THE THESIS OF DOCTOR OF PHILOSOPHY

Study on a Tele-operative Catheter System for Endovascular Neurosurgery

Xu Ma

Graduate School of Engineering
Kagawa University

Abstract

The incidence of cardiovascular and cerebrovascular diseases has been increasing with the quickening pace of modern life. These diseases are a major cause of death, about 1.7 million people annually worldwide or accounting for 29.2% of all mortality based on a World Health Organization survey. Intracavity intervention is expected to become increasingly popular in medical practice, both for diagnosis and treatment, because of the small incision, short recovery time, and reduced burden on patients. During the past years, significant research effects have been done in the development of technology for minimally invasive surgery (MIS), such as laparoscopy, has grown as a very suitable domain for robotic system. A lot of diagnosis and medical surgery with an endoscope or a catheter are performed for minimum invasive surgery recently. There are a lot of advantages as earliness etc. However, it requires a lot of skills for the operation so that this may do the operation in the inside of the body that cannot be watched directly. In order to solve these problems, two kinds of robotic catheter systems with force feedback and visual feedback have been developed in this thesis, in addition, the force sensors system and the feedback system have been developed to detect the contact force information so as to let neurosurgeon know the contact information during vascular interventional surgery. Based on this information, the neurosurgeon can avoid danger during the operation.

In this thesis, the two kinds of robotic catheter systems with the structure of master-slave have been developed. Firstly, we developed a novel kind of robot-assisted catheter system and this system can realize the force feedback and visual feedback to enhance the safety during the operation. The robot-assisted catheter system is equipped with the structure of master-slave. In the master side, there consists of the master control system and master manipulator which can accord with the operating habit of the neurosurgeon, it is the first one can simulate the operating skill and extract the operating skill from experienced neurosurgeon in the world. The neurosurgeon operates the right handle of the master manipulator meanwhile the operating instructions are transmitted to the slave manipulator. Based on the operating instructions transmitted from master side, the slave manipulator drives the catheter to insert and rotate inside the blood vessel of patient. A loadcell is linked to the catheter supporting frame to measure the friction force reflected on the catheter and the force measured by loadcell can be transmitted to the master manipulator and generate a haptic feedback to the neurosurgeon. We used an optical fiber force sensor to measure the contact force between catheter tip and blood vessel. The force information measured by the optical fiber force sensor will be transmitted to the master side and display to the neurosurgeon by using the user interface. In addition, we used an IP camera to monitor the situation of the operation and the monitoring image will be transmitted to the neurosurgeon. So the feedback system including

force feedback and visual feedback is realized and it is used to enhance the safety during vascular interventional surgery. Next, this thesis presents the evaluation performance of robot-assisted catheter system, including master manipulator and slave manipulator. In order to operate inside the blood vessel, PID fuzzy controller were developed to compensate for the catheter performance limitations and evaluated with "vitro" experiments. Through the addition of compensation terms, the system is able to achieve position tracking with less than 1 mm RMS error and rotation tracking with 3^{o} RMS error.

Unlike the conventional bedside technique, which requires surgeons to manipulate a catheter using their hands, employment of these remote manipulating systems removes the catheter from the surgeons' hands, thus removing his/her dexterous and intuitive skills from the procedure. Furthermore, the technological complexities of these systems may require long training times to ensure that the surgeons are skilled in their use. Therefore, it should be beneficial if a catheter manipulating system incorporated the dexterous skill set of an experienced surgeon during the procedures.

Secondly, a new prototype robotic catheter manipulating system has been designed and constructed based on the requirements for the endovascular surgery. Compared with system mentioned above, this system features a master controller called surgeon console, using two motion-sensing devices via control unit DSP to communicate the position and rotation information with slave side. Also, we designed the

haptic device to provide the feeling back to surgeons. The whole system was evaluated in aspect of dynamic and static performance of the axial and radial motions. The results of synchronization experiments had to evaluate the accuracy and precision of sensed and replicated motions. Finally, Tele-operation had been done by EVE simulator to provide the performance under the similar situations.

A number of improvements can be made to the robotic catheter system in the areas of mechanical design and control. The system can also be applied to new medical applications, including neurosurgery and peripheral vascular disease. In the future, the performance of the measurement mechanism of friction force and the haptic device will be discussed. Furthermore, the current limitations of the robotic catheter control system are accurate position control of the catheter tip and the 3D image servoing system. The model-based feedforward position control of the catheter tip is not able to adapt to changes in the environment or catheter configuration. This shortcoming could be overcome with accurate tip position measurements. Finally, utilizing this system to perform a range of interventional surgery in vivo is required to the clinical practice.

Contents

List of Tables

List of Figures

Acknowledgements

The author wishes to express his great gratitude to his supervisor Professor Shuxiang Guo for his invaluable guidance, support and friendly encouragement throughout my Ph.D. course and for providing me with first class resources.

The author would like to thank Prof. Hideyuki Hirata and Prof. Keisuke Suzuki. Thanks them for their valuable advices and suggestion on my research. Without their help, I can hardly finish this thesis.

Also the author would like to acknowledge the efforts of his laboratory members, especially Dr. Nan Xiao whose time and expertise were greatly appreciated.

This research was supported by Kagawa University Specially Promoted Research fund 2009-2011.

Finally, the author gives special appreciation to his parents for their love, patience and support.

Declaration

I hereby declare that this submission is my own work and that to the best of my knowledge and belief. It contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Chapter 1
Introduction

1.1 Preface

In this thesis, two kinds of robotic catheter systems with force feedback and visual feedback are proposed and developed for endovascular neurosurgery: the catheter operating system and the robot-assisted catheter system. Both two kinds of catheter systems can avoid danger effectively during operation based on the feedback system included force feedback and visual feedback. In addition, we developed a force sensors system and two kinds of user interfaces which is used to monitor the contact force measured by force sensors. The controller of the robot-assisted catheter system can simulate the neurosurgeon's operating skill to operate the catheter, furthermore, it can extract the operating skill of experienced neurosurgeon to train unskilled neurosurgeon or novice. We did the performance evaluation and carried out the simulation experiment in vitro, the evaluated results and experimental results indicated that the developed two kinds of robotic catheter system work well, both them can avoid danger effectively during vascular interventional surgery.

1.2 Background

With the quickening pace of modern life, the brain diseases of people are increasing, such as cerebral aneurysm and infarction and so on. The traditional surgery spends patients a lot of operation time and has long recovery time, the burden on patients is heavy. Minimally invasive surgery (MIS) has grown as a very suitable domain by using robot-assisted system. A lot of diagnosis and medical surgery with an endoscope or a catheter are performed for minimally invasive surgery recently. The diagram of the embolization operation is shown in Figure 1-1, and the Figure 1-2 shows the coil embolization surgery, it is difficult for novice to insert the catheter from femoral to the position of embolization in the blood vessel of patient's brain. There are a lot of advantages as earliness etc. However, it requires a lot of skills for the operation so that this may do the operation in the inside of the body that cannot be watched directly. Such surgery presents many challenges:

1) Doctors must be very well trained and possess the skills and experience to insert catheters. Intravascular neurosurgery is much more difficult than traditional surgery and there are few skilled doctors who can perform this type of operation. To keep pace with the growing number of patients, a mechanism is required to allow the training of sufficient numbers of doctors.

2) During the operation, doctors check the position of the catheter tip

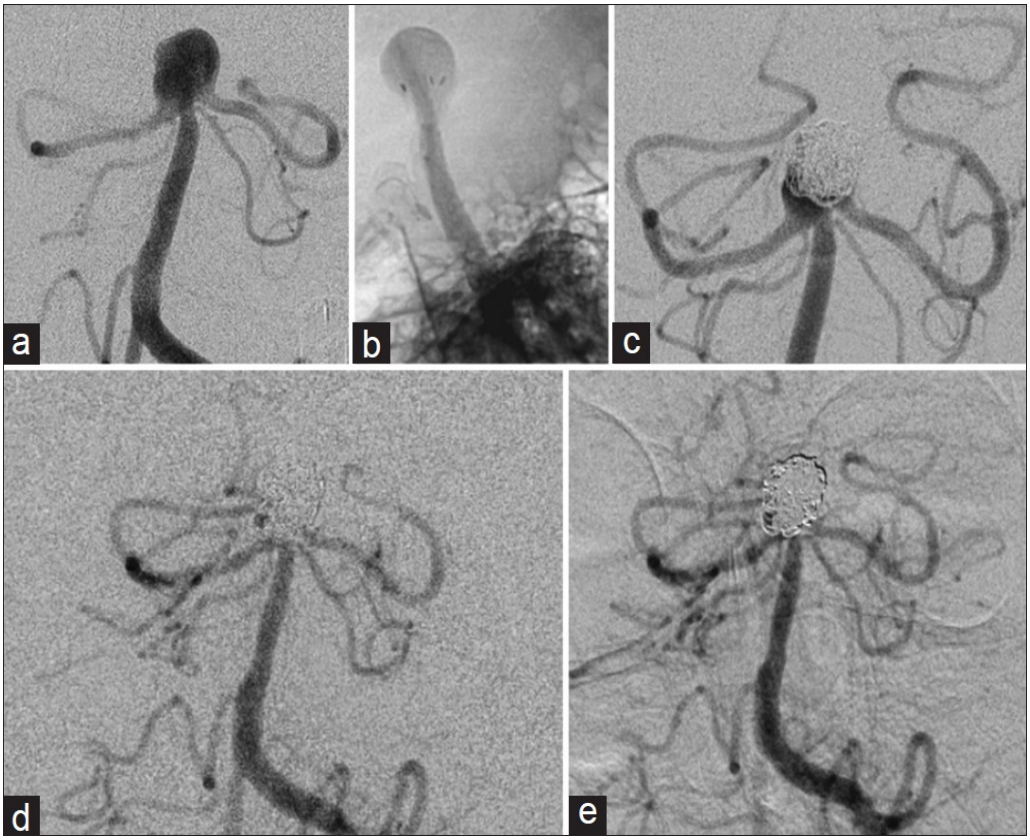
using the X-ray camera. Although they wear protective suits, it is very difficult to shield the doctor's hands and face from the effects of the X-ray radiation, which may result in radiation-related illness after long periods of exposure.

3) In intravascular neurosurgery, catheters are inserted into the patient's blood vessels, which in the brain are very sensitive. When operating in this area, extreme care is required to avoid damaging the fragile vessels. An experienced neurosurgeon can achieve an accuracy of about 2 mm. However, as the contact force between the blood vessel and the catheter cannot be judged accurately by the doctor, so how to measure the contact force and feedback to the surgeon become significant.

4) There is not a kind of robotic catheter system can imitate surgeon's operating skill to insert and rotate catheter. Therefore, a master-slave robot-assisted catheter system is required for such cases so that the operation can be proceeded.

Based on aforementioned background, in this paper, we proposed and developed two kinds of robotic catheter systems with force feedback and visual feedback, the first robotic catheter system used the Phantom Omni as the controller to control the catheter operating mechanism on the slave side, however, it cannot accord with the operating habit of the neurosurgeon, the second robot-assisted catheter system was proposed and developed based on the problems existing in the first robotic

catheter system, it is also with a structure of master-slave, it can transmit the neurosurgeon's skill at inserting and rotating a catheter. It avoids danger during vascular interventional surgery (VIS) by making use of force feedback and visual feedback.



(a) A basilar apex aneurysm pretreatment. (b) Lateral view of the Enterprise stent deployed in the aneurysm dome. (c) A small basilar apex aneurysm recurrence before redo treatment. (d) Redo coil embolization of recurrent aneurysm. (e) After redo coil embolization showing no further recurrence.

Figure1- 1 Cerebral angiogram [Padalino13]

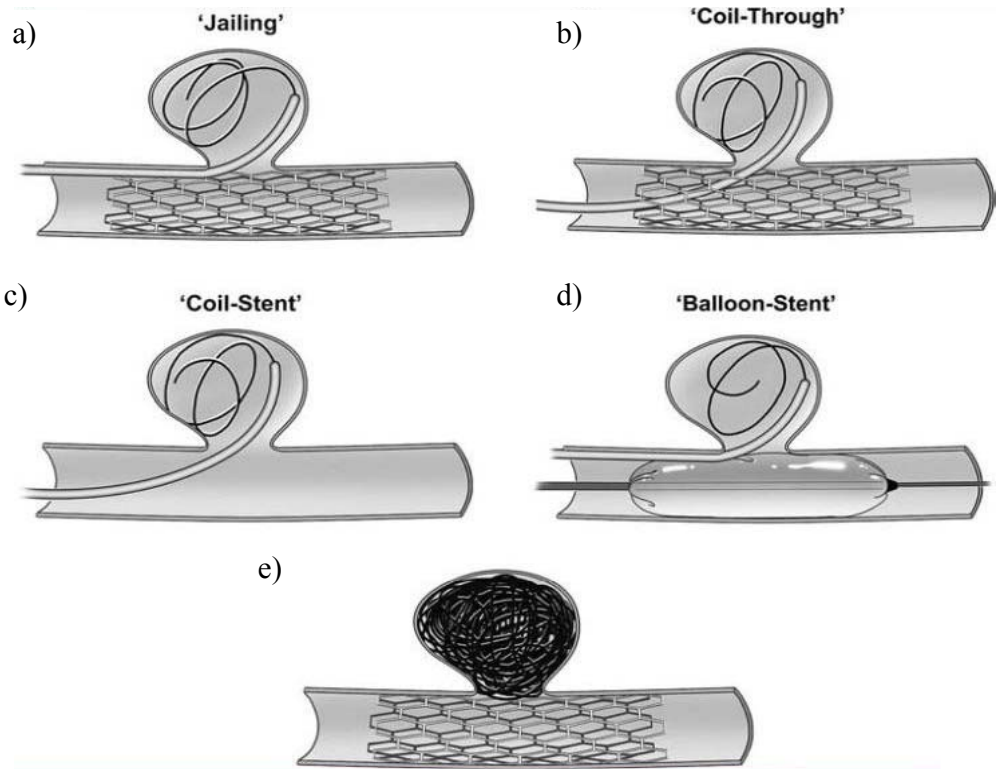


Figure1- 2 Stent assisted aneurysm coil embolization [Spiotta12]

1.3 Literature review

There are some relative products and researches on catheter system in the world. Recently, telesurgery performed using a microscopic micromanipulator system called the "NeuRobot" was reported in Neurosurgery. In this case report, the authors proved that the use of the NeuRobot was feasible using a private network [Goto09]. Many technological advances have been reported, including the following: a new prototype of a microcatheter with an active guidewire that has two bending degrees of freedom and is made using ionic conducting

polymer film (ICPF) with a shape memory alloy (SMA) actuator fixed at its end to act as a servo actuator [Fukuda94-96]; a master-slave catheterization system for positioning a steerable catheter [Fu11]; a new catheter driving method using a linear step mechanism for intravascular neurosurgery [Arai02]; force sensors based on a catheter operating system [Guo07-11]; research on an tele-operating system for medical application [Marcelli08]; state-of-the-art in force and tactile sensing for minimally invasive surgery [Puangmali08]; a feasibility study measuring the tip and side forces of a novel catheter prototype [Polygerinos09] and contact and friction between the catheter and blood vessel [Takashima07]; [Wang11] reported on the actuation and localization of an active capsule endoscope and [Marcelli08] reported a novel telerobotic system to navigate standard electrophysiology catheters remotely. Further, [Preusche02] reported the concept of teleoperation in minimally invasive surgery. In addition, a robot mechanism for the remote steering and positioning of interventional devices has been fabricated [Srima/thveeravalli10], remote-controlled vascular interventional surgery robot was reported [Wang10] and so on.

The ANGIO Mentor provides fully simulated, hands-on practice of interventional endovascular procedures [OKB Medical Ltd.]. It has been designed for interventional cardiologists, interventional radiologists and vascular surgeons and trainees, allowing the performance of full procedures as well as the training of basic skills.

Angio Mentor which is shown in Figure 1-3 allows trainees to perform a diagnostic angiogram by inserting a catheter into an artery under fluoroscopic guidance, along with subsequent injection of contrast agent.

Figure1- 3 ANGIO Mentor endovascular surgical training simulator (Simbionix Corporation in Israel, 2008)

One of the most popular product is a robotic catheter placement system called Sensei Robotic Catheter System [Kanagaratnam08] offered by Hansen Medical shown in Figure 1-4. The Sensei provides the physician with more stability and more force in catheter placement with the Artisan sheath compared to manual techniques, allowing for

more precise manipulation with less radiation exposure to the doctor, commensurate with higher procedural complications to the patient. Because of the sheath's multiple degrees of freedom, force detection at the distal tip is very hard.

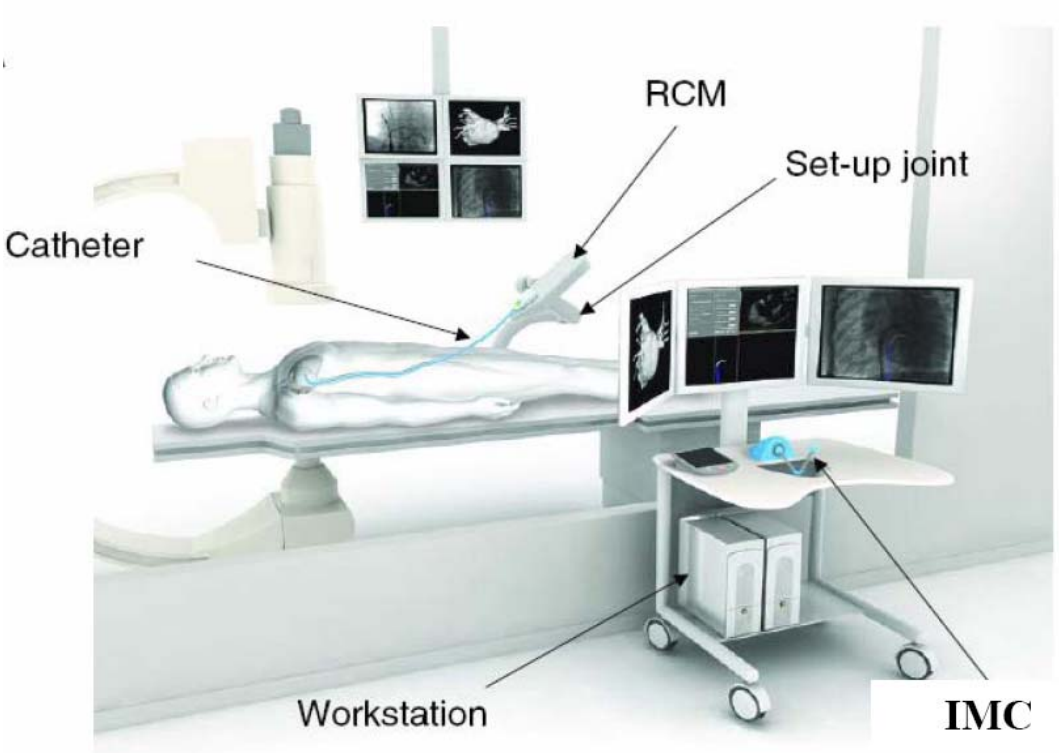


Figure1- 4 Sensei Robotic Catheter System

Catheter Robotics Inc. has developed a remote catheter system called Amigo shown in Figure 1-5. This system has a robotic sheath to steer catheters which is controlled at a nearby work station, in a manner similar to the Sensei system. The first in human use of this system was

in April 2010 in Leicester UK, where it was used to ablate atrial flutter.

Figure1- 5 Amigo remote catheter system

Figure1- 6 Catheter Guidance Control and Imaging' (CGCI) system

Magnatecs Inc. has produce their ‘Catheter Guidance Control and Imaging’ (CGCI) system as shown in Figure 1-6. This has 4 large magnets placed around the table, with customized catheters containing magnets in the tip. The catheter is again moved by the magnetic fields and is controlled at a nearby work station.

The Stereotaxis Inc. developed a magnetic navigation system: the Stereotaxis Niobe II [Stargen Inc.] shows in Figure 1-7. The system facilitates precise vector based navigation of magnetically-enabled guide wires for percutaneous coronary intervention (PCI) by using two permanent magnets located on opposite sides of the patient table to produce a controllable magnetic field.

Figure1- 7 Stereotaxis Niobe II

Although these products have been developed and marketed, most concern is still the safety of the system. Force information on the catheter during the operation is very important to insure the safety of the surgery. However, detection of the force on catheters is very hard to solve in these systems. A potential problem with a remote catheter control system is the lack of mechanical feedback that one would receive from manually controlling a catheter [Kanagaratnam08]. To solve the problems, we proposed two kinds of novel catheter systems with force feedback and visual feedback in this thesis.

1.4 Contributions

In this thesis, two kinds of robotic catheter operating systems with force feedback and visual feedback were proposed and developed, and the force sensors system were developed to measure the contact force between catheter and blood vessel. In addition, two kinds of user interfaces for monitoring the contact forces were developed. The mainly contributions of two kinds of robotic catheter systems are as following:

(1) The first robotic catheter system: A robot-assisted catheter system with force feedback and visual feedback was developed. We also proposed a mechanism which can be used to measure the operating force on the catheter, and the force information can be transmitted to

the master manipulator and generate a haptic feedback to the neurosurgery, via the haptic feedback, the neurosurgeon can decide whether inserting the catheter or not. We carried out the performance evaluation for the robot-assisted catheter system, and also, we did the tele-operation by PID fuzzy control method. The evaluated results and experimental results imply that the developed robot-assisted catheter system with force feedback and visual feedback work very well, it is suitable for supporting neurosurgeons to the vascular interventional surgery. In addition, the robot-assisted catheter system can be used to train unskilled or inexperience neurosurgeon.

(2) The second robot-assisted catheter system: a new prototype robotic catheter manipulating system has been designed and constructed based on the requirements for the endovascular surgery. Compared with system mentioned above, this system features a master controller called surgeon console, using two motion-sensing devices via control unit DSP to communicate the position and rotation information with slave side. Also, we designed the haptic device to provide the feeling back to surgeons. The whole system was evaluated in aspect of dynamic and static performance of the axial and radial motions. The results of synchronization experiments had to evaluate the accuracy and precision of sensed and replicated motions. Finally, Tele-operation had been done by EVE simulator to provide the performance under the similar situations.

(3) The user interface for monitoring the contact force between catheter and blood vessel were developed. The force monitoring system and danger avoiding system were applied in the robotic catheter system. They can monitor the contact force information in real time. If the force on the catheter goes beyond the safety range, the doctor will get warning information from both monitors and controller. The experimental results imply that the developed user interfaces for monitoring force is effective to help neurosurgeon avoid danger during operation.

1.5 Thesis structure

Figure 1-8 shows the structure of the thesis. Chapter 1 is the introduction. It is composed of preface, background, literature review, research purposes, research approaches and thesis structure. In chapter 2, one kind of catheter operating system will be introduced. We developed mechanical system, control system and feedback system. Then we carried out the performance evaluation to confirm the validity of the robot-assisted catheter system. Chapter 3 introduces the tele-operation by PID fuzzy control method. In chapter 4, a novel robotic catheter manipulating system has been proposed. The system structure and system control realization have been introduced. In chapter 5, we do the performance evaluation of the novel robotic catheter manipulating system. In chapter 6, we carry out the catheter inserting

experiment by using endovascular evaluator (EVE) model. In chapter 7, conclusions and future work are given.

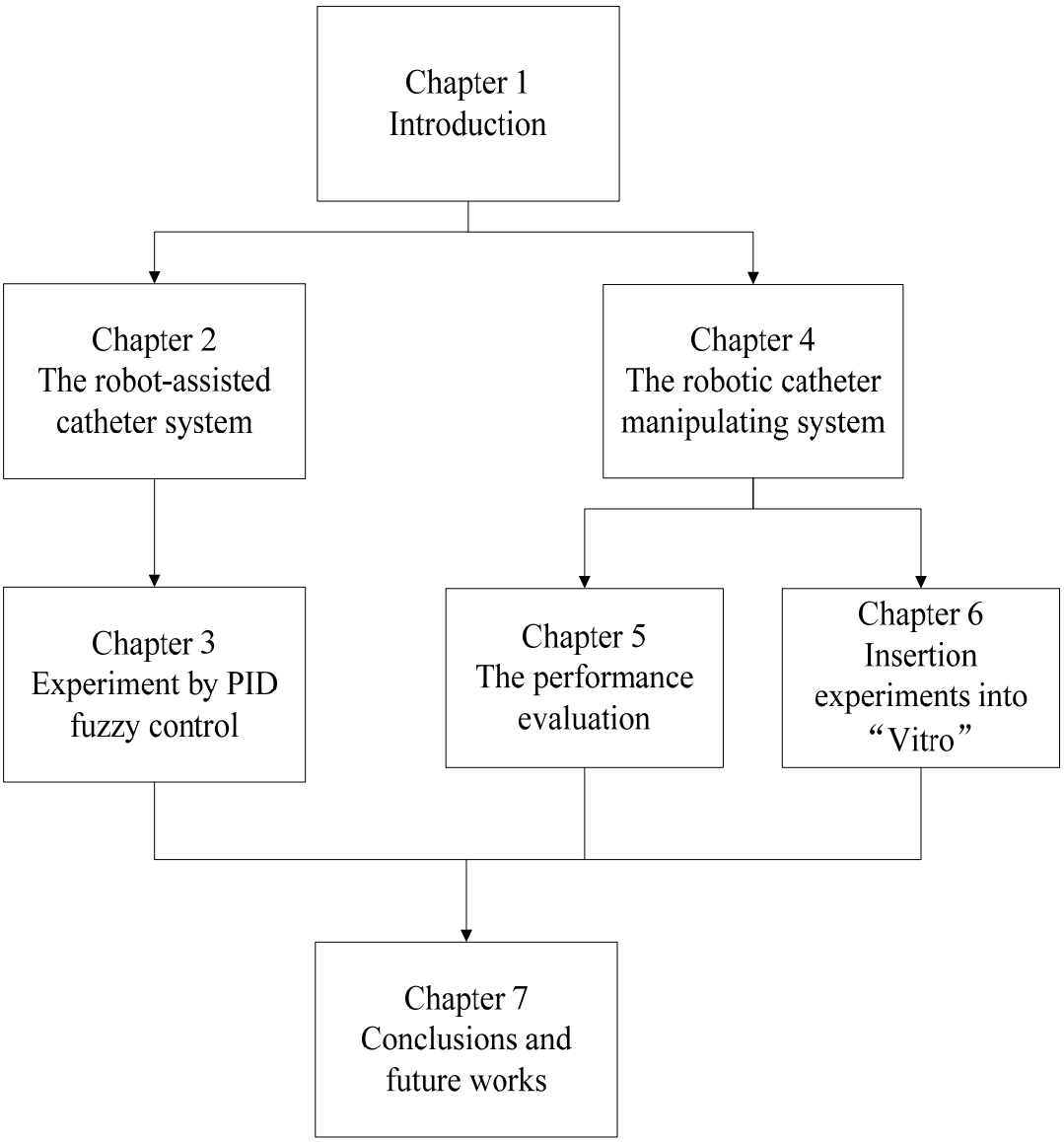


Figure1- 8 The structure of the thesis

Chapter 2

The robot-assisted catheter system

Based on the aforementioned problems in the chapter 1, we designed and developed a kind of robot-assisted catheter system for supporting neurosurgeons to do the operation of vascular interventional surgery, also, this kind of robotic catheter system can be used to train unskilled neurosurgeon to complete the operation, the conceptual diagram of the master-slave robotic catheter system is shown in Figure 2-1. On the master side, the surgeon views a monitor and operates the master manipulator. The control instructions are transmitted to the slave side. After receiving instructions, the slave manipulator inserts and rotates the catheter. The motions of the catheter on the slave side follow the motions of the catheter on the master side. Consequently, the surgeon appears to operate the catheter as though adjacent to the patient, controlling the position and velocity of the catheter. An IP camera is used to monitor the operation and provide visual feedback to surgeon. If the catheter contacts a blood vessel wall, the force is detected and transmitted to the neurosurgeon's hand, realizing force feedback. Figure 2-2 is a flow chart of the control instructions for our robotic catheter system. The robotic system avoids danger during the operation via force and visual feedback.

Both master manipulator and slave manipulator employ DSP (TI, TMS320F28335) as their control units, Communication between master manipulator and slave manipulator is realized by the TCP/IP protocol communication in tele-operation and SPI (RS 422) in local operation.

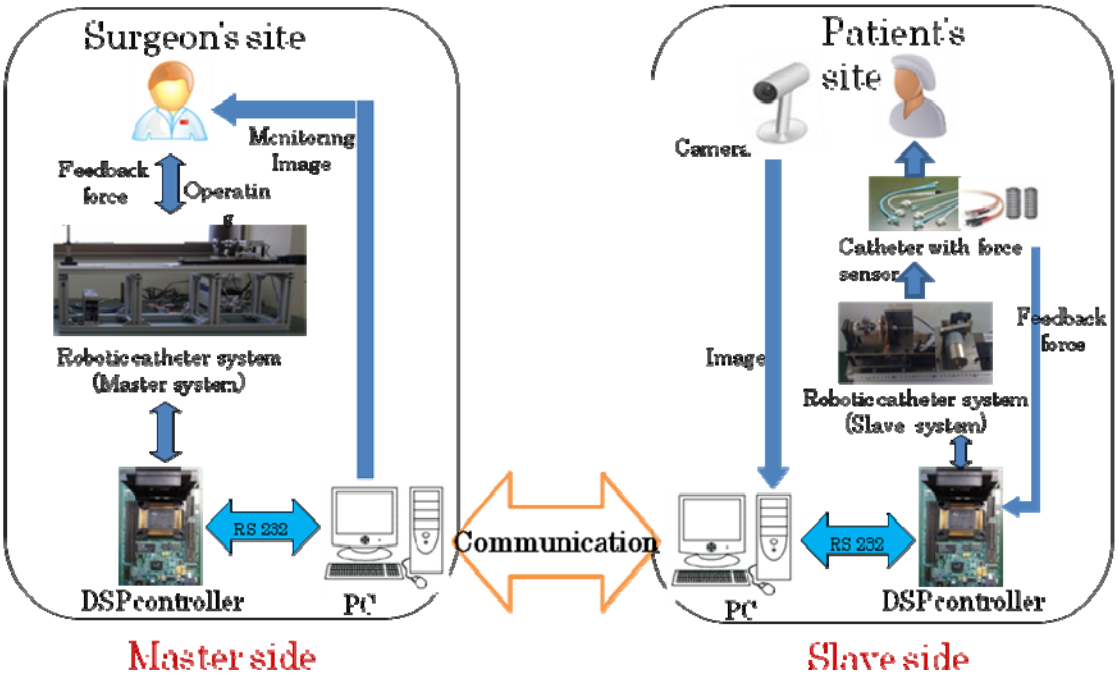


Figure 2- 1 The conceptual diagram of robot-assisted catheter system

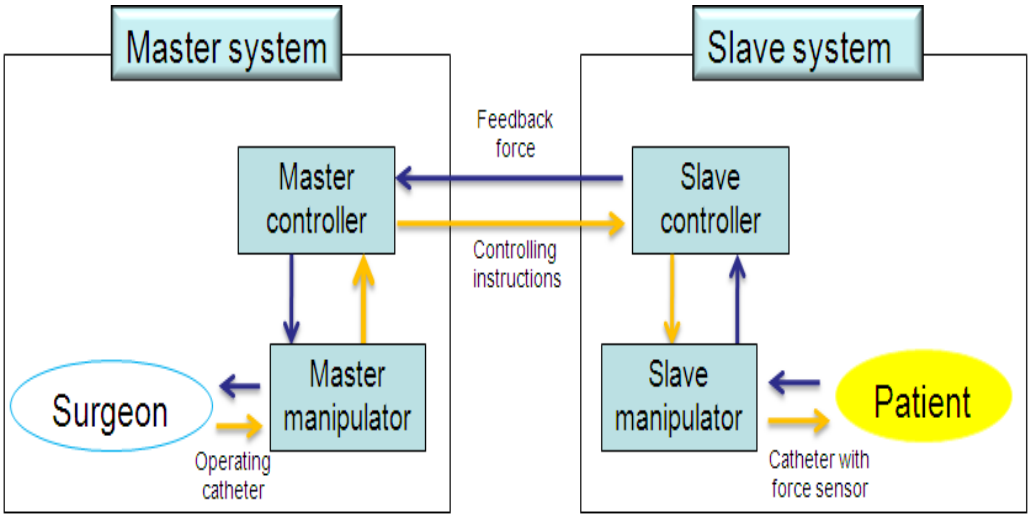


Figure 2- 2 The flow chart of the control instructions

2.1 The system structure

The Figure 2-3 shows the developed robot-assisted catheter system, which is with the structure of the mater-slave, both the master system and the slave system have two degree of freedoms (DOFs), the two parts keep the same moving mode at the same time, the communication between the two parts was realized based on RS 422 and internet.

2.1.1 The master system

On the master side, the slide platform is fixed on the supporting frame (Figure 2-4). The master system devices, including a left handle with one switch, a right handle, step motor, load cell, and maxon DC motor, are on the slide platform. A switch placed on the left handle is used to control these two graspers in slave side, only one switch is

enough because the catheter is clamped by one grasper at the same time. Operator's action is measured by using the right handle. The handle takes the same movement motion with the slave manipulator, it has two DOFs, one is axial motion and the other one is radial motion. The handle is sustained by a bearing, and is linked to a loadcell, a pulle is fixed on the handle, a DC motor is applied to generate torque feedback, a pulley which is couple to the upper one is fixed to the axle of the motor. The step motor is used to drive the slide platform forward and backward.

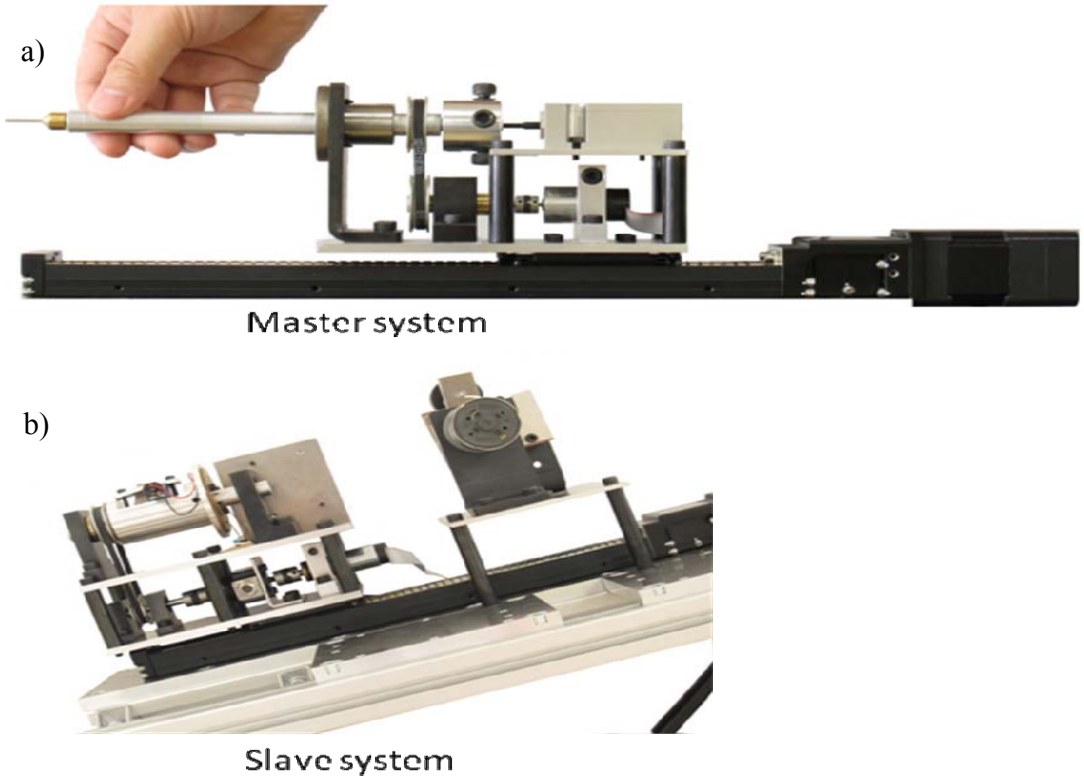


Figure 2- 3 The developed master-slave robot-assisted catheter system

The neurosurgeon moves the catheter forward and backward or rotates the right handle on the master side as though the neurosurgeon was beside the patient. The operating information from the handle is transmitted to the slave side, where the catheter clamp inserts and rotates the actual catheter as commanded from the master side. If the catheter contacts a blood vessel wall, the load cell detects it and the information is transmitted to the neurosurgeon's hand. Force feedback is realized by the master-slave robotic catheter operating system. The neurosurgeon feels the contact, and no x-rays are required during intravascular neurosurgery.

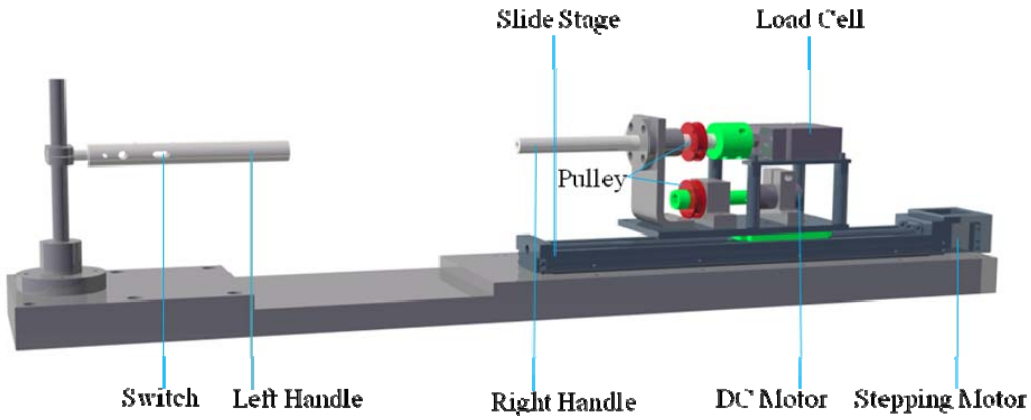


Figure 2- 4 The master system

2.1.2 The slave system

The slave side mechanism shown in Figure 2-5 is similar to the master side, it also has two DOFs, one is axial motion along the frame, and the other one is radial motion, two graspers are placed at this part.

The operator can drive the catheter to move along both axial and radial when the catheter is clamped by grasper 1. The catheter keeps its position and the catheter driven part can move smoothly when the catheter is clamped by grasper 2. The slave side consists of a catheter clamping device, two DC motors, a slide platform, step motor, maxon DC motor, load cell, torque sensor, and support frame. a slide platform is fixed on the supporting frame. The devices of the slave system are on the slide platform. The step motor is used to the drive slide platform forward and backward and the maxon DC motor is used to rotate the catheter. The two DC motors are used to control the catheter graspers. The load cell is used to measure the force between the catheter and blood vessel wall and the torque sensor and maxon DC motor are used to measure the information of catheter rotation during the operation. The measured force information is transmitted to the neurosurgeon's hand, so that the neurosurgeon can feel the feedback information from the slave side. A switch on the left handle on the master side controls the catheter graspers. When the operator wants to insert or rotate the catheter, grasper 2 is raised and grasper 1 clamps the catheter. The catheter navigator moves forward with the catheter for insertion or rotation. The grasper 2 then clamps the catheter, grasper 1 is raised and the catheter navigator moves backward. The moving mode of catheter is shown in Figure 2-6. Repeating these actions, the actions of the slave side follow the commands of the master side. If the catheter contacts the blood vessel wall, the force information is detected and transmitted

to the neurosurgeon's hand.

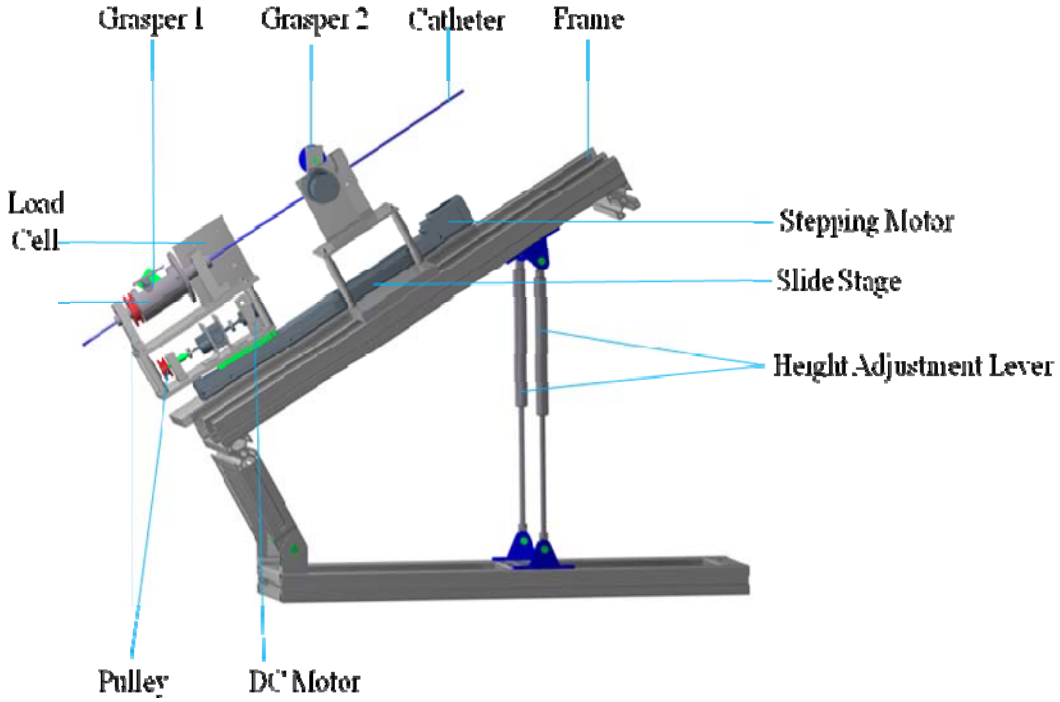


Figure 2- 5 The slave system

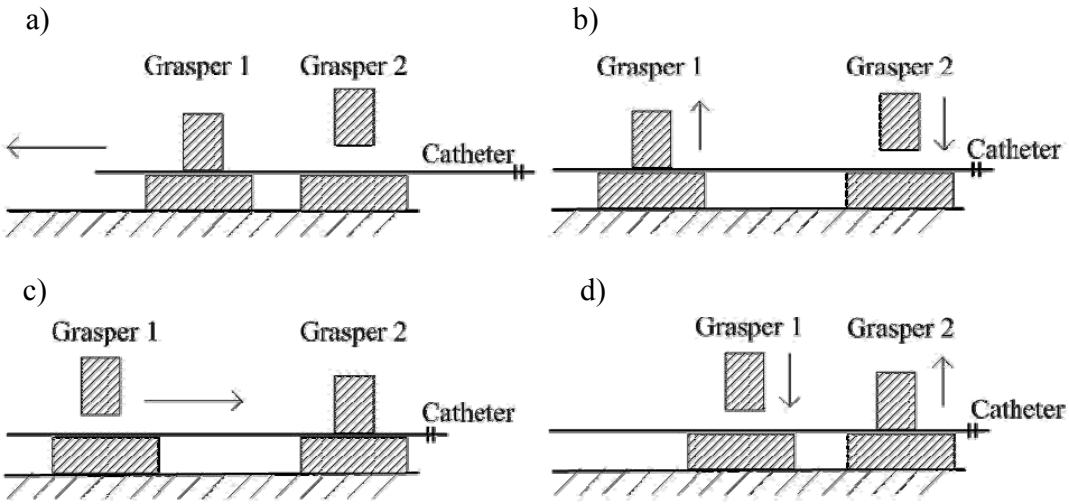


Figure 2- 6 The moving mode of catheter

2.2 Communication system

We designed two kinds of communication system, one kind is based on the RS 422, the conceptual diagram of the communication system based on RS 422 is shown in Figure 2-7, two pieces of DSP is connected by serial peripheral interface (SPI), the cable which is used to connect the two pieces of DSP is about 5m, the operating data both master side and slave side are saved to the PC which is connected to the DSP on the master side with RS 232. And the other kind is based on internet, the conceptual diagram of the communication system based on internet is shown in Figure 2-8. The two PC (one is on the master side, and the other is on the slave side) are connected with TCP/IP communication, and for each side serial communication was employed. The baud rate is set to 19200. The master side sends the information of axial displacement and rotation angle of the handle to the catheter manipulator on the slave side. At the same time the slave manipulator sends force information to the master side.

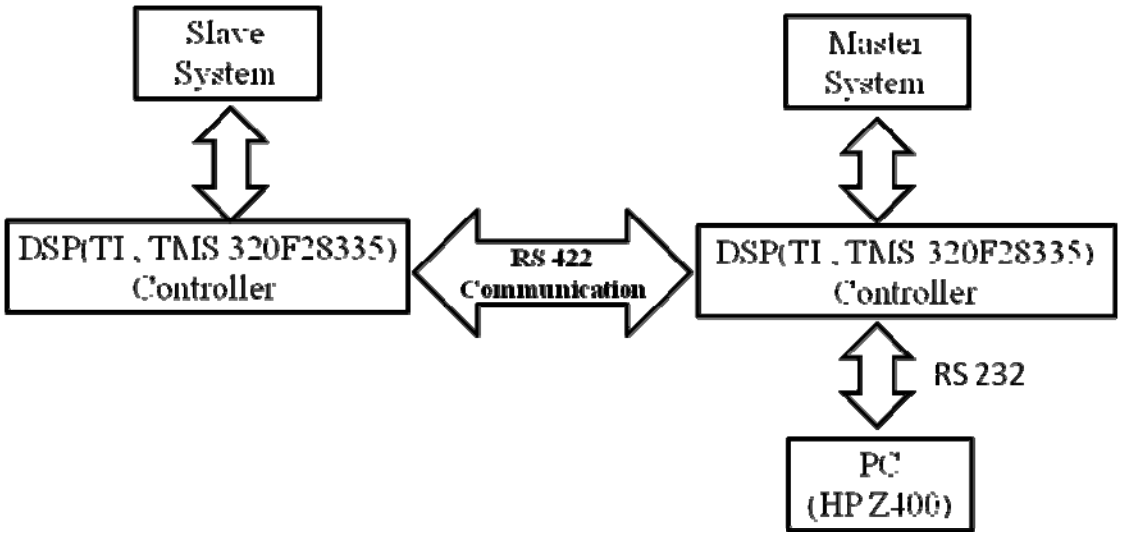


Figure 2- 7 The communication system based on RS 422

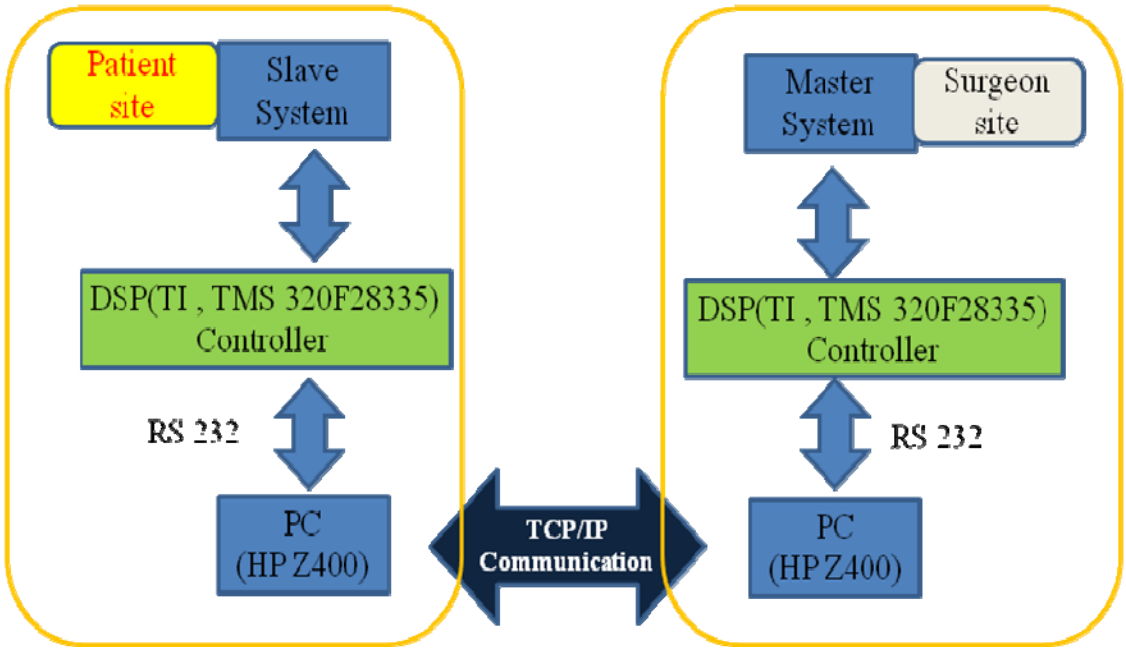


Figure 2- 8 The communication system based on internet

2.3 The design of control system

2.3.1 Modeling of the inserting motion

In order to keep the consistency and accuracy in inserting motion, meanwhile reducing the hysteresis in real time, we used the PID algorithm to resolve these problems. We establish a dynamic equation in inserting direction, is shown in Eq.2-1, numerical simulation experiments were carried out by Matlab.

$$F(t) = \mathrm{m}\,\ddot{x}(t) + c\,\dot{x}(t) + kx(t) \qquad \text{(Eq.2-1)}$$

Where $F(t)$ is the pull force, $x(t)$ is the displacement of master manipulator, $\dot{x}(t)$ is the velocity of master manipulator. Define $x_1(t) = x(t)$, $x_2(t) = \dot{x}(t)$ then

$$\begin{aligned} \dot{x}(t) &= AX(t) + Bu(t) \\ y(t) &= CX(t) \end{aligned} \qquad \text{(Eq.2-2)}$$

Where $X(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}, A = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix}, C = \begin{bmatrix} 1 & 0 \end{bmatrix}$

m is the quality of master manipulator, c is the viscous damping coefficient, k is the stiffness.

When the operator operates the right handle of master manipulator, the load cell will get the pull force from the operator and the resistance

force that is created by load cell part and stiffness. Based on this relationship, we can find that the whole operation is defined that the velocity of operator manipulation is as same as the step motor. To avoid the excessive overshoot, we use PID algorithm to control the output which can keep the consistence with the operation of operator. The control strategy is shown as following:

$$u(t) = k_p e(t) + k_i \int_0^t e(t)d(t) + k_d \frac{de(t)}{dt} \qquad \text{(Eq.2-3)}$$

As on the master side, based on the input and output of the step motor, we used the same PID control strategy on the slave side to control the consistency and response of the slave mechanism during insertion.

2.3.2 Numerical simulation for the inserting motion

Figure 2-9 shows the control principle diagram of the inserting motion by Matlab Simulink. Figure 2-10 shows the developed PID controller for the insertion control. During the simulation of an operation, the results of the system response with sinusoid input signal and the system response with step input signal are shown in Figure 2-11 and Figure 2-12. The blue line shows the disturbing signal and the red line shows the tracking signal. From the diagram we can see that the manipulation of mechanic system can follow well with operator operations. Between 0.1s and 0.2s, the system can keep up with the

actions of neurosurgeon. Despite there has a certain of overshoot, it can catch up as soon.

Parameters of the manipulation system are as following:

$m = 2kg, c = 0.02N/(m/s), k = 10N/m$

Parameters of the developed PID controller are as following:

$k_p = 30, k_i = 400, k_d = 1$

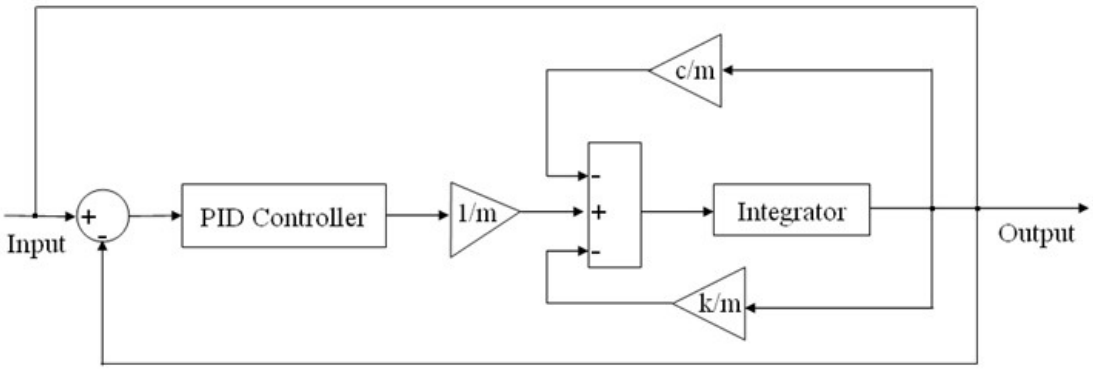


Figure 2- 9 The control of the inserting motion

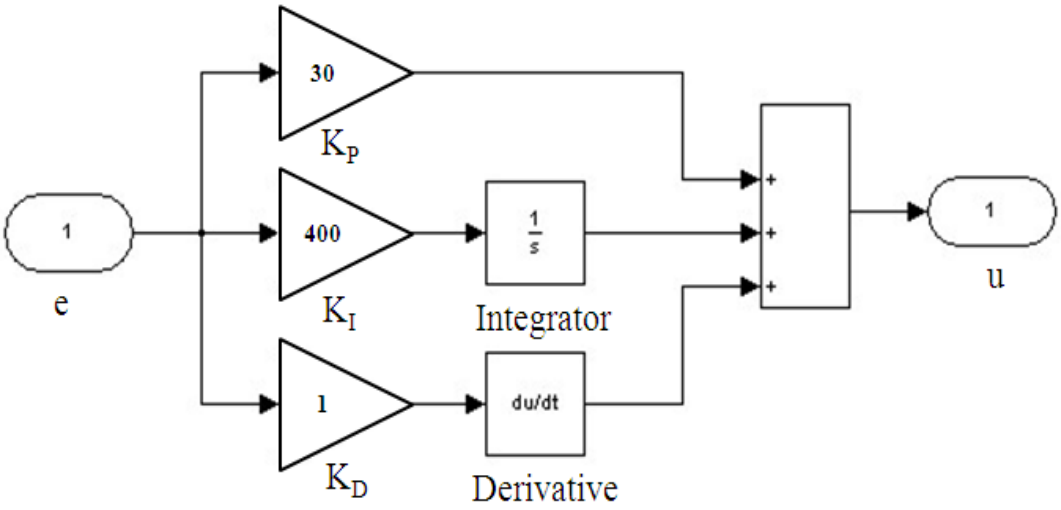


Figure 2- 10 The developed PID controller (Insertion)

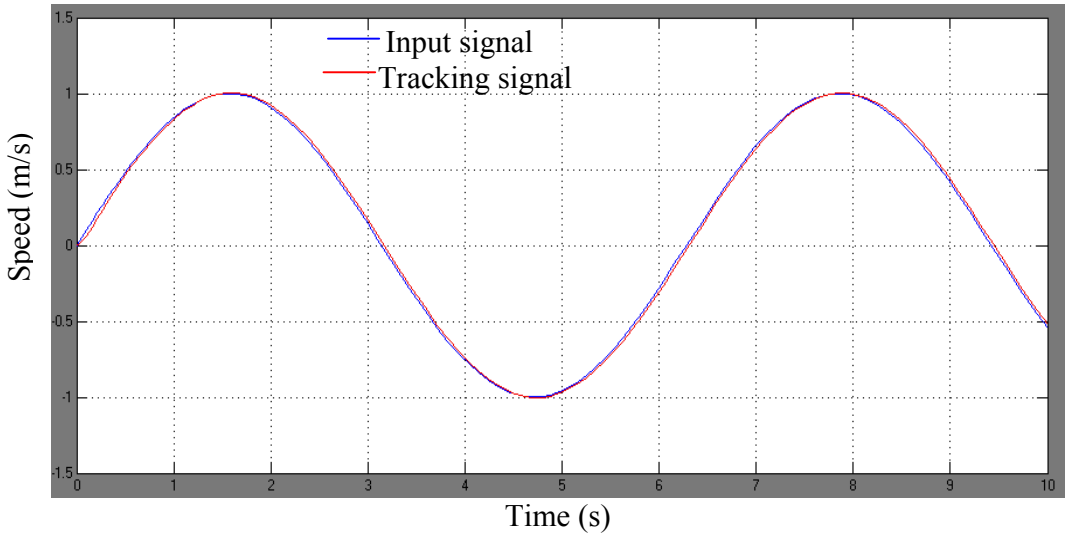


Figure 2- 11 System response with sinusoid input signal

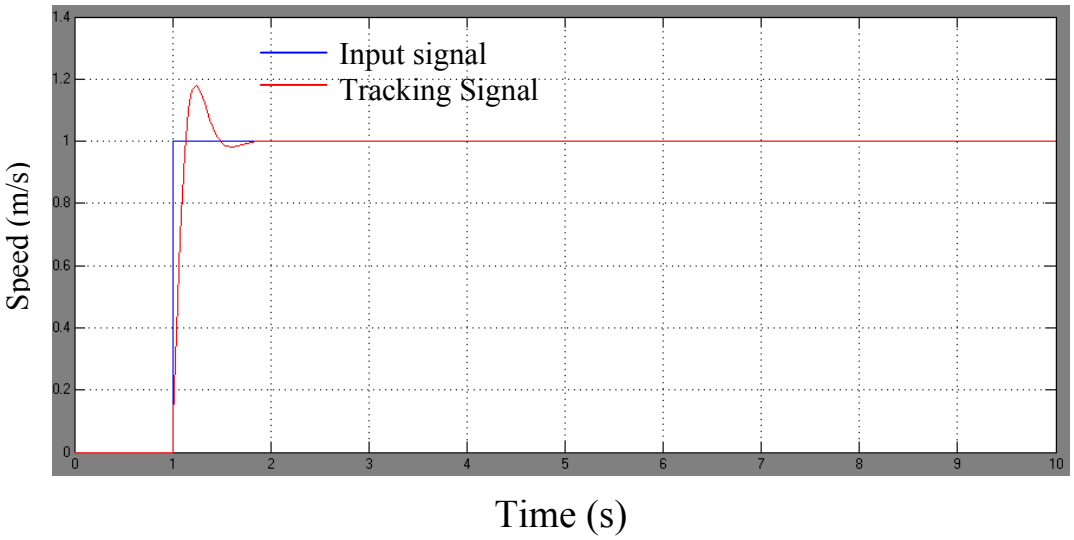


Figure 2- 12 System response with step input signal

2.3.3 Modeling of the rotating motion

An equation can be established using torque balance in rotating motion of master side, which is shown in Eq.2-4, m is the quality of manipulating system (on the movement stage of master side) , c is the viscous damping coefficient, $m = 2kg, c = 0.02N/(m/s)$, θ is rotating angle, u(t) is the variation of torque, which consists of torque of maxon motor and torque of right handle, $\dot{\theta}$ is the angular velocity, $\ddot{\theta}$ is the angular acceleration,

$$m\ddot{\theta} + c\dot{\theta} = u(t) \qquad \text{(Eq.2-4)}$$

The equation of state was established, which is shown in Eq.4-5, define $\theta_1(t) = \theta(t), \theta_2(t) = \dot{\theta}(t)$.

$$\begin{aligned} \dot{x}(t) &= AX(t) + Bu(t) \\ y(t) &= CX(t) \end{aligned} \qquad \text{(Eq.2-5)}$$

Where, $X(t)=\begin{bmatrix}\theta_1(t)\\ \theta_2(t)\end{bmatrix}, A=\begin{bmatrix}0 & 1\\ 0 & -\frac{c}{m}\end{bmatrix}, B=\begin{bmatrix}0\\ \frac{1}{m}\end{bmatrix}, C=\begin{bmatrix}1 & 0\end{bmatrix}$

The controlling strategy is shown in Eq.2-6,

$$u(t)=k_p e(t)+k_i\int_0^t e(t)d(t)+k_d\frac{de(t)}{dt} \qquad \text{(Eq.2-6)}$$

Where $k_p=480, k_i=7, k_d=28$

As on the master side, based on the input and output of the maxon DC motor, we used the same PID control strategy on the slave side to ensure the consistency and response of the slave mechanism for rotation.

2.3.4 Numerical simulation for the rotating motion

We do the numerical simulation in rotating motion for the robotic catheter system. Figure 2-13 shows the step/sinusoid controlling principle, Figure 2-14 shows the developed PID controller, the numerical simulation result with step input signal is shown in Figure 2-15, and the result of numerical simulation with sinusoid input signal is shown in Figure 2-16. When the neurosurgeon begins to operate the handle of master side, the input signal of the system is step signal, when the system is stable, the input signal of system is sinusoid signal, so the

numerical simulation with step wave signal and sine wave signal was done.

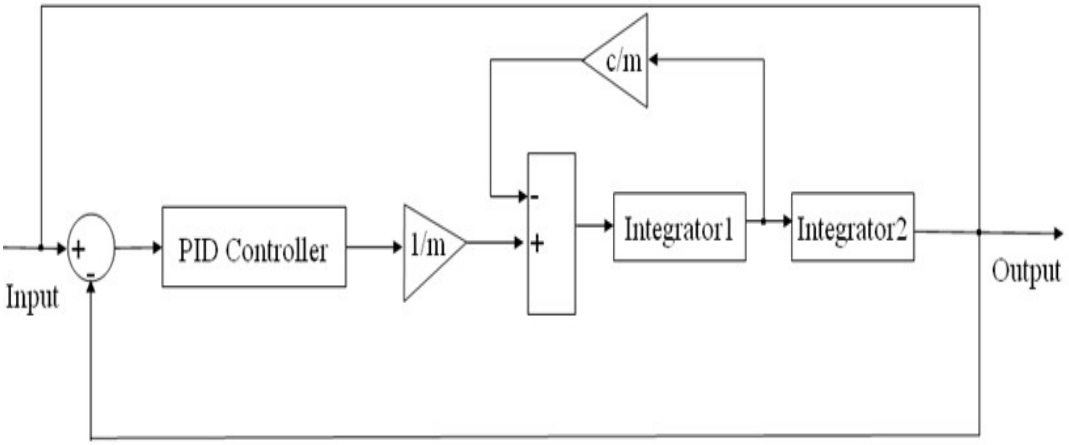


Figure 2- 13 The control of the rotating motion

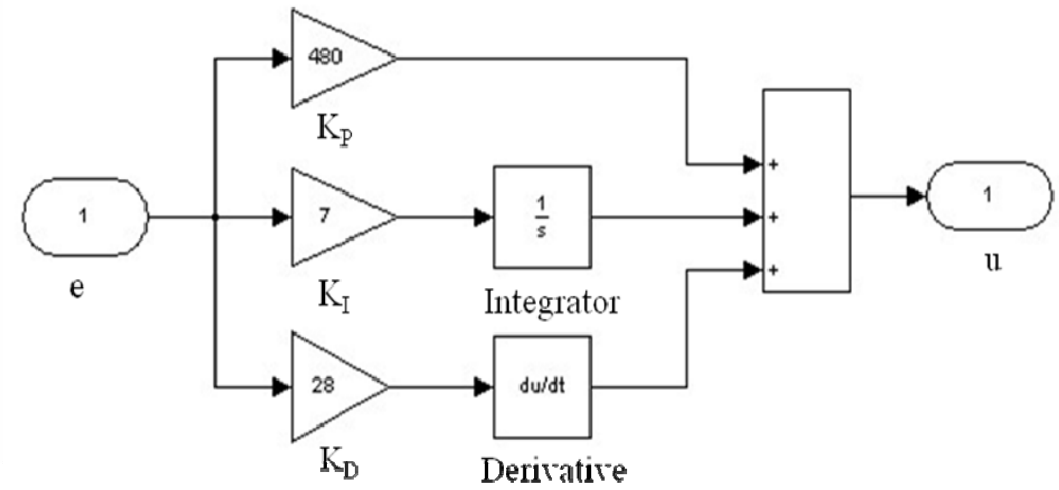


Figure 2- 14 The developed PID controller (Rotation)

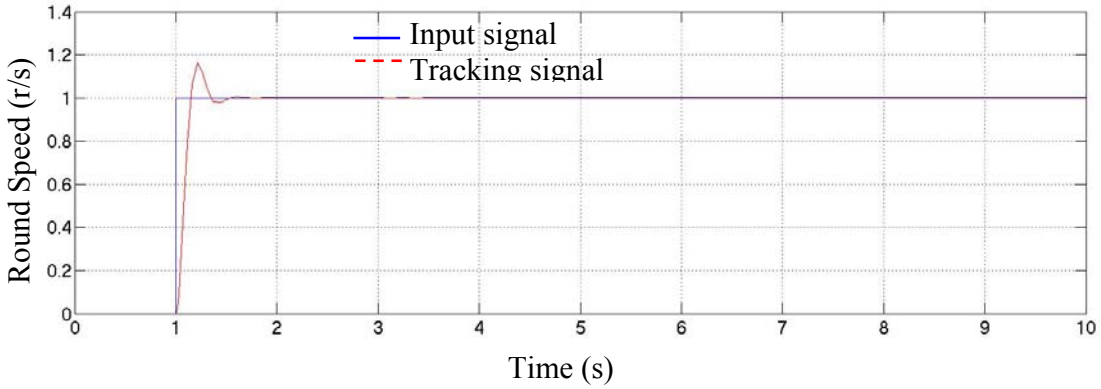


Figure 2- 15 System response with step input signal

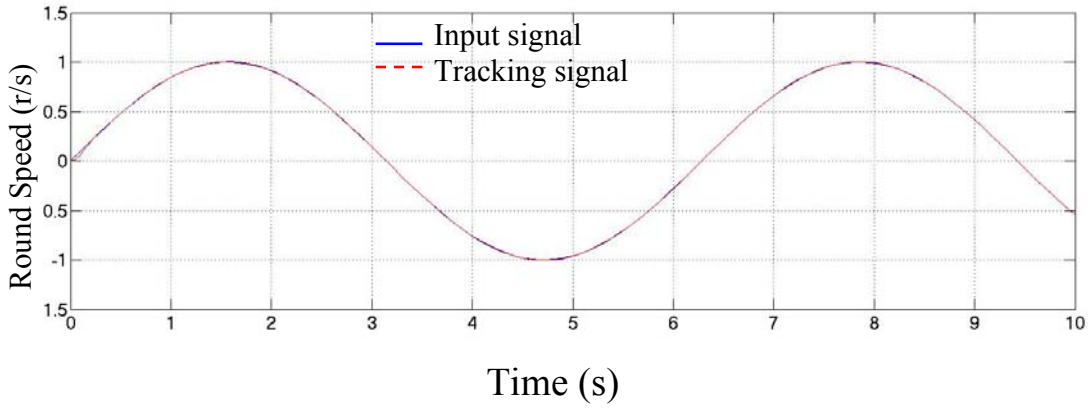


Figure 2- 16 System response with sinusoid input signal

2.4 Realization of the control system

Based on above control strategy, we build the hardware for the robot-assisted catheter system to realize the control system, the flow diagram of the control system both on the master side and the slave side is shown in Figure 2-17, the control system for the catheter graspers on the slave side is shown in Figure 2-18, the control circuit of the master

system is shown in Figure 2-19, the control circuit of the slave system is shown in Figure 2-20, both on the master system and slave system we used DSP (TI, TMS320F28335) as the control unit, the control circuit of catheter graspers is shown in Figure 2-21, in this circuit we used AVR (ATtiny 861) as the control unit.

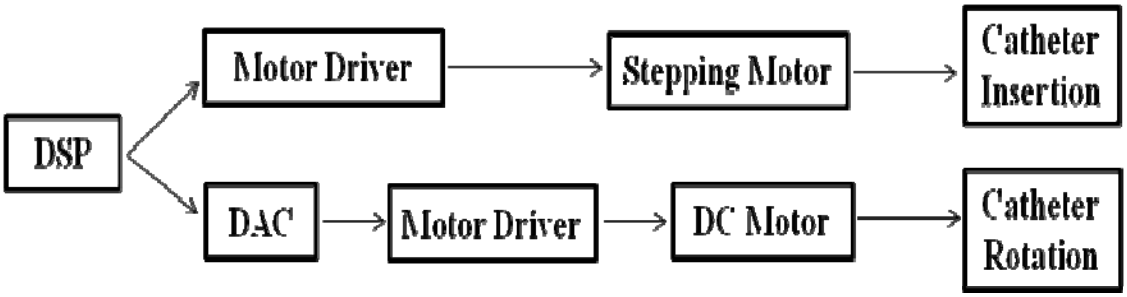


Figure 2- 17 The control system (both master side and slave side)

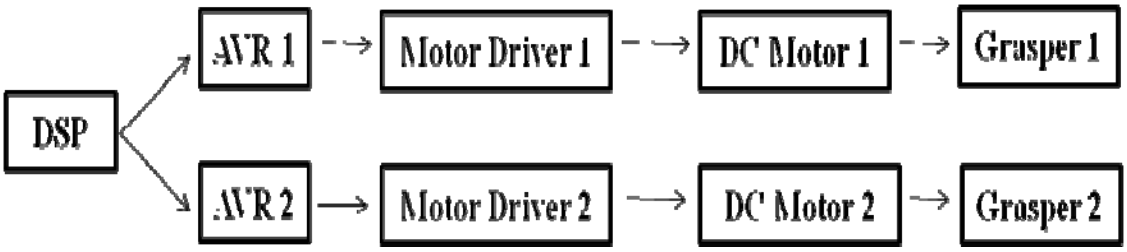


Figure 2- 18 The control system of catheter graspers

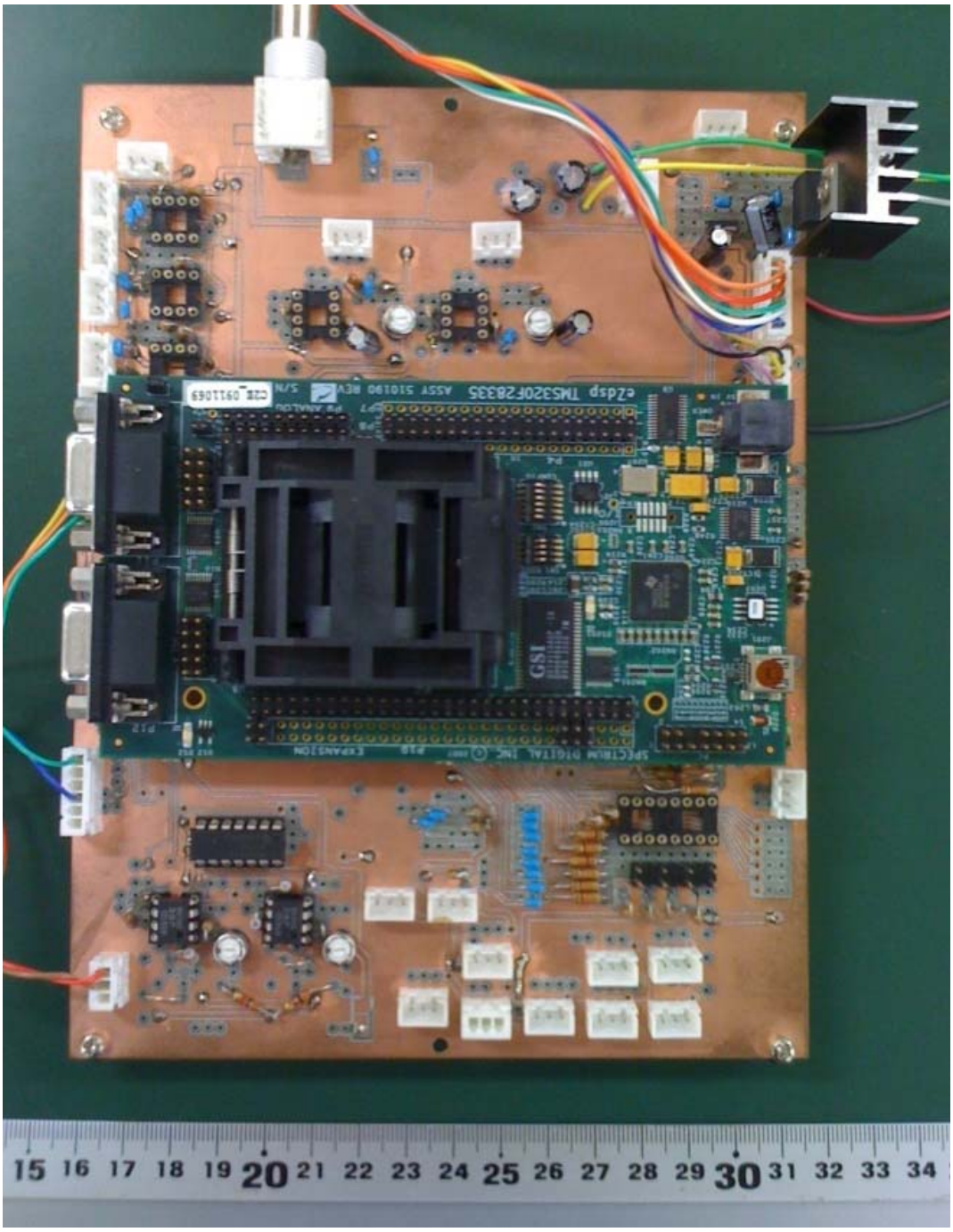


Figure 2- 19 The control circuit of the master system

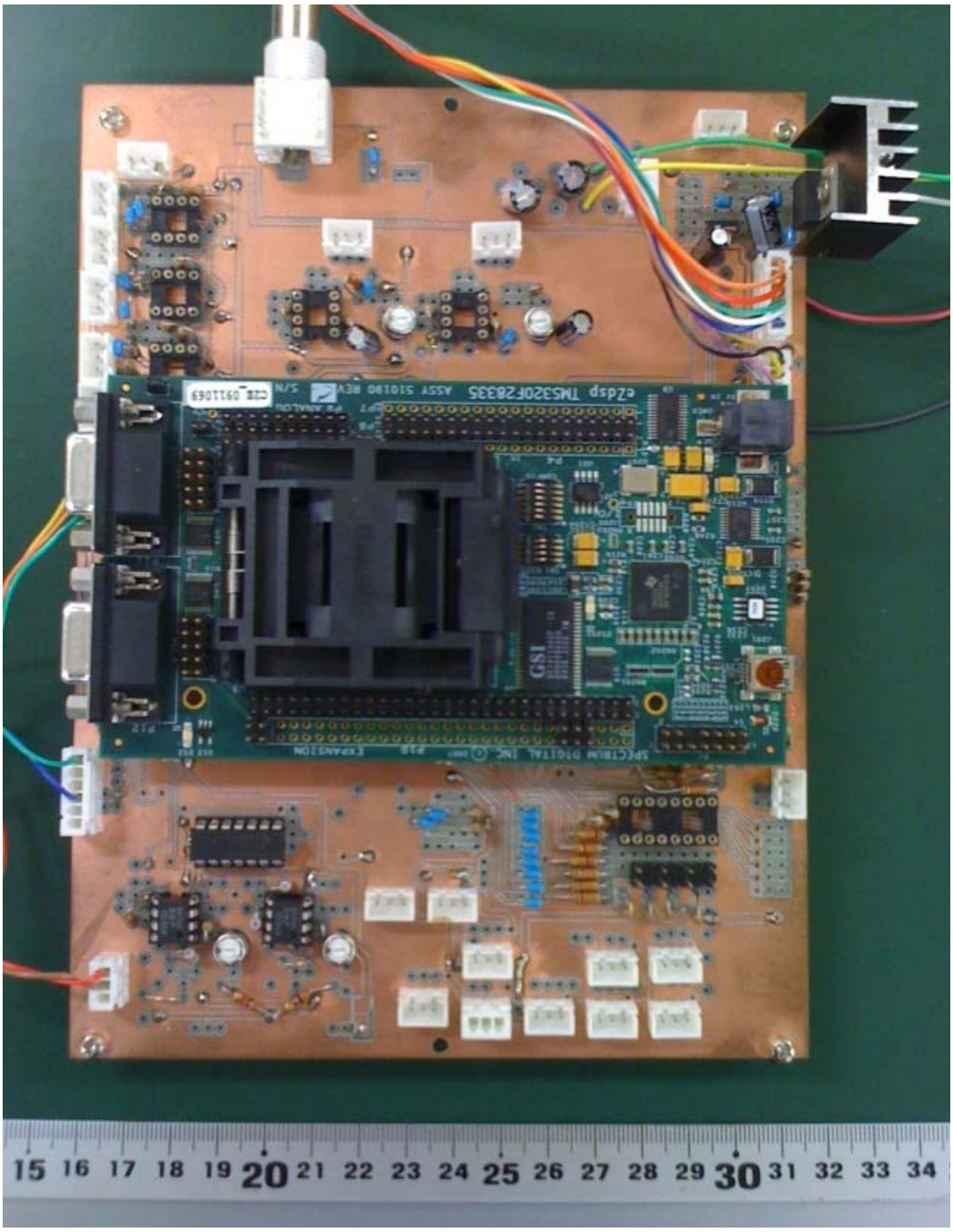


Figure 2- 20 The control circuit of the slave system

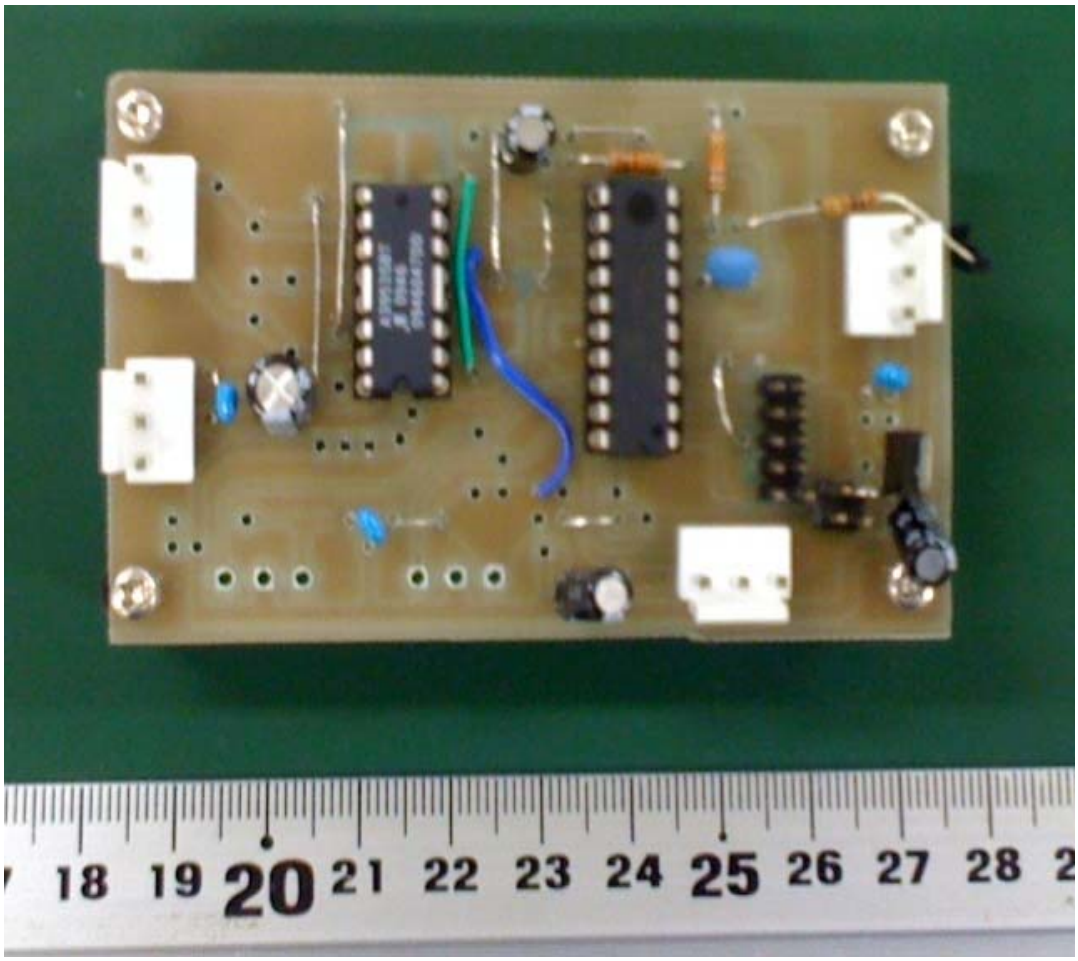


Figure 2- 21 The control circuit of catheter graspers

2.5 The design of the feedback system

2.5.1 Introduction

As we know, force information on the catheter is very important to insure the safety of an operation. Doctors always want to manage any details of an operation. The contact force between blood vessel and the catheter is a key detail. Stronger contact may get the blood vessel broken, so the method of getting the contact force information between the blood vessel and the catheter is urgent needed. There are two kinds of the contact force. One is generated when the side of the catheter contacts to the blood vessel, and the other one is generated when the end-tip contacts to the blood vessel. Both of them could cause damage during the operation. The conceptual diagram for measuring contact force is shown in Figure 2-22.

In this chapter, we developed a kind of feedback system to enhance the safety during operation, it includes two parts, one is the force feedback system, which provides the force feedback to neurosurgeon, let neurosurgeon to feel the contact force between catheter and blood vessel during operation. The other part is the visual feedback system, which provides the visual information to neurosurgeon during operation. We will introduce them in detail in this chapter.

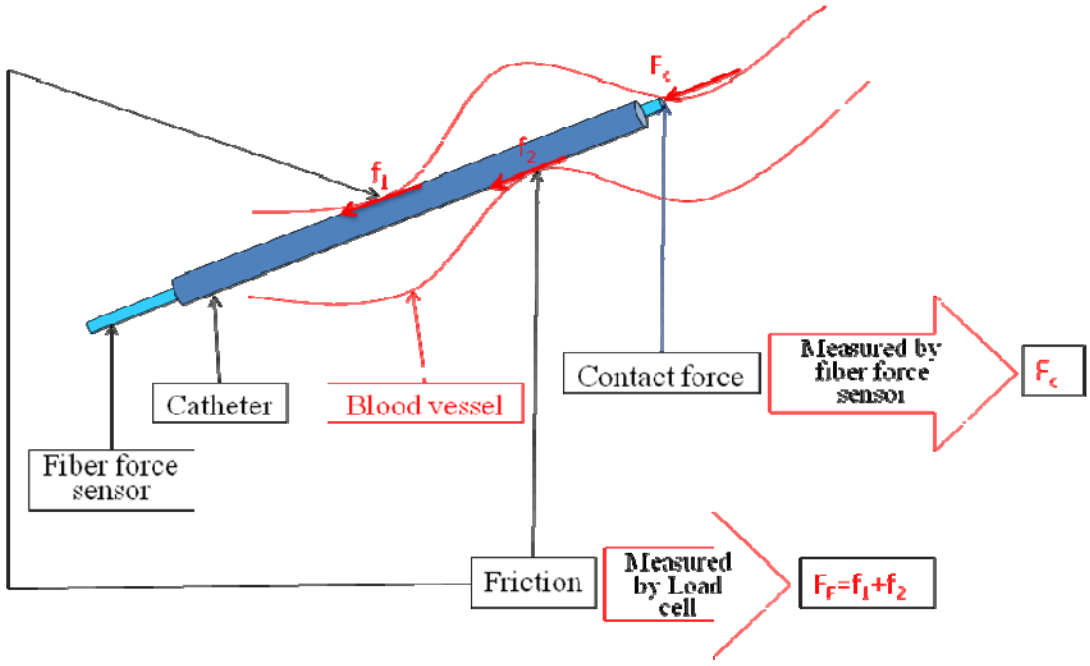


Figure 2- 22 Measuring contact force and friction

2.5.2 Realization of the force feedback

We all know that the force information on the catheter is very important during the vascular interventional surgery, we developed a force feedback system for the developed robot-assisted catheter system, firstly, we designed a mechanism to measure the friction between catheter and blood vessel, the friction is also the neurosurgeon's feeling force in real clinical vascular interventional surgery, further more, it is very important in order to let neurosurgeon feel the inserting force during operation by using the developed robot-assisted catheter system, the Figure 2-23 shows the feedback force measuring mechanism, the

mechanism is assembled with the slave manipulator, the catheter supporting frame is linked to the load cell which is used to measure the inserting force on the catheter during operation. When the catheter grasper clamped the catheter, the catheter and the catheter supporting frame keep the same moving mode, it has no movement between them, further more, if the force is applied on catheter, the force will be transmitted to the catheter supporting frame, then the force was measured by the load cell which is linked to the catheter supporting frame, the measured force can be transmitted to the slave system, then transmitted to the master system and generated a haptic feedback to the neurosurgeon, the conceptual diagram of the transmission of the force feedback is shown in Figure 2-24.

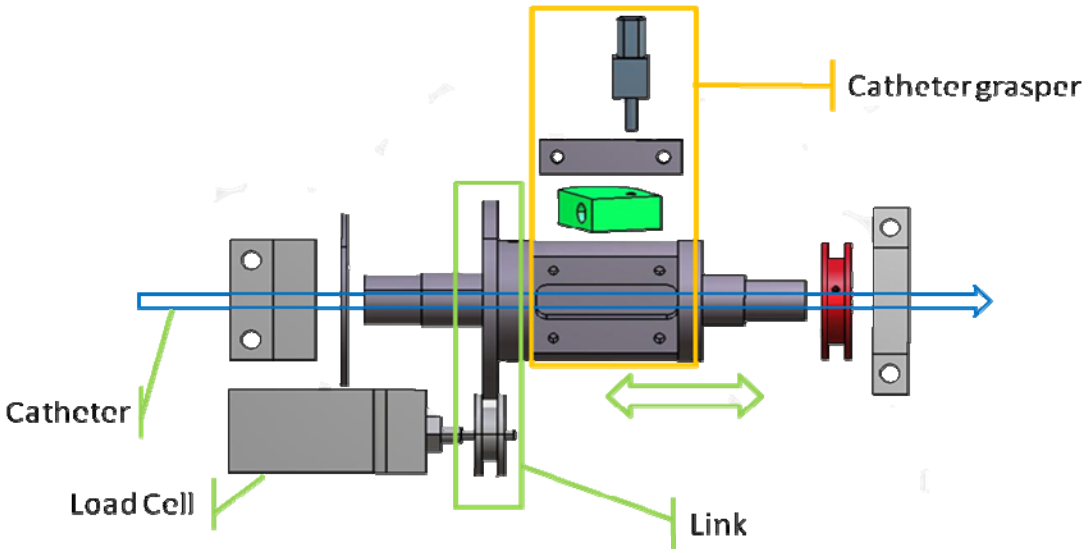


Figure 2- 23 The feedback force measuring mechanism

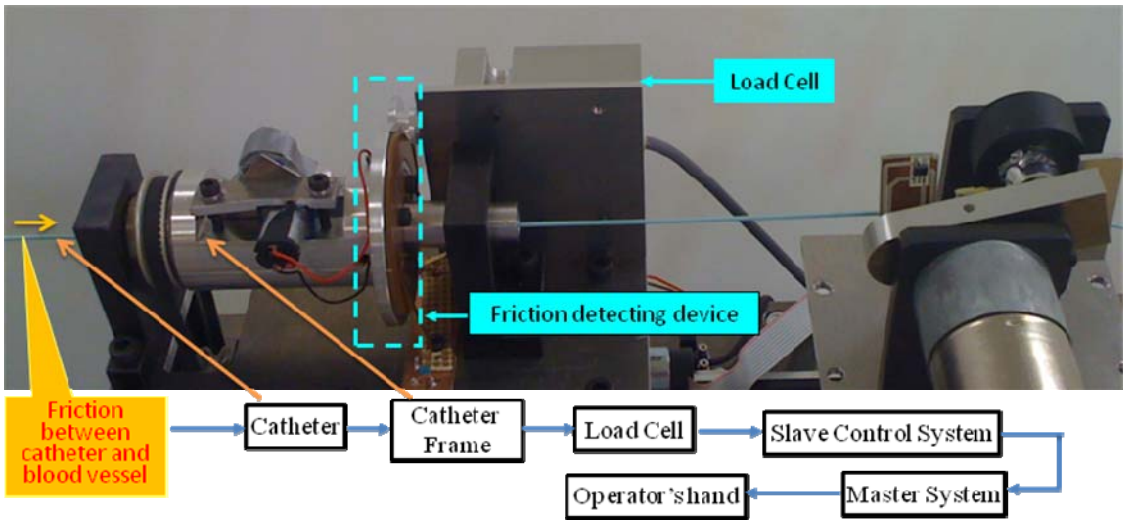


Figure 2- 24 The transmission of force feedback

2.5.3 Realization of the visual feedback

The contact force between catheter tip and blood vessel and the situation of the operational scene is also important excepting the operating force information between catheter and blood vessel during the vascular interventional surgery, so how to get them and feedback them to the neurosurgeon is essential, we designed a visual feedback system to solve it. We used an IP camera which is shown in Figure 2-25 to monitor the situation the operational scene, we developed a user interface to monitor the contact force between catheter and blood vessel, the photograph of the developed user interface is shown in Figure 2-26, the Figure 2-27 shows the transmission of visual feedback signal. As well as the force feedback, the visual feedback signal can be also transmitted to the neurosurgeon during the operation.

Figure 2- 25 The IP camera for monitoring situation of operation (VIVOTEK FD8133V)

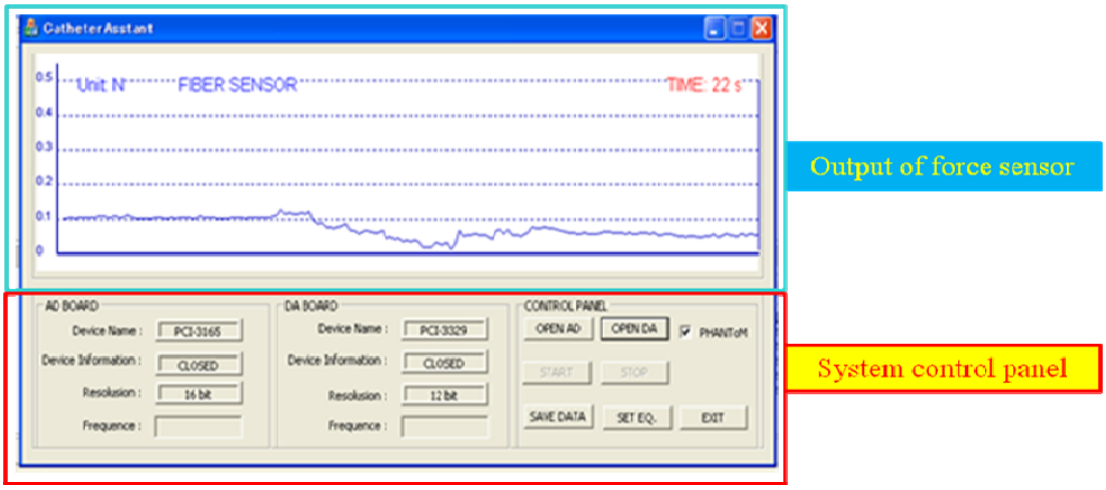


Figure 2- 26 The user interface for monitoring contact force

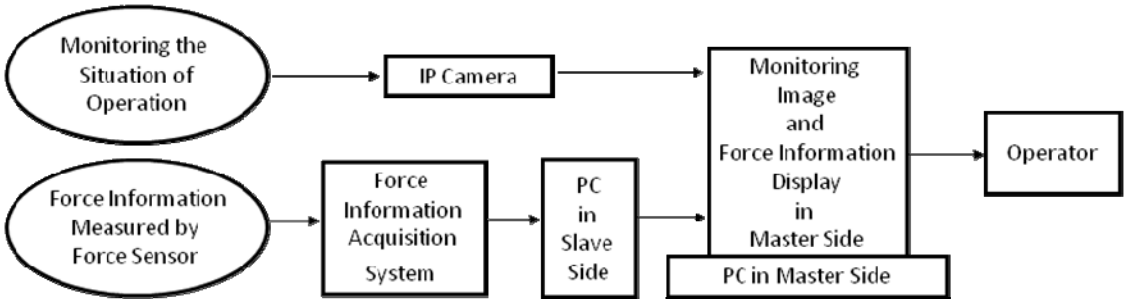


Figure 2- 27 The transmission of visual feedback signal

2.6 The performance evaluation

2.6.1 Introduction

In order to verify the validity of the developed robot-assisted catheter system, we did the performance evaluation for the developed robot-assisted catheter system, the performance evaluation consists of evaluation of the master manipulator, evaluation of the slave manipulator, and the evaluation of the tracking performance, after the performance evaluation, we give the results for the evaluations, in this chapter, we will introduce the evaluating method and give the evaluating results, finally, we analyzed and discussed the evaluating results in detail.

2.6.2 Evaluation of the master manipulator

To evaluate the measurement precision of controller in master side, two experiments were carried out. One is to evaluate the measurement precision of axial movement and the other one for radial movement.

The Figure 2-28 shows the evaluating system of the master manipulator.

For the first experiment, we pulled and pushed the handle to make the movement stage go forward and backward for about 2 minutes, then measured the axial displacement by a laser sensor (KEYENCE Inc., LK-500, high precision mode, 10μm/mV). An A/D convert board (Interface Inc., PCI3329) was applied to get the displacement data to the computer (HP, z400), the sampling frequency was 100Hz. And at the same time, the measurement data of the controller was send to the same computer by serial port, the sampling frequency of the controller was set to 100Hz, baud rate of the serial port was set to 9600. After comparing these two groups of data, the axial measurement precision of the controller could be evaluated. The Figure 2-29 shows the evaluated method for axial precision. In the second experiment, radial measurement precision is evaluated. As well as the first experiment, we rotated the handle in clock wise and anti clock wise for 2 minutes, then measure the rotation angle by outside sensor and by the encoder couple to the handle of controller. A 3-axis inertial sensor (Xsens Inc. MTx, resolution 0.1 deg) is fixed on the handle to get the rotation angle. The sampling frequency of the inertial sensor is set to100Hz as the same as the controller. Sampling data of the controller was send to the computer by serial port, the Figure 2- 30 shows the evaluated method for radial precision.

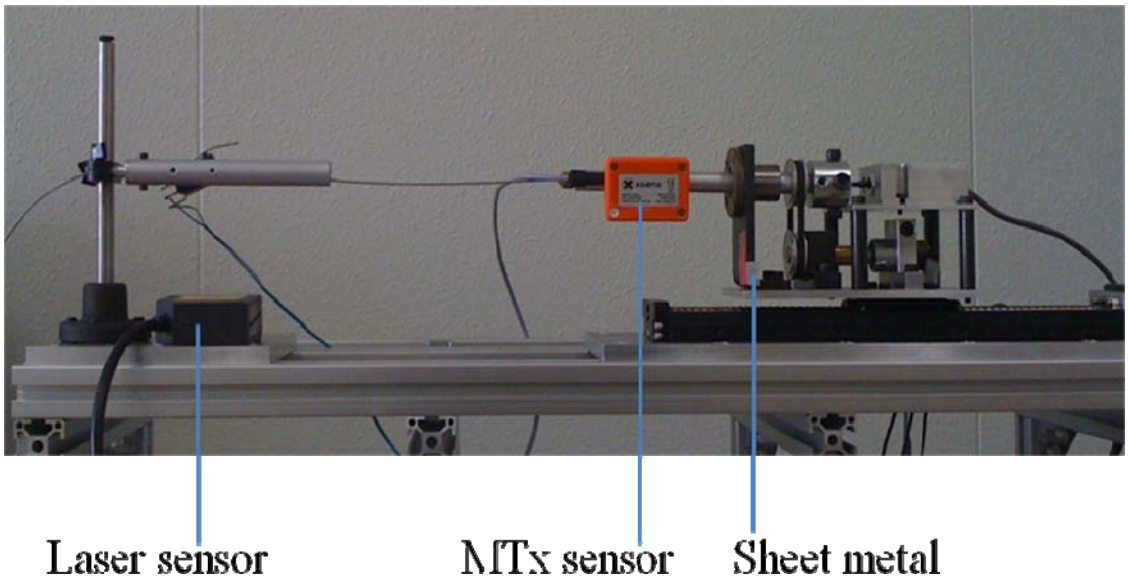


Figure 2- 28 The evaluating system of the master manipulator

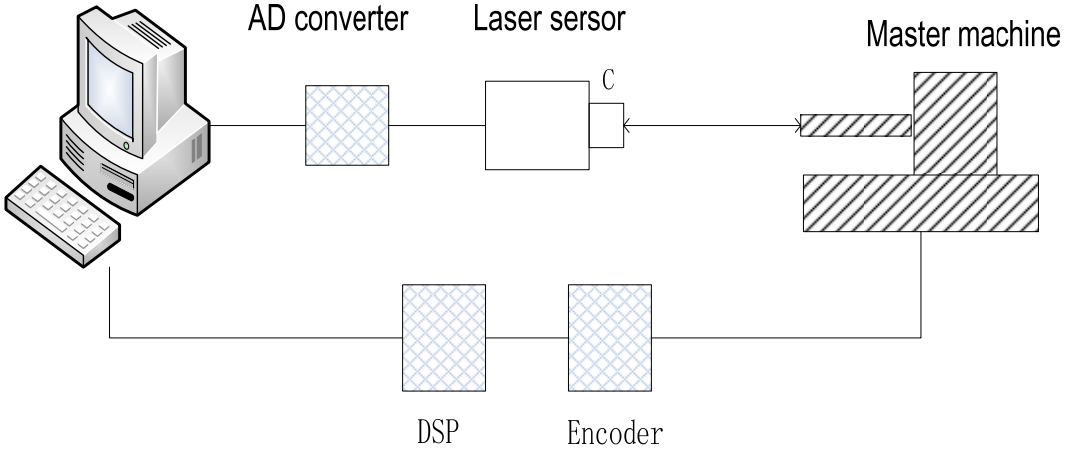


Figure 2- 29 The evaluated method for axial precision

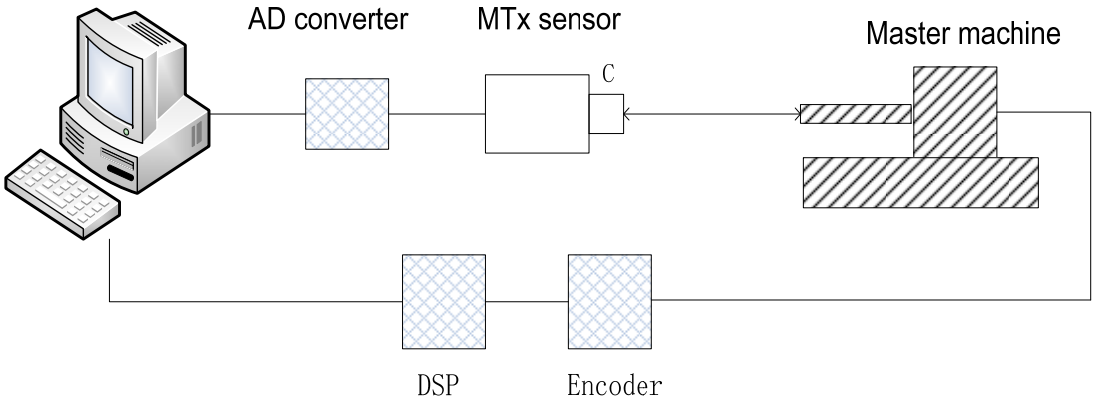


Figure 2- 30 The evaluated method for radial precision

2.6.3 Evaluation of the slave manipulator

To evaluate the measurement precision of the slave manipulator, the Figure 2- 31 shows the evaluating system of the master manipulator, as well as evaluation of the master manipulator, there are also two experiments for evaluating the slave manipulator. The first experiment is to evaluate the axial movement precision. The slave manipulator was programmed to move along a reciprocating trace in axial. Then the actual displacement of the machine is measured by laser sensor. The second experiment is to evaluate the radial precision. We programmed the machine to make the catheter frame rotated back and forth. The actual rotation angle is measured by inertial sensor. In these two experiments, actual data and theoretical date were compared to obtain the precision of the slave manipulator.

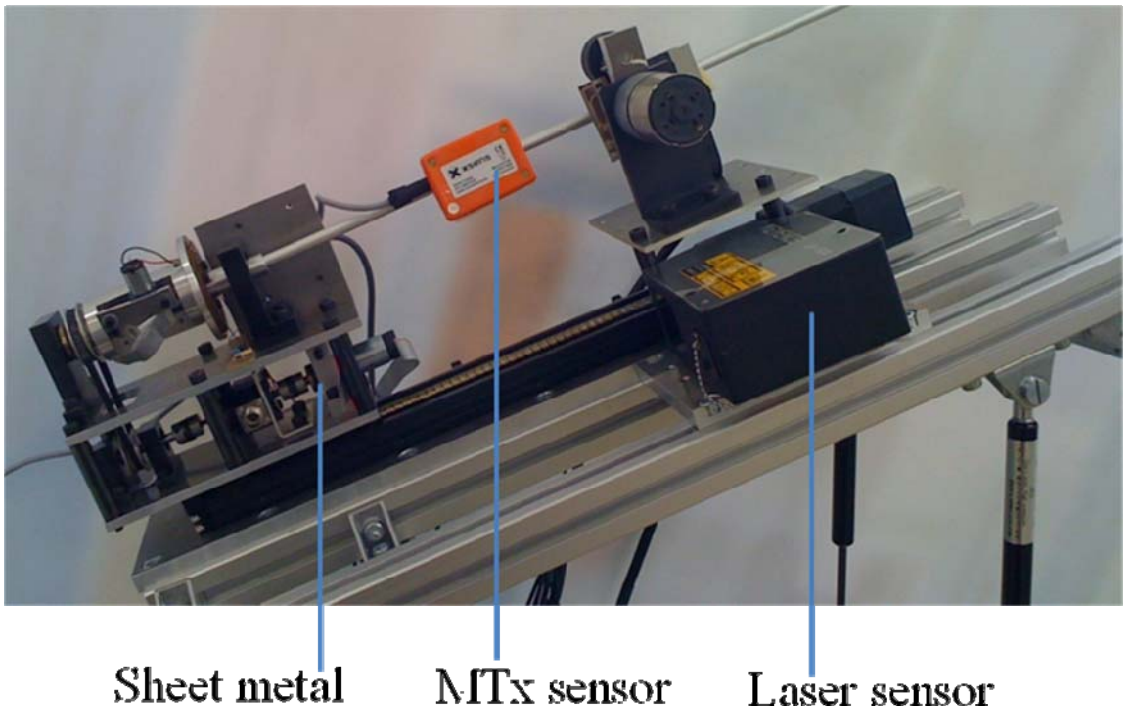


Figure 2- 31 The evaluating system of the slave manipulator

2.6.4 Evaluation of the tracking performance

In this part, we did experiment to evaluate the tracking performance. In the experiment, we ignored the time delay between the master manipulator and slave manipulator. We just evaluate the tracking performance of the axial displacement and rotation. Firstly, we pulled and pushed the controller in master side from right end to left end then go back, we kept the axial displacement by laser sensor in both master manipulator and slave manipulator. To compare these two groups of data, the synchronization of the master manipulator could be obtained. The evaluating system of the tracking performance is shown in Figure

2-32. This procedure was carried out for four times. Similarly, the synchronization of rotation could be obtained.

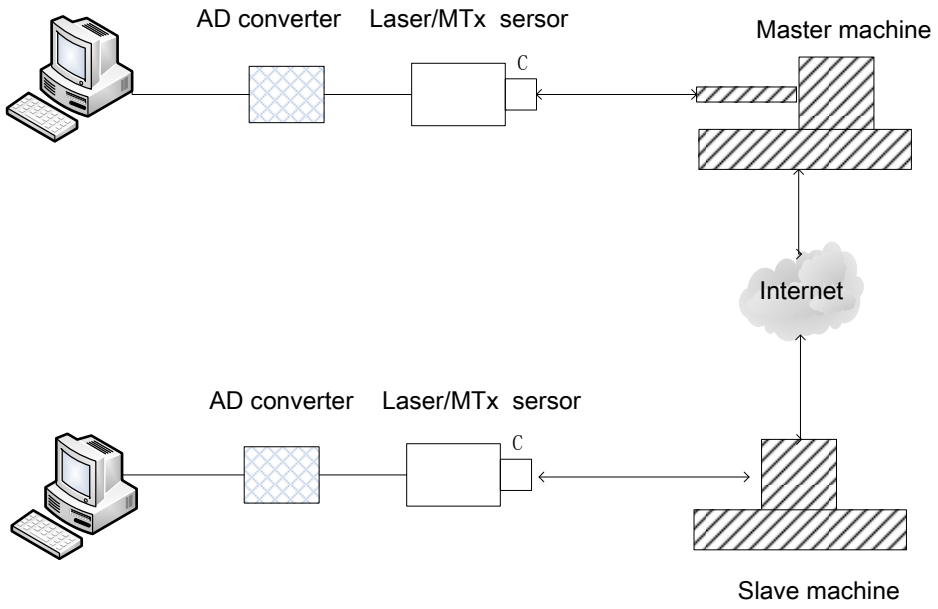


Figure 2- 32 The evaluating system of the tracking performance

2.6.5 Evaluated results

Both on the master side and the slave side, mean value and variance of the error between the laser sensor and the master manipulator are calculated. Similarly, we got mean value and variance of the error between the actual value and the theoretical value. The results are listed in Table 2-1 and Table 2-2. The precision evaluated result of the master manipulator is shown in Table 2-1, the precision evaluated result of the slave manipulator is shown in Table 2-2, Figure 2-33 shows the axial motion tracking curve, from Figure 2-33 we can know the movement

curve of the master manipulator is fast than slave manipulator, it indicated that the lag exists. The lag time is going to be measured in the future. Figure 2-34 shows the radial motion tracking curve, as well as the master manipulator, lag could be found in Figure 2-34.

The Figure 2-33 shows the evaluated results of synchronization in axial direction. The master manipulator and slave manipulator are controlled by DSPs, DSPs communicate with each other with the Serial Peripheral Interface (SPI) port and RS422. The distance between the master manipulator and slave manipulator is about 5m. Based on the numbers of communication data and the distance, the SPI communication is fast enough to make the lag below 1ms in theory. However, from Figure 2-33 we can know the movement curve of the master manipulator is fast than slave manipulator, it indicated that the lag exists. The lag time is going to be measured in the future. We repeated the procedure for 4 times, Figure 2-35 shows a statistic result to evaluate the tracking precision. We have to align the data of two sides. The movement value is about 20ms. From the result in Figure 2-35, the axial error between master manipulator and slave manipulator are below 0.6mm, the standard deviation is below 1mm. Figure 2-34 shows the rotation tracking error. As well as the master manipulator, lag could be found in Figure 2-34. Figure 2-36 shows the statistics of the evaluated results. The mean error of the rotation is below 1.5^{o}, but with a large standard deviation. It means a not stable radial motion. The

mean overshoot mainly caused by the electromagnetic interface to the AD convertor which is used to drive the motor. The control circuit should be redesigned. And with different operator the radial tracking error is different because of the different rotation speed.

Table 2- 1 Precision evaluation result of master manipulator

Master manipulator	mean	variance
Axial (mm)	0.35(0.35%)	0.025
Radial (deg)	3.1(1.72%)	2.1

Table 2- 2 Precision evaluation result of slave manipulator

Slave manipulator	mean	variance
Axial (mm)	0.23(0.23%)	0.04
Radial (deg)	2.2(1.22%)	3.0

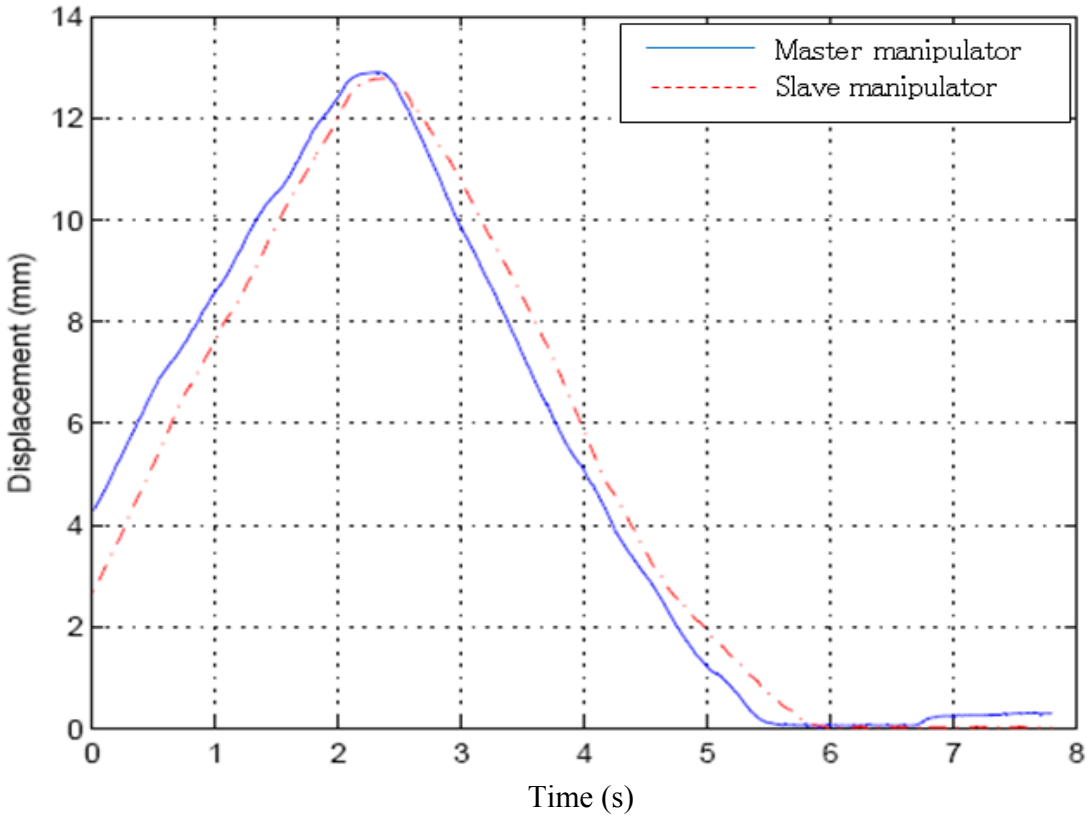


Figure 2- 33 The axial motion tracking curve

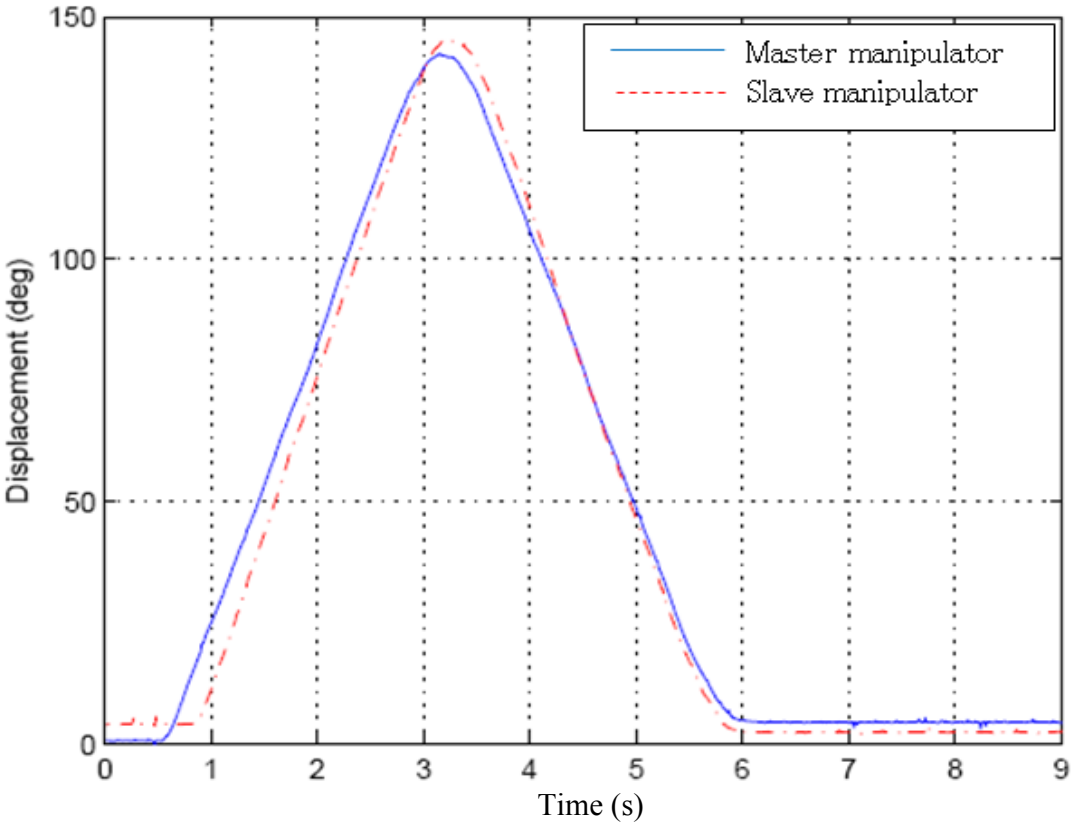


Figure 2- 34 The radial motion tracking curve

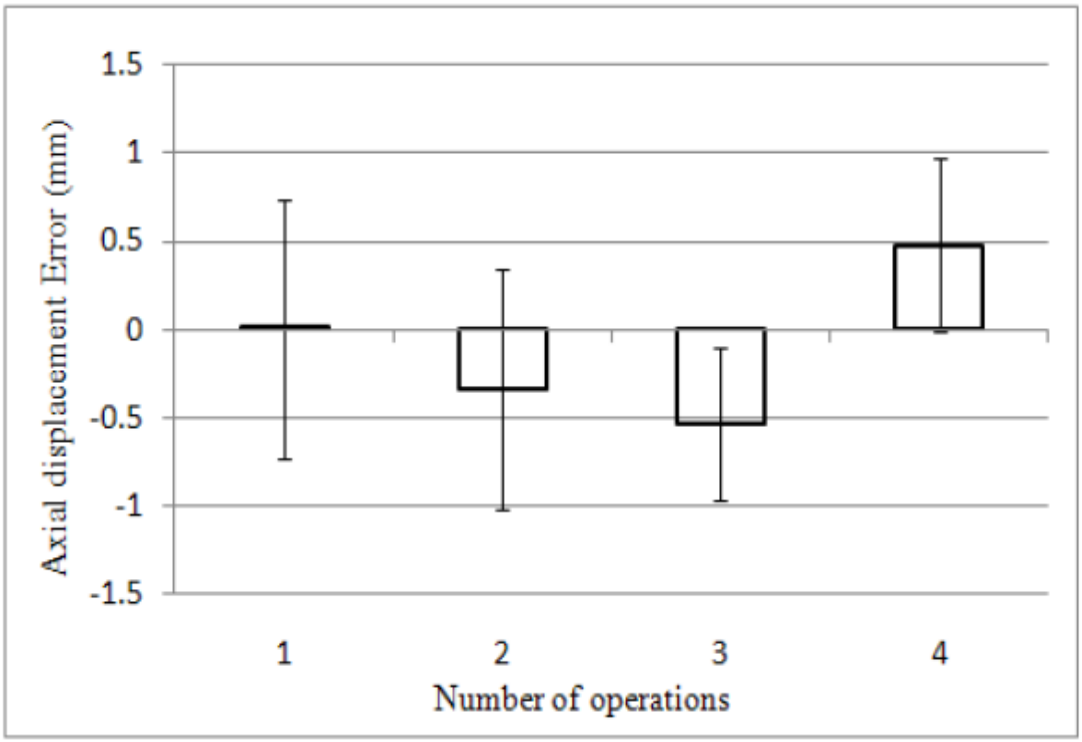


Figure 2- 35 The axial notion tracking statistic

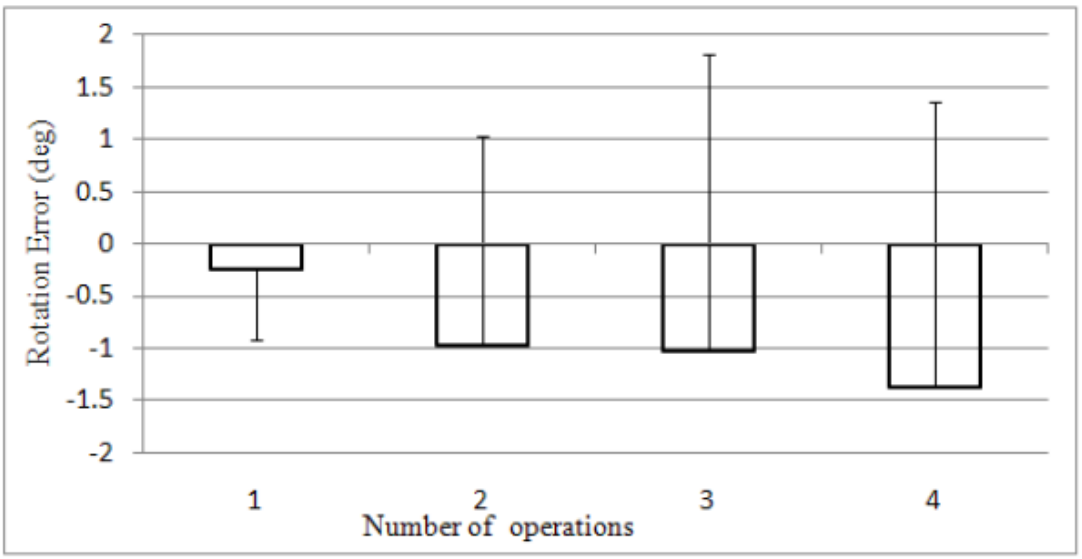


Figure 2- 36 The radial motion tracking statistic

2.7 Summary

In this chapter, firstly, we design the mechanisms which constructed the structure of the master-slave for the robot-assisted catheter system in order to solve the aforementioned problems instructed in chapter 1. The master manipulator simulates the neurosurgeon's operating skill and the slave manipulator driving the catheter to insert and rotate was developed in the slave system. We realized the communication based on RS 422 and internet between master system and slave system.

Secondly, we introduced the control system and realization of the hardware for the robot-assisted catheter system. We used a kind of PID control strategy to this system, and did the numerical simulation based on the developed PID control strategy both for the inserting motion and the rotating motion.

Then, we developed a feedback system including force feedback and visual feedback for this system. By using the developed feedback system, the neurosurgeon can know the contact information between catheter and blood vessel. According to the feedback system, the safety of the operation could be enhanced. In addition, the feedback system can replace the X-Ray, it can eliminate the radiation from X-ray to neurosurgeon.

Finally, the evaluation performance has been completed both master and slave. The evaluated results listed in Table 2-1 and Table 2-2 show

the precision of the catheter system, the precision of master manipulator measured is 0.35mm, and it has a very small variance. Rotation precision of the master manipulator is 3.1^{o} with the variance of 2.1^{o}. In the slave side, the precision of axial direction is 0.23mm and the rotation precision is 2.2^{o} with a variance of 3.0^{o}.

Chapter 3

Experiment by PID fuzzy control

3.1 Introduction

Increasing demands for flexibility and fast reactions in control method, fuzzy control (FC) can play an important role because the experience of experts can be combined in the fuzzy control rules to be implemented in the systems. We present a practical application of a fuzzy PID control to this developed system during the remote operations and compare with the traditional PID control experimentally. The feasibility and effectiveness of the control method are demonstrated. The synchronous manipulation performance with the fuzzy PID control is much better than using the conventional PID control method during the remote operations.

3.2 Fuzzy Control

Traditional control design methods use mathematical models of a system and its inputs to design controllers that analyse their effectiveness. FC uses fuzzy sets and fuzzy inference to derive control

laws in which no precise model of the system exist, and most of the a priori information is available only in qualitative form. The basic idea of FC is to make use of expert knowledge and experience to build a rule base with linguistic rules. A fuzzy rule is a conditional statement, expressed in the form IF Then. There are two difficulties in designing any fuzzy logic control system: (1) the shape of the membership functions and (2) the choice of the fuzzy rules.

The proposed FC system is shown in Figure 3- 1 and the fuzzy controller operation, in general, is typically divided into the following three categories: fuzzification, inference engine and defuzzification. The fuzzification block means that real world variables are translated in terms of fuzzy sets. In a fuzzy inference engine, the control actions are encoded by means of fuzzy inference rules. The results of the fuzzy computations are translated in terms of real values for the fuzzy control action in the defuzzification block.

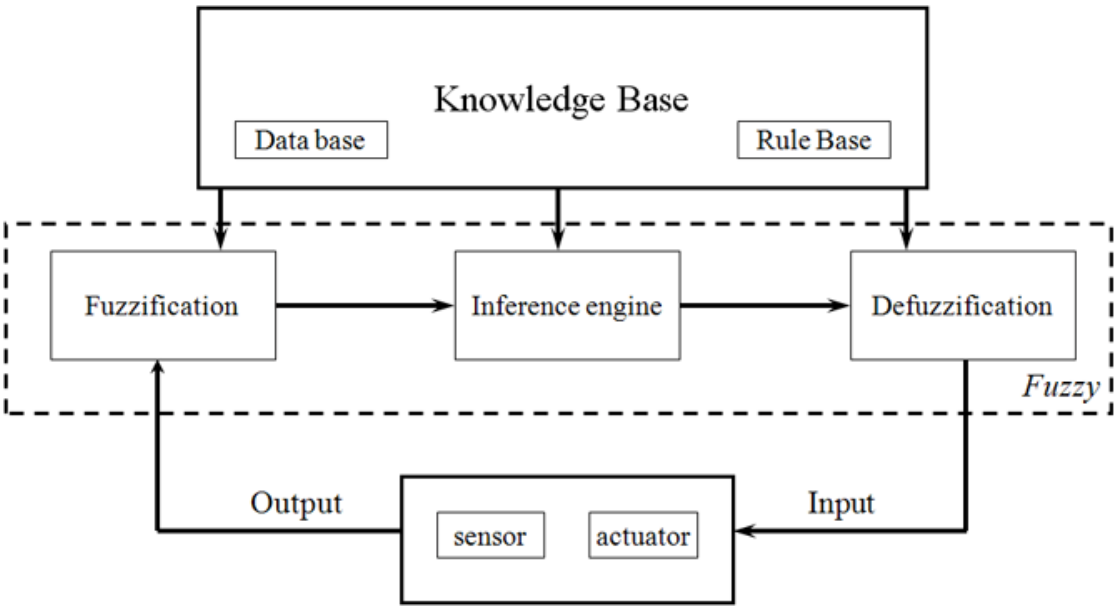


Figure 3- 1 General structure of fuzzy control

3.2.1 Input variables and normalization

The first step in FC is to take physical values of the system variables from the A/D converter and map them into a normalized domain. A fuzzy logic (FL) controller usually uses the error (e(k)) and the change of error (ce(k)) as the input variables.

$$e(k) = y_r(k) - y_m(k) \quad \text{(Eq.3-1)}$$

$$ce(k) = \frac{e(k) - e(k-1)}{T} \quad \text{(Eq.3-2)}$$

Where $y_r(k)$ and $y_m(k)$ are the reference and output, respectively, T is the sampling period, and $e \in [-l_e, l_e]$, $ce \in [-l_d, l_d]$, T , l_e, $l_d \in R^+$, R^+ denotes the set of all positive real values. To obtain the FC inputs, a reference value $y_r(k)$ has to be determined, and the system output $y_m(k)$ should be obtained from a sensor. Normalization of the $e(k)$ inputs and $ce(k)$ requires a scale transformation that maps the physical values of the system variables into a normalized domain as:

$$e_N(k) = k_e e(k) \quad \text{(Eq.3-3)}$$

$$ce_N(k) = k_d ce(k) \quad \text{(Eq.3-4)}$$

Where k_e and k_d are the input scaling factors, e_N , $ce_N \in [-L, L]$, and k_e , k_d , $L \in R^+$.

3.2.2 Fuzzification and membership functions

The membership functions used in the fuzzification play a crucial role in the final performance of a fuzzy control system, and the choice of the membership function has a strong influence on the control effect. There are several types of membership functions used in FC such as the triangular function, bell function, Gaussian function and trapezoidal

function. The triangular type membership function is most commonly used in FC applications because of its computational efficiency and simplicity.

3.2.3 Rule base

While the differential equations are the language of conventional control, IF-THEN rules about how to control the system are the language of fuzzy control, since an IF-THEN operator is the simplest and most widely used interpretation and, it provides computational efficiency. The control engineering knowledge method based on IF-THEN rules is the commonly used method to construct the rule base, since it needs more engineering skills and experience than plant information. Generally, the rules for the present problem are structured as:

$$r_i : \text{IF } e_N \text{ is } A_i \text{ and } ce_N \text{ is } B_i \text{ THEN } u_{fz} \text{ is } C_{ij}$$

Where r_i denotes the fuzzy rules; $i = 1, 2, ..., n$, n is the number of fuzzy rules; A_i and B_i denote input linguistic values from the fuzzy sets of the antecedent part of the controller for the e_N and ce_N ,

respectively, C_{ij} denotes the output linguistic values from the fuzzy set of the consequent part of the controller for the u_{fz} .

3.2.4 Inference engine

An inference engine is an interface that produces a new fuzzy set. The inference method for fuzzy control can be categorized into two groups: the direct inferencing and indirect inferencing. Control values are determined on the basis of the inferred state. It has been noted that indirect inference employs a small number of production rules, and this method is regarded as the Sugeno-Tagaki method. The direct method is commonly used in applications of FC because of the simplicity of using the min (T-norm)-max (T-conorm) operator. This operation is called Mamdani type inference, which is recommended for use because it produces stronger control action in some certain cases. To perform this method, first, the degree of match between the meaning of the crisp inputs and the fuzzy sets describing the meaning of the rule antecedent should be computed for each rule by using the min operator (T-norm approach).

$$\mu_{C_{ij}}(e_N, ce_N) = \min\{\mu_{A_i}(e_N), \mu_{B_i}(ce_N)\} \qquad \text{(Eq.3-5)}$$

Where $\mu_{C_{ij}}(e_N, ce_N) \in [0,1]$, and then, the max (T-conorm) operator is performed as:

$$\mu_{con}(u_{fz}) = \max\{\mu_{C_{ij}}(u_{fz})\} \in R \qquad \text{(Eq.3-6)}$$

Where μ_{con} denotes the meaning of the rule consequent part, $\mu_{C_{ij}}(u_{fz})$ denotes the value for the control output of the related rule.

3.2.5 Defuzzification

Defuzzification is the procedure that produces a real value from the result of the inference, which could be used as a fuzzy control input. The most widely used defuzzification method is the centre of gravity method, which extracts the value corresponding to the centre of gravity of the fuzzy set describing the involving signal, and it is computationally efficient. On the other hand, the Sugeno-Tagaki's defuzzification method is a kind of experimental design method. It is time consuming and not practical. According to the centre of gravity method, the crisp value of the fuzzy control output is given by

$$u_{fz} = \frac{\sum_{i=1}^{n_{rules}} \{membership(input_i) \times output_i\}}{\sum_{i=1}^{n_{rules}} \{membership(input_i)\}} \qquad \text{(Eq.3-7)}$$

Where i is the rule number.

3.2.6 Output normalization

The rules, along with the membership degree of the fuzzy inputs with the fuzzy inference engine, determine the fuzzy output u_{fz} in the defuzzification. This output should be denormalized by using a scaling factor to obtain the real control input $u_f(k)$.

$$u_f(k) = k_u u_{fz}(k) \qquad \text{(Eq.3-8)}$$

Where $u_{fz}(k) \in [-l_{fz}, l_{fz}]$, $u_f \in [-H, H]$ and k_u, l_{fz}, $H \in R^+$.

A diagram of the conventional FC is illustrated in Figure 3- 2. The appropriate selection of input and output scaling factors (k_e, k_d, k_u) is very important because they have significant effects on the stability and performance of the systems.

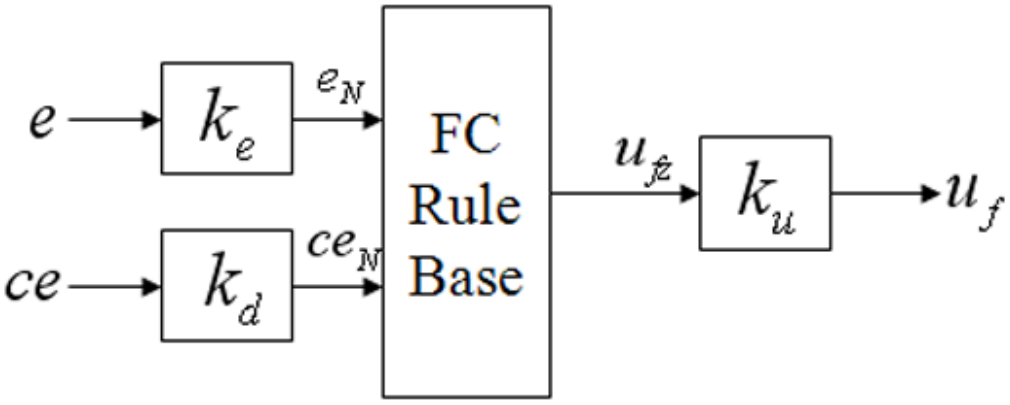
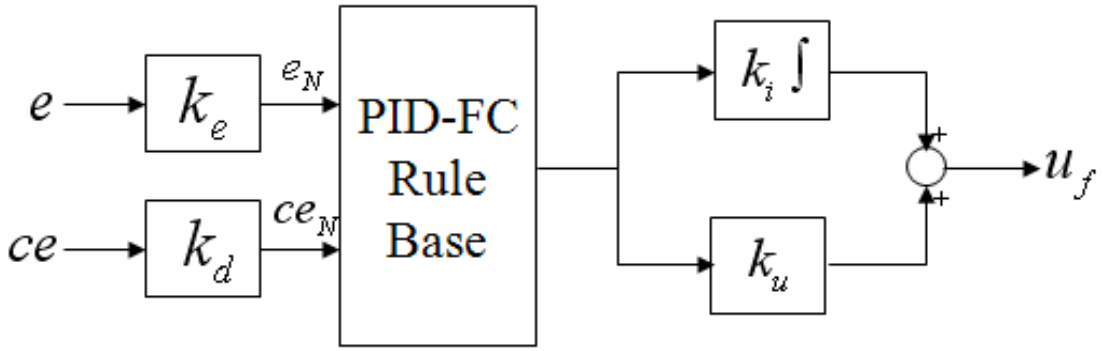


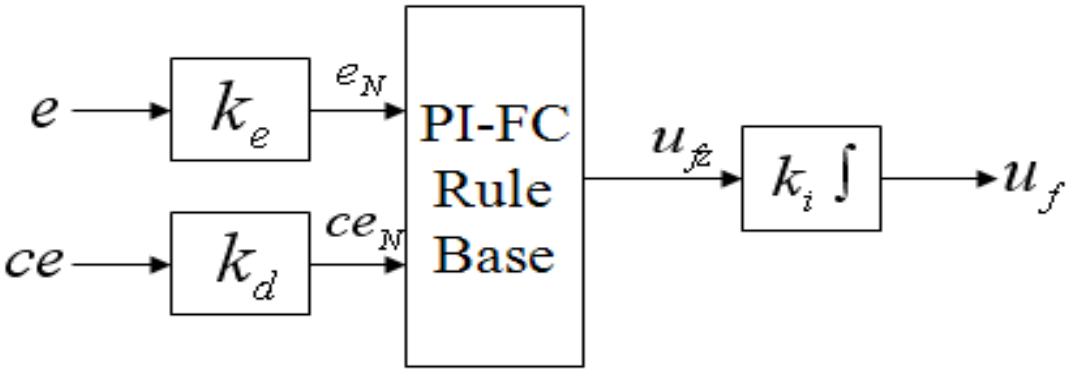
Figure 3- 2 Conventional FC diagram

3.2.7 PID fuzzy control

Conventional FC may result in steady state errors if the system does not have an inherent integrating property. To improve conventional FC, some algorithms have been proposed in the literature such as Fuzzy PID control. The PID type of FC is known to be more practical and generates incremental control output via integral action at the output. As the structural difference, the rule base of the PID-FC is different from that of the conventional FC in order to reduce overshoot and settling time. The structure of a PID type FC is shown in Figure 3- 3(a). The PID type of FC is capable of reducing steady state error, and it is known to give good performance in transient responses. Where k_i is the positive constant for the integral gain. PI-FLC has been developed to improve the transient response and Figure 3- 3(b) shows the PI-FC diagram.



(a) PID-FC



(b) PI-FC

Figure 3- 3 Fuzzy PID control diagram

3.3 Modeling of the RCMS dynamics

3.3.1 Dynamic model of the axial motion

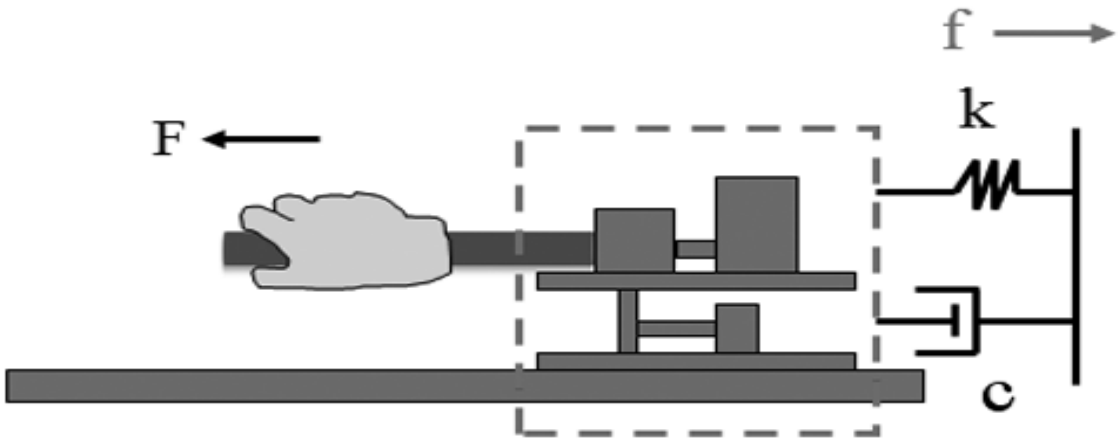


Figure 3- 4 Diagram of axial dynamic model

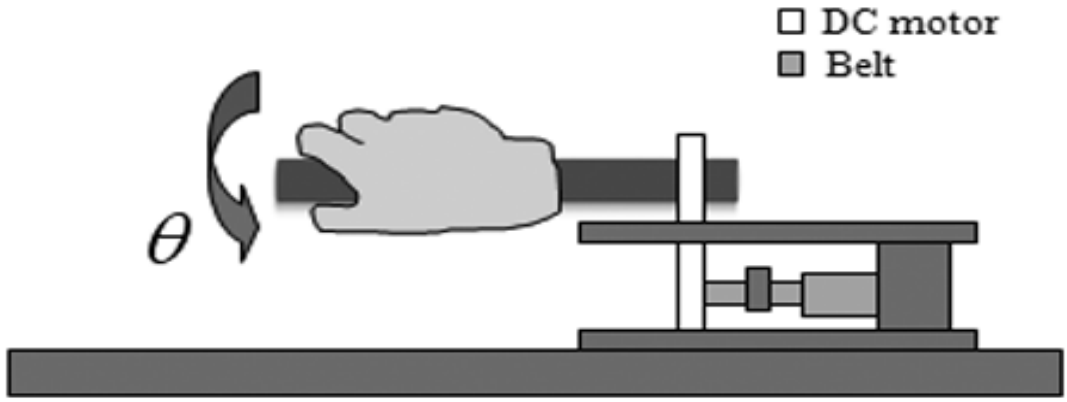


Figure 3- 5 Diagram of rotational dynamic model

Accurate model building is a crucial stage in practical control problems. An adequately developed system model is essential for reliability of the designed control. Consequently, the system modeling process is vital for control and identification problems. In modeling the axial motion, the aim is to find the governing equations that express the input pulling force and relate the stepping motor output displacement. A schematic diagram of axial motion built with regard to the aim expressed below is given in Figure 3- 4. The diagram shows the relationship between surgeon's input force and resistance. The equation that describes the physical relationship by using Newton's second law is as follows:

$$F(t) = m\ddot{x} + c\dot{x}(t) + kx(t) \qquad \text{(Eq.3-8)}$$

Where $F(t)$ is the pulling force, $x(t)$ is the moving displacement, $\dot{x}(t)$ is the velocity of hand movement. Define $x_1(t) = x(t)$, $x_2(t) = \dot{x}(t)$ then

$$\dot{x}(t) = AX(t) + Bu(t)$$
$$y(t) = CX(t) \qquad \text{(Eq.3-10)}$$

Where $X(t)=\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$, $A=\begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix}$, $B=\begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix}$,

$C=\begin{bmatrix} 1 & 0 \end{bmatrix}$, m is quality of the handle, c is the viscous damping coefficient, k is the coefficient of elasticity.

3.3.2 Dynamic model of the rotational motion

Like most rotational systems, the rotational motion in consideration can be modeled as a mass system that the mass connected with flexible shaft or spring. The model can be simplified further as a mass connected by an inertia free flexible shaft, where the mass represents the total load that the motor rotates. A schematic diagram of the simplified rotational model is illustrated in Figure 3- 5.

In modeling the dynamics of the simplified rotational movement, considering only linear dynamic or approximating the model as a linearized one is a common approach. However, the validity of the linear approximation and the sufficiency of the linearized dynamics in recovering the nonlinearities depend on the operating point and speed span of the rotational system.

For this case that the rotational motion can be accurately modelled without considering the major nonlinear effects by the speed dependent friction, dead time and time delay, a linear model becomes

$$m\ddot{\theta}(t)+c\dot{\theta}(t)=u(t) \qquad \text{(Eq.3-11)}$$

Where $u(t)$ is input torque, $\theta(t)$ is the rotational angle, $\dot{\theta}(t)$ is the velocity of rotational angle. Define $\theta_1(t)=\theta(t)$, $\theta_2(t)=\dot{\theta}(t)$ then

$$\begin{aligned}\dot{\theta}(t) &= A\Psi(t)+Bu(t)\\ y(t) &= C\Psi(t)\end{aligned} \qquad \text{(Eq.3-12)}$$

Where $\Psi(t)=\begin{bmatrix}\theta_1(t)\\ \theta_2(t)\end{bmatrix}$, $A=\begin{bmatrix}0 & 1\\ 0 & -\frac{c}{m}\end{bmatrix}$, $B=\begin{bmatrix}0\\ \frac{1}{m}\end{bmatrix}$, $C=\begin{bmatrix}1 & 0\end{bmatrix}$, m is quality of the handle, c is the viscous damping coefficient.

Parameters of the dynamic model are shown as following: $m=0.5kg, c=0.02N/(m/s), k=1N/m$.

3.3.3 Fuzzy PID controller design of the RCMS

The PID control is the most widely used controller in industry today, and it possesses the simple, robust, and stable properties. The analog version of a PID control is

$$u(t) = k_p e(t) + k_i \int_0^t e(t)dt + k_d \frac{de(t)}{dt} \qquad \text{(Eq.3-13)}$$

Where $e(t) = r(t) - y(t)$ and $u(t)$ is the PID controller output. The control signal is thus a sum of three terms: the P-term (which is proportional to the error), the I-term (which is proportional to the integral of the error), and the D-term (which is proportional to the derivative of the error). The controller parameters are proportional gain k_p, integral gain k_i, derivative gain k_d. The integral, proportional and derivative part can also be interpreted as control actions.

A PID controller used in the RCMS is easily designed with good performance during the remote control, however, when different condition of operation effects are concerned, a PID controller should be turn or redesign to maintain desirable responses. We design a fuzzy PID controller to provide a better control command to improve the performance of the system during remote control operations. The basic fuzzy PID configuration is shown in Figure 3- 6.

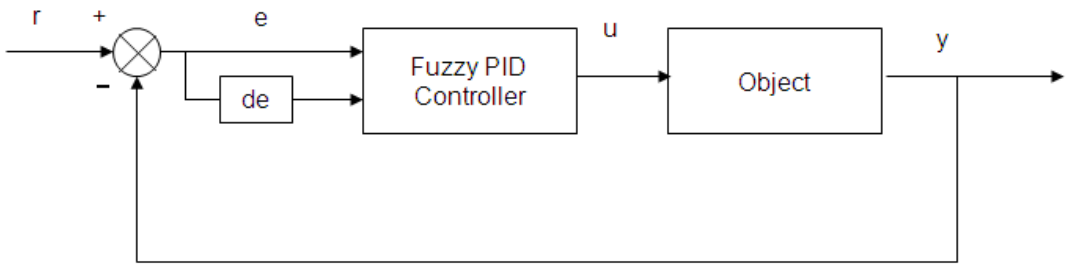


Figure 3- 6 Principle Diagram of fuzzy PID Control

For the design of fuzzy PID controller, first, the fuzzification stage should convert a crisp number into the fuzzy values within a universe of discourse U . The U is quantified and normalized to [-1, +1]. Then, we utilize the triangle-shaped membership function with seven term sets as shown in Figure 3- 7 and Figure 3- 8. They are NB (negative big), NM (negative medium), NS (negative small), ZR (zero), PS (positive small), PM (positive medium), PB (positive big). We construct the rule base according to control engineering knowledge, and the scaling factors are set to be 1.

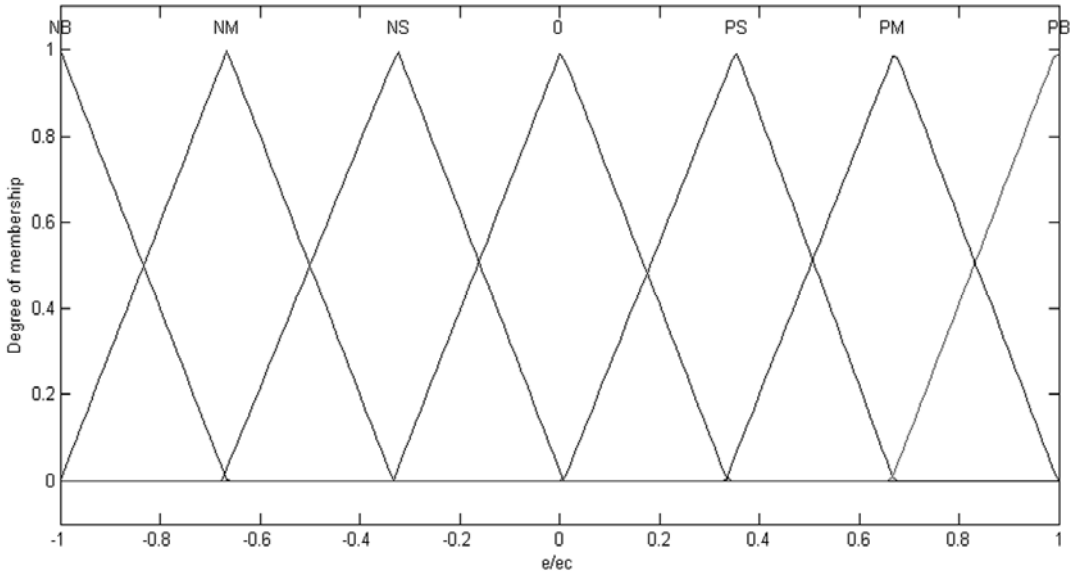
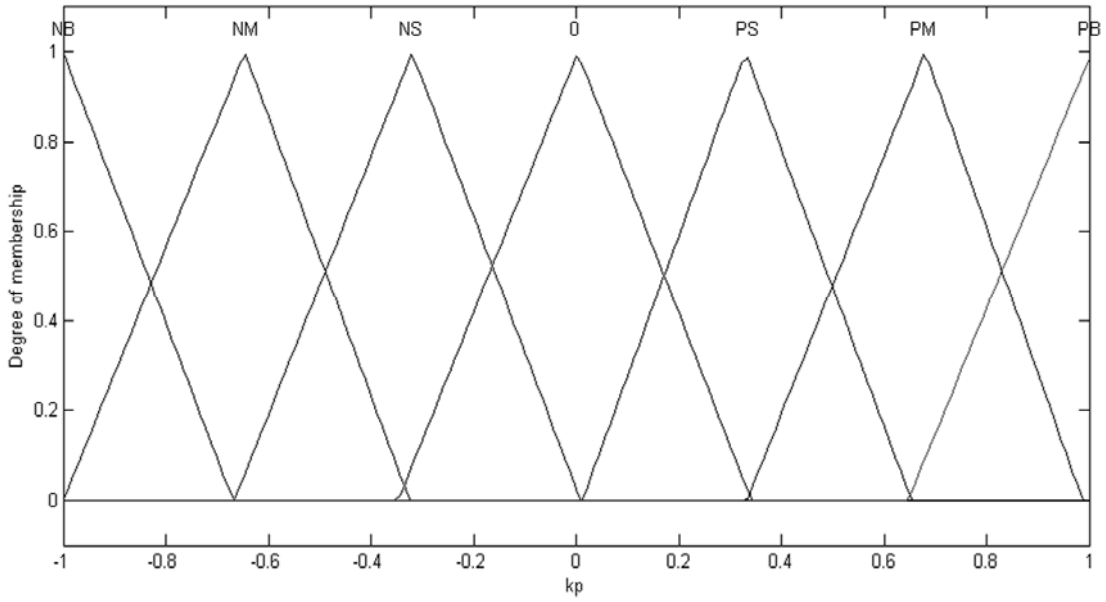
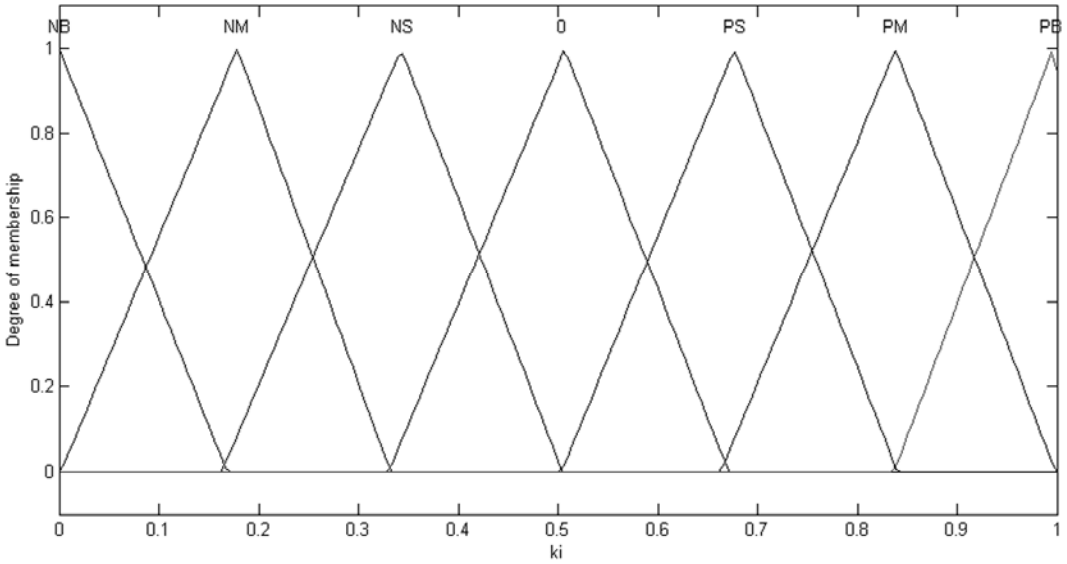


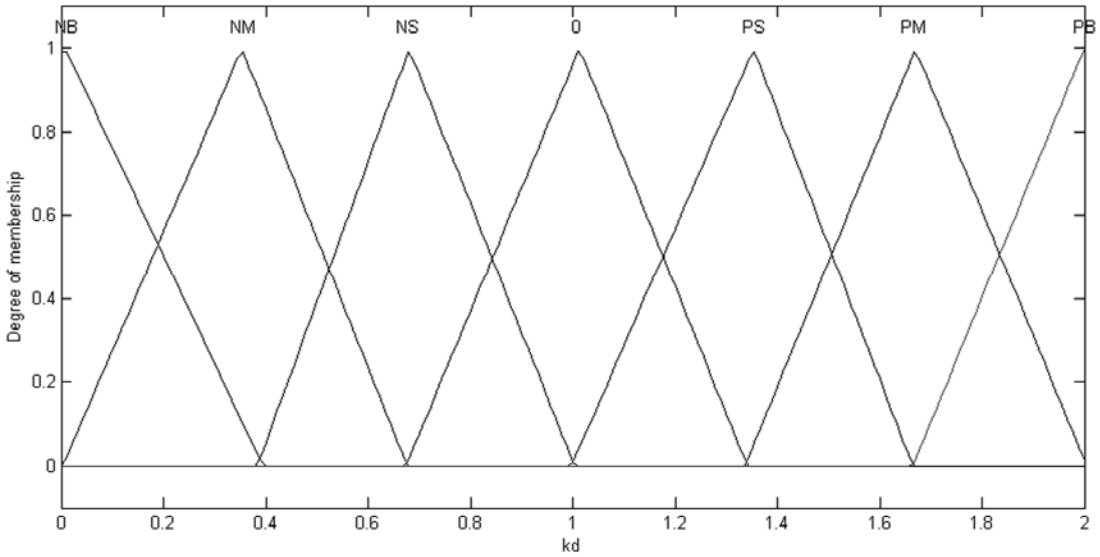
Figure 3- 7 Membership function for the $e(k)$ and $ce(k)$



(a) Membership function for the k_p



(b) Membership function for the k_i



(c) Membership function for the k_d

Figure 3- 8 Membership function

Each combination of error fuzzy set and error change fuzzy set need a control action. Forty-nine control rules are developed and presented. The Mamdani type controller is preferred because an extremely short time is required for its development and ease with which its functions can be understood. The defuzzification method based on the centre of gravity that is commonly used in applications of fuzzy control. The key issue in such control problem is to hold a variable to constant set point. As the design objective, the overshoot in displacement and angle are desired to be not bigger than 3% for nominal value. The set of fuzzy rules has been based on fast attaining of the desired one and avoiding its overshoots.

3.4 Experiments

3.4.1 Experimental setup

Experiments are conducted by using the RCMS that consists of master (surgeon console) and slave (catheter navigator) connected via internet. The stepping motor and dc motor which are employed to drive the catheter moved in axial and rotational direction. Consequently, examination of stepping motor and dc motor behaviours during the remote operations constitute a useful effort for analysis and control of many practical applications. The control diagram of the experimental setup is shown in Figure 3- 9 and the system is interfaced with the fuzzy PID controller as a two inputs and one output structure. The

displacement error ($e(k)$) and change of displacement error ($ce(k)$) are used as the inputs to the FC, since it has been shown that using the error and its derivative as inputs to FC leads to stable control in a large of control problems. The FC generates the primary command for the catheter navigator side of RCMS and adjusts axial and rotational status that uses to maintain good steady state and transient behaviour. The measured output by encode is transferred to the DSP (TI, TMS320F28335). The implementation of the FC is performed using the following procedures: measure the current output of displacement in stepping motor or angle in dc motor; calculate the error and error change; fuzzify the inputs using the rule base, otherwise the membership functions with IF-THEN operation; transform the fuzzified inputs into fuzzy inference using min-max operation; finally, defuzzify the information using the centre of gravity method to convert to fuzzy control. Then the defuzzified information consisted of k_p, k_i, k_d is transmitted to the PID controller and used as the input control signals to adjust the output signal $u(t)$.

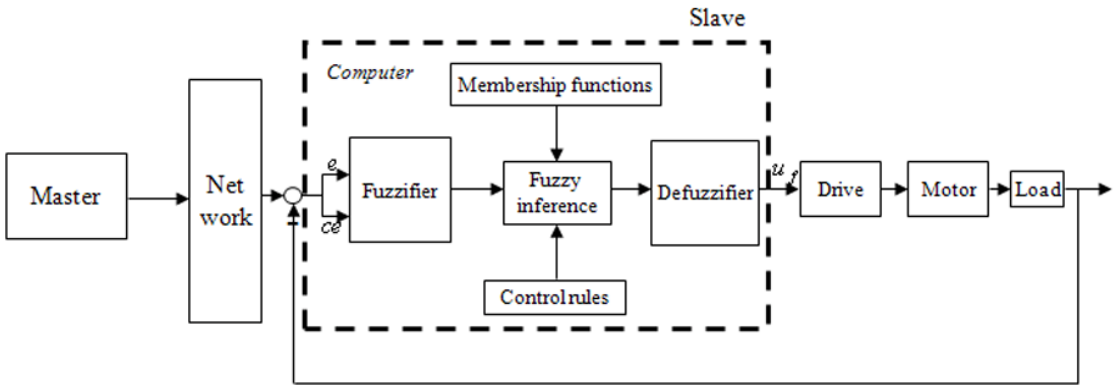


Figure 3- 9 Diagram of the experimental setup for the fuzzy PID controller

3.4.2 Experimental results

First, we perform a basic experiment during the remote control operations with PID controller. In this case, the characteristics of the performance can be obtained experimentally. Then we go ahead to use the designed fuzzy PID controller to improve the accuracy of remote operations. The experimental results obtained from axial displacement with PID are shown in Figure 3- 10 and the force input signal is described in Figure 3- 11. A smooth response is obtained without overshoot by using the fuzzy PID controller shown in Figure 3- 12 and the force input signal is demonstrated by Figure 3- 13. The values of conventional PID control parameters are determined, and coefficients are denoted to be: in term of axial movement as following:

$k_p = 10.8, k_i = 8.2, k_d = 1.5$; in term of rotational movement as following: $k_p = 7.3, k_i = 5, k_d = 4.1$.

The PID controller had many steady-state errors, and performance of tracking was much worse than the fuzzy PID controller. However, the conventional fuzzy PID performed well, but it has a few errors, these because of the time delay of remote operations. The input signal is the pulling force, we used the mathematic equation to calculate out values of axial displacement then used DSP control unit to translate the values as input signals for stepping motor.

The tracking performance of rotation with the fuzzy PID controller showing in Figure 3- 15 was much better than that with PID showing in Figure 3- 14, not only for the steady-state error and overshoot, but also for the shortest rise time and precision. It can provide much better tracking performance. Based on the experimental results, it can be concluded that the overshoot was significantly reduced to the desired level, and axial displacement and rotational angle were improved such that tracking speed and accuracy of behaviour tracking were obviously improved compared to these with PID controller.

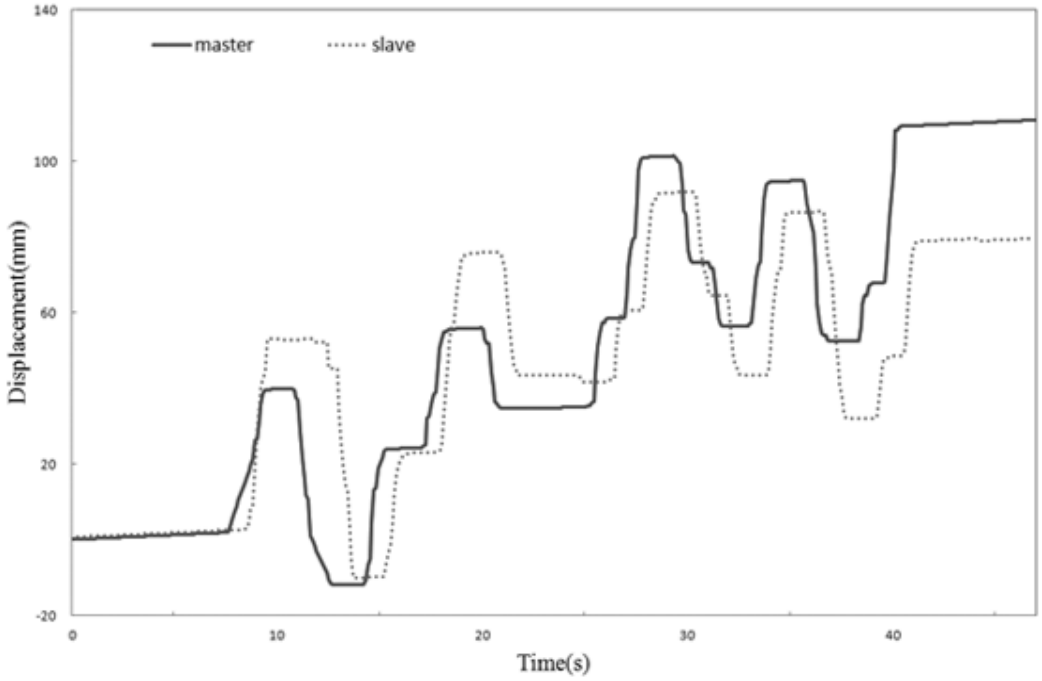


Figure 3- 10 Axial tracking curve with PID

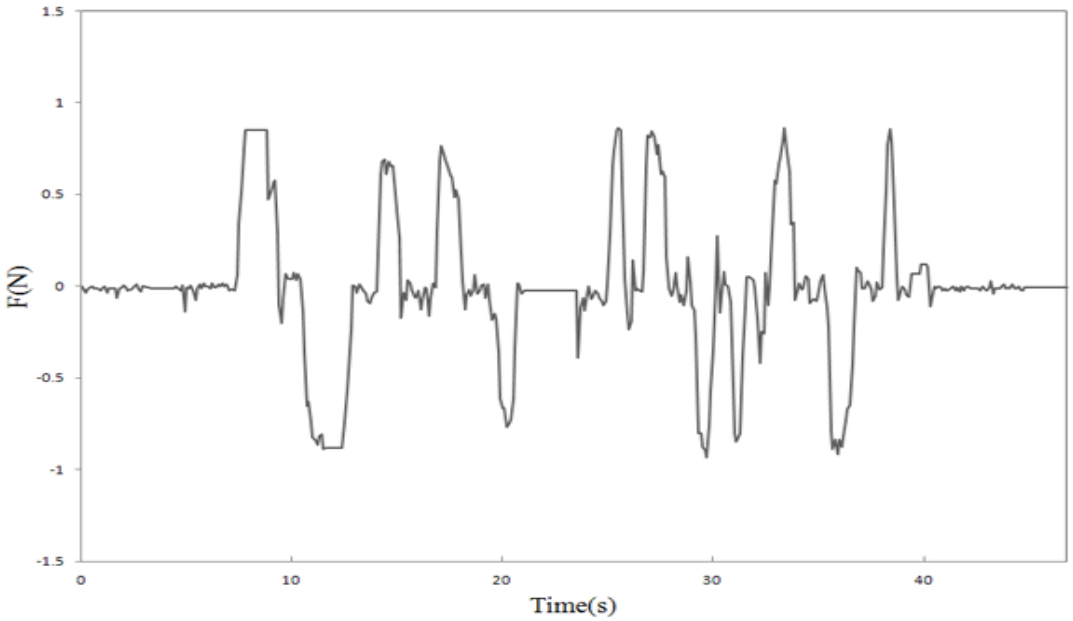


Figure 3- 11 Input force signal

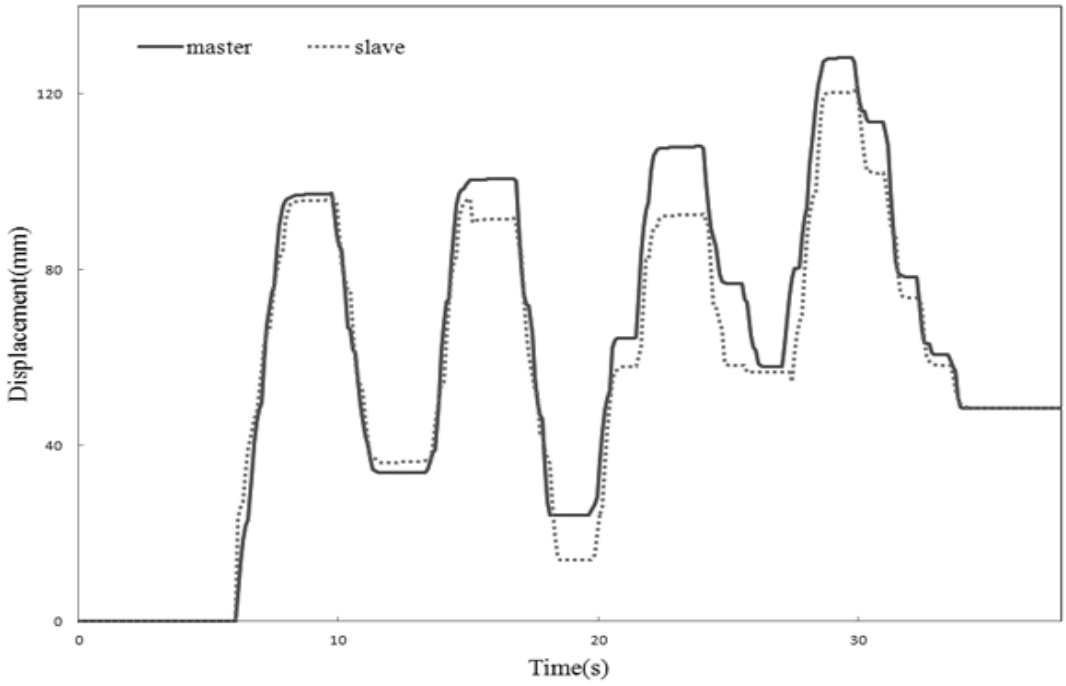


Figure 3- 12 Axial tracking curve with fuzzy PID

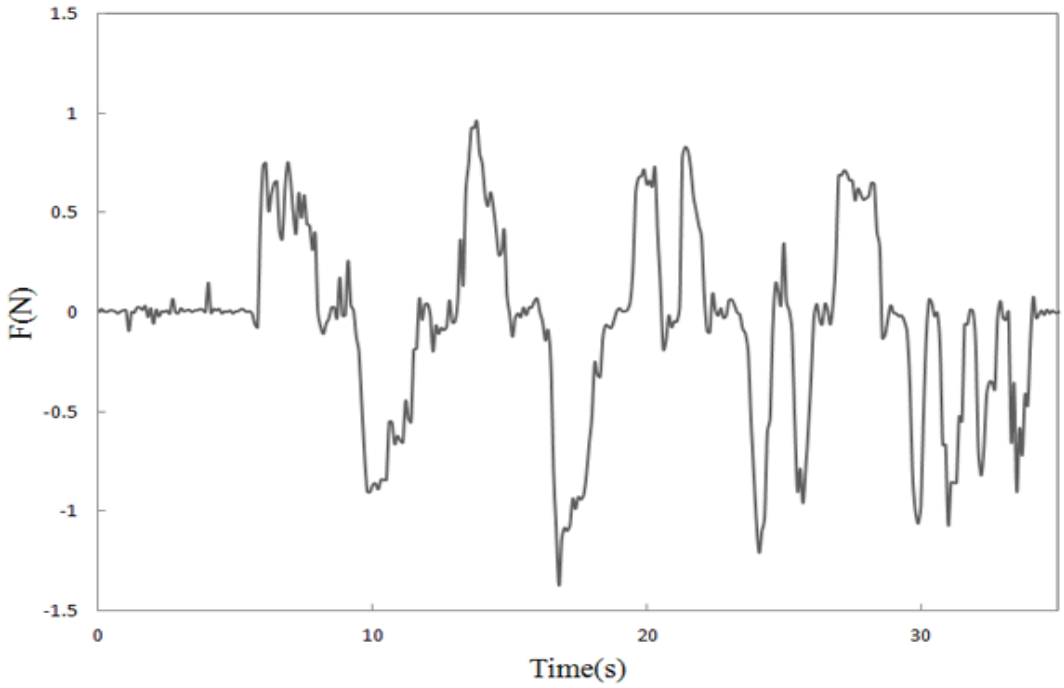


Figure 3- 13 Input force signal

However, for modelling the dc motor, we used the physical method to build the dynamic model of rotational motion, but we should consider that operate at varying conditions or require high precision operation raise the need for a nonlinear approach in modelling and identification. Most of mechanical systems used in industry are composed of masses moving under the action of position and velocity dependent forces. These forces exhibit nonlinear behaviour in certain regions of operation. For a system having two DOFs (rotation and insertion), the nonlinearities significantly influence the system operation when the velocity and rotation change the direction. So, we should focus on the nonlinear modelling of system dynamic and parameters identification in the future.

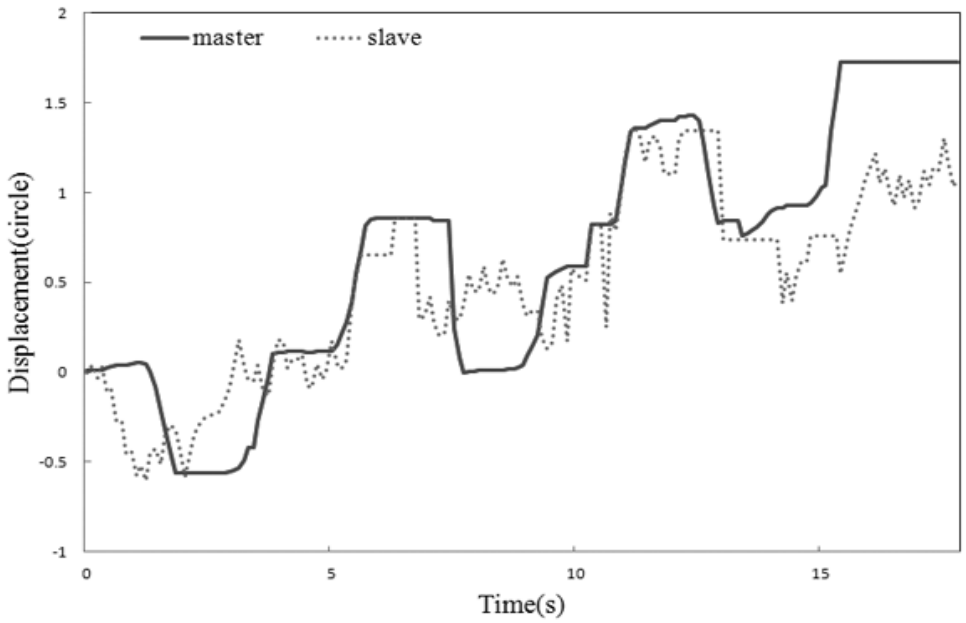


Figure 3- 14 Rotation tracking curve with PID

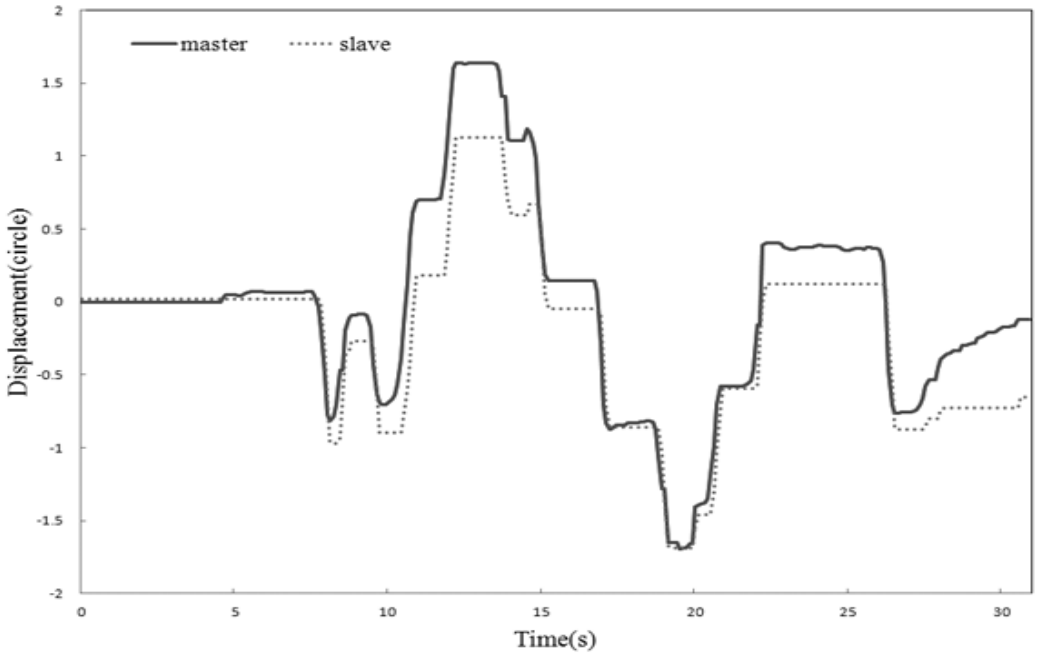


Figure 3- 15 Rotation tracking curve with fuzzy PID

3.5 Summary

In this chapter, we designed the fuzzy PID controller that is used in the slave side, the accuracy of axial and rotational movement during the remote operations have been improved and comparing with the conventional PID control method. The presented fuzzy PID controller is much better quality of response during insertion and rotation. The tracking error with fuzzy PID is below 3mm and 1.5° during the remote control. Although there also has a little error in the rotational and axial movement because of time delay, it also can satisfy the application of practical application in minimally invasive surgery. Experimental

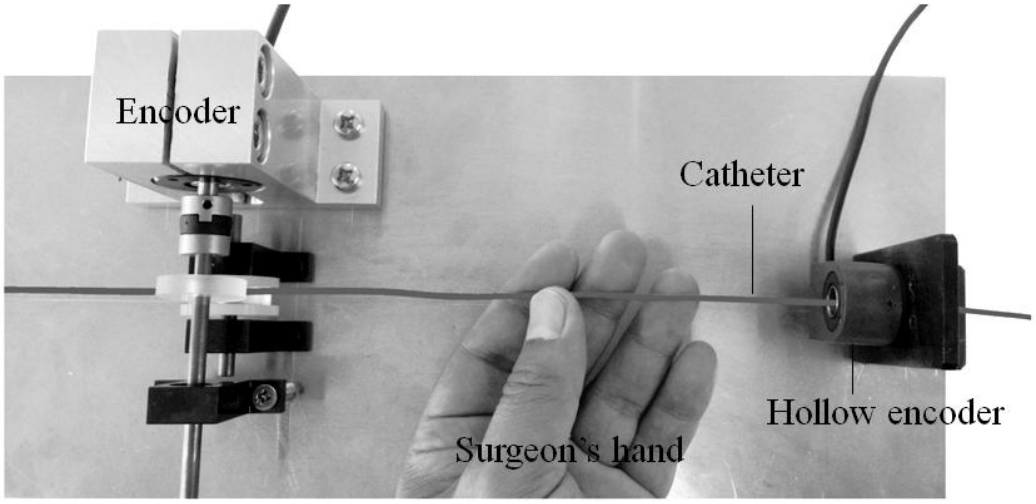
results confirm the fact that the fuzzy logic control method is suitable to be used and it is easy to understand and employ the system dynamic model freely. Finally, it should be said that the fuzzy PID control algorithm is the alternative approach to be used as the system control method.

Chapter 4
The novel robotic catheter manipulating system

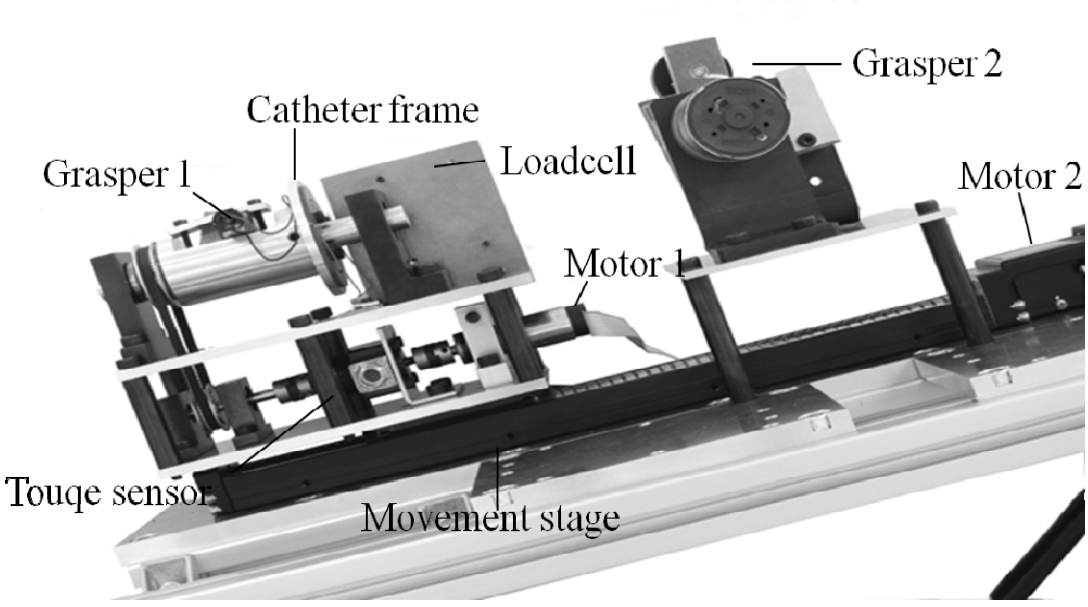
Unlike the conventional bedside technique, which requires surgeons to manipulate a catheter using their hands, employment of these remote manipulating systems removes the catheter from the surgeons' hands, thus removing his/her dexterous and intuitive skills from the procedure. Furthermore, the technological complexities of these systems may require long training times to ensure that the surgeons are skilled in their use. For example, a study conducted by Schiemann et al. demonstrated that equivalent navigation efficacy was achieved when comparing conventional navigation to remote navigation using the Niobe system in a glass phantom, after six months of surgeon training on the system. Therefore, it should be beneficial if a catheter manipulating system incorporated the dexterous skill set of an experienced surgeon during the procedures.

In this part a new prototype robotic catheter manipulating system has been designed and constructed based on the requirements for the endovascular surgery. Compared with robots mentioned above, our system features a slave manipulator that consists of one movement

stage and one rotation stage, allowing for steering and inserting the catheter simultaneously as Figure 4-1 (b) shown. Also, the slave has a new developed force feedback measurement mechanism to monitor the proximal force which has been generated during the inserting catheter and provide the force feedback to the surgeon. The robotic catheter manipulating system has a master controller called surgeon console in Figure 4-1 (a), using two motion-sensing devices via control unit DSP to communicate the position and rotation information with slave side. Also, we designed the haptic device to provide the feeling back to surgeons. The whole system was evaluated in aspect of dynamic and static performance of the axial and radial motions. The results of synchronization experiments had to evaluate the accuracy and precision of sensed and replicated motions. Finally, Tele-operation had been done by EVE simulator to provide the performance under the similar situations.



(a) The surgeon console



(b) The catheter manipulator

Figure 4- 1 The robotic catheter manipulating system

4.1 The system design

The catheter manipulating system was designed with the structure of master and slave. The surgeon console of the system is the master side and the catheter manipulator is the slave side. Moving mode of the catheter manipulator is designed as well as the surgeon console. The movable parts of surgeon console and catheter manipulator keep the same displacement, speed and rotational angles, therefore, the surgeon would operate the system smoothly and easily. Each of surgeon console and catheter manipulator side employs a DSP (TI, TMS320F28335) as their control unit. An internet based communication was built between the surgeon console and the catheter manipulator, the sketch map of the communication is shown in Figure 4-2. The surgeon console side sends axial displacement and rotational angle of the catheter to the catheter manipulator. At the same time, the catheter manipulator sends force information back to the surgeon console side. Serial communication is adopted between PC (HP Z400, Intel Xeon CUP 2.67GHz speed with 3GB RAM) and control unit of the mechanism. The baud rate of the serial is set to 19200.

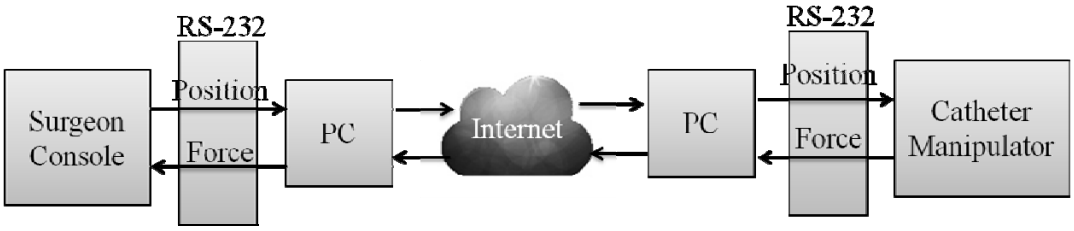


Figure 4- 2 The communication sketch map

To ensure remote operation using this system is appreciable with conventional bedside operation, the following criteria were used in the design process.

1) The system should be compatible with generic 6-7Fr (diameter: 2-2.3mm) catheters, sizes common in interventional surgery.

2) Axial motion and radial motion should not be hindered by either the surgeon console and catheter manipulator.

3) Accuracy of axial movement: 1mm for 1.5m catheter

4) Accuracy of radial movement: below 3^{o}

5) Synchronization performance between surgeon console and catheter manipulator

4.2 The catheter manipulator

Figure 4-3 shows the catheter manipulator. This mechanism is placed in the patient side. The catheter is inserted by using this mechanism. It could provide two DOFs, one is axial movement alone the frame, the other is radial one. Two graspers are placed on it. The surgeon can drive the catheter to move and rotate along both axial and radial direction only when the catheter is clamped by grasper 1 coupled with catheter frame. Because the stroke is limited, so we should withdraw the catheter by release the grasper 1 and keep the catheter at same place using grasper 2. Then, the catheter could be retreated for next insertion.

Inserting motion of the catheter is described in Figure 4-4. We could know that catheter insertion has been operated in the movement range of 2~3cm every operating procedures shown in Figure 4-4 (a). For the clinical catheter, it is flexible. However, it has enough rigid to deliver the F/T in this short distance similar to the procedures of clinical surgery. Usually, the neurosurgeons operate the catheter within 2~3cm movement range every operating procedures. The catheter manipulator was designed to imitate the operating procedures.

To realize axial movement, all catheter driven parts should be placed and fixed onto a movement stage (the plate under motor 1). The movement stage is driven by a screw which is actuated by a stepping motor (motor 2). On the other hand, one dc motor (motor 1) is employed to realize the radial movement of the catheter. It is coupled to the catheter frame by two pulleys connected with a belt together. The catheter is driven to rotate by motor 1 when the catheter is fixed on the frame by grasper1.

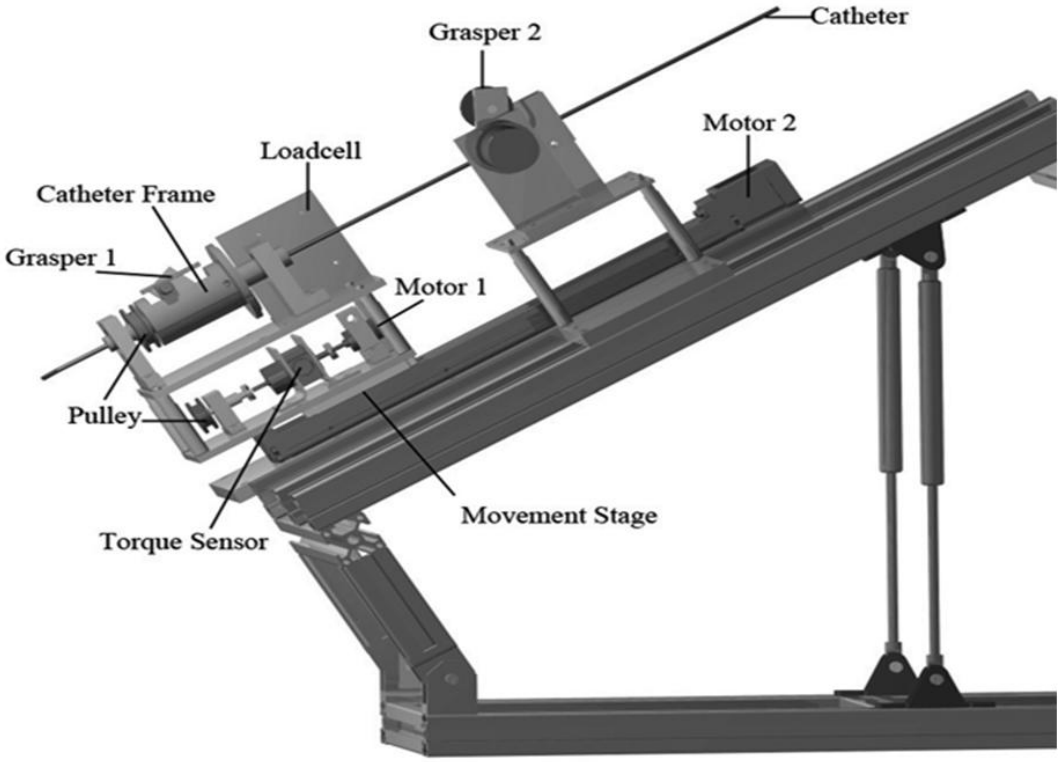


Figure 4- 3 The catheter manipulator

Torque sensor is applied in this system to measure the torque information during the operation. The torque data will be sent to the surgeon console. The torque sensor is linked to motor 1 and one shaft of the pulley below. The resistance torque on the catheter can be transferred to the torque sensor through coupled pulleys then measured by the torque sensor.

Resistance force acted on the catheter can be measured and it will be sent back to the surgeon console, then generate a haptic feedback to the surgeon. To measure the resistance force, a mechanism is designed as shown in Figure 4-5 (a) in details. A loadcell which is fixed on the

movement stage is employed to measure the resistance force. A clamp plate fixed on the loadcell is linked to the catheter frame which is supported by two bearings. The resistance force acted on the catheter during the insertion can be detected by the loadcell. As the movement range of catheter operation is about 2~3cm every operating procedures and the catheter is enough rigid in this range, therefore, it can guarantee that the generated resistance force could be transferred back and make the catheter frame have be a bit of movement. Then, this force could be measured by loadcell shown in Figure 4-5 (b). The clamp plate doesn't affect the rotation motion of the catheter 4rame. Although the designed measurement mechanism of resistance force could work, it also has some problems such as the sensitivity of the mechanism and so on need to be resolved.

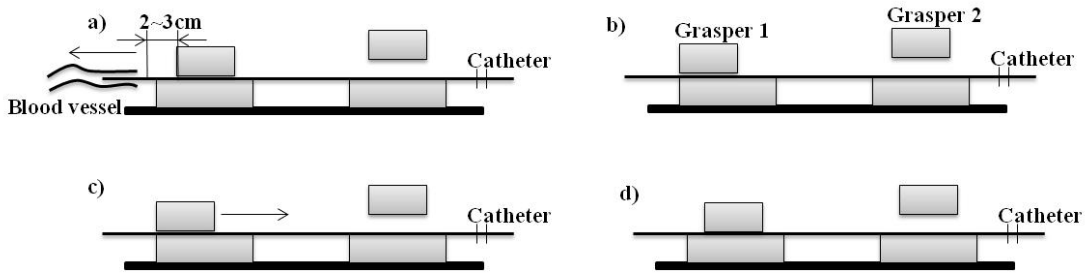


Figure 4- 4 The insertion motion

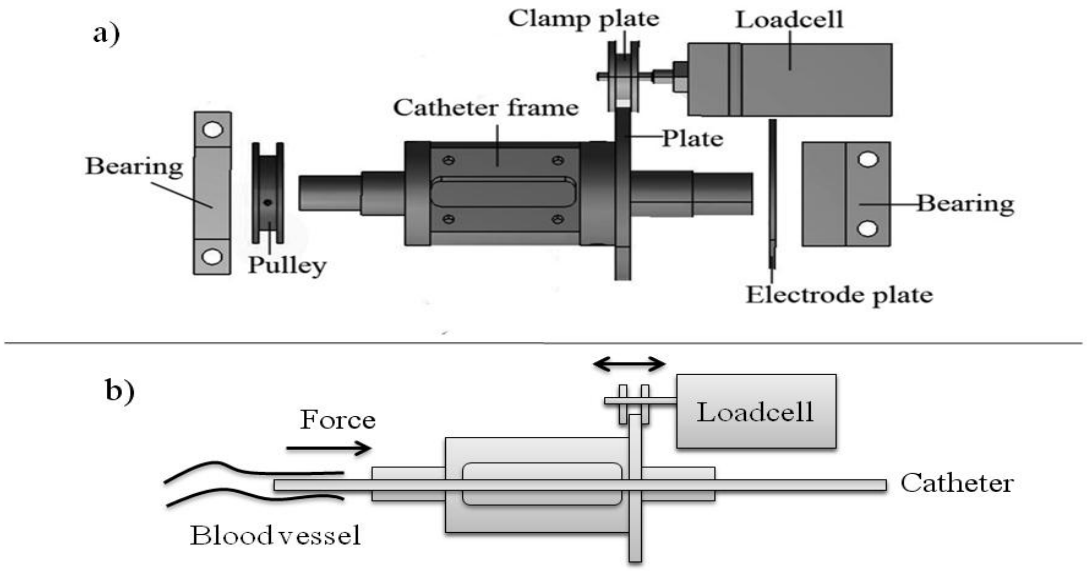


Figure 4- 5 The force measurement mechanism
a) Mechanical structure b) Force measurement method

4.3 The surgeon console

The prototype of surgeon console shown in Figure 4-6 is an electromechanical device that measures the axial and radial motion of the input and output catheter using two mechanically independent passive sensors. Each sensor contains a 2000 lines encoder, mechanically coupled to the catheter. Axial position of the catheter is measured using a mechanical structure that converts the axial motion of the catheter to a rotational motion of the shaft of an optical encoder (Rotary encoder, MES2000P, Japan) using two rollers which mechanically couple to the catheter described in Figure 4-7 (a). One of the rollers (the main roller1) is directly coupled to the encoder, while

the second idler roller2 passively ensures continuous contact between the primary roller and catheter. The position of the second roller is adjustable to allow variable contact friction between the catheter and the primary roller. The rollers were manufactured from MISUMI Corporation to ensure dimensional stability and the material is rubber. The axial position of the input catheter's shaft is determined as the product of roller circumference (approximate 75 mm). In the current implementation, detection of a single-counter increment yields a motion sensitivity of about 0.04 mm/count in the axial direction.

To measure radial motion, the catheter is used as a shaft to be coupled with the radial encoder shown in Figure 4-8 (b). A kind of hollow encoder (Rotary encoder, UN2000C4, Japan) is constructed to house catheter and coupling, which housed one screw. We processed the circular catheter into a irregular shapes, then the screw (diameter 2mm) can grip the catheter in the radial direction and holds it at the center of the encoder disk while allowing it to move freely in the axial direction; also, the screw which can be adjusted freely are loaded to ensure contact between the coupling and catheter. The outer edge of the coupling matching with hollow encoder enables the catheter to freely rotate the optical disk through the optical sensor. The radial position of the catheter can be measured directly by the encoder. In the current manipulation, detection of a single-counter increment yields a motion sensitivity of 0.18o/count in the radial direction.

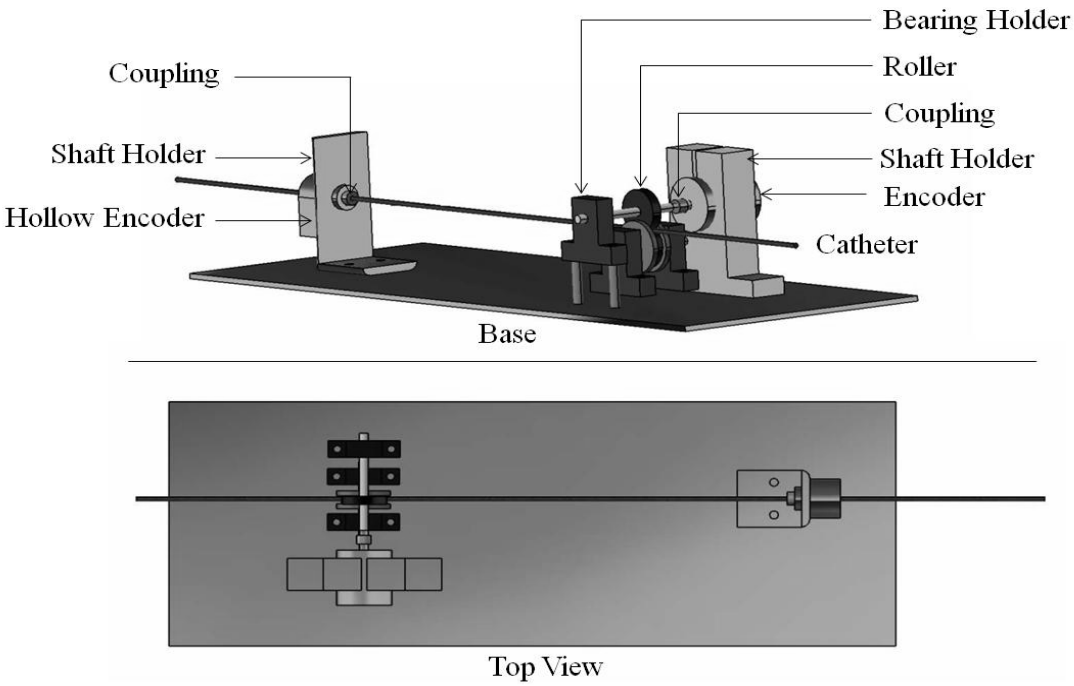


Figure 4- 6 The structure of the surgeon console

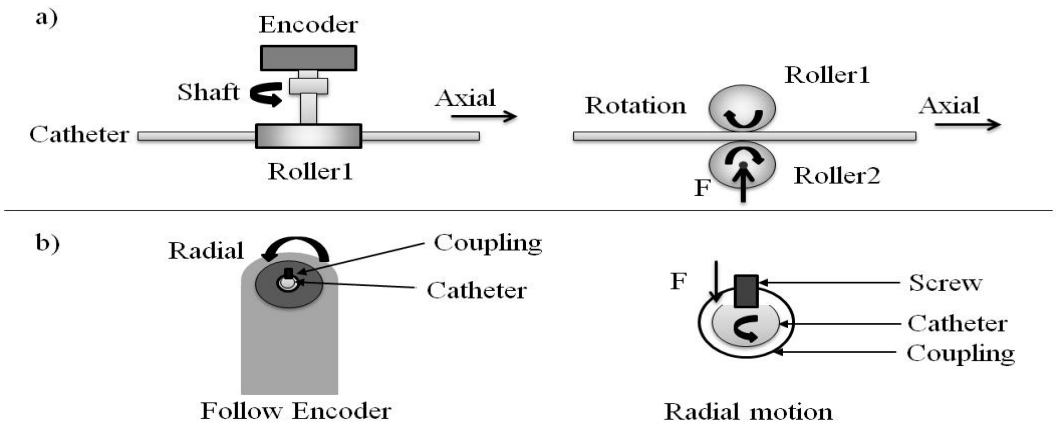


Figure 4- 7 The schematic diagram of surgeon console

a) The displacement measurement of axial direction b) The rotational angle measurement of radial direction

Haptic devices in tele-operation systems have an important place because surgeons must have the same felling as a real surgery. We designed a new haptic device which can change the friction on the catheter and provide the feeling back to the surgeon as Figure 4-8 shown. A plastic pipe is a bit bigger than the catheter can create friction on it as Figure 4-9 described. If the catheter manipulator side detects resistance force, the haptic device will generate it as near as the real one. We also used 4 springs for the system stability. In fact, when the catheter is totally stopped by the feedback pressure, we don't have any information about the force that the surgeon puts on the catheter. So, we will consider it and discuss the situation about the friction force measuring and force feedback providing in the future.

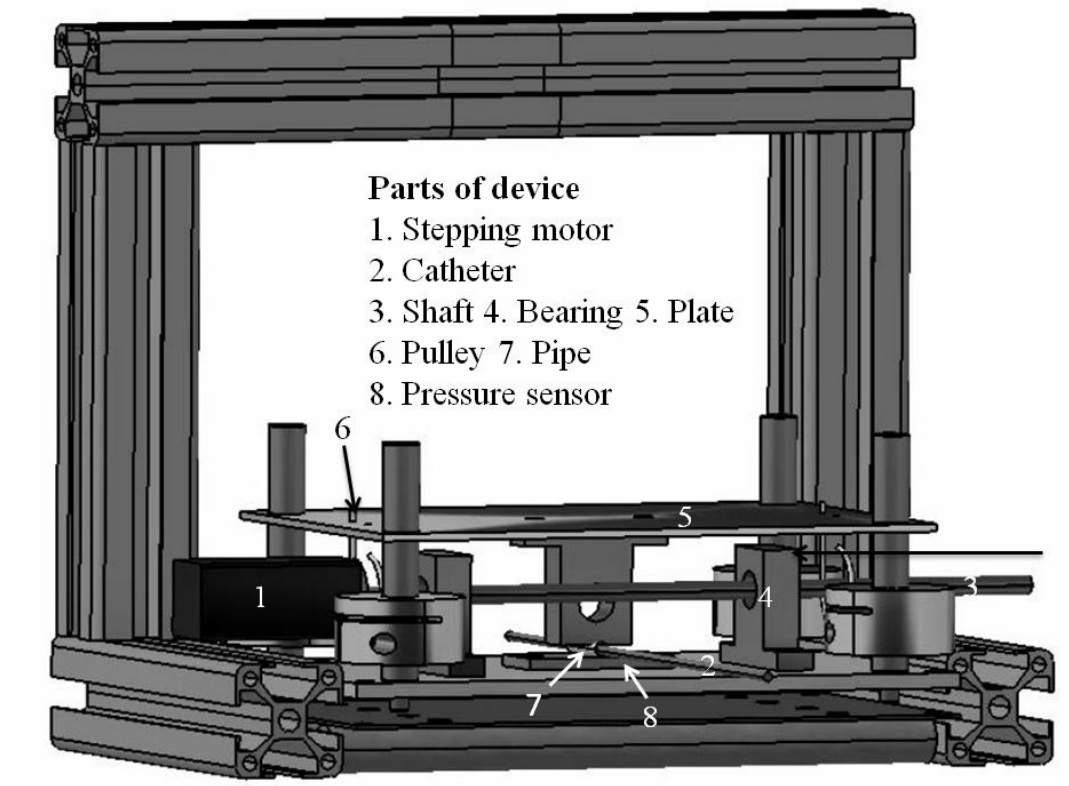
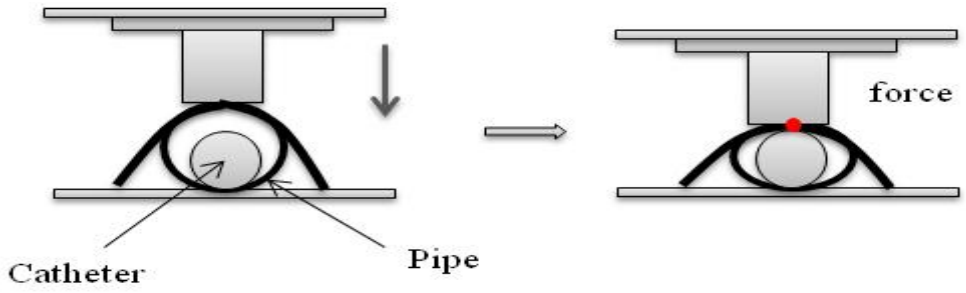


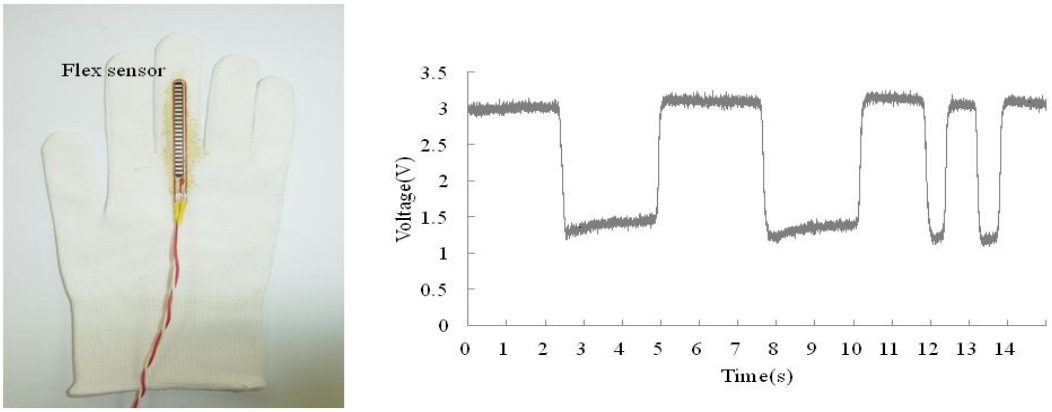
Figure 4- 8 Haptic device in the surgeon console



a) no haptic force b) force generated

Figure 4- 9 Schematic diagram of haptic device

As we need to withdraw the catheter every operating procedures by controlling the grasper. The flex sensor was used by bending it to change its resistance then obtain the voltage variation data shown in Figure 4-10. This voltage will be as the input signal transmitted to the DSP and control the grasper in the catheter manipulator side.



a) flex sensor b) bending test

Figure 4- 10 Schematic diagram of the flex sensor

4.4 Control of the system

Control of the surgeon console and catheter manipulator is achieved through two DSPs (TI, TMS320F28335) via RS-232 serial communication. Control software was implemented using C language, to enable synchronized motion control in the axial and radial direction, device control was multithreaded. The axial and radial motions

measured by two encoders in the surgeon console are represented for PSurgeon [XM θ_M] and solved to determine the corresponding position PPatient [XS θ_S] of the catheter manipulator in motor space. The position of each component of surgeon console is sampled at 1ms intervals; the corresponding velocity and acceleration values are determined and commands are then transmitted to the catheter manipulator controllers at 10 ms intervals. In the catheter manipulator side, the PD control algorithm has been added to improve the tracking accuracy.

4.5 Summary

The novel catheter manipulating system described in this chapter uses a novel method to control the catheter for remote operation. This operation method promises to enable surgeon to use their highly dexterous skills to remotely manipulate the catheter, potentially reducing radiation exposure and physical stress during long procedures. The current implementation of the system was designed for use with 6–7Fr catheters, commonly used in minimally invasive surgery procedures, but is easily adaptable for catheters of different sizes.

The presented catheter manipulating system has many potential advantages over commercially available systems. Unlike magnetic catheter navigation, where large permanent magnets are used to locate

the catheter, thereby requiring specialized catheters, the presented system can be easily integrated into existing fluoroscopic suites. The current system also uses generic catheters, with performance characteristics known to the surgeon, during remote operation. Most other commercially available remote navigation systems utilize joystick input devices to operate the remote catheter without providing tactile feedback to the surgeon. Because of the flexible nature of catheters, external forces applied to the catheter during catheter guidance occur when the tip of the catheter pushes directly into tissue or when twisting the catheter pushes its body against the vascular wall. In both situations, the external forces applied to the catheter are not fully transferred to the surgeon but instead result in catheter deformation. The operator uses these visual cues during catheter guidance, and we expect that the ability to exploit prior dexterous skills during remote catheter operation, as provided by our system, may provide additional benefit over remote operation systems employing joysticks or other non-intuitive master devices.

Chapter 5

The performance evaluation

5.1 Evaluation method

5.1.1 Evaluation of the surgeon console

To evaluate the performance of the surgeon console, two experiments were carried out. One is to evaluate the dynamic and static performance of axial motion and the other one for radial motion. Prior to evaluating the performance of the surgeon console, a series of experiments were performed to decrease the mechanical backlash. In the axial direction, mechanical backlash was measured by moving the catheter from 0 to 150 mm then back to 0 mm, ten times in succession. The difference between the start position and final position recorded by encoder, as reported by the surgeon console, was divided by the total number of iterations to determine error per direction change. The backlash error was then corrected. This process was repeated iteratively until the final error was below 1mm. In the radial direction, the methodology to calculate the mechanical backlash was similar; rotating the catheter from 0^o-180^o and back to 0^o, ten times, and then adjusting the backlash constant until it was below 3^o.

For the first experiment, we needed to evaluate dynamic and static performance of the axial motion. In terms of dynamic performance, we actuated movement stage as sinusoidal moving and change the moving frequency of movement stage from 0.1Hz to 100 Hz. The catheter was coupled to the movement stage as Figure 5- 1 shown.

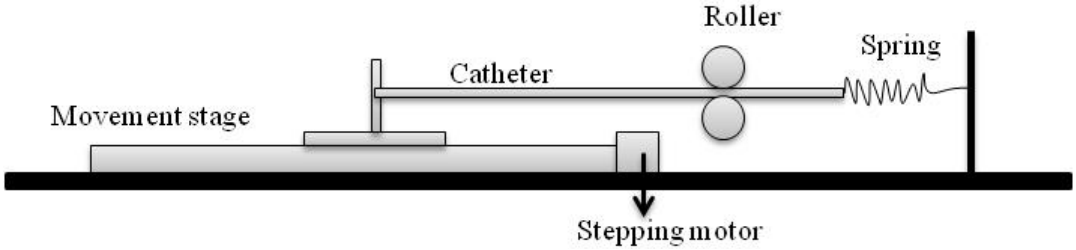


Figure 5- 1 Experimental setup of performance evaluation of axial motion

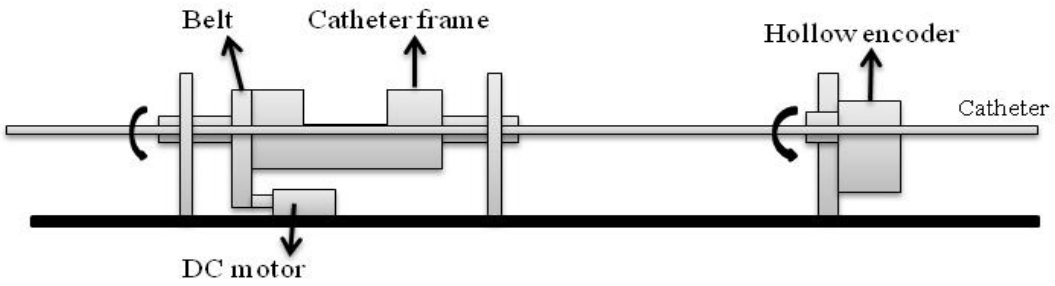


Figure 5- 2 Experimental setup of performance evaluation of radial motion

The catheter (6Fr) had been advanced and then retracted for 5 times at every frequency, then the axial displacement of movement stage that measured by an encoder inside the stepping motor was obtained. A DSP was applied to get the displacement data and sent it to the computer (HP, z400). These values were compared with the corresponding position reported by the catheter. At the same time, the measurement data by encoder of catheter was sent to the same computer by serial port, the sampling frequency of the controller was set to 1000Hz. Baud rate of the serial port was set to 19200. After comparing these two groups of data, the dynamic performance of axial motion could be evaluated. Next, we set the movement stage moving as the same velocity and direction to evaluate the static performance of axial direction. The catheter (6Fr) had been advanced for 5 times at certain velocity. We changed the velocity from 0.7mm/s to 300mm/s and measured the moving displacement of catheter and movement stage. Comparing with the two groups of date, we calculated the average error of displacement between catheter and movement stage. The static performance could be evaluated.

In the second experiment, only static performance of radial motion was evaluated using the 6F catheter. As well as the first experiment, we actuated the catheter frame rotating as the same velocity and direction rotating and change the rotating velocity of catheter frame from 180o/s to 540o/s. The catheter was coupled to the movement stage as Figure 5-

2 shown. Comparing with the rotational degree between catheter and catheter frame, the static performance of radial motion could be evaluated.

5.1.2 Evaluation of the catheter manipulator

For the first experiment, we advanced and then retracted the catheter(6Fr) for about 2 minutes (approximate) in a 120mm range, then the axial displacement were obtained that measured by a laser sensor (KEYENCE Inc., LK-500, high precision mode, 10μm/mV). An A/D convert board (Interface Inc., PCI3329) was applied to get the displacement data to the computer (HP, z400), the sampling frequency was 100Hz. These values were compared with the corresponding position reported by the catheter manipulator and accuracy was calculated as the average difference between laser sensor measurement and encoder measurement. At the same time, the measurement data of the catheter manipulator was sent to the same computer by serial port, the sampling frequency of the controller was set to 100Hz. Baud rate of the serial port was set to 19200. After comparing these two groups of data, the axial measurement precision could be evaluated.

In the second experiment, the accuracy and precision of radial position measurements were evaluated using the 6F catheter. As well as the first experiment, we rotated the catheter in clockwise and anticlockwise for 2 minutes. Accuracy was evaluated by obtaining measurements at -1800 and 1800, then calculating the mean error in the

measurement. Radial measurement precision was evaluated by rotating the rod by 3600 and then calculating the standard deviation. Then measure the rotation angle by a 3-axis inertial sensor (Xsens Inc. MTx, resolution 0.1 deg) fixed on the catheter. The sampling frequency of the inertial sensor is set to 100Hz as the same as the controller. Sampling data of the controller was send to the computer by serial port.

5.1.3 Evaluation of the synchronization performance

The synchronization experiments were carried out and we did not consider the lag. We just evaluated the tracking performance of the axial and radial displacement. Firstly, we advanced and retreated the catheter many times. Then, we got the axial displacement by encoders both in surgeon console side and catheter manipulator side. This procedure was carried out for four times. Similarly, the synchronization of rotation could be obtained.

5.2 Experimental results

5.2.1 Evaluation of surgeon console and catheter manipulator

The dynamic and static performance of surgeon console was evaluated described in Figure 5- 4, Figure 5- 6 and Figure 5- 8, and accuracy and precision of the catheter manipulator were listed in the Table 5- 1. All of the evaluation had been measured after backlash correction and calibration. Figure 5- 4 and Figure 5- 6 shows the dynamic and static performance of axial motion. In the Figure 5- 4,

position error described the difference of catheter moving distance (actual output) and movement stage moving distance (ideal output). Vertical axis (position error rate) meant the ratio of position error and ideal position. Horizontal axis meant the moving frequency of movement stage and catheter from 0.1Hz to 100Hz. The surgeon console worked well between 1Hz and 10Hz. From 0.1Hz to 1Hz, the position error was much bigger, that because the backlash was generated. Although we corrected the backlash before, it was still generated during the test. In the Figure 5- 6, position error described the difference of moving distance of catheter and movement stage. Horizontal axis meant the moving velocity of catheter and movement stage from 0.7mm/s to 300mm/s. The surgeon console worked well in static characterization. The Figure 5- 3, Figure 5- 5 and Figure 5- 7 described the dynamic and static performance of surgeon console side in detail.

Table 5- 3 Evaluation of the precision and accuracy

Catheter Manipulator	Precision	Accuracy
Axial (mm)	0.23	0.04
Radial (deg)	2.2	3.0

5.2.2 Evaluation of the synchronization performance

In this experiment, we implemented synchronization experiment between the surgeon console and catheter manipulator. The axial and radial displacement both of surgeon console and catheter manipulator was compared. Figure 5- 9 and Figure 5- 10 showed the axial tracking experimental results. Similarly, Figure 5- 11 and Figure 5- 12 showed the radial tracking experimental results.

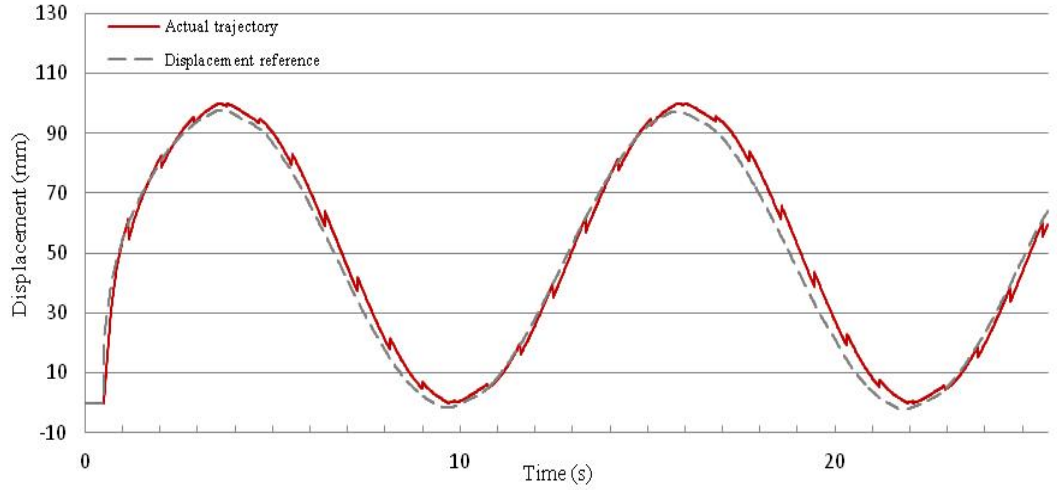


Figure 5- 3 Dynamic performance of axial motion at 0.1Hz

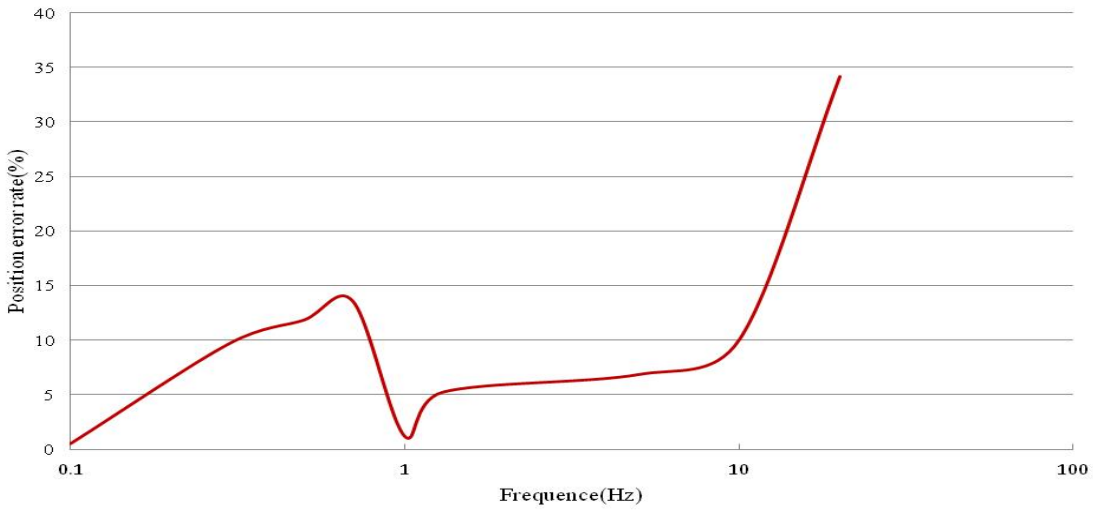


Figure 5- 4 Dynamic characterization of axial motion

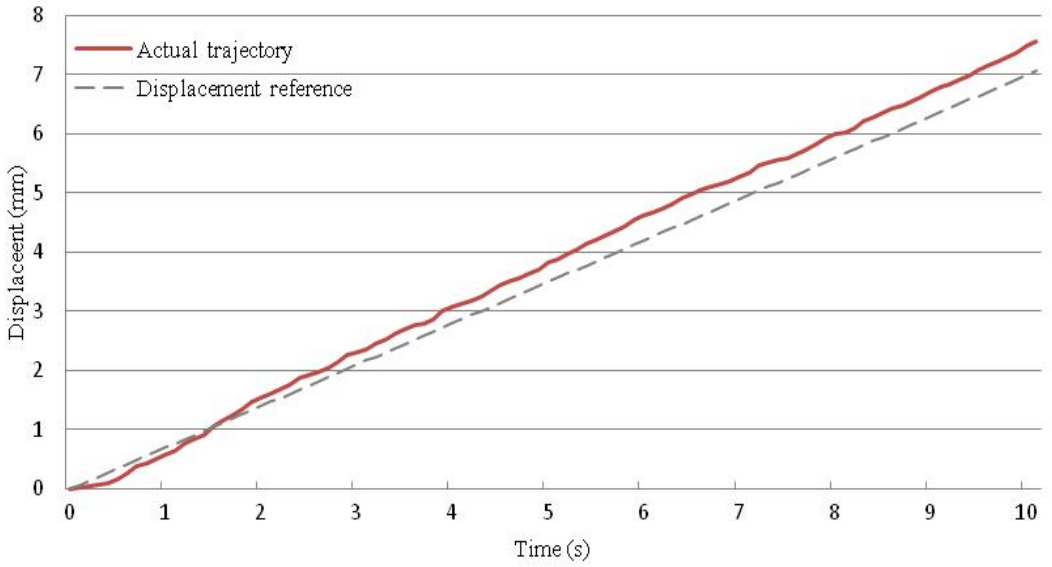


Figure 5- 5 Static performance of axial motion at 0.7mm/s

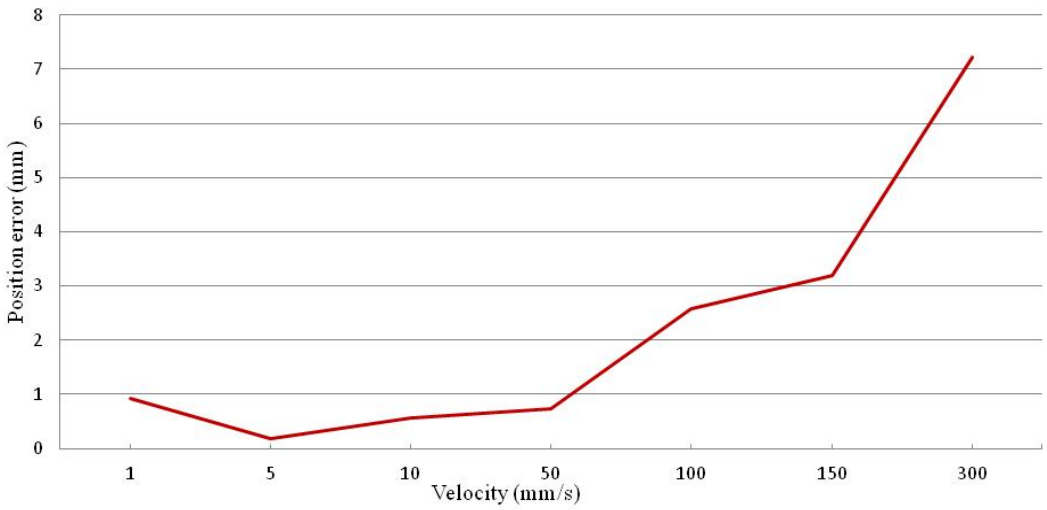


Figure 5- 6 Static characterization of axial motion

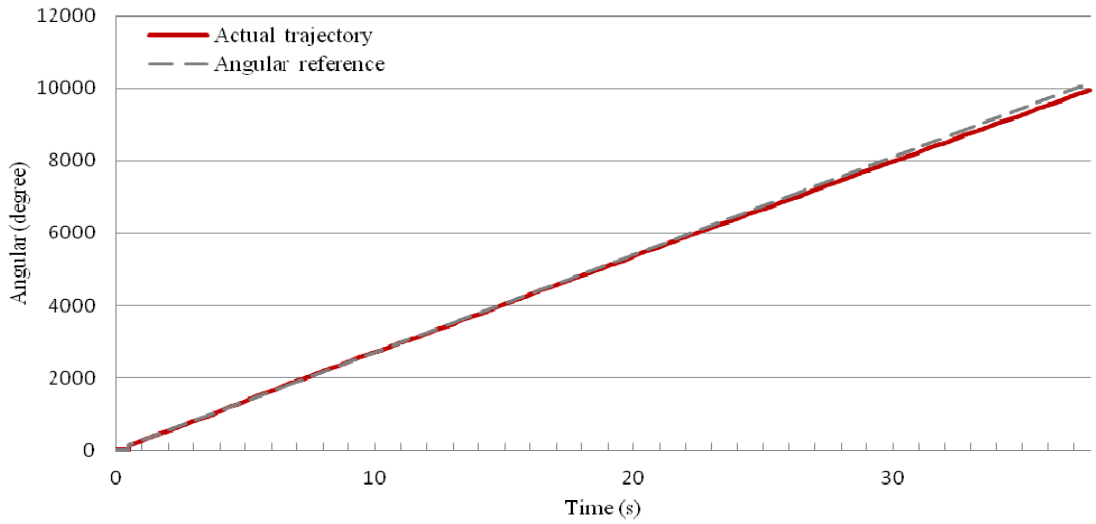


Figure 5- 7 Static performance of radial motion at 270 deg/s

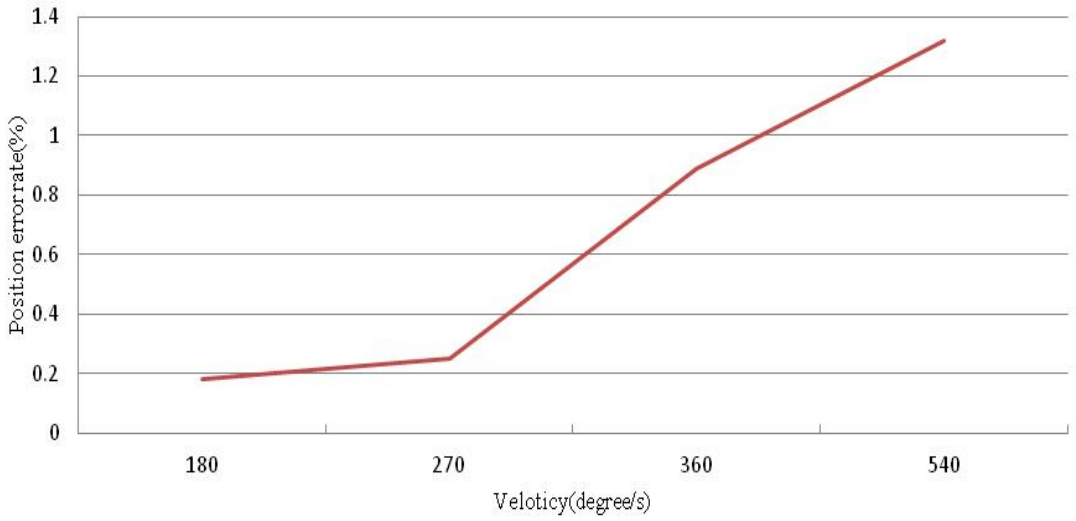


Figure 5- 8 Static characterization of radial motion

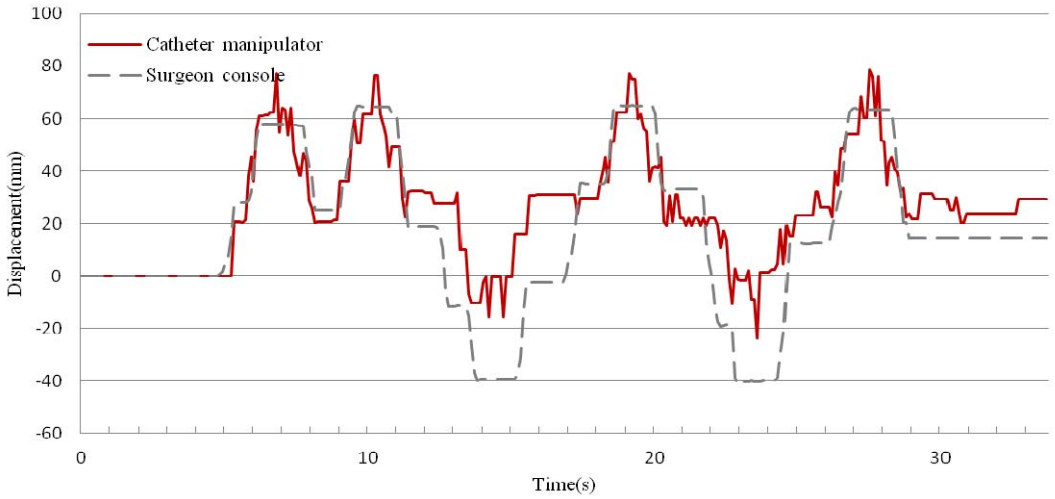


Figure 5- 9 The tracking curve of axial motion

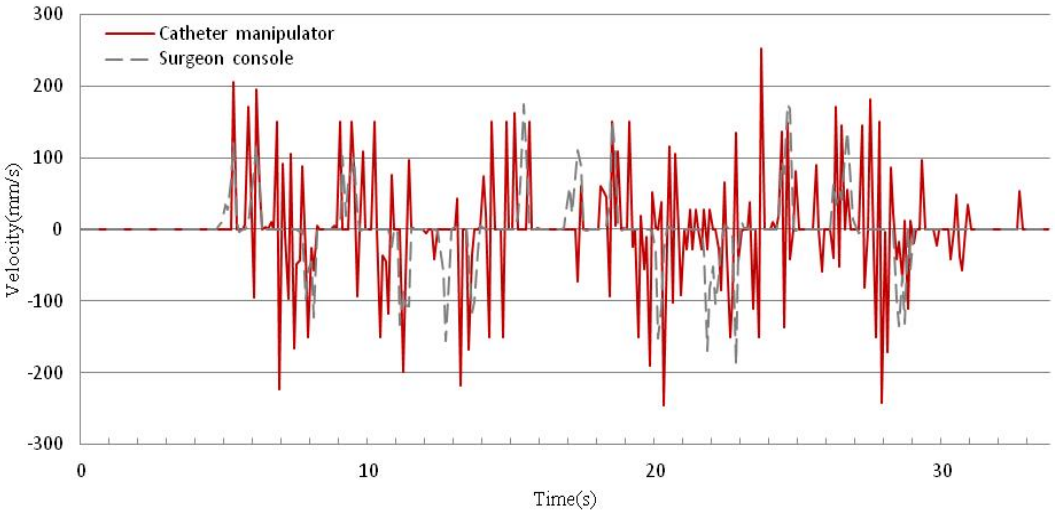


Figure 5- 10 The inserting velocity in both sides

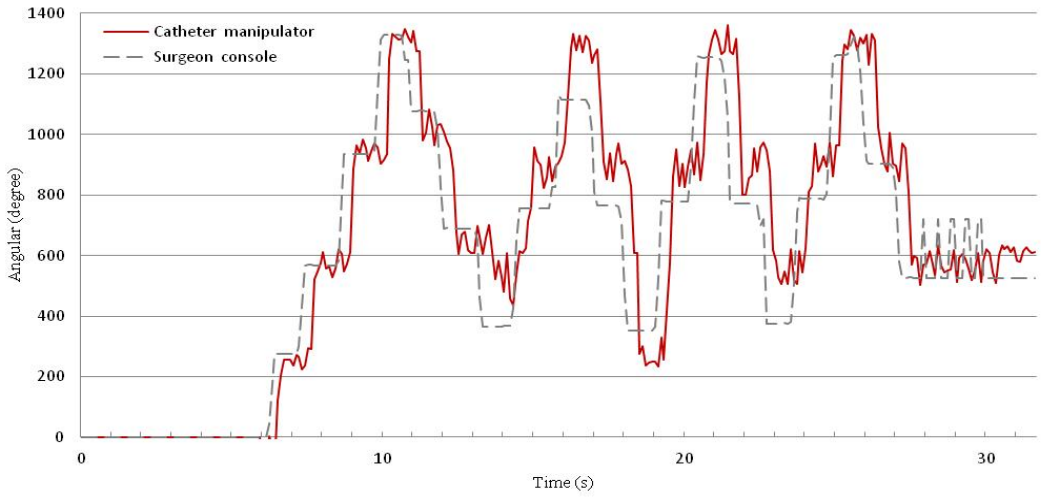


Figure 5- 11 The tracking curve of radial motion

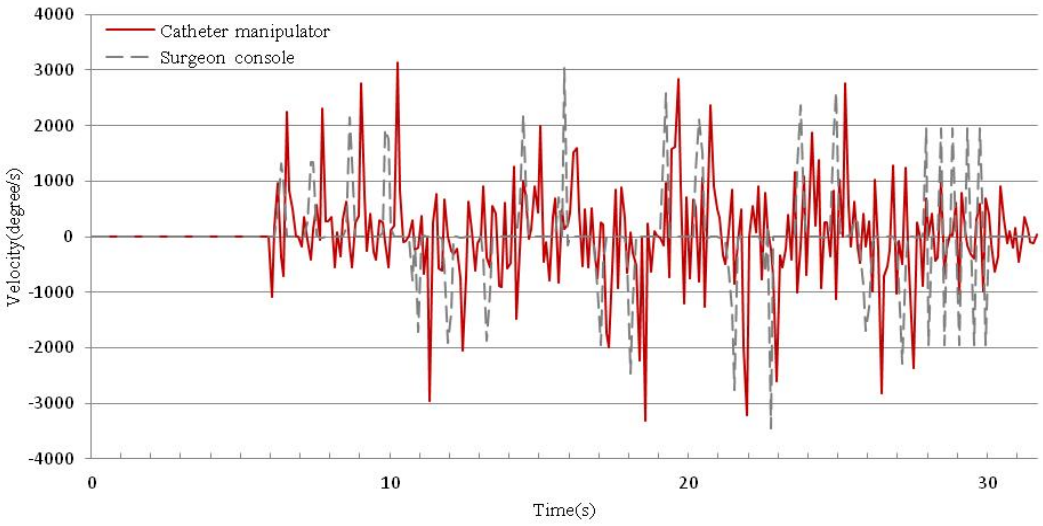


Figure 5- 12 The rotational velocity in both sides

5.3 Summary

The performance evaluation demonstrated the system's ability to measure and implement catheter motion within good specifications. The reported dynamic and static performance of surgeon console and accuracy and precision in motion implementing of catheter manipulator were good. In addition, to use the dexterous skills possessed by the surgeon should enable rapid acceptance of this technology while maintaining the remarkable success of conventional intervention.

Experiments are carried out to evaluate the performance of the robotic catheter manipulating system. The first experimental results described from Figure 5- 3 to Figure 5- 8 showed the dynamic and static characterization of the surgeon console. The surgeon console works well regardless of insertion or rotation the catheter as different frequency or the same velocity. In the catheter manipulator side, the precision of axial direction is 0.23mm, the rotation precision is 2.2^0 with a accuracy of 3.0^0. From the results we can find that the precision and accuracy in axial direction is better than the radial direction for both sides.

Figure 5- 9 shows the evaluation of the synchronization of axial direction. In this part, we did not consider the time delay during the performance evaluation. But for the tele-operation experiments, there has big lag at beginning operation. Maybe the tele-operation is difficult

in the unstable and uncertainty network environment. The surgeon console and the catheter manipulator were controlled by DSP, DSP communicates with each other by the serial port and LAN network. Based on the numbers of communication data and the distance, the local network communication was fast enough to make the lag below 1ms in theory. However, from Figure 5- 9 we could understand the movement curve of the surgeon console was faster than the catheter manipulator. It means that the lag existed. The lag time will be measured in the future. We repeated the procedures for 4 times. The manipulation time was about 30s. The axial average error between the surgeon console and the catheter manipulator was big that because of time delay and serial communication problem. Figure 5- 10 shows the rotation tracking error. As well as the surgeon console side, lag could be found in Figure 5- 11. The mean error of the rotation is below 15^0. It means a not stable radial motion. The Large overshoot mainly caused by the electromagnetic interference to the AD convertor which is used to drive the motor. The control circuit should be redesigned. And with different operator the radial tracking error is different because of the different rotation velocity. Finally, the fiber optic pressure sensor used to detect the contact force on tip of catheter was useful during the tele-operation. The surgeon could understand the situation about the catheter tip and know that how to avoid the damage to the patient as well as how to continue the next insertion.

Chapter 6

Insertion experiment in “Vitro”

6.1 Introduction

To validate the effectiveness of the robotic catheter manipulating system, we performed a simulation experiment to evaluate the characteristics of the master-slave robotic catheter operating system by using an endovascular evaluator (EVE) model, in this chapter, we will introduce the catheter inserting experiment in vitro by using an EVE model and its results, also analyzed and discussed the validity of the developed robotic catheter system based on the experimental results.

6.2 Experimental environment

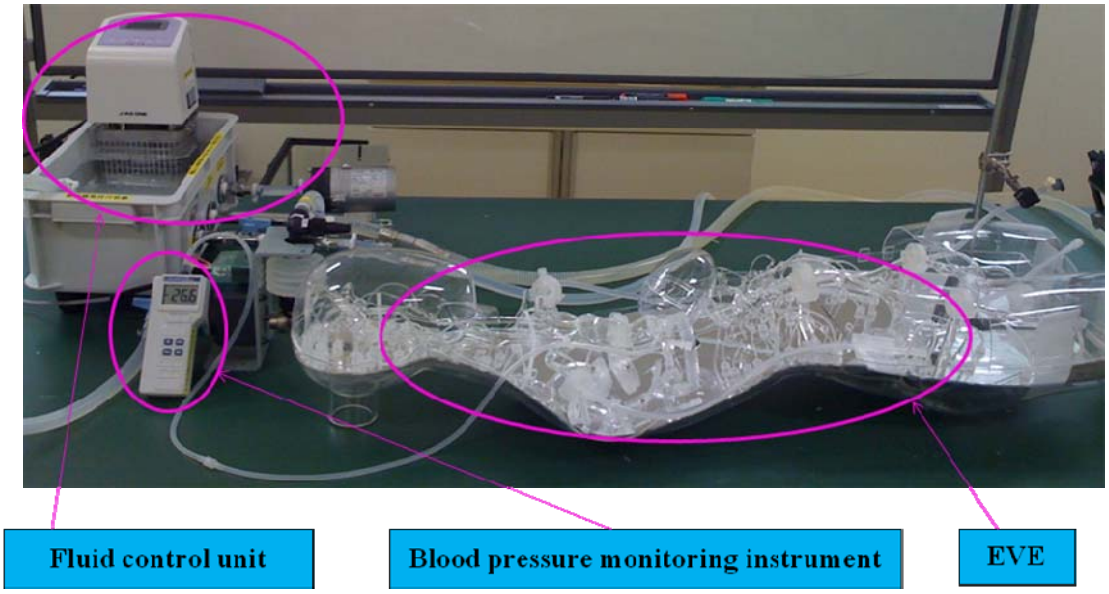


Figure 6- 1 EVE (Endovasucular Evaluator) model

We used EVE model as the experimental environment to carry out the catheter inserting experiment. The Figure 6-1 shows the EVE model which consisted of a fluid control unit and blood pressure monitoring instrument. The bending angles and radii of the tubes in the EVE are close to those of human arteries. The tubes were made of silicon rubber. The elasticity of the tubes was similar to that of a blood vessel wall. In order to keep the blood pressure of the EVE close to the blood pressure of a human, the fluid control unit was used to adjust the blood pressure, which was monitored with the blood pressure monitoring instrument. The operator operates the right handle on the master side to insert and

rotate the catheter, which is inserted into the EVE from the femoral artery, controlling the speed and position of the catheter.

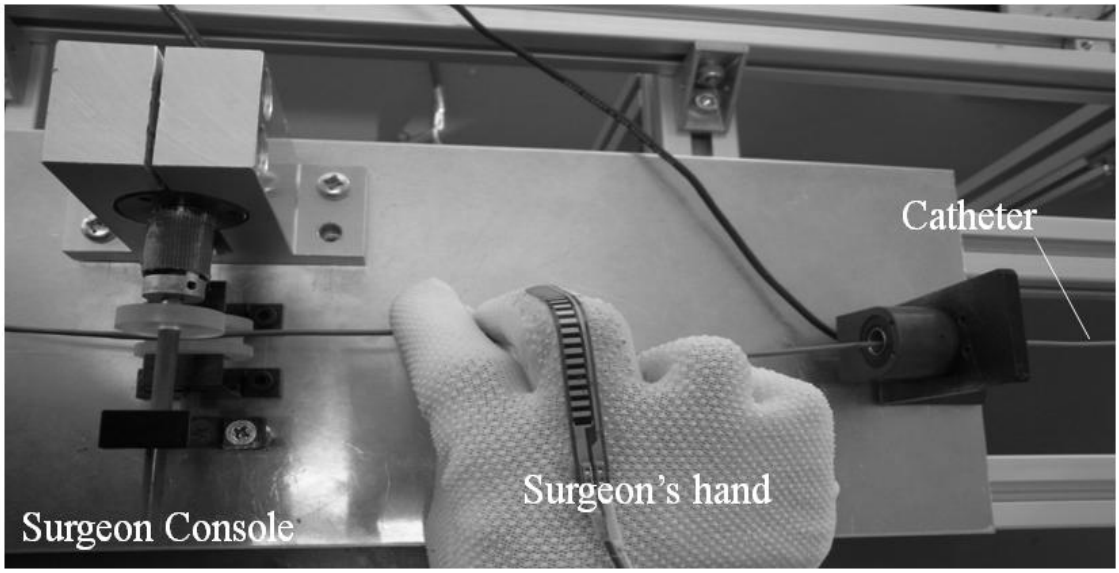
6.3 Experiment and results

The tele-operation experiments were carried out with LAN and the lag was above 300ms, especially much larger when started the operation. Figure 6- 2 shows the configuration for this case, an EVE simulator which consisted of a fluid control unit and blood pressure monitoring instrument was employed. The properties of the EVE simulator are similar to the blood vessels of human body. In order to keep the blood pressure of the EVE simulator similar to human body, the fluid control unit was used to adjust it every time.

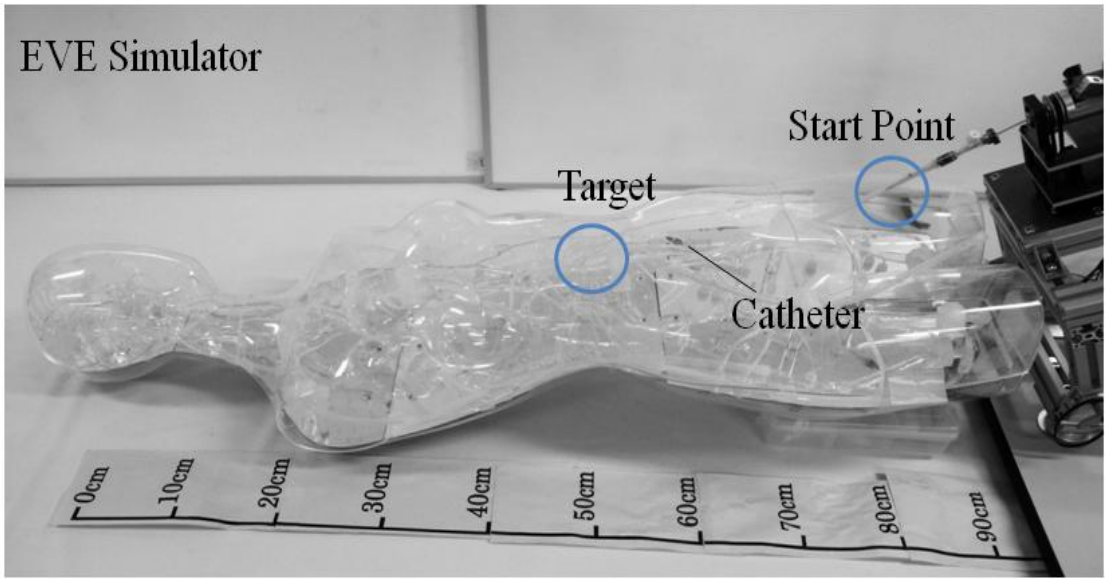
The server of communication was built in the surgeon console side. The surgeons would see the position of the catheter from the screen. In this experiment, our target was to insert the catheter to reach a goal position. The displacement of the surgeon console and the catheter manipulator are kept same. Figure 6- 3 shows the position tracking trajectory of the axial direction.

At beginning, we moved the catheter forward in the surgeon console, the catheter manipulator inserted the catheter suddenly after kept the stationary state in 3s because of the time delay. In consequence, the LAN was not stable at anytime. Then, we performed the experiments

from the start point to the target in two times. Due to the inserting route was not straight route, it has one branch after the start point. When the catheter went through the branch at the second time, the tip of it touched onto the wall of blood vessel. The fiber optic pressure sensor was used to measure the contact force in Figure 6- 4 described. The measured contact force was about 15mN. In the future, we will test the performance under the real situations like animal organs or clinical surgery.



(a) The surgeon console



(b) The catheter manipulator and the EVE simulator

Figure 6- 2 Experimental system

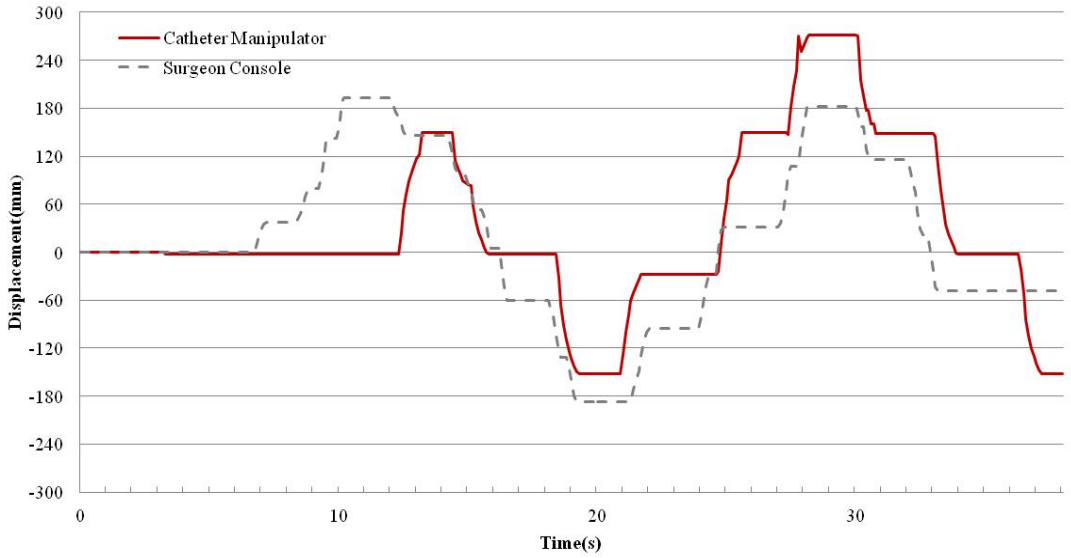


Figure 6- 3 Tracking trajectory of the axial direction

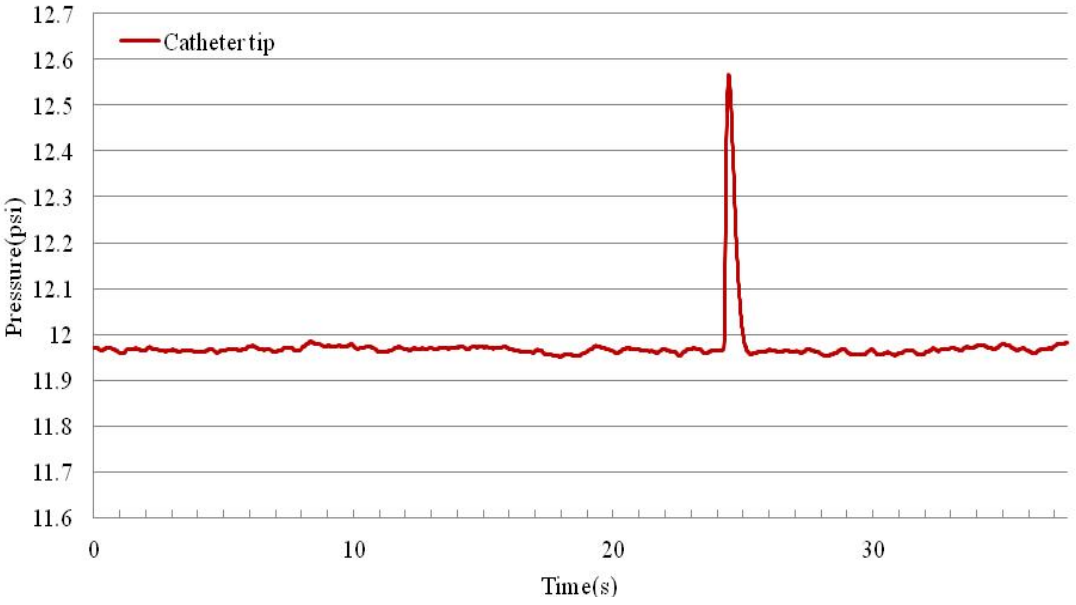


Figure 6- 4 Contact force signal described as pressure

6.4 Summary

In this chapter, the fiber optic pressure sensor used to detect the contact force on tip of catheter was useful during the tele-operation. The surgeon could understand the situation about the catheter tip and know that how to avoid the damage to the patient as well as how to continue the next insertion.

The presented experiments have demonstrated the system’s ability to measure and drive catheter in axial and radial motions. Combining the remote navigation on EVE simulator, implementation with the surgeon’s skill verifies this system as an effective approach to reduce surgeon’s radiation exposure and physical discomfort.

Chapter 7
Conclusions and future work

7.1 Conclusions

This thesis presents the design and evaluation of two kinds of robotic catheter systems for minimally invasive surgery. First, the intravascular environmental conditions were examined to understand the forces, displacements and tissue stiffness values experienced by an intravascular device while interacting with the fast-moving structures inside the vascular (Chapter 1). These information and insights were used to develop the mechanical design of the system, including end effectors and intravascular bracing strategies (Chapter 2). The robotic catheter system was analyzed to determine how best to control the position of the catheter and to measure the forces both the catheter tip and proximal end force applies to the operation procedures. In addition, the PID fuzzy controller has been applied in the tele-operation to improve operation accuracy (Chapter 3). Based on the first catheter operation system, we found that the surgeon would like use clinical catheter as the master controller by their dexterous skills. A new prototype robotic catheter manipulating system has been designed and constructed based on the requirements for the endovascular surgery

(Chapter 4). Finally, clinically relevant applications of the system were examined. These applications included a performance evaluation of the robotic catheter manipulating system (Chapter 5) and the insertion experiment into “Vitro” (Chapter 6). This work provides a solid foundation for the development of robotic catheter system and the design of 3D-guided to improve aneurysm repair (Chapter 7).

The contribution of this work is the elucidation of the benefits and challenges of performing surgery on the brain aneurysm with a robotic catheter system. Specific contributions include:

- Mechanical design of two kinds of catheter-based surgical systems
- Position control between master and slave side
- Establishing of force and visual feedback system
- Exploration of clinical applications of the system and experimental validation into “Vitro”

This work has produced a number of conclusions and insights regarding the challenges of implementing minimally invasive surgery using a robotic catheter system.

7.1.1 System Design

(1) The first robotic catheter system: A robot-assisted catheter system with force feedback and visual feedback was developed. We also proposed a mechanism which can be used to measure the operating force on the catheter, and the force information can be transmitted to

the master manipulator and generate a haptic feedback to the neurosurgery, via the haptic feedback, the neurosurgeon can decide whether inserting the catheter or not. We carried out the performance evaluation for the robot-assisted catheter system, and also, we did the tele-operation by PID fuzzy control method. The evaluated results and experimental results imply that the developed robot-assisted catheter system with force feedback and visual feedback work very well, it is suitable for supporting neurosurgeons to the vascular interventional surgery. In addition, the robot-assisted catheter system can be used to train unskilled or inexperience neurosurgeon.

(2) The second robot-assisted catheter system: a new prototype robotic catheter manipulating system has been designed and constructed based on the requirements for the endovascular surgery. Compared with system mentioned above, this system features a master controller called surgeon console, using two motion-sensing devices via control unit DSP to communicate the position and rotation information with slave side. Also, we designed the haptic device to provide the feeling back to surgeons. The whole system was evaluated in aspect of dynamic and static performance of the axial and radial motions. The results of synchronization experiments had to evaluate the accuracy and precision of sensed and replicated motions. Finally, Tele-operation had been done by EVE simulator to provide the performance under the similar situations.

7.1.2 User Interface

The user interface for monitoring the contact force between catheter and blood vessel were developed. The force monitoring system and danger avoiding system were applied in the robotic catheter system. They can monitor the contact force information in real time. If the force on the catheter goes beyond the safety range, the doctor will get warning information from both monitors and controller. The experimental results imply that the developed user interfaces for monitoring force is effective to help neurosurgeon avoid danger during operation.

7.1.3 Real-Time Position Sensing

We designed the fuzzy PID controller that is used in the slave side, the accuracy of axial and rotational movement during the remote operations have been improved and comparing with the conventional PID control method. The presented fuzzy PID controller is much better quality of response during insertion and rotation. The tracking error with fuzzy PID is below 3mm and 1.5° during the remote control. Although there also has a little error in the rotational and axial movement because of time delay, it also can satisfy the application of practical application in minimally invasive surgery. Experimental results confirm the fact that the fuzzy logic control method is suitable to be used and it is easy to understand and employ the system dynamic model freely. Finally, it should be said that the fuzzy PID control

algorithm is the alternative approach to be used as the system control method.

7.1.4 Haptic Feedback

Simple haptic feedback has limited value for catheter procedure guidance. The catheter force sensor used in this work in slave side only provides single DOF force information and thus can only measure forces applied to the catheter tip along the length of the tool. Lateral forces are not sensed at present. As a result, many of the forces involved in surgical procedures, such as cutting, approximating, and affixing tissue, cannot be perceived by the user. One area where haptic feedback might be useful is for augmenting the poor quality of some 3D imaging. The haptic interface could inform the clinician when the tool first makes contact with the vascular and thus prevent the clinician from accidentally puncturing the vascular or applying unwanted forces.

Also we designed a new haptic device which can change the friction on the catheter and provide the feeling back to the surgeon in the master side as Figure 4-8 shown. If the catheter manipulator side detects resistance force, the haptic device will generate it as near as the real one. We also used 4 springs for the system stability. In fact, when the catheter is totally stopped by the feedback pressure, we don’t have any information about the force that the surgeon puts on the catheter. So, we will consider it and discuss the situation about the friction force measuring and force feedback providing in the future.

7.1.5 Clinical Applications

The robotic catheter system has the capacity to deliver a number of procedure-specific end effectors. The eventual goal of the system is to use multiple, coordinated robotic catheters to complete more complex surgical tasks, such as closing heart defects and repairing damaged vascular parts. This level of performance, however, will require significant improvements in imaging and robot control. In the meanwhile the robotic catheter system can make the greatest contribution by improving the success of currently conducted procedures and deploying percutaneous implants. For example, the deployment of the MitralClip mitral valve repair system (Abbott Laboratories, Abbott Park, Illinois, USA), which requires grabbing the moving mitral valve leaflets, could be made easier if it used the 3DUS-guided motion compensation system employed in this research.

7.2 Future work

A number of improvements can be made to the robotic catheter system in the areas of mechanical design and control. The system can also be applied to new medical applications, including neurosurgery and peripheral vascular disease.

7.2.1 Mechanical Design

The areas where the mechanical design of the current robotic catheter

system could be improved include the limited DOF of the actuation system, the lack of sophisticated intravascular bracing and new end effectors for more complex surgical procedures.

Additional fast-servoed degrees of freedom would allow the catheter to track aneurysm with complex three dimensional trajectories. This would expand the range of aneurysm structures that can be treated with the robotic catheter system to almost any part of the aneurysm surface. Increasing the actuated DOF on the catheter could be accomplished by adding smart material to the bending and twisting mechanism.

Finally, more sophisticated end effectors must be developed to realize the goal of completing complicated surgeries with robotic catheters. The main challenge of developing these tools is miniaturizing existing technologies and integrating them with the catheter system.

7.2.2 Control

The current limitations of the robotic catheter control system are accurate position control of the catheter tip and the 3D image servoing system. The model-based feedforward position control of the catheter tip is not able to adapt to changes in the environment or catheter configuration. This shortcoming could be overcome with accurate tip position measurements.

An ideal method to measure the catheter tip position is the 3D system; however, the current ultrasound systems are limited by low resolution, noise, and substantial latencies. Improved imaging would

not only improve catheter control, but also would allow for more complex tissue motions to be tracked. For example, an arbitrary point on the ventricle wall could be tracked for ablation therapy. This is an ongoing area of research that requires innovations to improve both the 3D technology and the computer vision algorithms to enable tracking of less distinct objects and features in the ultrasound images.

7.2.3 New Applications

In addition to the clinical applications described in this thesis, the robotic catheter system could be adapted for a number of medical procedures and therapies. In the area of cardiac surgery, the system could be used to deploy prosthetic valves, suture annuloplasty rings to improve valve function, and repair or reattach chordae and other parts of the subvalvular apparatus. Beyond the heart, the catheter system could also be used to treat peripheral vascular disease in other parts of the body. For example, the catheter system could be used to reconstruct occluded vessels or reposition stents in the appendages. The system would use 3D guidance to track vascular structures and force control to regulate tool-tissue interactions and perform effective surgical repairs.

All of these applications will be explored in future work to identify medical challenges where the robotic catheter system could make the greatest impact and provide the greatest improvements to patient care.

References

[Kanagaratnam08] Prapa Kanagaratnam, Michael Koa-Wing, Daniel T. Wallace, Alex S. Goldenberg, Nicholas S. Peters, D, Wyn Davies, Experience of robotic catheter ablation in humans using a novel remotely steerable catheter sheath, *Journal of Interventional Cardiac Electrophysiology,* vol.21, pp.19–26, 2008.

[Chun07] JKR. Chun, S. Ernst, S. Matthews, B. Schmidt, D. Bansch, S. Boczor, et al, Remote-controlled catheter ablation of accessory pathways: results from the magnetic laboratory, *European Heart Journal,* vol. 28(2), pp. 190–195, 2007.

[OKB Medical Ltd.] http://www.okbmedical.com/angio

[Pappone06] Carlo Pappone, Gabriele Vicedomini, Francesco Manguso, Filippo Gugliotta, Patrizio Mazzone, Simone Gulletta, Nicoleta Sora, Simone Sala, Alessandra Marzi, Giuseppe Augello, Laura Livolsi, Andreina Santagostino, Vincenzo Santinelli, Robotic Magnetic Navigation for Atrial Fibrillation Ablation, *Journal of the American College of Cardiology,* vol. 47, pp 1390-1400, 2006.

[Willems10] S. Willems, D. Steven, H. Servatius, BA. Hoffmann, I. Drewitz, K. Mullerleile, MA. Aydin, K. Wegscheider, TV. Salukhe, T. Meinertz, T. Rostock, Persistence of Pulmonary Vein Isolation After Robotic Remote-Navigated Ablation for Atrial Fibrillation and its Relation to Clinical Outcome, *Journal of Interventional*

Cardiac Electrophysiology, vol 21: pp. 1079–1084, 2010.

[Stargen Inc.] http://www.stargen.eu/products/niobe/.

[Saliba08] Walid Saliba, Vivek Y. Reddy, Oussama Wazni, Jennifer E. Cummings, et.al, Atrial Fibrillation Ablation Using a Robotic Catheter Remote Control System: Initial Human Experience and Long-Term Follow-Up Results, *Journal of the American College of Cardiology , vol.* 51, pp 2407-2411, 2008

[Preusche02] Carsten Preusche, Tobias Ortmaier, Gerd Hirzinger, Teleoperation concepts in minimal invasive surgery. *Control Engineering Practice,* vol. 10, pp. 1245-1250, 2002.

[Ikeda05] S. Ikeda, F. Arai, T. Fukuda, M. Negoro, K.Irie, and I. Takahashi, et. al., In Vitro Patient-Tailored Anatomaical Model of Cerebral Artery for Evaluating Medical Robots and Systems for Intravascular Neurosurgery, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1558-1563, 2005.

[Peirs04] Jan Peirs, Joeri Clijnen, Dominiek Reynaerts, Hendrik Van Brussel, Paul Herijgers, Brecht Corteville, et. al., A micro optical force sensor for force feedback during minimally invasive robotic surgery, *Sensors and Actuators A : Physical*, vol.115: pp. 447-455.

[Sedaghati05] R. Sedaghati, J. Dargahi, H. Singh. Design and modeling of an endoscopic piezoelectric tactile sensor, *International Journal of Solids and Structures*, vol.42, pp.5872–5886, 2005.

[Feng06] Weixing Feng, Shuxiang Guo, Changmin Chi1, Huanran Wang, Kejun Wang and Xiufen Ye, Realization of a Catheter Driving Mechanism with Micro tactile sensor for Intravascular Neurosurgery, *Proceedings of the 2006 IEEE International Conference on Robotics and Biomimetics,* pp1628-1633, 2006.

[Goto09] T. Goto, T. Miyahara, K. Toyoda, J. Okamoto, Y. Kakizawa, J. Koyama, M. G. Fujie, K. Hongo, Telesurgery of microscopic micromanipulator system "NeuRobot" in neurosurgery: Interhospital Preliminary Study, Journal of Brain Disease, 2009; 1:45-53.

[Fukuda94] T. Fukuda, S. Guo, K. Kosuge, F. Arai, M. Negoro, K. Nakabayashi, Micro Active Catheter System with Multi Degree of Freedom, Proceedings of 1994 IEEE International Conference on Robotic and Automation, 1994; 3:2290-2295.

[Guo96] S. Guo, T. Nakamtra, T. Fukuda, K. Oguro, M. Negoro, Micro active catheter using ICPF actuator characteristic evaluation, Proceedings of IEEE the 22nd Annual International Conf. on Industrial Electronics, Control, and Instrumentation, 1996; pp.1312-1317.

[Fu11] Y. Fu, A. Gao, H. Liu, S. Guo, The master-slave catheterization system for positioning the steerable catheter, International Journal of Mechatronics and Automation, 2011; 1(3), in press.

[Pan07] Bo Pan, Yili Fu and Shuguo Wang, Position Planning for Laparoscopic Robot in Minimally Invasive Surgery, Proceedings of 2007 IEEE International Conference on Mechatronics and Automation, 2007; pp. 1056-1061.

[Arai96] F.Arai, M.Tanimoto, T. Fukuda, K.Shimojima, H.Matsuura and M. Negoro, Distributed Virtual Environment for Intravascular tele-surgery using Multimedia Telecommnication, Proceedings of VRAIS' 96, pp.79-85, 1996.

[Arai02] F. Arai, R. Fujimura, T. Fukuda, M. Negoro, New catheter driving method using linear stepping mechanism for intravascular neurosurgery. Proceedings of the 2002 IEEE International Conference on Robotic & Automation, 2002; pp. 2944-2949.

[Feng06] W. Feng, C. Chi, H. Wang, K. Wang, X. Ye, S. Guo, Highly precise catheter driving mechanism for intravascular neurosurgery. Proceedings of 2006 IEEE International Conference on Mechatronics and Automation, 2006; pp. 990-995.

[Guo07] S. Guo, H. Kondo, J. Wang, J. Guo, T. Tamiya, A new catheter operating system for medical applications. Proceedings of the 2007 IEEE/ICME International Conference on Complex Medical Engineering, 2007; pp. 82-87.

[Wang08] J. Wang, S. Guo, H. Kondo, J. Guo, T. Tamiya, A novel catheter operating system with force feedback for medical

applications, International Journal of Information Acquisition, 2008; 5(1):83-91.

[Guo10] J. Guo, N. Xiao, S. Guo, A force display method for a novel catheter operating system, Proceedings of the 2010 IEEE International Conference on Information and Automation, 2010; pp. 782-786.

[Guo10] J. Guo, N. Xiao, S. Guo, T. Tamiya, Development of a force information monitoring method for a novel catheter operating system, Information: An International Interdisciplinary Journal, 2010; 1(6):1999- 2009.

[Puangmali08] P. Puangmali, K. Althoefer, L.D. Seneviratne, D. Murphy, P. Dasgupta, State-of-the-art in force and tactile sensing for minimally invasive surgery. IEEE Sensors Journal, 2008; 8(4):371-381.

[Polygerinos09] P. Polygerinos, T. Schaeffter, L. Seneviratne, K. Althoefer, Measuring tip and side forces of a novel catheter prototype: a feasibility study, Proceedings of the 2009 International Conference on Intelligent Robots and System, 2009; pp. 966-971.

[Takashima05] K. Takashima, K. Yoshinaka, T. Okazaki, K. Ikeuchi, An endoscopic tactile sensor for low invasive surgery. Sensors and Actuators, A 2005; 119:372-383.

[Takashima07] K. Takashima, R. Shimomura, T. Kitou, H. Terada, K.

Yoshinaka, K. Ikeuchi, Contact and friction between catheter and blood vessel. Tribology International, 2007; 40:319-328.

[Wang11] X. Wang, M. Meng, Perspective of active capsule endoscope: actuation and localization, International Journal of Mechatronics and Automation, 2011; 1(1):38-45.

[Marcelli08] E. Marcelli, L. Cercenelli, G. Plicchi, A novel telerobotic system to remotely navigate standard electrophysiology catheters, Computer in Cardiology, 2008; 35:137-140.

[Preusche02] C. Preusche, T. Ortmaier, G. Hirzinger, Teleoperation concepts in minimal invasive surgery. Control Engineering Practice, 2002; 10:1245-1250.

[Srimathveeravalli10] G. Srimathveeravalli, T. Kesavadas, X. Li, Design and fabrication of a robot mechanism for remote steering and positioning of interventional devices, The International Journal of Medical Robotics and Computer Assisted Surgery, 2010; 6:160-170.

[Wang10] Tianmiao Wang, Dapeng Zhang and Liu Da, Remote-controlled vascular interventional surgery robot, The International Journal of Medical Robotics and Computer Assisted Surgery, 2010; 6:194-201.

[Da08] Liu Da, Dapeng Zhang and Tianmiao Wang, Overview of the vascular international robot, The International Journal of Medical

Robotics and Computer Assisted Surgery, 2008; 4:289-294.

[Jayender06] J. Jayender, R.V.Patel and S.Nikumb, Robot-assisted Catheter Insertion using Hybrid Impedance Control, Proceedings of the 2006 IEEE International Conference on Robotics & Automation, pp.607-612, 2006.

[Jayender07] J. Jayender, M.Azizian, R.V.Patel, Autonomous Robot-Assisted Active Catheter Insertion using Image Guidance, Proceedings of the 2007 International Conference on Intelligent Robots and System, 2007; pp. 889-894.

[Jayender08] J. Jayender, M.Azizian, R.V.Patel, Autonomous Image Guided Robot-Assisted Active Catheter Insertion, IEEE Transactions on Robotics, Vol.24, No.4, August 2008.

[Lim96] Geunbae Lim, Kitae Park, M.Sugihara, Kazuyuki Minami, Msayoshi Esashi, Future of active catheters, Sensors and Actuators A 56 (1996) 113-121.

[Ikuta10] Precise bending angle control of hydraulic active catheter by pressure pulse drive, Proceedings of the 2010 International Conference on Robotics and Automation, 2010; pp. 5588-5593.

[Guo08] S.Guo, J.Guo, N.Xiao and T.Tamiya, Force sensors-based a Novel Type of Catheter Operating System, the 26th Annual Conference of the Robotics Society of Japan, Kobe, 2k1-01, 2008(in Japanese).

[Tanimoto98] M. Tanimoto, F. Arai, T. Fukuda, and M. Negoro. Augmentation of Safety in Teleoperation System for Intravascular Neurosurgery, Proceedings of the 1998 IEEE International Conference on Robotics & Automation, pp.2890-2895, 1998.

[Onda08] Kazushi Onda, Shigen Yasunaka, Jumpei Arata, Naohiko Sugita et al, A remote surgery experiment between Japan-Thailand using a minimally invasive surgical system, the 26th Annual Conference of the Robotics Society of Japan, Kobe, 1k2-05, 2008(in Japanese).

[Ide08] Masaru Ide, Yuusuke Fujii, Bumpei Fujioka, et al, Development of the training system for catheter guide, the 26th Annual Conference of the Robotics Society of Japan, Kobe, 2k1-07, 2008(in Japanese).

[Ide08] Masaru Ide, Takashi Komeda, et al, Development of the Force-Reflecting Master Slave System for Catheter Guide without Force Sensor, the 26th Annual Conference of the Robotics Society of Japan, Kobe, 2k1-04, 2008(in Japanese).

[Tung08] Alexander T. Tung, Byong-Ho Park, David H.Liang and Gunter Niemeyer, Laser-machined shape memory alloy sensors for position feedback in active catheters, Sensors and Actuators A 147 (2008) 83-92.

[Mitsuishi07] Mamoru Mitsuishi, Medical robot and master slave

system for minimally invasive surgery, Proceedings of the 2007 IEEE/ICME International Conference on Complex Medical Engineering, 2007; pp. 8-13.

[Tholey05] Gregory Tholey, BS, Jaydev P. Desai and Andres E. Castellanos, Force Feedback Plays a Significant Role in Minimally Invasive Surgery, Annals of Surgery, 2005 January, 241(1):102-109.

[Ernst08] Sabine Ernst, Magnetic and robotic navigation for catheter ablation "Joystick ablation", Journal of Interventional Cardiac Electrophysiology, Vol.23, No.1, 41-44, 2008.

[Hasegawa07] Shin Hasegawa, Hidetaka Funayama and Daming Wei, A Virtual Reality Simulating Catheter Manipulations, International Journal of Bioelectromagnetism, Vol.9, No.2,125-126, 2007.

[Kanagaratnam08] Prapa Kanagaratnam, Michael Koa-Wing, Daniel T. Wallace et al, Experience of robotic catheter ablation in humans using a novel remotely steerable catheter sheath, J Interv Card Electrophysiol (2008) 21: 19-26.

[Darvish09] Behafarid Darvish, Siamak Najarian, Elham Shirzad and Roozbeh Khodambashi, A Novel Tactile Force Probe for Tissue Stiffness Classification, American Journal of Applied Sciences 6 (3): 512-517, 2009.

[Tercero07] C.Tercero, S.Ikeda, T.Uchiyama, T.Fukuda, F.Arai et al,

Autonomous catheter insertion system using magnetic motion capture sensor for endovascular surgery, The international journal of medical robotics and computer assisted surgery, 2007; 3:52-58.

[Chen10] Chen Daguo, Shen Jie and Yan Yonghua, An Overview of Robot Assisted Catheter Insertion System, Chinese Journal of Medical Instrumentation, Vol.34, No.1, pp.35-38, 2010(In Chinese).

[Thakur09] Yogesh Thakur, Jeffrey S. Bax, David W. Holdsworth and Maria Drangova, Design and Performance Evaluation of a Remote Catheter Navigation System, IEEE Transactions on Biomedical Engineering, Vol.56, No.7, pp.1901-1908, July, 2009.

[Arata08] Jumpei Arata, Hiroki Takahashi, Shigen Yasunaka, et al, Impact of networktie-delay and force feedback on tele-surgery, International Journal of CARS (2008) 3:371-378.

[Onda10] Kazushi Onda, Takayuki Osa, Naohiko Sugita, Makoto Hashizume and Mamoru Mitsuishi, Asynchronous Force and Visual Feedback in Teleoperative Laparoscopic Surgical System, Proceedings of the 2010 International Conference on Intelligent Robots and System, 2010; pp. 18-22.

[Padalino13] David J Padalino, Amit Singla, Walter Jacobsen and Eric M Deshaies, Enterprise Stent for Waffle-cone Stent-assisted Coil Embolization of Large Wide-necked Arterial Bifurcation Aneurysms, Surgical Neurology International, 2013.

[Spiotta 12] Alejandro M Spiotta, Anne Marie Wheeler, Saksith Smithason, Ferdinand Hui and Shaye Moskowitz, Comparison of Techniques for Stent Assisted Coil Embolization of Aneurysms, Journal of Neurointerventional Surgery, 4(5):339-344, 2012.

Publication List

International Journal Papers

1. **Xu Ma**, Shuxiang Guo, Nan Xiao, Shunichi Yoshida, Takashi Tamiya and Masahiko Kawanishi, ‘’Evaluating Performance of a Novel Developed Robotic Catheter Manipulating System’’, Journal of Micro-Bio Robotics (JMBR), DOI 10.1007/s12213-013-0068-2, 2013. 03.
2. **Xu Ma**, Shuxiang Guo, Nan Xiao, Jian Guo, Shunichi Yoshida, Takashi Tamiya and Masahiko Kawanishi, ‘’Development of a Novel Robotic Catheter Manipulating System with Fuzzy PID Control’’, International Journal of Intelligent Mechatronics and Robotics, Vol. 2, No. 2, pp. 58-77, 2012.
3. Jian Guo, Shuxiang Guo, Nan Xiao, **Xu Ma**, Shunichi Yoshida, Takashi Tamiya and Masahiko Kawanishi, ‘’A Novel Robotic Catheter System with Force and Visual Feedback for Vascular Interventional Surgery’’, International Journal of Mechatronics and Automation, Vol. 2, No. 1, pp. 15-24, 2012.

International Conference Papers

4. **Xu Ma**, Shuxiang Guo, Nan Xiao, Shunichi Yoshida and Takashi Tamiya, "Development of a Novel Robotic Catheter Manipulating System", Proceedings of 2012 International conference on Manipulation, Manufacturing and Measurement on the Nanoscale, No.012, 2012. 08.

5. **Xu Ma**, Shuxiang Guo, Nan Xiao, Jin Guo, Shunichi Yoshida, Takashi Tamiya, Masahiko Kawanishi and Baofeng Gao, "NARX Model-based Identification for the Developed Novel Robotic Catheter Manipulating System", Proceedings of 2012 IEEE International Conference on Mechatronis and Automation, pp.2225-2229, 2012. 08.

6. **Xu Ma**, Shuxiang Guo, Nan Xiao, Baofeng Gao, Jin Guo, Takashi Tamiya and Masahiko Kawanishi, "Remote Catheterization Using a New Robotic Catheter Manipulating System", Proceedings of the 2013 IEEE/ICME International Conference on Complex Medical Engineering, pp.394-398, 2013. 05.

7. **Xu Ma**, Shuxiang Guo, Nan Xiao, Jian Guo, Shunichi Yoshida, "Development of Fuzzy PID Controller for a Novel Robotic Catheter Operating System", Proceedings of the 2011 3M-NANO International Conference on Manipulation, Manufacturing and Measurement on the Nanoscale, No.092. 2011. 08.

8. **Xu Ma**, Shuxiang Guo, Nan Xiao, Jian Guo and Shunichi Yoshida,

"Development of a Novel Robot-Assisted Catheter System with Force Feedback", Proceeding of the 2011 IEEE International Conference on Mechatronics and Automation, pp.107-111, 2011, 08.

9. **Xu Ma**, Shuxiang Guo, Nan Xiao, Jian Guo, Xufeng Xiao, Shunichi Yoshida, "Development of a PID Controller for a Novel Robotic Catheter System", Proceedings of the 2011 IEEE/ICME International Conference on Complex Medical Engineering, pp. 64-68, 2011. 05.

10. 田中 貴雄, 郭 書祥, 肖 楠, **馬 旭**, "遠隔カテーテル操作システムに用いる医師の指運動計測システムの開発", 平成24年度 電気関係学会四国支部連合大会, 14-20, 高松, 日本, 2012.9.29

11. Nan Xiao, Shuxiang Guo, Baofeng Gao, **Xu Ma**, Takashi Tamiya and Masahiko Kawanishi, "Internet-based Robotic Catheter Surgery System --System Design and Performance Evaluation", Proceedings of the 2012 IEEE International Conference on Automation and Logistics, pp.645-651, 2012. 08.

12. Jian Guo, Shuxiang Guo, Nan Xiao, **Xu Ma**, Shunichi Yoshida, Takashi Tamiyaand Masahiko Kawanishi, "Feasibility Study for a Novel Robotic Catheter System", Proceeding of the 2011 IEEE International Conference on Mechatronics and Automation, pp.205-210, 2011. 08.

13. Jian Guo, Shuxiang Guo, Nan Xiao, **Xu Ma**, Shunichi Yoshida, Takashi Tamiya and Masahiko Kawanishi, "A New Master-slave Robotic Catheter System", Proceedings of the 2011 IEEE/ICME International Conference on Complex Medical Engineering, pp. 610-613, Harbin, China, 2011. 05.

14. Nan Xiao, Shuxiang Guo, Baofeng Gao, **Xu Ma**, Takashi Tamiya and Masahiko Kawanishi, "Controller Design for a Robotic Catheter Teleoperation System", Proceedings of 2012 IEEE International Conference on Mechatronics and Automation, pp.353-358, 2012. 08.

15. Jin Guo, Shuxiang Guo, Nan Xiao, Baofeng Gao, **Xu Ma** and Mohan Qu, "A Method of Decreasing Time Delay for A Tele-surgery System", Proceedings of 2012 IEEE International Conference on Mechatronics and Automation, pp.1191-1195, 2012. 08.

Appendix I

The program code of the robot-assisted catheter system (Visual Studio 2010)

```
#include "stdafx.h"
#include "MasterData.h"
#include "DlgMasterSever.h"
#include "MasterGlobalVariables.h"
#include "MasterGlobalFunctions.h"
#include "afxdialogex.h"
/*#include "serialport.h"*/

#define DEBUG_LAN_WORK 0

IMPLEMENT_DYNAMIC(CDlgMasterSever, CDialog)

CDlgMasterSever::CDlgMasterSever(CWnd* pParent /*=NULL*/)
	: CDialog(CDlgMasterSever::IDD, pParent)
	, m_bIsListening(false)
	, m_nPortNum(6803)
	, m_nportNO(6803)
	,m_bIsSerialPortOpen(false)
	,m_bIsSerialComm(false)
	,m_nSerialCNT(0)
	,m_bIsLanComm(false)
{
	//m_serversocket = new CServerSocket();
}

CDlgMasterSever::~CDlgMasterSever()
{
	//delete m_serversocket;
}

void CDlgMasterSever::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	DDX_Text(pDX, IDC_EDIT1, m_nportNO);
}


BEGIN_MESSAGE_MAP(CDlgMasterSever, CDialog)
	ON_BN_CLICKED(IDOK, &CDlgMasterSever::OnBnClickedBtnCreateServer)
	ON_BN_CLICKED(IDCANCEL, &CDlgMasterSever::OnBnClickedBtnSendByLan)
	ON_BN_CLICKED(IDC_BTN_EXIT, &CDlgMasterSever::OnBnClickedBtnExit)
	ON_MESSAGE(WM_MSG_LAN_RX, OnGetLanRXMsg)                    //      for lan rx
```

```
        ON_MESSAGE(WM_MSG_LAN_TX, OnGetLanTXMsg)                    //            for lan
tx
        ON_MESSAGE(WM_COMM_RXCHAR,OnGetComRxMsg)            // for com rx
        ON_MESSAGE(WM_MSG_COM_TX, OnGetComTXMsg)            //for com tx
        ON_BN_CLICKED(IDC_BTN_STRCOM, &CDlgMasterSever::OnBnClickedBtnStrcom)
        ON_BN_CLICKED(IDC_BTN_OPENCOM, &CDlgMasterSever::OnBnClickedBtnOpencom)
END_MESSAGE_MAP()



void CDlgMasterSever::showMessage(const char* msg)
{
        CListBox* plistbox = (CListBox*)GetDlgItem(IDC_LIST1);
        plistbox->SetTopIndex(plistbox->AddString((LPCTSTR)msg));
}


void CDlgMasterSever::OnBnClickedBtnCreateServer()
{
        MasterSTFromCom[0] = 0xa;
        MasterSTFromCom[1] = 0xb;
        MasterSTFromCom[2] = 0xc;
        MasterSTFromCom[3] = 0xd;
        MasterSTFromCom[15] = 0x1;

        if (!m_blsListening)
        {
                        UpdateData(true);
                        m_nPortNum = m_nportNO;
                        m_listensocket.m_hwnd = this->GetSafeHwnd();

                        AfxSocketInit();
                        if (m_listensocket.Create(m_nPortNum, SOCK_STREAM, 0))
                        {
                                                showMessage("Server built.");

                                                BYTE   *p;
                                                char   temp[100];
                                                struct   hostent   *hp;
                                                CString ip;

                                                if(gethostname(temp,
sizeof(temp)) == 0)

                                                if((hp=gethostbyname(temp)) !=
0)

                                                {
                                                                        p
=(BYTE   *)hp-> h_addr;
```

```
        ip.Format(_T("Sever IP: %d.%d.%d.%d : %d"),    p[0],    p[1],    p[2],    p[3],
m_nPortNum);

        showMessage((LPSTR)(LPCTSTR)ip);
                                                                }

                                                                if
( !m_listensocket.Listen(5))
                                                                {
                                                                        int
nErrorCode = 0;

        nErrorCode = m_listensocket.GetLastError();
                                                                        TCHAR
errMsg[2048] = {0};

        FormatMessage( FORMAT_MESSAGE_FROM_SYSTEM,
                                                        NULL,

                                                        nErrorCode,

                                                        MAKELANGID(LANG_NEUTRAL,
SUBLANG_DEFAULT),

                                                        errMsg,

                                                        sizeof(errMsg)/sizeof(TCHAR),

                                                        NULL            );


        showMessage((LPSTR)errMsg);
                                                                        return;
                                                                }else
                                                                {

        m_blsListening = !m_blsListening;

        GetDlgItem(IDOK)->SetWindowText(_T("Disconnect"));
                                                                }

                                        }else                   // create error
                                        {
                                                                int nErrorCode = 0;
                                                                nErrorCode =
m_listensocket.GetLastError();
                                                                TCHAR errMsg[2048] = {0};
                                                                FormatMessage(
```

```
		FORMAT_MESSAGE_FROM_SYSTEM,

				NULL,

				nErrorCode,

				MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),

				errMsg,

				sizeof(errMsg)/sizeof(TCHAR),

				NULL		);

							showMessage((const char*)errMsg);
							return;
			}
		}else			//	is listening
		{
			if (m_listensocket.m_blsConnected)
			{
							m_listensocket.m_SeverSocket-
>Send("sbs", 3, 0);
			}
			m_listensocket.Close();
			m_blsListening = !m_blsListening;
			GetDlgItem(IDOK)->SetWindowText(_T("Create Server"));
			showMessage("Server disconnected.");
		}

}

void CDlgMasterSever::OnBnClickedBtnSendByLan()		// btn send by lan
{
		if (m_listensocket.m_blsConnected)
		{
				m_serversocket = m_listensocket.m_SeverSocket;
				m_serversocket->Send(MasterSTFromCom, 16, 0);
		}
}

void CDlgMasterSever::OnBnClickedBtnExit()
{
		if (m_blsListening)
		{
			m_listensocket.Close();
		}
		OnCancel();
}
```

```
LRESULT CDlgMasterSever::OnGetLanRXMsg(WPARAM wParam, LPARAM lParam)
{
                        switch(wParam)
                        {
                        case    SOCK_CLIENT_REQUEST :
                                                        {
                                                                        showMessage("Client
Request.");
                                                                        break;
                                                        }
                        case SOCK_SERVER_RECIEVED:
                                                        {
                                                                //      showMessage((const
char*)recvbuf);
                                                                        for (int i = 0; i<16;
i++)
                                                                        {

        *(SlaveSTFromLan+i) = *(recvbuf+i);

                                                                        }
                                                                        break;
                                                        }
                        }
                        return 0;
}

LRESULT CDlgMasterSever::OnGetLanTXMsg(WPARAM wParam, LPARAM lParam)
{

        if (m_listensocket.m_blsConnected)
        {
                        m_serversocket = m_listensocket.m_SeverSocket;
                        m_serversocket->Send(MasterSTFromCom, 16, 0);
                        return 0;
        }
                        return -1;
}

void CDlgMasterSever::OnBnClickedBtnStrcom()
{

                        CButton* pBTN = (CButton*)GetDlgItem(IDC_BTN_STRCOM);

                        if (!m_blsListening)
                        {
                                        MessageBox(_T("No client connected or No server is
build."), _T("Error"), MB_OK|MB_ICONEXCLAMATION);
                                        return;
```

```
                        }
                        if (!m_blsSerialPortOpen)
                        {
                                MessageBox(_T("No serial port is opened."),
_T("Error"), MB_OK|MB_ICONEXCLAMATION);
                                return;
                        }

                        //      start serial port and Lan communication
                        if (m_blsLanComm)
                        {
                                pBTN->SetWindowText(_T("Start Comm"));
                                //              stop serial port comm
                                m_blsSerialComm = false;
                                m_SerialPort.StopMonitoring();
                                ThreadSerialPort->SuspendThread();

                                //              stop LAN comm
                                ThreadLanTX->SuspendThread();
                                m_blsLanComm = false;
                        }else
                        {
                                pBTN->SetWindowText(_T("Stop Comm"));
                                //              start serial port comm
                                m_SerialPort.StartMonitoring();
                                m_blsSerialComm = true;
                                ThreadSerialPort =
AfxBeginThread(SerialPortTXDataThread,(LPVOID)m_hDlgHwnd);

                                //              start LAN comm
                                ThreadLanTX = AfxBeginThread( LanTXDataThread,
(LPVOID)m_hDlgHwnd);
                                m_blsLanComm = true;
                        }

        }

void CDlgMasterSever::OnBnClickedBtnOpencom()
{
                                CButton* pBTN =
(CButton*)GetDlgItem(IDC_BTN_OPENCOM);
                        //      pBTN->EnableWindow(false);
                                CComboBox* pCBX1 =
(CComboBox*)GetDlgItem(IDC_COMBO1);
                                CComboBox* pCBX2 =
(CComboBox*)GetDlgItem(IDC_COMBO2);
                                if (m_blsSerialPortOpen)
                                {
                                        pBTN->SetWindowText(_T("Open SerialPort"));
```

```
                                        m_SerialPort.ClosePort();
                                        m_blsSerialPortOpen = true;
                                //      MessageBox(_T("Serial Port has already
opened."), _T("ERROR"), MB_OK|MB_ICONERROR);
                                        return;
                                }

                                pBTN->SetWindowText(_T("Close SerialPort"));
                                char strTemp[32];
                                pCBX1->GetLBText(pCBX1->GetCurSel(),
(LPTSTR)strTemp);
                                m_nSerialPortNum = strTemp[3]-48;

                                switch(pCBX2->GetCurSel())
                                {
                                case 0:
                                        {     m_nSerialPortBuad = 4800; break;
        }
                                case 1:
                                        {     m_nSerialPortBuad = 9600; break;
        }
                                case 2:
                                        {     m_nSerialPortBuad = 19200; break;}
                                case 3:
                                       {      m_nSerialPortBuad = 38400; break;}
                                }
                                m_blsSerialPortOpen = true;
                                m_SerialPort.InitPort(this, m_nSerialPortNum,
m_nSerialPortBuad, 'N', 8, 1 );
}


BOOL CDlgMasterSever::OnInitDialog()
{
        CDialog::OnInitDialog();
        CComboBox* pCBX =(CComboBox*)GetDlgItem(IDC_COMBO2);
        pCBX->SetCurSel(2);
        findCommPort();

        m_hDlgHwnd = this->GetSafeHwnd();
        return TRUE;  // return TRUE unless you set the focus to a control
}

void CDlgMasterSever::findCommPort()
{
//      CStringArray strCommArr;
        HKEY hKey;
        int rtn;
        CComboBox* pCBX = (CComboBox*)GetDlgItem(IDC_COMBO1);
```

```
	rtn = RegOpenKeyEx( HKEY_LOCAL_MACHINE, _T("Hardware¥¥DeviceMap¥¥SerialComm"),
		NULL, KEY_READ, &hKey);
	if( rtn == ERROR_SUCCESS)
	{
				int  i=0;
				char  portName[256],commName[256];
				DWORD  dwLong,dwSize;
				while(1)
				{
					dwSize =  sizeof(portName);
					dwLong  =  dwSize;
					rtn = RegEnumValue( hKey, i, portName, &dwLong,
						NULL, NULL, (PUCHAR)commName, &dwSize );

					if( rtn == ERROR_NO_MORE_ITEMS )
					break;
					pCBX->AddString(_T(commName));
						//	strCommArr.Add(commName);
					i++;
				}
				RegCloseKey(hKey);
	}

	pCBX->SetCurSel(0);
}


LRESULT CDlgMasterSever::OnGetComRxMsg(WPARAM ch,LPARAM port)
{

	BYTE temp  = (BYTE)ch;

	if (temp == 0xAB)
	{
		g_MSTFromCom = 0;
		g_COM_Event.SetEvent();
		g_LAN_Event.SetEvent();
	}
	if (g_MSTFromCom > 16)
	{
		return -1;
	}
	MasterSTFromCom[g_MSTFromCom++] = temp;
	return 0;
}

LRESULT CDlgMasterSever::OnGetComTXMsg(WPARAM wParam, LPARAM lParam)
{
```

```
	m_SerialPort.WriteToPort(SlaveSTFromLan, 16);
	return 0;
}


#include "stdafx.h"
#include "MasterData.h"
#include "DlgDataReceive.h"
#include "MasterGlobalFunctions.h"
#include "Serialport.h"

#define DEBUG_WM 0
#define MSCOMM 0

IMPLEMENT_DYNAMIC(CDlgDataReceive, CDialog)

CDlgDataReceive::CDlgDataReceive(CWnd* pParent /*=NULL*/)
: CDialog(CDlgDataReceive::IDD, pParent)
,m_PortNum(5)
,m_IsPortCreat(0)
,m_IsPortOpen(0)
,m_InBufferSize(1024)
,m_OutBufferSize(1024)
{

}

CDlgDataReceive::~CDlgDataReceive()
{
	DataBaseQuit();
	gbSerialPortThreadBreak = FALSE;
}

void CDlgDataReceive::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
}


BEGIN_MESSAGE_MAP(CDlgDataReceive, CDialog)
	ON_BN_CLICKED(IDOK, &CDlgDataReceive::OnBnClickedOk)
	ON_BN_CLICKED(IDC_BUTTON_START, &CDlgDataReceive::OnBnClickedButtonStart)
	ON_BN_CLICKED(IDC_BUTTON_STOP, &CDlgDataReceive::OnBnClickedButtonStop)
	ON_BN_CLICKED(IDCANCEL, &CDlgDataReceive::OnBnClickedCancel)
	ON_WM_PAINT()
	ON_WM_TIMER()
	ON_WM_DESTROY()
	ON_WM_CLOSE()
	ON_WM_CREATE()
```

```
END_MESSAGE_MAP()


void CDlgDataReceive::OnBnClickedOk()
{
	//	int portNum;
	//	portNum = m_PortNum;
	//	SerialPortConnect(portNum);
}

void CDlgDataReceive::OnBnClickedButtonStart()
{
	UINT lpParam=0;
	AfxBeginThread(LPVOID lpParam)

	GetDlgItem(IDC_BUTTON_START)->EnableWindow(FALSE);
	GetDlgItem(IDC_BUTTON_STOP)->EnableWindow(TRUE);

	//-------------------------------------------------------------------------------------
-------------------------------------------------------------
//	Serial Port
	SERIAL_INFO SerialInfo;
	SerialInfo.PortNum = m_PortNum;
//	ThreadSerialPort = AfxBeginThread(SerialPortTXDataThread,(LPVOID)&SerialInfo);
	gbSerialPortThreadBreak = FALSE;

//-------------------------------------------------------------------------------------------------
-----------------------------------------------------
//	Draw
	glsDrawStart = 1;
	glsDrawCircle = 0;
	SetTimer(0,100,NULL);
//-------------------------------------------------------------------------------------------------
-----------------------------------------------------
//	Data Save
	nExcelFileCnt = 1;
	DataBaseInit();
	gbSaveDataThreadBreak = FALSE;
	THREAD_INFO ThreadInfo;
	ThreadSaveData = AfxBeginThread(SaveDataThread,  (LPVOID)&ThreadInfo);
}

void CDlgDataReceive::OnBnClickedButtonStop()
{
	gbSerialPortThreadBreak = TRUE;

	m_IsPortOpen = 0;
	GetDlgItem(IDC_BUTTON_START)->EnableWindow(TRUE);
```

```
	GetDlgItem(IDC_BUTTON_STOP)->EnableWindow(FALSE);
	KillTimer(0);
	gbSaveDataThreadBreak = TRUE;
}

void CDlgDataReceive::OnBnClickedCancel()
{
	OnCancel();
	//MessageBox(_T(""),_T(""),MB_OK);
}

void CDlgDataReceive::OnPaint()
{
	CPaintDC dc(this); // device context for painting

}

void CDlgDataReceive::OnTimer(UINT_PTR nIDEvent)
{
	switch(nIDEvent)
	{
	case 0:
		{
			DrawWave();
			break;
		}

	}

	CDialog::OnTimer(nIDEvent);
}


void CDlgDataReceive::DrawWave()
{
	CWnd *WaveWindowPICT;
	CRect WaveWindowRect;
	CDC *pDC;
	CBitmap memBitmap;
	CBitmap* pOldBmp = NULL;
	CPen *Oldpen;
	CFont  font,font2;
	CFont  *pOldFont=NULL;
	POINT EndPoint;
	double dCalibration;
	int nCntTmp;
	int nRow;

	//	InvalidateRect(WaveWindowRect);
```

```
        font.CreatePointFont(80,_T("Arial"));
        CPen GraphYellowPen(PS_SOLID, 3, RGB(255,255,0));
        CPen SolidGreenPen(PS_SOLID, 2, RGB(50, 150,50));
        CPen GraphPen(PS_SOLID, 1, RGB(0,0,0));

        WaveWindowPICT = GetDlgItem(IDC_STATIC);
        WaveWindowPICT->GetClientRect(WaveWindowRect);
        pDC = WaveWindowPICT->GetDC();
        WaveWindowPICT->Invalidate();

        // Create memory draw dc
        memDC.CreateCompatibleDC(pDC);
        memBitmap.CreateCompatibleBitmap(pDC, WaveWindowRect.right, WaveWindowRect.bottom);
        pOldBmp = memDC.SelectObject(&memBitmap);
        memDC.BitBlt(WaveWindowRect.left,WaveWindowRect.top,WaveWindowRect.right,WaveWindow
Rect.bottom, pDC, 0,0, SRCCOPY);

/*****************************************************************************************
**************/
        memDC.FillSolidRect(WaveWindowRect, (0,0,0));
        Oldpen = memDC.SelectObject(&SolidGreenPen);

        for (int i =1; i<40;i++)
        {
                memDC.MoveTo(WaveWindowRect.Width()*i/40, WaveWindowRect.top);
                memDC.LineTo(WaveWindowRect.Width()*i/40, WaveWindowRect.bottom);
        }
        for (int i =1; i<10; i++)
        {
                memDC.MoveTo(WaveWindowRect.left, WaveWindowRect.Height()*i/10);
                memDC.LineTo(WaveWindowRect.right, WaveWindowRect.Height()*i/10);
        }

        if (glsDrawStart )
        {
                dCalibration =WaveWindowRect.Width()/256.0;
                if (gTotalCnt > DATANUM/2)
                {
                        glsDrawCircle = 1;
                }

                if (!glsDrawCircle)
                {
                        nCntTmp = (gTotalCnt<256)? gTotalCnt : 256;
                }
                else
                {
                        nCntTmp = 256;
                }
```

```
                Oldpen = memDC.SelectObject(&GraphYellowPen);
                memDC.MoveTo(WaveWindowRect.right,WaveWindowRect.bottom);

                for (int i=0;i<nCntTmp;i++)
                {
                        nRow = (gTotalCnt-i<0)?(DATANUM-i+gTotalCnt): (gTotalCnt-i);
                         if (gTotalCnt-i<0)
                         {
                                EndPoint.x = 1;
                         }
                        EndPoint.x = WaveWindowRect.right - (i-1)*dCalibration;
                        EndPoint.y = WaveWindowRect.bottom -
(MasterStatus[nRow][LC]/4096.0)*(WaveWindowRect.Height());
                        memDC.LineTo(EndPoint);
                }
        }
        pDC->BitBlt(WaveWindowRect.left,WaveWindowRect.top,WaveWindowRect.right,
WaveWindowRect.bottom, &memDC, 0,0,SRCCOPY);
        memDC.SelectObject(pOldBmp);
        memDC.SelectObject(Oldpen);
        memDC.DeleteDC();
        memBitmap.DeleteObject();

        GraphYellowPen.DeleteObject();
        GraphPen.DeleteObject();
        SolidGreenPen.DeleteObject();
        font.DeleteObject();
        font2.DeleteObject();
        ReleaseDC(pDC);
        ReleaseDC(&memDC);
}

int CDlgDataReceive::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
        if (CDialog::OnCreate(lpCreateStruct) == -1)
                return -1;

        return 0;
}



//
// If there is a error, give the right message
//
void CSerialPort::ProcessErrorMessage(char* ErrorText)
{
        char *Temp = new char[200];
```

```
        LPVOID lpMsgBuf;

        FormatMessage(
                FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
                NULL,
                GetLastError(),
                MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), // Default language
                (LPTSTR) &lpMsgBuf,
                0,
                NULL
        );

        sprintf(Temp, "WARNING:  %s Failed with the following error: ¥n%s¥nPort: %d¥n",
(char*)ErrorText, lpMsgBuf, m_nPortNr);
        MessageBox(NULL, Temp, "Application Error", MB_ICONSTOP);

        LocalFree(lpMsgBuf);
        delete[] Temp;
}

//
// Write a character.
//
void CSerialPort::WriteChar(CSerialPort* port)
{
        BOOL bWrite = TRUE;
        BOOL bResult = TRUE;

        DWORD BytesSent = 0;

        ResetEvent(port->m_hWriteEvent);

        // Gain ownership of the critical section
        EnterCriticalSection(&port->m_csCommunicationSync);

        if (bWrite)
        {
                // Initailize variables
                port->m_ov.Offset = 0;
                port->m_ov.OffsetHigh = 0;

                // Clear buffer
                PurgeComm(port->m_hComm, PURGE_RXCLEAR | PURGE_TXCLEAR | PURGE_RXABORT |
PURGE_TXABORT);

                bResult = WriteFile(port->m_hComm,
        // Handle to COMM Port
                                                port->m_szWriteBuffer,
                // Pointer to message buffer in calling finction
```

```
                                                port->m_szWriteBufferSize,
/*                                              strlen((char*)port-
>m_szWriteBuffer),*/   // Length of message to send
                                                &BytesSent,
                                // Where to store the number of bytes sent
                                                &port->m_ov);
                        // Overlapped structure

                // deal with any error codes
                if (!bResult)
                {
                        DWORD dwError = GetLastError();
                        switch (dwError)
                        {
                                case ERROR_IO_PENDING:
                                        {
                                                // continue to GetOverlappedResults()
                                                BytesSent = 0;
                                                bWrite = FALSE;
                                                break;
                                        }
                                default:
                                        {
                                                // all other error codes
                                                port->ProcessErrorMessage("WriteFile()");
                                        }
                        }
                }
                else
                {
                        LeaveCriticalSection(&port->m_csCommunicationSync);
                }
        } // end if(bWrite)

        if (!bWrite)
        {
                bWrite = TRUE;

                bResult = GetOverlappedResult(port->m_hComm,  // Handle to COMM port
                                                                &port->m_ov,
        // Overlapped structure
                                                                &BytesSent,
        // Stores number of bytes sent
                                                                TRUE);
        // Wait flag

                LeaveCriticalSection(&port->m_csCommunicationSync);

                // deal with the error code
```

```
                if (!bResult)
                {
                        port->ProcessErrorMessage("GetOverlappedResults() in WriteFile()");
                }
        } // end if (!bWrite)

        // Verify that the data size send equals what we tried to send
//      if (BytesSent != strlen((char*)port->m_szWriteBuffer))
        if (BytesSent != port->m_szWriteBufferSize)
        {
                TRACE("WARNING: WriteFile() error.. Bytes Sent: %d; Message Length: %d¥n",
BytesSent, strlen((char*)port->m_szWriteBuffer));
        }
}

//
// Character received. Inform the owner
//
void CSerialPort::ReceiveChar(CSerialPort* port, COMSTAT comstat)
{
        BOOL  bRead = TRUE;
        BOOL  bResult = TRUE;
        DWORD dwError = 0;
        DWORD BytesRead = 0;
        unsigned char RXBuff;

        for (;;)
        {
                if(WaitForSingleObject(port->m_hShutdownEvent,0)==WAIT_OBJECT_0)
                        return;
                // Gain ownership of the comm port critical section.
                // This process guarantees no other part of this program
                // is using the port object.

                EnterCriticalSection(&port->m_csCommunicationSync);

                // ClearCommError() will update the COMSTAT structure and
                // clear any other errors.

                bResult = ClearCommError(port->m_hComm, &dwError, &comstat);

                LeaveCriticalSection(&port->m_csCommunicationSync);

                // start forever loop.  I use this type of loop because I
                // do not know at runtime how many loops this will have to
                // run. My solution is to start a forever loop and to
                // break out of it when I have processed all of the
                // data available.  Be careful with this approach and
                // be sure your loop will exit.
```

```
                // My reasons for this are not as clear in this sample
                // as it is in my production code, but I have found this
                // solutiion to be the most efficient way to do this.

                if (comstat.cbInQue == 0)
                {
                        // break out when all bytes have been read
                        break;
                }

                EnterCriticalSection(&port->m_csCommunicationSync);

                if (bRead)
                {
                        bResult = ReadFile(port->m_hComm,             // Handle to COMM port
                                                        &RXBuff,
        // RX Buffer Pointer
                                                        1,
        // Read one byte
                                                        &BytesRead,                 //
Stores number of bytes read
                                                        &port->m_ov);               //
pointer to the m_ov structure
                        // deal with the error code
                        if (!bResult)
                        {
                                switch (dwError = GetLastError())
                                {
                                        case ERROR_IO_PENDING:
                                                {
                                                        // asynchronous i/o is still in
progress
                                                        // Proceed on to
GetOverlappedResults();
                                                        bRead = FALSE;
                                                        break;
                                                }
                                        default:
                                                {
                                                        // Another error has occured.  Process
this error.
                                                        port-
>ProcessErrorMessage("ReadFile()");
                                                        break;
                                                }
                                }
                        }
                        else
                        {
```

```
                                // ReadFile() returned complete. It is not necessary to call
GetOverlappedResults()
                                bRead = TRUE;
                        }
                }  // close if (bRead)

                if (!bRead)
                {
                        bRead = TRUE;
                        bResult = GetOverlappedResult(port->m_hComm,  // Handle to COMM port
                                                                        &port->m_ov,
        // Overlapped structure
                                                                        &BytesRead,
        // Stores number of bytes read
                                                                        TRUE);
                        // Wait flag

                        // deal with the error code
                        if (!bResult)
                        {
                                port->ProcessErrorMessage("GetOverlappedResults() in
ReadFile()");
                        }
                }  // close if (!bRead)

                LeaveCriticalSection(&port->m_csCommunicationSync);

                // notify parent that a byte was received
                ::SendMessage((port->m_pOwner)->m_hWnd, WM_COMM_RXCHAR, (WPARAM) RXBuff,
(LPARAM) port->m_nPortNr);

        } // end forever loop

}

//
// Write a string to the port
//
void CSerialPort::WriteToPort(char* string)
{
        assert(m_hComm != 0);

        memset(m_szWriteBuffer, 0, sizeof(m_szWriteBuffer));
        strcpy(m_szWriteBuffer, string);

        // set event for write
        SetEvent(m_hWriteEvent);
}
```

```
void CSerialPort::WriteToPort(char* buf, UINT uSize)
{
        assert(m_hComm != 0);

          memcpy(m_szWriteBuffer, buf, uSize);
          m_szWriteBufferSize=uSize;
        // set event for write
        SetEvent(m_hWriteEvent);
}

//
// Return the device control block
//
DCB CSerialPort::GetDCB()
{
        return m_dcb;
}

//
// Return the communication event masks
//
DWORD CSerialPort::GetCommEvents()
{
        return m_dwCommEvents;
}

//
// Return the output buffer size
//
DWORD CSerialPort::GetWriteBufferSize()
{
        return m_nWriteBufferSize;
}


//------------------------------------------------------------------------------
void CSerialPort::ClosePort()
{
        // if the thread is alive: Kill
        if (m_bThreadAlive)
        {
                MSG message;
                while (m_bThreadAlive)
                {
                        if(::PeekMessage(&message,m_pOwner->m_hWnd,0,0,PM_REMOVE))
                        {
                                ::TranslateMessage(&message);
                                ::DispatchMessage(&message);
                        }
```

```
                        SetEvent(m_hShutdownEvent);
                }
                TRACE("Thread ended¥n");

        }
        if(m_szWriteBuffer != NULL)
        {
                delete [] m_szWriteBuffer;
                m_szWriteBuffer= NULL;
        }

        if(m_hComm)
        {
                CloseHandle(m_hComm);
                m_hComm = NULL;
        }
}


#include "stdafx.h"
#include "MasterGlobalVariables.h"
#include "MasterGlobalFunctions.h"
#include <afx.h>
#include <odbcinst.h>
#include "Serialport.h"
#include "DlgMasterSever.h"

void DataBaseInit(void);
void DataBaseQuit(void);

/*************************************************************************/
/*      save data thread
*/
/*************************************************************************/

UINT SaveDataThread(LPVOID lpParam)
{
        CString strSql;
        UINT Res = 0;
        int tmp;

        unsigned int nRound = 0;

        while (!gbSaveDataThreadBreak)
        {
                                        tmp = gTotalCnt;
                                        if (tmp == DATANUM)
                                        {
                                                                for(int i=0;i<tmp; i++)
```

```
                                                        {

        strSql.Format(_T("INSERT INTO
M(START,CRC,STPPRD,STPSPD,STPDEG,STPDIR,DCSPD,DCDEG,DCDIR,SWCLAMP,LC,TQ,SEN1,SEN2,SEN3,STOP
) VALUES(%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d)")¥

        ,(int)MasterStatus[i][0],(int)MasterStatus[i][1],(int)MasterStatus[i][2],(int)Maste
rStatus[i][3],(int)MasterStatus[i][4],(int)MasterStatus[i][5],(int)MasterStatus[i][6],(int)
MasterStatus[i][7],(int)MasterStatus[i][8]¥

        ,(int)MasterStatus[i][9],(int)MasterStatus[i][10],(int)MasterStatus[i][11],(int)Mas
terStatus[i][12],(int)MasterStatus[i][13],(int)MasterStatus[i][14],(int)MasterStatus[i][15]
);

        DataBase.ExecuteSQL(strSql);

        strSql.Format(_T("INSERT INTO
S(START,CRC,STPPRD,STPSPD,STPDEG,STPDIR,DCSPD,DCDEG,DCDIR,SWCLAMP,LC,TQ,SEN1,SEN2,SEN3,STOP
) VALUES(%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d)")¥

        ,(int)SlaveStatus[i][0],(int)SlaveStatus[i][1],(int)SlaveStatus[i][2],(int)SlaveSta
tus[i][3],(int)SlaveStatus[i][4],(int)SlaveStatus[i][5],(int)SlaveStatus[i][6],(int)SlaveSt
atus[i][7],(int)SlaveStatus[i][8]¥

        ,(int)SlaveStatus[i][9],(int)SlaveStatus[i][10],(int)SlaveStatus[i][11],(int)SlaveS
tatus[i][12],(int)SlaveStatus[i][13],(int)SlaveStatus[i][14],(int)SlaveStatus[i][15]);

        DataBase.ExecuteSQL(strSql);
                                                        }

        ////////////////////////////////////////////////////////////////
                                                //      gTotalCnt = 0;
                                                        AfxMessageBox(_T("Finished."),
MB_OK, NULL);
                                        }
        }

        DataBaseQuit();
        Res = 1;
        return Res;
}

void DataBaseInit(void)
{
        CString strSql;
        CTime CurTime = CTime::GetCurrentTime();
        CString strTime = CurTime.Format("%m-%d-%H-%M-%S");
        CString strTmp;
        strTmp.Format(_T(" F-%d"),nExcelFileCnt);
```

```
	CString strExcelFileName =_T( "C:¥¥Datafile¥¥") + strTime+strTmp + _T(".xls");
	CString strDriver("MICROSOFT EXCEL DRIVER (*.XLS)");

	TRY
	{

	strSql.Format(_T("DRIVER={%s};DSN='''';FIRSTROWHASNAMES=1;READONLY=FALSE;CREATE_DB=
¥"%s¥";DBQ=%s"),
			strDriver, strExcelFileName, strExcelFileName);

		if (DataBase.OpenEx(strSql,CDatabase::noOdbcDialog))
		{
			strSql= "CREATE TABLE M(START NUMBER,CRC NUMBER,STPPRD NUMBER,STPSPD
NUMBER,STPDEG NUMBER,STPDIR NUMBER,DCSPD NUMBER,DCDEG NUMBER,DCDIR NUMBER,SWCLAMP NUMBER,LC
NUMBER,TQ NUMBER,SEN1 NUMBER,SEN2 NUMBER,SEN3 NUMBER,STOP NUMBER)";
			DataBase.ExecuteSQL(strSql);
			strSql= "CREATE TABLE S(START NUMBER,CRC NUMBER,STPPRD NUMBER,STPSPD
NUMBER,STPDEG NUMBER,STPDIR NUMBER,DCSPD NUMBER,DCDEG NUMBER,DCDIR NUMBER,SWCLAMP NUMBER,LC
NUMBER,TQ NUMBER,SEN1 NUMBER,SEN2 NUMBER,SEN3 NUMBER,STOP NUMBER)";
			DataBase.ExecuteSQL(strSql);
		}

	}
	CATCH_ALL (e)
	{
			TRACE("Excel driver is not installed: %s", strDriver);
	}
	END_CATCH_ALL

	nExcelFileCnt++;
}

void DataBaseQuit(void)
{
	if (DataBase.IsOpen())
	{
		DataBase.Close();
	}
}



UINT SerialPortTXDataThread(LPVOID lParam)
{
	HWND hwnd = (HWND)lParam;

	while(1)
	{
		WaitForSingleObject(g_COM_Event, INFINITE);
```

```
                ::SendMessage(hwnd, WM_MSG_COM_TX, NULL, NULL);
                g_COM_Event.ResetEvent();
//              Sleep(100);
        }

        return 0;
}

UINT LanTXDataThread(LPVOID lParam)
{
        HWND hwnd = (HWND)lParam;

        while(1)
        {
                WaitForSingleObject(g_LAN_Event, INFINITE);          //      ?
                ::SendMessage(hwnd, WM_MSG_LAN_TX, NULL, NULL);
                g_LAN_Event.ResetEvent();
//              Sleep(100);
        }
        return 0;
}

//-------------------------------------------------------------------------------------------
--------------------------
//
//
        Global variables used in MasterData
//
        [2/22/2011 Administrator]
//
        MasterGlobalVariables.cpp
//-------------------------------------------------------------------------------------------
--------------------------
#include "stdafx.h"
#include <afxdb.h>
#include "MasterGlobalVariables.h"
#include "CMSComm.h"

LONG gGpCnt;
LONG gTotalCnt;

// BYTE ByteBuffer[128];
unsigned int MasterStatus[DATANUM][17];
unsigned int SlaveStatus[DATANUM][17];

BYTE MasterSTFromCom[16];
int g_MSTFromCom;
char SlaveSTFromLan[16];
int g_SSTFromCom;
```

```
//-----------------------------------------------------------------------------------------------
---------------------------
unsigned char glsDrawCircle;
unsigned char glsDrawStart;

//-----------------------------------------------------------------------------------------------
---------------------------
CDatabase DataBase;
unsigned int nExcelFileCnt;

CWinThread* ThreadSaveData;
// CWinThread* ThreadSerialPort;

BOOL gbSaveDataThreadBreak;
BOOL gbSerialPortThreadBreak;

unsigned char glsSaveFinished;

char recvbuf[SOCK_BUF_LEN];
char sendbuf[SOCK_BUF_LEN];


//-----------------------------------------------------------------------------------------------
---------------------------
//             serial port

CEvent g_COM_Event;
CEvent g_LAN_Event;


// MasterDataDoc.cpp : CMasterDataDoc ク¤¡¥ラ¤¨¦ス¤1のè?実g装Áiã
//

#include "stdafx.h"
#include "MasterData.h"

#include "MasterDataDoc.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif


// CMasterDataDoc

IMPLEMENT_DYNCREATE(CMasterDataDoc, CDocument)

BEGIN_MESSAGE_MAP(CMasterDataDoc, CDocument)
```

```
END_MESSAGE_MAP()



CMasterDataDoc::CMasterDataDoc()
{

}

CMasterDataDoc::~CMasterDataDoc()
{
}

BOOL CMasterDataDoc::OnNewDocument()
{
	if (!CDocument::OnNewDocument())
		return FALSE;

	return TRUE;
}
// CMasterDataDoc シ¤¡¤ ¤?Eヾé£¤꾀E?

void CMasterDataDoc::Serialize(CArchive& ar)
{
	if (ar.IsStoring())
	{

	}
	else
	{

	}
}


#ifdef _DEBUG
void CMasterDataDoc::AssertValid() const
{
	CDocument::AssertValid();
}

void CMasterDataDoc::Dump(CDumpContext& dc) const
{
	CDocument::Dump(dc);
}
#endif //_DEBUG


#include "stdafx.h"
```

```
#include "MasterData.h"
#include "MainFrm.h"

#include "MasterDataDoc.h"
#include "MasterDataView.h"
#include "CMSComm.h"
#include "MasterGlobalFunctions.h"
#include "DlgDataReceive.h"
#include "DlgConnectServer.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#endif


// CMasterDataApp

BEGIN_MESSAGE_MAP(CMasterDataApp, CWinApp)
	ON_COMMAND(ID_APP_ABOUT, &CMasterDataApp::OnAppAbout)
	ON_COMMAND(ID_FILE_NEW, &CWinApp::OnFileNew)
	ON_COMMAND(ID_FILE_OPEN, &CWinApp::OnFileOpen)
	ON_COMMAND(ID_FILE_PRINT_SETUP, &CWinApp::OnFilePrintSetup)
END_MESSAGE_MAP()


CMasterDataApp::CMasterDataApp()
{

}

CMasterDataApp theApp;


BOOL CMasterDataApp::InitInstance()
{
	INITCOMMONCONTROLSEX InitCtrls;
	InitCtrls.dwSize = sizeof(InitCtrls);
	InitCtrls.dwICC = ICC_WIN95_CLASSES;
	InitCommonControlsEx(&InitCtrls);

	CWinApp::InitInstance();

	if (!AfxOleInit())
	{
		AfxMessageBox(IDP_OLE_INIT_FAILED);
		return FALSE;
	}
	AfxEnableControlContainer();
	LoadStdProfileSettings(4);
```

```
	CSingleDocTemplate* pDocTemplate;
	pDocTemplate = new CSingleDocTemplate(
		IDR_MAINFRAME,
		RUNTIME_CLASS(CMasterDataDoc),
		RUNTIME_CLASS(CMainFrame),       // メ¤¨¢イ¤ié  ¤?ESDI フ¤?  ¤?E`  ¤?Eウ¤|
ィ¤iê ン¤¨®ド¤¨¦ウ¤|
		RUNTIME_CLASS(CMasterDataView));
	if (!pDocTemplate)
		return FALSE;
	AddDocTemplate(pDocTemplate);

	CCommandLineInfo cmdInfo;
	ParseCommandLine(cmdInfo);

	if (!ProcessShellCommand(cmdInfo))
		return FALSE;

	m_pMainWnd->ShowWindow(SW_SHOW);
	m_pMainWnd->UpdateWindow();
	return TRUE;
}


class CAboutDlg : public CDialog
{
public:
	CAboutDlg();

	enum { IDD = IDD_ABOUTBOX };

protected:
	virtual void DoDataExchange(CDataExchange* pDX);    protected:
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
END_MESSAGE_MAP()

void CMasterDataApp::OnAppAbout()
{
```

```
        CAboutDlg aboutDlg;
        aboutDlg.DoModal();
}

#include "stdafx.h"
#include "ListenSocket.h"
#include "DlgMasterSever.h"

CListenSocket::CListenSocket(): m_blsConnected(FALSE)
{

}

CListenSocket::~CListenSocket()
{

}

void CListenSocket::OnAccept(int nErrorCode)
{
                        CServerSocket *psk = new CServerSocket();
                        if (Accept(*psk))
                        {
                                        //      AfxMessageBox(_T("Connecting request"));
                                                m_SeverSocket = psk;
                                                m_SeverSocket->m_hwnd = m_hwnd;
                                                m_blsConnected = true;
                                                ::SendMessage(m_hwnd, WM_MSG_LAN_RX,
SOCK_CLIENT_REQUEST, NULL);
//                                              PostMessage(m_hwnd, WM_MYMSG,
SOCK_CLIENT_REQUEST, NULL);
                                                CSocket::OnAccept(nErrorCode);
                        }else
                        {
                                                delete  psk;
                        }
                        AsyncSelect( FD_READ|FD_WRITE|FD_CLOSE );
}
```

Appendix II

The program code of the robot-assisted catheter system (CCStudio v3.3)

The program code of the master system:

```
# include "DSP2833x_Device.h"

#define MASTER_SIDE 1

#define SLAVE_SIDE 0

//---------------------------------------------------------------------

volatile Uint16 Flag_1ms; // Define the main loop timer flag

volatile Uint16 GlobalDelayCount;   //        Global delay

//---------------------------------------------------------------------

//        //        variables from GPIO

//        Uint16 SteppingTim1;          //        GPIO76

Uint16 SteppingAlarm;         //        GPIO75

Uint16 SteppingEnd;  //        GPIO74

Uint16 ExSw1;         //        GPIO68

Uint16 ExSw10;        //        Last status of sw1

Uint16 ExSw2;         //        GPIO67

Uint16 ExSw20;        // Last status of Sw2

Uint16 ExSw3;         //        GPIO66

Uint16 ExSw4;         //        GPIO65

Uint16 ExSw5;         //        GPIO64
```

```
Uint16 LimitSw1;      //        GPIO46
Uint16 LimitSw2;      //        GPIO47
Uint16 LimitSw3;      //        GPIO80
//---------------------------------------------------------------------
//  sensor variables from adc
//
// float LoadCell;      //        ADCA3
// Uint16 LoadCellOffset;
/*
float Torque;   //        ADCA1
float Pressure0;        //        ADCA2
float Pressure1;        //        ADCA3
float Pressure2;        //        ADCA4
float VariableRegister0;        //        ADCA5
float VariableRegister1;        //        ADCA6
float Reserve;  //        ADCA7
*/
//---------------------------------------------------------------------
//        // output variables for GPIO
//
Uint16 SteppingACL;//        GPIO79
Uint16 Steppingx10;  //        GPIO78
```

```
Uint16 SteppingCOFF;        //      GPIO77
Uint16 MaxonDacCs; //       GPIO73
Uint16 MaxonDacSck;         //      GPIO72
Uint16 MaxonDacSdi;         //      GPIO71
Uint16 MaxonDacLdac;        //      GPIO70
Uint16 MaxonEN;     //      GPIO69
Uint16 LED1; //       GPIO40
Uint16 LED2; //       GPIO41
Uint16 LED3; //       GPIO42
Uint16 LED4; //       GPIO43
//---------------------------------------------------------------------
//       // output variables for EQEP
//POSSPEED qepSTP_posspeed = POSSPEED_DEFAULTS;
POSSPEED qepDC_posspeed = POSSPEED_DEFAULTS;
//---------------------------------------------------------------------
// Global DATA
// Spi communication data
//
Uint16 MasterStatus[DATALEN];
Uint16 SlaveStatus[DATALEN];
Uint16 SCI_CNT;
Uint16 SCI_COPY;
```

```
char SciCopyBuff[2];
Uint16 gTest;
volatile Uint16 BusyFlag;
//---------------------------------------------------------------------
// Control variable
//
float StpV0;
int DcDeg0;             //        Q15
int StpDeg0;   //        Q15
#include "DSP2833x_Device.h"
#define MASTER_SIDE 1
#define SLAVE_SIDE 0
//---------------------------------------------------------------------
//
//                                        *       Device Initial  *
//                                                (Both M & S)
// //--------------------------------------------------------------------
////                      Stepping initial
//void InitStepping(void)
{
        Uint16 i;
        GpioDataRegs.GPCSET.bit.GPIO79= 1;      //        Clear Alarm    Gpio79= 1;
```

```
        //DELAY_US(1L);
        for(i=0;i<500;i++){}
        GpioDataRegs.GPCCLEAR.bit.GPIO79= 1; // Set alarm to low; Gpio79= 0
                // GpioDataRegs.GPCCLEAR.bit.GPIO78= 1;        // Set x10; 1 - x10; 0 -
none;   Gpio78= 0;
        // GpioDataRegs.GPCCLEAR.bit.GPIO77= 1;        // Stepping C.Off; 1 - off; 0 -
on;     Gpio77= 0;
}
//---------------------------------------------
//                      DAC Initial
void InitDAC(void)
{
        //      Other pins was set in Mcbspb initial
        GpioDataRegs.GPCCLEAR.bit.GPIO70= 1; //      Set #LDAC to low, P2-12
}
/---------------------------------------------
//              Global variables initial
void InitGlobalVars(void)
{
        Uint16 i;

// Status initial
```

```
        for(i=0;i<DATALEN;i++)
        {
                MasterStatus[i] = 0x0000;
                SlaveStatus[i] = 0x0000;
        }
//      MasterStatus[STR] = 0x0808;
        MasterStatus[STR] = 0xABCD;
        MasterStatus[CRC]= 0x0888;
//      MasterStatus[STOP]= 0x0C0C;
        MasterStatus[STOP] = 0xDCBA;
//      Ex switch for clamp
        ExSw1 = 0;
        ExSw10 = 0;
        ExSw2 = 0;
        ExSw20 = 0;
// Qep posspeed var initial
        qepSTP_posspeed.DevSel = STEPPING;
        qepDC_posspeed.DevSel = DC;

        qepSTP_posspeed.mech_scaler= 16776;          //      Q26
        qepDC_posspeed.mech_scaler= 512;         //      Q0
        qepSTP_posspeed.theta_raw= 0;
```

```
        qepDC_posspeed.theta_raw= 0;
        qepSTP_posspeed.init(&qepSTP_posspeed);
        qepDC_posspeed.init(&qepDC_posspeed);
// Global Delay Count
        GlobalDelayCount = 0;
        SCI_CNT = 8;
        SCI_COPY = 0;
        gTest = 0;
        StpV0= 0.0;
        StpDeg0 = 0;  // Q15
        DcDeg0= 0;              // Q15
}
//---------------------------------------------------------------------
////      ******************          Functions used in Master side
******************
// //--------------------------------------------------------------------
#if MASTER_SIDE
//---------------------------------
//                        Step 1
void GetUserStatus_M(void)
{
        GetADC();     //        User's input was got
```

```
        GetGpio();      //      Control panel, Limit switch
        GetEQep();      //      Stepping displacement and speed, DC rolling angle and
speed

//      DataSendToSlave[STPSPD] = qepSTP_posspeed.Speed_fr
}
//-----------------------------------
//                      Step 2
void SendCMDtoSlave_M(void)
{
/*-----------------------------------------------------
        Uint16 i;
        Uint16 tmp;
        if(GpioDataRegs.GPBDAT.bit.GPIO58 == 0)
        {
                for(i=0;i<DATALEN;i++)
                {
                        while(SpiaRegs.SPISTS.bit.BUFFULL_FLAG == 1){}
                        SpiaRegs.SPITXBUF= MasterStatus[i];
                        while(SpiaRegs.SPISTS.bit.INT_FLAG == 0) {}
                        tmp = SpiaRegs.SPIRXBUF;
                        if(i != 0)
```

```
                        {
                                SlaveStatus[i-1] = tmp;
                        }
                }

        }
----------------------------------------------------------*/
}
//-----------------------------------
//              Step 3
void SetMasterStatus_M(void)
{

//      float tmp;
//---------------------------
//      Set Stepping
        if(GlobalDelayCount % 10 == 0)
        {
                        SetMasterStepping();
                }
//---------------------------
//      Set Maxon through DAC
```

```
//      tmp = 3.0*SlaveStatus[LC]/4096.0;
        SetDAC(1.65,0);     //      0-A, 1-B
        // just for test
}
#endif //      END of MASTER_SIDE
//---------------------------------------------------------------------------
//
//                              *       Sub functions  *
//                                      (Both M & S)
//---------------------------------------------------------------------------
//
//      Get user information from ADC
//
void GetADC(void)
 {

        AdcRegs.ADCTRL2.bit.RST_SEQ1= 1;            // Reset SEQ1
        // AdcRegs.ADCTRL2.bit.RST_SEQ2= 1;         // Reset SEQ2
        //      Start SEQ1
  AdcRegs.ADCTRL2.all = 0x2000;
        while (AdcRegs.ADCST.bit.INT_SEQ1== 0) {} // Wait for interrupt
        AdcRegs.ADCST.bit.INT_SEQ1_CLR= 1;  // Clear SEQ1 IF
```

```
        // AdcRegs.ADCST.bit.INT_SEQ2_CLR= 1;

        // Get sensors status from ADC
                MasterStatus[LC]= (AdcRegs.ADCRESULT3>>4);
        //      (float)(3.0*((AdcRegs.ADCRESULT3>>4)/4095.0)) - LoadCellOffset ;
        // ADC0 Get Load Cell;
/*              Torque= (AdcRegs.ADCRESULT1>>4);    // ADC1 Torque
                Pressure0= (AdcRegs.ADCRESULT2>>4);         // ADC2 Pressure0
                Pressure1= (AdcRegs.ADCRESULT3>>4);         // ADC3 Pressure1
                Pressure2= (AdcRegs.ADCRESULT4>>4);         // ADC4 Pressure2
                VariableRegister0= (AdcRegs.ADCRESULT5>>4);         // ADC5 VR0
                VariableRegister1= (AdcRegs.ADCRESULT6>>4);         // ADC6 VR0
*/

 }

//-----------------------------------------------------
//      Get user information from GPIO
void GetGpio(void)
{
        SteppingTim1= GpioDataRegs.GPCDAT.bit.GPIO76;
        SteppingAlarm= GpioDataRegs.GPCDAT.bit.GPIO75;
```

```
        SteppingEnd= GpioDataRegs.GPCDAT.bit.GPIO74;

//      ExSw1= GpioDataRegs.GPCDAT.bit.GPIO68;
//      ExSw2= GpioDataRegs.GPCDAT.bit.GPIO67;
        ExSw3= GpioDataRegs.GPCDAT.bit.GPIO66;
        ExSw4= GpioDataRegs.GPCDAT.bit.GPIO65;
        ExSw5= GpioDataRegs.GPCDAT.bit.GPIO64;

        LimitSw1= GpioDataRegs.GPBDAT.bit.GPIO46;
        LimitSw2= GpioDataRegs.GPBDAT.bit.GPIO47;
        LimitSw3= GpioDataRegs.GPCDAT.bit.GPIO80;

// --------        Get Clamp Sw State ----------
        if( GlobalDelayCount % 100 == 0)
        {

                if((GpioDataRegs.GPCDAT.bit.GPIO68 == 0) && (ExSw10 == 0))
                {
                  ExSw1 = (~ExSw1) & (0x0001);          // change last bit
                  ExSw10 = 1;
                }
                if(GpioDataRegs.GPCDAT.bit.GPIO68 == 1)
```

```
                {
                        ExSw10 = 0;
                }
                if((GpioDataRegs.GPCDAT.bit.GPIO67 == 0) && (ExSw20 == 0))
                {
                  ExSw2 = (~ExSw2) & (0x0001);              // change last bit
                  ExSw20 = 1;
                }
                if(GpioDataRegs.GPCDAT.bit.GPIO67 == 1)
                {
                  ExSw20 = 0;
                }
        }
                MasterStatus[SWCLAMP] = ExSw1;
//----------------------------------------
        /*----------------------*/
//      BusyFlag = GpioDataRegs.GPCDAT.bit.GPIO83;          // SPI BUSY
//      BusyFlag = 0; //        just for test, comment it when
finished////////////////////////////////////
}
//-------------------------------------------------------------
// Get EQep information position & speed of stepping & DC
```

```
void GetEQep(void)
{
        int tmp;
        qepSTP_posspeed.calc(&qepSTP_posspeed);
        qepDC_posspeed.calc(&qepDC_posspeed);
        //////////
//      EQep1Regs.QEPCTL.bit.SWI = 1;
        MasterStatus[STPDEG] = qepSTP_posspeed.theta_raw;
        MasterStatus[STPSPD] = qepSTP_posspeed.Speed_pr;                    // _iq,
Q0
        MasterStatus[STPDIR] = qepSTP_posspeed.DirectionQep;
        //////////
        tmp = DcDeg0 + qepDC_posspeed.theta_raw;
        DcDeg0 = tmp;
        EQep2Regs.QEPCTL.bit.SWI = 1;
        MasterStatus[DCDEG] = tmp;
        MasterStatus[DCSPD] = qepDC_posspeed.Speed_pr;      //_q, Q0
        MasterStatus[DCDIR]= qepDC_posspeed.DirectionQep;
        /*
        STPDisp= qepSTP_posspeed.theta_mech * STPPITCH;     // Q15 * Q0 = Q15,
displacement= angle*pitch
        STPSpeed= qepSTP_posspeed.Speed_pr;                 // _iq, Q0
```

```
        STPDir= qepSTP_posspeed.DirectionQep;
        DCDisp= qepDC_posspeed.theta_mech;          // Q15
        DCSpeed= qepDC_posspeed.Speed_pr;           // _iq, Q0
        DCDir= qepDC_posspeed.DirectionQep;
*/
}
//---------------------------------------------------------
//      SetDAC voltage
//      DAC is MCP4922, 16 bits command word, bits11:0 are data bits
//      bit15: 0: DAC_B, 1: DAC_A
//      bit14: 0: Unbuffered Vref, 1: buffered Vref
//      bit13: 0: (Vout=2*Vref*D/4096), 1: (Vout=1*Vref*d/4096)
//      bit12: #SHDN, 0: output buffer disable, high impedance, 1: Output power donw
control bit
//---------------------------------------------------------
void SetDAC(float voltage, int Sel)
{
        Uint16 Tmp;
        if(Sel == 0)            //      DAC-A       bit15 = 0
        {
                Tmp= (Uint16)(4096*voltage/3.35) & 0x0FFF | 0x3000;
        }else if(Sel == 1)
```

```
        {
                Tmp= (Uint16)(4096*voltage/3.35) & 0x0FFF | 0xA000;
        }
        McbspbRegs.DXR1.all= Tmp;
}
//---------------------------------------------------------
//      Set Master Stepping
//      Direction, Frequency
//
//---------------------------------------------------------
void SetMasterStepping(void)
{
        float dF,v, Fl, C0, tmp;
        float Fs;
        Uint16 tbprd;
        C0 = 0.001;
        //      Just for test comment when release
        if(SlaveStatus[STR] == 0xABCD)
        {
                Fs = (SlaveStatus[LC]*3.0/4096.0)-SLC_VOLTAGE_OFS;
        }
        Fl= (((int)MasterStatus[LC])*3.0/4096) - MLC_VOLTAGE_OFS;
```

```
//      Fl= (((int)MasterStatus[LC])*3.0/4096);
        if((Fl<0.1) && (Fl>(-0.1)))
        {
                dF = 0;
        }else{
//              dF= Fl + 0.5* Fs;
                dF = Fl;
                dF*= 5/LC_VOLTAGE_RANGE;
        }
        tmp = 1000* MasterStatus[STPSPD]/1500000.0;
        C0 = C0 + tmp;
        v= 5*MASTER_M * C0*dF*1.55 + 0.5*StpV0;
        StpV0= v;
        if((Fl>0.1)||(Fl<(-0.1)))
        {
                if(dF>0)
                {

                        if(Fl>0)
                        {
                                tbprd = (int)(1500/v - 1);
                                if(tbprd<1000)// MAX frq is 15K
```

```
                    {
                        tbprd = 1000;
                    }
                    EPwm1Regs.TBPRD= tbprd;
                    EPwm1Regs.CMPA.half.CMPA= (tbprd)/2; // Duty cycle=
50%
                    EPwm1Regs.CMPB = (tbprd)/2;       // Duty cycle= 50%

                    STOPEPWM2();
                    STARTEPWM1();
                    MasterStatus[STPPRD] = tbprd;
                    MasterStatus[STPDIR] =0;
                //  MasterStatus[STPDIR] = 1;
                }else if(Fl<0)
                {
                    MasterStatus[STPPRD] = 0;
                    STOPEPWM1();
                    STOPEPWM2();
                }
            }else if(dF<0)
            {
                if(Fl<0)
```

```
                {
                        tbprd = (-1)*((int)1500/v-1);
                        if(tbprd<1000)
                        {
                                tbprd = 1000;
                        }
                        EPwm2Regs.TBPRD= tbprd;
                        EPwm2Regs.CMPA.half.CMPA= (tbprd)/2; // Duty cycle=
50%
                        EPwm2Regs.CMPB = (tbprd)/2;       // Duty cycle= 50%
                        STOPEPWM1();
                        STARTEPWM2();
                        MasterStatus[STPPRD] = tbprd;
                        MasterStatus[STPDIR] = 1;
                }else if(Fl>0)
                {
                        MasterStatus[STPPRD] = 0;
                        STOPEPWM1();
                        STOPEPWM2();
                }
            }
        }else
```

```
    {
        MasterStatus[STPPRD] = 0;
        STOPEPWM1();
        STOPEPWM2();
    }
```

The program code of the slave system:

```
#include "RCS_Project.h"
#pragma CODE_SECTION(cpu_timer0_isr, "ramfuncs");
#pragma CODE_SECTION(spiaRxIsr, "ramfuncs");
#pragma CODE_SECTION(wakeint_isr, "ramfuncs");
//#pragma CODE_SECTION(wakeint_ist, "ramfuncs");
//---------------------------------------------------------
// declear interrupt functions here
// interrupt void cpu_timer0_isr(void);
interrupt void spiaRxIsr(void);
interrupt void wakeint_isr(void);
//interrupt void wakeint_isr(void);
//---------------------------------------------------------
// #define FLASH 0
#define MASTER_SIDE 0
#define SLAVE_SIDE 1
```

```
void main(void)
{
// Step 1 initial system control
// PLL, WatchDog, peripheral clk
        InitSysCtrl();
// Step 2 initial gpio
        InitGpio();
#if FLASH
/* Copy the PIEVecTable section */
        MemCopy(&RamfuncsLoadStart, &RamfuncsLoadEnd, &RamfuncsRunStart);
/* Initialize the on-chip flash registers */
        InitFlash();
#endif
// Step 3 Clear all interrupts, initialize PIE vector table
// Disable cpu interrupts
        DINT;
        // Initialize PIE control register to their efault state.
// The default state is all PIE interrupts disabled and flags
// are cleared.
// This function is found in the DSP280x_PieCtrl.c file.
        InitPieCtrl();
// Disable CPU interrupts and clear all CPU interrupt flags:
```

```
        IER = 0x0000;
        IFR = 0x0000;
// Initialize the PIE vector table with pointers to the shell Interrupt
// Service Routines (ISR).
// This will populate the entire table, even if the interrupt
// is not used in this example.  This is useful for debug purposes.
// The shell ISR routines are found in DSP280x_DefaultIsr.c.
// This function is found in DSP280x_PieVect.c.
   InitPieVectTable();
// Reload isr here
   EALLOW;
   PieVectTable.TINT0= &cpu_timer0_isr;
   PieVectTable.SPIRXINTA = &spiaRxIsr;
   PieVectTable.WAKEINT = &wakeint_isr;
   EDIS;
// Step 4. Initialize all the Device Peripherals:
// This function is found in DSP280x_InitPeripherals.c
// Initialize epwm
        InitEPwm();
        InitAdc();
        InitEQep();
        Init_mcbsp_spi();
```

```
        InitSpi();
//      InitSci();
// Initialize devices;
        InitStepping();
        InitDAC();
        InitGlobalVars();
// CPU timer0 is used for main loop
// CPU timer0 initialize here
        InitCpuTimers();
// Config cpu timer
// cputimers.c was modified,timer1 and timer2 can not be initialized
// Timer0 is used CPU CLK - 150MHz, main loop period= 1ms
                ConfigCpuTimer(&CpuTimer0, 150, 1000);
        StartCpuTimer0();
// Step 5. Enable Interrupts:
// Connect the watchdog to the WAKEINT interrupt of the PIE
// Write to the whole SCSR register to avoid clearing WDOVERRIDE bit
  EALLOW;
  SysCtrlRegs.SCSR = BIT1;
  EDIS;
// Enable CPU timer0
        IER |= M_INT1;
```

```
        IER |= M_INT6;
// Enable TINT0 in PIE
        PieCtrlRegs.PIEIER1.bit.INTx7= 1;
    PieCtrlRegs.PIEIER1.bit.INTx8 = 1;   // Enable PIE Gropu 1 INT8
        PieCtrlRegs.PIEIER6.bit.INTx1=1;     // Enable PIE Group 6, INT 1 rx
// Enable other PIE INTx here
//// Enable global Interrypts and higher priority real-time debug events:
        EINT;   // Enable Global interrupt INTM
        ERTM;   // Enable Global realtime interrupt DBGM
// Reset the watchdog counter
   ServiceDog();
      / Enable the watchdog
   EALLOW;
   SysCtrlRegs.WDCR = 0x0028;
   EDIS;
// ---------------------------------------------------------------------
//
// Initial Finished
//
// ---------------------------------------------------------------------
////---------------------------------------------------------------------
//
```

```
// Program for slave side
//
//---------------------------------------------------------------------
  for(;;)
        {
                if(Flag_1ms == 1)
                {
                        GetUserStatus_S();
                        SendCMDtoMaster_S();
                        SetSlaveStatus_S();
                                GlobalDelayCount++;
                        Flag_1ms = 0;
        }
                if(IsReset == 0)
                {
                  ServiceDog();
                }
        }
}
// ---------------------------------------------------------------------

interrupt void cpu_timer0_isr(void)
```

```
{
        CpuTimer0.InterruptCount++;
        Flag_1ms= 1;

/*      GetUserStatus_S();
        SendCMDtoMaster_S();
        SetSlaveStatus_S();

        GlobalDelayCount++;
        Flag_1ms = 0;
*/
        EALLOW;
        CpuTimer0Regs.TCR.bit.TRB= 0x1;// reload timer count
        CpuTimer0Regs.TCR.bit.TIF= 1;      // Clear int flag
// Acknowledge this interrupt to receive more interrupts from group 1
   PieCtrlRegs.PIEACK.all|= PIEACK_GROUP1;
        EDIS;
        }
interrupt void spiaRxIsr(void)
{
        Uint16 temp;
        temp = SpiaRegs.SPIRXBUF;
```

```
        if(temp == 0xABCD)
        {
            gPointer = 0;
        }
        MasterStatus[gPointer] = temp;
        if(gPointer<16)
        {
            gPointer++;
            SpiaRegs.SPITXBUF = SlaveStatus[gPointer];
        }else
        {
            GpioDataRegs.GPBSET.bit.GPIO58= 1;      //      Set busy to 1
        }
        PieCtrlRegs.PIEACK.all|=0x20;       // Issue PIE ack
}

interrupt void wakeint_isr(void)
{
        SysCtrlRegs.SCSR |= 0x2;
//      IsReset = 0;
        // Acknowledge this interrupt to get more from group 1
        PieCtrlRegs.PIEACK.all = PIEACK_GROUP1;
```

```
}
#include "DSP2833x_Device.h"

#define MASTER_SIDE 0
#define SLAVE_SIDE 1
//---------------------------------------------------------------------
//
//                                    *       Device Initial  *
//                                            (Both M & S)
// ---------------------------------------------------------------------
////                        Stepping initial
//
void InitStepping(void)
{
        Uint16 i;
        GpioDataRegs.GPCSET.bit.GPIO79= 1;      //      Clear Alarm    Gpio79= 1;
        //DELAY_US(1L);
        for(i=0;i<500;i++){}
        GpioDataRegs.GPCCLEAR.bit.GPIO79= 1; // Set alarm to low; Gpio79= 0

        // GpioDataRegs.GPCCLEAR.bit.GPIO78= 1;        // Set x10; 1 - x10; 0 - none;
        Gpio78= 0;
```

```
        // GpioDataRegs.GPCCLEAR.bit.GPIO77= 1;        // Stepping C.Off; 1 - off; 0 -
on;     Gpio77= 0;


}
//---------------------------------------------
//                      DAC Initial
void InitDAC(void)
{
        //      Other pins was set in Mcbspb initial
        GpioDataRegs.GPCCLEAR.bit.GPIO70= 1; //      Set #LDAC to low, P2-12
}
//---------------------------------------------
//              Global variables initial
void InitGlobalVars(void)
{
        Uint16 i;


// Status initial
        for(i=0;i<DATALEN;i++)
        {
                MasterStatus[i] = 0x0000;
                SlaveStatus[i] = 0x0000;
```

```
        }
        SlaveStatus[START] = 0x0909;
        SlaveStatus[CRC] = 0x0999;
        SlaveStatus[STOP] = 0x1111;
        gPointer = 0;
// Qep posspeed var initial
        qepSTP_posspeed.DevSel = STEPPING;
        qepDC_posspeed.DevSel = DC;
        qepSTP_posspeed.mech_scaler= 16776;          //      Q26
        qepDC_posspeed.mech_scaler= 256;          //      Q15
        qepSTP_posspeed.theta_raw= 0;
        qepDC_posspeed.theta_raw= 0;
        qepSTP_posspeed.init(&qepSTP_posspeed);
        qepDC_posspeed.init(&qepDC_posspeed);
// Global Delay Count
        GlobalDelayCount = 0;
        StpV0= 0.0;
        StpDeg0 = 0;   // Q15
        DcDeg0= 0;          // Q15
        IsReset = 0;
}
//---------------------------------------------------------------------------
```

```
//
// ******************      Functions used in Slave side  ******************
// //---------------------------------------------------------------------------
#if SLAVE_SIDE
//                          Step 1
void GetUserStatus_S(void)
{
                GetADC();     //        User's input was got
        GetGpio();     //        Control panel, Limit switch
        GetEQep();     //        Stepping displacement and speed, DC rolling angle and
speed
//      DataSendToSlave[STPSPD] = qepSTP_posspeed.Speed_fr
}
//-----------------------------------
//                          Step 2
void SendCMDtoMaster_S(void)
{
        SpiaRegs.SPITXBUF = SlaveStatus[START];
        GpioDataRegs.GPBCLEAR.bit.GPIO58= 1; // Set busy to 0,then start trams
}
//-----------------------------------
//                Step 3
```

```
void SetSlaveStatus_S(void)
{
	float volt;
	float bal;
	int dcerr;
//---------------------------
// Set Clamp Motor
	if(MasterStatus[CRC] == 0x0888)
	{
		if(MasterStatus[SWCLAMP] == 0)
		{
			GpioDataRegs.GPBSET.bit.GPIO44 = 1;
			GpioDataRegs.GPBCLEAR.bit.GPIO45 = 1;
		}else if(MasterStatus[SWCLAMP] == 1)
		{
			GpioDataRegs.GPBCLEAR.bit.GPIO44 = 1;
			GpioDataRegs.GPBSET.bit.GPIO45 = 1;
		}
	}
//---------------------------
//	Set Stepping
	if(GlobalDelayCount > 9)
```

```
        {
                SetSlaveStepping();
                GlobalDelayCount = 0;
        }
//--------------------------
//      Set Maxon through DAC
//      int rdata1= 0x0000;
        if(MasterStatus[CRC] == 0x0888)
        {
                bal = 0.826;
                dcerr = MasterStatus[DCDEG]- SlaveStatus[DCDEG];
                        volt = 2.0*bal*(1 + (dcerr)/100.0);
                DCENABLE();
                if(((volt - bal) < 0.5) && ((volt -bal) > -0.5))
                {
                        DCDISABLE();
                        volt = bal;
                }
                if(volt > 3.3)
                {
                        volt = 3.3;
                }
```

```
                if(volt < -3.3)
                {
                        volt = 0;
                }
                SetDAC(volt,1);         //          0-A, 1-B
        // just for test
//      while(McbspbRegs.SPCR1.bit.RRDY == 0) {;}
//      rdata1= McbspbRegs.DRR1.all;
//      asm("   nop");
        }
//      SetDAC(0,1);
}
#endif //       END of SLAVE_SIDE
//*************************************************************************
****
//-------------------------------------------------------------------------
//
//                              *       Sub functions  *
//                                      (Both M & S)
//-------------------------------------------------------------------------
//
//      Get user information from ADC
```

```
//
void GetADC(void)
 {
//        float tmp;
          AdcRegs.ADCTRL2.bit.RST_SEQ1= 1;              // Reset SEQ1
    AdcRegs.ADCTRL2.all = 0x2000;
          while (AdcRegs.ADCST.bit.INT_SEQ1== 0) {} // Wait for interrupt
          AdcRegs.ADCST.bit.INT_SEQ1_CLR= 1;  // Clear SEQ1 IF
          // AdcRegs.ADCST.bit.INT_SEQ2_CLR= 1;
                    // Get sensors status from ADC
          SlaveStatus[LC]= (AdcRegs.ADCRESULT0>>4);
          SlaveStatus[TQ]= (AdcRegs.ADCRESULT3>>4);
//        tmp = 3.0 * (SlaveStatus[LC]/4096.0);
 }
//-------------------------------------------------------
//        Get user information from GPIO
void GetGpio(void)
{
          SteppingTim1= GpioDataRegs.GPCDAT.bit.GPIO76;
          SteppingAlarm= GpioDataRegs.GPCDAT.bit.GPIO75;
          SteppingEnd= GpioDataRegs.GPCDAT.bit.GPIO74;
          ExSw1= GpioDataRegs.GPCDAT.bit.GPIO68;
```

```
        ExSw2= GpioDataRegs.GPCDAT.bit.GPIO67;
        ExSw3= GpioDataRegs.GPCDAT.bit.GPIO66;
        ExSw4= GpioDataRegs.GPCDAT.bit.GPIO65;
        ExSw5= GpioDataRegs.GPCDAT.bit.GPIO64;
        LimitSw1= GpioDataRegs.GPBDAT.bit.GPIO46;
        LimitSw2= GpioDataRegs.GPBDAT.bit.GPIO47;
        LimitSw3= GpioDataRegs.GPCDAT.bit.GPIO80;
        /*----------------------*/
}
//---------------------------------------------------------------
// Get EQep information position & speed of stepping & DC
void GetEQep(void)
{
        int tmp;

        qepSTP_posspeed.calc(&qepSTP_posspeed);
        qepDC_posspeed.calc(&qepDC_posspeed);
        //////////

//      EQep1Regs.QEPCTL.bit.SWI = 1;
        SlaveStatus[STPDEG] = qepSTP_posspeed.theta_raw;
        SlaveStatus[STPSPD] = qepSTP_posspeed.Speed_pr;                 // _iq,
```

```
Q0
        SlaveStatus[STPDIR] = qepSTP_posspeed.DirectionQep;
        //////////
        tmp = DcDeg0 + qepDC_posspeed.theta_raw/19;
        DcDeg0 = tmp;
        EQep2Regs.QEPCTL.bit.SWI = 1;
//      SlaveStatus[DCDEG] = qepDC_posspeed.theta_raw;
        SlaveStatus[DCDEG] = tmp;
        SlaveStatus[DCSPD] = qepDC_posspeed.Speed_pr;//_q, Q0
        SlaveStatus[DCDIR]  = qepDC_posspeed.DirectionQep;
/*
        STPDisp= qepSTP_posspeed.theta_mech * STPPITCH;     // Q15 * Q0 = Q15,
displacement= angle*pitch
        STPSpeed= qepSTP_posspeed.Speed_pr;                 // _iq, Q0
        STPDir= qepSTP_posspeed.DirectionQep;

        DCDisp= qepDC_posspeed.theta_mech;          // Q15
        DCSpeed= qepDC_posspeed.Speed_pr;           // _iq, Q0
        DCDir= qepDC_posspeed.DirectionQep;
*/
}
//---------------------------------------------------------
```

```
//      SetDAC voltage
//      DAC is MCP4922, 16 bits command word, bits11:0 are data bits
//      bit15: 0: DAC_A, 1: DAC_B
//      bit14: 0: Unbuffered Vref, 1: buffered Vref
//      bit13: 0: (Vout=2*Vref*D/4096), 1: (Vout=1*Vref*d/4096)
//      bit12: #SHDN, 0: output buffer disable, high impedance, 1: Output power donw
control bit
//-------------------------------------------------------------
void SetDAC(float voltage, int Sel)
{
        Uint16 Tmp;
        if(Sel == 0)            //      DAC-A       bit15 = 0
        {
                Tmp= ((Uint16)(4096*(voltage)/3.32)) & 0x0FFF | 0x3000;
        }else if(Sel == 1)
        {
                Tmp= ((Uint16)(4096*(voltage)/3.32)) & 0x0FFF | 0xA000;
        }
        McbspbRegs.DXR1.all= Tmp;
}

//-------------------------------------------------------------
```

```
//      Set Slave Stepping
//      Direction, Frequency
//
//-------------------------------------------------------
void SetSlaveStepping(void)
{
    if((MasterStatus[STPPRD] != 0) && (MasterStatus[CRC]== 0x0888) &&
(MasterStatus[STPDEG] != 0))
    {
        if(MasterStatus[STPDIR] == 0)       //      +
        {
            EPwm1Regs.TBPRD= MasterStatus[STPPRD];
            EPwm1Regs.CMPA.half.CMPA= MasterStatus[STPPRD]/2;
            EPwm1Regs.CMPB= MasterStatus[STPPRD]/2;
            STOPEPWM2();
            STARTEPWM1();
        }else if(MasterStatus[STPDIR] == 1)             //-
        {
                    EPwm2Regs.TBPRD= MasterStatus[STPPRD];
            EPwm2Regs.CMPA.half.CMPA= MasterStatus[STPPRD]/2;
            EPwm2Regs.CMPB= MasterStatus[STPPRD]/2;
            STOPEPWM1();
```

```
                    STARTEPWM2();
                }
                }else
        {           STOPEPWM1();
                STOPEPWM2();

        }
//                  EPwm1Regs.TBPRD= 9999;
//                  STARTEPWM1();
}
# include "DSP2833x_Device.h"
//---------------------------------------------------------------------------

volatile Uint16 Flag_1ms; // Define the main loop timer flag
volatile Uint16 GlobalDelayCount;   //      Global delay
//---------------------------------------------------------------------------
//
//      variables from GPIO
//
Uint16 SteppingTim1;        //      GPIO76
Uint16 SteppingAlarm;       //      GPIO75
Uint16 SteppingEnd;  //      GPIO74
```

```
Uint16 ExSw1;        //      GPIO68
Uint16 ExSw2;        //      GPIO67
Uint16 ExSw3;        //      GPIO66
Uint16 ExSw4;        //      GPIO65
Uint16 ExSw5;        //      GPIO64
Uint16 LimitSw1;     //      GPIO46
Uint16 LimitSw2;     //      GPIO47
Uint16 LimitSw3;     //      GPIO80

Uint16 IsReset;
//---------------------------------------------------------------------
//  sensor variables from adc
//
float LoadCellOfs;   //      ADCA3
// Uint16 LoadCellOffset;
/*
float Torque;  //      ADCA1
float Pressure0;     //      ADCA2
float Pressure1;     //      ADCA3
float Pressure2;     //      ADCA4
float VariableRegister0;     //      ADCA5
float VariableRegister1;     //      ADCA6
```

```
float Reserve;  //        ADCA7
*/
//---------------------------------------------------------------------
//
// output variables for GPIO
//
Uint16 SteppingACL; //        GPIO79
Uint16 Steppingx10;   //        GPIO78
Uint16 SteppingCOFF;        //        GPIO77
Uint16 MaxonDacCs; //        GPIO73
Uint16 MaxonDacSck;         //        GPIO72
Uint16 MaxonDacSdi;         //        GPIO71
Uint16 MaxonDacLdac;        //        GPIO70
Uint16 MaxonEN;     //        GPIO69
Uint16 LED1; //        GPIO40
Uint16 LED2; //        GPIO41
Uint16 LED3; //        GPIO42
Uint16 LED4; //        GPIO43
//---------------------------------------------------------------------
//        // output variables for EQEP
//POSSPEED qepSTP_posspeed = POSSPEED_DEFAULTS;
POSSPEED qepDC_posspeed = POSSPEED_DEFAULTS;
```

```
//---------------------------------------------------------------------
// Global DATA
// Spi communication data
//
Uint16 MasterStatus[DATALEN];
Uint16 SlaveStatus[DATALEN];
Uint16 gPointer;
volatile Uint16 BusyFlag;
//---------------------------------------------------------------------
// Control variable
//
float StpV0;
int StpDeg0;   //        Q15
int DcDeg0;    //        Q15
```

Biographic Sketch

Xu Ma received his B.Sc. degree and his M.Sc. degree from Changchun University of Science and Technology in 2006 and 2010, China. Currently, he is a Ph.D. candidate in Kagawa University, Japan. He researches on robotic catheter systems and haptic devices.

He has published about 9 refereed journals and conference papers in recent there years.