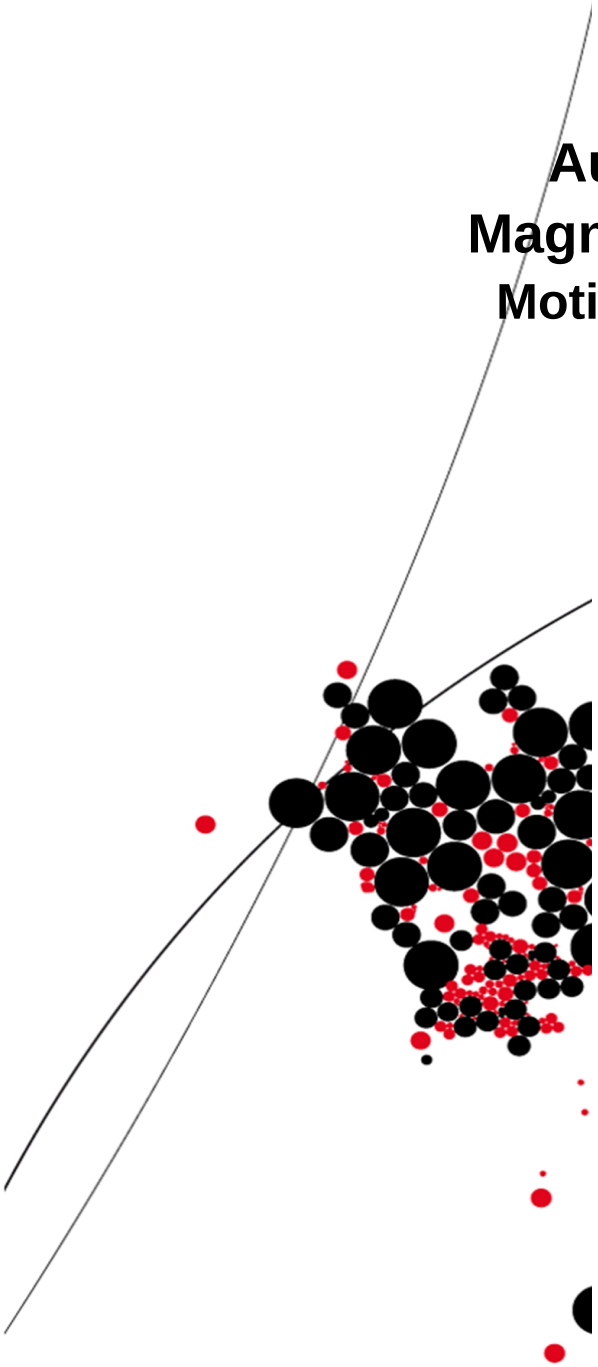
# UNIVERSITY OF TWENTE.

**Surgical Robotics Lab.**
**Faculty of Engineering Technology.**
**BioMechanical Engineering.**

# Autonomous Manipulation of MagnetoSperm Robots: Closed-loop Motion Control and Characterization Inside Fluidic Channels



**Mohamed Ehab Aly Hafez**
**Bachelor Thesis**
**August 2017**

**Supervisors:**
Prof. Dr. Sarthak Misra.
Dr. Ir. Stefano Scheggi
Dr.Islam Khalil

Surgical Robotics Lab Group
Faculty of Engineering Technology.
University of Twente.
P.O. Box 217
7500 AE Enschede
The Netherlands

## Declaration of Authorship

I, Mohamed Ehab Aly Hafez, declare that this thesis titled, 'Autonomous Manipulation of MagnetoSperms Robots: Closed-loop Motion Control and characterization Inside Fluidic Channels' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at the University of Twente.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Mohamed Ehab Aly Hafez

Date: 31/8/2017

"One, remember to look up at the stars and not down at your feet. Two, never give up work. Work gives you meaning and purpose and life is empty without it. Three, if you are lucky enough to find love, remember it is there and don't throw it away."

Stephen Hawking

Because of that We, Engineers have to work hard every day since our responsibility is to invest our minds, effort and time to do more for those who really need our help. You, scientists and researchers, are changing the world and you have a great impact on People's life so please keep working on developing lives. Do not give up until you cure a disease or save people's life. The world and people are waiting for you.

# ACKNOWLEDGMENTS

I would like to dedicate this work to my beloved mother, Amira Mohamed, my father, Ehab Hafez and my sister, Nada Ehab. Without your love and support, I could not have made it until the end and thank you for giving me the chance to live this great research experience abroad in the University of Twente. It is with immense gratitude that I acknowledge the support and help of my supervisor Dr. Stefano Scheggi who believed in me and was continuously supporting me in every problem I faced during this research with a very good friend and a nice person. Very special thanks to Prof. Sarthak Misra and Dr. Islam Khalil for their eagerness and great assistance; their comments were of critical importance and their help through every step was a great support and motivation for me. Finally, I owe you all everything.

To the memory of my grandfathers, Aly Hafez, Mohamed Assal and my grandmothers, Samia El Domiaty, Faiza El Kholi. Also, to the memory of everyone that died because of a disease that we are trying every day to innovate, create and apply better solutions for it in order to decrease its effects, possibilities of infection and increasing number of survival patients in this world.

TABLE OF CONTENTS

Page

LIST OF FIGURES . . . . . viii

LIST OF TABLES . . . . . x

SYMBOLS . . . . . xi

ABSTRACT . . . . . xii

1 Introduction . . . . . 1

1.1 Motivation . . . . . 1

1.2 Objective . . . . . 3

1.3 Thesis structure . . . . . 5

2 MobiMag setup and fluidic channel . . . . . 6

2.1 MobiMag setup . . . . . 6

2.1.1 Hardware and the electromagnetic coils . . . . . 6

2.1.2 Controllers and software techniques . . . . . 7

2.2 Fluidic open channels . . . . . 9

2.2.1 Different widths channel . . . . . 9

2.2.2 Bifurcations channel . . . . . 11

3 Modeling and control . . . . . 13

3.1 Development and modeling of the MagnetoSperm . . . . . 13

3.1.1 Fabrication of the MagnetoSperm . . . . . 13

3.1.2 Modeling and characterization of the MagnetoSperm . . . . 14

3.2 Motion control design . . . . . 17

3.2.1 Tracking system . . . . . 17

3.2.2 MagnetoSperm motion control strategy and technique . . . . 23

3.2.3 MagnetoSperm velocity estimation . . . . . 30

4 Experimental procedure . . . . . 35

Page

4.1 Motion control of MagnetoSperm . . . . . 35
4.1.1 Point-to-point manipulation . . . . . 35
4.2 Characterization of MagnetoSperm inside fluidic channels . . . . . 36
4.2.1 Effect of channel width on the MagnetoSperm dynamics . . 36
4.2.2 Effect of different precession angles and frequency response on the MagnetoSperm . . . . . 36
4.2.3 Experiments inside the Aorta channel . . . . . 37
5 Experimental results . . . . . 38
5.1 Characterization inside open fluidic channels . . . . . 38
5.1.1 Influence of the channel wall on the flagellar propulsion of the MagnetoSperm . . . . . 38
5.1.2 Oscillation frequency response and precession angles effect . 40
5.1.3 Experiments inside the Aorta channel . . . . . 44
6 Conclusion and future recommendations . . . . . 46
7 Appendix . . . . . 49
7.1 MagnetoSperm controller $C++$ source code . . . . . 49
7.2 MagnetoSperm controller $C++$ header code . . . . . 69
7.3 The tracking $C++$ code for closed trajectories and predefined path 74
LIST OF REFERENCES . . . . . 78

# LIST OF FIGURES


| Figure | | Page |
|---|---|---|
| 1.1 | Plaque inside heart arteries. | 3 |
| 1.2 | Artery inflammation. | 4 |
| 1.3 | Decrease of artery width. | 4 |
| 2.1 | MobiMag setup with its four electromagnetic coils. | 6 |
| 2.2 | MobiMag setup controllers and electronics. | 8 |
| 2.3 | MobiMag electrical enclosure containing: (*i*) Elmo drivers; (*ii*) Arduino Uno; and (*iii*) Power supply. | 9 |
| 2.4 | Fluidic channel with its 4 widths. | 10 |
| 2.5 | Side and top view MagnetoSperm swimming in the different channel widths. | 10 |
| 2.6 | Aorta channel. | 11 |
| 2.7 | Aorta inside human body. | 12 |
| 2.8 | Fabricated Aorta channel. | 12 |
| 3.1 | MagnetoSperm under microscope. | 15 |
| 3.2 | Numerical differentiation velocity without FIR. | 19 |
| 3.3 | Numerical differentiation velocity without FIR. | 19 |
| 3.4 | FIR effect on MagnetoSperm experiments. | 20 |
| 3.5 | FIR effect on Microparticles experiments. | 21 |
| 3.6 | Some of the closed trajectory paths used in experiments. | 22 |
| 3.7 | Testing a square trajectory with Microparticle. | 22 |
| 3.8 | Testing a triangle trajectory with Microparticle. | 23 |
| 3.9 | Magnetic field magnitude ratio for the 4 MobiMag coils. | 25 |
| 3.10 | Oscillation angles of MagnetoSperm swimming straight towards Coil 2. | 28 |
| 3.11 | Comparison between numerical differentiation, iterative learning observer and Zero-phase filter for some of our experimental data. | 33 |

Figure Page

3.12 Comparison between iterative learning observer and Zero-phase filter for position data of cosine wave and velocity of -sine wave. . . . . 34

5.1 Average velocity of the Microparticle in three different channel widths. 38

5.2 Average velocity of the MagnetoSperm in four different channel widths. 39

5.3 Average velocity of MagnetoSperm at each frequency along all the precession angles in channel width 1 mm. . . . . 41

5.4 Average velocity of MagnetoSperm at each precession angle along all the frequencies in channel width 1 mm. . . . . 41

5.5 Average velocity of MagnetoSperm at each frequency along all the precession angles in channel width 0.75 mm. . . . . 43

5.6 Average velocity of MagnetoSperm at each precession angle along all the frequencies in channel width 0.75 mm. . . . . 43

5.7 Controlling MicroParticle inside the Aorta channel using predefined path. . . . . 45

# LIST OF TABLES


| Table | | Page |
|---|---|---|
| 2.1 | Electromagnetic coils specifications. | 7 |
| 2.2 | Arduino Uno specifications. | 8 |
| 2.3 | Open fluidic channels specifications. | 10 |
| 3.1 | PID controller gains effect. | 26 |
| 3.2 | Ziegler-Nichols method. | 27 |
| 5.1 | Average velocity of MagnetoSperm with respect to precession angles and frequencies. | 42 |
| 5.2 | Average velocity of MagnetoSperm with respect to precession angles and frequenciesin channel width 0.75 mm. | 44 |

# SYMBOLS

| | |
|---|---|
| $P$ | Position of the MagnetoSperm |
| $\dot{P}$ | Velocity of the MagnetoSperm |
| $\eta$ | Dynamic viscosity of water |
| $F()$ | Magnetic force |
| $F_d$ | Drag force |
| $T()$ | Magnetic torque |
| $B()$ | Induced magnetic field |
| $H$ | Magnetic field strength |
| $N_t$ | Number of coil turns |
| $L$ | Length of the coil |
| $I$ | Current passing through coils |
| $m$ | Magnetic dipole moment of the MagnetoSperm |
| $\dot{\theta}$ | Angular velocity of the MagnetoSperm |
| $\omega_b$ | Boundary frequency |
| $h[z]$ | Transfer function of the running average filter |
| $Ts$ | Sampling time |
| $\hat{x}, \hat{y}$ | Estimated value of MagnetoSperm x and y position |
| $v_x, v_y$ | MagnetoSperm velocity in x and y direction |
| $x_{ref}, y_{ref}$ | the setpoint x and y position |
| $x_d, y_d$ | Distance in x and y direction between the setpoint and the agent position |
| $R$ | Radius of the circle that its center point is the MagnetoSperm position and the setpoint is on that circle |
| $p_x, p_y$ | MagnetoSperm position with respect to the x and y axis |

# ABSTRACT

In this thesis, we study the motion control of MagnetoSperm inside fluidic channels. This research is performed in order to evaluate the possibility to use our MagnetoSperm in robotic surgeries where the high accuracy of control is desired to reach the unreachable parts of the human body. Using wireless controlled translational forces, torques and feature tracking software, a MagnetoSperm with an average head diameter of 25 $\mu$m and the total length of 300 $\mu$m, is able to perform point-to-point positioning. Moreover, we study how the velocity of the agent is affected by: ($i$) Channel width; ($ii$) frequency response; and ($iii$) precession angle. Then, we perform experiments on the MagnetoSperm in a biological-inspired channel to simulate a more complicated environment. We develop a closed-loop control algorithm which applies oscillations on the tail of the MagnetoSperm by alternating magnetic fields. Although MagnetoSperm can move in various fluids, it is preferable to have a medium which is viscous enough to allow the tail to oscillate in a proper way. In this regard, we investigate the motion of such agents in water and glycerin mixture.

# 1. INTRODUCTION

## 1.1 Motivation

This year the National Cancer Institute announced that in 2030 the rate of worldwide cancer cases will increase by 50% from 14 million to 21 million and the cancer deaths will increase by 60% from 8 million to 13 million with respect to 2012 deaths' numbers. In 2012, there were new 14 million cancer cases and in 2016, there were 8.8 million deaths because of cancer. My country, Egypt is one of the highest communities in the number of cancer cases among our people. Maybe every day we know that one of our relatives or our friends discover that he or she has cancer. Every day we ask each others, what we can do for these people and every time we do not have an answer for that. Till now, They beat their hard times and their chemotherapy sessions only with their hope and their belief in the God without any help from us. Without hope, we can not live a better life, and without science, we can not live. So we need both of them to continue our life in the best way.

Because of that, I started three years ago working in Medical Robotics field, having hope and believing in this science that could be one day change someone's life. Of course, I am not sure 100% that all the projects we are working on will succeed and help these people, but we have to try, develop, improve, create and innovate.

So six months ago I decided to work on this project and be a small part of this future science and hope.

In this research, we want to continue our and other scientists' hope towards full micromanipulation of MagnetoSperm agent. Then, we are able to use these microagents in micro-surgeries of the most murdering diseases. These operations are performed at micro and nano scale in order to reach the hard to access regions inside the human body. Also, performing the delicate surgical operations wirelessly and autonomously

allows to replace the need for the human intervention and thus the possibility for error. In the last years, many microagents were introduced with their different shapes, characteristics, and applications. B.J.Nelson *et al.* [11] have shown different microagents with their different dynamics and swimming ways. Moreover, they compared these agents to determine the best ones for the different medical applications.

Finally, microagents have the potential to revolutionize many aspects of medicine. These untethered, wirelessly controlled and powered agents will make existing therapeutic and diagnostic procedures less invasive. Also, this will enable new surgical procedures that were never possible before.

## 1.2 Objective

The objective of this thesis is to implement motion control algorithms for the MagnetoSperm. The experiments will include point-to-point control and closed trajectories. Then, we investigate the effect of the channel width on the MagnetoSperm speed through controlling the agent in different widths of fluidic channel. We perform these experiments and studies since we are inspired by the plaque effect inside the blood capillaries and arteries or even the arteries inflammation. This plaque or inflammation led to decrease in the diameter of the capillaries.

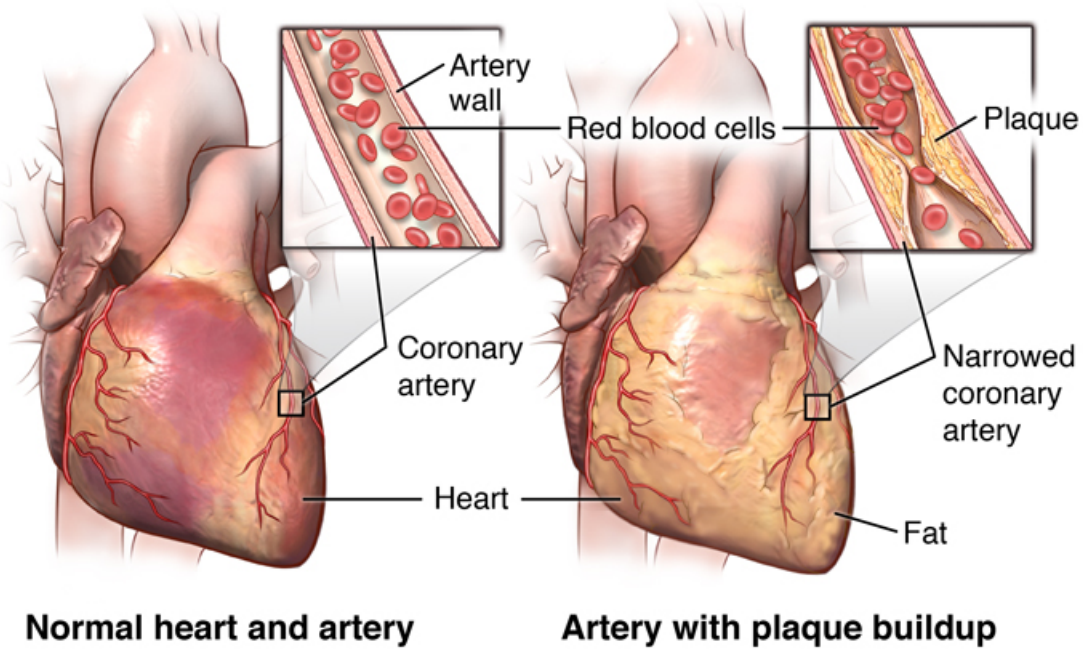



Fig. 1.1.: Plaque inside heart arteries.

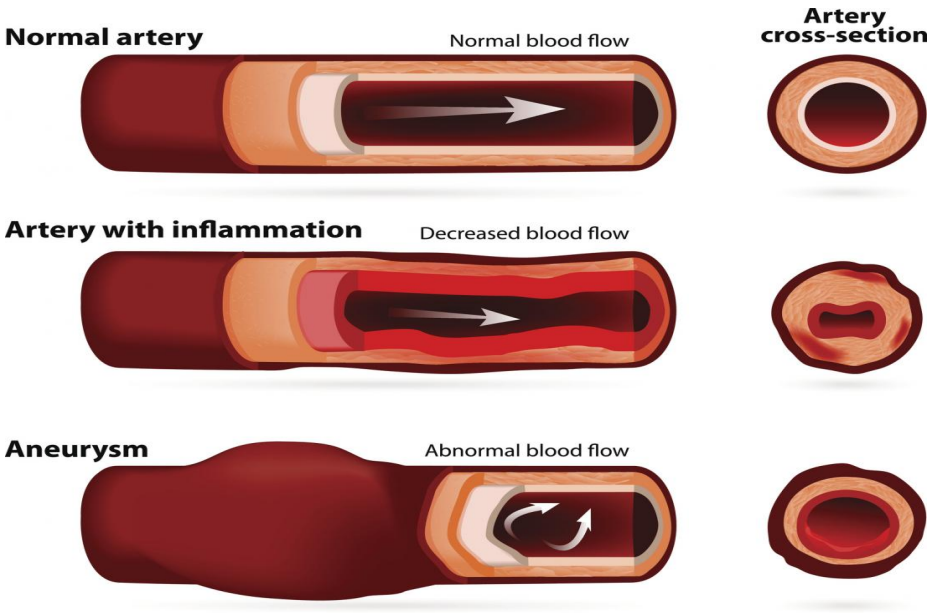



Fig. 1.2.: Artery inflammation.

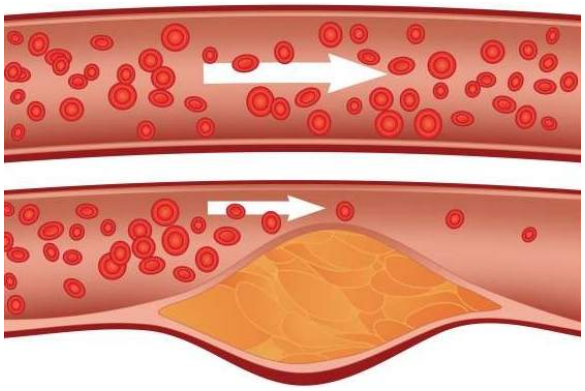


Fig. 1.3.: Decrease of artery width.

Also, we investigate the effect of different precession angles and frequency response on the agent. Finally, we evaluate the motion control algorithm in a more complicated environment in order to simulate the movement of the MagnetoSperm inside the human body.

## 1.3 Thesis structure

Through the thesis, we present the MobiMag setup and all its hardware and software features and parameters. Then, we present how we design and fabricate the open fluidic channel with different widths. We introduce the modeling and fabrication of our MagnetoSperm and the closed loop motion control design and technique features. Finally, we describe the experimental procedure of: (*i*) How we go through our experiments; (*ii*) why we managed to do the experiments in that way; and (*iii*) what is the aim of each experiment. Then, we present the experimental results, our conclusion and our future recommendations for more research improvements.

# 2. MOBIMAG SETUP AND FLUIDIC CHANNEL

## 2.1 MobiMag setup

In this section, we will show the hardware components of the MobiMag setup and also the specifications of electromagnetic coils and their magnetic field. Finally, we discuss the controllers and software techniques used to operate the setup.

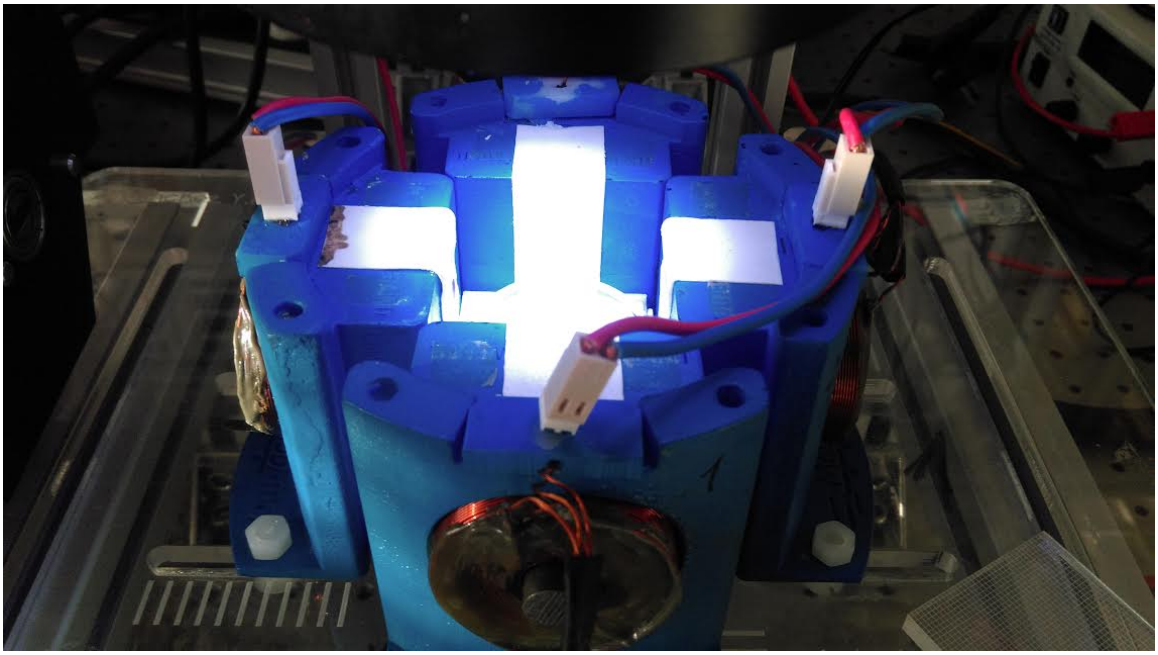


Fig. 2.1.: MobiMag setup with its four electromagnetic coils.

### 2.1.1 Hardware and the electromagnetic coils

MobiMag setup consists of 4 identical electromagnetic coils in an orthogonal array (Fig. 2.1). The distance between each two opposite coils is 4.2 cm to be able to include a petri-dish of 4 cm diameter. Also, the MobiMag setup is composed of a high accuracy microscopic camera.

The 4 identical electromagnetic coils have been designed with certain considerations to satisfy the needed magnetic field for controlling the microagents.
Each coil has the following specifications [8]:

| Parameter | Specifications |
|---|---|
| Coil Diameter | 40 cm |
| Coil material | copper |
| Core Diameter | 1 cm |
| Core Material | Iron copper alloy |
| Core Relative Magnetic permeability($\mu_r$) | 15000 |
| Number of Turns | 1700 |

Table 2.1: Electromagnetic coils specifications.

The intensity of the magnetic field is,

$$B = \mu_m H = \mu_m \frac{I \times N_t}{L}. \tag{2.1}$$

The electromagnetic coils have been designed to guarantee that a magnetic field intensity of 5 mT is generated in the middle point of the workspace, whenever the current in one of the coils is set to 1 A. Magnetic fields are measured using a calibrated three-axis Hall magnetometer (Sentron AG, Digital Teslameter 3MS1-A2D3-2-2 T, Switzerland).

### 2.1.2 Controllers and software techniques

Elmo whistle drivers are used to activate the coils. The Elmo drivers are able to provide a current supply of 1 A [8]. The drivers are controlled using the analog reference command that consists of three pins. The drivers are connected directed to the coils through the right way [8].
The command used for the drivers is received from a microcontroller arduino Uno kit. The electronics specifications are as follows [8]:

| Parameter | Specification |
|---|---|
| Microcontroller | Arduino Uno Atmega328 |
| Operating voltage | 5 V |
| DC Current per I/O pin | 40 mA |
| DC current for 3.3V pin | 50 mA |
| Clock Speed | 16 MHz |

Table 2.2: Arduino Uno specifications.

(a) Elmo driver.

(b) Arduino Uno.

Fig. 2.2.: MobiMag setup controllers and electronics.

All the electronics and controllers are put inside an electrical enclosure to avoid any hazards that can bring its functionality to danger.

Fig. 2.3.: MobiMag electrical enclosure containing: (*i*) Elmo drivers; (*ii*) Arduino Uno; and (*iii*) Power supply.

We have many options to get the best real time control and vision. We choose $C++$ to be used in the realization of this project. The OpenCv library is used for image processing. The library is an open source computer vision library founded by Intel Russia research center in Nizhny Novgorod in 1999. It provides very powerful built-in functions for image acquisition, feature tracking, thresholding, image filtering, background subtraction and color detection. The library is supported on both Windows and Linux and is compatible with a variety of language like $C$, Java, python and $C++$. As mentioned before, the main controller code is written in $C++$ where image processing is done to provide the accurate position of the microagents. The signal is sent via serial communication to the Arduino board. The baud rate of the serial communication is 115200 bit/s.

## 2.2 Fluidic open channels

### 2.2.1 Different widths channel

We fabricate an open channel in a pyramidal shape with 4 different widths. This enables us to investigate the effect of these channels' walls on the characterization of

the MagentoSperm. This channel are fabricated from acrylic material with a thickness of 3 mm and cut using very accurate laser cutter.

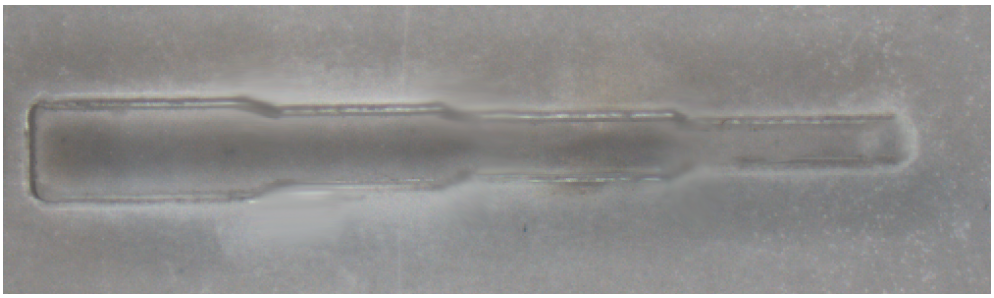


Fig. 2.4.: Fluidic channel with its 4 widths.

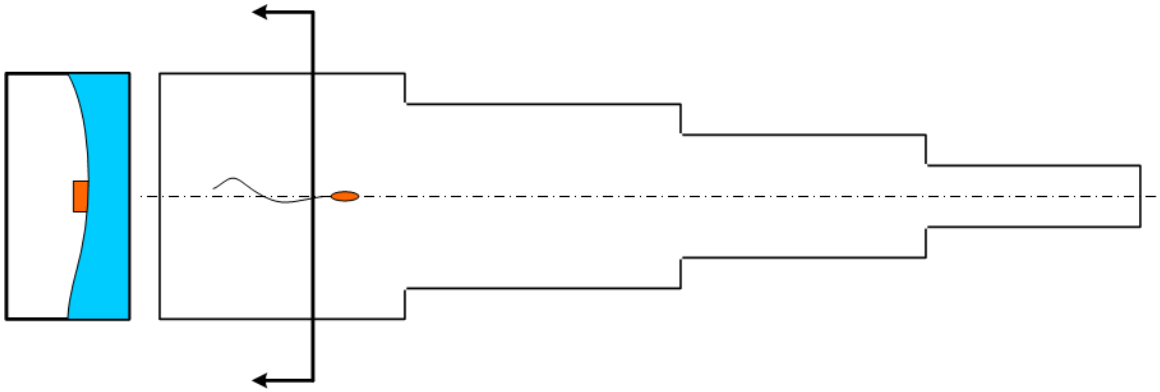


Fig. 2.5.: Side and top view MagnetoSperm swimming in the different channel widths.

| Parameter | Specifications |
|---|---|
| Total channel length | 12 mm |
| Each channel length | 3 mm |
| 1st channel width | 1 mm |
| 2nd channel width | 0.75 mm |
| 3rd channel width | 0.5 mm |
| 4th channel width | 0.25 mm |

Table 2.3: Open fluidic channels specifications.

These dimensions for the channels are calculated based on the average MagnetoSperm head width of 50 $\mu$m and the total MagnetoSperm length of 300 $\mu$m.

### 2.2.2 Bifurcations channel

We want to go further through our experiments and inspiration of seeing the MagnetoSperm swimming inside the human body. So, we manage to simulate the movement of the MagnetoSperm inside human body arteries. We got inspiration of the Aorta and we fabricate a fluidic open channel much similar to the Aorta and its scale is about 1:20 with respect to the real Aorta average dimensions and diameters. We choose the middle part of the Aorta to be able to have as many bifurcations as possible.

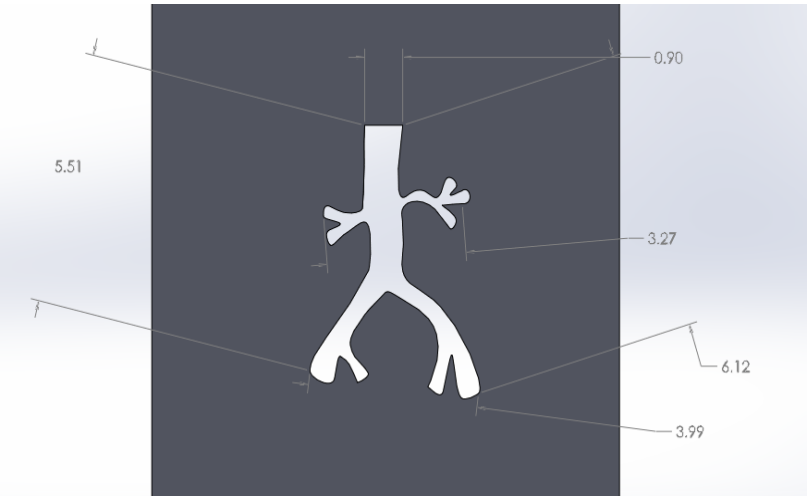



Fig. 2.6.: Aorta channel.

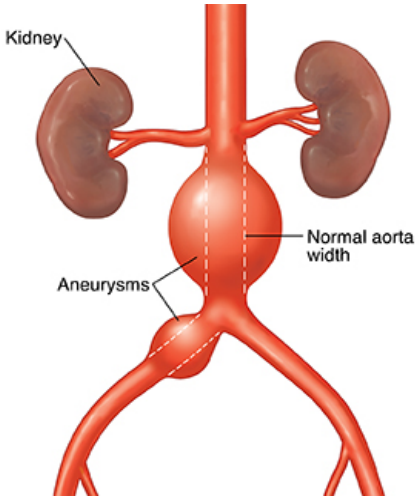



Fig. 2.7.: Aorta inside human body.

We fabricate our Aorta channel from the acrylic material with a thickness of 3 mm. The maximum channel width is about 1 mm and the minimum is 0.25 mm. The longest channel length for the longest possible MagnetoSperm path is around 6.5 mm.

Fig. 2.8.: Fabricated Aorta channel.

# 3. MODELING AND CONTROL

## 3.1 Development and modeling of the MagnetoSperm

MagnetoSperm consists of a magnetic head with a Cobalt-Nickel layer ($Co_{80}Ni_{20}$) and a flexible tail, as shown in Fig. 2. Applying oscillating magnetic fields exerts a magnetic torque on the magnetic dipole moment of the MagnetoSperm. This torque oscillates the flexible tail and results in a thrust the force. The thrust force overcomes the drag force while swimming and friction force while sliding on a surface. In this section, we describe the fabrication steps of MagnetoSperm and it's linear and rotational dynamics modeling.

### 3.1.1 Fabrication of the MagnetoSperm

Fabrication of the MagnetoSperm is done using two steps [1]. First, development of the main body (head, neck, and tail). Second, deposition of the magnetic material on the head. SU-8 is selected as the structural material for the main body due to its mechanical stability and ease of fabrication. A silicon wafer with $<100>$ crystal orientation is initially spin coated with SU-8-5. The thickness and the diameter of this wafer are 500 $\mu$m and 100 mm, respectively. Following the pre-bake of the wafer, the MagnetoSperm patterns are transferred to the SU-8 layer by ultraviolet (UV) exposure. The wafer is further baked (post-exposure bake) and the MagnetoSperm bodies are realized by developing the SU- 8 layer in RER600 (ARCH Chemicals, Basel, Switzerland). The mechanical stability is achieved by hard baking the wafer. The magnetic material on the head of the MagnetoSperm is developed through a lift-off procedure. The layout of the magnetic material is transferred onto the MagnetoSperm by spin-coating the wafer a 10 $\mu$m-thick photo-resist (AZ 9260)

and UV-exposure. After 30 seconds of oxygen plasma treatment, to enhance the adhesion of the metal layer to the SU-8, a 200 nm-thick cobalt-nickel layer is deposited by e-beam evaporation and lifted-off in acetone. Oxygen plasma treatment is then performed on the MagnetoSperm for 30 seconds to make the SU-8 hydrophilic. Finally, the MagnetoSperm is released by etching away the silicon wafer in a 5% TMAH solution at 85°C and the solution is diluted. After the development of MagnetoSperm, a magnetic system with an orthogonal array of electromagnetic coils is used to control its motion.

### 3.1.2 Modeling and characterization of the MagnetoSperm

Under the influence of a magnetic field, the magnetic force (F(P) $\in \mathbb{R}^{3\times1}$) and torque (T(P) $\in \mathbb{R}^{3\times1}$) experienced by the magnetic head of MagnetoSperm located at position (P $\in \mathbb{R}^{3\times1}$) are given by [4],[5]

$$F(P) = (m.\bigtriangledown)B(P) \tag{3.1}$$

$$T(P) = m \times B(P), \tag{3.2}$$

where m $\in \mathbb{R}^{3\times1}$ and B(P) $\in \mathbb{R}^{3\times1}$ are the magnetic dipole moment of MagnetoSperm and the induced magnetic field, respectively. A lower limit of the drag force on MagnetoSperm can be obtained by neglecting its head and assuming that its morphology is similar to a long thin needle of length $l$ and diameter $d$ [6]:

$$F_d(\dot{P}) = \eta \frac{l}{ln\frac{2l}{d} - 0.81}(\dot{P}), \tag{3.3}$$

where $F_d$ $(\dot{P}) \in \mathbb{R}^{3\times1}$ and $\dot{P} \in \mathbb{R}^{3\times1}$ are the drag force and the velocity of MagnetoSperm, respectively. Further, $\eta$ is the dynamic viscosity of 20% water and 80% Glycerol (percentage is for the mass fraction of water and Glycerol) mixture (50 mPa.s), $l$ and $d$ are the length (50 $\mu$m) and diameter (300 $\mu$m) of MagnetoSperm, respectively. Using (3.2), the linear drag force is calculated to be $3.77 \times 10^{-10}$ N at a speed of 100 $\mu$m/s (maximum average speed of MagnetoSperm at frequency of 5 Hz).

Using (3.1), the magnetic force exerted on the magnetic dipole of MagnetoSperm is given by

$$F(P) = \int_v M_s \mathrm{d}v. \bigtriangledown B(P) = m. \bigtriangledown B(P), \tag{3.4}$$

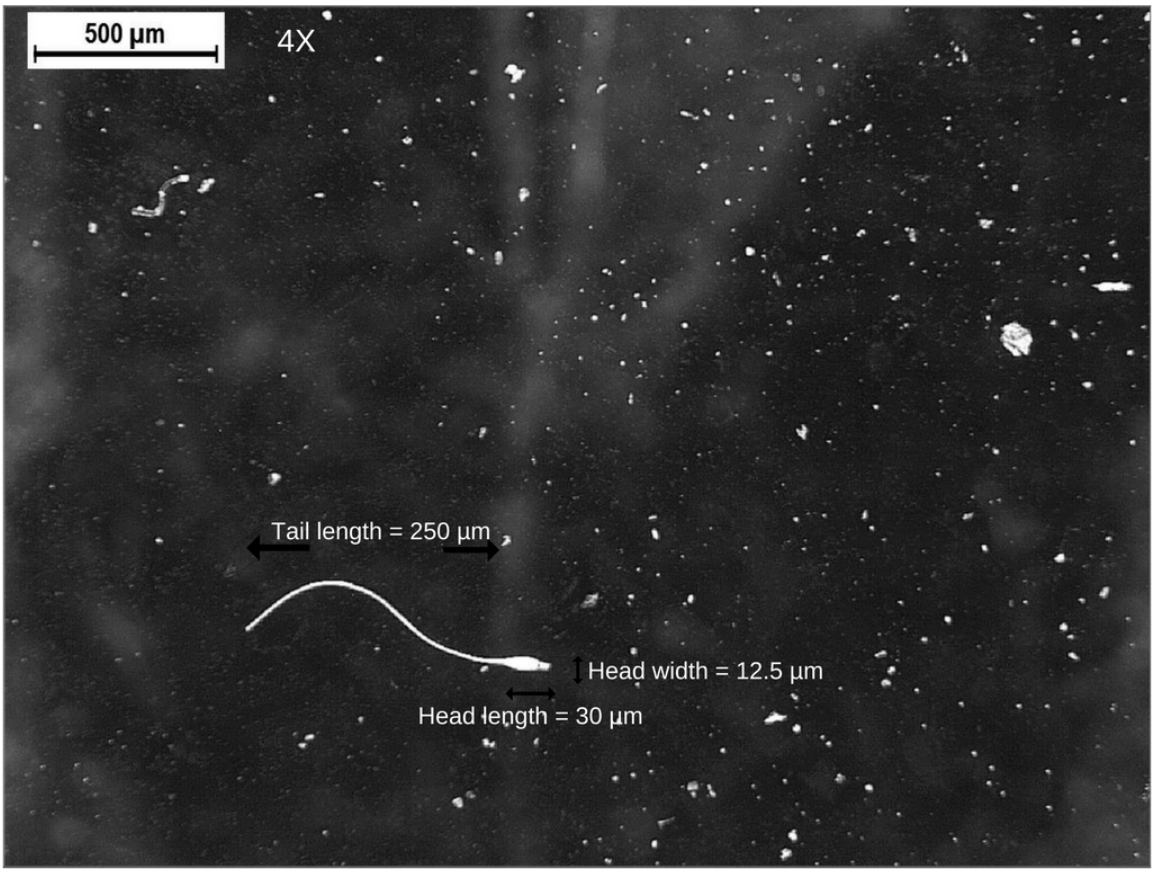



Fig. 3.1.: MagnetoSperm under microscope.

where F(P) is the magnetic force at point (P). Further, $M_s$ and $v$ are the magnetization saturation ($1.19 \times 10^6 A/m$) and volume of the $Co_{80}Ni_{20}$ that is deposited on the ellipse head of the MagnetoSperm, respectively.

The volume, $v$ of the MagnetoSperm head is measured by

$$\frac{4}{3}\pi(Length)(Width)(Width) = 1.96 \times 10^4 \mu m^3 \tag{3.5}$$

Using (3.4), the maximum magnetic force exerted on the MagnetoSperm is calculated to be $6.43 \times 10^{-14}$ N at magnetic field gradient of 2.75 mT/m. This calculation

shows that the maximum magnetic force exerted on MagnetoSperm by our magnetic system is 2 orders-of-magnitude smaller than the drag force. Therefore, the motion of MagnetoSperm is mainly due to the thrust force generated based on the oscillation of the flexible tail. This result allows us to steer and propel the MagnetoSperm using weak magnetic fields without the magnetic field gradients. In *vivo* applications, contact could occur between the MagnetoSperm and a surface (blood vessel or tissue). Therefore, equation of motion of the MagnetoSperm is given by [2]

$$F(P) + F_d(\dot{P}) + F_t + \mu_s R = 0, \tag{3.6}$$

where $F_t$ is the propulsion force generated by the flexible tail due to the oscillating magnetic fields. Further, $\mu_s$ and R are the static coefficient of friction between the MagnetoSperm and a surface, and the reaction force on the MagnetoSperm, respectively. In our experiments, the MagnetoSperm is not in contact with the surface. So this term ($\mu_s R$) in our modeling will be equal to zero. So The rotational dynamics of the MagnetoSperm is given by [2],[9]

$$|B(P)||m| \sin(\beta) + \alpha\dot{\theta} = 0, \tag{3.7}$$

where $\beta$ is the angle between the induced magnetic field and the magnetic dipole moment of the MagnetoSperm. Further, $\dot{\theta}$ and $\alpha$ are the angular velocity of the MagnetoSperm and the rotational drag coefficient, respectively.

Characterization of the magnetic dipole moment requires determination of its boundary frequency ($\omega_b$). This frequency can be determined by gradually increasing the frequency of rotating field and observing the frequency after which the agent can no longer follow the field, i.e., $\dot{\theta} = \omega_b$, when $\sin\beta = 1$. Therefore, (3.7) can be written as [9]

$$|B(P)||m| + \alpha\omega_b = 0. \tag{3.8}$$

In (3.4), the magnetic force (F(P)) does not allow for pulling the MagnetoSperm towards the reference. However, it contributes to the propulsion force generated by the

oscillating tail in overcoming the drag force (3.5), indicating that the magnetic torque must overcome the rotational drag. Therefore, the angle ($\beta$) between the magnetic fields (B(P)) and the magnetic dipole moment (m) has to be 90° to maximize the magnetic torque exerted on the magnetic dipole of the MagnetoSperm.

## 3.2 Motion control design

### 3.2.1 Tracking system

#### FIR (Running Average Filter) and detection of set-point and MagnetoSperm position

The detection of MagnetoSperm position is done using image processing tracker algorithm in $C++$ after getting the visual feedback from the camera. The user can easily set the desired setpoint before or change it during the controlling process using only the PC's mouse. Also, we can set the position of the MagnetoSperm before the controlling process and indicate which agent we want to control if there are many.

To increase the accuracy of our agent position tracking and filter any vibrations due to the agent oscillation, we use FIR (Finite Impulse Response). FIR filter is a filter whose impulse response is of finite duration, because it settles to zero in finite time and also because there is no feedback. The condition of not having feedback guarantee that the impulse response will be finite. The impulse response of an Nth-order discrete-time FIR filter lasts exactly $N + 1$ samples before it then settles to zero. For a causal discrete-time FIR each value of the output sequence is a weighted sum of the most recent input values:

$$\begin{aligned} y[n] &= b_0 x[n] + b_1 x[n-1] + .... + b_N x[n-N] \\ &= \sum_{i=0}^{N} bi.x[n-i], \end{aligned} \tag{3.9}$$

where $x[n]$ is the input signal, $y[n]$ is the output signal, $N$ is the filter order, $b_i$ is the value of the impulse response at the $i$th instant. We use a special simple type of

FIR which is moving average filter. Average running filter is a simple type of FIR, in which the $N$th prior sample is subtracted each time a new sample comes in. This filter has a finite impulse response even though it uses feedback: after $N$ samples of an impulse, the output will always be zero. The filter coefficients, $b_0$,....,$b_N$, are found via the following equation.

$$b_i = \frac{1}{N+1} \tag{3.10}$$

To provide a more specific example, we select the filter order: $N = 2$ The impulse response of the resulting filter is:

$$h[n] = \frac{1}{3}\delta[n] + \frac{1}{3}\delta[n-1] + \frac{1}{3}\delta[n-2] \tag{3.11}$$

The transfer function is:

$$h[z] = \frac{1}{3} + \frac{1}{3}z^{-1} + \frac{1}{3}z^{-2} = \frac{1}{3}\frac{z^2+z+1}{z^2} \tag{3.12}$$

So, we can conclude that the transfer function of the moving average filter at the end is:

$$h[z] = \frac{1}{n}\sum_{i=0}^{n-1} Z^{-i} \tag{3.13}$$

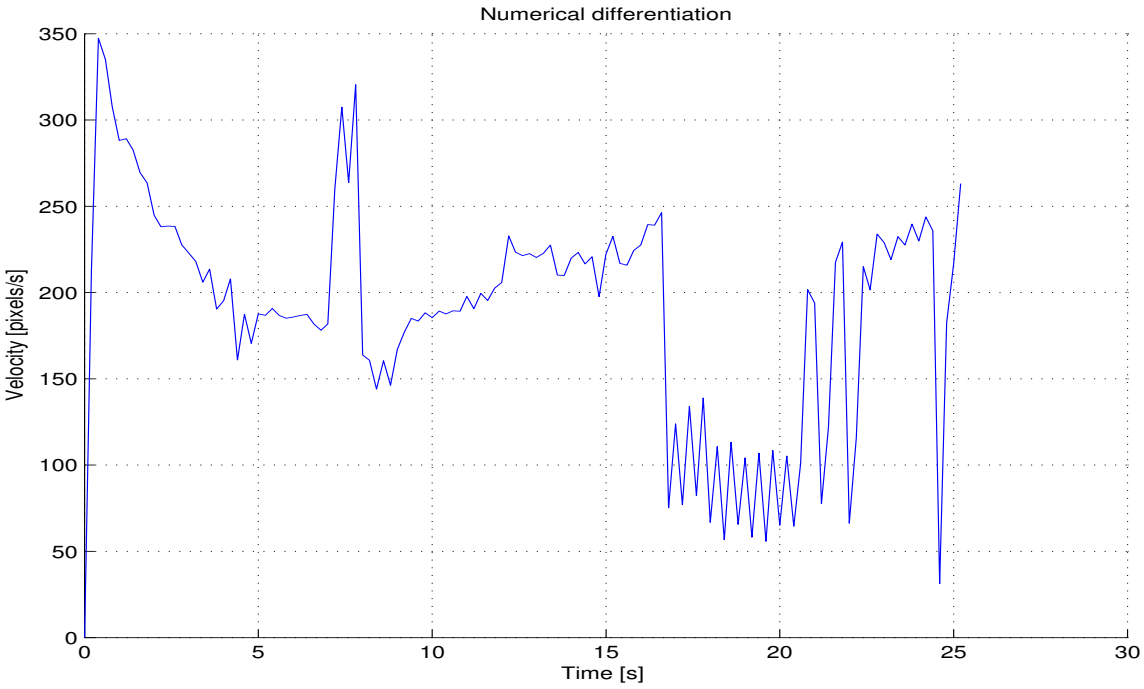



Fig. 3.2.: Numerical differentiation velocity without FIR.

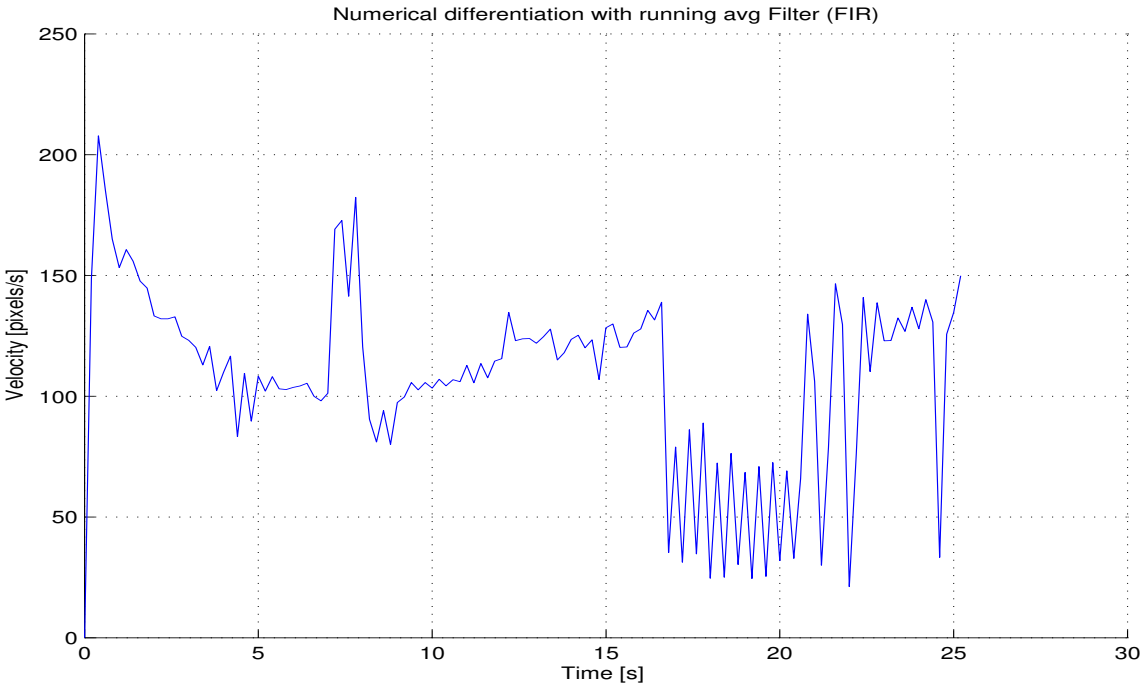
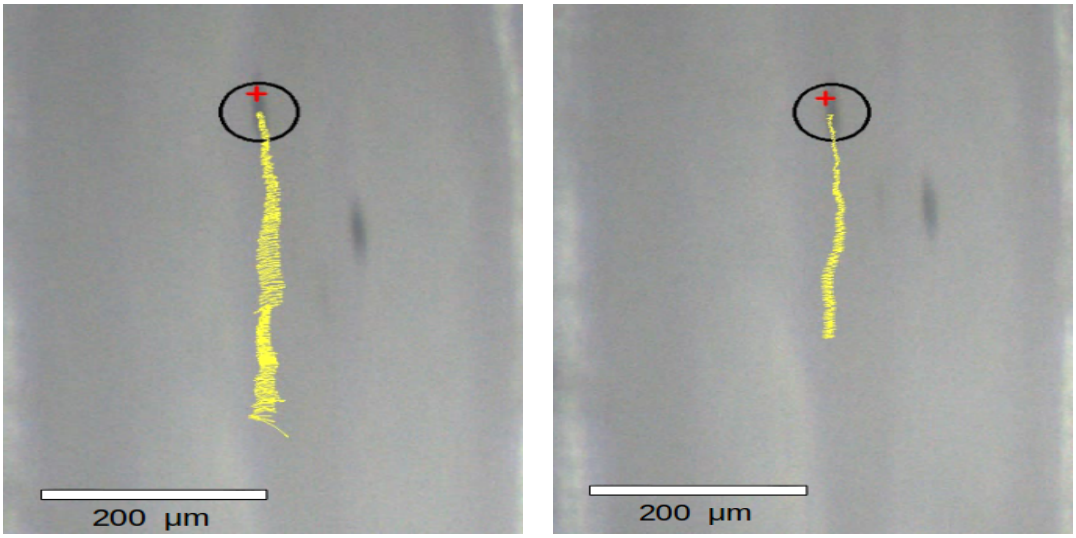



Fig. 3.3.: Numerical differentiation velocity without FIR.

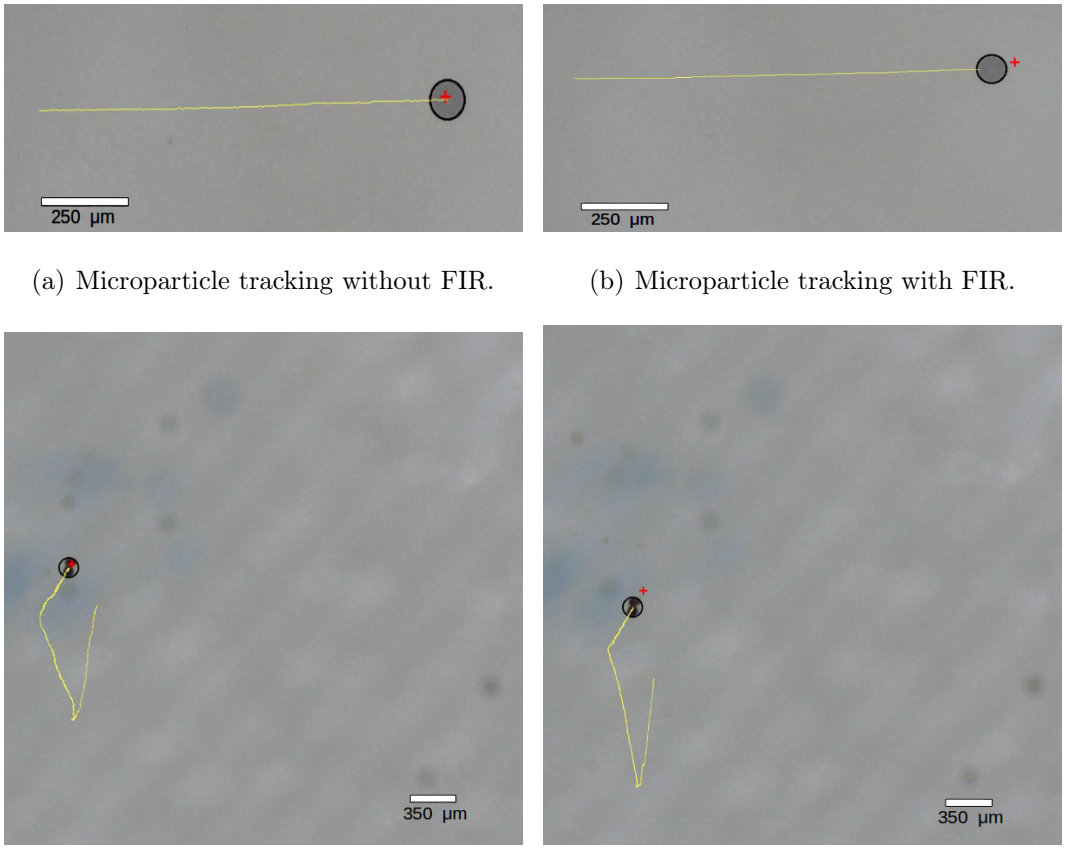
In conclusion, the difference between (a) and (b) is that using running average filter with its special case FIR (Finite Impulse Response). Then, we can notice the reduction in the amplitude of the velocity noise we get.

In the following figures, we present the effect of FIR on the motion tracking of the MagnetoSperm and Microparticle.




(a) MagnetoSperm tracking without FIR. (b) MagnetoSperm tracking with FIR.

Fig. 3.4.: FIR effect on MagnetoSperm experiments.




(a) Microparticle tracking without FIR.

(b) Microparticle tracking with FIR.

(c) Microparticle tracking without FIR.

(d) Microparticle tracking with FIR.

Fig. 3.5.: FIR effect on Microparticles experiments.

**Motion tracking trajectory of the MagnetoSperm and drawing the desired closed trajectory**

We manage to track the MagnetoSperm motion and draw it online during the controlling process. Also, we can compare the drawn path of the agent motion with the actual trajectory path that we drew before controlling, this technique gives us a lot of confidence in our controller and makes us go much further through our experiments. Also, drawing the trajectory for the agent makes us able to control the MagnetoSperm in more complicated environments. This control is applicable, because we are able to

decide the MagnetoSperm path till it reaches the desired setpoint like the experiments are done in the bifurcations channel chapter 2 in section 2.2.2.

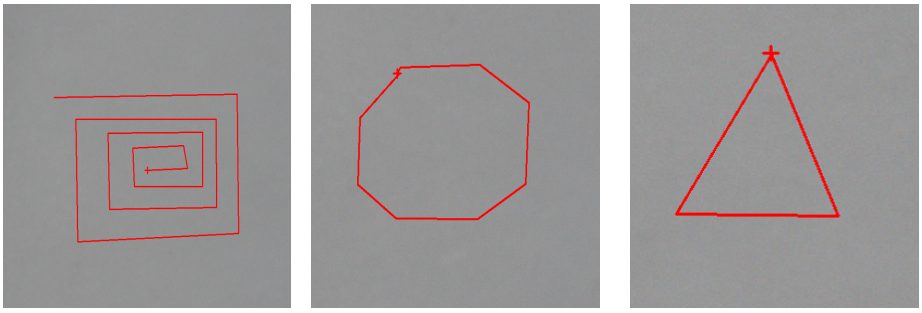


Fig. 3.6.: Some of the closed trajectory paths used in experiments.

Here, we present some of our experiments on Microparticles in order to test our tracking system through the motion tracker and the trajectory path drawing.

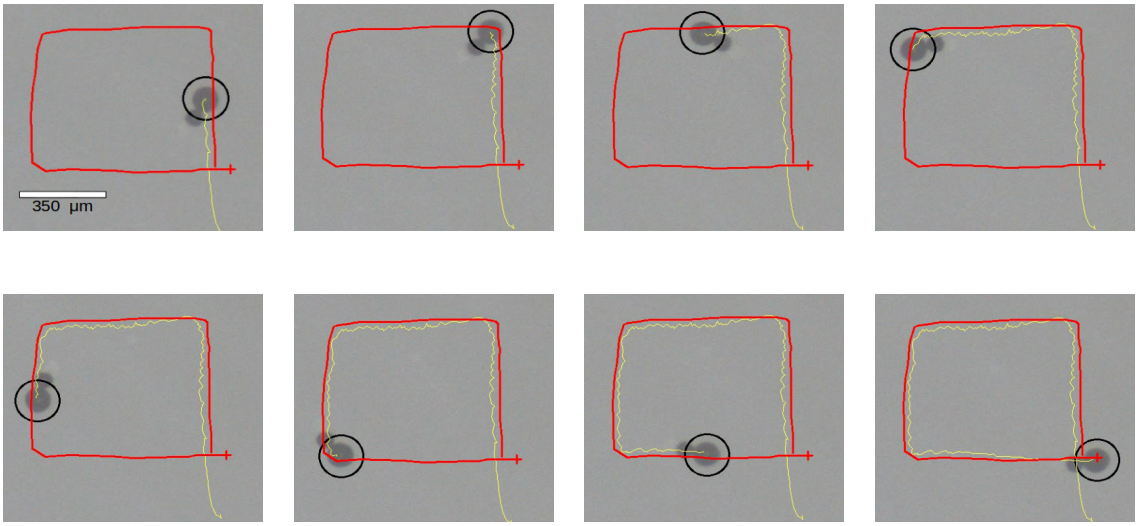



Fig. 3.7.: Testing a square trajectory with Microparticle.

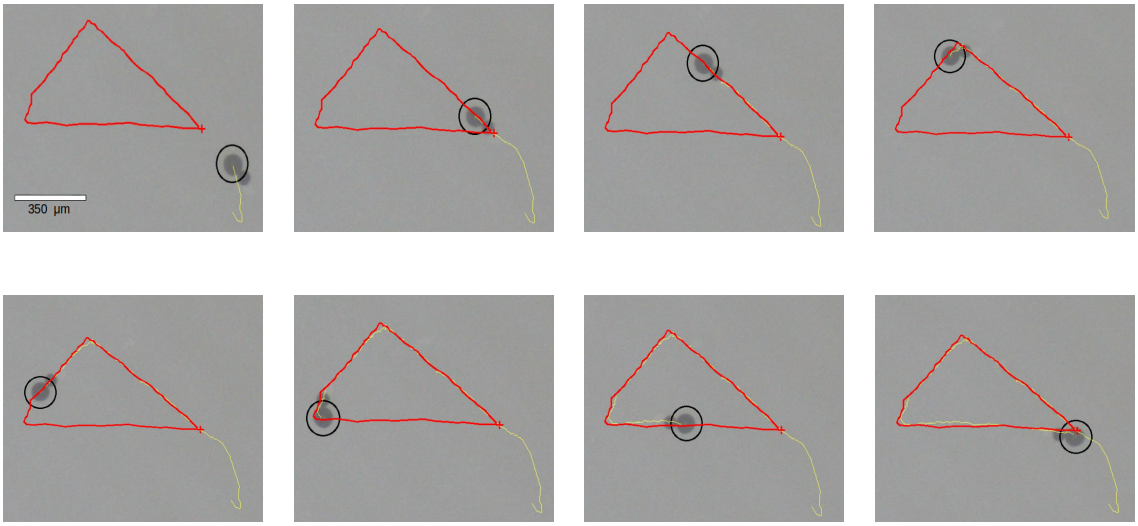



Fig. 3.8.: Testing a triangle trajectory with Microparticle.

### 3.2.2 MagnetoSperm motion control strategy and technique

We present our motion control theory technique for controlling the translational and rotational motion of the MagnetoSperm.

Based on the translational and rotational dynamics of the MagnetoSperm, we design our control strategy. We divide our design technique into two main parts: $(i)$ Directing and orienting the MagnetoSperm towards the direction of motion; and $(ii)$ oscillation of the MagnetoSperm tail in order to be able to swim.

Therefore, we use two coils for the uniform magnetic field for directing and orienting the MagnetoSperm. The other two coils are used for oscillating. The choose of the coils is based on the theta of MagnetoSperm motion with respect to the x-axis. This theta is measured by

$$\theta = \tan^{-1}\left(\frac{|x_d|}{|y_d|}\right) \tag{3.14}$$

where $x_d$ and $y_d$ are the x and y distances between the setpoint and the MagnetoSperm head position. Each electromagnet either produces an oscillating field or a uniform field within the center of the workspace. The electromagnets producing the oscillating

fields are always in the opposite direction to the direction of movement of the MagnetoSperm. The other two electromagnets provide uniform magnetic fields towards the reference position. The electromagnets that generate uniform magnetic fields are determined based on the position of the reference (small blue circle) and the MagnetoSperm. Then the current magnitude at each of these electromagnets is calculated to adjust the direction of the MagnetoSperm. When $\theta$ is 45, both electromagnets are supplied with the same current input, i.e., k. As $\theta$ increases or decreases, the current input on one electromagnet must be decreased to allow the other electromagnet to orient the MagnetoSperm towards its direction.

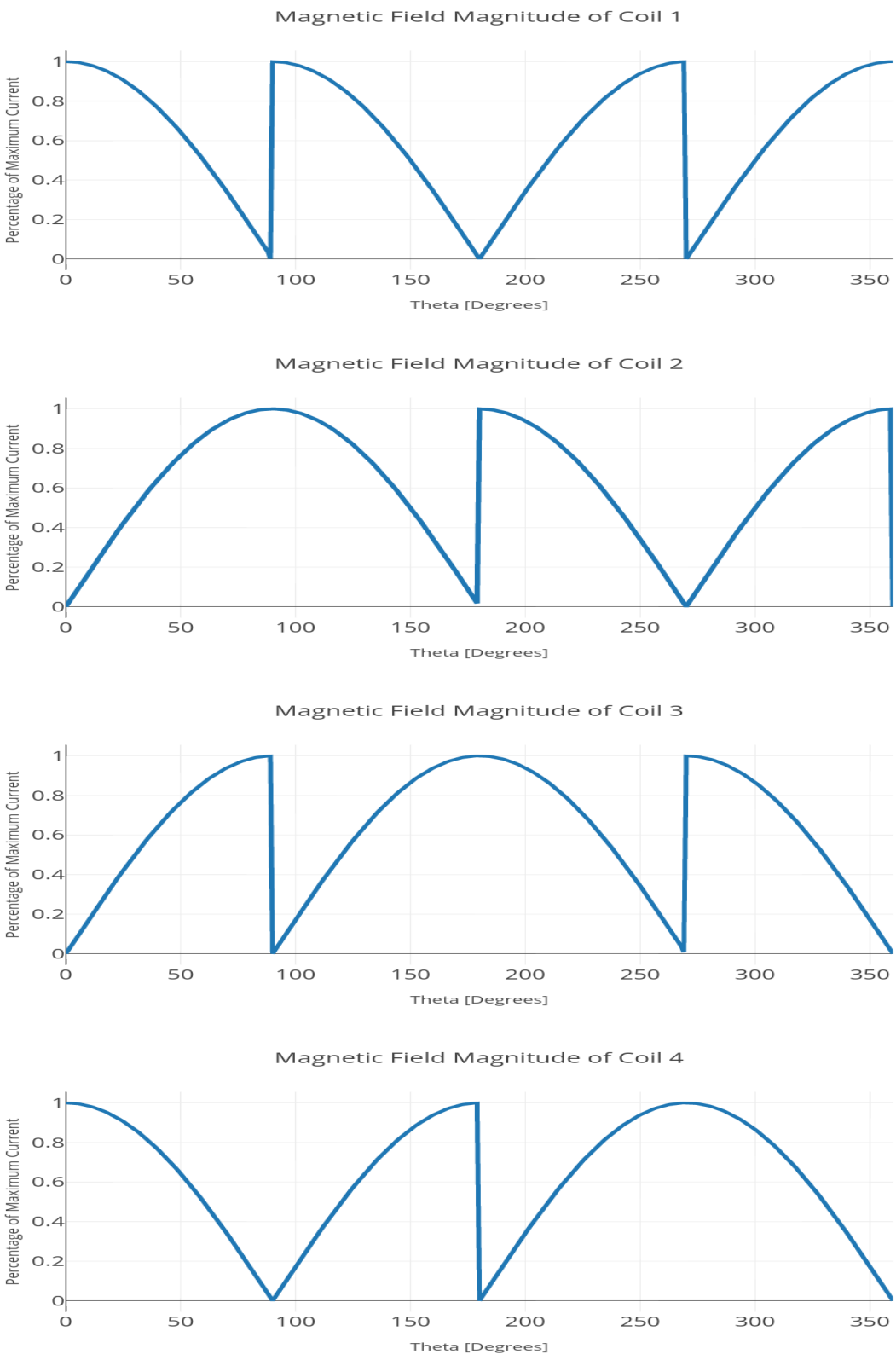



Fig. 3.9.: Magnetic field magnitude ratio for the 4 MobiMag coils.

**Overview of PID control design**

The proportional-integral-derivative controller (PID controller) is a control loop feedback mechanism that depend mainly on the error status of the system between a current state and a setpoint and adjust the system behavior to enable it to reach the reference. A PID controller consists mainly of: $(i)$ A proportional term $(P)$ that depends on the present error; $(ii)$ Integral term $(I)$ that depends on the accumulation of past errors; and $(iii)$ a derivative term $(D)$ is a prediction of future errors, based on current rate of change. A PID control is mainly evaluated by the system characteristics and behavior. First, by how much the system overshoots when it reaches the desired setpoint. Second, the steady state error between the final output and the desired reference. Third, the time term affects the system behavior according to the following table :

| Parameter | Overshoot | Settling time | Steady state error | Rise time |
|---|---|---|---|---|
| $K_p$ | increase | small change | decrease | decrease |
| $K_i$ | increase | increase | eliminate | decrease |
| $K_d$ | decrease | decrease | no effect | minor change |

Table 3.1: PID controller gains effect.

A good PID controller is mainly judged by its optimization, stability and the time needed to eliminate the steady state error. Tuning can be done manually based on the proper understanding of the control parameters.
Also, there is a heuristic tuning method called Ziegler-Nichols method introduced by John G.Ziegler and Nathaniel B. Nichols in the 1940s. It starts with setting the $K_i$ and $K_d$ gains to zero. The proportional gain $K_p$ is increased until it reaches a certain value called $K_u$, at which the output starts oscillating. $K_u$ and the oscillation period $T_u$ are used to adjust the gains as shown:

| Control Type | $K_p$ | $K_i$ | $K_d$ |
|---|---|---|---|
| $P$ | $0.5K_{u/T_u}$ | 0 | 0 |
| $PI$ | $0.45K_u/T_u$ | $0.54K_u/T_u$ | 0 |
| $PID$ | $0.60K_u/T_u$ | $1.2K_u/T_u$ | $3K_uT_u/40$ |

Table 3.2: Ziegler-Nichols method.

**Point-to-point PID motion control**

Here, we present our PID closed-loop controller. The MagnetoSperm swims using the main thrust force from its tail oscillations. We achieve this tail oscillations using our electromagnets to apply magnetic forces in different magnitudes and directions. Therefore, we use our PID controller to control these forces to have the desired output.

The magnetic forces of these electromagnets to be controlled depend on many parameters like: $(i)$ The distance between the agent position and the setpoint; $(ii)$ the velocity of the MagnetoSperm we want it to swim with; $(iii)$ the direction the agent should swim to reach the setpoint; and $(iv)$ the translational and rotational dynamics of the MagnetoSperm.

The MagnetoSperm exerts two magnetic forces in two directions $F_x$ and $F_y$. Due to the symmetry of the two dimensional configuration, we can assume that the control signal needed for one force is the same needed for the other. Therefore, our control signal of each force can be described in the form:

$$F_m(p) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \tag{3.15}$$

where this form consists of three components: $(i)$ Proportional error $e$; $(ii)$ integral error $\int_0^t e(\tau)d\tau$; and $(iii)$ dervative error $\frac{de(t)}{dt}$.

The proportional error is used to reduce the steady state error of the controller as much as possible through continually getting the difference between the desired setpoint and the MagnetoSperm.

Since the MagnetoSperm moves using its tail oscillation. Therefore, we have to set two desired setpoints for the head of the MagnetoSperm, not only one. The two setpoints should be on both sides of the main desired setpoint. We can explain more using the 3.1.

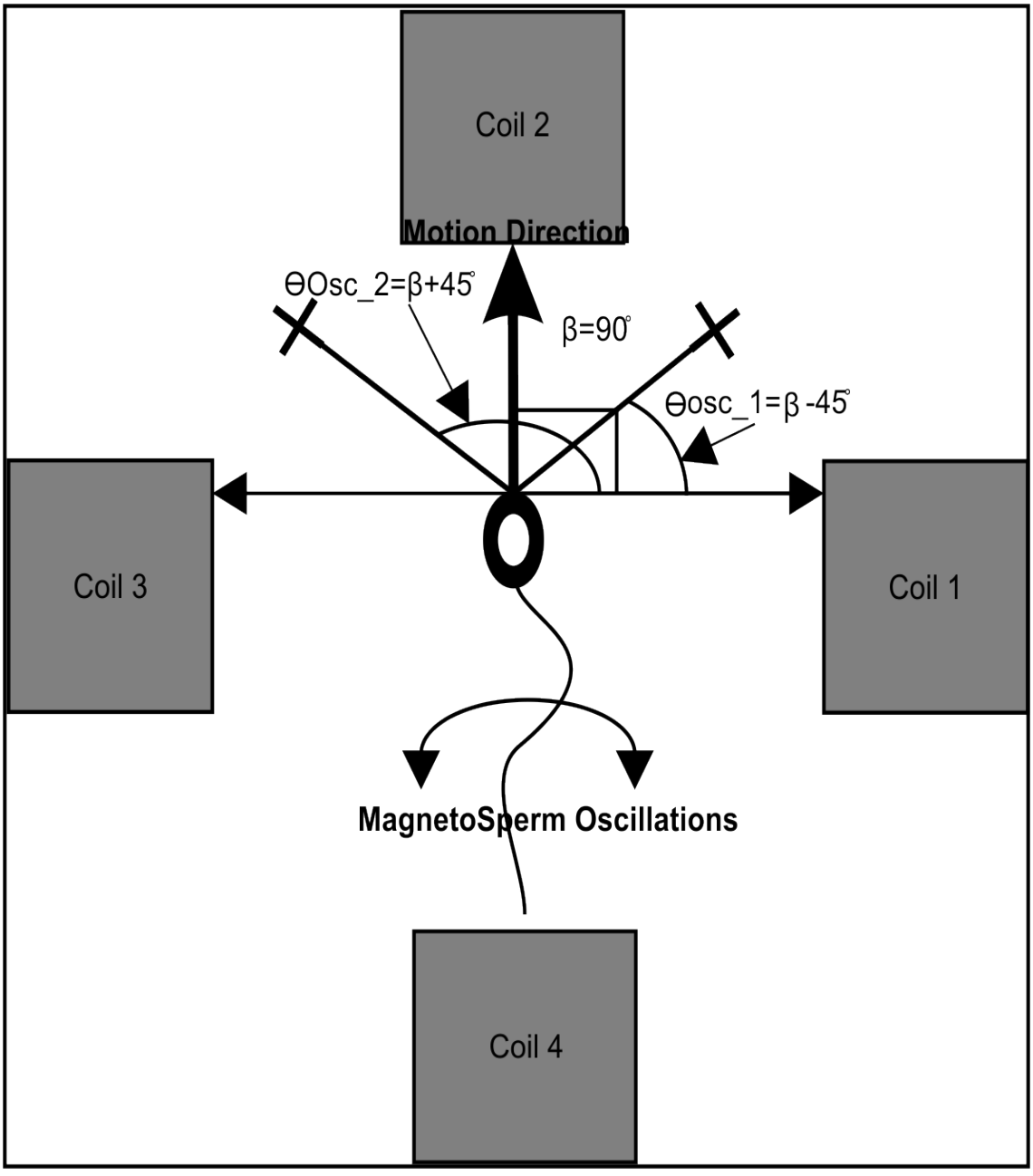



Fig. 3.10.: Oscillation angles of MagnetoSperm swimming straight towards Coil 2.

Therefore, to calculate our proportional error, we have to know how far is the main setpoint from the agent position.

$$\begin{aligned} x_d(t) &= x_{ref}(t) - x(t) \\ y_d(t) &= y_{ref}(t) - y(t). \end{aligned} \tag{3.16}$$

Then, calculating the radius $R(t) = \sqrt{x_d^2(t) + y_d^2(t)}$ of the circle that its center point is the agent position and the main setpoint is on the circle. The proportional error is the distance from the agent position to each one of the secondary set points

$$\begin{aligned} e_x(t) &= R\cos(\theta_{osc}) \\ e_y(t) &= R\sin(\theta_{osc}) \end{aligned} \tag{3.17}$$

where $\theta_{osc}$ is how much the agent oscillates on the two sides of the motion direction. Its value equals $\beta \pm 45$.

The integral error, $\int_0^t e(\tau)d\tau$ is used to accelerate the movement towards the setpoint and eliminate the steady-state error. Also, it integrates and continually sum the instantaneous controller error, $e(t)$, over time and gives the accumulated offset that should corrected previously.

In our controller, the integral term in x direction is defined as

$$\begin{aligned} \int e_x(t)dt &\approx \ t((e_x(t) + e_x(t-1))/2) \\ \int_0^t e_x(\tau)d\tau &\approx \int e_x(t)dt + \int e_x(t-1)dt. \end{aligned} \tag{3.18}$$

The dervative error, $\frac{de(t)}{dt}$ aims at flattening the error trajectory into a horizontal line, dampening the force applied, and so reduces overshoot. Our controller defines term derivative term in the form

$$\frac{de_x(t)}{dt} \approx -\frac{e_x(t-1) - e_x(t)}{dt}. \tag{3.19}$$

All of the previous equations for the error of $x$ position is applied in the same way on the $y$ position. Finally, we have two control signal (3.15) for each $x$ and $y$ position control.

### 3.2.3 MagnetoSperm velocity estimation

In this section, we present how we estimate the velocity of the MagnetoSperm.

We use more than one method to compare their results until we get the best and most accurate results. All the methods are used offline. We perform some mathematical operations on the experimental data we get. These mathematical operations are applied via Matlab (MathWorks, Natick, Massachusetts, U.S.A).

First, we try to estimate the agent velocity with numerical differentiation method. We use this method to measure the $v_x$ and $v_y$ velocities. The differentiation of each one of the two velocities is in the form:

$$v_x(t) \approx (p_x(t) - p_x(t-1))/Ts \tag{3.20}$$

$$v_y(t) \approx (p_y(t) - p_y(t-1))/Ts \tag{3.21}$$

After that, we get the equivalent velocity component of the $x$ and $y$ velocities.

$$v(t) = \sqrt{v_x(t)^2 + v_y(t)^2} \tag{3.22}$$

Second, we try the iterative learning observer (ILO) method. ILO is proposed to alleviate the constraint of identifying constant and slowly time-varying parameters by adaptive observers. We use [10],[12] for the iterative learning observer design to identify time-varying parameters relies on the additional term that is updated in real time by previous information. The updating mechanism is called learning. If we consider a linear system described by

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) + F\theta(t) \\ y(t) &= Cx(t) \end{aligned} \tag{3.23}$$

where $x(t) \in \mathbb{R}^n, y(t) \in \mathbb{R}^p, u(t) \in \mathbb{R}^m, \theta \in \mathbb{R}^q$ the unknown time-varying parameter that may be constant, periodic, or aperiodic signal, and $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times m}, C \in$

$\mathbb{R}^{p\times n}, F \in \mathbb{R}^{n\times q}$ are constant matrices.

The iterative learning observer is given by the following equations:

$$\begin{aligned}\dot{\hat{x}}(t) &= A\hat{x}(t) + Bu(t) + L(y(t) - \hat{y}(t)) + Fv(t)\\ \hat{y}(t) &= C\hat{x}(t)\\ v(t) &= K_1 v(t-\tau) + K_2(y(t) - \hat{y}(t))\end{aligned} \tag{3.24}$$

where $\hat{x}$ and $\hat{y}$ are the estimated state and output. The matrix $L$ is selected such that $(A - LC)$ is stable. The signal $v(t)$ is the estimate of the time-varying parameter, $\theta(t)$. It is updated online by both its previous information, $v(t-\tau)$, and output estimation errors, $y(t) - \hat{y}(t)$. The parameter, $\tau > 0$, is the updating interval. A better choice of the parameter, $\tau$, is the sampling interval. $K_1 \in \mathbb{R}^{q\times q}, K_2 \in \mathbb{R}^{q\times p}$ are the gain matrices.

In our system, the estimated velocity is obtained by

$$\begin{aligned}\hat{v}_x(t) &= K_1\hat{v}_x(t-1) + K_2(x(t) - \hat{x}(t))\\ \hat{v}_y(t) &= K_1\hat{v}_y(t-1) + K_2(y(t) - \hat{y}(t))\\ \hat{x}(t+1) &= \hat{x}(t) + Ts(L(x(t) - \hat{x}(t)) + \hat{v}_x(t))\\ \hat{y}(t+1) &= \hat{y}(t) + Ts(L(y(t) - \hat{y}(t)) + \hat{v}_y(t))\end{aligned} \tag{3.25}$$

where $\hat{x}$ and $\hat{y}$ are the estimated agent $x$ and $y$ position.

Third, we use the built-in zero-phase filter in Matlab. Zero-phase filter form is $output = filtfilt(b, a, input)$. It performs zero-phase digital filtering by processing the input data, in both the forward and reverse directions. The zero-phase filter operates along the first non-singleton dimension of the input data. The vectors a, b are Transfer function coefficients of the filter. They are row vectors of length n + 1 for lowpass and highpass filters and 2n + 1 for bandpass and bandstop filters.

For digital filters, the transfer function is expressed in terms of b and a as

$$H(z) = \frac{B(z)}{A(z)} = \frac{b(1) + b(2)z^{-1} + ... + b(n+1)z^{-n}}{a(1) + a(2)z^{-1} ... + a(n+1)z^{-n}} \tag{3.26}$$

For analog filters, the transfer function is expressed in terms of b and a as

$$H(s) = \frac{B(s)}{A(s)} = \frac{b(1)s^n + b(2)s^{n-1} + ... + b(n+1)}{a(1)s^n + a(2)s^{n-1} + ... + a(n+1)} \tag{3.27}$$

Also, the zero-phase filter minimizes start-up and ending transients by matching initial conditions, and we can use it for both real and complex inputs. The output of zero-phase filter has the characteristics:

- Zero-phase distortion
- A filter transfer function, which equals the squared magnitude of the original filter transfer function
- A filter order that is double the order of the filter specified by b and a vectors.

We compare all of these three methods to get the best and most accurate estimation of the MagnetoSperm velocity. Therefore, we present the comparison

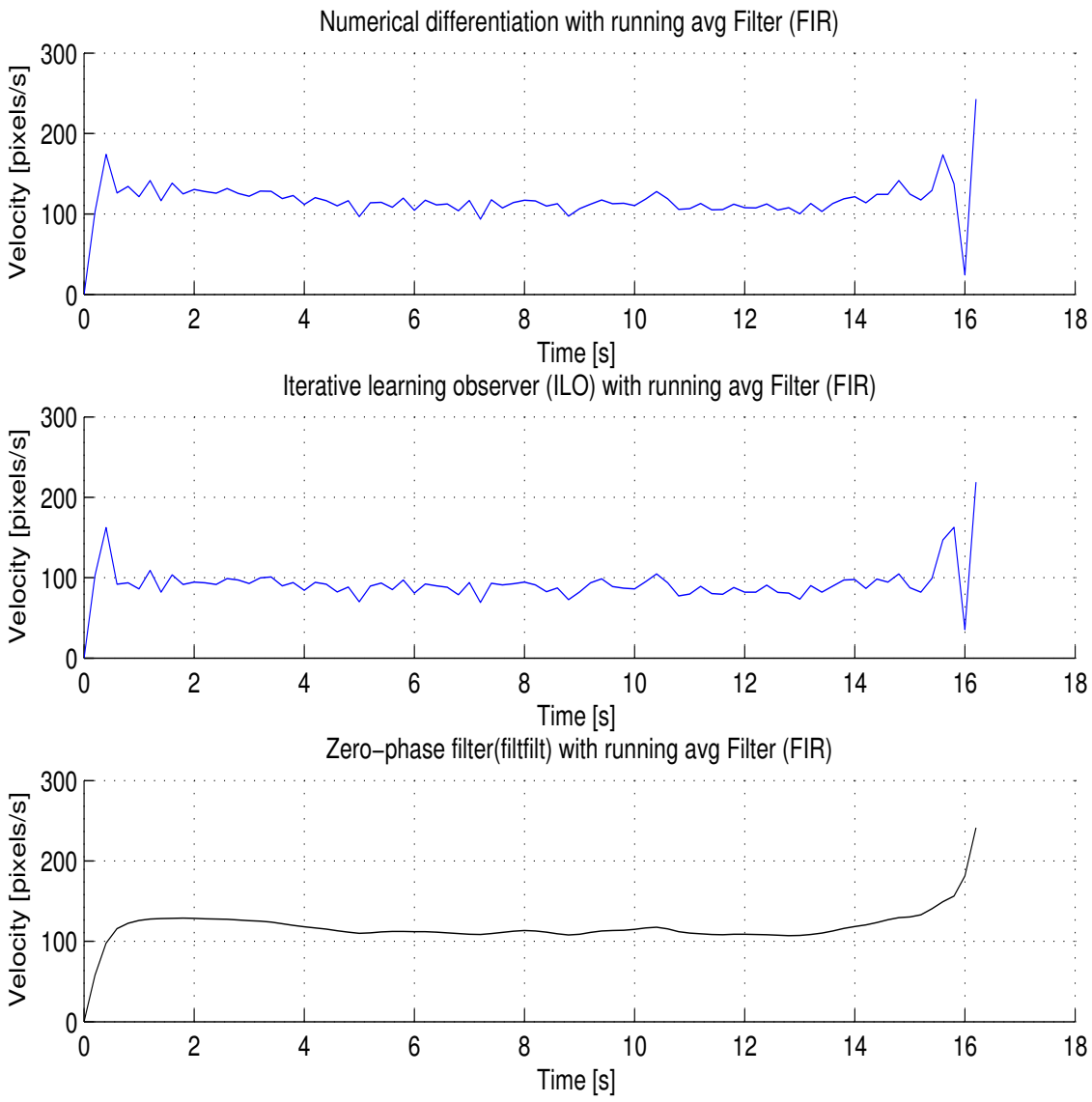



Fig. 3.11.: Comparison between numerical differentiation, iterative learning observer and Zero-phase filter for some of our experimental data.

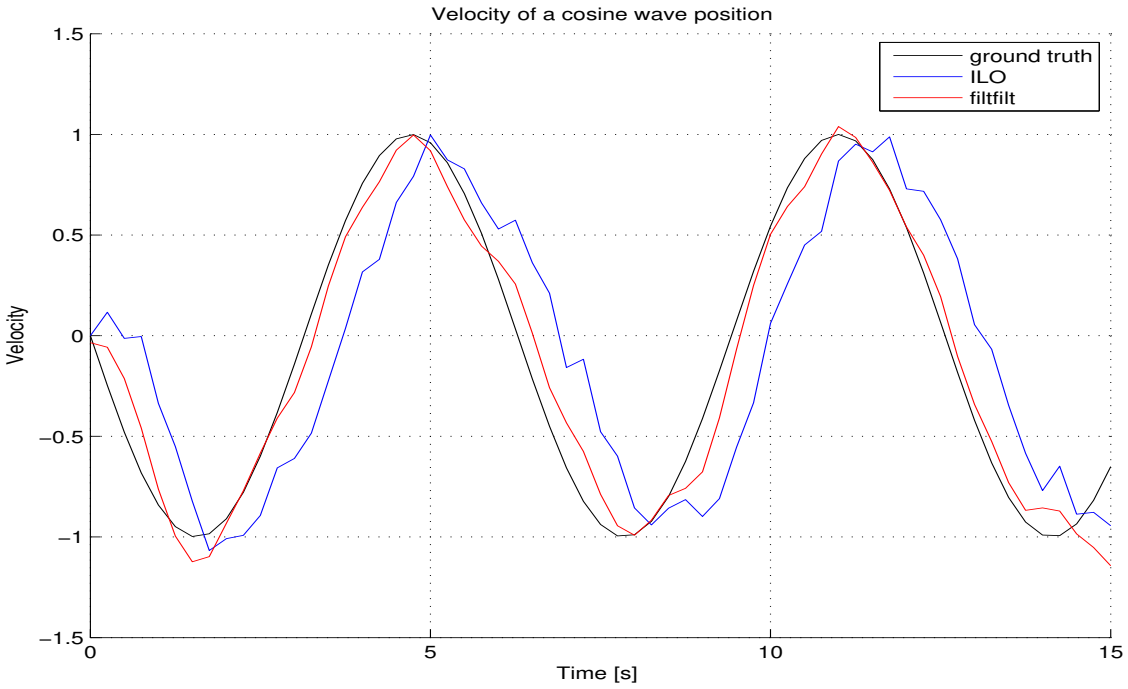



Fig. 3.12.: Comparison between iterative learning observer and Zero-phase filter for position data of cosine wave and velocity of -sine wave.

In conclusion, Zero-phase filter is the least method that have noise, most accurate with respect to the ground truth and the best method for this agent velocity estimation. Also, it is hard to tune to the iterative learning observer gains in an accurate way. Therefore, all our results for the agent average velocity are estimated using Zero-phase filter in Matlab.

# 4. EXPERIMENTAL PROCEDURE

## 4.1 Motion control of MagnetoSperm

In this chapter, we will show the way we perform the experiments. We perform the motion control of the MagnetoSperm in a petri-dish of diameter 4 cm using water and glycerol mixture. These experiments include closed-loop PID control point-to-point and closed trajectories. Also, one aim of these experiments is to prove the effect of FIR position tracking filter on the MagnetoSperm movement. We manage to validate our PID controller on the MagnetoSperm and get the best tuning for the PID gains through as many and different experiments as we can.

### 4.1.1 Point-to-point manipulation

We start with closed-loop using PID controller. We manage to do point-to-point experiments to see the behavior of the MagnetoSperm and validate our controller strategy and technique. Moreover, one of the benefits is to determine the best ratio between the length of the head and tail of the MagnetoSperm and we discuss this in details in chapter 5. We try in these experiments to minimize the position MagnetoSperm error with respect to the setpoint.

## 4.2 Characterization of MagnetoSperm inside fluidic channels

In this section, we try to investigate the effect of the channel walls width on the MagnetoSperm propulsion force. These experiments are done with many trials to get more accurate results about the dynamics and velocity of the MagnetoSperm. Also, we discuss the effect of getting the distance from walls smaller within the experiments by changing the precession angle of the MagnetoSperm swimming direction. All of our experiments are done in open fluidic channel like figure 2.2 and 2.4 within the glycerol and water mixture with its characteristics as we mention in chapter 3 in page 14

### 4.2.1 Effect of channel width on the MagnetoSperm dynamics

There is an effect from the solid boundaries on Bacteria [6]. Therefore, we start to investigate this on our microagents. We perform 24 experiments with Microparticles through three different fluidic channel widths (1 mm, 0.75 mm, 0.5 mm). Then, we start with the MagnetoSperm to do the same experiments, but in four different widths (1 mm, 0.75 mm, 0.5 mm, 0.25 mm) doing 8 experiments in each channel with 5 Hz oscillation frequency and it ends with 32 successful experiments. We perform half of the experiments back and half of them forth to remove any gradient effect by the coils to only measure the oscillation magnetic field effect.

### 4.2.2 Effect of different precession angles and frequency response on the MagnetoSperm

Here, we study the merged effect of the two parameters on the velocity and dynamics of the MagnetoSperm.

The precession angle, $\beta$ is what we refer to in chapter 3 in section 3.1.2. It indicates the angle between the magnetic dipole moment of the MagnetoSperm and the induced magnetic field of the coils. Therefore, we perform experiments on different precession

angles for each channel width to see its effect on the MagnetoSperm dynamics. We choose 4 angles from 35 till 65 degrees with a step of 10 degrees in-between. The experiments are done using the same closed loop controller for all the other experiments. The angle is set autonomously during the controlling process without any interface from the user to make sure of the precession angle is same during all the experiments.

The frequency of the oscillated tail has a very great effect on oscillated microagents and this what we discuss in chapter 5. The experiments are performed at 5 frequencies begins from 5 Hz till 25 Hz with difference scale of 5 Hz.

We manage to perform these experiments to study best conditions for the agent to swim depending on which oscillation frequency and precession angle.

### 4.2.3 Experiments inside the Aorta channel

In order to have a complicated environment for the MagnetoSperm to move in, we perform experiments inside a channel inspired by the human body Aorta shape. First, we perform experiments on Microparticles inside the Aorta channel. Second, we have experiments on the MagnetoSperm with different predefined paths. The experiments in the Aorta channel shows how we can control the MagnetoSperm autonomously with predefined path inside inside the human body arteries.

# 5. EXPERIMENTAL RESULTS

## 5.1 Characterization inside open fluidic channels

### 5.1.1 Influence of the channel wall on the flagellar propulsion of the MagnetoSperm

As we mentioned before in chapter 4 in section 4.2.1, we perform experiments on the MagnetoSperm in open Channels of different widths.
The results are quite close to what we expected due to the effect of the channel width and drag force on the flagellar propulsion of the MagnetoSperm. However, before doing the experiments on the MagnetoSperm, we did on Microparticle using the gradient effect to prove our concept of the walls effect on the agent dynamics.

Fig. 5.1 reports the result of the experimental validation.

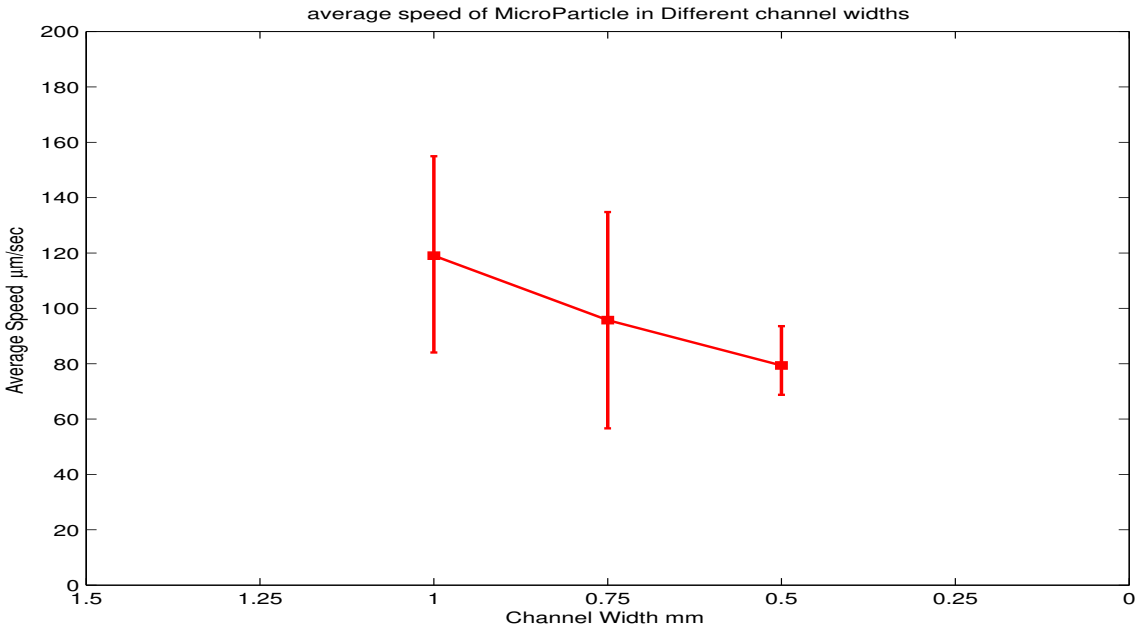



Fig. 5.1.: Average velocity of the Microparticle in three different channel widths.

As what we present in the graph, there is a very clear relation between the velocity of the Microparticle and the channel width. The wider the open channel is, the faster the Microparticle is because of less drag force acts on it.

The average velocity of each channel width was as follows:
Channel 1.0mm :119.52 $\mu$m/s
Channel 0.75mm: 96.57 $\mu$m/s
Channel 0.5mm: 79.45 $\mu$m/s.

After we proved our concept, we move to the MagnetoSperm experiments in the different channel widths, but this time we add one more channel width which is 0.25 mm to investigate and validate more of our results and conclusions. The results are reported in Fig. 5.2,

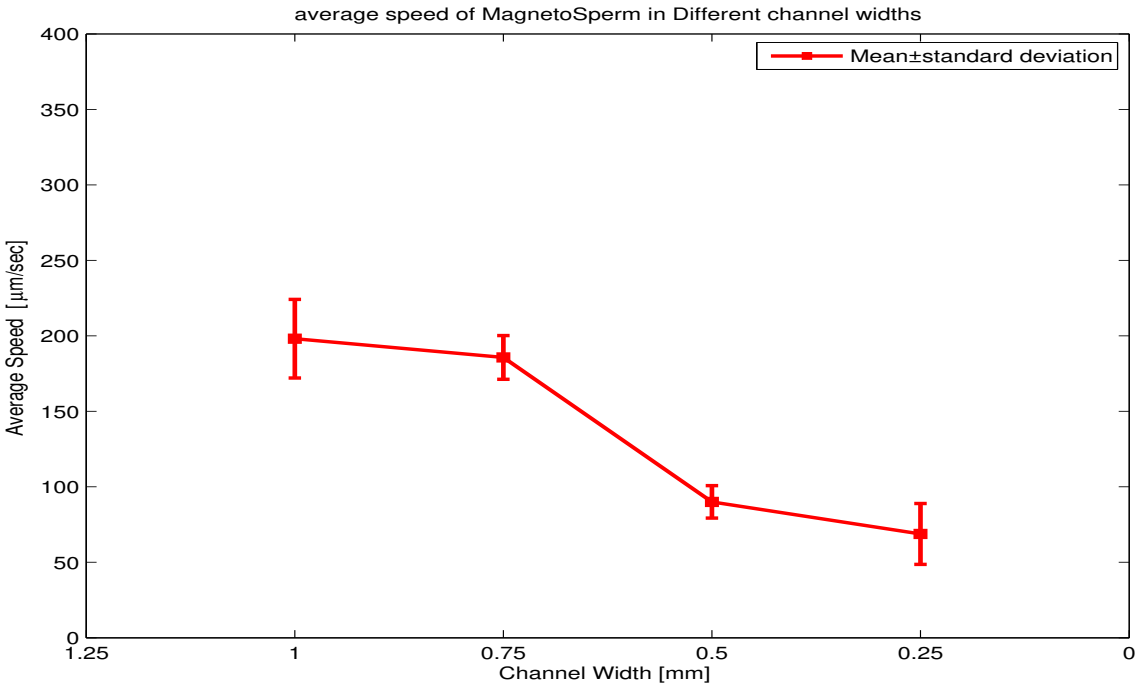



Fig. 5.2.: Average velocity of the MagnetoSperm in four different channel widths.

The results of the MagnetoSperm experiments demonstrate what we first proved in the Microparticle experiments. Also, we can notice here how is the MagnetoSperm

affected by the channel width. As the MagnetoSperm velocity is decreasing with decreasing the channel width, the average velocity of the agent in all the different channel widths were as follows:

Channel 1.0mm : 198.08 $\mu$m/s

Channel 0.75mm : 185.7 $\mu$m/s

Channel 0.5mm : 90 $\mu$m/s

Channel 0.25mm : 68.8 $\mu$m/s

The most obvious thing is the big difference between the agents velocities in the first two channels with respect to the last two ones. This big difference is because of the width of 0.5 mm is small for the MagnetoSperm to complete its oscillation amplitude. The decreasing in the oscillation amplitude affected the MagnetoSperm dynamics. This affect may be compensated by increasing the oscillation frequency to reduce the amplitude and increase the gradient effect on the MagnetoSperm.

### 5.1.2 Oscillation frequency response and precession angles effect

We perform 80 experiments on the same MagnetoSperm to have a fair comparison between all the results. The MagnetoSperm velocity and results are calculated based on 4 experimental trials at each precession angle at each frequency.

The results we get are very interesting, because they explained a lot about the characterization of the MagnetoSperm.

The next two graphs show the same data, but in two different ways.

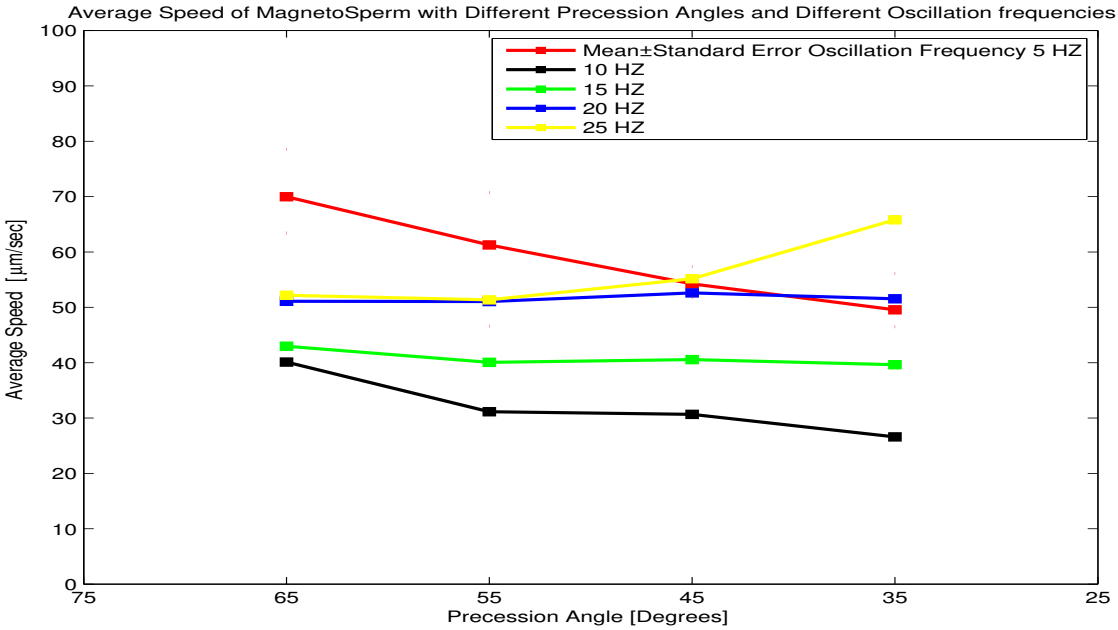



Fig. 5.3.: Average velocity of MagnetoSperm at each frequency along all the precession angles in channel width 1 mm.

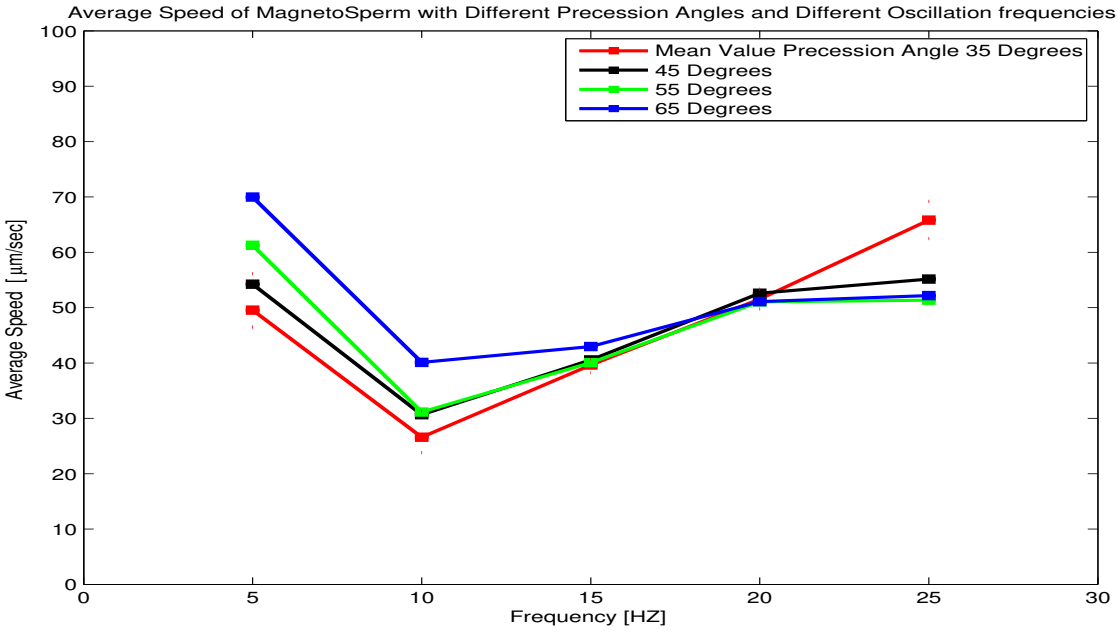



Fig. 5.4.: Average velocity of MagnetoSperm at each precession angle along all the frequencies in channel width 1 mm.

The average velocity of the MagnetoSperm can be presented as follows:

| | Frequency | | | | |
|---|---|---|---|---|---|
| Precession angle | 5 Hz | 10 Hz | 15 Hz | 20 Hz | 25 Hz |
| 35 | 49.55 $\mu$m/s | 26.6 $\mu$m/s | 39.64 $\mu$m/s | 51.54 $\mu$m/s | 65.82 $\mu$m/s |
| 45 | 54.23 $\mu$m/s | 30.65 $\mu$m/s | 40.55 $\mu$m/s | 52.6 $\mu$m/s | 55.15 $\mu$m/s |
| 55 | 61.27 $\mu$m/s | 31.13 $\mu$m/s | 40.06 $\mu$m/s | 51.05 $\mu$m/s | 51.34 $\mu$m/s |
| 65 | 69.97 $\mu$m/s | 40.12 $\mu$m/s | 43.0 $\mu$m/s | 51.1 $\mu$m/s | 52.17 $\mu$m/s |

Table 5.1: Average velocity of MagnetoSperm with respect to precession angles and frequencies.

Moreover, we perform 60 experiments on a MagnetoSperm, but this time in channel width 0.75 mm. The MagnetoSperm velocity is calculated based on 4 trials for each precession angle at each oscillation frequency.
The results we get prove some of our conclusions from the previous experiments and other conclusions that are based on the different channel width.
The next two graphs show the same results, but in two different ways.

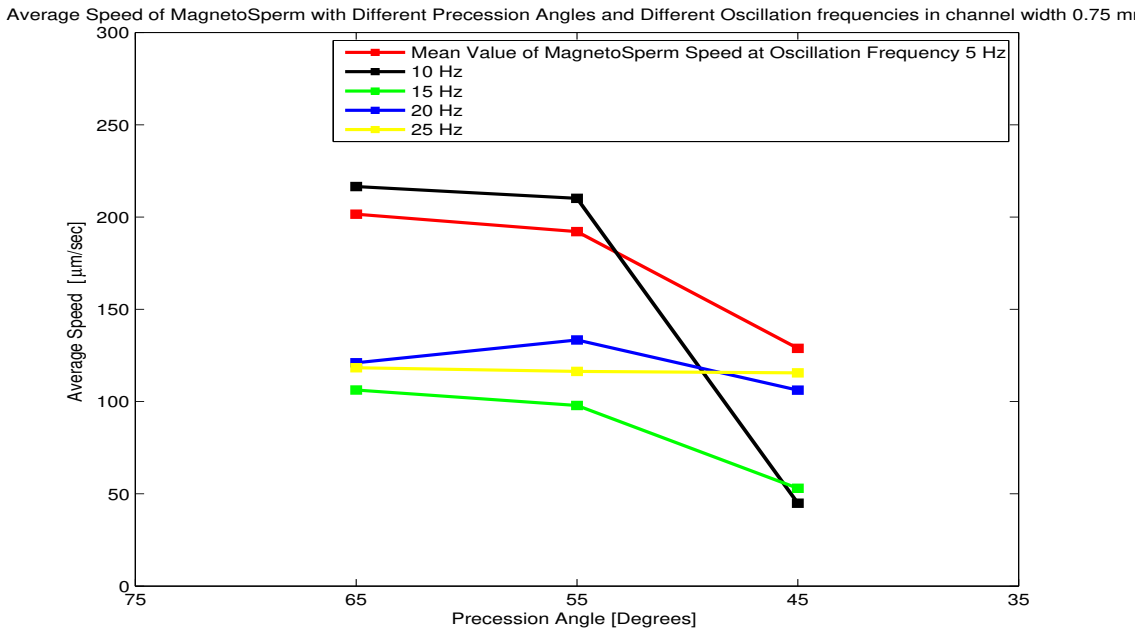



Fig. 5.5.: Average velocity of MagnetoSperm at each frequency along all the precession angles in channel width 0.75 mm.

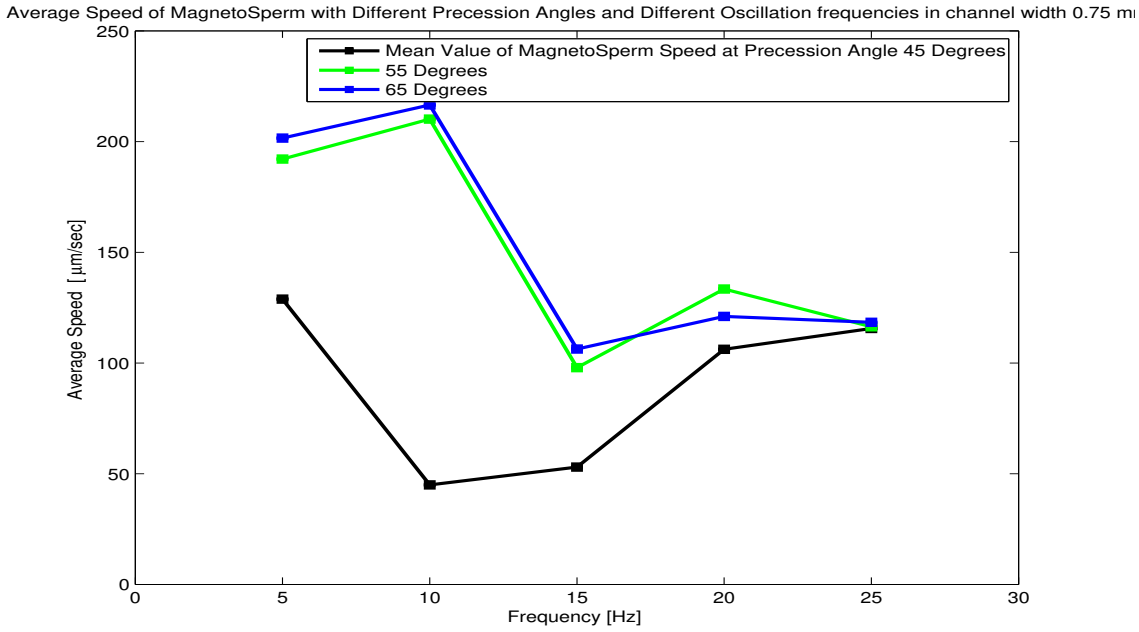



Fig. 5.6.: Average velocity of MagnetoSperm at each precession angle along all the frequencies in channel width 0.75 mm.

The average velocity of the MagnetoSperm can be presented as follows:

| | Frequency | | | | |
|---|---|---|---|---|---|
| Precession angle | 5 Hz | 10 Hz | 15 Hz | 20 Hz | 25 Hz |
| 45 | 128.8 $\mu$m/s | 44.92 $\mu$m/s | 53.0 $\mu$m/s | 106.2 $\mu$m/s | 115.6 $\mu$m/s |
| 55 | 192.1 $\mu$m/s | 210.2 $\mu$m/s | 97.88 $\mu$m/s | 133.4 $\mu$m/s | 116.3 $\mu$m/s |
| 65 | 210.6 $\mu$m/s | 216.5 $\mu$m/s | 106.3 $\mu$m/s | 121.0 $\mu$m/s | 118.4 $\mu$m/s |

Table 5.2: Average velocity of MagnetoSperm with respect to precession angles and frequenciesin channel width 0.75 mm.

### 5.1.3 Experiments inside the Aorta channel

We perform experiments on Microparticles to validate and evaluate our controller inside the Aorta channel. The experimental results prove the controller theory and the tracking algorithm for the predefined bath for the Microparticle.

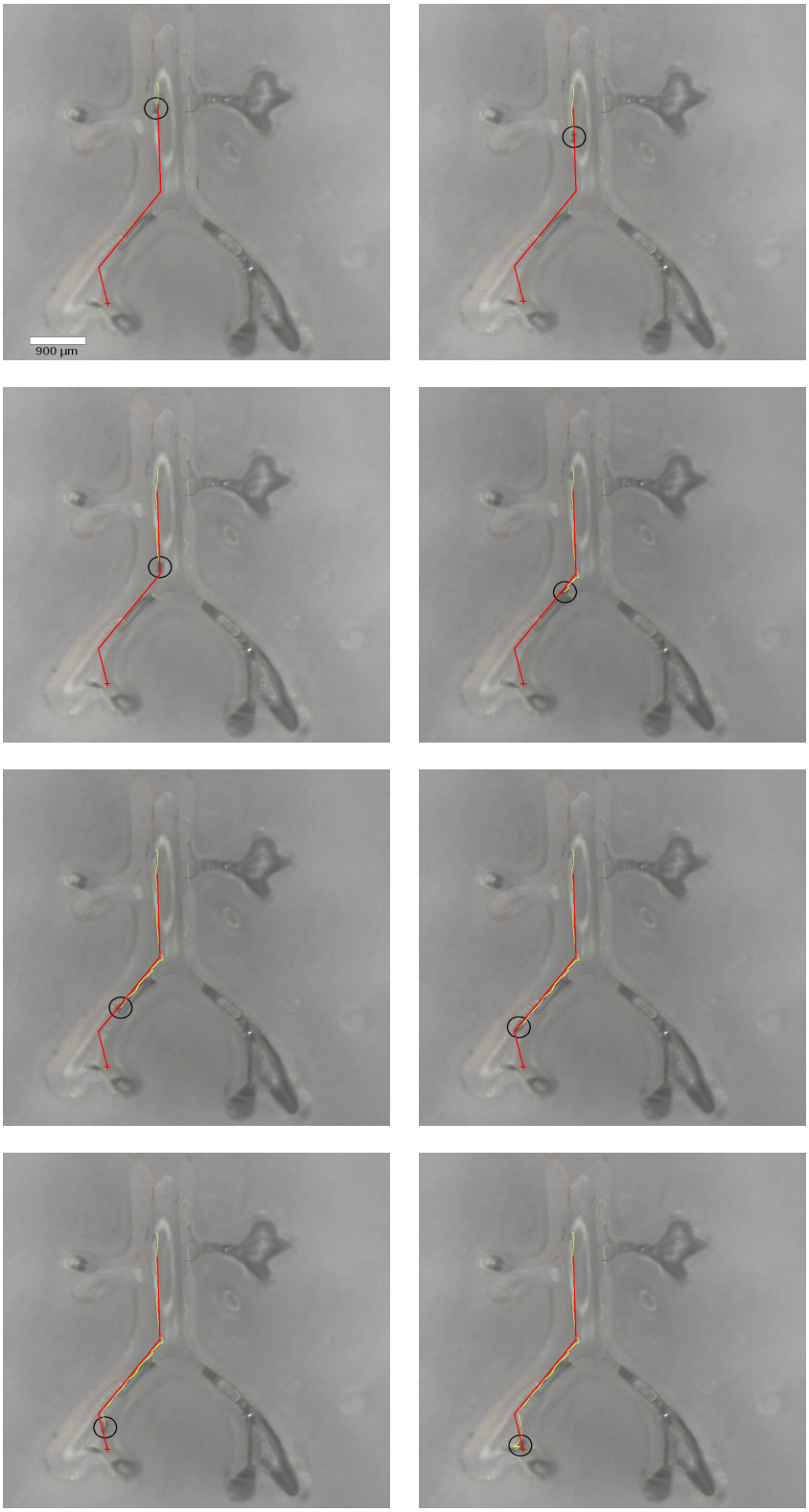



Fig. 5.7.: Controlling MicroParticle inside the Aorta channel using predefined path.

# 6. CONCLUSION AND FUTURE RECOMMENDATIONS

The aim of this thesis is to manipulate the MagnetoSperm and perform closed-loop motion control on it. Also, it is intriguing initially to ask how will the MagnetoSperm dynamics and propulsion force will be affected inside fluidic channels. We present in the thesis what is the effect of being near the surface for the MagnetoSperm velocity. Also, we investigate the frequency response with different precession angles of where is the motion direction of the MagnetoSperm concerning the Magnetic field of the coils. Moreover, we are inspired of how we can use the MagnetoSperm inside the human body for microsurgeries and drug delivery. Therefore, we perform experiments with a predefined path for the MagnetoSperm in a biological inspired fluidic channel same as the Aorta artery.

In conclusion, we perform successful point-to-point closed-loop control on the MagnetoSperm. We try to find the best fabrication conditions of the MagnetoSperm. We conclude that one of the most parameters that affect the MagnetoSperm dynamics and swimming is the ratio between the head and tail lengths. Through experiments, the best MagnetoSperm dynamics for swimming and making the tail more flexible were with ratio 1:6 until 1:8 between the head and tail lengths. Also, the effect of the fluidic channels is discussed, and there is a significant effect of swimming The MagnetoSperm in narrower channel widths. As we present in chapter 5 the difference between the MagnetoSperm velocity among all the four channel widths (1 mm, 0.75 mm, 0.5 mm, 0.25 mm). Also, we conclude that there is a great difference in the MagnetoSperm velocity between the first two channels (1 mm, 0.75 mm) and the last two (0.5 mm, 0.25 mm). In our experiments, we observe that in the very narrow channels the width is too small for the MagnetoSperm tail to able to complete his oscillation cycle. Therefore, the MagnetoSperm can not get enough tail oscillation, so it does not get enough propulsion force from his tail to able to swim as in the wider channels.

This led us to investigate more about the characterization of the MagnetoSperm inside the fluidic channels. Therefore, we show in chapter 5 fig. 5.2 how is the relation between the frequency response and the precession angle of the MagnetoSperm motion. We present in figures 5.5 and 5.6 affect the agent dynamics. Fortunately, the results of these experiments are so interesting as they show a lot of characterization of the MagnetoSperm. We prove that the fastest MagnetoSperm velocity is at 5 Hz oscillation frequency. The best frequency is 5 Hz since, at 10 Hz and higher frequencies, the amplitude of the MagnetoSperm oscillation decreases more than the that at 5 Hz. From fig. 5.5, we conclude that the MagnetoSperm velocity is gradually increased starting from 10 Hz till 25 Hz. The last conclusion is because the MagnetoSperm is swimming with gradient effect more than the oscillation effect, the higher oscillation frequency, the less oscillations effect and more gradient effect. Furthermore, in fig. 5.6, there is a fascinating point that the high frequency can compensate for the decrease in the MagnetoSperm velocity due to the small precession angles effect. Finally, we conclude that the best conditions for the MagnetoSperm motion is to have oscillation at 5 Hz and also the precession angle should be as wide as possible till 90°. Finally, we perform experiments on Microparticles and MagnetoSperms inside the biologically-inspired Aorta channel. We demonstrate our controller on the agents inside this channel to be closer to our vision of clinical and biological experiments inside the human body.

Moreover, we have many future recommendations for more improvements of this study on the motion control and characterization of the MagnetoSperm. One of these recommendations can be mainly based on studying more about the frequency response and precession angles effect in narrower fluidic channels to complete a full characterization that we start in this thesis. Also, motion control of the MagnetoSperm in 3D should be done to evaluate more about its ability for more complicated future tasks inside the human body. Furthermore, more experiments in more complicated channels should be done. The most important future work that we recommend is trying to have robust and optimal control on the MagnetoSperm against time-varying

flow inside fluidic channels. We try so much to do this. We faced many problems of inserting the MagnetoSperm with the flow inside the channel.

With the various research done over the past few years in studying the motion control and characterization of the MagnetoSperm, it seems that this MagnetoSperm is one of the most promising medical Micro-robots to have more tasks like drug delivery and micro surgeries inside the human body. All the research that was done proves the minimum width of the MagnetoSperm that can reach less than 15 $\mu$m. Therefore, the MagnetoSperm is the smallest Micro-robot that can go inside the human body for medical tasks. Furthermore, the MagnetoSperm dynamics of motion and its tail oscillation make it move in more viscous solutions and liquids than any other agent.

# 7. APPENDIX

## 7.1 MagnetoSperm controller $C++$ source code

```
#include "controllerMagnetoSperm.h"
#include <stdio.h>    /* printf */
#include <math.h>     /* atan2 */
#include "serialthread.h"
#include <chrono>
#define PI 3.14159265
//PID controller gains
#define kp  8
#define ki  0.000005
#define kd  3

                              /*Introduction about the code and the
                                 controller concept**/
        /*the code simply is divided into three functions:
        1-"Magnetosperm()": the constructor that intialize all the elements
           of the clas
        2-"updateValues()": the function that update all the elements
           values after each time we stop running the controller
        3-"controlMagnetoSperm(agentPosition, setpoint)": the function that
           contains all the controller concept*/

                                          /**Controller Concept**/
        /*May be the controller seems to you that is much complicated, but
           do not worry I will simply explain it here in small words
```

```
20          First this controller was done to control the MagnetoSperm inside
                the fluidic channels against flow, also it works in 2D
21          and it was done more general to fit easily for 3D after doing
                some modifications for more number of 3D coils.
22
23          The controller approach is using two electromagnets for directing
                the MagnetoSperm through been supplied with uniform magnetic
24          field and the other two electromagnets is for oscillating the
                Magnetosperm through been supplied with sinusoidal PID control
                signal.
25                      That is all about the controller concept and every
                            step will be explained in comments through the
                            code*/
26
27
28  MagnetoSperm::MagnetoSperm()
29
30    {
31       cout<<"New MagnetoSperm is created"<<endl;
32     //creating the text data files to be plotted using Matlab afterwards.
33       file.open ("XData.txt", std::ofstream::in | std::ofstream::out |
             std::ofstream::trunc);
34       file.close();
35
36       file.open ("YData.txt", std::ofstream::in | std::ofstream::out |
             std::ofstream::trunc);
37       file.close();
38
39     //reference point of the setpoint relative to agnet position
40       refx=0;
41       refy=0;
```

```
42
43        //previous agent position
44          prev_agentx=0;
45          prev_agenty=0;
46
47        //previous setpoint
48          prev_setx=0;
49          prev_sety=0;
50
51        //current and previous theta
52          theta=0;
53          prev_theta=0;
54
55        //precision angle of the Torque
56          beta=0;
57
58        //errors
59          x_error=0;
60          y_error=0;
61
62        //previous errors
63          prev_x_error_1=0;
64          prev_y_error_1=0;
65          prev_x_error_2=0;
66          prev_y_error_2=0;
67
68        //error average
69          x_error_Integration=0;
70          y_error_Integration=0;
71
72        //error difference
```

```
73          x_error_difference=0;
74          y_error_difference=0;
75
76        //updating the elapsed time for 1000 milliseconds
77          elapsed_time=1000;
78          prev_elapsed_time=0;
79
80        //displacement, velocity and change in distance covered by the robot
81          displacement=0;
82          distance=0;
83          velocity=0;
84
85
86        //time difference
87          dt=1000;
88
89        //integral errors
90          x_error_integral=0;
91          y_error_integral=0;
92
93        //previous integral errors
94          x_error_integral_prev=0;
95          y_error_integral_prev=0;
96
97        //derivative error
98          x_error_dervative=0;
99          y_error_dervative=0;
100
101       //control signals
102         x_control_signal=0;
103         y_control_signal=0;
```

```
104
105       //uniform currents magnitude
106         I[0]=0;
107         I[1]=0;
108         I[2]=0;
109         I[3]=0;
110
111       //Coils uniform or oscillating magnetic fields conditions
112       // 1:means this coil has oscillating magnetic field 0:means this coil
              has uniform magnetic field
113         m[0]=0;
114         m[1]=0;
115         m[2]=0;
116         m[3]=0;
117
118         //the turn of the upcoming oscillation angle
119           turn=0;
120
121       //FIR(Finite Impulse Response) Filter-->Running average filter
              parameters
122         agentTracker << 0,0;
123
124       //flag: reaching the target or not to stop the uniform magnetic field
              coils
125         reached=false;
126
127       //tracking
128         count=0;
129
130       //flag for running the controler or not
131         run=false;
```

```
132     }
133
134   void MagnetoSperm::updateValues()
135
136     {
137      //t is representing the time when plotting the data
138        t=0;
139
140      //reference point of the setpoint relative to agnet position
141        refx=0;
142        refy=0;
143
144      //previous agent position
145        prev_agentx=0;
146        prev_agenty=0;
147
148      //previous setpoint
149        prev_setx=0;
150        prev_sety=0;
151
152      //reset the errors and time to zero.
153        x_error=0;
154        y_error=0;
155
156      //updating the elapsed time for 1000 milliseconds
157        elapsed_time=1000;
158        prev_elapsed_time=0;
159
160      //displacement, velocity, and change in distance covered by the robot
161        distance=0;
162        displacement=0;
```

```
163         velocity=0;
164
165       //previous errors
166         prev_x_error_1=0;
167         prev_y_error_1=0;
168         prev_x_error_2=0;
169         prev_y_error_2=0;
170
171       //error average
172         x_error_Integration=0;
173         y_error_Integration=0;
174
175       //error difference
176         x_error_difference=0;
177         y_error_difference=0;
178
179       //time difference
180         dt=1000;
181
182       //integral errors
183         x_error_integral=0;
184         y_error_integral=0;
185
186       //previous integral errors
187         x_error_integral_prev=0;
188         y_error_integral_prev=0;
189
190       //derivative error
191         x_error_dervative=0;
192         y_error_dervative=0;
193
```

```
194        //control signals
195          x_control_signal=0;
196          y_control_signal=0;
197
198        //uniform currents magnitude
199          I[0]=0;
200          I[1]=0;
201          I[2]=0;
202          I[3]=0;
203
204        //Coils uniform or oscillating magnetic fields conditions
205          m[0]=0;
206          m[1]=0;
207          m[2]=0;
208          m[3]=0;
209
210        //the turn of the upcoming oscillation angle
211          turn=0;
212
213        //FIR(Finite Impulse Response) Filter-->Running average filter
               parameters
214          agentTracker<<0,0;
215
216        //flag: reaching the target or not to stop the uniform magnetic field
               coils
217          reached=false;
218
219        //tracking
220          count=0;
221
222        //flag for running the controler or not
```

```
223         run=false;
224
225         cout<<"The MagSperm was updated successfully"<<endl;
226       }
227
228     Eigen::Vector4d MagnetoSperm::controlMagnetoSperm(Eigen::Vector2d
          agentPosition, Eigen::Vector2d setpoint, double prec_angle)
229
230       {
231
232        //vector with 4 components
233         Eigen::Vector4d currents;
234        //currents[0] -> first elements
235         std::cout << "controller is running" << std::endl;
236
237         if(run==false){
238             filename="xxx";
239        file.open(filename, std::ofstream::in| std::ofstream::out |
             std::ofstream::trunc);
240        file <<5<<endl;
241        file.close();
242         run=true;
243         }
244
245        //x-> agentPosition[0]
246        //y-> agentPosition[1]
247        //x-> setpoint[0]
248        //y-> setPoint[1]m
249
250         double setx=setpoint[0];
251         double sety=setpoint[1];
```

```
252         double agentx=agentPosition[0];
253         double agenty=agentPosition[1];
254         cout<<"setx"<<setx<<"sety"<<sety<<endl;
255
256       //FIR(Finite Impulse Response) Filter-->Running average filter
257         agentTracker=agentPosition;
258         poseFilter.filt_point(agentTracker,agentTracker);
259
260       //intializing the reference point according to the agent position to
            get the theta between them after that.
261         refx=  setx-agentx;
262         refy= -(sety-agenty); // there is a negative sign in the y to make
              it like normal coordinates.
263
264       //Getting the theta between the agent position and the setpoint.->
            theta is the angle of the direction we want to control the
            magnetosperm in.
265         theta= (atan2(refy,refx)) * 180 / PI;
266         if(theta < 0)
267             theta=theta+360;
268         cout<<"theta "<<theta<<endl;
269       //getting proportional error of the position of the agent and it is
            also a function in the theta of oscillation
270       //to shift our set point by the oscillation angle(thetaosc), we have
            to multiply the radius of a circle by the angle we need to shift
            by.
271       //the radius of the circle(its center the agent position and the
            setpoint is the end point of its radius) is equal R= x^2+y^2
272         double R=sqrt((pow(refx,2.0))+(pow(refy,2.0)));
273
```

```
274        //this is to know the turn of which oscillating angle since theta is
               divided into 2 oscillating angles (right and left)
275          double thetaosc;
276          turn=1-turn;
277          cout<<"turn  "<<turn<<endl;
278          if(turn==1)
279           {
280              thetaosc=theta-45;
281              cout<<"tik"<<endl;
282            //getting proportional error of the position of the agent and it
                   is also a function in the theta of oscillation
283            //to shift our set point by the oscillation angle(thetaosc), we
                   have to multiply the radius of a circle by the angle we need
                   to shift by.
284              x_error=R*cos(thetaosc*PI/180);
285              y_error=R*sin(thetaosc*PI/180);
286
287            //error Difference (is it increasing or decreasing).
288              x_error_difference= (prev_x_error_1-x_error);
289              y_error_difference= (prev_y_error_1-y_error);
290            //updating the error values.
291              prev_x_error_1=x_error;
292              prev_y_error_1=y_error;
293            //error average
294              x_error_Integration=((x_error+prev_x_error_1)/2);
295              y_error_Integration=((y_error+prev_y_error_1)/2);
296           }
297          else
298           {
299              thetaosc=theta+45;
300              cout<<"tok"<<endl;
```

```
301           //getting proportional error of the position of the agent and it
                  is also a function in the theta of oscillation
302           //to shift our set point by the oscillation angle(thetaosc), we
                  have to multiply the radius of a circle by the angle we need
                  to shift by.
303             x_error=R*cos(thetaosc*PI/180);
304             y_error=R*sin(thetaosc*PI/180);
305           //error Difference
306             x_error_difference= (prev_x_error_2-x_error);
307             y_error_difference= (prev_y_error_2-y_error);
308           //updating the error values.
309             prev_x_error_2=x_error;
310             prev_y_error_2=y_error;
311           //error average
312             x_error_Integration=((x_error+prev_x_error_2)/2);
313             y_error_Integration=((y_error+prev_y_error_2)/2);
314         }
315
316        //time difference
317         dt=elapsed_time;
318
319       //   double time = elapsed_time+prev_elapsed_time;
320         cout<<"dt"<<dt<<endl;
321        //  prev_elapsed_time=time;
322        //Integral error
323         x_error_integral=(x_error_Integration*dt)+x_error_integral_prev;
324         y_error_integral=(y_error_Integration*dt)+y_error_integral_prev;
325
326        //Derivative error = -dpv/dt ***the negative sign is because the
             dervative term has an opposite impact on the control signal
327         x_error_dervative=-x_error_difference/dt;
```

```
328           y_error_dervative=-y_error_difference/dt;
329
330         //calculating the control signal for the three errors in the PID
              control
331           x_control_signal=abs(kp*x_error+ki*x_error_integral+kd*x_error_dervative);
332           y_control_signal=abs(kp*y_error+ki*y_error_integral+kd*y_error_dervative);
333
334           if(x_control_signal>1000)
335            {
336             x_control_signal=1000;
337            }
338
339           if(y_control_signal>1000)
340            {
341             y_control_signal=1000;
342            }
343
344         //Precision Angle of the Torque
345           if(theta<=90)
346             beta=theta;
347           else if(theta<=180)
348               beta=theta-90;
349           else if(theta<=270)
350               beta=theta-180;
351           else
352               beta=theta-270;
353
354         //magnitude of uniform magnetic fields
355           cout<<theta<<endl;
356
357           if(theta<=90 || theta>=270)
```

```
358          {
359           I[0]=cos(theta*PI/180);
360           I[2]=abs(sin(theta*PI/180));
361          }
362         else
363          {
364           I[0]=abs(sin(theta*PI/180));
365           I[2]=abs(cos(theta*PI/180));
366          }
367
368         if(theta<=180 && theta>=0)
369          {
370           I[1]=sin(theta*PI/180);
371           I[3]=abs(cos(theta*PI/180));
372          }
373         else
374          {
375           I[1]=abs(cos(theta*PI/180));
376           I[3]=abs(sin(theta*PI/180));
377          }
378       //setting the uniform and oscillating magnetic field conditions for
            each coil to determine if it is uniform or oscillating.
379       // 0:uniform magnetic field 1:oscillating magnetic field
380         if((theta<=80 || theta>=280)) //80->60should be changed in the full
              closed loop control 280 -> 300
381          {
382           m[0]=0;
383          }
384         else if(theta<=190 && theta>=170)//170>150 190>210
385          {
386           m[0]=0;
```

```
387             I[0]=0;
388            }
389           else
390            {
391             m[0]=1;
392            }
393           if((theta<=170 && theta>=10)) //10->30should be changed in the full
                 closed loop control 170>150
394            {
395             m[1]=0;
396            }
397           else if(theta<=280 && theta>=260) //240>260 280>300
398            {
399             m[1]=0;
400             I[1]=0;
401            }
402           else
403            {
404             m[1]=1;
405            }
406           if((theta<=260 && theta>=100)) //260>240 100>120
407            {
408             m[2]=0;
409            }
410           else if(theta<=10 || theta>=350)//330> 10->30should be changed in
                 the full closed loop control
411            {
412             m[2]=0;
413             I[2]=0;
414            }
415           else
```

```
416          {
417           m[2]=1;
418          }
419         if((theta<=350 && theta>=190))//210>190 330>350
420          {
421           m[3]=0;
422          }
423         else if(theta<=100 && theta>=80)//120>100 80->60should be changed
               in the full closed loop control
424          {
425           m[3]=0;
426           I[3]=0;
427          }
428         else
429          {
430           m[3]=1;
431          }
432
433         double turn_coil[4]; //an array for the oscillation turn of the 4
               coils.
434         bool first=false;  //first: a flag to know which oscillating coil
               will generate magnetic field first.
435         double max_uni_current=3;//maximum current for uniform magnetic
               field
436         double max_osc_current=1.5;//maximum current for oscillation
               magnetic field
437
438         if(abs(refx)<=10&&abs(refy)<=10)
439          {
440           reached=true;
441          }
```

```
442          else
443           {
444            reached=false;
445           }
446
447          if(reached==true&&(prev_setx==setx)&&(prev_sety==sety))
448           {
449            max_uni_current=0;
450            max_osc_current=0;
451            count=count+1;
452           }
453
454         for(int i=0;i<4;++i) //i: not more than a counter for the while loop
               to be a pointer for the coils.
455           {
456            if(m[i]==1)
457             {
458              if(first==false)
459               {
460                turn_coil[i]=turn;
461                first=true;
462               }
463              else
464               {
465                turn_coil[i]=1-turn;
466               }
467
468              switch(i)
469               {
470                //setting the oscillating magnetioc field currents for the 4
                      coils.
```

```
471                 //the currents for the X coils: coils 0 & 2.
472                 //the currents for the y coils: coils 1 & 3.
473
474                 //the "turn" terms are the oscillating factors to determine
                        whose coil's turn to
475                 //generate magnetic field and the other one will be zero and
                        vice versa.
476                   case(0):currents[0]=(x_control_signal/1000)*max_osc_current*turn_coil[0];b
477                   case(1):currents[1]=(y_control_signal/1000)*max_osc_current*turn_coil[1];b
478                   case(2):currents[2]=(x_control_signal/1000)*max_osc_current*turn_coil[2];b
479                   case(3):currents[3]=(y_control_signal/1000)*max_osc_current*turn_coil[3];b
480
481 //                case(0):currents[0]=0;break;
482 //                case(1):currents[1]=0;break;
483 //                case(2):currents[2]=0;break;
484 //                case(3):currents[3]=0;break;
485                   }
486               }
487              else
488               {
489                switch(i)
490                 {
491                  //setting the uniform magnetic field currents for the 4 coils.
492
493                  //the currents for the X coils: coils 0 & 2.
494                  //the currents for the y coils: coils 1 & 3.
495                    case(0): currents[0]=I[0]*max_uni_current;break;
496                    case(1): currents[1]=I[1]*max_uni_current;break;
497                    case(2): currents[2]=I[2]*max_uni_current;break;
498                    case(3): currents[3]=I[3]*max_uni_current;break;
499
```

```
500 //             case(0):currents[0]=0;break;
501 //              case(1):currents[1]=0;break;
502 //              case(2):currents[2]=0;break;
503 //              case(3):currents[3]=0;break;
504              }
505             }
506           }
507         cout<<"currents "<<currents[0]<<" "<<currents[1]<<"
              "<<currents[2]<<" "<<currents[3]<<endl;
508         cout<<"Turns "<<turn_coil[0]<<" "<<turn_coil[1]<<"
              "<<turn_coil[2]<<" "<<turn_coil[3]<<endl;
509         cout<<turn<<endl;
510
511       //saving the values of the errors
512         x_error_integral_prev=x_error_integral;
513         y_error_integral_prev=y_error_integral;
514
515       //saving the values of the setpoint position
516         prev_setx=setx;
517         prev_sety=sety;
518
519       //updating the agent position
520         prev_agentx=(double)agentTracker[0];
521         prev_agenty=(double)agentTracker[1];
522
523       //t is representing the time when plotting the data
524         t++;
525
526       //writing in text file to be plot the data using Matlab
527       //You just import these text files in Matlab and plot it
```

```
528         file.open ("XData.txt", std::fstream::in | std::fstream::out |
                std::fstream::app);
529         file <<x_error<<"       "<<x_error_integral<<"
                "<<x_error_dervative<<"       "<<x_control_signal<<"     "<<t<<"
                      "<<x_control_signal*3/1000<<"
                "<<x_error_difference<<"      "<<theta<< endl ;
530         file.close();
531         file.open ("YData.txt", std::fstream::in | std::fstream::out |
                std::fstream::app);
532         file <<y_error<<"          "<<y_error_integral<<"
                "<<y_error_dervative<<"      "<<y_control_signal<<"      "<<t<<"
                      "<<y_control_signal*3/1000<<"
                "<<y_error_difference<<"      "<<theta<< endl ;
533         file.close();
534
535         file.open(filename, std::fstream::in| std::fstream::out |
                std::fstream::app);
536         file <<agentx<<"   "<<agenty<<"   "<<dt<<"
                "<<(double)agentTracker[0]<<"   "<<(double)agentTracker[1]<<"
                "<<prec_angle<<"   "<<theta<<"   "<<currents[0]<<"
                "<<currents[1]<<"   "<<currents[2]<<"   "<<currents[3]<<endl;
537         file.close();
538
539         cout<< "distance"<<R<<endl;
540         return currents;
541     }
```

## 7.2 MagnetoSperm controller $C++$ header code

```
#pragma once

#include "Eigen/Core"
#include <iostream>
#include <stdio.h>    /* printf */
#include <math.h>     /* atan2 */
#include <chrono>
#include <fstream>
#include <iostream>
#include <string>
#include "PointFilter.h"

#define PI 3.14159265
using namespace std;

  class MagnetoSperm
    {
      PointFilter poseFilter;
    public:
     MagnetoSperm();//The Constructor.

    //external class members
      std::fstream file;

    //text file name
      string filename;

    //the turn of the upcoming oscillation angle
      int turn;
```

```
30
31      //total time
32        int t;
33
34      //reference point of the setpoint relative to agnet position
35        double refx;
36        double refy;
37
38      //execution time
39        double elapsed_time;
40        double prev_elapsed_time;
41
42      //gains
43        double kp;
44        double ki;
45        double kd;
46
47      //previous agent position
48        double prev_agentx;
49        double prev_agenty;
50
51      //previous setpoint
52        double prev_setx;
53        double prev_sety;
54
55      //current and previous theta
56        double theta;
57        double prev_theta;
58
59      //precision angle of the Torque
60        double beta;
```

```
61
62      //errors
63        double x_error;
64        double y_error;
65
66      //previous errors
67        double prev_x_error_1;
68        double prev_x_error_2;
69        double prev_y_error_1;
70        double prev_y_error_2;
71
72      //error average
73        double x_error_Integration;
74        double y_error_Integration;
75
76      //error difference
77        double x_error_difference;
78        double y_error_difference;
79
80      //time difference
81        double dt;
82
83      //integral errors
84        double x_error_integral;
85        double y_error_integral;
86
87      //previous integral errors
88        double x_error_integral_prev;
89        double y_error_integral_prev;
90
91      //derivative error
```

```
92        double x_error_dervative;
93        double y_error_dervative;
94
95      //control signals
96        double x_control_signal;
97        double y_control_signal;
98
99      //displacement, velocity and change in distance covered by the robot
100       double distance;
101       double displacement;
102       double velocity;
103
104     //uniform currents magnitude
105       double I[4];
106
107     //Coils uniform or oscillating magnetic fields conditions
108     // 1:means this coil has oscillating magnetic field 0:means this coil
          has uniform magnetic field
109       double m[4];
110
111     //flag: reaching the target or not to stop the uniform magnetic field
          coils
112       bool reached;
113
114     //tracking
115      int count; //the counter for setpoints of the closed trajectories
116
117      //flag for running the controler or not
118        bool run;
119
```

```
120      //FIR(Finite Impulse Response) Filter-->Running average filter
             parameters
121       Eigen::Vector2d agentTracker;
122
123     //The main method of the controller.
124       Eigen::Vector4d controlMagnetoSperm(Eigen::Vector2d agentPosition,
              Eigen::Vector2d setpoint,double prec_angle);
125
126     //Method to update the values of the class members after turning off the
           controller.
127       void updateValues();
128     };
```

## 7.3 The tracking $C++$ code for closed trajectories and predefined path

```
1                //changing the precision angle
2                  prec_angle=0;
3                  if(prec_angle>0&&MagRobot.run==false)
4                   {
5                      Setpoint.x=centerTrackerMaster.x+(350*cos(prec_angle*PI/180));
6                      Setpoint.y=centerTrackerMaster.y-(350*sin(prec_angle*PI/180));
7                   }
8
9                // Draw the setpoint on the image
10                  drawSetPoint(Setpoint,frameOut);
11               // Mat imgLines = Mat::zeros(frameTmp.size(),CV_8UC3);
12
13                  if(mRunController && MagRobot.agentTracker[0]>0 &&
                        MagRobot.agentTracker[1]>0)
14                  {
15
16               // Draw moving agent trajectory
17                  cv::Scalar traj_color(70,255,255); //green
18                  //trajMaster.push_back(centerTrackerMaster);
19                  trajMaster.push_back(cv::Point2f(MagRobot.agentTracker[0],MagRobot.agentT
20                  for(i=1; i<trajMaster.size();++i)
21                  {
22                  line(frameOut,trajMaster[i],trajMaster[i-1],traj_color,1);
23                  }
24                  }
25
26               // Draw closed Targeted trajectory
27                  cv::Scalar Setpoint_color(0,0,255); //red
```

```
28                  auto contains=
                        std::find(SetpointMaster.begin(),SetpointMaster.end(),Setpoint);
29
30                  if(abs(Setpoint.x)>0 && abs(Setpoint.y)>0 && contains ==
                        SetpointMaster.end()) //to check that the new setpoint
                        is not at the origin of the screen and was not added
                        to the vector before
31                  {
32                  SetpointMaster.push_back(Setpoint);
33                   }
34
35                  for(i=1; i<SetpointMaster.size();++i)
36                  {
37                  line(frameOut,SetpointMaster[i],SetpointMaster[i-1],Setpoint_color,2);
38
39                  d=sqrt((pow((SetpointMaster[i].x-SetpointMaster[i-1].x),2.0))+(pow(Setpoi
                        //distance between each two main setpoints
40
41                 v = SetpointMaster[i]-SetpointMaster[i-1]; //vector of the
                       line between the two main setpoints.
42
43                  double normalized=sqrt((pow(v.x,2.0))+(pow(v.y,2.0)));
44                 u.x= (v.x)/normalized;
45                 u.y=(v.y)/normalized; //normalization for vector v
46
47                  if(SetpointMaster.size()>1 &&
                        setpoints.size()==4*(SetpointMaster.size()-2))
                        //setpoints vector is vector for the secondary
                        setpoints along the predefined path
48                  {
49                  for(k=0; k<4;++k)
```

```
50                  {
51                  double factor=(k+1)/(double)5;
52                      waypoint=SetpointMaster[i-1]+(factor*v);
53                      setpoints.push_back(waypoint);
54                      cout<<"waypoint "<<factor<<endl;
55                  }
56                  }
57                  }
58  //                  cout<<"size  "<<SetpointMaster.size()<<endl;
59  //                  cout<<"size2  "<<setpoints.size()<<endl;
60
61                  if(SetpointMaster.size()==1&&setpoints.size()==0&&total_setpoints.size()=
62                   {   total_setpoints.push_back(SetpointMaster[0]);
63                      cout<<"here total setpoints"<<endl;
64                         }
65                  if(total_setpoints.size()<=setpoints.size()+SetpointMaster.size()-5
                        && SetpointMaster.size()>0)
66                  {
67                      total_setpoints.resize(setpoints.size()+SetpointMaster.size());
68
69              for(i=0;i<SetpointMaster.size()-1;++i)
70                  {
71                      cout<< "i "<<i<<endl;
72                      for(k=(i*5)+1;k<(i*5)+5 ;++k)
73                      {
74                        cout<< "k "<<k<<endl;
75                        total_setpoints[k]= setpoints[k-i-1];
76                      }
77                      total_setpoints[(5*i)+5]= SetpointMaster[i+1];
78
79                      cout<<"final vector "<<total_setpoints<<endl;
```

```
80                      cout<<"secondary setpoints "<<setpoints<<endl;
81                      cout<<"Main setpoints  "<<SetpointMaster<<endl;
82                  }
83                  }
```

# LIST OF REFERENCES

# LIST OF REFERENCES


[1] I. S. M. Khalil, H. C. Dijkslag, L. Abelmann, and S. Misra, "Magnetosperm: A microrobot that navigates using weak magnetic fields," *Applied Physics Letters*, pp. 223701–1–223701–4, 2014.

[2] I. S.M.Khalil, K.Youakim, A.Sánchez, and S. Misra, "Magnetic-based motion control of sperm-shaped microrobots using weak oscillating magnetic fields," *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS), Chicago, USA*, pp. 4686–4691, 2014.

[3] E. Lauga, W. R. Diluzio, G. M.Whitesides, and H. A.Stone, "Swimming in circles: Motion of bacteria near solid boundaries," *Biophysical Journal*, vol. 90, pp. 400–412, 2006.

[4] T. H. Boyer, "The force on a magnetic dipole," *American Journal of Physics*, vol. 56, pp. 688–692, August.

[5] S. S. Shevkoplyas, A. C. Siegel, R. M. Westervelt, M. G. Prentiss, and G. M. Whitesides, "The force on a magnetic dipole," *Lab on a Chip*, vol. 7, pp. 1294–1302, July 2007.

[6] T. J. Ui, R. G. Hussey, and R. P. Roger, "Strokes drag on a cylinder in axial motion," *Physics of Fluids*, vol. 27, pp. 787–795, April 1984.

[7] A. Sanchez, Y. M. Sadek, I. S. Khalil, and S. Misra, "Mobi-mag: A compact device for medical research using wireless control of magnetic microrobots," 2014.

[8] Y. M. Sadek, "Autonomous manipulation with full pose control," *Bachelor Thesis, German University in Cairo*, September 2014.

[9] I.S.M.Khalil, L. M.P.Pichel, L.Yondervan, and S. Misra, "Characterization and control of biological microrobots," *13th International Symposium on Experimental Robotics-Springer Tracts in Advanced Robotics,Quebec City, Canada*, June 2012.

[10] A. Denasi and S. Misra, "A robust controller for micro-sized agents: The prescribed performance approach," *Proceedings of the Annual International Conference on Manipulation, Automation, and Robotics at Small Scales (MARSS)*, July 2016.

[11] K.B.Yesin, K.Vollmers, and B.J.Nelson, "Modeling and control of untethered biomicrorobotics in a fluidic enviroment using electromagnetic fields," *International Journal of Robotics Research*, vol. 25, pp. 527–536, May 2006.

[12] B.J.Nelson, I.K.Kaliakatsos, and J.J.Abbott, "Microrobots for minimally invasive medicine," *The Annual Review of Biomedical Engineering*, vol. 12, April 2010.

[13] Y. Kawakami, Y. Eguchi, T. Nimiya, H. Suemitsu, and T. Matsuo, "Velocity and acceleration estimation by iterative learning observer and performance validation with mems-based inertial sensors," vol. 5, pp. 113–121, 01 2013.