UNIVERSITÉ DE MONTRÉAL

DESIGN AND IMPLEMENTATION OF A FUZZY CONTROLLER FOR STEERING MICROPARTICLES INSIDE BLOOD VESSELS BY USING A MRI SYSTEM

KE PENG
DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
DÉCEMBRE 2011

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

DESIGN AND IMPLEMENTATION OF A FUZZY CONTROLLER FOR STEERING MICROPARTICLES INSIDE BLOOD VESSELS BY USING A MRI SYSTEM

présenté par : PENG, Ke
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

M. LANGLOIS, J.M. Pierre, Ph. D., président
M. MARTEL, Sylvain, Ph. D., membre et directeur de recherche
M. GOURDEAU, Richard, Ph. D., membre

有三种单纯而强烈的感情支配着我的一生：

对爱情无可抑制的渴望，

对知识永不停止的追求，

以及对人类苦难痛彻心扉的怜悯。

——伯特兰·罗素

ACKNOWLEDGMENT

First of all, I would like to express my great thanks particularly to my research supervisor, Mr. Sylvain Martel, for providing me with the opportunity to study and do research at the NanoRobotics Laboratory, and for guiding me into the interesting and charming Magnetic Resonance Submarine (MR-Sub) project. He instructs his students with great patience and all his enthusiasm. Under his direction, the students are able to not only acquire knowledge and have their skills trained, but also find the art and beauty in researching. He respects all his students' ideas, and is always open to any discussion or talk so as to encourage their independence and creativity. He concerns his students with all his heart for all kinds of difficulties and problems encountered in their projects, and in their daily lives as well. I would like to say that it is my great honour to have him as my director for my master`s program. And Mr. Sylvain Martel is a perfect example for me in almost every aspect.

I would like to thank Mr. Charles Tremblay, who gave hand-to-hand instructions to me for obtaining necessary skills in using all the facilities at the laboratory and at the Magnetic Resonance Imaging (MRI) room. He prepared all the materials and tools that I might need in my experiments. Thanks, Charles!

To Madam Neïla Kaou, who helped me in dealing with all the paper work related to the school, and in arranging my schedule and plans for my project and for my master`s program.

To Gaël Bringout, who led me to the MR-Sub project. He suggested useful reading materials for me to let me to get familiar with the whole project and setup. He is warm-hearted, kind and patient, and is ready for any question all the time, even the simplest one. I thank him sincerely for his great patience.

To Behnam Izadi, who did excellent work in the tracking program to take pictures from a camera, to locate and track the beads, and to get theirs' coordinates from the pictures. He provided bright "eyes" for my controller.

To Guillermo Vidal, who was always with me in setting up all the hardware for my experiment, the power supply, the pump, etc. He is willing to provide some help all the time.

To Viviane Lalande, who made nice and clean phantoms and beads for our experiments. Even if she is not a member of our team, she tried her best to help me and solved almost all the mechanical machining problems for me.

To Manuel Vonthron, who is head of the computer team. He kept an eye on my project and gave really useful suggestions when I had problems. He corrected the format of my paper carefully and patiently when I had no experience for addressing a paper to a conference. And he is also an expert in programming MR sequences.

To Frederick Gosselin, who gave me a two-hour introduction in fluid physics when I had problems in modelling for the friction force inside the blood vessel. His words helped me in constructing a complete mathematical model for the microparticles under control.

To Benjamin Conan and Alexandre Bigot, who spent their time every Friday in listening to my presentations and in discussing my project in case of need.

To all the former members and current members of the NanoRobotics laboratory, École Polytechnique de Montréal. I have to say that I have spent two and a half years full of happiness and joy in the lab. As a foreign student who is not so fluent in French, I received respect from all of them. English always has the priority to be spoken and to be written at every meeting, as long as I am present, even if it makes themselves not so comfortable sometimes. During my whole life, I will never forget those times working with every one of them.

To my parents, Mr. Weiping Peng and Madam Qin Zhang. They showed their love to me, understood me, supported me all the time, and gave me confidence when I was passing hard times.

To my little sisters Ting Zheng and Yanji Zhang. May they be happy forever.

ABSTRACT

In this thesis, a Single-Input-Multiple-Output (SIMO) fuzzy controller is designed to drive an upgraded clinical real-time Magnetic Resonance Imaging (MRI) machine to provide steering forces for a single microparticle and an aggregation of ferromagnetic microparticles in the human cardiovascular system according to a pre-defined pathway. Based on a fluid dynamic mathematical model, the validity of this kind of controller has firstly been tested by preliminary 2-Dimensional (2-D) simulation results with MATLAB/C++ hybrid programming. With both the beads and real microparticles, real-time experiments were also performed with simulated Magnetic Resonance (MR) sequences and 2-D pulsatile flow. Related experimental data also illustrates that, despite some limitations, this kind of fuzzy controller has the potential to be the appropriate controller for Magnetic Resonance Navigation (MRN).

RÉSUMÉ

Le présent mémoire porte sur l'étude de la conception et la réalisation d'un contrôleur flou avec une seule entrée et multiples sorties. Une telle étude vise à pouvoir contrôler un appareil clinique d'Imagerie par résonance magnétique (IRM) pour fournir des forces de pilotage dans le but de naviguer une microparticule ferromagnétique ou une agrégation de ces microparticules le long d'une trajectoire prédéfinis à l'intérieur du système cardio-vasculaire humaine. L'algorithme de ce contrôleur a été proposé sur un modèle mathématique du fluide dynamique, et sa validité a été vérifiée par les résultats préliminaires de simulations en 2-D générés avec les logiciels MATLAB et C++. À l'aide d'un IRM clinique, des expériences de navigation en temps réel sur des petites perles ainsi que des microparticules ont également été réalisées dans un flux pulsatile. Connexes données expérimentales peuvent prouver que, malgré certaines limites, ce type de contrôleur flou a le potentiel pour devenir le contrôleur approprié appliqué à la navigation par résonance magnétique (NRM).

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

2-D	Two-Dimension
3-D	Three-Dimension
$\overrightarrow{\nabla B}$	Induced magnetic gradient (T/m)
AUV	Autonomous Underwater Vehicle
D	Blood vessel diameter(m)
E	Fuzzy input: difference of x-coordinates between current position and next waypoint
EC	Fuzzy input: velocity in x-axis
$\vec{F}_{mag}$	magnetic force (N)
G	Fuzzy output: magnetic gradient value
$\overrightarrow{M}$	Microparticle magnetization (A/m)
MIMO	Multiple-Input-Multiple-Output
MR	Magnetic Resonance
MRI	Magnetic Resonance Imaging
MRN	Magnetic Resonance Navigation
MR-Sub	Magnetic Resonance - Submarine
PID	Proportional-Integral-Derivative
PMMA	Poly(methyl methacrylate)
R_{e_p}	Reynolds number in x-axis
R_{e_v}	Reynolds number in y-axis
r	Radius of a microparticle (m)
SIMO	Single-Input-Multiple-Output
T	Fuzzy output: MR sequence maintaining time adjustment
U	Average blood flow velocity(m/s)
V_f	volume of the ferromagnetic entity (m^3)

V_y	Steering velocity of particles (m/s)
ρ	Blood flow density(kg/m^3)
ρ_b	Bead density(kg/m^3)
μ	Dynamic viscosity of blood fluid
ν	Kinematic viscosity of blood fluid

LIST OF ANNEXES

INTRODUCTION

Nowadays, in the biology and medical society, using microparticles as specialized drug carriers in the human cardiovascular system is considered to be a promising approach against diseases such as some particular types of cancers. By operating minimally invasive interventions, this kind of method is capable of significantly reducing the risk of bacterial infection and shortening the recovering time for the patient after the surgery.

In this method, the micro carriers are injected at a certain point of human body, and are supposed to be propelled and navigated to travel along human vascular system from the injection point to the tumour position. Proper catheters and endoscopes are firstly used to deliver those micro carriers. However, catheters tend to have limitations in providing a pathway for micro carriers to pass through various kinds of blood vessels which could be as thick as the aorta or as thin as the capillaries, due that the manufacturing process for catheters requires a minimum diameter and special cross section shapes. Hence, there is a specific area inside human body that the micro carriers could not reach, if only the catheters are used for the drug delivery.

In our Magnetic Resonance Submarine (MR-Sub) project, ferromagnetic microparticles are employed as robot carriers. And we plan to apply external magnetic field to take over the navigation for the microparticles, as soon as they are released from the end of a catheter at a point of blood vessels which is inaccessible for the catheter.

Previous work has already proven that an upgraded clinical Magnetic Resonance Imaging (MRI) platform is capable of providing micro devices with adequate magnetic fields and gradients for endovascular propulsion [1, 2, 3] with programmable Magnetic Resonance (MR) sequences. At first, a MR full scan for the human body ensures to discover a proper pathway for microparticles from the current point directly to the tumour with several waypoints indicated. Being equipped with a real-time tracking unit, the positioning unit embedded inside the system is able to feed back the coordinates of tiny microparticles from a three-dimensional (3-D) MR image, which would then allow us to perform a closed-loop control [4]. Finally, a MR sequence is generated and then applied according to the output of the control algorithm, until the next tracking-controlling period.

The major limitation of this MR application is that such a MRN technique is not readily applicable in smaller diameter vessels such as arterioles and capillaries. The spatial resolution of

clinical MR imaging to gather path information by imaging the blood vessel is possible for arteries but not possible at the present time for smaller diameter vessels including arterioles and capillaries [5].

This thesis describes a new control algorithm to navigate microparticles to overcome the shortcomings of traditional control methods. A Single-Input-Multiple-Output (SIMO) classical fuzzy controller was chosen for its simplicity, fast response time and independence from complex, time-varying environment parameters.

This thesis is organised as follows: the literature review discusses previous studies of using a MRI scanner as a propelling machine for microparticles. A mathematic model based on dynamic fluid physics is then established to describe the motion of microparticles inside the blood vessels under navigation. Using that model, a SIMO fuzzy control algorithm is created and its validity is verified by computer-aided simulation results. After that, real experiments are designed and performed with simulated MR sequences in 2-D pulsatile flow.

CHAPTER 1: LITERATURE REVIEW

This chapter mainly focuses on the previous work of using MRI scanners as a method to propel microparticles in the human vascular system. This chapter is to state the feasibility of our MR-Sub project, to discuss the advantages and disadvantages of traditional controllers that are commonly used, and thus to demonstrate the possibility and necessity of finding a new approach for our control problem. Similar applications using underwater navigation technique are also reviewed.

In [1, 2, 3, 4], the authors propose the method of using a MRI scanner as a means of propulsion for microparticles. Preliminary studies have been done on the magnetic force induced by the scanner and on the evaluation of performance of ferromagnetic artefacts. The authors show that the size and material of the micro artefacts seem to be critical so that the position of robots could be retrieved from the distorted MR images.

Using an upgraded MRI scanner equipped with propulsion gradients coils, the authors in [1] performed a series of experiments to steer aggregating magnetic microparticles at a Y-shaped bifurcation. In their experiments, no controller for magnetic fields was applied. The intensity for magnetic gradients is fixed in a single experiment attempt and the direction for gradients is switched manually. The authors conclude that the magnetic particles could be steered towards a particular branch. Also, the steering ratio can be enhanced with higher magnetic gradients.

Another method to navigate the artefacts manually inside blood vessels is to use a handle console. In [6], the authors bring a console with 6 degrees of freedom. The scanner takes images for the bead at a rate of 1 frame per second and displays them on the screen. Then the MRI machine is able to create and apply magnetic resonance sequences to the bead to fulfill the navigation according to the commands received from the handle console operated by the user.

In order to achieve the automatic rather than manual control, a simple Proportional-Integral-Derivative (PID) controller is firstly presented in 2-D for real-time closed loop navigation of a ferromagnetic bead along a pre-defined trajectory with a clinical MRI in [4, 7]. In their design which is based on an approximate mathematical model in describing the magnetic force, the fluid drag force and the friction forces applied to the bead, a PID controller is designed to act along the tangent direction of the trajectory segment while a PD controller acts along the normal direction. 1-D pulsatile flow control experiment and 2-D quiescent flow control experiment are conducted

and analysed to verify their controllers. In general, this kind of PID controller is capable of guiding the bead to follow the waypoints along the trajectory by performing a point-to-point servo control. Showing its simplicity to be realised and operated, the PID controller has its advantages. Its stability could also be promised in a no-boundary 2-D quiescent flow [4, 7, 8]. On the other hand, the difficulties of PID control due to wide range of vessel diameters as well as time-varying environment parameters were also mentioned as probable constraints to any *in-vivo* experiment attempt [4].

Similarly in [9], a PID control algorithm is also validated on the control of a small-scale rotorcraft. Waypoints and trajectories are also specified for the PID which includes a double loop system for hover control and a triple loop system for forward flight. However, since the experiments are conducted in a unique simulation environment, it minimises the possibility to encounter complex, time-varying environment parameters.

In the fuzzy logic area, research has also been done for the application of Autonomous Underwater Vehicle (AUV) real-time control in [10, 11]. The control for AUV resembles the control for artefacts navigation inside blood vessels in some respects as both of the techniques need to be applied in fluid. However, AUVs could be equipped with a powerful internal motor so as to be driven to fulfill any 3-D motion at a low velocity while artefacts inside a blood vessel tend to rely strongly on external control methods and could hardly resist the fast-moving blood flow. In [11], the authors develop a self-adaptive fuzzy PID control for AUV. The fuzzy rules help to determine the PID parameters while the external environment has altered so as to overcome the perturbation brought from the unstable water waves. This kind of real-time continuous control method is extremely suitable for the AUV who travels at a low velocity underwater.

In [12], the authors verify the control effect on the guided glide missile while only the classical fuzzy algorithm is applied by SIMULINK/C++ hybrid programming. Through their simulation results, the pure fuzzy control algorithm improves the robustness of the whole system, but may have restrictions on the control precision as well. The consequence of control dead zones could not be discarded.

The work of Laurent Arcese [13] concentrates on proposing a nonlinear model and robust controller-observer for a magnetic micro carrier in a fluidic environment. To better describe the

motion of underwater micro carriers influenced by MR gradients, he uses the modern control theory and state space representation to depict the state transition of micro carriers. His simulation proves that the control law has an improved efficiency compared to the PID controllers in the quiescent flow of 1-D trajectory and 2-D Y-shaped trajectory, and that the high gain observer also has good performance in tracking and in filtering measurement noises under an ideal condition to increase the robustness.

CHAPTER 2: RESEARCH BACKGROUND

2.1 Research Object

The object of this study is to design and implement a SIMO fuzzy controller, which is able to propel and navigate a single microparticle or microparticle aggregations in pulsatile flow in real time to pass through Y-shaped blood vessel bifurcations with MR sequences according to a pre-defined trajectory.

2.2 Problem description

The microparticles, used as specialized drug carriers in our MR-Sub project, are supposed to be released from a catheter at a branch of human cardiovascular system where the catheter cannot access due to its size. Therefore, an appropriate control algorithm is required to navigate the microparticles after being released to travel along a series of waypoints and to pass through multiple bifurcations of blood vessels with MR gradients to finally reach the tumour position. Figure 2-1 shows a schematic diagram of this kind of intravascular navigation.

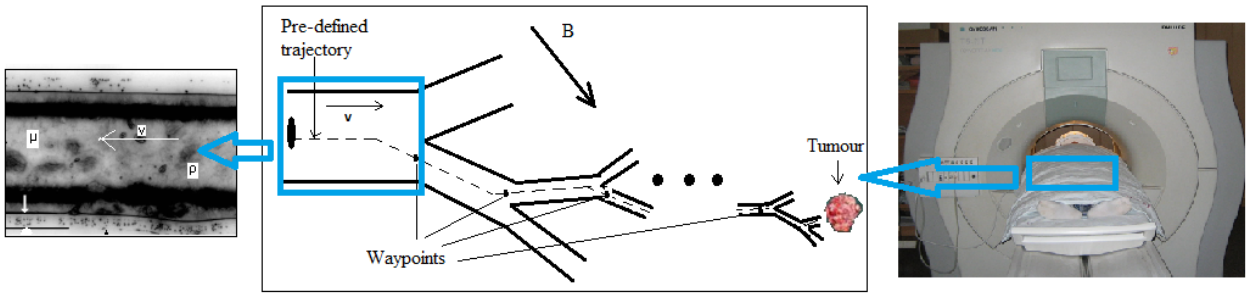


Figure 2- 1Schematic diagram of MR-Sub intravascular navigation

There are several constraints to the dedicated control algorithm.

1. It is demanded to become a real-time closed loop control, as feedbacks could be obtained from the MRI scanner in real-time.
2. The environment parameters such as the flow velocity or the blood pressure alter rapidly every time microparticles travel in different kinds of blood vessels, thus the controller needs to have good performance on robustness to resist external interferences.

3. Until now, clear and precisely-described mathematical models could hardly be found to model the blood fluid in human cardiovascular system. Complex non-linearity has been introduced. All kind of factors such the heartbeat, the vasomotion and the blood flow swirl increase vastly the uncertainty as well as the randomness of the whole human vascular system environment.
4. It would be more preferable if the controller has a very short responds time so as to save time for navigating the microparticles because sometimes microparticles travel at a really high velocity inside the blood vessel.
5. As a result of the high horizontal velocity of the blood flow, there is no method to define the microparticle entry zone to a certain bifurcation, i.e., the microparticle may access anywhere inside the parent branch of the bifurcation when injected. Hence, the real travelling trajectory of the microparticle could never be pre-calculated or previewed and each time the controller has to face to a new and also unique control problem.
6. The output of the controller would be programmed as MR sequences to generate gradients for steering the microparticles. However, the MR gradients seem not to be a perfect continuous output for control but have non-negligible rising time and falling time. It is also fairly important to take the sequence properties into consideration when designing the controller.

Here, we may now look further into the main obstacle of control, i.e., microparticles usually travel with the flow inside blood vessels at a very high velocity.

Figure 2-2 shows one typical bifurcation of human cardiovascular system on which analysis will be conducted.

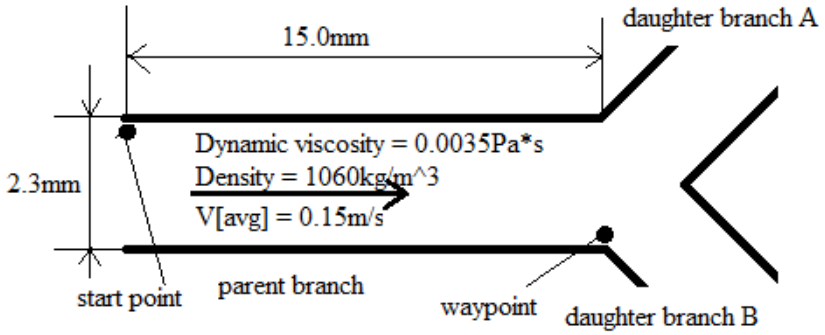


Figure 2-2 Schematic diagram of typical blood vessel bifurcation model

In Figure 2-2, the parent branch of the bifurcation has a length of 15.0mm as well as a diameter of 2.3mm. The start point is placed in the up left section of the parent branch while the current waypoint is placed in the down side which indicated that microparticles will need to be navigated from the start point into the daughter branch B. The dynamic viscosity and the density of the blood flow inside the vessel are 0.0035Pa·s and 1060kg/m^3 respectively. The flow, as well as the microparticles, goes from left to right and travels at an average horizontal velocity of 0.15m/s, as it is commonly assumed that microparticles travel with the flow inside blood vessels.

In this case, the ideal control solution should be capable of driving microparticles to travel at least 1.15mm in vertical direction before microparticles reach the junction or current waypoint. With the model described in Figure 2-2, it could be proved (in Chapter 3) that, in the vertical direction, the drag force follows the Stoles law thus the absolute value of maximum vertical velocity to which microparticles might possibly reach could be calculated as follows:

$$\left\|\overrightarrow{V_y}[max]\right\| = \frac{\left\|\vec{F}_{mag}\right\|}{6\pi\mu r} \qquad \textbf{Equation 2.1}$$

Hence, the minimum time to be consumed for one microparticle covering a distance of 1.15mm in vertical direction would be:

$$t_{y\_100\%}[min] = \frac{y[min]}{\left\|\overrightarrow{V_y}[max]\right\|} \qquad \textbf{Equation 2.2}$$

Combining Equation 2.1 and 2.2, and substituting the symbols with real clinic values, we could calculate that $t_{y\_100\%}[min] \approx 0.158\ s$.

It should be noticed that this result is obtained under all ideal circumstances. In the deduction, the necessity of acceleration procedure in the vertical direction of the microparticle is ignored. Moreover, the magnetic steering force is assumed to be induced at maximum power all the way along the steering process on the microparticle, which is also hardly possible to be realised.

However, with an average horizontal velocity of 0.15m/s, the previewed average time for a microparticle to pass the parent branch of the bifurcation could be estimated as follows:

$$t_x[avg] = {}^{0.15m}/_{u_{avg}} \approx 0.1s \qquad \textbf{Equation 2.3}$$

As we have $t_{y\_100\%}[min] > t_x[avg]$ here, therefore, it could be then concluded that basically it is impossible to navigate hundred percent of microparticles into the correct daughter branch as desired. The optimal control method could only concern about how to raise the percentage of the quantity of navigated microparticles over a total amount. This high velocity obstacle will be further discussed in details in chapter 5.3.1.

2.3 Methodology

Nowadays, the most popular control theories include the classical control, the modern control theory using state transition matrix, the fuzzy control theory, the predictive control theory, etc., and all of their combinations.

One common trait for the classical control theory such as the simple PID theory or the modern control theory is that both of them strongly rely on an accurately-described mathematical model for the analysis of the controlled object motion. Hence, while in the situations that contain multiple non-linear, time-varying parameters which is hardly possible to find a consistent mathematical model or is hardly possible to run the precise model with a computer, the classical control theory and the modern control theory may show their limitations.

In the case of MR-Sub project, the lack of a precise mathematical model that fits all the conditions of the motion of microparticles inside different kinds of blood vessels is a major obstacle for designing a classical PID controller or a modern controller as most of the environment parameters are non-linear and time-varying. Due to the instability of blood flow, all those parameters need to be re-collected each time the microparticles enter a new bifurcation and thus the control parameters require online re-adjustments. Also, considering the MR sequence features and the fast responds time requirements, a self-adaptive fuzzy control algorithm is then proposed.

The fuzzy control algorithm replaces the concept of precise values with the concept of fuzzy values [14]. For an arbitrary set $\boldsymbol{A}$ in a domain $\boldsymbol{Y}$ and an element x, unlike the classical set theory raised by Georg Cantor which has an absolute declaration of $x \in \boldsymbol{A}$ or $x \notin \boldsymbol{A}$, the fuzzy set theory utilizes a membership function μ_A to reflect the extents of elements belonging to the set. For example, $\mu_A(x)$, whose range is a closed interval [0, 1], reflects the extent of x belonging to $\boldsymbol{A}$. When $\mu_A(x) \to 1$, it shows that x has a very high tendency to be included in $\boldsymbol{A}$, while it shows that x has a low tendency to belong to $\boldsymbol{A}$ if $\mu_A(x) \to 0$. When $\boldsymbol{Y}$ is a finite set $\{x_1, x_2, x_3 \dots, x_n\}$, the Zadeh representation of fuzzy set $\boldsymbol{A}$ is given as follows [14]:

$$\boldsymbol{A} = \frac{\mu_A(x_1)}{x_1} + \frac{\mu_A(x_2)}{x_2} + \frac{\mu_A(x_3)}{x_3} + \cdots + \frac{\mu_A(x_n)}{x_n} \qquad \textbf{Equation 2.4}$$

where $\frac{\mu_A(x_i)}{x_i}$ is the membership of x_i belongs to $\boldsymbol{A}$.

The fuzzy control algorithm maps every input fuzzy set to an output fuzzy set according to multiple fuzzy rules. The creation of fuzzy rules imitates the human thought process which mostly depends on common experiences of human experts. For example, suppose that we have the input fuzzy set $\boldsymbol{A}$ = {raining heavily} in the domain $\boldsymbol{Y}$ = {weather conditions} and an output fuzzy set $\boldsymbol{O}$ = {umbrella} in the domain $\boldsymbol{Z}$ = {Belongings}, a fuzzy rule $\boldsymbol{R\_F}$ could be created as equation 2.2 to control the human behaviour:

$$\boldsymbol{R\_F} : \text{if } \boldsymbol{A} \text{ then } \boldsymbol{O} \qquad \textbf{Equation 2.5}$$

which means that if it is raining heavily then take an umbrella. Hence, in such weather systems, once the precise input weather value x has a high extent of belonging to $\boldsymbol{A}$, i.e., $\mu_A(x) \to 1$, the fuzzy controller responses automatically with an output control command of taking un umbrella.

As illustrated above, the fuzzy control process could be concluded as follows: first the controller takes the precise values for the inputs, and then it does fuzzifications of the precise values with membership functions of the input fuzzy sets. After deciding the extents of belonging to the fuzzy sets, the controller picks a specific value from the output fuzzy sets as the output of the whole control system in accordance with the corresponding fuzzy rules.

Since this kind of control is based on the experiences of human operators, and since it uses membership functions instead of precise values for describing the inputs, it tends to have less dependence on mathematical models and thus has more tolerance of external noises or alterations of environment parameters. Moreover, the fast responding time of the fuzzy controller could also be promised as its control process is as simple as a process of table look-up and output. Therefore, for the intravascular navigation of our MR-Sub application, the fuzzy controller has the potential to be a possible solution.

In the MR-Sub application, our crucial consideration for applying the fuzzy control is that, microparticles that have distinctions in positions or velocities should be treated differently. As shown in Figure 2-3, the whole zone of a branch of blood vessel that microparticles may access to is divided into finite number of sections. Each section is represented as an input fuzzy set. The boundaries between sections need to be defined by calculations of membership functions.

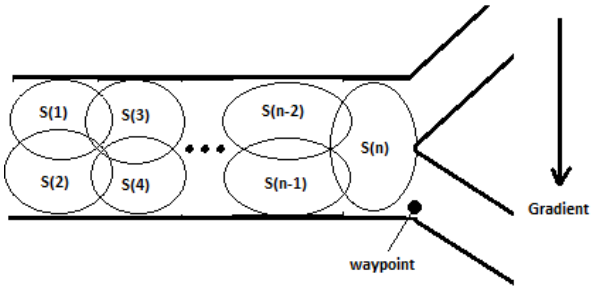


Figure 2- 3 Schematic diagram of fuzzification process of inputs

Having obtained the feedbacked coordinates of the microparticles from the MRI scanner at the beginning of a tracking-propulsion time period, the fuzzy controller would first calculate the fuzzy representations for inputs, and then determine one or several input fuzzy sets which conform with the particular position and velocity of microparticles at that time by comparing their membership functions. After that, a fuzzy reasoning process would be carried out using fuzzy rules corresponding to the related fuzzy sets based on expert operators' experiences. A unique controller output for the MR gradient is given afterwards which would then be transferred to MR sequences and be executed during this tracking-propulsion time period to steer microparticles until another tracking process begins.

The steering magnetic force for microparticles would be mainly applied to the vertical direction of the blood flow. And in the horizontal direction we assume that microparticles travel at the same velocity of blood flow and their motion could hardly be affected by magnetic forces.

A Matlab/C++ simulation platform for navigating a single microparticle and microparticle aggregations was set up to verify the fuzzy control algorithm for intravascular navigation. Fuzzy parameters are amended according to simulation and experiment results. The optimisation of fuzzy control will be discussed in the next chapters.

CHAPTER 3: MODELING

3.1 MR sequence and magnetic force

The system considered to fulfill the intravascular navigation is an upgraded Siemens Avanto 1.5T clinical MRI machine.

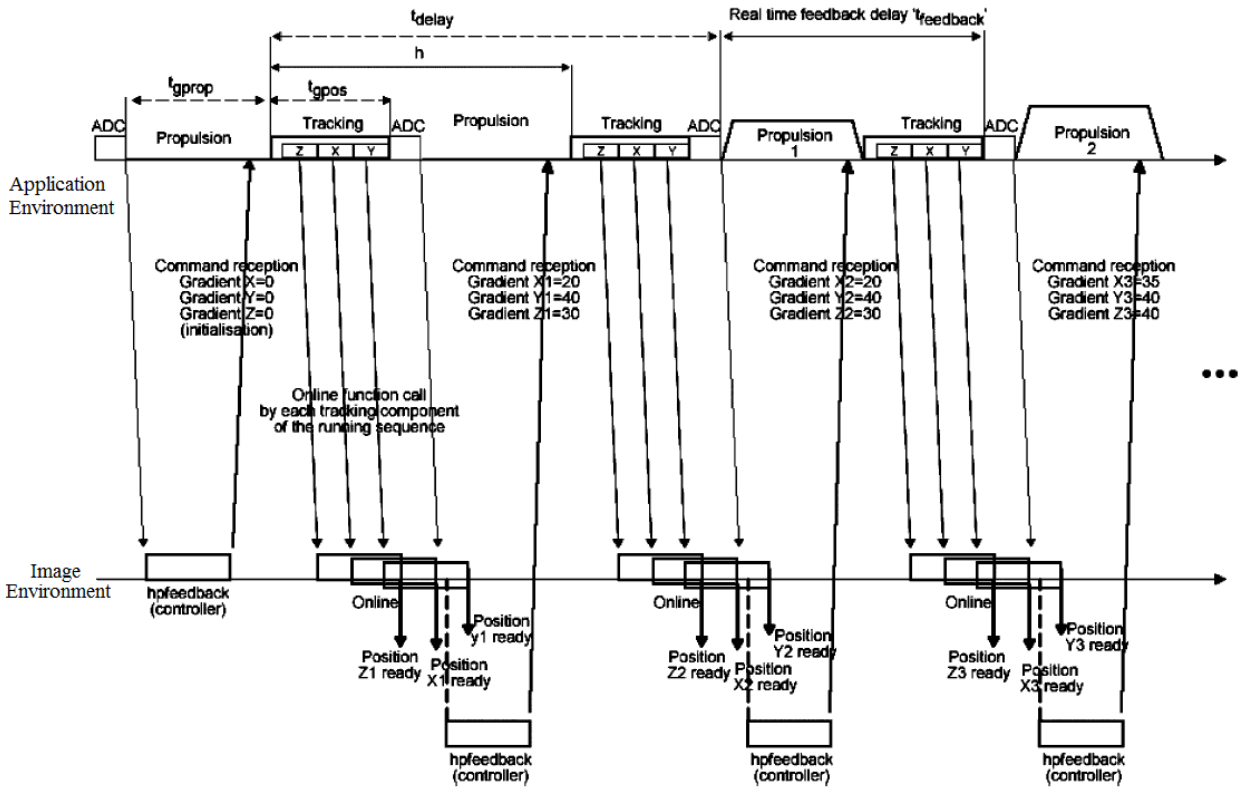


Figure 3- 1 Overview of real-time pulse sequence for 3-D control environment [4, 15]

The software architecture of the standard MRI system consists of two major parts: the Applications Environment and the Image Environment, as shown in Figure 3-1 [4, 15]. The Application environment is mainly responsible for the generation of MR sequences for tracking or propulsion while the Image Environment reconstructs MR images using the k-space data acquired from the running tracking sequence. Figure 3-1 shows an overview of a standard 3-D real-time sequence with time multiplexed positioning and propulsion phases.

As shown in Figure 3-1, the Application Environment module starts a period circle by performing a 3-D tracking process. The Image Environment collects k-space data from the

tracking process triggered in the Application Environment module. With the k-space data, the Image Environment is able to calculate and finally obtain the current 3-D coordinates of the under-controlled microparticle. After that, the Image Environment module will send the coordinate information back to the main propulsion controller located inside the Application Environment Module. The controller makes decisions according to the given coordinates and generates propulsion gradients. After the execution of the propulsion gradients, the Application Environment starts another "Tracking-Propulsion" circle.

Nowadays, with the modern real-time MRI feedback capabilities, the Image Environment module reconstructs images at the same time when Application Environment is running sequences and it is able to feedback before the next tracking-propulsion process begins which then allows the presence of a closed loop control infrastructure [3, 4]. In this case, the minimal time to acquire the xyz-coordinates of current position of a certain microparticle is t_{gpos}, and the paused running sequence is $t_{feedback}$. Thus the time left for a controller to calculate and apply 3-D propulsion gradients is t_{gprop}.

In the next several chapters of this thesis, a much simplified MR sequence generation model as shown in Figure 3-2 which combines the Application module and Image module will be used for the simulations as well as the experiments.

In Figure 3-2, there are some important factors related to the control process, i.e., *t_track* for the tracking unit of the Application module to take images and return the current coordinates of a microparticle; the maximum magnetic gradient ∇B to be reached in the next propulsion period and the time *t_maintain* to maintain that maximum gradient; the rising time *t_rise* and the falling time *t_fall* allowing the gradient to go to its maximum in the next propulsion period and to drop back to 0 afterwards. The adjustment factor $\Delta t\_maintain$ is used to add to the reference value *t_maintain* to make it more flexible and accurate. The important point here is that with current MRI system, the propulsion could not be executed while the tracking sequence is running at the same time. That is to say, the MRI propulsion unit has to maintain a zero output during the time when tracking and positioning are in progress (*t_track*). This kind of trapezoidal MR sequence will be repeated finite times to provide a steering force for microparticles to select a correct pathway to the tumour position.

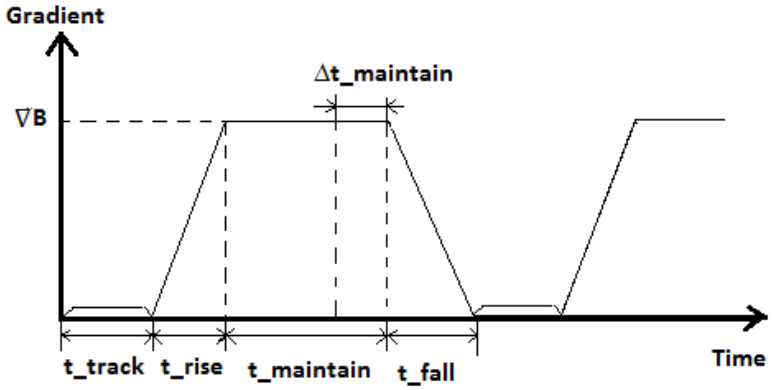


Figure 3- 2 Simplified MR real-time propulsion pulse sequence diagram

Taking the magnetic gradient $\overrightarrow{\nabla B}$, the magnetic force which is induced by a MRI gradient coil and which is applied to a piece of ferromagnetic artefact inside the magnetic gradient environment could be expressed as

$$\vec{F}_{mag} = V_f \cdot (\vec{M} \cdot \vec{\nabla})\vec{B} \qquad \textbf{Equation 3.1}$$

where $\vec{F}_{mag}$ is the magnetic force (N), V_f is the volume of the ferromagnetic entity (m^3), $\vec{M}$ is the magnetization of the material (A/m), $\overrightarrow{\nabla B}$ is the induced magnetic gradient (T/m).

3.2 Blood fluid

As a start, for a single bifurcation of the blood vessel, a 2-D coordinate system depicted in Figure 3-3 is set up in our study of modelling for the blood flow.

First the blood is assumed to be a Newtonian fluid as the Newtonian model has been applied adequately so far in the common blood flow problems including the local dynamics of flow through vascular junctions [16] because of its simplicity and because of the lack of alternatives as well.

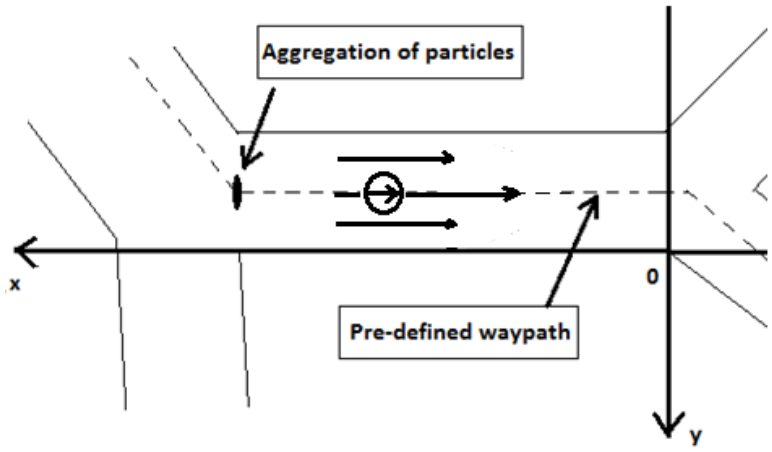


Figure 3- 3 Coordinate system for a single bifurcation

To further determine the fluid type along the x-axis, the Reynolds number must be specified. In the x-axis, we have:

$$R_{e\_p} = \frac{UD}{\nu} = \frac{UD\rho}{\mu} \qquad \textbf{Equation 3.2}$$

where R_{e_p} is the Reynolds number for the blood flow parallel to the x-axis, U is the average flow velocity in that direction (m/s), D is the diameter of the pipe (m), ν and μ represent the kinematic viscosity and the dynamic viscosity respectively, ρ is the flow density (kg/m^3). In the case of intravascular microparticle navigation, parameters are introduced to get a result of $R_{e\_p} \approx 104.5$.

Since in the x-axis, the Reynolds number $R_{e\_p} \approx 104.5 < 2000$, the fluid along could be treated as Laminar flow in which fluid elements move only in the main flow direction [16], as shown in Figure 3-3, i.e., in our simulation and experiments, the fluid component of the main flow along y-axis is not considered.

In the direction along y-axis, an approximate result of the Reynolds number is also calculated as follows so as to calculate the fluid drag force applied to microparticles when they are driven to veer inside the blood vessel:

$$R_{e\_v} = \frac{V_y \times 2r \times \rho}{\mu} \leq \frac{\frac{F_{mag}}{6\pi\mu r} \times 2r \times \rho}{\mu} = \frac{4}{9}\frac{M\nabla B \rho r^3}{\mu^2} \qquad \textbf{Equation 3.3}$$

where R_{e_v} is the Reynolds number for the flow perpendicular to x-axis (or parallel to y-axis), V_y is the steering velocity of the particle (m/s), r is the radius of the microparticle (m). Here, in our case, the maximum Reynolds number in y-axis obtained is $R_{e\_v} \approx 0.1283 < 1$. Hence, the simplified Stokes Law could be applied to calculate the fluid drag force for the direction perpendicular to x-axis which is:

$$\vec{f}_{drag\_y} = -6\pi\mu r\vec{V}_y \qquad \textbf{Equation 3.4}$$

where $\vec{f}_{drag\_y}$ is the fluid drag force in y-axis.

There is another important property for a fluid inside a tube which is called "No-Slip Boundary Condition". It states that in fluid flow there could not be any "step" change in velocity at any point within the flow field. As a result, fluid in contact with the stationary boundary must have zero velocity. This is because if a step change occurs near the boundary, the velocity gradient needs to be infinite and thus the force required maintaining the velocity gradient has to be infinite as well which is absolutely not possible [16].

Bases on the conclusion that the blood fluid is within the type of Laminar flow, a Poiseuille flow model assumption is then raised to better embody the "No-Slip Boundary Condition". Figure 3-4 shows the velocity profile in steady fully developed Poiseuille flow [16, 17].

In the Poiseuille flow model, as one of its properties, a numerical relationship could always be discovered between the local velocity in x-axis of a certain point inside the blood vessel and the average velocity which could be indicated as follows:

$$u_l = \left[1 - \left(\frac{\ell}{L}\right)^2\right] \times u_{max} = 2u_{avg}\left[1 - \left(\frac{\ell}{L}\right)^2\right] \qquad \textbf{Equation 3.5}$$

where u_l is a local velocity at a certain point inside the blood vessel, u_{max} and u_{avg} are the maximum velocity and average velocity of the flow in x-axis, ℓ stands for the distance from the certain point to the middle of the vessel, L stands for the radius of the vessel.

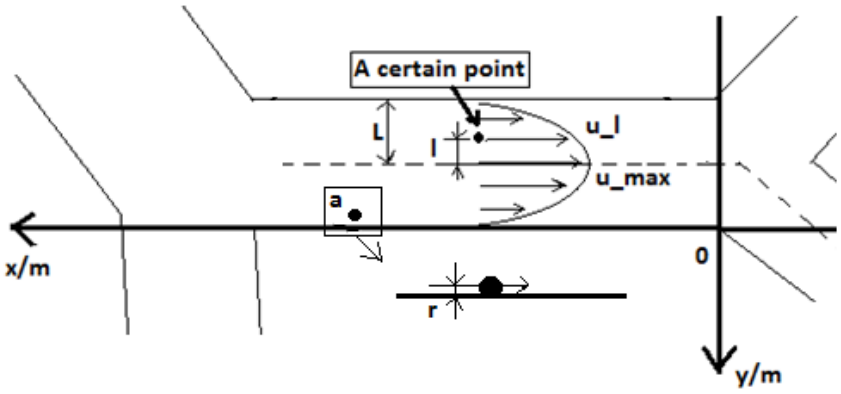


Figure 3- 4 Velocity profile in steady fully developed Poiseuille flow

In the present simulation, if a microparticle *a* is already stuck into a boundary as it appears in Figure 3-4, we assume that the boundary is rigid thus the particle still has a tiny distance against the boundary which would be its diameter *r* and still has a small velocity along x-axis. In reality however, it tends to be possible that microparticles would be pushed fiercely by the magnetic force against a boundary and stop moving. Further discussion will be done in the next several chapters to propose a method to prevent this situation from happening.

3.3 Force analysis for microparticles

Since the gravity force and the buoyancy force are negligible comparing with the other forces applied, 2-D force decomposition on a single microparticle is depicted in Figure 3-5. As microparticles are always travelling at the same speed as the blood flow in x-axis, they keep a stationary state relative to the blood flow in horizontal direction. Thus the compression forces from the flow to the microparticle are not depicted in Figure 3-5.

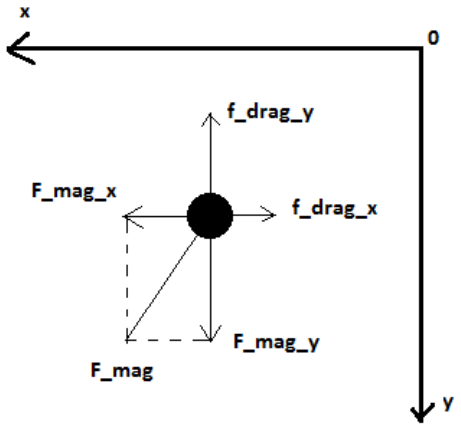


Figure 3- 5 Decomposition of forces on a single microparticle under navigation

Taking the equations 3.1 and 3.4, in the major steering direction (y-axis), an equation to describe the motion of microparticles under navigation is established from the simple Newton's Law:

$$\vec{F}_{mag\_y} + \vec{f}_{drag\_y} = V_f \cdot (\vec{M} \cdot \vec{\nabla})\vec{B} - 6\pi\mu r \frac{d\vec{s}}{dt} = m\frac{d^2\vec{s}}{dt^2} \qquad \textbf{Equation 3.6}$$

where m represents the mass of the ferromagnetic microparticle while $\vec{s}$ is the distance that the particle has travelled along the y-axis.

CHAPTER 4: CONTROLLER

4.1 Fuzzy controller overview

To provide steering force along the y-axis (as plotted in Figure 3-3), a SIMO Mamdani fuzzy controller is designed based on the mathematical model proposed in Chapter 3.

Figure 4-1 depicts the block diagram of closed-loop real time control applying the SIMO fuzzy controller.

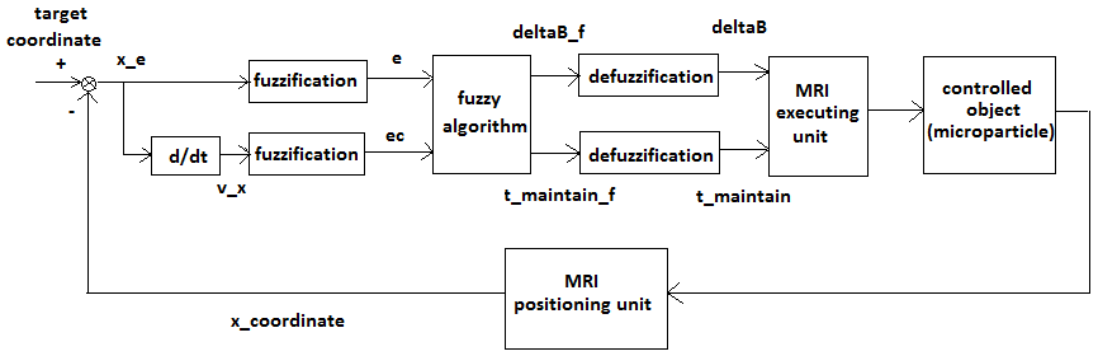


Figure 4- 1 Closed-loop SIMO fuzzy control block diagram

For navigating a single microparticle, the single input of the whole control system would be the difference of the *x*-coordinate of the target microparticle and the *x*-coordinate of the next waypoint which is feedbacked by the MRI positioning unit by conducting a tracking sequence. Having obtained the velocity along *x*-axis of the particle from the derivative of the input, the controller determines the values of ∇B and $\Delta t\_maintain$ shown in Figure 3-2 according to the current position and velocity of the microparticle as the two critical elements of the MR sequence and outputs. The MRI propulsion unit would then execute the sequence in the following propulsion period to create a magnetic force in y-axis to steer the target microparticle and maintain it until the next tracking-propulsion period begins.

For navigating a microparticle aggregation, the only difference is that the input would be the difference of the x-coordinate of the gravity center of the aggregation and the x-coordinate of the next waypoint.

In the following sections, the fuzzification process of inputs for the fuzzy controller core and the defuzzification process of outputs will be discussed. Fuzzy rule sets will be given depending on operators' experience.

4.2 Fuzzification of inputs and defuzzification of outputs

The fuzzy sets for both the inputs and the outputs of the controller core are defined as {Negative Big (NB), Negative Medium (NM), Negative Small (NS), Zero (O), Positive Small (PS), Positive Medium (PM), Positive Big (PB)}. The membership function definitions for the two inputs are given over a field of {-6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6} while those for the two outputs are defined over a field of {-7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7}.

4.2.1 Input of coordinate

The membership functions selected for the coordinate input are normal distribution functions, as the normal distribution is the most common probability distributions in nature. Therefore, the membership functions represented with normal distributions tend to allow input errors over a wider range.

The membership functions for coordinate input are given as follows:

$$\begin{cases} PB:\ E_1(x) = e^{-(\frac{x-6}{2})^2} \\ PM: E_2(x) = e^{-(\frac{x-4}{2})^2} \\ PS: E_3(x) = e^{-(\frac{x-2}{0.95})^2} \\ O\ : E_4(x) = e^{-(\frac{x}{0.75})^2} \\ NS: E_5(x) = e^{-(\frac{x+2}{0.95})^2} \\ NM: E_6(x) = e^{-(\frac{x+4}{2})^2} \\ NB: E_7(x) = e^{-(\frac{x+6}{2})^2} \end{cases} \quad \textbf{Equation 4.1}$$

where all the normal distribution parameters have been corrected through all simulations and experiments.

Figure 4-2 plots the coordinate input membership functions over a field of $\{x \in [-6, 6], y \in [0, 1]\}$.

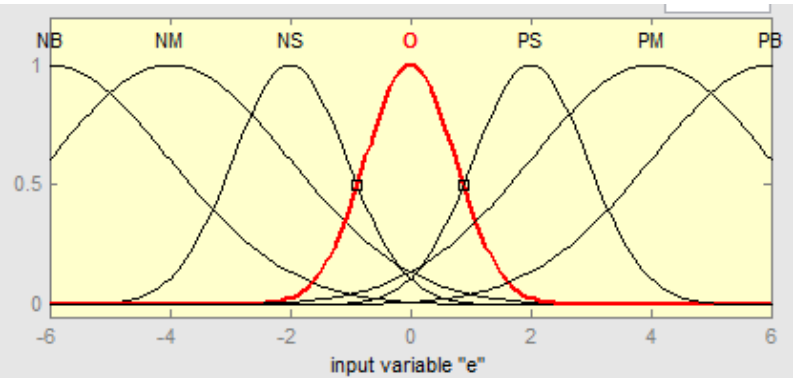


Figure 4- 2 Membership functions for coordinate input -E

Figure 4-2 indicates that when a coordinate input has been mapped from the real number field to the fuzzy field, it could have multiple corresponding input fuzzy sets. For example, the fuzzy controller would understand that an input of "+4" in the fuzzy field has a high possibility of belonging to the fuzzy set Positive Medium (PM), a fair possibility of belonging to the fuzzy set Positive Big (PB), a low possibility of belonging to the fuzzy set Positive Small (PS), and a scarce possibility of being included in any other fuzzy sets.

Table 4.1 lists the quantized membership functions of the coordinate input, preserving one significant figure after the decimal point.

Table 4. 1 Quantized membership functions of coordinate input -E

	-6	**-5**	**-4**	**-3**	**-2**	**-1**	**0**	**+1**	**+2**	**+3**	**+4**	**+5**	**+6**
NB	1.0	0.8	0.4	0.1									
NM	0.4	0.8	1.0	0.8	0.4	0.1							
NS				0.3	1.0	0.3							
O						0.2	1.0	0.2					
PS								0.3	1.0	0.3			
PM								0.1	0.4	0.8	1.0	0.8	0.4
PB										0.1	0.4	0.8	1.0

In our MR-Sub application, since it is crucial to decide whether the microparticles have passed the current waypoint or not (If it has already passed the current waypoint, although not significant, a new navigation loop needs to be started as soon as possible to save time for the second bifurcation navigation.), a high resolution and control sensibility are required. Therefore, the membership functions describing the concepts “O”, “PS”, “NS” are set to be acuter at the top than the other ones [11, 18].

4.2.2 Input of velocity

Ideally, the input of velocity needs to be first calculated from the derivative of the input of coordinate. However, since the tracking is a sampling process in discrete time, the velocity along x-axis of a target microparticle or the center of mass of an aggregation of microparticles is obtained using the formula below:

$$\|\overrightarrow{v_x}\| = \frac{\Delta p_x}{\Delta t} = \frac{\|p_x - p'_x\|}{t_{track} + t'_{rise} + t'_{maintain} + t'_{fall}} \qquad \textbf{Equation 4.2}$$

where $\|\overrightarrow{v_x}\|$ is the absolute value of the velocity along x-axis of the particle or the aggregation, p_x is the real-time x-coordinate feedback, p'_x is the recorded x-coordinate for the penultimate tracking process, t_track is the time for tracking, $t'_{rise}, t'_{maintain}, t'_{fall}$ are the rising time, maintaining time and falling time for the last sequence respectively.

For the same reason, the membership functions of the velocity input are using normal distribution functions defined below:

$$\begin{cases} PB:\ EC_1(x) = e^{-(\frac{x-6}{0.75})^2} \\ PM: EC_2(x) = e^{-(\frac{x-4}{0.75})^2} \\ PS: EC_3(x) = e^{-(\frac{x-2}{1})^2} \\ O\ : EC_4(x) = e^{-(\frac{x}{1.5})^2} \\ NS: EC_5(x) = e^{-(\frac{x+2}{1})^2} \\ NM: EC_6(x) = e^{-(\frac{x+4}{0.75})^2} \\ NB: EC_7(x) = e^{-(\frac{x+6}{0.75})^2} \end{cases} \qquad \textbf{Equation 4.3}$$

where all the normal distribution parameters have been tested and corrected through all simulations and experiments.

Figure 4-3 plots the velocity input membership functions over a field of $\{x \in [-6, 6], y \in [0, 1]\}$.

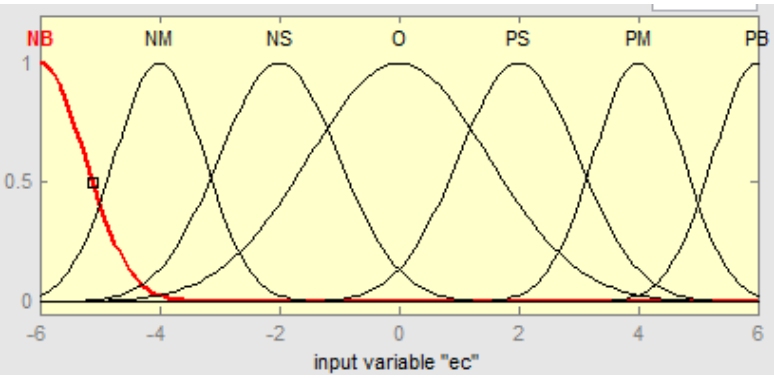


Figure 4- 3 Membership functions for velocity input -EC

Table 4.2 lists the quantized membership functions of the velocity input, preserving one significant figure after the decimal point.

Table 4. 2 Quantized membership functions of velocity input -EC

	-6	-5	-4	-3	-2	-1	0	+1	+2	+3	+4	+5	+6
NB	1.0	0.2											
NM		0.2	1.0	0.2									
NS				0.4	1.0	0.4							
O					0.2	0.6	1.0	0.6	0.2				
PS								0.4	1.0	0.4			
PM										0.2	1.0	0.2	
PB												0.2	1.0

In this case, the membership functions corresponding to fuzzy sets “NB” and “PB” have acuter tops. This is because when microparticles have a high x-velocity, the difficulty of control is increased and a higher control sensibility is required.

4.2.3 Output of magnetic gradient ∇B

The magnetic gradient is one critical output for the fuzzy control core as it decides the maximum absolute value for ∇B depicted in Figure 3-2.

As there is no outputting error, triangular distribution functions are chosen for membership functions mainly because of the ease of calculation.

$$\begin{cases} PB: G_1(x) = 1\cdot(x\geq 6) + \left(\frac{1}{3}x - 1\right)\cdot(6 > x \geq 3) + 0\cdot(3 > x) \\ PM:\ G_2(x) = \left(-\frac{1}{3}x + \frac{7}{3}\right)\cdot(x\geq 4) + \left(\frac{1}{3}x - \frac{1}{3}\right)\cdot(4 > x \geq 1) + 0\cdot(1 > x) \\ PS:\ G_3(x) = 0\cdot(x\geq 4) + \left(-\frac{1}{3}x + \frac{4}{3}\right)\cdot(4 > x \geq 1) + x\cdot(1 > x \geq 0) + 0\cdot(0 > x) \\ O:\ G_4(x) = 0\cdot(x\geq 1) + (-x+1)\cdot(1 > x \geq 0) + (x-1)\cdot(0 > x \geq -1) + 0\cdot(-1 > x) \\ NS: G_5(x) = 0\cdot(x\geq 0) + (-x)\cdot(0 > x \geq -1) + (\frac{1}{3}x + \frac{4}{3})\cdot(-1 > x \geq -4) + 0\cdot(-4 > x) \\ NM:\ G_6(x) = 0\cdot(x\geq -1) + \left(-\frac{1}{3}x - \frac{1}{3}\right)\cdot(-1 > x \geq -4) + \left(\frac{1}{3}x + \frac{7}{3}\right)\cdot(-4 > x) \\ NB:\ G_1(x) = 0\cdot(x\geq -3) + \left(-\frac{1}{3}x - 1\right)\cdot(-3 > x \geq -6) + 1\cdot(x > -6) \end{cases}$$

Equation 4.4

Figure 4-4 plots the magnetic gradient output membership functions over a field of $\{x \in [-7,7], y \in [0,1]\}$.

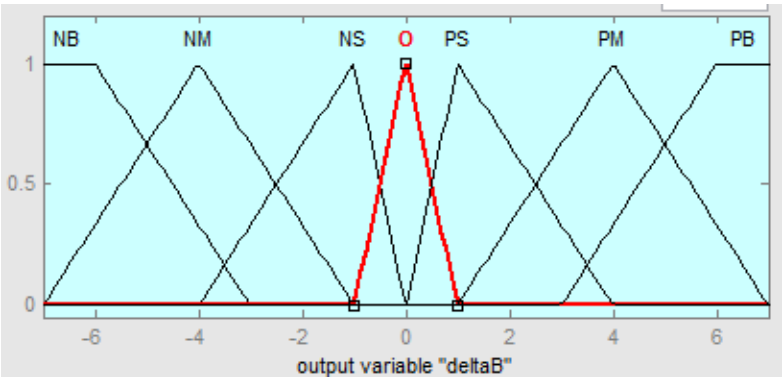


Figure 4- 4 Membership functions for magnetic gradient output -G

Table 4.3 lists the quantized membership functions of the magnetic gradient output, preserving one significant figure after the decimal point.

Table 4. 3 Quantized membership functions of magnetic gradient output -G

	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
NB	1.0	1.0	0.7	0.3											
NM		0.3	0.7	1.0	0.7	0.3									
NS					0.3	0.7	1.0								
O								1.0							
PS									1.0	0.7	0.3				
PM										0.3	0.7	1.0			
PB												0.3	0.7	1.0	1.0

4.2.4 Output of maximum gradient maintaining time adjustment $\Delta t\_maintain$

The maximum gradient maintaining time adjustment output describes the factor *t_maintain* and $\Delta t\_maintain$ depicted in Figure 3-2. This output is essential as the controller needs to discriminate the situations when the microparticles demand full power propulsion or when the microparticles are travelling at a low velocity thus more times for tracking could be gained.

The reason why the controller gives a direct output of $\Delta t\_maintain$ instead of *t_maintain* is that the time factor could not be negative. If *t_maintain* is used for direct output, the whole negative part of the fuzzy controller would then be meaningless, resulting in a narrow self-regulation range.

The triangular distributed membership functions are given as follows:

$$\begin{cases} PB: T_1(x) = 1 \cdot (x \geq 6) + \left(\frac{1}{3}x - 1\right) \cdot (6 > x \geq 3) + 0 \cdot (3 > x) \\ PM: T_2(x) = \left(-\frac{1}{3}x + \frac{7}{3}\right) \cdot (x \geq 4) + \left(\frac{1}{3}x - \frac{1}{3}\right) \cdot (4 > x \geq 1) + 0 \cdot (1 > x) \\ PS: T_3(x) = 0 \cdot (x \geq 4) + \left(-\frac{1}{3}x + \frac{4}{3}\right) \cdot (4 > x \geq 1) + (\frac{1}{3}x + \frac{2}{3}) \cdot (1 > x \geq -2) + 0 \cdot (-2 > x) \\ O: T_4(x) = 0 \cdot (x \geq 1) + \left(-\frac{1}{3}x + 1\right) \cdot (3 > x \geq 0) + \left(\frac{1}{3}x + 1\right) \cdot (0 > x \geq -3) + 0 \cdot (-1 > x) \\ NS: T_5(x) = 0 \cdot (x \geq 2) + \left(-\frac{1}{3}x + \frac{2}{3}\right) \cdot (2 > x \geq -1) + (\frac{1}{3}x + \frac{4}{3}) \cdot (-1 > x \geq -4) + 0 \cdot (-4 > x) \\ NM: T_6(x) = 0 \cdot (x \geq -1) + \left(-\frac{1}{3}x - \frac{1}{3}\right) \cdot (-1 > x \geq -4) + \left(\frac{1}{3}x + \frac{7}{3}\right) \cdot (-4 > x) \\ NB: T_1(x) = 0 \cdot (x \geq -3) + \left(-\frac{1}{3}x - 1\right) \cdot (-3 > x \geq -6) + 1 \cdot (x > -6) \end{cases}$$

Equation 4.5

Figure 4-5 plots the maximum gradient maintaining time adjustment output membership functions over a field of $\{x \in [-7,7], y \in [0,1]\}$.

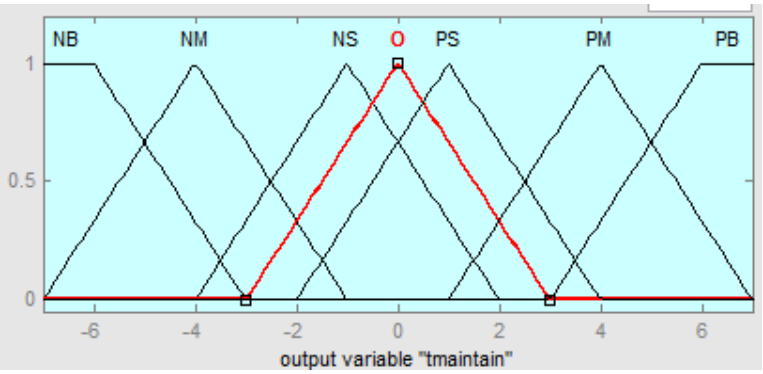


Figure 4- 5 Membership functions for maximum gradient maintaining time adjustment -T

Table 4.4 lists the quantized membership functions of the maximum magnetic gradient maintaining time adjustment output, preserving one significant figure after the decimal point.

Table 4. 4 Quantized membership functions of maintaining time adjustment output -T

	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7
NB	1.0	1.0	0.7	0.3											
NM		0.3	0.7	1.0	0.7	0.3									
NS					0.3	0.7	1.0	0.7	0.3						
O						0.3	0.7	1.0	0.7	0.3					
PS							0.3	0.7	1.0	0.7	0.3				
PM										0.3	0.7	1.0	0.7	0.3	
PB												0.3	0.7	1.0	1.0

4.3 Fuzzy rule sets

Normally, the fuzzy rule sets consist of a series of "IF *A* THEN *B*" condition judgements, where *A* usually is a combination of fuzzy input sets and *B* represents fuzzy output sets.

4.3.1 "IF...THEN..." fuzzy judgments

The "IF...THEN..." fuzzy judgements are determined mainly according to expert operators' experience as well as simulation and experiment results.

In our application, the basic design philosophy is that microparticles with different positions or velocities should be treated separately.

The rule sets are designed in the format as follows:

$$IF\ E\ and\ EC, THEN\ G\ and\ T$$

$$\begin{cases} IF\ (E\ is\ NB)\ and\ (EC\ is\ NB)\ THEN\ (G\ is\ NB) and\ (T\ is\ O) \\ IF\ (E\ is\ NB)\ and\ (EC\ is\ NM)\ THEN\ (G\ is\ NB) and\ (T\ is\ NS) \\ \vdots \\ IF\ (E\ is\ NM)\ and\ (EC\ is\ NB)\ THEN\ (G\ is\ NB) and\ (T\ is\ NS) \\ \vdots \\ IF\ (E\ is\ PB)\ and\ (EC\ is\ PM)\ THEN\ (G\ is\ PB) and\ (T\ is\ PB) \\ IF\ (E\ is\ PB)\ and\ (EC\ is\ PB)\ THEN\ (G\ is\ PB) and\ (T\ is\ PB) \end{cases} \quad \textbf{Equation 4.6}$$

Since we do not have any coupling between the two outputs, the SIMO fuzzy control problem (MIMO for the control core) could then be divided into two Single-Input-Single-Output problems. Table 4.5 and Table 4.6 list the two rule sets R1 for magnetic gradient output and R2 for maximum gradient maintaining time adjustment output respectively.

Table 4. 5 Rule sets R1 for magnetic gradient output

ec / e	**NB**	**NM**	**NS**	**O**	**PS**	**PM**	**PB**
NB	O	O	O	O	NS	NS	NM
NM	PS	PS	O	O	O	O	NS
NS	PM	PM	PS	PS	O	O	NS
O	PB	PM	PM	PM	PM	PS	O
PS	PB	PM	PM	PM	PM	PS	O
PM	PB	PB	PM	PM	PS	PS	O
PB	PB	PB	PM	PS	PS	O	O

Table 4. 6 Rule sets R2 for maintaining time adjustment output

ec / e	NB	NM	NS	O	PS	PM	PB
NB	NB	NB	NB	NB	NM	NS	O
NM	NB	NB	NB	NB	NM	NM	NM
NS	O	NM	NB	NB	NB	NB	NB
O	PS	O	NM	NB	NB	NB	NB
PS	PM	PS	NM	NM	NB	NB	NB
PM	PB	PM	NM	NM	NB	NB	NB
PB	PB	PB	NM	NM	NB	NB	NB

4.3.2 Quantized fuzzy outputs

For each output, we have 49 rules. Then the total fuzzy implication relation could be expressed as follows:

$$\boldsymbol{R}_i = \bigcup_{j=1}^{49} R_{ij}\,, i = 1{,}2 \qquad \textbf{Equation 4.7}$$

The quantized outputs are calculated using the formulas below (Take rule set R1 as an example):

$$\begin{aligned} &\boldsymbol{R}_1 = (E \times EC) \to G \\ &\therefore G = E \times EC \circ \boldsymbol{R}_1 = (E \times EC) \circ \bigcup_{i=1}^{n} \boldsymbol{R}_{1i} \\ &\quad = \bigcup_{i=1}^{n} \{(E \times EC) \circ [(E \times EC) \to G_i]\} \\ &\quad = \bigcup_{i=1}^{n} \{[E \circ (E_i \to G_i)] \cap [EC \circ (EC_i \to G_i)]\} \\ &\quad = \bigcup_{i=1}^{n} \{G_{iE} \cap G_{iEC}\} = \bigcup_{i=1}^{n} G_i \end{aligned} \qquad \textbf{Equation 4.8}$$

In these formulas, the interaction "×" simply uses the minimum weight of all the antecedents, the synthetic algorithm "∘" uses the max-min method and the implication "→" also takes the minimum weight.

The defuzzification of fuzzy outputs uses the "centroid" method, i.e., to take the weighted arithmetic mean of its membership function as the output to the executing unit of the MRI system, e.g., for the output ∇B, the defuzzification applies

$$g_0 = \frac{\sum_{i=1}^{7} g_i \mu_g(g_i)}{\sum_{i=1}^{7} \mu_g(g_i)} \qquad \textbf{Equation 4.9}$$

where $\mu_g(g_i)$ denotes the membership functions which are shown in Figure 4-4.

By programming with Matlab/Simulink Tool box, the quantized fuzzy control output tables are presented in Table 4.7 and Table 4.8 (See Annexe 1), preserving two significant figures after the decimal point. Linear transformations have been already operated to these tables to enlarge controllable area.

Table 4. 7 Quantized fuzzy output table for magnetic gradient

ec / e	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
-6	1.17	1.75	1.17	1.40	0.00	0.00	0.00	-1.44	-1.60	-1.63	-1.6	-3.50	-4.20
-5	1.39	1.75	1.39	1.40	0.00	0.00	0.00	-1.44	-1.39	-1.63	-1.39	-3.50	-3.73
-4	1.60	1.75	1.60	1.40	0.00	0.00	0.00	-1.44	-1.17	-1.63	-1.17	-3.50	-3.28
-3	3.17	3.77	3.17	2.95	1.24	1.40	1.24	0.78	0.41	-1.00	-0.64	-2.86	-2.56
-2	4.20	4.08	4.20	2.83	1.60	1.66	1.60	1.66	0.88	0.00	0.00	-1.75	-1.93
-1	4.86	4.94	4.45	3.50	3.50	3.50	3.50	3.11	3.27	3.27	1.56	0.00	-1.91
0	6.88	5.25	4.67	4.67	4.67	4.67	4.67	4.67	4.67	4.32	1.93	1.75	0.00
1	6.42	5.25	4.89	4.89	4.67	4.67	4.67	4.45	4.45	4.25	2.33	1.75	0.00
2	6.88	5.25	5.13	5.02	4.67	4.67	4.67	4.10	4.20	3.96	1.93	1.75	0.00
3	6.78	5.25	6.05	5.02	4.55	4.53	4.55	4.10	3.28	3.67	1.90	1.75	0.00
4	6.88	6.42	6.88	5.02	4.45	4.10	4.20	4.10	2.83	1.88	1.60	1.75	0.00
5	6.78	6.42	6.78	5.02	4.43	3.82	3.73	3.82	2.97	1.88	1.39	1.75	0.00
6	6.88	6.42	6.88	5.02	4.45	3.63	3.28	3.63	2.83	1.88	1.17	1.75	0.00

Table 4. 8 Quantized fuzzy output table for maximum gradient maintaining time adjustment

ec / e	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
-6	-6.88	-6.42	-6.88	-6.53	-6.88	-6.67	-6.88	-5.71	-4.89	-3.50	-2.42	-2.33	-1.71
-5	-6.78	-6.42	-6.78	-6.53	-6.78	-6.67	-6.78	-5.71	-4.90	-3.50	-2.92	-2.33	-2.33
-4	-6.88	-6.42	-6.88	-6.53	-6.88	-6.67	-6.88	-5.71	-4.89	-3.50	-3.42	-2.33	-2.96
-3	-4.30	-2.92	-6.05	-5.77	-6.78	-6.67	-6.78	-5.71	-5.01	-4.52	-4.56	-3.79	-4.37
-2	-2.18	-2.92	-5.13	-5.77	-6.88	-6.67	-6.88	-5.71	-6.11	-5.38	-6.11	-5.25	-6.11
-1	-0.89	-2.06	-3.06	-3.50	-5.54	-5.54	-6.42	-5.92	-5.92	-5.92	-5.92	-5.72	-5.92
0	1.17	0.58	0.00	-3.05	-4.89	-5.71	-6.89	-6.67	-6.88	-6.53	-6.88	-6.42	-6.88
1	3.55	2.64	1.52	-1.46	-5.08	-5.08	-5.08	-5.25	-5.54	-6.42	-6.42	-6.42	-6.42
2	5.13	3.50	2.42	-1.10	-4.67	-4.67	-4.67	-5.23	-6.42	-6.53	-6.88	-6.42	-6.88
3	6.05	3.50	3.65	-0.83	-4.67	-4.67	-4.67	-5.23	-6.26	-6.53	-6.78	-6.42	-6.78
4	6.88	5.25	5.13	-0.70	-4.67	-4.67	-4.67	-5.23	-6.42	-6.53	-6.88	-6.42	-6.88
5	6.78	5.25	5.63	-0.70	-4.67	-4.67	-4.67	-5.23	-6.26	-6.53	-6.78	-6.42	-6.78
6	6.88	5.25	6.11	-0.70	-4.67	-4.67	-4.67	-5.23	-6.42	-6.53	-6.88	-6.42	-6.88

Figure 4-6 and Figure 4-7 plot the input-output surfaces of the fuzzy controller.

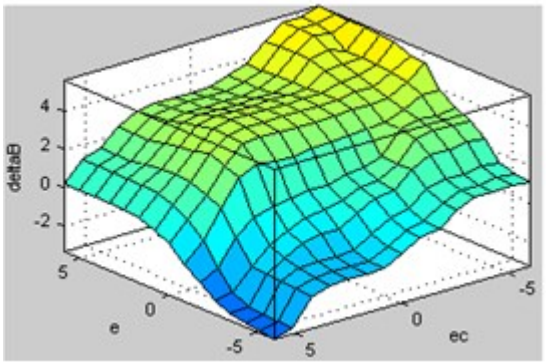


Figure 4- 6 Input-Output surface for the fuzzy implication (E×EC)->G

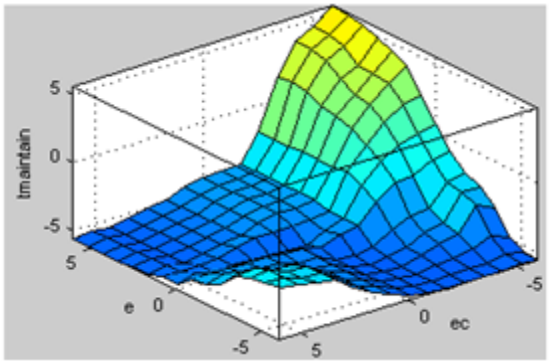


Figure 4- 7 Input-Output surface for the fuzzy implication (E×EC)->T

4.4 Controller design discussion

The assumption that blood follows a Poiseuille flow model is fairly important in the design of this kind of fuzzy controller. Since the MRI positioning unit is capable of feedbacking in real-time the x-coordinate as well as the y-coordinate of microparticles, a Multiple-Input-Multiple-Output (MIMO) fuzzy controller is firstly considered. However, as the Poiseuille flow model is introduced, a numerical relation could then be always discovered between y-coordinate and x-velocity of the same microparticles by using Equation 3.5. Hence, the SIMO fuzzy control algorithm is finally released with the purpose of decreasing its complexity. To design the fuzzy rule sets ***R***, we should be aware that the velocity input ***EC*** not only describes the motion of microparticles, but also reports the position of the navigation target in y-axis.

In essence, since the current MRI system is only able to run a sequence of propulsion or a sequence of tracking at a time, the outputs of fuzzy controller for magnetic gradient ***G*** and maintaining time adjustment ***T*** reflect a weigh of balance in putting the priority on propulsion or on tracking. Figure 4-8 shows the design approach of the fuzzy controller rule sets.

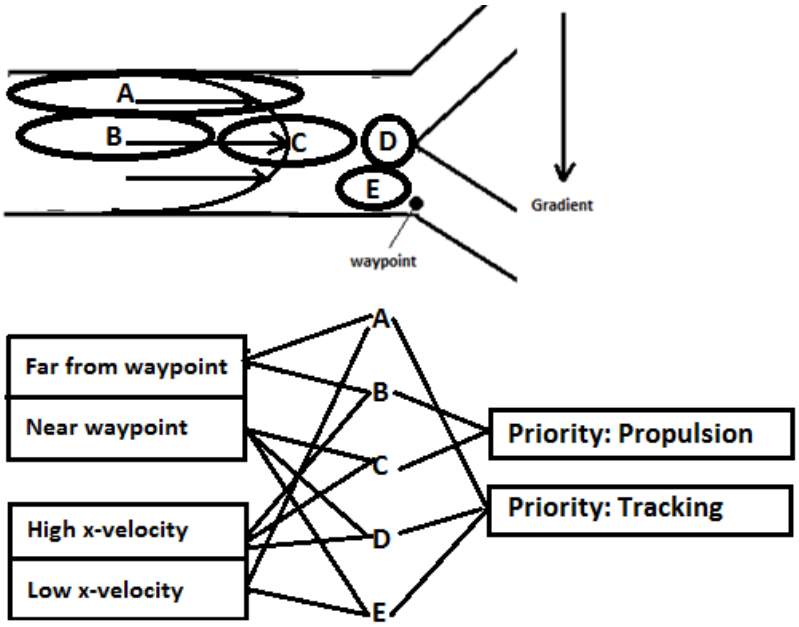


Figure 4- 8 Sketch of design approach for fuzzy rule sets

High magnetic gradient output always brings long rising time and falling time. Hence, low magnetic gradient as well as short maintaining time could be expected to save time for more tracking chances when priority is put on tracking, while propulsion priority normally leads to high magnetic gradient outputs and long maximum gradient maintaining time.

Among the five typical intravascular zones *A, B, C, D, E* plotted in Figure 4-8, priority will be put on propulsion for *B* and *C* to perform maximum power propulsion because of the high x-velocity. Microparticles inside zone *A* are supposed to be monitored frequently for the reason that the magnetic gradient may need to be switched to maximum power at any time. Zone *D* or *E* has a priority on tracking as the control effect should be evaluated as soon as possible before preparing for the next bifurcation navigation and having self-adjustments.

The fuzzy control rule sets also have definitions of outputs when the x-coordinate of the controlled target becomes negative or when its x-velocity becomes positive (which means that microparticles are actually receding from the next waypoint in x direction). This is not only due to the completeness requirement of controller design, but also because it is supposed that a well-designed fuzzy controller may demand a self-adaptive capability in case that the microparticles have already passed the branch but in a wrong direction misled by some burst errors or environmental disturbances. Although it is not possible to drive microparticles to travel upstream inside the blood vessel with our current clinical MRI system, we leave the potential to do this kind of self-adaptive control targeted to increase the robustness of the controller.

As to the problem described in the section 3.2 that microparticles may sometimes be pushed fiercely against a vessel boundary by the induced magnetic force which could prevent them from moving towards the next waypoint, new fuzzy rules may be added. A possible solution to this problem is that to create new rules for the controller to apply a tiny magnetic gradient in the opposite direction, as long as zero x-velocity of the target is being observed for a period of time. However, since the boundary is assumed to be rigid in our simulations and our *in-vitro* experiments, this kind of problem has never been encountered and thus is not considered in the proposed fuzzy controller.

CHAPTER 5: SIMULATIONS

5.1 Introduction

Using the mathematical model proposed by Equation 3.6, a computer-aided simulation platform is established to simulate the motion of microparticles under the navigation of the SIMO fuzzy controller inside blood vessels with Matlab/C++ hybrid programming.

Branch models of blood vessels with one single bifurcation and multiple bifurcations are both introduced to the simulation. To better verify the performance of the controller, navigation attempts on a single microparticle and on an aggregation of microparticles are also both made.

5.2 Software architecture

The program flow chart for simulation platform is shown in Figure 5-1.

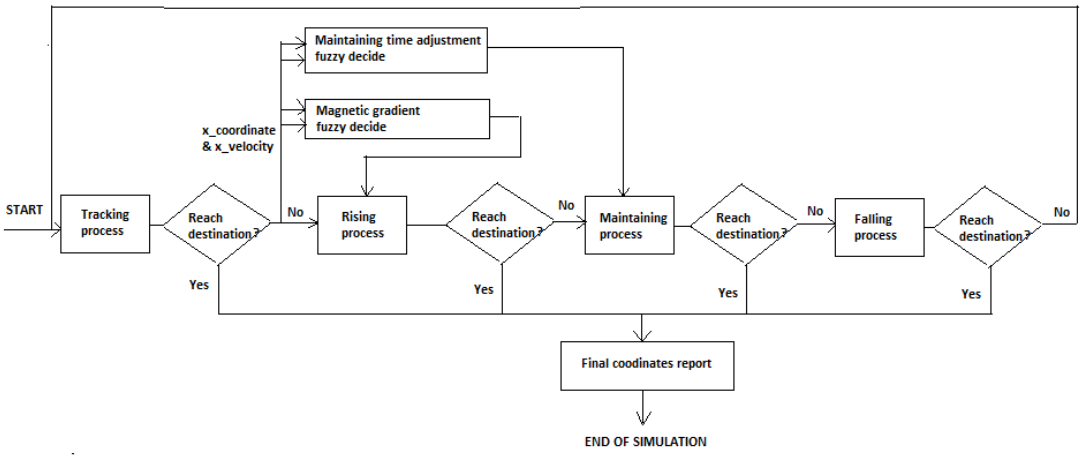


Figure 5- 1 Program flow chart for simulation platform

The program is run with an infinite loop consisting of four sequence processes in the order of a real MR sequence until the navigated target reaches its destination.

Inside the four main processes, time is divided into tiny time slots by a certain time step to perform the differential operation according to Equation 3.6, as depicted in Figure 5-2.

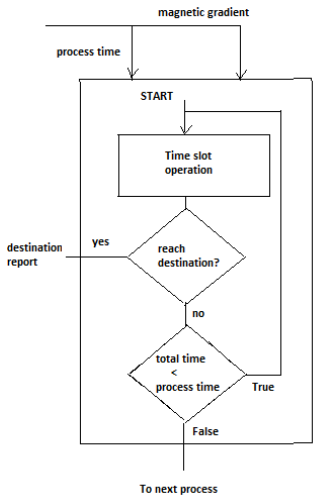


Figure 5- 2 Process programming flow chart

Figure 5-3 shows the deducing process of coordinates as well as velocities in a single time slot operation.

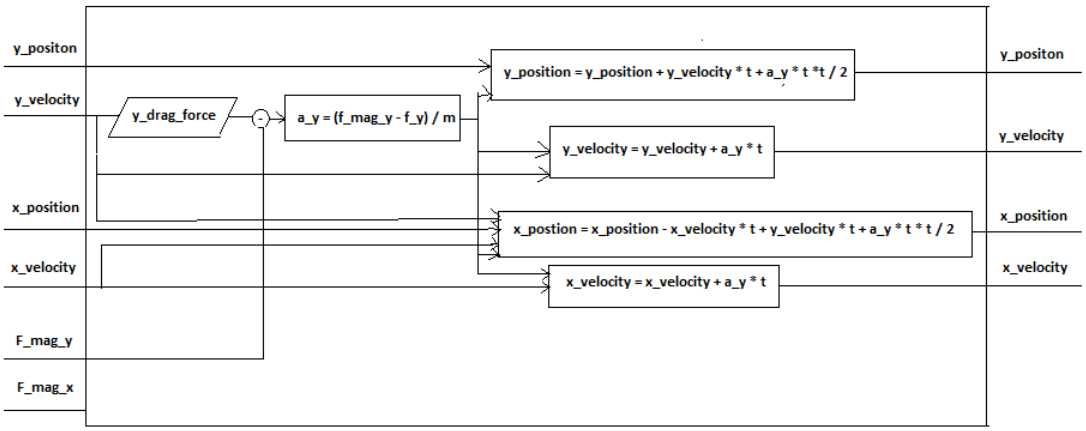


Figure 5- 3 Deducing process of coordinates and velocities in a single time slot operation

5.3 Results

5.3.1 Single-bifurcation simulation

The single bifurcation branch model of blood vessel is depicted in Figure 5-4.

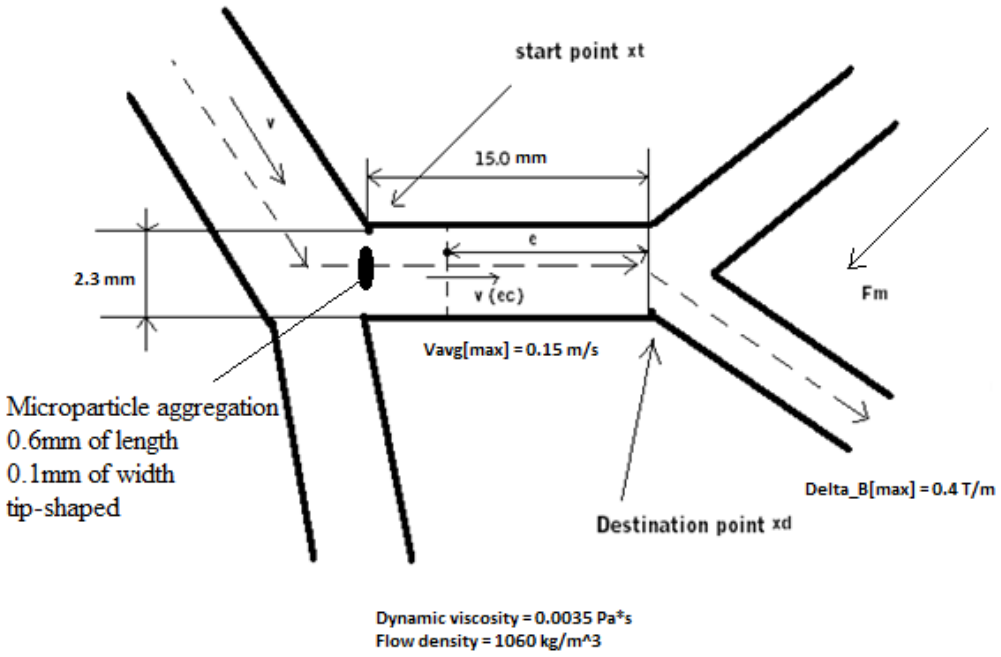


Figure 5- 4 Single bifurcation model of blood vessel for simulation

All the results of single bifurcation tests are obtained by simulations using the environment parameters specified in Table 5.1.

Table 5. 1 Simulation parameters used in single-bifurcation tests

Microparticle			**Blood vessel**			**MRI sequence**		
Name	*sym*	*Value*	*Name*	*sym*	*Value*	*Name*	*sym*	*Value*
Diameter (μm)	r	58	Length (m)	X_f	0.015	Tracking time(ms)	t_{track}	30
Mass ($kg \times 10^{-10}$)	m	4.56	Diameter (m)	Y_f	0.0023	Acceleration (T/m·s)	a_{mag}	40
Magnetic weight percentage(%)		45	Avg flow velocity (m/s)	u_{avg}	0.15	Reference maintaining time(ms)	$t_{maintai}$ $-\Delta t_{mai}$	60
Saturation Magnetization (A/m)	M	401440	Dynamic viscosity (Pa·s)	μ	0.0035	Main magnetic field(T)		1.5
Microparticle density (kg/m^3)	ρ_p	4460	Flow density (kg/m^3)	ρ	1060	Maximum magnetic gradient(T/m)	∇B_{max}	0.4

5.3.1.1 Single microparticle navigation

Figure 5-5 plots the different simulated trajectories inside a gridded blood branch model for microparticles starting from different initial positions and velocities. Figure 5-6 shows the magnetic gradients applied to a corresponding particle during its travel.

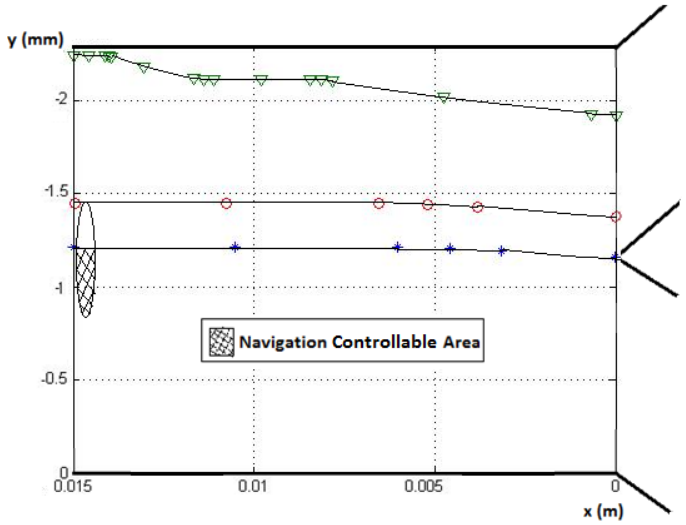


Figure 5- 5 Simulated trajectories for particles under navigation

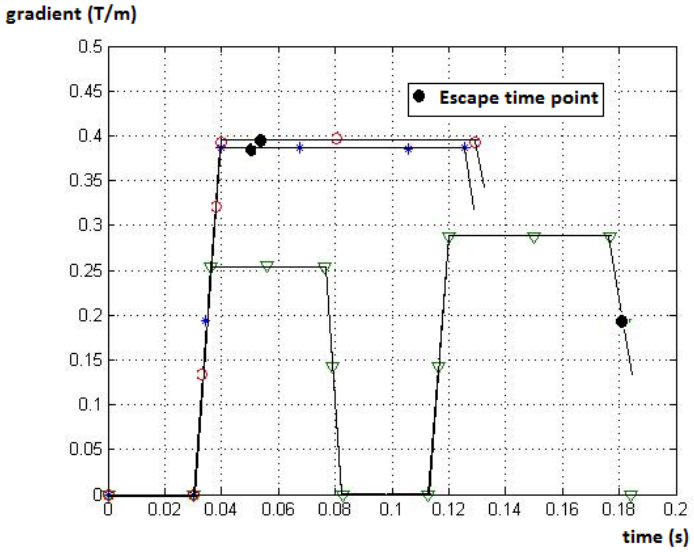
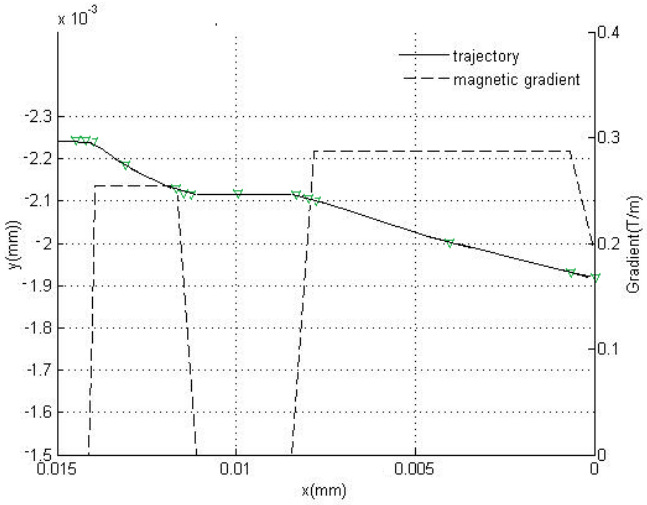


Figure 5- 6 Magnetic sequences applied to a corresponding particle in Figure 5-5

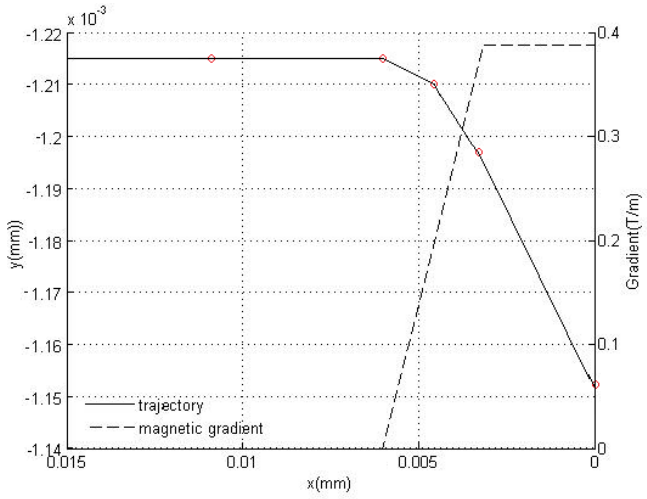
In Figure 5-5, the "Navigation Controllable Area" specifies a certain zone of a microparticle aggregation where all the microparticles could be successfully navigated into the bifurcation

underneath. The "escape time point" in Figure 5-6 indicates the time at which the x-coordinate of the particle reaches 0.

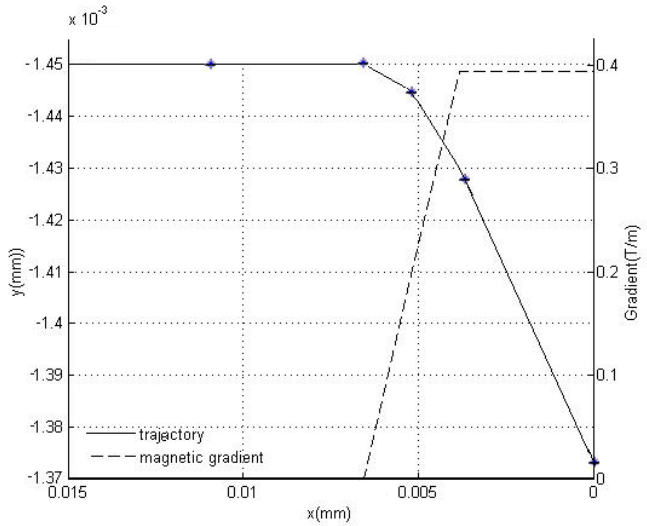
Figure 5-7(a)(b)(c) are the enlarged microparticle trajectories corresponding to those in Figure 5-5 as well as the magnetic gradients by *x* coordinates.



(a) Start position (0.015, -0.002242)



(b) Start position (0.01495, -0.001450)



(c) Start position (0.015, -0.001215)

Figure 5- 7 Trajectories and applied MR sequences by *x* coordinates for the corresponding microparticles in Figure 5-5 navigated in a single-bifurcation model

From Figure 5-7 we could better observe the impact of particle positions and velocities on the different shape of MR sequence outputs. After the first tracking and feedback, priority is still put on tracking when the microparticles travel at low velocities and have a distance from the waypoint (as shown in Figure 5-7(a)). When microparticles have high velocities and the time left tends to be insufficient, the MRI system just "forgets" about tracking and performs full power propulsion (Figure 5-7(b)(c)).

5.3.1.2 Microparticle aggregation navigation

We made an assumption that the ellipse-shaped aggregation of microparticles is close to a needle which is always parallel to the MRI main magnetic field [19]. Suppose that the interaction force between particles is negligible, Figure 5-8 depicts the rotation of ferromagnetic microparticle aggregations according to the main magnetic field, which is 1.5T with our current clinical MRI system.

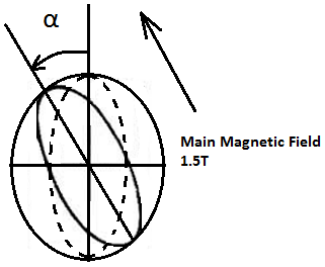


Figure 5- 8 Rotation of aggregations according to main magnetic field

In the simulation, using the coordinate system established in Figure 3-3, the elliptic equation for the typical particle aggregation shown in Figure 5-4 is given below:

$$\frac{(x-0.01495)^2}{0.00005^2}+\frac{(y+0.00115)^2}{0.0003^2}=1 \qquad \textbf{Equation 5.1}$$

Table 5.2 provides the navigation rates in our simulation as the angle α between the ellipse major axis and y-axis changes.

Table 5. 2 Navigation rate for single bifurcation test by alpha from simulation

α (radian)	**Total test samples**	**Navigated samples**	**Navigation rate (%)**
0	47078	29775	63.2
π/6	47078	30746	65.3
π/4	47078	32299	68.6
π/3	47078	35541	75.5
π/2	47078	47078	100.0

5.3.2 Multiple-bifurcation simulation

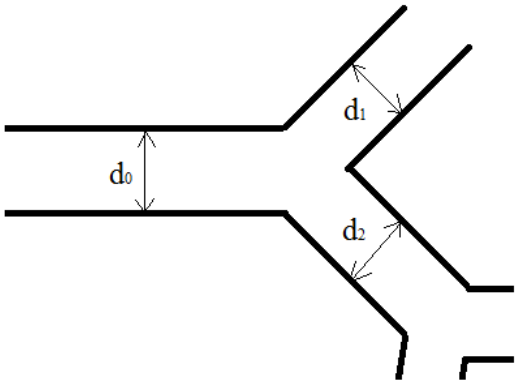


Figure 5- 9 Parent branch and daughter branches of a human vascular bifurcation

In [20], Murray's law is introduced to describe the relationship between the diameter of the parent branch (d_0) and the daughter branches (d_1 and d_2) of a bifurcation inside human vascular system, as depicted in Figure 5-9, i.e.:

$$d_0^3 = d_1^3 + d_2^3 \qquad \textbf{Equation 5.2}$$

For a symmetric bifurcation where $d_1 = d_2$, it follows that

$$d_0^3 = 2d_1^3 \qquad \textbf{Equation 5.3}$$

Hence, without loss of generality, a two-bifurcation blood vessel model could be proposed for the multi-bifurcation simulation as depicted in Figure 5-10 [21, 22, 23].

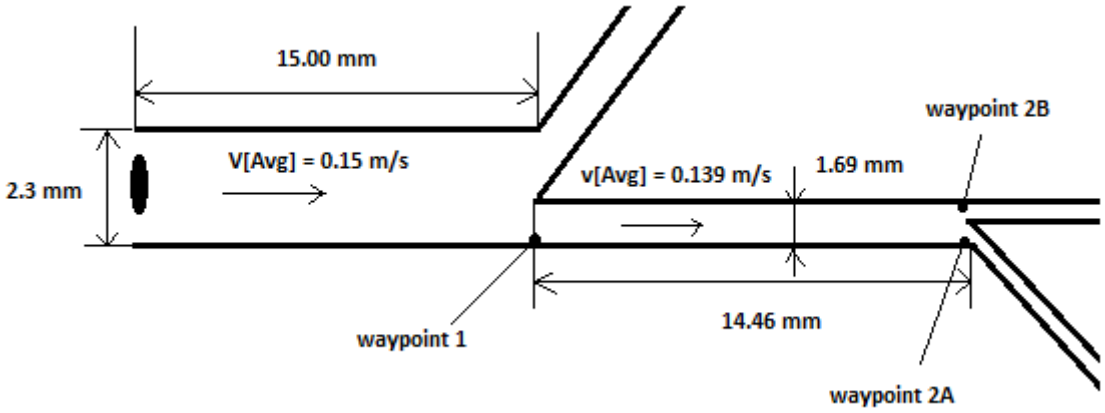


Figure 5- 10 Two-bifurcation vessel model for simulation

All the results of multiple-bifurcation tests are obtained by simulations using the environment parameters specified in Table 5.3.

In the simulations, the waypoint 1 for microparticles to pass the first bifurcation is always placed to the bottom-right. For the second bifurcation, two kinds of simulations are performed. Waypoint 2A is still placed to the bottom-right while waypoint 2B is placed to the up-right of the bifurcation. In the following sections, results for waypoint 1 – waypoint 2A navigation as well as waypoint 1 – waypoint 2B navigation will be presented.

First the fuzzy controller takes waypoint 1 as the destination point. The control process will be exactly the same as that in the single-bifurcation navigation. Having noticed that the microparticles have passed waypoint 1 by reading the coordinates feedbacked by a MR tracking process, the controller replaces waypoint 1 with waypoint 2A or 2B as the final destination point. In this case, the x-coordinate input for the fuzzy controller $\boldsymbol{E}$ would be the difference between the x-coordinate of the current target microparticle and the x-coordinate of waypoint 2A or 2B.

Table 5. 3 Simulation parameters used in single-bifurcation tests

Microparticle			**Blood vessel**			**MRI sequence**		
Name	*sym*	*Value*	*Name*	*sym*	*Value*	*Name*	*sym*	*Value*
Diameter (μm)	r	58	B1* Length (m)	X1	0.015	Tracking time(ms)	t_{track}	30
Mass (kg $\times 10^{-10}$)	m	4.56	B1* Diameter (m)	Y1	0.0023	Acceleration (T/m·s)	a_{mag}	40
Magnetic weight percentage(%)		45	B1* Avg flow velocity (m/s)	$u1_{avg}$	0.15	Reference maintaining time(ms)	$t_{maintai}$ $-\Delta t_{mai}$	10
Saturation Magnetization (A/m)	M	401440	B2* Length (m)	X2	0.01446	Maximum maintaining time adjustment (ms)	$\Delta t_{maintı}$	10
Microparticle density (kg/m^3)	ρ_p	4460	B2* Diameter (m)	Y2	0.00169	Maximum magnetic gradient(T/m)	∇B_{max}	0.4
			B2* Avg flow velocity (m/s)	$u2_{avg}$	0.139	Main magnetic field(T)		1.5
			Dynamic viscosity (Pa·s)	μ	0.0035			
			Flow density (kg/m^3)	ρ	1060			

***B1 = Bifurcation 1, B2 = Bifurcation 2**

5.3.2.1 Single microparticle navigation

The microparticles to be depicted in figures would be: the microparticles which could be precisely navigated to pass the first bifurcation, the microparticles which could be precisely navigated to pass the second bifurcation, and the microparticles at the top or bottom end of the ellipse-shaped aggregation.

5.3.2.1.A Waypoint 1 – Waypoint 2A navigation

Figure 5-11 plots the different simulated trajectories inside a gridded two-bifurcation model with pre-set waypoints 1 – 2A for microparticles starting from different initial positions and velocities.

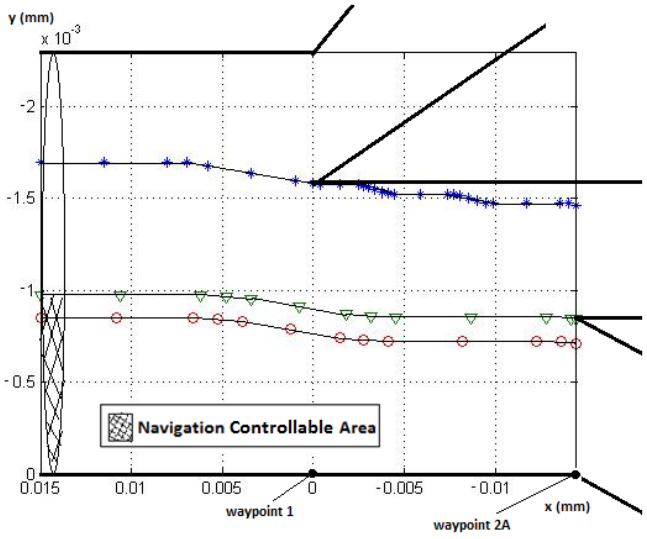
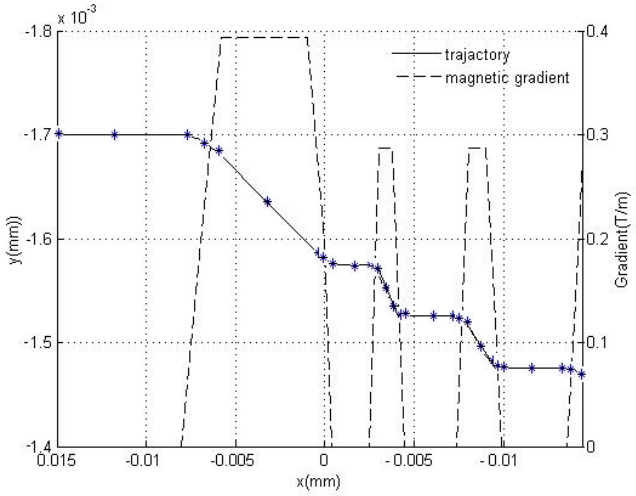
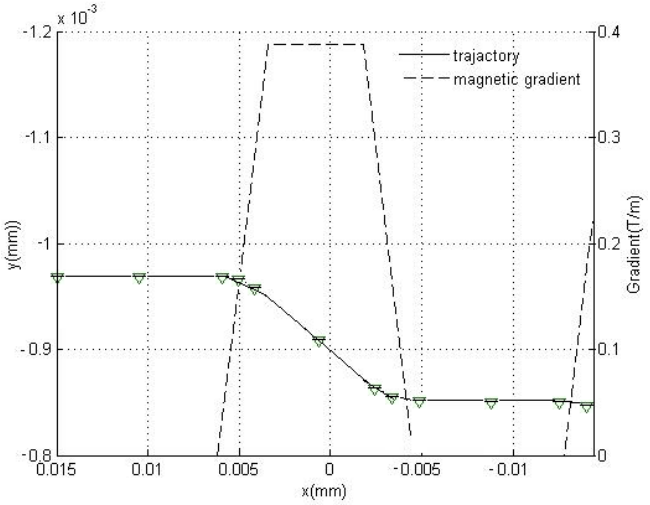


Figure 5- 11 Simulated trajectories for particles under navigation inside a two-bifurcation blood vessel model with waypoints 1-2A

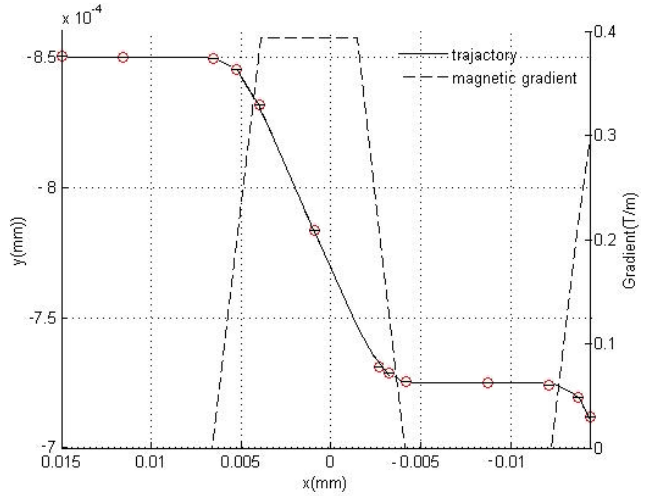
Figure 5-12(a)(b)(c) are the enlarged microparticle trajectories corresponding to those in Figure 5-11 as well as the magnetic gradients by *x* coordinates.



(a) Start position (0.015, -0.0017)



(b) Start position (0.015, -0.000969)



(c) Start position (0.015, -0.00085)

Figure 5- 12 Trajectories and applied MR sequences by *x* coordinates for the corresponding microparticles in Figure 5-11 navigated in two-bifurcation model with Waypoints 1-2A

5.3.2.1.B Waypoint 1 – Waypoint 2B navigation

Figure 5-13 plots the different simulated trajectories inside a gridded two-bifurcation model with pre-set waypoints 1 – 2B for microparticles starting from different initial positions and velocities.

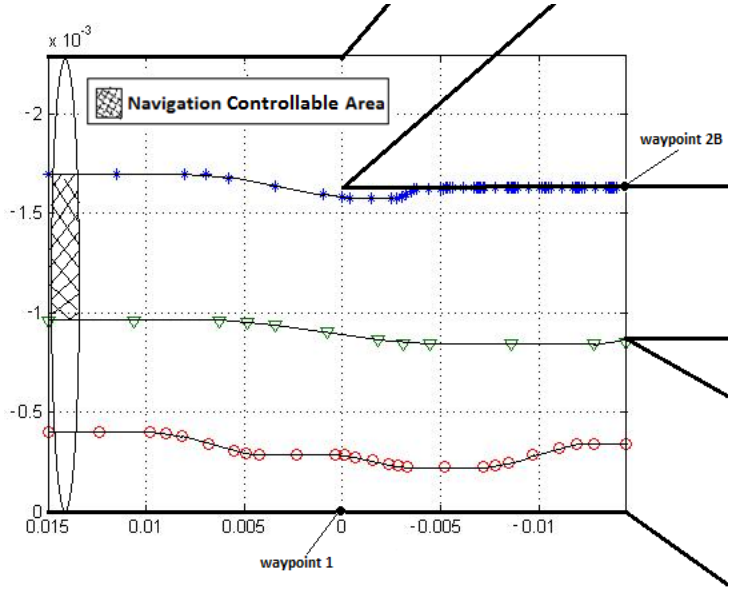
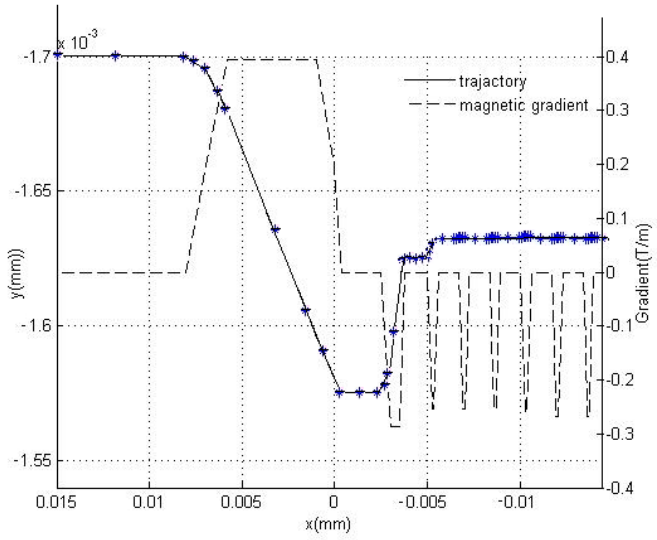
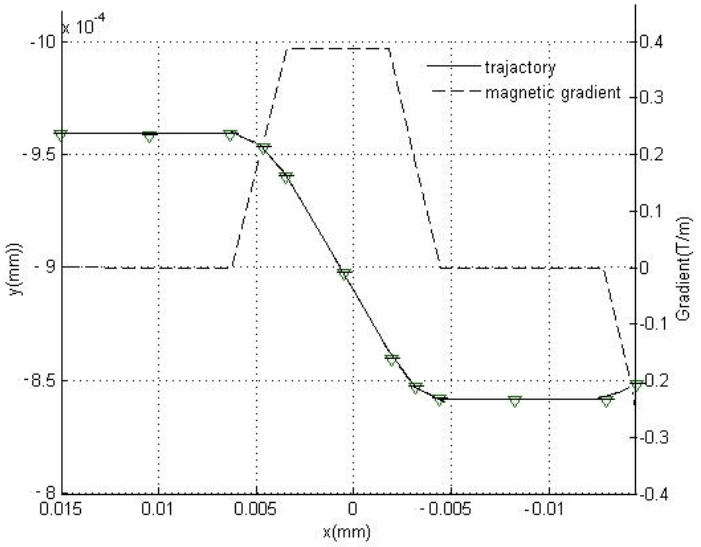


Figure 5- 13 Simulated trajectories for particles under navigation inside a two-bifurcation blood vessel model with waypoints 1-2B

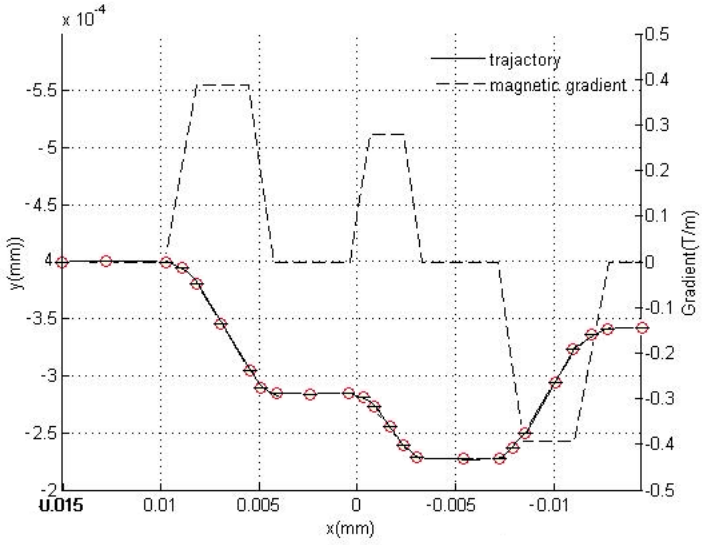
Figure 5-14(a)(b)(c) plot the enlarged microparticle trajectories corresponding to those in Figure 5-13 as well as the magnetic gradients by *x* coordinates.



(a) Start position (0.015, -0.0017)



(b) Start position (0.015, -0.009595)



(c) Start position (0.015, -0.0004)

Figure 5- 14 Trajectories and applied MR sequences by *x* coordinates for the corresponding microparticles in Figure 5-13 navigated in two-bifurcation model with Waypoints 1-2B

5.3.2.2 Microparticle aggregation navigation

With the same definition of angle α in Figure 5-8, the microparticle aggregation navigation tests apply the elliptic formula below to describe the typical aggregations, as shown in Figure 5-11 and Figure 5-13:

$$\frac{(x-0.01495)^2}{0.00005^2} + \frac{(y+0.00115)^2}{0.001121^2} = 1$$ **Equation 5.4**

5.3.2.2.A Waypoint 1 – Waypoint 2A navigation

Table 5.4 provides the navigation rates in our two-bifurcation simulation with waypoint 1 – waypoint 2A as the angle α between the ellipse major axis and y-axis changes.

Table 5. 4 Navigation rate by alpha for two-bifurcation test with waypoints 1-2A from simulation

α (radian)	Total test samples	Navigated samples	Navigation rate (%)
0	341	132	38.7
π/6	341	131	38.4
π/4	341	123	36.1
π/3	341	102	29.9
π/2	341	0	0.0

5.3.2.2.B Waypoint 1 – Waypoint 2B navigation

Table 5.5 provides the navigation rates in our two-bifurcation simulation with waypoint 1 – waypoint 2B as the angle α between the ellipse major axis and y-axis changes.

Table 5. 5 Navigation rate by alpha for two-bifurcation test with waypoints 1-2B from simulation

α (radian)	Total test samples	Navigated samples	Navigation rate (%)
0	456	252	55.3
π/6	456	287	62.9
π/4	456	334	73.2
π/3	456	383	84.0
π/2	456	456	100.0

5.4 Simulation discussion

5.4.1 Single-bifurcation navigation discussion

From Figure 5-5 and Table 5-2, we could read that even though this kind of SIMO fuzzy controller is applied, we still suffer a great loss of particles for one single bifurcation. This phenomenon could be explained by the following calculation.

We have already established a mathematical model to describe the motion for microparticles under navigation which is summarized by Equation 3.6:

$$V_f \cdot (\vec{M} \cdot \vec{\nabla})\vec{\mathrm{B}} - 6\pi\mu r \frac{d\vec{s}}{dt} = m\frac{d^2\vec{s}}{dt^2}$$

Since all of our calculation is along the y-axis, we could turn all the vector variables into scalar variables. Do integration to both sides of the equation to resolve the y-axis distance *s*, we get:

$$s = M\nabla \mathrm{B} V_f \left(\frac{m}{36\pi^2\mu^2 r^2} e^{-\frac{6\pi\mu R}{m}t} + \frac{1}{6\pi\mu r} t - \frac{m}{36\pi^2\mu^2 r^2}\right) \qquad \textbf{Equation 5.5}$$

where the representative meanings of all the symbols have already been explained in chapter 3.

Substituting all the symbols with real values specified in Table 5.1, we know that, in order to have a minimum moving distance of $s_{min} = \frac{s}{2} = 0.00115\ m$ in y-axis, which means that the 100% of the microparticles could be navigated into the desired branch, the actual minimum time required for navigation is approximately calculated to be $t_{y\_min} \approx 0.1585\ s$.

The average flow rate in x-axis of the given blood vessel u_{avg} could be obtained as follows, using the poiseuille flow assumption:

$$u_{avg} = {u_{max}}/{2} = 0.15\ m/s \quad \textbf{Equation 5.6}$$

Thus the average time for microparticles to pass the navigation zone t_{x_avg} is

$$t_{x\_avg} = {X_F}/{u_{avg}} \approx 0.1s \quad \textbf{Equation 5.7}$$

Hence, since $t_{y\_min} > t_{x\_avg}$, to take the time reserved for tracking and acceleration of magnetic gradient into consideration, we could conclude that it is a non-solution control problem aiming at navigating all the microparticles inside a blood vessel as desire. Only an optimal control algorithm could be studied to raise the percentage of navigated microparticles among all the microparticles.

One possible improvement to the controller that we have already applied in our simulation is shown in Figure 3-5 as well as Figure 5-4. That is, the magnetic gradient produced by MRI coil for propelling microparticles is designed to have an angle of 45 degrees to the negative x-axis all the time (if propulsion with steering is desirable). That is to say, the magnetic force induced by the gradient is always trying to pull the particle back in x-axis while propelling in y-axis at the same time, although not significant, so as to obtain more time for navigation.

The maximum backward x-velocity created by magnetic gradient $u_{x\_back\_max}$ could be estimated below:

$$u_{x\_back\_max} = \frac{Fm_{x\_max}}{6\pi\mu r} = \frac{M\nabla B_{\text{x\_max}} V_f}{6\pi\mu r} \approx 0.0073\ m/s \quad \textbf{Equation 5.8}$$

where Fm_{x_max} and $\nabla B_{\text{x\_max}}$ are the maximum magnetic force and gradient induced by the MRI core in x-direction respectively.

Comparing $u_{x\_back\_max}$ with the average flow rate of the blood flow u_{avg} in x-axis which is 0.15 m/s, we could see that the attempt of deceleration for microparticles is not capable of letting us to receive great extra time for navigation. Nonetheless, it still could be a positive solution, especially when more puissant magnetic coils may be phased in the future.

In our simulation, there is another important assumption that microparticles will always gather as an ellipse-shaped aggregation with a uniform distribution, as depicted in Figure 5-8. Besides, in our model, the interaction force between particles is also assumed to be negligible. In reality however, the structure inside the aggregation of microparticles is unpredictable. References [24, 25] show the importance for modeling for aggregations and interaction forces.

5.4.2 Multiple-bifurcation navigation discussion

The ideal circumstance for multiple-bifurcation navigation is that when entering a new bifurcation, the majority of microparticles could be propelled to the middle of the tube by the former MR sequences for the last bifurcation. In our case, since the controller is not able to receive the coordinates of a new waypoint unless the navigation for the current waypoint terminates, the situation of the next bifurcation always keeps unknown to the controller. Therefore, rather to propel microparticles to the sides of the tube, keeping the microparticles travelling in the center would be the only best solution so that we do not risk losing much particles in one bifurcation.

The consequence of such control principle is obvious. If we remove the time for the execution of former sequences, the time left for driving the microparticles in a second bifurcation becomes very small, due to the high velocity in the flow center. From Figure 5-12(b)(c) and Figure 5-14(b) we could read that the time left for propelling microparticles in the second bifurcation is not sufficient to fulfill even only the rising process of a MR propulsion sequence, which has a definite negative effect on the navigation rate of the second bifurcation.

One solution to this problem is to let the controller collect the information of more than one bifurcation. That is, with an upgraded “predictive fuzzy controller”, the control system could be aware of not only the coordinates of the current waypoint at a time, but also the coordinates of the waypoint for the next bifurcation that the microparticles are about to entering. With a knowledge that the next waypoint is either in the same side of the blood vessel or in the other side of the blood vessel (i.e. the next waypoint is either along the same boundary of the vessel or near the opposite side of the boundary), the controller could better decide the requirement of magnetic forces or navigation time so that the propulsion would never be performed in excess of need.

Another restriction of this SIMO fuzzy controller is that it has only the x-coordinate of the target as its input. For the situation of the navigated target along y-axis, the controller simply uses its x-velocity as a variable to decide, as we have a Poiseuille flow model. Hence, as a result, due to the symmetry of the flow, the controller is not capable of separating the two particles symmetrically distributed to the center of the blood flow, as depicted in Figure 5-15.

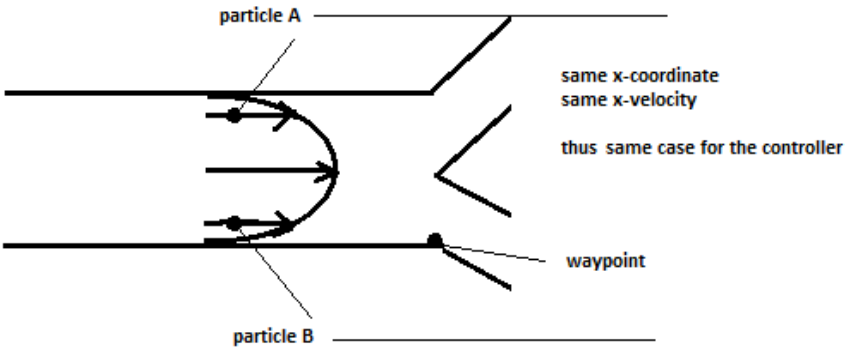


Figure 5- 15 Restriction of controller due to the symmetry of Poiseuille flow

The direct effect of such restriction could be observed in Figure 5-11 together with Figure 5-12(a). Although for the second bifurcation, the microparticle is far from its desired waypoint, the controller is confused that the target is in the “Particle A” case or “Particle B” case described in Figure 5-15, so it decides to put a priority on tracking to follow closely on the future change of the target. As a result, a lot of time would then be wasted in tracking processes.

CHAPTER 6: EXPERIMENT

6.1 Introduction

The experiments aim at testing the performance of the newly designed SIMO fuzzy controller in propelling and steering microparticles inside blood vessels. These in-vitro experiments are carried out in pulsatile flow with a 2-D. For now, only the one-bifurcation tests are fulfilled. Related results and discussions are presented in the following sections.

6.2 General hardware setup

The whole hardware platform of the *in-vitro* control experiments are shown in Figure 6-1, which mainly consists of a glass Y-shaped phantom, Maxwell coils for propulsion, a high resolution real-time camera, a high-power pump as well as an upgraded MRI scanner system to provide outside 1.5T main magnetic field.

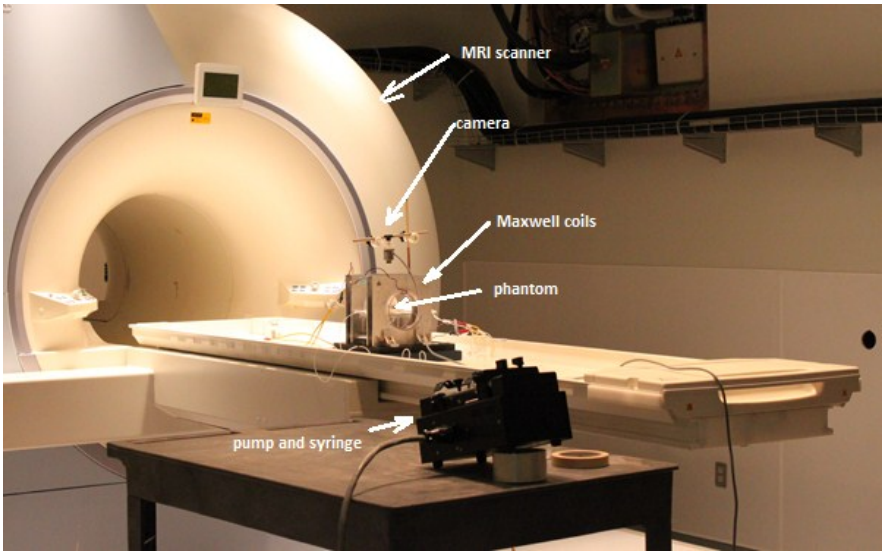


Figure 6- 1 Overview of *in-vitro* experiment hardware setup

6.2.1 Bead and microparticles

In our single particle navigation test, a chrome steel bead with 1.0 mm diameter [4, 7] and mass density of ρ_b = 8.41 kg/m^3 is used firstly instead of a tiny microparticle so that its trajectory under navigation could be better observed and feedbacked by the tracking unit. A value of

1.35x106 A/m (1.69 T) for the saturation magnetization of this kind of bead was already measured with a Gaussmeter (Walker Scientific MG-50, 10G to 10kG).

The microparticles (mode: PS-MAG-S1986) used in our aggregation navigation tests have the same properties as those used for the simulations. Thus their features are listed in Table 5.1. The microparticles have already been tested for the particular usage of magnetic intravascular navigation [26]. However, in our experiments, the interaction force between particles is also assumed to be negligible therefore is not taken into consideration.

6.2.2 Maxwell coil platform

In our experiments, a Maxwell coil platform is designed to replace the real MRI scanner system as at the present time the coils of our upgraded MRI system fail to provide adequate propulsion sequences and are thus not ready for such navigation experiments.

The setup of the Maxwell coil platform is presented in Figure 6-2.

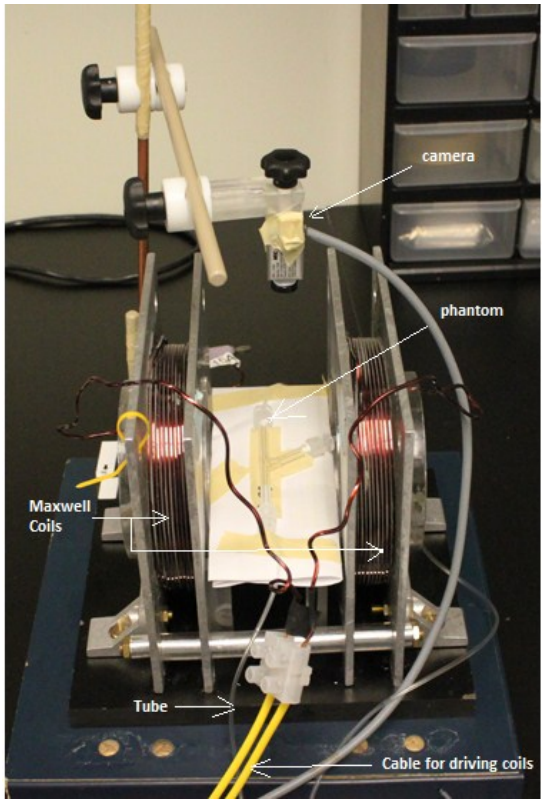


Figure 6- 2 Maxwell coil platform overview

The Maxwell coils are driven by a KIKUSHI PBX 20-10 bipolar power supply which is under control of the main fuzzy controller by remote serial communication. A high resolution camera takes images of the phantom as well as the bead or microparticle aggregations inside the phantom at a frame rate of 15 frames per second, resulting in a tracking and feedback time of nearly 100ms. A tracking program is developed to compare the frames taken by the camera and feedback the coordinates in pixels of the moving target. To better simulate the features of real MR sequences, the power supply is programmed to be forced to be shut down and maintain a zero output when the camera tracking is currently in process. Also, as programmed, the output driving sequence of the power supply to the Maxwell coils rises and falls at the same acceleration and deceleration rate as the real MR sequence to make the experiment more close to the clinical situation when a real MR scanner system is applied.

The transmission delay is estimated by observing the phase difference of two finite time period sequences given by a same command displayed on the screen of the main control computer and on the screen of an oscilloscope connected with the Maxwell coils. In our case, it takes roughly 20ms for the power supply to carry the control command and to run the programmed sequence as demanded.

6.3 Experiment Results

6.3.1 Bead navigation experiment

The first experiment carried out is to navigate a single bead inside the Y-shaped glass phantom applying the fuzzy control algorithm.

6.3.1.1 Phantom and fluid

A Y-shaped single-bifurcation phantom manufactured with glass is chosen to decrease the friction drag coefficient for the ferromagnetic bead. The start point of navigation, the waypoint and the flow direction are defined in Figure 6-3.

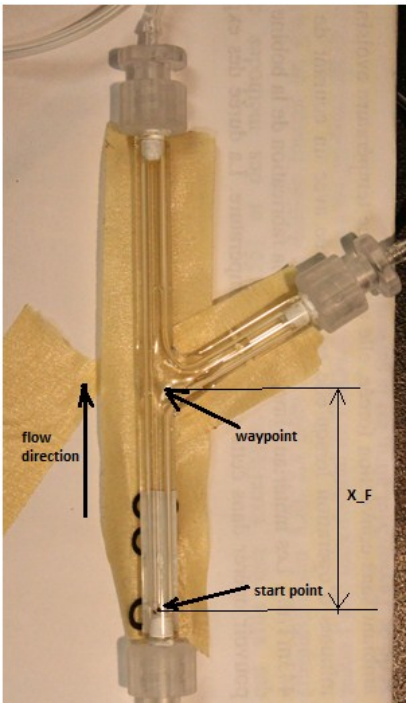


Figure 6- 3 Glass phantom used for navigation tests and definition of waypoints

The length of full navigation path *X_F* in Figure 6-3 is 35 mm. The inner diameter of the glass phantom is always 3mm along the whole *X_F*.

A Harvard PHD 22/2000 syringe pump is used to provide pulsatile flow inside the phantom through the tubes connected to the end of the phantom. The fluid is pure water whose density and viscosity are 1 g/cm^3 and 0.001003 Pa·s respectively. In our experiments, the pump is pumping at a constant speed of 35ml/min, which furnishes a poiseuille flow inside the phantom travelling at an average velocity of 0.0825 m/s in x-axis.

The Reynolds number is re-calculated to ensure the Laminar flow as well as the Poisseuille flow model. From Equation 3.2, we obtain that in this case, the Reynolds number equals to 246.76. Thus the two fluid assumptions stand.

6.3.1.2 Bead trajectory and corresponding magnetic sequences

Figure 6-4 contains the images showing the trajectory of the bead obtained from a captured video.

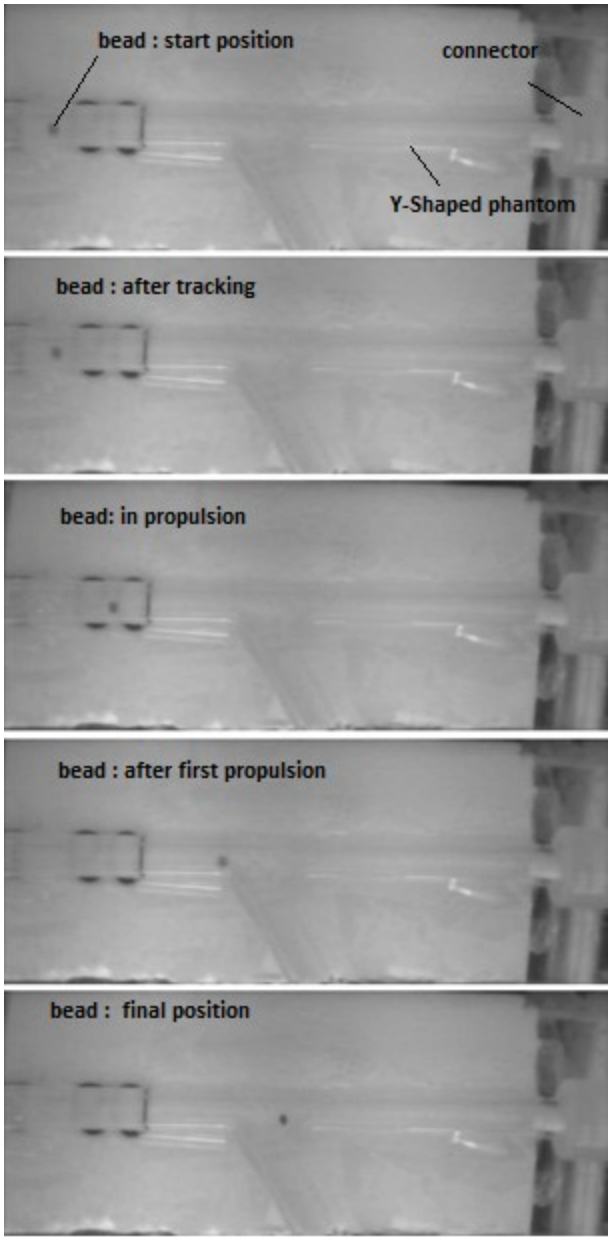


Figure 6- 4 Bead trajectory under navigation in a single bifurcation experiment

Figure 6-5 plots the trajectory on a gridded background and Figure 6-6 plots the magnetic sequences applied to the bead during the navigation process respectively.

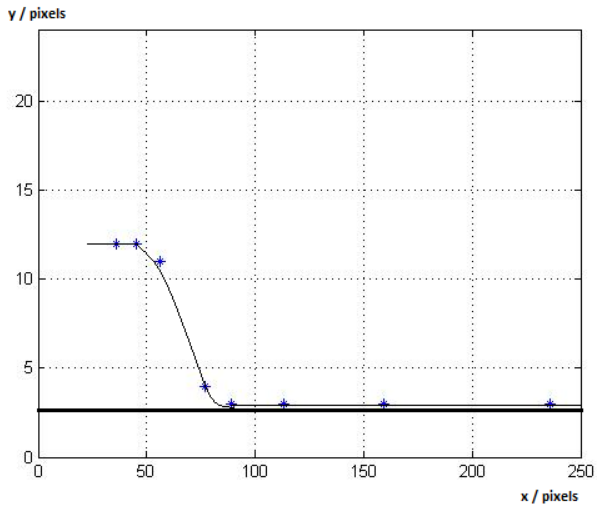


Figure 6- 5 Trajectory of bead inside the given Y-shaped phantom

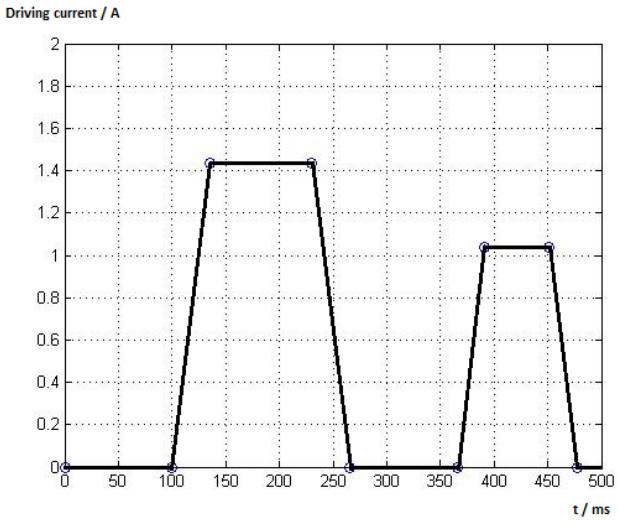


Figure 6- 6 Single bead experiment: driving current to Maxwell coils which corresponds with created magnetic gradient

From the figures it could be clearly observed that the bead has been successfully navigated to the bottom part of this Y-shaped bifurcation, although it finally stopped at the junction point and could not enter the path correctly due to the large friction force. At the beginning of the navigation, having noticed the high velocity of the bead, the controller decides to put the priority on propulsion while later the priority is put on tracking as the bead approaches the waypoint.

6.3.2 Pre-test for aggregation navigation (Common Y-shaped phantom)

Aggregation navigation pre-tests are firstly conducted with 40% weigh percentage of microparticle mix liquid with oil, using a common Y-shaped phantom made of PMMA.

Figure 6-7 shows one attempt to navigate microparticle aggregations to yield the waypoint in the right side.

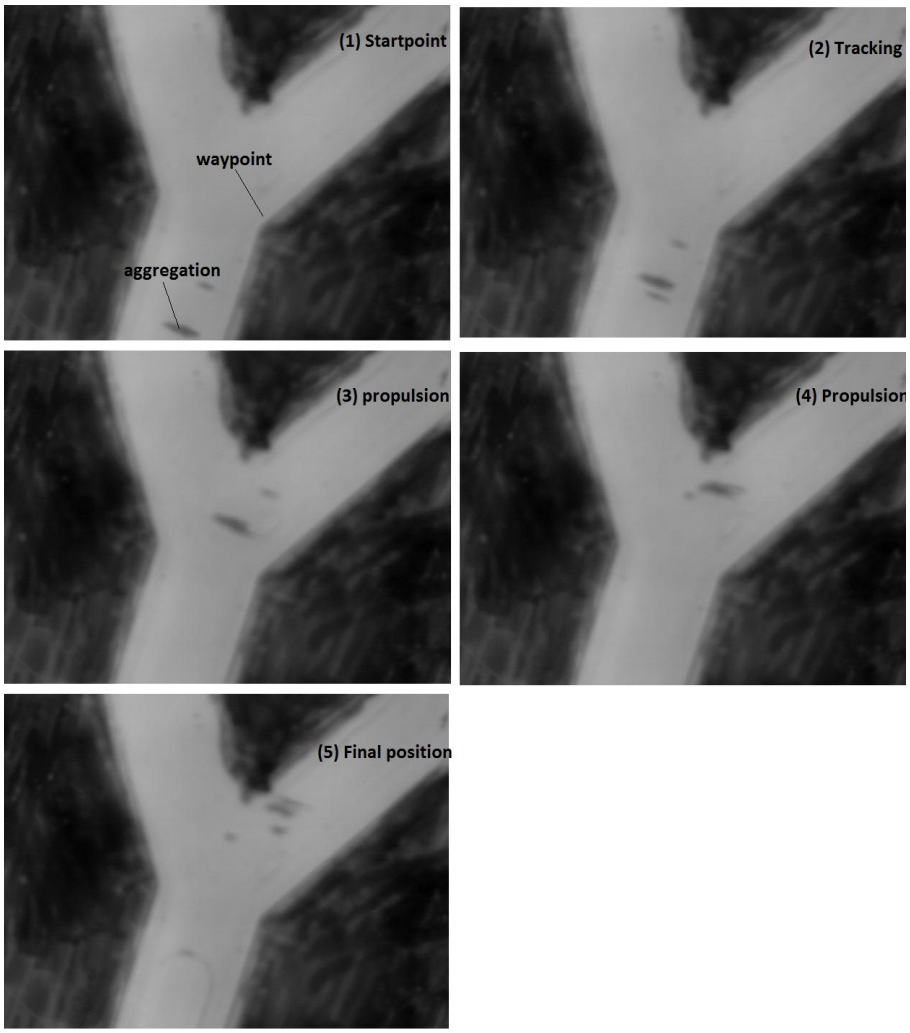


Figure 6- 7 Aggregation navigation experiment result in a Y-shaped phantom

Figure 6-8 depicts the magnetic sequence applied to the aggregation during the navigation process.

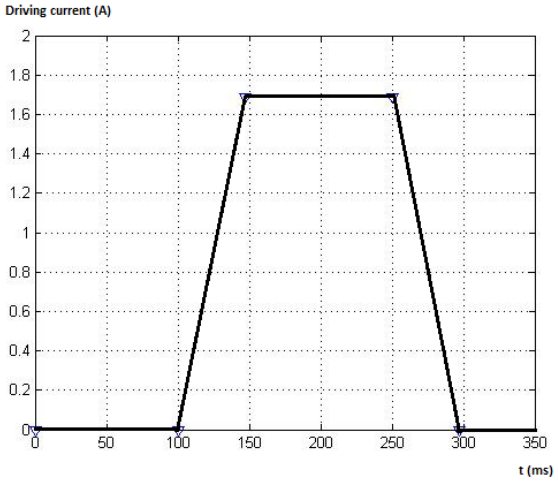


Figure 6- 8 Microparticle aggregation experiment: driving current to Maxwell coils which corresponds with created magnetic gradient

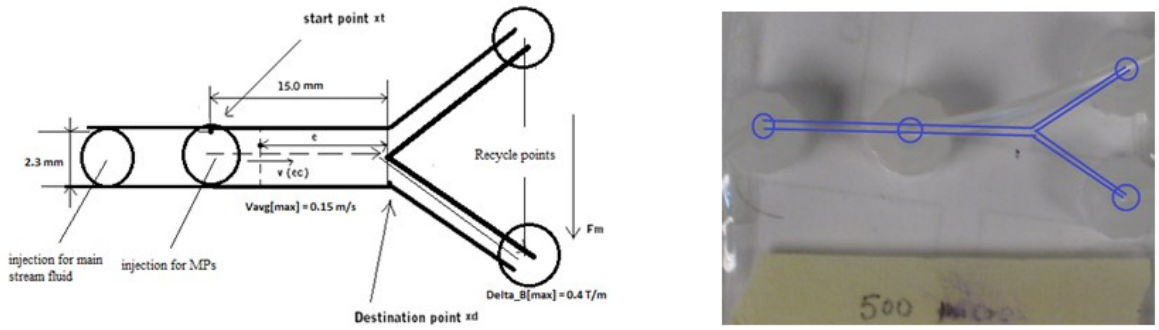
From the figures it is noticeable that most part of the microparticle aggregation has been successfully navigated to the right side of phantom, following the waypoint at the right end of boundary. However, the controller just gets only one chance to track and locate the targets. As soon as it realises the high velocity at the centre, it decides to "forget about" tracking and do full-power propulsion, as at that time there might not be enough time left for another time consuming tracking process.

6.3.3 Aggregation Navigation experiment (Simulated vascular phantom)

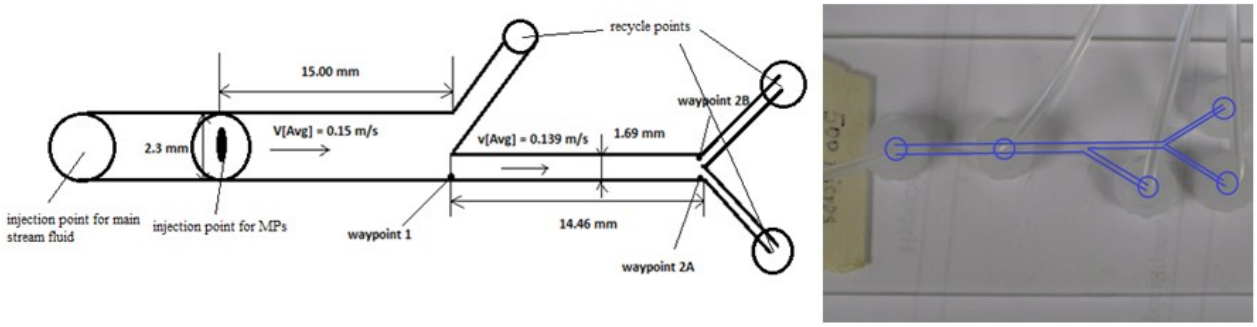
After that we have obtained the preliminary results from the navigation experiment conducted with a common Y-shaped phantom, tests with simulated vascular phantoms were decided to be performed to evaluate the robustness of the controller as well as its adaptability to the real clinic environment.

6.3.3.1 Phantom

The simulated vascular phantoms shown in Figure 6-9 are manufactured from the designs depicted in Figure 5-4 and Figure 5-10.



(a)



(b)

Figure 6- 9 Simulated vascular phantom samples made of PMMA. (a) Single-bifurcation vascular phantom; (b) Multiple-bifurcation vascular phantom.

In all the simulated vascular phantoms, the vertical cross section area is rectangular-shaped and its depth is set to be 500μm.

6.3.3.2 Fluid

The Harvard PHD 22/2000 syringe pump is used to provide pulsatile flow inside the phantom through the tubes connected to the end of the phantom. For the aggregation navigation experiments, the fluid is mixed with 60% (in volume) of pure water and 40% (in volume) of glycerine to achieve a density of 1.1 g/cm^3 and viscosity of 0.0035 Pa·s respectively. The properties of this kind of mixed liquid are considered the most similar to that of real blood [4].

To have an average horizontal fluid velocity of 0.15m/s inside the main channel of the phantom, the pump needs to pump at a constant speed of 10.35ml/min. However, due to the

limitation of the pump's power, and due that the velocity of 0.15m/s only allows the controller to track and propel once, we decided to decrease the pumping rate to 0.25ml/min so as to obtain several continuous magnetic sequences to evaluate the performance of controller. The pumping rate of 0.25ml/min furnishes a flow inside the phantom travelling at an average velocity of 0.00362 m/s in x-axis.

The Reynolds numbers have to be re-checked to verify the laminar flow assumption and Poisseuille flow model. With all the symbols in Equation 3.2 substituted with real experimental parameters, the Reynolds number is 2.52 for the first bifurcation and 2.34 for the second bifurcation. The results for both bifurcations show that the whole fluid inside the phantom is Laminar and could be applied with a Poisseuille flow model.

6.3.3.3 Microparticle aggregation injection

According to [19], tests have been done to certify that an aggregation of around 280 microparticles (PS-MAG-S1986) turn out to be small enough not to block the catheter used for delivery. Hence, with the microparticle mixture having a weight concentration of 20mg/ml, the amount for each injection is calculated to be 0.006384ml (6.384μl).

The injection device is shown in Figure 6-10. Syringe A, controlled precisely by a micro-fluid pump, is used to inject microparticles into the tube while syringe B is mainly responsible to deliver the microparticles into the channel of the phantom. Magnets are then used to hold the microparticles to form an aggregation at the start point before the experiment.

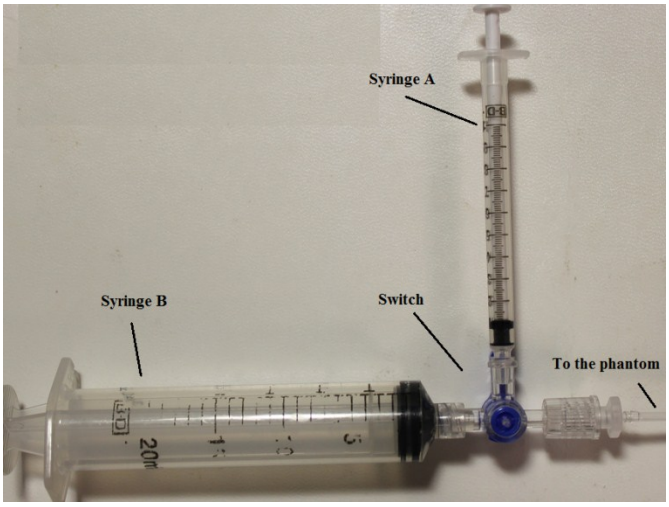


Figure 6- 10 Microparticle injection devices

A manual switch is installed to prevent the refilling from one syringe to the other while only one syringe is trying to inject into the tube.

6.3.3.4 Aggregation trajectory and corresponding magnetic sequences

Figure 6-11 contains the images showing the trajectory of the under-navigated microparticle aggregation travelling inside the simulated vascular phantom.

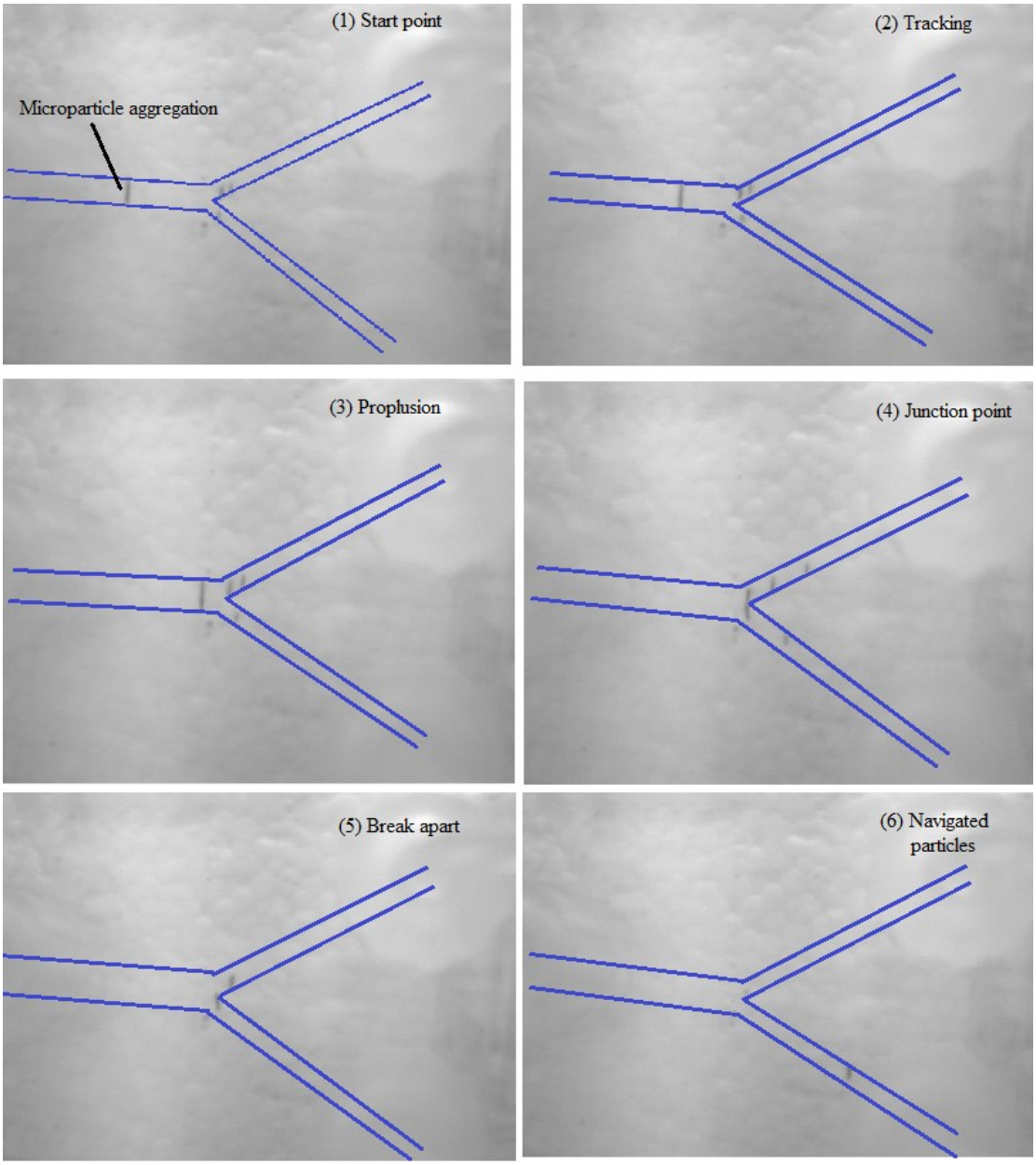


Figure 6- 11 Trajectory of microparticle aggregation under navigation

Figure 6-12 plots the coordinate information of the microparticle aggregation received by the fuzzy controller (which would then be used as the inputs to the controller and affect the fuzzy judgement process) and Figure 6-13 plots the magnetic sequences generated and applied to the aggregation according to the outputs of the controller during the navigation process respectively.

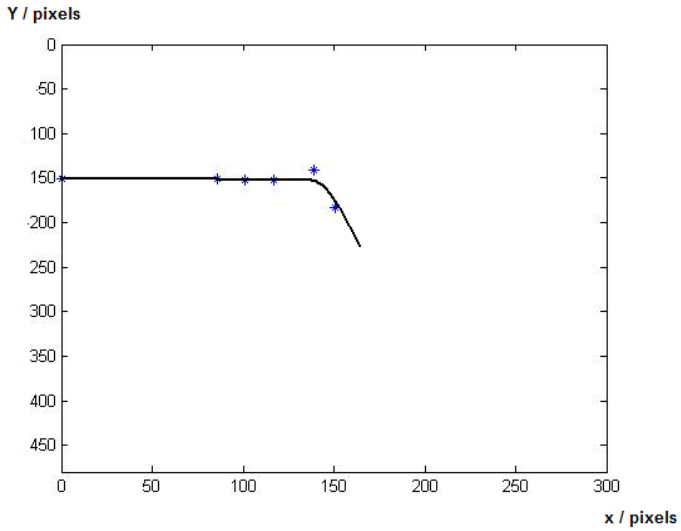


Figure 6- 12 Trajectory of microparticle aggregation (collected by the controller)

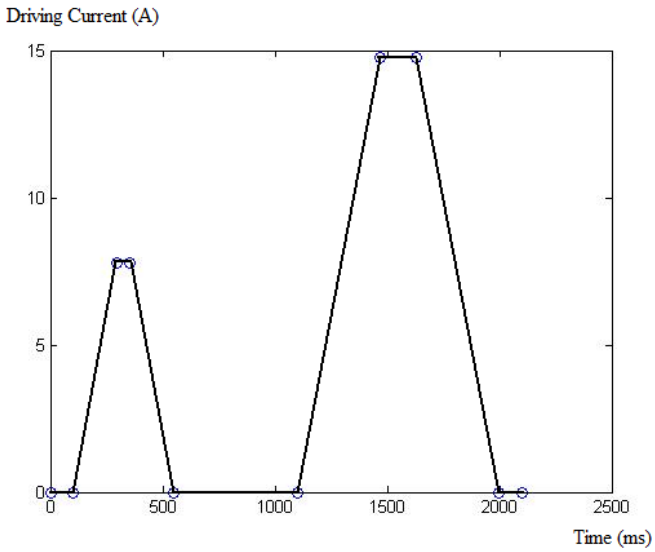


Figure 6- 13 Corresponding magnetic sequences generated and applied to navigate microparticle aggregation inside the simulated vascular phantom.

In Figure 6-13, at the time point t = 100ms, the microparticle aggregation has an x-coordinate of 0 (pixel) and an x-velocity of 0 (pixel/s), thus the first sequence (from 0ms to 549ms) is generated with a maximum driving current of 7.78 A as well as a maintaining time of 60.0ms. At the time point t = 1100ms, the x-coordinate of the aggregation is 117 (pixels) and its x-velocity is 460.9 (pixels/s). As a result, the controller gives the command to generate the second sequence (from 1000ms to 1996ms) with a maximum driving current of 14.74 A and a maintaining time of 158.97 (ms).

To better observe the motion of the aggregation under navigation, the complete tracking results captured by the camera, although only a few of them (plotted in Figure 6-12) would be submitted to the controller as inputs according to needs, are presented in Figure 6-14.

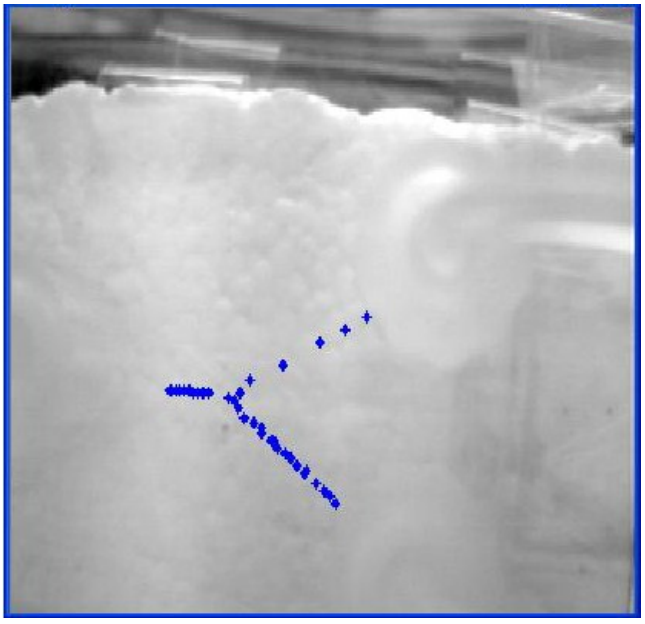


Figure 6- 14 Complete trajectory of the microparticle aggregation (collected by the camera)

From Figure 6-11 to Figure 6-14 we could see that the majority of the microparticle aggregation has been navigated correctly into the right branch of the single bifurcation phantom. As we have decreased the main flow speed, two magnetic sequences are executed completely. At the time point t = 0 ms, the aggregation is far from the junction with a zero speed, thus the controller decides to put the priority on “tracking”. While at the time point t = 1100 ms, the

aggregation is fairly close to the next waypoint with a high horizontal velocity. As a result, the controller then switch the priority from tracking to propulsion with a higher and longer magnetic sequence.

Hence, it can be concluded that with this kind of fuzzy controller, for a single-bifurcation navigation case, the rate of the quantity of microparticles following correctly the waypoints among all the microparticles in the aggregation could be raised. The adaptability of the navigation system to different situations and changes of outside environment could also be partly proved by this series of *in-vitro* navigation attempts.

Since the multiple-bifurcation phantom is still under test and is not ready for any *in-vitro* test, the multiple-bifurcation experiment has not yet been conducted.

6.4 Discussion

The related experiments presented above (with beads and microparticles) have proven that the SIMO fuzzy controller is capable of making right decisions according to different situations. Repetitive experiments have also shown that the beads and microparticle aggregations could be navigated from a random start point at most of the time as long as they could be discovered by the camera.

However, there are also significant limitations on these *in-vitro* experiments. First of all, none of these experiments were conducted in a main stream fluid with a typical average velocity of actual blood flow which is at least 0.15m/s. The first reason is that the pump fails to provide appropriate fluid for the phantoms. For example, for the glass phantom, the pump is only able to provide a fluid with an average velocity of 0.085m/s at its maximum power. For the PMMA simulated vascular phantoms, the high velocity of main stream fluid always challenges all the connection points and tubing, and make the main stream fluid non-symmetrical for the two branches of a bifurcation, as the depth of the channel of the phantom is really tiny. The other reason that the velocity has to be decreased is that, when the microparticle aggregation inside the PMMA simulated vascular phantom is travelling at a minimum velocity of 0.15m/s, the time left for control would be only dozens of microseconds, which is even insufficient to fulfill the rising process of a sequence. In that case, the effect of navigation would not be observed clearly and would not be easy to be distinguished and presented by the camera. Hence, when the controller

has to be applied in real clinical applications, facing to the real blood flow average velocity which is at least 0.15m/s, its performance would then be questioned.

When the bead is used, the mathematical model that is established in Equation 3.6 would no longer be applicable. In that case, the gravity force and buoyancy force could not be assumed to be negligible. Hence, the bead moving inside the glass phantom suffers great friction force such as the sliding friction force and the rolling friction force. As a result, in Figure 6-4, we could observe that although the bead has been successfully navigating to the bottom part of the phantom, it stops right away at the junction point as the magnetic steering force disappears after the waypoint. A suitable mathematical model has already been raised to describe the bead's motion in fluid in [4].

Moreover, the effect of heartbeat on the blood flow is not taken into consideration in these experiments. That means in reality, the blood fluid might have much more instability than a standard Poiseuille flow. As we have mentioned the difficulty of control due to the high velocity of blood flow in intravascular navigation, one possible consideration is to keep the tracking-propulsion frequency synchronized with the heartbeat rate. The tracking process only occurs when the heart releases. Thus the fluid velocity would be at its minimum and more chances for tracking could then be ensured.

CHAPTER 7: CONCLUSION

This project mainly focuses on the design and tests of controllers for propelling and navigating ferromagnetic microparticles as drug carriers along various kinds of blood vessels in human cardiovascular systems using an upgraded MRI scanner for medical interventions against diseases such as some particular types of cancers. In this work, the author proposes a new and unique approach for such intravascular navigation which is to apply a SIMO fuzzy controller. According to the preliminary simulation results and experiment results, this kind of controller could potentially be considered as an appropriate controller. In spite of certain limitations, this controller has shown its advantages in a fast responding time for a 'real-time' feedback control and its adaptability in single-bifurcation navigation as well as multiple-bifurcation navigation with varieties of complex, nonlinear, time-varying environment parameters.

In chapter 1, the literature review talks about the previous studies of such application and thus concludes its feasibility. Similar applications such as AUV control are also referred to discuss the possibility of introducing the fuzzy controller into the intravascular navigation field.

Chapter 2 states in details the purpose of the author's study as well as the difficulties of such real-time control for traditional controllers. Hence, the necessity of designing a new control algorithm is illustrated, which leads to the research into fuzzy logic area. The basic thoughts and principles for fuzzy logic are stated as adopted solutions.

Taking the mathematical model based on dynamic fluid physics established in chapter 3, chapter 4 proposes a SIMO fuzzy controller to solve the problem. For navigation targets inside blood vessels, having combined the information on their different positions and velocities, priority is put on tracking or on propulsion by the fuzzy controller according to a series of fuzzy rule sets.

Simulations in chapter 5 as well as real experiments in chapter 6 are designed and realized to evaluate the performance of the controller. Related test results have shown that for a single-bifurcation navigation case, the fuzzy controller is capable of increasing rapidly the navigation rate for microparticle aggregations by an average of 20%-30%. In the meantime, results of multiple-bifurcation simulation have shown the adaptation of the controller to the sudden environment changes or external perturbation. The fast responds time and the robustness features could be ensured. Hence, this controller is able to be transferred to be directly applied in a

multiple-bifurcation blood vessel case and clinical applications as well in the future.

However, the limitations of the fuzzy controller are also mentioned. To pursue the simplicity of control, the controller is designed to have a single input thus is lacking of enough information for deciding the future trend of navigated targets. Moreover, according to the controller stability analysis, the step response of the controller tends to have oscillations and tends to infinity with time. This is because that it seems to be not possible to find a 2-D stable point in a pulsatile flow. The high velocity in x-axis does not allow any position trimming around the waypoint and thus there is no need, and no possibility to perform a precise "point to point" servo control. Nonetheless, insurmountable control dead zones may be brought to the system due to the limited number of divided classes in fuzzy algorithm as possible constraints of the controller.

Future work may concern the improvements on the fuzzy controller to overcome the short-sighting by adding a full-scaled waypoint map of human cardiovascular system into its knowledge base. Always checking the full pre-defined trajectory and to perform a kind of "rolling optimal fuzzy control" will help the controller to give the most strategic decision which is in closest proximity to the current target condition.

REFERENCES

[1] J-B. Mathieu, and S. Martel, "Steering of aggregating magnetic microparticles using propulsion gradients coils in an MRI scanner," *Magnetic Resonance in Medicine*, vol. 63, pp. 1336-1345, May 2010.

[2] J-B. Mathieu, S. Martel, L'H. Yahia, G. Soulez, and G. Beaudoin, "Preliminary studies for using magnetic resonance imaging systems as a mean of propulsion of ferromagnetic artefacts," presented at *Canadian Conference on Electrical and Computer Engineering*, Montréal, Vol. 2, pp. 835-838, 2003.

[3] A. Chanu, and S. Martel, "Real-time software platform design for In-vivo navigation of a small ferromagnetic device in a swine carotid artery using a magnetic resonance imaging system," presented at *the 29th Annual International Conference of the IEEE-EMBS*, Lyon, pp. 6584-6587, August 2007.

[4] S. Tamaz, R. Gourdeau, A. Chanu, J-B. Mathieu, and S. Martel, "Real-time MRI-based control of a ferromagnetic core for endovascular navigation," *IEEE Transactions on Biomedical Engineering*, vol. 55, Issue 7, pp. 1854-1863, July 2008.

[5] S. Martel, "Interactive System for Medical Interventions Based on Magnetic Resonance Targeting", *Lab document*, Unpublished, 2010.

[6] K. Zhang, A. J. Krafft, R. Umathum, F. Maier, W. Semmler, and M. Bock, "Real-time MR navigation and localization of an intravascular catheter with ferromagnetic components", *Magnetic Resonance Material Physics*, Vol.23, pp. 153-163, 2010.

[7] S. Tamaz, R. Goudeau, and S. Martel, "Bidimensional MRI-based navigation System using a PID controller," presented at *the 28th Annual International Conference of the IEEE-EMBS*, New York city, pp. 4424-4427, August-September 2006.

[8] S. Tamaz, and S. Martel, "Impact of the MRI-based navigation system constraints on the step response using a PID controller," presented at *the 27th Annual International Conference of the IEEE-EMBS*, Shanghai, August-September 2005.

[9] A. Brown, and R. Garcia, "Concepts and Validation of a Small-Scale Rotorcraft Proportional Integral Derivative (PID) Controller in a Unique Simulation Environment", *Intelligent Robot System*, vol.54, pp. 511-532, 2009.

[10] S. Zhao, J. Yuh, and S.K. Choi, "Adaptive DOB control for AUVs", in *Proc. 2004 IEEE –Robot. Auto. Soc. Annu, Int. Conf.*, New Orleans, April-May 2004, pp. 4899 -4905.

[11] T. Tian, J. Liu, and K. Liu, "Application Research of Self-adaptive PID control Used for AUV", Computer Information, vol.3-1, no.24, pp.4-7, 2008.

[12] J.L. Zhang, J.W. An, and M.N. Wang, "Simulation for Fuzzy-Control System Based on Simulink and C/C++ Mixed Programme Technique", Journal of System Simulation, Vol.16, pp. 2774-2776, 2010.

[13] L. Arcese, M. Fruchard, and A. Ferreira, "Nonlinear modelling and robust controller-observer for a magnetic microrobot in a fluidic environment using MRI gradients", *The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, St.Louis, Octorber 11-15, 2009, pp.534-539.

[14] G.Q. Zeng, J.A. Hu, D.Wang, and C.L. Liu, *Fuzzy Control Algorithm and Applications in Engineering*. Huazhong Univsersity of Science and Technology Press, Wuhan, 2006.

[15] A. Chanu, S. Martel, and G.Beaudoin, "Real-time Magnetic Resonance Gradient-based Propulsion of a Wireless Microdevice Using Pre-Acquired Roadmap and Dedicated Software Architecture", presented at *27th IEEE-EMBS Annual International Conference of the Engineering in Medicin and Biology Society*, pp.5190-5193, shanghai, China, Sept.1-4, 2005.

[16] M. Zamir, *The Physics of Pulsatile Flow*, Springer-Verlag New York, Inc. pp. 39-47, 2000.

[17] In Wikipedia: The free encyclopedia, [Online]. Available: http://en.wikipedia.org/wiki/Hagen%E2%80%93Poiseuille_equation, 2011.

[18] C. C. Lee, "Fuzzy Logic in control systerm", *IEEE Trans*, vol.20, 1990.

[19] G. Bringout, "Actionneur pour le guidage de micro-particules magnétiques et thérapeutiques dans le système vasculaire : bobines de gradients pulsés pour des essais pré-clinique", *Thesis*, École polytechnique de Montréal, Unpublished, 2011.

[20] R.W. Barber, and D.R. Emerson, "Optimal design of microfluidic networks using biologically inspired principles", *Microfluid Nanofluid*, vol.4, pp.179-191, 2008.

[21] C.A. Basciano, C. Kleinstreuer, A.S. Kennedy, W.A. Dezarn, and E. Childress, "Computer Modelling of Controlled Microsphere Release and Targeting in a Representative Hepatic Artery System", *Annals of Biomedical Engineering*, Vol.38, No.5, pp.1862-1879, May 2010.

[22] B. Lee, and J.Y. Yoo, "Droplet bistability and its application to droplet control", *Microfluid Nanofluid*, published online, 22 June 2011.

[23] C.N. Baroud, and H. Willaime, "Multiphase flows in microfluidics", *Comptes Rendus Physique*, vol.5, pp.547-555, 2004.

[24] N-J. Darton, A-J. Sederman, A. Lonescu, C. Ducati, R-C. Darton, and L-F. Gladden, "Manipulation and tracking of superparamagnetic nanoparticles using MRI, " *Nanotechnology*, vol.19, Issue 39, Oct 2008.

[25] J-B Mathieu, and S. Martel, "Aggregation of magnetic microparticles in the context of targeted therapies actuated by a magnetic resonance imaging system", *Journal of Appied Physics*, vol. 106, Issue 4, 2009.

[26] P.Pouponneau, J-C, Leroux, S. Martel, "Magnetic nanoparticles encapsulated into biodegradable microparticles steered with an upgraded magnetic resonance imaging system for tumor chemoebolization", *Biomaterials*, Vol.30, pp.6327-6332, 2009.

ANNEXE 1 – **Matlab program for fuzzy reasoning**.

INPUT: E,EC, OUTPUT: G, same for E+EC->T.

```
clc;
clear;

%%membership function of the inputs and outputs
E = [1,0.8,0.4,0.1,0,0,0,0,0,0,0,0,0;
    0.4,0.8,1,0.8,0.4,0.1,0,0,0,0,0,0,0;
    0,0,0,0.3,1,0.3,0,0,0,0,0,0,0;
    0,0,0,0,0,0.2,1,0.2,0,0,0,0,0;
    0,0,0,0,0,0,0,0.3,1.0,0.3,0,0,0;
    0,0,0,0,0,0,0,0.1,0.4,0.8,1,0.8,0.4;
    0,0,0,0,0,0,0,0,0,0.1,0.4,0.8,1.0]%%7*13

EC =[1,0.2,0,0,0,0,0,0,0,0,0,0,0;
    0,0.2,1,0.2,0,0,0,0,0,0,0,0,0;
    0,0,0,0.4,1,0.4,0,0,0,0,0,0,0;
    0,0,0,0,0.2,0.6,1,0.6,0.2,0,0,0,0;
    0,0,0,0,0,0,0,0.4,1,0.4,0,0,0;
    0,0,0,0,0,0,0,0,0,0.2,1,0.2,0;
    0,0,0,0,0,0,0,0,0,0,0,0.2,1]%%7*13

U=[1,1,0.7,0.3,0,0,0,0,0,0,0,0,0,0,0;
    0,0.3,0.7,1,0.7,0.3,0,0,0,0,0,0,0,0,0;
    0,0,0,0,0.3,0.7,1,0,0,0,0,0,0,0,0;
    0,0,0,0,0,0,0,1,0,0,0,0,0,0,0;
    0,0,0,0,0,0,0,0,1,0.7,0.3,0,0,0,0;
    0,0,0,0,0,0,0,0,0,0.3,0.7,1,0.7,0.3,0;
    0,0,0,0,0,0,0,0,0,0,0,0.3,0.7,1,1]%%7*15

%%fuzzy rule table
rulelist=  [2,3,3,4,4,4,4;
            3,4,4,4,4,5,5;
            3,4,4,5,5,6,6;
            4,5,6,6,6,6,7;
            4,5,6,6,6,6,7;
            4,5,5,6,6,7,7;
            4,4,5,5,6,7,7]%%7*7

%%-----------------------------------------------------------------------
for iii=1:13              %E fuzzy value loop
   for jjj=1:13          %EC fuzzy value loop
         for ii=1:7        %E fuzzy regulation loop
             for jj=1:7    %EC fuzzy regulation loop
                 A_rulelist = rulelist(ii,jj); %refer to fuzzy table
                 %************get C1A'***************
                 A = E(ii,:);       %A'
                 C_A = U(A_rulelist,:);     %Ci
                 for i=1:13     %get R1A
                     for j=1:15
                         if(A(i) > C_A(1,j))
                             Ra(i,j) = C_A(1,j);
                         else
```

```
                        Ra(i,j) = A(i);
                    end
                end
            end
            AA = zeros(1,13);
%AA=A'  AA=[1,0,0,0,0,0,0,0,0,0,0,0,0]; single point fuzzy set

            AA(1,iii) = 1;

            for i=1:15      %intersection operation
                for j=1:13
                    if(AA(j) > Ra(j,i))
                        A_qux(j,i) = Ra(j,i);
                    else
                        A_qux(j,i) = AA(j);
                    end
                end
            end
            for i=1:15     %union operation, get CiA
                max = A_qux(1,i);
                for j=1:13
                    if(max < A_qux(j,i))
                        max = A_qux(j,i);
                    end
                end
                CiA(i) = max;
            end
            %********CiA finished*********
            %*********get CiB'*************

            B = EC(jj,:);      %B'
            C_B = U(A_rulelist,:);     %Ci

            for i=1:13     %get R1B
                for j=1:15
                    if(B(1,i) > C_B(1,j))
                        Rb(i,j) = C_B(1,j);
                    else
                        Rb(i,j) = B(1,i);
                    end
                end
            end

            BB = zeros(1,13);
%BB=B'  BB=[1,0,0,0,0,0,0,0,0,0,0,0,0];  single point fuzzy set

            BB(1,jjj) = 1;

            for i=1:15      %intersection operation
                for j=1:13
                    if(BB(j) > Rb(j,i))
                        B_qux(j,i) = Rb(j,i);
                    else
                        B_qux(j,i) = BB(j);
                    end
```

```
                    end
                end

                for i=1:15     %union operation,get CiB
                    max = B_qux(1,i);
                    for j=1:13
                        if(max < B_qux(j,i))
                            max = B_qux(j,i);
                        end
                    end
                    CiB(i) = max;
                end
                %*******CiB' finished****
                %*******Ci'=CiA' intersect CiB'*********
                for i=1:15
                    if CiA(i) > CiB(i)
                        Ci(i) = CiB(i);
                    else
                        Ci(i) = CiA(i);
                    end
                end
                %*******Ci'finished***********
                C((ii-1)*7+jj,:) = Ci;
% store Ci to C, which is a matrix of 56*13
            end
        end
        %Ui=C'=C1' union C2'union ……union C49'
        for i=1:15
            max = C(1,i);
            for j=1:49
                if(max < C(j,i))
                    max = C(j,i);
                end
            end
            Ui(i) = max;
        end
        %clarity method : weighted average
        sum_molecular= 0;
        sum_denominator = 0;
        for i=1:15
            sum_molecular = sum_molecular + (i-8)*Ui(i);
            sum_denominator = sum_denominator + Ui(i);
        end
        core = sum_molecular /sum_denominator;
        U_control(iii,jjj) = core;
    end
end
U_control = round(U_control.*10000)/10000;  %keep 2 float bits
```

ANNEXE 2 – C++ Simulation platform

```
/*****************************************************************
NAME : fuzzy_header.h
FUNCTION: to define the constant parameters used in the simulation
AUTHOR: Ke PENG
      Laboratory of NanoRobotics, Ecole Polytechnique de Montreal
VERSION: 2.01
EDIT HISTORY: 2010/11/19 File created.
              2010/11/23 Constants declaration
              2010/12/03 Constants and variables declaration. Fill in with numbers.
              2010/12/20 Change all the parameters of particle BM247 to NanoDePierre.
Version 1.1 finished.
              2010/12/20 Change sequence parameters. Version 1.11 finished.
              2011/03/22 Modify magnetic force fuzzy table. Add another fuzzy table for
maintaining time. Version 2.0 finished.
              2011/03/24 Change rising_time&falling_time from 26.7 to 40. Change p_ms from
341892 ro 401440. Version 2.01.
*****************************************************************/

#ifndef _fuzzy_header_h_
#define _fuzzy_header_h_

/*fuzzy table description*/
const double fuzzy_table[13][13] =
{{-4.20,-3.50,-1.60,-1.63,-1.60,-1.44, 0.00, 0.00, 0.00, 1.40, 1.17, 1.75, 1.17},
  {-3.73,-3.50,-1.39,-1.63,-1.39,-1.44, 0.00, 0.00, 0.00, 1.40, 1.39, 1.75, 1.39},
  {-3.28,-3.50,-1.17,-1.63,-1.17,-1.44, 0.00, 0.00, 0.00, 1.40, 1.60, 1.75, 1.60},
  {-2.56,-2.86,-0.64,-1.00, 0.41, 0.78, 1.24, 1.40, 1.24, 2.95, 3.17, 3.77, 3.17},
  {-1.93,-1.75, 0.00, 0.00, 0.88, 1.66, 1.60, 1.66, 1.60, 2.83, 4.20, 4.08, 4.20},
  {-1.91, 0.00, 1.56, 3.27, 3.27, 3.11, 3.50, 3.50, 3.50, 3.50, 4.45, 4.94, 4.86},
  { 0.00, 1.75, 1.93, 4.32, 4.67, 4.67, 4.67, 4.67, 4.67, 4.67, 4.67, 5.25, 6.88},
  { 0.00, 1.75, 2.33, 4.25, 4.45, 4.45, 4.67, 4.67, 4.67, 4.89, 4.89, 5.25, 6.42},
  { 0.00, 1.75, 1.93, 3.96, 4.20, 4.10, 4.67, 4.67, 4.67, 5.02, 5.13, 5.25, 6.88},
  { 0.00, 1.75, 1.90, 3.67, 3.28, 4.10, 4.55, 4.53, 4.55, 5.02, 6.05, 5.25, 6.78},
  { 0.00, 1.75, 1.60, 1.88, 2.83, 4.10, 4.20, 4.10, 4.45, 5.02, 6.88, 6.42, 6.88},
  { 0.00, 1.75, 1.39, 1.88, 2.97, 3.82, 3.73, 3.82, 4.43, 5.02, 6.78, 6.42, 6.78},
  { 0.00, 1.75, 1.17, 1.88, 2.83, 3.63, 3.82, 3.63, 4.45, 5.02, 6.88, 6.42, 6.88}};

const double fuzzy_maintain[13][13] =
{{-1.71,-2.33,-2.42,-3.50,-4.89,-5.71 -6.88,-6.67,-6.88,-6.53,-6.88,-6.42,-6.88},
  {-2.33,-2.33,-2.92,-3.50,-4.90,-5.71,-6.78,-6.67,-6.78,-6.53,-6.78,-6.42,-6.78},
  {-2.96,-2.33,-3.42,-3.50,-4.89,-5.71,-6.88,-6.67,-6.88,-6.53,-6.88,-6.42,-6.88},
  {-4.37,-3.79,-4.56,-4.52,-5.01,-5.71,-6.78,-6.67,-6.78,-5.77,-6.05,-2.92,-4.30},
  {-6.11,-5.25,-6.11,-5.38,-6.11,-5.71,-6.88,-6.67,-6.88,-5.77,-5.13,-2.92,-2.18},
```

```
  {-5.92,-5.72,-5.92,-5.92,-5.92,-5.92,-6.42,-5.54,-5.54,-3.50,-3.06,-2.06,-0.89},
  {-6.88,-6.42,-6.88,-6.53,-6.88,-6.67,-6.89,-5.71,-4.89,-3.05, 0.00, 0.58, 1.17},
 {-6.42,-6.42,-6.42,-6.42,-5.54,-5.25,-5.08,-5.08,-5.08,-1.46, 1.52, 2.64, 3.55},
 {-6.88,-6.42,-6.88,-6.53,-6.42,-5.23,-4.67,-4.67,-4.67,-1.10, 2.42, 3.50, 5.13},
 {-6.78,-6.42,-6.78,-6.53,-6.26,-5.23,-4.67,-4.67,-4.67,-0.83, 3.65, 3.50, 6.05},
 {-6.88,-6.42,-6.88,-6.53,-6.42,-5.23,-4.67,-4.67,-4.67,-0.70, 5.13, 5.25, 6.88},
 {-6.78,-6.42,-6.78,-6.53,-6.26,-5.23,-4.67,-4.67,-4.67,-0.70, 5.63, 5.25, 6.78},
 {-6.88,-6.42,-6.88,-6.53,-6.42,-5.23,-4.67,-4.67,-4.67,-0.70, 6.11, 5.25, 6.88}};

/*fuzzy constants*/
const double x_p_reg = 6.0;   //no unit
const double x_v_reg = 6.0;   //no unit
const double DeltaB__y_reg = 7.0;    //no unit
const double t_maintain_reg = 7.0;    //no unit
/*sequence fuzzy constants*/
const double t_fuzzy_full = 0.01;       //unit: s
/*position constants*/
const double x_positive_full_1 = 0.015;       //unit: m
const double y_positive_full_1 = 0.0023;      //unit: m
const double y_positive_centre_1 = 0.00115;           //unit: m
const double x_positive_full_2 = 0.01446;
const double y_positive_full_2 = 0.00169;
const double y_positive_centre_2 = 0.000845;
/*flow constants*/
const double x_v_avg_max_1 = 0.15;          //unit: m/s
const double x_v_max_max_1 = 0.30;          //unit: m/s, changed
const double x_v_avg_max_2 = 0.139;
const double x_v_max_max_2 = 0.278;
/*particle constants*/
const double p_ms = 401440;           //unit: A/m, changed
const double p_viscosity = 0.0035;              //unit: Pa*s
const double p_m = 0.00000000042;               //unit: kg
const double p_volum = 0.000000000000102;                   //unit: m^3, changed
const double p_radius = 0.000058;                       //unit: m, changed
/*sequence constants*/
const double DeltaB_y_full = 0.4;               //unit: T/m
const double fm_y_rising = 40.0;                //unit: T/m/s, changed
const double fm_y_falling = -40.0;              //unit: T/m/s, changed
const double t_tracking = 0.03;                 //unit: s
const double t_maintaining = 0.01;              //unit: s, changed
/*time constants*/
const double time_slot = 0.0000001;             //unit: s
/*other constants*/
const double pi = 3.141592654;                  //no unit
/*function definition*/
double fuzzy_decide(double x_position, double x_v);
double t_maintain_fuzzy_decide(double x_position, double x_v);
```

```
int single_time_slot(double DeltaB);
int fm_maintained(double DeltaB_mt, double t_mt);
int fm_tracking(double DeltaB_tr, double t_tr);
int fm_rising(double DeltaB_rs);
int fm_falling(double DeltaB_fl);

/*paticle variables*/
extern double p_x;                      //unit: m
extern double p_y;                      //unit: m
extern double p_v_x;                    //unit: m/s
extern double p_v_y;                    //unit: m/s
extern double DeltaB_mt;
extern double DeltaB_tr;
extern double DeltaB_rs;
extern double DeltaB_fl;
extern double t_maintain_fuzzy_result;        //unit: s

#endif
```

```
/*****************************************************************
NAME : time_slot_operation.cpp
FUNCTION: to simulation how the particle will reacte according to the outside factors in single time slot.
AUTHOR: Ke PENG
      Laboratory of NanoRobotics, Ecole Polytechnique de Montreal
VERSION: 2.0
EDIT HISTORY: 2010/12/03 File created.
                  2010/12/09 return value: void -> int. Version 1.0 finished.
                  2011/03/22 Add horizontal magnetic force. Maintain the angle between the two megnetic forces to be 45 degrees. The conditions (if no outside force is applied) in x-axie and y-axie are similar. Version 2.0 finished.
*****************************************************************/
#include "fuzzy_header.h"
#include "math.h"
#include "iostream"
using namespace std;

int single_time_slot(double DeltaB)
{
        double p_x_temp;
        double p_y_temp;
        double p_v_x_temp;
        double p_v_y_temp;

        double p_a_y;
```

```
double p_y_ratio;

double p_v_x_mag;

double y_positive_centre;
double x_v_max_max;
double y_positive_full;

p_x_temp = p_x;
p_y_temp = p_y;
p_v_x_temp = p_v_x;
p_v_y_temp = p_v_y;

if(p_x_temp >= 0)
{
        y_positive_centre = y_positive_centre_1;
        x_v_max_max = x_v_max_max_1;
        y_positive_full = y_positive_full_1;
}
else
{
        y_positive_centre = y_positive_centre_2;
        x_v_max_max = x_v_max_max_2;
        y_positive_full = y_positive_full_2;
}

if(DeltaB == 0)
                                                        //No Mg force applied
{
        p_v_y_temp = 0;
}
else
{
        /*First calculate the velocity and the distance covered in y-axies.*/
        p_a_y = (p_ms*DeltaB*p_volum + (-
1)*6.0*pi*p_viscosity*p_radius*p_v_y_temp)/p_m;          //a=(Fm+f)/m
        p_y_temp = p_y_temp + p_v_y_temp*time_slot + p_a_y*time_slot*time_slot/2;
                //s=s+v0t+at^2/2
        p_v_y_temp = p_v_y_temp + p_a_y*time_slot;
                                                //v=v0+at
}

p_v_x_mag = fabs(p_v_y_temp);

/*Then calculate the velocity and the distance covered in x-axies.*/
p_y_ratio = (p_y_temp + y_positive_centre) / y_positive_centre;
```

```
        p_v_x_temp = x_v_max_max * (1 - (p_y_ratio*p_y_ratio)) - p_v_x_mag;

        p_x_temp = p_x_temp - (p_v_x + p_v_x_temp) * time_slot / 2;

        if((p_x_temp < 0)&&(p_x >= 0))
        {
                if(p_y_temp < (0 - y_positive_full_2))
                        return -2;
        }

        /*Exception Handling*/

        if(p_x_temp >= 0)
        {
                if(p_y_temp < (p_radius - y_positive_full_1))
                        p_y_temp = p_radius - y_positive_full_1;
                if(p_y_temp > (0 - p_radius))
                        p_y_temp = 0 - p_radius;
        }
        else
        {
                if(p_y_temp < (p_radius - y_positive_full_2))
                        p_y_temp = p_radius - y_positive_full_2;
                if(p_y_temp > (0 - p_radius))
                        p_y_temp = 0 - p_radius;
        }

        if(p_x_temp < (0 - x_positive_full_2))
        {
                p_x_temp = (0 - x_positive_full_2);
                return -1;
                                                                //simulation ends.
        }

        p_x = p_x_temp;
        p_y = p_y_temp;
        p_v_x = p_v_x_temp;
        p_v_y = p_v_y_temp;

        return 0;       //normal exit
}
```

```
/*****************************************************************
```

NAME : tracking_process.cpp
FUNCTION: to simulate while the magnetic force is set to track the particles in a nearly-zero value

```
AUTHOR: Ke PENG
      Laboratory of NanoRobotics, Ecole Polytechnique de Montreal
VERSION: 1.0
EDIT HISTORY: 2010/12/09 File created.
                2010/12/09 File version 1.0 finished.
**********************************************************************/
#include "fuzzy_header.h"
#include "iostream"
using namespace std;

int fm_tracking(double DeltaB_tr, double t_tr)
{
        int times;
        times = int(t_tr / time_slot);

        int i;
        int single_result = 0;

        for(i=0;i<times;i++)
        {
                single_result = single_time_slot(DeltaB_tr);

                if(single_result == 0)
                        continue;
                else if(single_result == -1)
                {
                        //t_ending = (i+1)*time_slot;
                        //cout<<"ending time is"<<t_ending<<endl;
                        return -1;
                }
                else if(single_result == -2)
                        return -2;
                else
                {
                        cout<<endl<<"ERROR! Tracking process problem!"<<endl;
                        exit(-100);
                }
        }
        return 0;
}
```

```
/**********************************************************************
NAME : rising_process.cpp
FUNCTION: to simulate while the magnetic force is rising to a certain value for propulsion
AUTHOR: Ke PENG
      Laboratory of NanoRobotics, Ecole Polytechnique de Montreal
VERSION: 1.0
```

```
EDIT HISTORY: 2010/12/09 File created.
              2010/12/10 Version 1.0 Finished.
**********************************************************************/
#include "fuzzy_header.h"
#include "iostream"
using namespace std;

int fm_rising(double DeltaB_rs)
{
        double DeltaB_it = 0;

        int times;
        times = int((DeltaB_rs/fm_y_rising) / time_slot);

        int i;
        int single_result = 0;

        for(i=0;i<times;i++)
        {
                DeltaB_it = DeltaB_it + fm_y_rising*time_slot;
                single_result = single_time_slot(DeltaB_it);

                if(single_result == 0)
                        continue;
                else if(single_result == -1)
                        return -1;
                else if(single_result == -2)
                        return -2;
                else
                {
                        cout<<endl<<"ERROR! Rising process problem!"<<endl;
                        exit(-100);
                }
        }
        return 0;
}
```

```
/**********************************************************************
NAME : maintain_process.cpp
FUNCTION: to simulate while the magnetic force is maintained in a non-zero value
AUTHOR: Ke PENG
      Laboratory of NanoRobotics, Ecole Polytechnique de Montreal
VERSION: 1.0
EDIT HISTORY: 2010/12/09 File created.
              2010/12/09 File version 1.0 finished.
**********************************************************************/
```

```
#include "fuzzy_header.h"
#include "iostream"
using namespace std;

int fm_maintained(double DeltaB_mt, double t_mt)
{
	int times;
	times = int(t_mt / time_slot);

	int i;
	int single_result = 0;

	for(i=0;i<times;i++)
	{
		single_result = single_time_slot(DeltaB_mt);

		if(single_result == 0)
			continue;
		else if(single_result == -1)
			return -1;
		else if(single_result == -2)
			return -2;
		else
		{
			cout<<endl<<"ERROR! Maintain process problem!"<<endl;
			exit(-100);
		}
	}
	return 0;
}
```

```
/*****************************************************************
NAME : falling_process.cpp
FUNCTION: to simulate while the magnetic force is falling from a certain value for propulsion to 0;
AUTHOR: Ke PENG
	Laboratory of NanoRobotics, Ecole Polytechnique de Montreal
VERSION: 1.1
EDIT HISTORY: 2010/12/10 File created.
		2010/12/10 Version 1.0 Finished.
		2010/12/20 Bug correction.(DeltaB_it = DeltaB_fl).
Version 1.1 finished.
*****************************************************************/
#include "fuzzy_header.h"
#include "iostream"
using namespace std;
```

```
int fm_falling(double DeltaB_fl)
{
	double DeltaB_it = DeltaB_fl;
	int times;
	times = int(((-1)*DeltaB_fl/fm_y_falling) / time_slot);

	if(times <= 0)
		times = -times;

	int i;
	int single_result = 0;

	for(i=0;i<times;i++)
	{
		if(DeltaB_fl >= 0)
			DeltaB_it = DeltaB_it + fm_y_falling*time_slot;
		else
			DeltaB_it = DeltaB_it - fm_y_falling*time_slot;

		single_result = single_time_slot(DeltaB_it);

		if(single_result == 0)
			continue;
		else if(single_result == -1)
			return -1;
		else if(single_result == -2)
			return -2;
		else
		{
			cout<<endl<<"ERROR! Falling process problem!"<<endl;
			exit(-100);
		}
	}
	return 0;
}
```

```
/*************************************************************
NAME : fuzzy_decide.cpp
FUNCTION: to give the DeltaB with x_position and x_v by using fuzzy decide process
AUTHOR: Ke PENG
      Laboratory of NanoRobotics, Ecole Polytechnique de Montreal
VERSION: 2.0
EDIT HISTORY: 2010/11/23 File created.
	2010/11/23 Version 1.0 finished. Test passed.
	2010/12/03 Version 1.1 finished. Replace fm with DeltaB. Test passed.
	2010/12/20 Version 1.11 finished. Replace maximum distance with 2*.
	2011/03/22 Version 2.0 finished. Change back the maximum distance to 1*.
```

```
**********************************************************************/

#include "fuzzy_header.h"

double fuzzy_decide(double x_p, double x_v)
{
	double DeltaB = 0;
	int x_p_decide = 0;
	int x_v_decide = 0;
	double x_p_result = 0;
	double x_v_result = 0;

	if(x_p >= 0)
	{
		x_p_result = x_p * x_p_reg / x_positive_full_1;
		x_v_result = x_v * x_v_reg / x_v_max_max_1;
	}
	else
	{
		x_p_result = (x_p + x_positive_full_2) * x_p_reg / x_positive_full_2;
		x_v_result = x_v * x_v_reg / x_v_max_max_2;
	}

	/*linear transformation*/

	int i;

	/*To get the fuzzy decide input value of x_postion*/
	if(x_p_result >= x_p_reg - 0.5)
		x_p_decide = (int)x_p_reg;
	else if(x_p_result <= 0.5 - x_p_reg)
		x_p_decide = (int)(0 - x_p_reg);
	else
	{
		for(i = 0;i<=2*x_p_reg-2;i++)
		{
			if((x_p_result >= (x_p_reg-i-1.5))&&(x_p_result <= (x_p_reg-i-0.5)))
				x_p_decide = (int)(x_p_reg - i);
		}
	}

	/*To get the fuzzy decide input value of x_velocity*/
	if(x_v_result >= x_v_reg - 0.5)
		x_v_decide = (int)x_v_reg;
	else if(x_v_result <= 0.5 - x_v_reg)
		x_v_decide = (int)(0 - x_v_reg);
	else
```

```
	{
		for(i = 0;i<=2*x_v_reg-2;i++)
		{
			if((x_v_result >= (x_v_reg-i-1.5))&&(x_v_result <= (x_v_reg-i-0.5)))
				x_v_decide = (int)(x_v_reg - i);
		}
	}

	/*Two inputs ready*/

	/*fuzzy decide, return fm*/
	DeltaB = fuzzy_table[x_p_decide + (int)(x_p_reg)][x_v_decide + (int)(x_v_reg)];

	if(x_p < 0)
		DeltaB = - DeltaB;

	return DeltaB * DeltaB_y_full / DeltaB__y_reg;
}
```

```
/*****************************************************************
NAME : t_maintain_fuzzy_decide.cpp
FUNCTION: to give the maintaining time with x_position and x_v by using fuzzy decide process
AUTHOR: Ke PENG
	Laboratory of NanoRobotics, Ecole Polytechnique de Montreal
VERSION: 1.0
EDIT HISTORY: 2011/03/22 File created.Version 1.0 finished. Test passed.
*****************************************************************/
#include "fuzzy_header.h"

double t_maintain_fuzzy_decide(double x_p, double x_v)
{
	double t_maintain = 0;
	int x_p_decide = 0;
	int x_v_decide = 0;
	double x_p_result = 0;
	double x_v_result = 0;

	if(x_p >= 0)
	{
		x_p_result = x_p * x_p_reg / x_positive_full_1;
		x_v_result = x_v * x_v_reg / x_v_max_max_1;
	}
	else
	{
		x_p_result = (x_p + x_positive_full_2) * x_p_reg / x_positive_full_2;
		x_v_result = x_v * x_v_reg / x_v_max_max_2;
	}
```

```
	/*linear transformation*/

	int i;

	/*To get the fuzzy decide input value of x_postion*/
	if(x_p_result >= x_p_reg - 0.5)
		x_p_decide = (int)x_p_reg;
	else if(x_p_result <= 0.5 - x_p_reg)
		x_p_decide = (int)(0 - x_p_reg);
	else
	{
		for(i = 0;i<=2*x_p_reg-2;i++)
		{
			if((x_p_result >= (x_p_reg-i-1.5))&&(x_p_result <= (x_p_reg-i-0.5)))
				x_p_decide = (int)(x_p_reg - i);
		}
	}

	/*To get the fuzzy decide input value of x_velocity*/
	if(x_v_result >= x_v_reg - 0.5)
		x_v_decide = (int)x_v_reg;
	else if(x_v_result <= 0.5 - x_v_reg)
		x_v_decide = (int)(0 - x_v_reg);
	else
	{
		for(i = 0;i<=2*x_v_reg-2;i++)
		{
			if((x_v_result >= (x_v_reg-i-1.5))&&(x_v_result <= (x_v_reg-i-0.5)))
				x_v_decide = (int)(x_v_reg - i);
		}
	}

	/*Two inputs ready*/

	/*fuzzy decide, return fm*/
	t_maintain = fuzzy_maintain[x_p_decide + (int)(x_p_reg)][x_v_decide + (int)(x_v_reg)];
	return t_maintain * t_fuzzy_full / t_maintain_reg;
}
```

```
/*****************************************************************
NAME : simulation_main.cpp
FUNCTION: The main function of the simulation.
AUTHOR: Ke PENG
    Laboratory of NanoRobotics, Ecole Polytechnique de Montreal
VERSION: 3.0
EDIT HISTORY: 2010/12/10 File created.
```

```
2010/12/10 Version 1.0 Finished.
2010/03/22 Let fuzzy controllor to decide the maintaining time. Version 2.0 finished.
2010/03/24 Add a loop to test the elliptical-shaped model with a uniform distribution. Version 2.1.
2010/03/28 Add a progress report for looped-simulation. Test passed. Version 2.2.
2010/04/04 Add an angle-related test loop. Version 3.0
**********************************************************************/
#include "stdio.h"
#include "stdlib.h"
#include "fuzzy_header.h"
#include "math.h"
#include "iostream"
using namespace std;

double p_x;
double p_y;
double p_v_x;
double p_v_y;
double DeltaB_mt;
double DeltaB_tr;
double DeltaB_rs;
double DeltaB_fl;

double t_maintain_fuzzy_result;

int main(void)
{
	cout<<"Looped simulation test starts:"<<endl;
	int test_total = 0;
	int test_fulfill = 0;

	double test_p_x = 0;
	double test_p_y = 0;

	double p_x_step = 0.00001;
	double p_y_step = 0.00005;

	int loop_total = 0;
	int loop_current = 0;

	/*Angle related definition*/
	double p_x_center = 0.01495;
	double p_y_center = -0.00115;
	double p_x_copy = 0;
	double p_y_copy = 0;
	double angle = 0;
	double angle_table[5] = {0, pi/6, pi/4, pi/3, pi/2};
```

```
	int table_current = 0;

	loop_total = (int)((0.015 - 0.0149)/p_x_step)*((0.002271 - 0.000029)/p_y_step);

	for(table_current=0;table_current<=4;table_current++)
	{
		loop_current = 0;
		test_fulfill = 0;
		test_total = 0;

		angle = angle_table[table_current];
		for(test_p_x = 0.015; test_p_x >= 0.0149; test_p_x = test_p_x - p_x_step)
		{
			for(test_p_y = -0.002271; test_p_y <= -0.000029; test_p_y = test_p_y +
p_y_step)
			{
				p_x = test_p_x;
				p_y = test_p_y;

				loop_current++;

				if(loop_current == loop_total/10)
					cout<<"10% of simulation finished."<<endl;
				else if(loop_current == loop_total*2/10)
					cout<<"20% of simulation finished."<<endl;
				else if(loop_current == loop_total*3/10)
					cout<<"30% of simulation finished."<<endl;
				else if(loop_current == loop_total*4/10)
					cout<<"40% of simulation finished."<<endl;
				else if(loop_current == loop_total*5/10)
					cout<<"50% of simulation finished."<<endl;
				else if(loop_current == loop_total*6/10)
					cout<<"60% of simulation finished."<<endl;
				else if(loop_current == loop_total*7/10)
					cout<<"70% of simulation finished."<<endl;
				else if(loop_current == loop_total*8/10)
					cout<<"80% of simulation finished."<<endl;
				else if(loop_current == loop_total*9/10)
					cout<<"90% of simulation finished."<<endl;
				else if(loop_current == loop_total)
					cout<<"100% of simulation finished."<<endl;
				else
					;

			if((p_x - 0.01495)*(p_x - 0.01495) / 0.0000000025 + (p_y + 0.00115)*(p_y +
0.00115) / 0.000001256641 <= 1.0)
			{
```

```
                test_total++;
                /*Angle related                         */
                p_x_copy = p_x;
                p_y_copy = p_y;
                p_x = (p_x_copy - p_x_center)*cos(angle) - (p_y_copy - p_y_center)*sin(angle) + p_x_center;
                p_y = (p_x_copy - p_x_center)*sin(angle) + (p_y_copy - p_y_center)*cos(angle) + p_y_center;

                p_v_x = (1 - ((p_y+y_positive_centre_1)/y_positive_centre_1)*((p_y+y_positive_centre_1)/y_positive_centre_1))*x_v_max_max_1;
                p_v_y = 0.0;

                int process_result = 0;

                while(1)
                {
                        process_result = fm_tracking(0.0,t_tracking);
                                //tracking
                        if(process_result == -1)
                                break;
                        else if(process_result == -1)
                                break;
                        else
                        {
                                cout<<"x position is:  "<<p_x<<endl;
                                cout<<"y_position is:  "<<p_y<<endl;*/
                        }
                        t_maintain_fuzzy_result = t_maintain_fuzzy_decide(p_x,p_v_x);

                        DeltaB_rs = fuzzy_decide(p_x,p_v_x);
                                        //rising

                        process_result = fm_rising(DeltaB_rs);
                        if(process_result == -1)
                                break;
                        else if(process_result == -2)
                                break;
                        else
                                ;
                        DeltaB_mt = DeltaB_rs;
                        //maintaining
                        process_result = fm_maintained(DeltaB_mt,t_maintaining+t_maintain_fuzzy_result);

                if(process_result == -1)
```

```
                        break;
                    else if(process_result == -2)
                        break;
                    else
                        ;
                    DeltaB_fl = DeltaB_mt;
                                //falling
                    process_result = fm_falling(DeltaB_fl);
                    if(process_result == -1)
                        break;
                    else if(process_result == -2)
                        break;
                    else
                        ;
                }
                        if(p_y >= -0.000845)
                            test_fulfill++;
                    }
                    else
                        continue;
                }

            }

            cout<<"Simultion ends"<<endl;
            cout<<"Current angle is : "<<angle<<endl;
            cout<<"Total test number is "<<test_total<<endl;
            cout<<"Fulfilled test number is "<<test_fulfill<<endl;
            cout<<"Navigation rate is "<<(test_fulfill*100 / test_total)<<endl;
            cout<<"--------------------------"<<endl;
            //p_x = x_positive_full;
            //p_y = (p_radius - y_positive_full)/2;
            //p_y = -0.00119;
        }

        getchar();
        return 0;

}
```

ANNEXE 3 – **Power supply serial control**

```
#ifndef _controller_h_
#define _controller_h_

#include <windows.h>

/*fuzzy tables definition*/
const double fuzzy_table[13][13] =
{{-4.20,-3.50,-1.60,-1.63,-1.60,-1.44, 0.00, 0.00, 0.00, 1.40, 1.17, 1.75, 1.17},
 {-3.73,-3.50,-1.39,-1.63,-1.39,-1.44, 0.00, 0.00, 0.00, 1.40, 1.39, 1.75, 1.39},
 {-3.28,-3.50,-1.17,-1.63,-1.17,-1.44, 0.00, 0.00, 0.00, 1.40, 1.60, 1.75, 1.60},
 {-2.56,-2.86,-0.64,-1.00, 0.41, 0.78, 1.24, 1.40, 1.24, 2.95, 3.17, 3.77, 3.17},
 {-1.93,-1.75, 0.00, 0.00, 0.88, 1.66, 1.60, 1.66, 1.60, 2.83, 4.20, 4.08, 4.20},
 {-1.91, 0.00, 1.56, 3.27, 3.27, 3.11, 3.50, 3.50, 3.50, 3.50, 4.45, 4.94, 4.86},
 { 0.00, 1.75, 1.93, 4.32, 4.67, 4.67, 4.67, 4.67, 4.67, 4.67, 4.67, 5.25, 6.88},
 { 0.00, 1.75, 2.33, 4.25, 4.45, 4.45, 4.67, 4.67, 4.67, 4.89, 4.89, 5.25, 6.42},
 { 0.00, 1.75, 1.93, 3.96, 4.20, 4.10, 4.67, 4.67, 4.67, 5.02, 5.13, 5.25, 6.88},
 { 0.00, 1.75, 1.90, 3.67, 3.28, 4.10, 4.55, 4.53, 4.55, 5.02, 6.05, 5.25, 6.78},
 { 0.00, 1.75, 1.60, 1.88, 2.83, 4.10, 4.20, 4.10, 4.45, 5.02, 6.88, 6.42, 6.88},
 { 0.00, 1.75, 1.39, 1.88, 2.97, 3.82, 3.73, 3.82, 4.43, 5.02, 6.78, 6.42, 6.78},
 { 0.00, 1.75, 1.17, 1.88, 2.83, 3.63, 3.82, 3.63, 4.45, 5.02, 6.88, 6.42, 6.88}};

const double fuzzy_maintain[13][13] =
{{-1.71,-2.33,-2.42,-3.50,-4.89,-5.71 -6.88,-6.67,-6.88,-6.53,-6.88,-6.42,-6.88},
 {-2.33,-2.33,-2.92,-3.50,-4.90,-5.71,-6.78,-6.67,-6.78,-6.53,-6.78,-6.42,-6.78},
 {-2.96,-2.33,-3.42,-3.50,-4.89,-5.71,-6.88,-6.67,-6.88,-6.53,-6.88,-6.42,-6.88},
 {-4.37,-3.79,-4.56,-4.52,-5.01,-5.71,-6.78,-6.67,-6.78,-5.77,-6.05,-2.92,-4.30},
 {-6.11,-5.25,-6.11,-5.38,-6.11,-5.71,-6.88,-6.67,-6.88,-5.77,-5.13,-2.92,-2.18},
 {-5.92,-5.72,-5.92,-5.92,-5.92,-5.92,-6.42,-5.54,-5.54,-3.50,-3.06,-2.06,-0.89},
 {-6.88,-6.42,-6.88,-6.53,-6.88,-6.67,-6.89,-5.71,-4.89,-3.05, 0.00, 0.58, 1.17},
 {-6.42,-6.42,-6.42,-6.42,-5.54,-5.25,-5.08,-5.08,-5.08,-1.46, 1.52, 2.64, 3.55},
 {-6.88,-6.42,-6.88,-6.53,-6.42,-5.23,-4.67,-4.67,-4.67,-1.10, 2.42, 3.50, 5.13},
 {-6.78,-6.42,-6.78,-6.53,-6.26,-5.23,-4.67,-4.67,-4.67,-0.83, 3.65, 3.50, 6.05},
 {-6.88,-6.42,-6.88,-6.53,-6.42,-5.23,-4.67,-4.67,-4.67,-0.70, 5.13, 5.25, 6.88},
 {-6.78,-6.42,-6.78,-6.53,-6.26,-5.23,-4.67,-4.67,-4.67,-0.70, 5.63, 5.25, 6.78},
 {-6.88,-6.42,-6.88,-6.53,-6.42,-5.23,-4.67,-4.67,-4.67,-0.70, 6.11, 5.25, 6.88}};

/*fuzzy constants*/
const double x_p_reg = 6.0;
const double x_v_reg = 6.0;
const double DeltaB__y_reg = 7.0;
const double t_maintain_reg = 7.0;
const double t_maintain_ref = 0.1;
/*sequence parameters*/
const double DeltaB_y_full = 2;
```

```
const double t_fuzzy_full = 0.06;
const double fm_y_rising = 40.0;
extern double t_maintain;
extern double deltaB;

/*enviroment parameters*/
const double x_position_full = 236;          //unit: pixels
const double x_v_max_max = 485.7;            //unit: pixels/s
const double start_point = 16;

/*particle parameters*/
extern double p_x;
extern double p_y;
extern double p_v_x;

extern int LastBlobLocationX;
extern int LastBlobLocationY;
extern bool fDataReady;
DWORD WINAPI maincontroller(LPVOID iValue);

#endif
```

```
#include <windows.h>
#include <stdio.h>
#include "controller.h"
#include "string.h"
#include "iostream"

#using <System.dll>

using namespace std;
using namespace System;
using namespace System::IO::Ports;
using namespace System::Threading;

int x_coordinate;
int y_coordinate;

double p_x;
double p_y;
double p_v_x;

double t_maintain;
double deltaB;

double start_time;
double initial_time;
```

```
double t_sequence;
double t_rising;
double t_falling;
double p_x_stored;

char rt_s[100];
char mt_s[100];
char ft_s[100];

int GetCoordinates(int *x, int *y)
{
	while (!fDataReady);
	*x=LastBlobLocationX;
	*y=LastBlobLocationY;
	return 0;
}

double gradient_fuzzy_decide(double x_p, double x_v)
{
	double DeltaB = 0;
	int x_p_decide = 0;
	int x_v_decide = 0;
	double x_p_result = 0;
	double x_v_result = 0;

	if(x_p >= 0)
	{
		x_p_result = x_p * x_p_reg / (x_position_full - start_point);
		x_v_result = x_v * x_v_reg / x_v_max_max;
	}
	else
	{
		x_p_result = (x_p + x_position_full) * x_p_reg / x_position_full;
		x_v_result = x_v * x_v_reg / x_v_max_max;
	}

	/*linear transformation*/

	int i;

	/*To get the fuzzy decide input value of x_postion*/
	if(x_p_result >= x_p_reg - 0.5)
		x_p_decide = (int)x_p_reg;
	else if(x_p_result <= 0.5 - x_p_reg)
		x_p_decide = (int)(0 - x_p_reg);
	else
	{
```

```
		for(i = 0;i<=2*x_p_reg-2;i++)
		{
			if((x_p_result >= (x_p_reg-i-1.5))&&(x_p_result <= (x_p_reg-i-0.5)))
				x_p_decide = (int)(x_p_reg - i);
		}
	}

	/*To get the fuzzy decide input value of x_velocity*/
	if(x_v_result >= x_v_reg - 0.5)
		x_v_decide = (int)x_v_reg;
	else if(x_v_result <= 0.5 - x_v_reg)
		x_v_decide = (int)(0 - x_v_reg);
	else
	{
		for(i = 0;i<=2*x_v_reg-2;i++)
		{
			if((x_v_result >= (x_v_reg-i-1.5))&&(x_v_result <= (x_v_reg-i-0.5)))
				x_v_decide = (int)(x_v_reg - i);
		}
	}

	/*Two inputs ready*/

	/*fuzzy decide, return fm*/
	DeltaB = fuzzy_table[x_p_decide + (int)(x_p_reg)][x_v_decide + (int)(x_v_reg)];
	return DeltaB * DeltaB_y_full / DeltaB__y_reg;
}


double t_maintain_fuzzy_decide(double x_p, double x_v)
{
	double t_maintain = 0;
	int x_p_decide = 0;
	int x_v_decide = 0;
	double x_p_result = 0;
	double x_v_result = 0;

	if(x_p >= 0)
	{
		x_p_result = x_p * x_p_reg / (x_position_full - start_point);
		x_v_result = x_v * x_v_reg / x_v_max_max;
	}
	else
	{
		x_p_result = (x_p + x_position_full) * x_p_reg / x_position_full;
		x_v_result = x_v * x_v_reg / x_v_max_max;
	}
```

```
        /*linear transformation*/

        int i;

        /*To get the fuzzy decide input value of x_postion*/
        if(x_p_result >= x_p_reg - 0.5)
                x_p_decide = (int)x_p_reg;
        else if(x_p_result <= 0.5 - x_p_reg)
                x_p_decide = (int)(0 - x_p_reg);
        else
        {
                for(i = 0;i<=2*x_p_reg-2;i++)
                {
                        if((x_p_result >= (x_p_reg-i-1.5))&&(x_p_result <= (x_p_reg-i-0.5)))
                                x_p_decide = (int)(x_p_reg - i);
                }
        }

        /*To get the fuzzy decide input value of x_velocity*/
        if(x_v_result >= x_v_reg - 0.5)
                x_v_decide = (int)x_v_reg;
        else if(x_v_result <= 0.5 - x_v_reg)
                x_v_decide = (int)(0 - x_v_reg);
        else
        {
                for(i = 0;i<=2*x_v_reg-2;i++)
                {
                        if((x_v_result >= (x_v_reg-i-1.5))&&(x_v_result <= (x_v_reg-i-0.5)))
                                x_v_decide = (int)(x_v_reg - i);
                }
        }

        /*Two inputs ready*/

        /*fuzzy decide, return fm*/
        t_maintain = fuzzy_maintain[x_p_decide + (int)(x_p_reg)][x_v_decide + (int)(x_v_reg)];
        return t_maintain * t_fuzzy_full / t_maintain_reg;
}



public ref class PortChat
{
private:
   static bool _continue;
```

```
    static SerialPort^ _serialPort;


public:
    static void Main()
    {
        String^ name;
        //String^ message;
        StringComparer^ stringComparer = StringComparer::OrdinalIgnoreCase;
        Thread^ readThread = gcnew Thread(gcnew ThreadStart(PortChat::Read));

        // Create a new SerialPort object with default settings.
        _serialPort = gcnew SerialPort();

        // Allow the user to set the appropriate properties.
        _serialPort->PortName = SetPortName(_serialPort->PortName);
        _serialPort->BaudRate = SetPortBaudRate(_serialPort->BaudRate);
        _serialPort->Parity = SetPortParity(_serialPort->Parity);
        _serialPort->DataBits = SetPortDataBits(_serialPort->DataBits);
        _serialPort->StopBits = SetPortStopBits(_serialPort->StopBits);
        _serialPort->Handshake = SetPortHandshake(_serialPort->Handshake);

        // Set the read/write timeouts
        _serialPort->ReadTimeout = 500;
        _serialPort->WriteTimeout = 500;

        _serialPort->Open();
        _continue = true;
        readThread->Start();

        Console::Write("Name: ");
        name = Console::ReadLine();

        //Console::WriteLine("Type QUIT to exit");

            initial_time = GetTickCount();
            start_time = GetTickCount();
            t_sequence = 0;
            t_rising = 0;
            t_falling = 0;
            p_x_stored = start_point;
            while(1)
            {
                    if(GetTickCount() - start_time - 2000 - t_sequence >= 0)
                    {
                            GetCoordinates(&x_coordinate,&y_coordinate);
```

```
p_x = (double)x_coordinate;
p_y = (double)y_coordinate;

p_v_x = (p_x_stored - p_x)/((100 + t_sequence)/1000);
p_x_stored = p_x;

deltaB = gradient_fuzzy_decide(x_position_full - p_x,p_v_x);
t_maintain = t_maintain_fuzzy_decide(x_position_full - p_x,p_v_x) +
t_maintain_ref;
t_rising = ((int)(deltaB * 10000)) / (fm_y_rising * 10000);
t_falling = t_rising;
t_maintain = t_maintain * 1000;
t_rising = t_rising * 1000;
t_falling = t_falling * 1000;
t_sequence = t_maintain + t_rising + t_falling;

cout<<"x_coordinate = "<<x_coordinate<<endl;
cout<<"y_coordinate = "<<y_coordinate<<endl;
cout<<"p_v_x = "<<p_v_x<<endl;
cout<<"DeltaB = "<<deltaB<<endl;
cout<<"t_maintain = "<<t_maintain<<endl;
cout<<"t_rising = "<<t_rising<<endl;
cout<<"--------------------------------"<<endl;

deltaB = -deltaB;    //In this attempt, minus is the positive direction

_serialPort->Write("EXECUTE 0\n");
_serialPort->Write("NEWSEQ 2,1\n");
_serialPort->Write("SEQUENCE 1,1,1,0,0\n");
_serialPort->Write("PROGRAM 1\n");

sprintf(rt_s,"STEP 1,1,%.2f,0,1,0,%dms\n",deltaB,int(t_rising));
String^ mystr_rt=gcnew String(rt_s);
_serialPort->Write(mystr_rt);

sprintf(mt_s,"STEP 2,0,%.2f,0,1,0,%dms\n",deltaB,int(t_maintain));
String^ mystr_mt=gcnew String(mt_s);
_serialPort->Write(mystr_mt);

sprintf(ft_s,"STEP 3,1,0.00,0,1,0,%dms\n",int(t_falling));
String^ mystr_ft=gcnew String(ft_s);
_serialPort->Write(mystr_ft);

_serialPort->Write("EOS\n");
_serialPort->Write("EXECUTE 1\n");
_serialPort->Write("RUN 1\n");
```

```
                start_time = GetTickCount();

            }

            if(x_coordinate > x_position_full + 50)
                break;
            else
                continue;

        }

    readThread->Join();
    _serialPort->Close();
}

static void Read()
{
    while (_continue)
    {
        try
        {
            String^ message = _serialPort->ReadLine();
            Console::WriteLine(message);
        }
        catch (TimeoutException ^) { }
    }
}

static String^ SetPortName(String^ defaultPortName)
{
    String^ portName;

    Console::WriteLine("Available Ports:");
    for each (String^ s in SerialPort::GetPortNames())
    {
        Console::WriteLine("   {0}", s);
    }

    Console::Write("COM port({0}): ", defaultPortName);
    portName = Console::ReadLine();

    if (portName == "")
    {
        portName = defaultPortName;
    }
    return portName;
```

```
}

static Int32 SetPortBaudRate(Int32 defaultPortBaudRate)
{
   String^ baudRate;

   Console::Write("Baud Rate({0}): ", defaultPortBaudRate);
   baudRate = Console::ReadLine();

   if (baudRate == "")
   {
      baudRate = defaultPortBaudRate.ToString();
   }

   return Int32::Parse(baudRate);
}

static Parity SetPortParity(Parity defaultPortParity)
{
   String^ parity;

   Console::WriteLine("Available Parity options:");
   for each (String^ s in Enum::GetNames(Parity::typeid))
   {
      Console::WriteLine("   {0}", s);
   }

   Console::Write("Parity({0}):", defaultPortParity.ToString());
   parity = Console::ReadLine();

   if (parity == "")
   {
      parity = defaultPortParity.ToString();
   }

   return (Parity)Enum::Parse(Parity::typeid, parity);
}

static Int32 SetPortDataBits(Int32 defaultPortDataBits)
{
   String^ dataBits;

   Console::Write("Data Bits({0}): ", defaultPortDataBits);
   dataBits = Console::ReadLine();

   if (dataBits == "")
   {
```

```
        dataBits = defaultPortDataBits.ToString();
    }

    return Int32::Parse(dataBits);
}

static StopBits SetPortStopBits(StopBits defaultPortStopBits)
{
    String^ stopBits;

    Console::WriteLine("Available Stop Bits options:");
    for each (String^ s in Enum::GetNames(StopBits::typeid))
    {
        Console::WriteLine("   {0}", s);
    }

    Console::Write("Stop Bits({0}):", defaultPortStopBits.ToString());
    stopBits = Console::ReadLine();

    if (stopBits == "")
    {
        stopBits = defaultPortStopBits.ToString();
    }

    return (StopBits)Enum::Parse(StopBits::typeid, stopBits);
}

static Handshake SetPortHandshake(Handshake defaultPortHandshake)
{
    String^ handshake;

    Console::WriteLine("Available Handshake options:");
    for each (String^ s in Enum::GetNames(Handshake::typeid))
    {
        Console::WriteLine("   {0}", s);
    }

    Console::Write("Handshake({0}):", defaultPortHandshake.ToString());
    handshake = Console::ReadLine();

    if (handshake == "")
    {
        handshake = "XOnXOff";
                        //handshake = defaultPortHandshake.ToString();
    }

    return (Handshake)Enum::Parse(Handshake::typeid, handshake);
```

```
    }
};

DWORD WINAPI maincontroller(LPVOID iValue)
{
        PortChat::Main();

        system("pause");
        return 1;
}
```