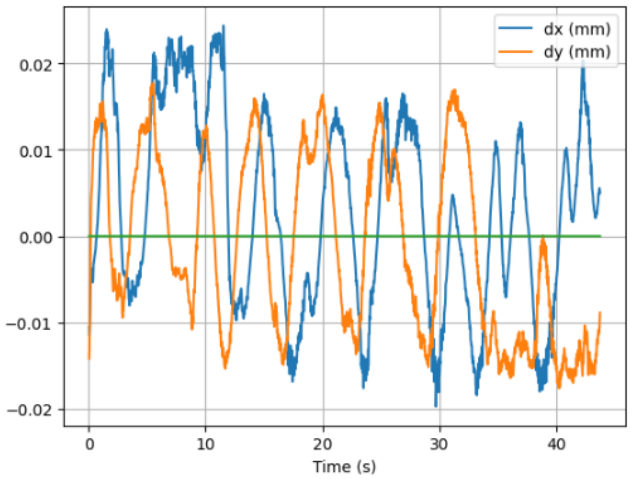


Fig. 4.1 dx and dy of Exp.1

Figure 4.1 provide an indication of the mean velocity of the paramecium. For an estimation, we can consider 0.02 mm as a result of the travelled distance $\sqrt{dx^2+dy^2}$. This yields $\frac{0.02\,\mathrm{mm}}{0.02\,\mathrm{s}} = 1\,mm/s$ (0.02 seconds since we are working at 50 fps).

It is possible to reconstruct a typical trajectory. Utilizing the 'cumsum' function (built-in Python function), we obtain a cumulative sum of the relative displacement frame by frame.

We remark that we are well aware that this is not the correct way to proceed and thus the reconstructed trajectories are a very crude approximation.

The saved values of dx and dy are reduced by a factor of 10. To reconstruct the real trajectories, we need to consider the real displacements that bring the *Paramecium* to the center of the ROI, therefore, it is necessary to find a way to determine the real position of the translational stage. As discussed in Section 5.1, we encountered unexpected problems

and timing issues to get the actual position of the stage, preventing us from correctly reconstructing the trajectories at 50 Hz and even at lower frame rates.

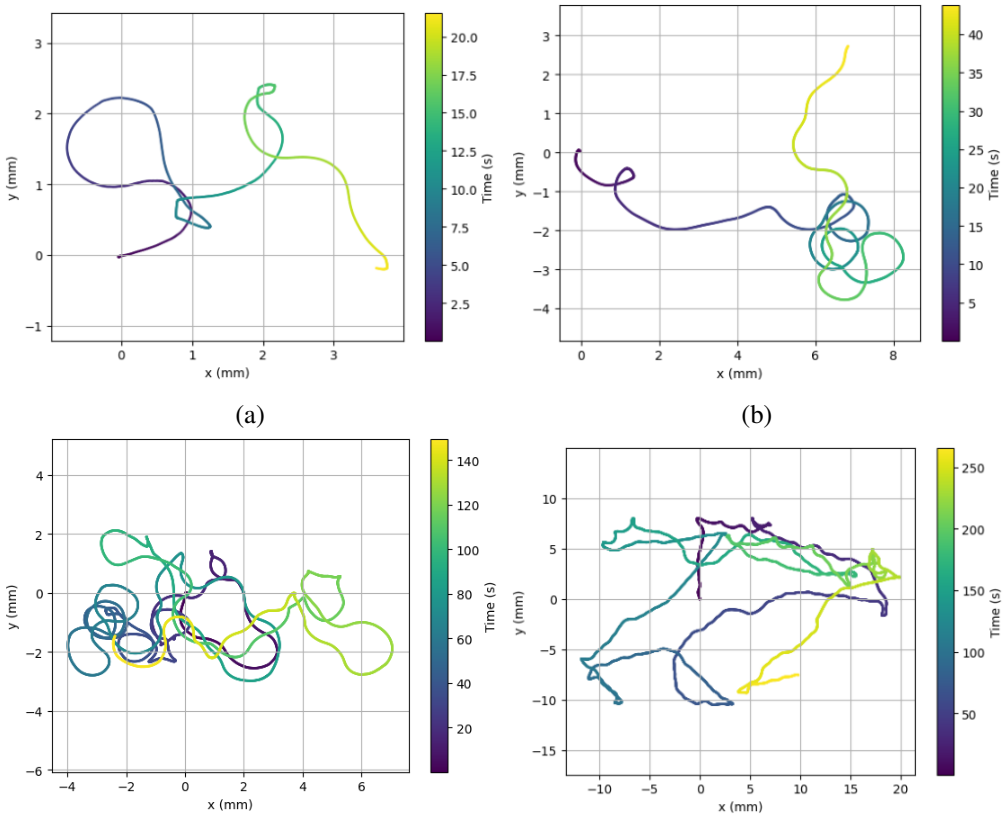


Fig. 4.2 Trajectories: (a) Exp.0 (b) Exp.1 (c) Exp.2 (d) Exp.3

From Figure 4.2, we observe the characteristic 'run and tumble like' movement of the paramecium and how it spreads over a larger region with increasing observation time.

Figures (a), (b), and (c) (Figure 4.2) suggest that a hydrodynamic interaction of the cell with the upper and lower boundaries could be at play here, explaining the highly curved trajectories. In contrast, Figure (d) shows a different behavior, suggesting that the *Paramecium* is moving in the middle of the boundaries [9]. Hence, there is a need to track it in a 3d environment (as recalled also in Section 5.3).

Although being affected by some inaccuracies, we determined the mean velocity of the four paramecia by summing the results of travelled distances for each time step and dividing the total by the duration of the experiment.

- mean velocity of the *Paramecium* of Exp.0 —> $\frac{14.99\,\text{mm}}{21.54\,\text{s}} = 0.70\,mm/s$
- mean velocity of the *Paramecium* of Exp.1 —> $\frac{33.44\,\text{mm}}{43.78\,\text{s}} = 0.76\,mm/s$
- mean velocity of the *Paramecium* of Exp.2 —> $\frac{104.33\,\text{mm}}{149.44\,\text{s}} = 0.70\,mm/s$
- mean velocity of the *Paramecium* of Exp.3 —> $\frac{240.86\,\text{mm}}{265.54\,\text{s}} = 0.91\,mm/s$

Actually, the value found for the mean velocity is not completely at odds with the findings reported in the literature [2], where it is reported that a *Paramecium* can cover a distance equal to four times its body length in one second (approximately 0.6 mm/s).

The *Paramecium* in Exp.3 has a higher velocity compared to the others. This, once again, remarks our decision to follow only one *Paramecium* at a time, as there could be different phenotypes with varying characteristics.

To ensure that we are maintaining the *Paramecium* at the center of the ROI, we can monitor the centroid position frame by frame.

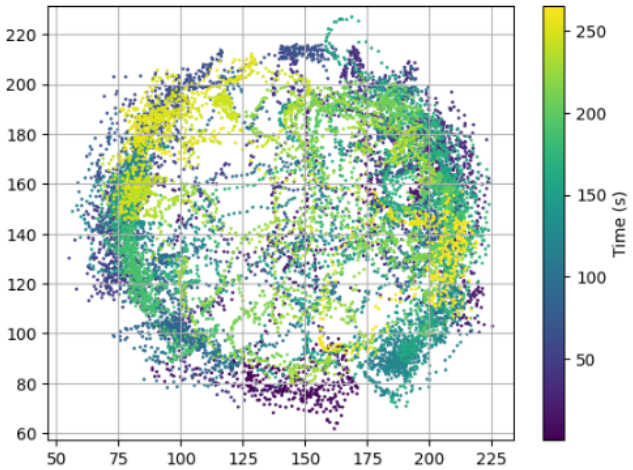


Fig. 4.3 cx and cy of Exp.3 (px position)

From figure 4.3, it's evident that factors such as the velocity of the *Paramecium*, mechanical constraints of the stage system, and time limitations of the program contribute to instances where the *Paramecium* isn't consistently positioned at the center of the ROI. However, the primary objective remains to track its movement without losing it.

4.2 Mean square displacement

We can measure the deviation of the position of a *Paramecium* with respect to a reference position over time. The mean square displacement is the most common measure of the spatial extent of random motion. We measured it for Exp.3, which represents the Paramecium's motion over the longest time.

We divided the path of our *Paramecium* in different ways, according to values given by a vector of different time steps. We chose these values randomly in a more or less linear manner.

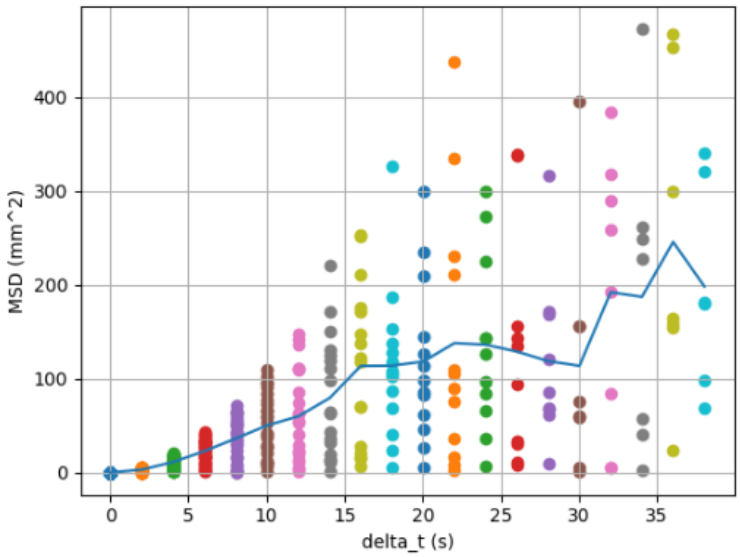


Fig. 4.4 Mean square displacement

For example, for Δt equal to 5 s, we selected the x and y positions of our *Paramecium* at intervals of 5 s and discarded all other values.

Then, we calculated the squared distance traveled from one point to the next and finally averaged these values. We repeated this procedure for different values of Δt (Figure 4.4).

We can approximately observe:

1. a ballistic regime at small times. The mean square displacement exhibits a parabolic behavior.
2. a diffusive regime at larger times. The mean square displacement increases linearly with time [10].

4.3 Avoiding reactions

Another feature that we can delve into is the one linked to the avoiding reactions. Avoiding reaction (AR) occurs when the *Paramecium* is stimulated by various means (mechanical, chemical, optical, or thermal). It often swims backward, then turns, and swims forward again in a new direction.

We wrote a code that classifies the movement as 'Run' whenever the angle formed by the old direction and the new one is less than 40°, and as 'AR' otherwise (we analyzed only Exp.3 here as well).

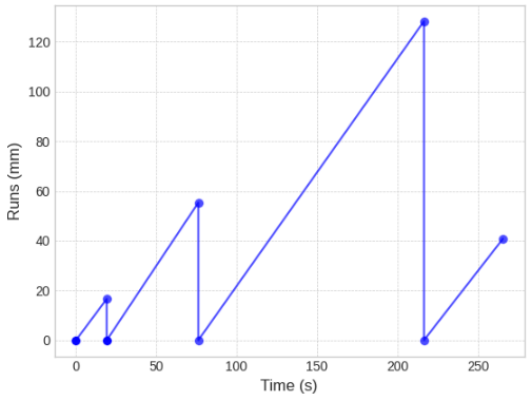


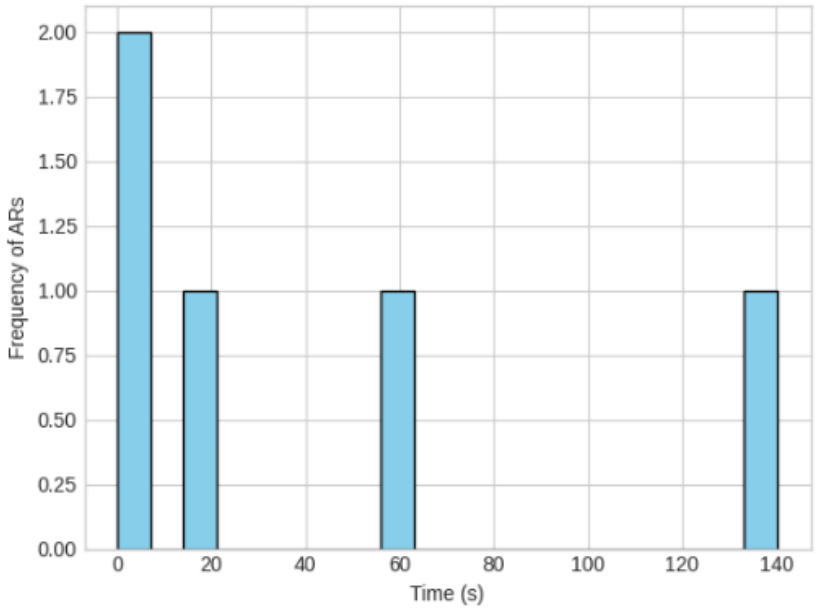
Fig. 4.5 Runs and ARs (40° Threshold)

From Figure 4.5, we can see that there are some short runs along with a very long one, which lasts approximately 140 s and covers about 130 mm.

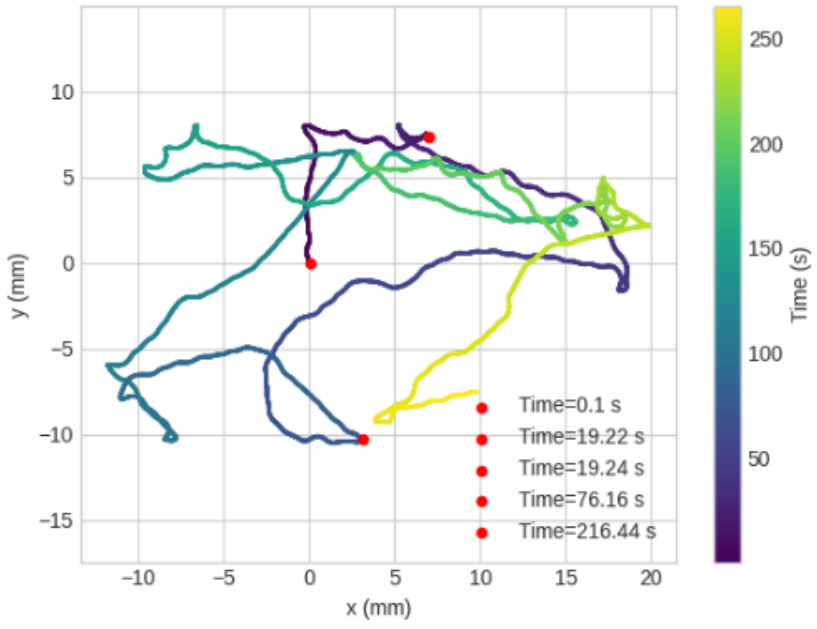
Then we differentiated between the angles formed by the old direction and the new one to see if there were any changes in the total number of avoiding reactions. We discovered that, for this specific trajectory, when the angles are less than the chosen threshold of 40°, the number of avoiding reactions grows significantly. However, if we increase the threshold, the number of avoiding reactions remains more or less the same (unless the angles are too large, causing the trajectory to be considered a single run).

This suggests that the 40-degree angle is a suitable threshold for differentiating and identifying when an avoiding reaction occurs.

We remark that this is a rough analysis. We identified an AR without considering the backward movement and the velocity value. Indeed, compared to Escoubet's work, our AR frequency is much lower (we obtained 0.02 avoiding reactions per second, instead of about 0.16, Figure 4.6, [2]).



(a)



(b)

Fig. 4.6 Avoiding reactions. (a) Avoiding reactions frequency (b) Shows where the avoiding reactions occur.

Chapter 5

Conclusions and Next Goals

There are still some problems with the trajectory plots, and the analysis that has just been completed only regards the behavior of *Paramecia* in a free environment for a maximum duration of 4 minutes. We want to track them for longer times and in an environment with obstacles to investigate potential memory effects in their movement.

5.1 Real position of the *Paramecium*

The trajectories plotted in Section 4.1 are actually not really accurate. It is not clear what the x and y axes represent. These trajectories are obtained using a 'cumsum' function and represent the relative movement of the *Paramecia*. To have faithful trajectories, it is required to know the real position of the *Paramecia* on the translation stage and on the glass slide at all times. We are working on this, but it turns out to be a very technical problem, mainly concerning timing issues, and it is taking more time than expected.

We have modified the program by removing the NI DAQ counter and generating a TTL output to trigger the camera directly from the Marzhauser Tango controller.

Using a 'readpos_ontrigger.m' function, we can read the position of the *Paramecium* on the stage and then move it with a 'go' command (provided by Marzhauser translational stage) to bring the *Paramecium* to the center of the ROI. Unlike the 'mor' command, the 'go' command does not complete its movement before accepting a new instruction; it takes the new instruction immediately.

This new version takes between 20 and 40 ms, significantly slowing down the program's speed.

5.2 From Matlab to Python

Testing all parts of the program, we realized that there are some limitations arising from the Matlab language and the serial communication protocol used to sending/receiving commands to/from the stage controller. The 'threading structure' in Python might help and speed up the program. Using threads allows a program to perform multiple operations concurrently within the same process space.

Another reason to switch to Python could be to leverage its AI capabilities. Indeed, for most AI applications, Python is generally considered easier and more versatile due to its wide range of libraries, supportive community, and simple syntax. For example, YOLO (You Only Look Once) is a well-known object detection algorithm used for object detection in images and videos. It is renowned for its speed and accuracy, as it performs object prediction and localization within a single convolutional neural network. In practice, YOLO can detect objects of various classes within an image, returning the coordinates of bounding boxes that surround the detected objects along with their class labels and detection probabilities [11]. Already tested in our group, YOLO could be a perfect tool for detecting *Paramecia* and distinguishing them from obstacles and borders.

5.3 General objectives

To conclude, we can enhance the analysis results through three different approaches:

1. Adding a third dimension, allowing the *Paramecium* to move also along the z direction and thus exhibit its natural motion. Detection along this direction is made possible by using a piezoelectric cantilever, taking advantage of the piezoelectric effect.
2. Extending our analysis to other paramecium species, for example, *Pawn* (genetically modified *Paramecia* that exhibit almost no ARs along their path), or testing the movement of *Paramecia* in different media (studying the effect of the environment on foraging) and under different experimental conditions (varying temperature, varying ion concentration, adding obstacles, ...).
3. Improving the analysis of the results by averaging multiple trajectories: more precise values of mean velocity, mean square displacement, and avoiding reactions frequency.

5.4 Conclusion

In summary, we have further developed an operative tracking system that successfully tracks *Paramecia* in natural conditions at high speeds (typically up to ~ 1 mm/s). However, achieving accurate measurements of their real positions is crucial and must be implemented to reconstruct correct trajectories. Despite these acknowledged limitations, due to the short duration of the internship, we were unable to solve this issue. By addressing these technical challenges, exploring the capabilities of Python, and considering additional dimensions and species, we should improve our understanding of *Paramecium* behavior in complex environments and better investigate potential memory effects.

Appendix A

Protocols

A.1 Protocol for preparation of medium for *Klebsiella pneumoniae*

LB BROTH

Use the LB broth from Sigma-Aldrich (L3022) and and follow the instructions provided. Briefly:

1. Dissolve 20 g of LB broth powder in 1 L of distilled;
2. Autoclave for 15 min at 121 °C;
3. Let cool.

AMPICILLIN

Solubility Ampicillin is soluble in water (50 mg/mL).

Storage/Stability The stability of ampicillin solutions depends on temperature and pH. Ampicillin solutions should not be autoclaved. Stock solutions should be sterilized by filtration. Ampicillin solutions can be added to agars or culture media that have been autoclaved and cooled to 45–50 °C. Culture plates with ampicillin can be stored at 2–8 °C for up to two weeks. Stock solutions may be stored at 2–8 °C for up to 3 weeks. For long-term storage (4–6 months), stock solutions should be stored at -20 °C. At 37 °C in culture, ampicillin is stable for up to 3 days.

pH-dependent behavior Ampicillin in solution is not very stable at pH 7. The optimal pH of the stock solution should be less than 7. Additionally, the buffer identity affects solution stability. For example,

Tris is deleterious to ampicillin at pH 7, but not at pH 5. Conversely, citrate is suitable at pH 7, but not at pH 5. Acetate buffer seems optimal at pH 6.

Preparation of a 5 mL stock solution of ampicillin at 50 mg/mL:

1. For each mL, use 50 mg of ampicillin powder;
2. Fill a 15 mL Falcon tube with 5 mL of distilled water;
3. Mix the 250 mg of ampicillin powder with the water until completely dissolved;
4. Filter-sterilize it using a 0.22 μm pore size syringe filter.
5. Aliquot the solution into Eppendorf tubes;
6. Store the aliquots in the freezer at -20 °C.

PETRI DISHES WITH LB AGAR

To prepare 4 Petri dishes with 2% (w/v) agar in LB broth, using a total volume of 100 mL:

1. Pour 100 mL of LB broth into a glass bottle;
2. Weigh 2 g of agar powder and add it to the bottle. Mix well until the agar is homogeneously or nearly dissolved;
3. Microwave the mixture for at least 3 min.
4. Allow the mixture to cool to between 45 and 50 °C. Once it reaches this temperature range, add 100 L of ampicillin (1/1000 dilution, total volume 100 mL of LB broth);
5. Pour the solution into 4 Petri dishes, with approximately 25 mL per dish.

GROWTH CONDITIONS

The *K. pneumoniae* bacteria can be cultivated in either a ventilated incubator set at 37 °C or a smaller incubator used for *Paramecia* set at 27 °C, as both conditions are suitable. However, caution is necessary when using the ventilated incubator because its powerful fan may cause Petri dishes to desiccate. To prevent this, place the Petri dishes inside a cardboard box and add a water-soaked handkerchief.

A.2 Protocol for *Paramecia* medium

For 1 L of solution WGP (concentrated version):

- **900 mL of milliQ water**.
- **50 mL of buffer solution 20X.** To prepare 1 L of such 20X buffer solution, we take 1 L of milliQ water, then we dissolve into it the following chemicals:

1. 15 g of Trizma (Tris) base;
2. 4 g of NaH2PO4 x H2O;
3. 15 g of Na2HPO4 x 2 H2O.

- **50 mL of 20X WGP infusion.** For preparing 1 L of WGP 20X infusion, we follow these steps:

1. Add 100 g of wheat grass powder to 800 mL of milliQ water in a large flask or a beaker;
2. Stir continuously with a magnetic stirrer and heat with the heater to bring it to a boil. Boil for 15 minutes;
3. Once cooked, allow it to cool slightly before filtering it using 500 mL disposable filtration units. Attach the filter to a pump to draw the infusion through, removing any remaining aggregates that could form a thick layer. After filtering, the volume is typically reduced to significantly less than 800 mL. The filtered infusion appears quite dark but not turbid;
4. Adjust the volume to 1 L and store it in the refrigerator at +4 °C. If deposits form at the bottom of the bottle, refilter the entire solution using disposable Nalgene filters.

The pH is adjusted to 7.0.

The solution is then autoclaved at 121 °C for 20 min.

IMPORTANT NOTE ABOUT THE USE OF THE WGP

To make the WGP medium usable with *Paramecia*, *Klebsiella pneumoniae* must be injected a day before and allowed to grow. These bacteria not only serve as food for the*Paramecia*, but they also make the WGP usable by digesting chemical components that are toxic to the *Paramecia*. In summary, without the bacteria, the WGP would not be suitable for cultivating *Paramecia*. The bacteria are essential for preparing a usable WGP (WGP is bad for *Paramecia*; WGPb is good for *Paramecia*).

PROTOSLO COMPOSITION

Chemical Name	CAS #	%
Water	7732-18-5	98.25
Hydroxyethyl Cellulose	9004-62-0	1.7
Methylparaben	99-76-3	0.05

Appendix B

Callback functions

```
thresh = 125;%32767;
imgb = imcomplement(img)>thresh;
s = regionprops(imgb,'Centroid','Area');
centroids = cat(1, s.Centroid);
areas = cat(1,s.Area);
ikeep = find(areas > 150); %para thresh
cx = centroids(ikeep,1);
cy = centroids(ikeep,2);

d2center = ((cx-width_x/2).^2+(cy-width_y/2).^2).^0.5;
[~,idx] = sort(d2center);
cx = cx(idx(1));
cy = cy(idx(1));
dx = (cx - width_x/2)*pix;
dy = (cy - width_y/2)*pix;

translate(tango,-dx/10,-dy/10);

cx_array(k) = cx;
cy_array(k) = cy;
dx_array(k) = dx/10;
dy_array(k) = dy/10;

fTIF.WriteIMG(img);
```

Fig. B.1 A part of the code of the callback function from the preliminary version of the automatic tracking program, addressing the centering problem

```
thresh = 125;%32767;
imgb = imcomplement(img)>thresh;
s = regionprops(imgb,'Centroid','Area');
if ~isempty(s)
    centroids = cat(1, s.Centroid);
    areas = cat(1,s.Area);
    ikeep = find(areas > 150); %para thresh
    if numel(ikeep) ~= 0
        cx = centroids(ikeep,1);
        cy = centroids(ikeep,2);
        if e == 1
            disp('looking for the paramecium...')
            d2center = ((cx-width_x/2).^2+(cy-width_y/2).^2).^0.5;
            [~,idx] = sort(d2center);
            cx = cx(idx(1));
            cy = cy(idx(1));
            if cx > (width_x-width_x1)/2 && cx < (width_x+width_x1)/2 && cy > (width_y-width_y1)/2 && cy < (width_y+width_y1)/2
                disp('here we are...')
                e = 0;
                dx = (cx - width_x/2)*pix;
                dy = (cy - width_y/2)*pix;
                translate(tango,-dx/10,-dy/10);
                cx_array(k) = cx;
                cy_array(k) = cy;
                dx_array(k) = dx/10;
                dy_array(k) = dy/10;
                k = k + 1;
            end
        else
            if numel(cx) == 1
                cx = cx(1);
                cy = cy(1);
                dx = (cx - width_x/2)*pix;
                dy = (cy - width_y/2)*pix;
            else
                diffx = abs(cx - cx_array(k-1));
                [~, idxx] = sort(diffx);
                top_2_indices = idxx(1:2);
                diffy1 = abs(cy(top_2_indices(1)) - cy_array(k-1));
                diffy2 = abs(cy(top_2_indices(2)) - cy_array(k-1));
                if diffy1 + diffx(top_2_indices(1)) < diffy2 + diffx(top_2_indices(2))
                    cx = cx(top_2_indices(1));
                    cy = cy(top_2_indices(1));
                    dx = (cx - width_x/2)*pix;
                    dy = (cy - width_y/2)*pix;
                else
                    cx = cx(top_2_indices(2));
                    cy = cy(top_2_indices(2));
                    dx = (cx - width_x/2)*pix;
                    dy = (cy - width_y/2)*pix;
                end
            end
            translate(tango,-dx/10,-dy/10);
            cx_array(k) = cx;
            cy_array(k) = cy;
            dx_array(k) = dx/10;
            dy_array(k) = dy/10;
            k = k + 1;
        end
    end
end
fTIF.WriteIMG(img);
```

Fig. B.2 A part of the code of the callback function from the updated version of the automatic tracking program, addressing the centering problem

References

[1] Ralph Wichterman. *The Biology of Paramecium*. Springer US, Boston, MA, 1986.

[2] Nicolas Escoubet. *Comportement en milieu encombré de la paramécie, un micronageur mécanosensible*. PhD thesis, 2023. Thèse de doctorat dirigée par Prevost, AlexisBrette, Romain et Pontani, Léa-Laetitia Physique Sorbonne université 2023.

[3] C Kung and Y Saimi. The Physiological Basis of Taxes in Paramecium. *Annual Review of Physiology*, 44(1):519–534, October 1982.

[4] Amandine Hamel, Cathy Fisch, Laurent Combettes, Pascale Dupuis-Williams, and Charles N. Baroud. Transitions between three swimming gaits in Paramecium escape. *Proceedings of the National Academy of Sciences*, 108(18):7290–7295, May 2011.

[5] Itsuki Kunita, Tatsuya Yamaguchi, Atsushi Tero, Masakazu Akiyama, Shigeru Kuroda, and Toshiyuki Nakagaki. A ciliate memorizes the geometry of a swimming arena. *Journal of the Royal Society Interface*, 13:20160155, 2016.

[6] Sindy K.Y. Tang and Wallace F. Marshall. Cell learning. *Current Biology*, 28:R1171–R1189, October 2018.

[7] T. Darnige, N. Figueroa-Morales, P. Bohec, A. Lindner, and E. Clément. Lagrangian 3D tracking of fluorescent microscopic objects

in motion. *Review of Scientific Instruments*, 88(5):055106, May 2017.

[8] Romain Brette. Integrative Neuroscience of *Paramecium* , a "Swimming Neuron". *eneuro*, 8(3):ENEURO.0018–21.2021, May 2021.

[9] Eric Lauga, Willow R. DiLuzio, George M. Whitesides, and Howard A. Stone. Swimming in Circles: Motion of Bacteria near Solid Boundaries. *Biophysical Journal*, 90(2):400–412, January 2006.

[10] Edward A Codling, Michael J Plank, and Simon Benhamou. Random walk models in biology. *Journal of The Royal Society Interface*, 5(25):813–834, August 2008.

[11] Peiyuan Jiang, Daji Ergu, Fangyao Liu, Ying Cai, and Bo Ma. A review of yolo algorithm developments. *Procedia Computer Science*, 199:1066–1073, 2022. The 8th International Conference on Information Technology and Quantitative Management (ITQM 2020 2021): Developing Global Digital Economy after COVID-19.