Interactive Navigation and Control of Neurosurgical Robotic Systems

by

Meftah Mohamed

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
August 2017

To my family

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Trauma and head injury are an important cause of overall mortality and morbidity. Millions of people around the world suffer from neurological and functional disorders, but there are only a limited number of neurosurgeons and neurologists to help them. Due to the complex distribution of blood vessels and nerves under the skull, neurosurgical procedures are risky and require surgeons to have expertise and excellent coordination to avoid injury to the dura and to achieve safety, efficiency, and accuracy. Neurosurgical procedures are usually performed manually by a surgeon using drilling or milling devices. Most of the tools currently used in surgery depend only on the surgeon's manual skills to stop the penetration when drilling a hole or removing cranial bone to access the brain.

This work presents the control theory of a robotic neurosurgical system. The aim is to build a reliable neurosurgical control system featuring intelligent control, force-feedback control, dexterity, and flexibility, along with human-robot interaction. A modular neurosurgical robotic system is developed to validate the control algorithms in a safe manner. The proposed robotic system could be used as a training platform to help students acquire the skills needed to perform surgeries on real patients and practice their technical skills without any risk to patients. This work introduces the interactive navigation of the surgical planning and control of robot movement in order to increase a surgeon’s integration in the control of a robotic system. Safety can be potentially increased due to faster surgeon reactions during the process, while drilling process parameters could be controlled automatically.

LIST OF ABBREVIATIONS AND SYMBOLS USED

ABI	Acquired Brain Injury
TBI	Traumatic Brain Injury
VRML	Virtual Reality Modeling Language
V-REP	Virtual Robot Experimentation Platform
ROS	Robot Operating System
MRI	Magnetic Resonance Imaging
CT	Computed Tomography
BRW	Brown-Roberts-Wells
FDA	Food and Drug Administration
MODICAS	Modular Interactive Computer Assisted Surgery
MKM	Mehrkoordinaten Manipulator
API	Application Program Interface
DOF	Degree of Freedom
MHI	Mitsubishi Heavy Industries
D-H	Denavit Hardenberg
ci	Cos Θi
si	Sin Θi
J	Jacobian Matrix
Q	Joint Angular Position
M	Inertia Matrix
G	Gravitational Force
t	Input Torque Vector
Fr	Friction

PID	Proportional–Integral–Derivative
LQR	Linear Quadratic Regulator
n	Rotating Speed
V	Linear Velocity
D	Diameter
RMSE	Relative Mean Squared Error
CC	Cross-Correlation
MAE	Mean Absolute Error
NN	Neural Networks
CAD	Computer-Aided Design
XML	Extensible Markup Language
MCS	Motion Control Section
OCS	Operation Control Section
VTK	Visualization Toolkit
ITK	Insight Segmentation, Registration Toolkit
FSL	FMRIB Software Library
SPM	Statistical Parametric Mapping
MIA	Medical Image Analysis
MITK	Medical Imaging Interaction Toolkit
URTC	Unified Real-Time Communications
ICS	Image Coordinate System
PCS	Patient Coordinate System
RAS	Right Anterior Superior
OR	Operation Room

SAPI	Speech Application Programming Interfaces
TTS	Text into Speech
VR	Virtual Reality
ANN	Artificial Neural Network

ACKNOWLEDGEMENTS

First of all I want to thank God for his help and his mercy to complete this thesis. In many ways, I am deeply indebted to all the people, who contributed to this work. Especially, I would like to express my sincere appreciation to my supervisor, Dr. Jason Gu. Without his unlimited support, kindness, patience, guidance, and advice, this work would have not been possible. Not only has he provided me with academic guidance, he has also given constant caring and encouragement in the pursing of my Ph.D. studies.

I am also thankful to Dr. William Phillips and Dr. Mohamed E. El-Hawary for their kindness in serving as committee members and for their help and support. I am also grateful to the External Examiner Dr. Rickey Dubay, for spending his valuable time reading and evaluating my thesis.

I want to emphasize my special thank Libyan Ministry of Higher Education and CBIE for providing financial assistance in forms of scholarships. I want to thank all other people, who have followed and contributed to this work, which I cannot name all individually here.

Special thanks are due to my mother, sisters, and brothers for supporting me emotionally and spiritually in the writing of this thesis and in my life in general. I cannot thank you enough for encouraging me, and I know that your prayers helped sustain me throughout this journey. My greatest thanks my lovely wife Abeer Almuhalhil, for her unremitting encouragement, approvals, acknowledgments and her love.

CHAPTER 1 INTRODUCTION

1.1 STATEMENT OF PROBLEM

Applying robotics to neurosurgical procedures is considered to be promising, yet challenging. Interactive navigation and control theory has been developed for the proposed neurosurgical robotic manipulator. Simulations and experiments have further verified the proposed approach.

1.2 MOTIVATION FOR THE STUDY

Trauma and head injury are an important cause of overall mortality and morbidity worldwide [1] [2], following cardiovascular disease and cancer. Approximately 90 million people suffer from neurological and functional disorders [3]. However there is a limited number of neurosurgeons and neurologists [4]. Head trauma treatment may necessitate relieving pressure in the patient's skull by drilling into it [5]. This risky surgery is further complicated when it is performed remotely [6]. Usually, there is increased risk when patients have to travel longer distances that may aggravate their condition [7]. In Canada, there are over 165,000 brain injuries per year, associated with more than 11,000 deaths each year[8]. Head injuries contribute to around 30% of all injury deaths in the United States [9].

An acquired brain injury (ABI) can involve traumatic or non-traumatic events. A traumatic brain injury (TBI) is an injury caused by an external force, such as a sports injury, a motor vehicle crash, a fall, an assault, or a gunshot wound. In contrast, a non-traumatic brain injury is an internal injury which can result from a stroke, a loss of oxygen to the brain, or meningitis [8]. The impact of an ABI is felt by the survivor, the family, and caregivers. ABI is the leading cause of death and disability for Canadians under the age of 40 [1]. The combined economic burden of acquired brain injuries and treatment has been estimated to be greater than $12.7 billion per year [10]. Each year, TBI contributes to a substantial number of deaths and cases of permanent disability in Canada and the United States. An estimated 1.7 million people sustain a TBI annually in the United States. Of this number, on average 52,000 people die, 275,000 are hospitalized, and 1.365 million are treated and released from an emergency department [11]. TBI is a contributing factor in 30.5% of all

injury-related deaths in the United States, where direct medical costs and indirect costs of TBI, such as lost productivity, have been estimated to total $60 billion [12].

The few hours immediately after a brain injury has occurred are the most critical. There are three stages of treatment: Initial treatment, acute treatment, and surgical treatment. In initial treatment, a medical doctor aims to stabilize the brain injury and swelling of the brain. Monitoring devices and drainage devices are used to help regulate the pressure on the brain. Healthcare staff works together to resuscitate the patient and stabilize the vital signs. Acute treatment aims to assist breathing, restore the healthy circulation of oxygen to the blood, and remove blood clots. The last stage is surgical treatment, which aims to reduce swelling, as well as pressure on the brain and skull. Pressure on the brain occurs when the brain tissue swells and is compressed against the skull. Because the skull cannot expand, blood is prevented from circulating, and the brain cells become permanently damaged [13].

Neurosurgery focuses on the nervous system. The discipline of neurosurgery arose as a result of an increasing need for specialized expertise in the surgical and non-surgical treatment of the nervous system through physical examination and surgery. Based on the Royal College and Pathway Evaluation Program profile for neurosurgery, in Canada, there are only 314 registered neurosurgeons. This is a small number in comparison with the number of patients who have had an incident of TBI [14].Therefore, there is a need to develop technologies that minimize the waiting time for TBI treatment and improve the overall results. The 2015 Waiting Your Turn survey indicates that, in general, the total waiting time for elective medical care across Canada is slightly longer than in 2014, and that it remains at a historically high level. In 2015, among the various specialties, the shortest total waits for neurosurgery were 12 weeks [15]. This survey reveals that wait times in Canada are longer than the periods which physicians consider to be clinically reasonable.

Head injuries themselves remain the number one killer in major trauma and place an enormous burden on the healthcare system. Injury levels, the geographic situation and the distribution of healthcare facilities do not allow every patient to have access to the appropriate medical treatment. Most patients should be transferred to primary trauma centers within one hour. This fast response time is very critical and is usually impossible in many situations, even with the aid of air transport. As a result, some people die as a result of the traumatic event before reaching the acute care stage.

All of these factors point to the need for an efficient method of providing treatment quickly, reliably, accurately, and in a cost-effective manner. This problem could be partially solved through the use of robotics.

Medical science has become one of the largest industries in the civilized world. There are many possible ways in which robots might be used for biomedical applications such as surgical robotics. Precision and miniaturization are the major potential advantages of robotic surgery, together with the possibility of three-dimensional magnification and articulation exceeding normal manipulation. Robotic surgery could be used in a wide range of procedures because it provides improved vision, dexterity, and precision. The main drawback of robotic surgery is the high initial capital investment for the robot system. In addition to the cost, robotic systems present many challenges such as the lack of tactile feedback, and operating field limitations that must be overcome before they can be fully integrated into the existing healthcare system.

Technological advances have made many neurosurgical procedures achievable today that only a few years ago were considered risky. Biomedical robotic systems include the Da Vinci, Aesop, and Hermes systems. Trends in the surgical robot industry indicate that it could be a dominant tool in complex procedures such as cardiac surgery, gynecology, neurosurgery, orthopedics, gastrointestinal surgery, pediatrics, urology, etc. In the past twenty years, researchers have aimed to make robots more autonomous and to enhance the control of robots by surgeons. Surgical robots can be used as training tools to extend the surgical skills of a new surgeon.

In future, surgeons may use robotic manipulators to perform a series of surgical applications [16].For example, surgical robots could be integrated with real-time imaging techniques and could have tactile feedback. The systems may become smaller and even less invasive. More improvements in size, tactile sensation, and cost are expected for the future. Over the last decade, new equipment and techniques have been designed to assist humans in the field of surgery. There are many challenges facing doctors which point to the use and development of surgical robots, such as the lack of availability of physicians in times of emergency, and the absence of state-of-the-art medical facilities in smaller cities and towns. This has motivated researchers to consider how to design robotic arm controllers to be more efficient, with centralized and supervisory control systems; and how to optimize the sophisticated human-robot interaction to simplify data acquisition and monitoring techniques.

Neurosurgical robotics could be used to perform difficult tasks in head surgery, such as drilling a hole in the skull of a patient to relieve the pressure associated with head trauma, and addressing chronic conditions, so as to save time and costs associated with transport. Neurosurgical robotics could thus be used to improve health services by increasing cost-effectiveness, reducing care delivery time, and enabling surgeons to perform interventions more precisely, flexibly, and safely, from greater distances.

1.3 THESIS OBJECTIVES

The aim is to build a reliable neurosurgical robotic manipulator which can be used in biomedical applications such as head trauma surgical treatments. The proposed system has three main parts: A "planner", guided by a map of the robot workspace environment; a Mitsubishi PA10 robotic manipulator, which performs the surgery; and sensing devices, which perform position, orientation and force measurements to keep the arm on course. Artificial intelligence algorithms are used to improve overall system performance. Modeling of the robotic manipulator makes it possible to study force feedback control, kinematic control, and interactive navigation.

Control algorithms are developed for the surgical manipulator, and simulations and experiments are performed to verify the proposed approach. Moreover, the design helps to determine accuracy and improve reliability. An efficient real-time control algorithm is proposed to control the robotic arm. The proposed robotic neurosurgical control system features force feedback control, dexterity, and flexibility. The challenging problems associated with the system include a differential kinematic control for the robot, force-related processing, real-time tissue identification, skull breakthrough detection, and optimized trajectory planning.

1.4 CONTRIBUTIONS

This research aims to solve the control problems related to the proposed robotic neurosurgical system through:

- Modular design of an open architecture neurosurgical robotic system featuring intelligent control, force feedback control, and human-robot interaction. The proposed robotic system could be used as a training and evaluation platform to allow testing of the skills needed to

perform surgery on real patients, and to reduce the time restrictions of training under senior supervision.

- Design of interactive navigation and control of the neurosurgical robotic system, featuring surgical planning using medical images, pose registration using force steering, and virtual reality surgery simulation. The proposed architecture could increase the integration of the surgeon's role and improve safety during the process.
- Development of a trajectory planner for the neurosurgical robot, and design of a trajectory controller incorporating force limit constraints on the robot arm. Implementation of a force sensor at the end effector and development of a signal processing algorithm to minimize the effect of vibration and to compensate for noise effects by applying a reliable sensor fusion technique. Artificial neural networks (ANN) are used to design a drilling state classifier. An ANN-based force state classification is proposed to determine the robot state when performing tasks such as rotation, waiting, starting and stopping of drilling, and retrieval. In the proposed approach, an ANN is constructed and trained to classify various states via the force data.
- Semi-automatic control. To permit safe use of the proposed system, the system can be switched from an automatic to a manual control state. Task planning is proposed to enable the surgeon to define the area of interest and safety margins.
- Control theory concerning the proposed neurosurgical robotic system is presented in this thesis, which focuses on the modeling and control of the Mitsubishi PA10-7C robotic manipulator, LQR controllers, ANN drilling state detection, human-robot interactions for neurosurgical manipulators, 3D simulation of the operating room, and a semi-automatic control architecture that makes the proposed system work in a stable, reliable, safe and efficient manner.
- Systematic modeling and control of the Mitsubishi PA10-7C robot manipulator are presented. The Mitsubishi PA10-7C robot arm is modeled systematically, with the aid of kinematic models and the Jacobian matrix. General set-point control and trajectory tracking control use a feedforward controller to move the perforator smoothly.

1.5 RESEARCH CHALLENGES

- Because neurosurgical robotics is an interdisciplinary research area, an open architecture design is necessary to permit the integration of different system components.
- The neurosurgical robotic manipulator is difficult for neurosurgeons to use. Human-robot interactions should be designed carefully to facilitate the training of unskilled doctors.
- The neurosurgical robotic manipulator should be able to overcome poor decision-making, thus improving safety measures for different scenarios to build an expert system.
- The challenging problems associated with the system are; differential kinematic control for the robot, force data processing, real-time tissue identification, skull breakthrough detection, and optimized trajectory planner.

1.6 OVERVIEW OF THE THESIS

Chapter one provides a detailed presentation of the problem. Analysis of specific objectives and specifications forms the basis of implementation. Moreover, challenges associated with the proposed system are explained.

Chapter two describes state-of-the-art neurosurgical robotics by analyzing the current situation and future trends. The architecture of the proposed system is constructed, and problems associated with the system are stated.

Chapter three presents kinematic and dynamic models for serial manipulators, and serves as a foundation and reference for the remainder of this work. Kinematic modeling of the Mitsubishi PA10-7C robot arm is developed in Section 3.1. Section 3.2 presents the Jacobian matrix computation. Section 3.3 describes the dynamics of serial robots, and the final section summarizes the chapter.

Chapter four reviews the operation of the robotic control system. Part of this evaluation and associated conclusions give information about aspects of the functioning of the arm itself, in order to assist future operators.

Chapter five presents force information processing and drilling state detection methodology based on neural networks. Force information from the JR3 force sensor is analyzed first. Robot state detection is used to distinguish the main states, including rotation, waiting, starting and stopping of drilling, and retrieval.

Chapter six presents simulation and experimental results of manipulator control algorithms. In the simulation part, the mechanism for building a modular design of a neurosurgical robotic manipulator is implemented by using multiple simulation tools such as VRML, MATLAB, and V-REP. This approach provides a unified framework for fast, cost-effective testing of control algorithms. A remote controller is implemented in the MATLAB/Simulink environment. Moreover, the system physical environment is modeled on the virtual robotics experimentation platform (V-REP). The MATLAB/Simulink controller is synchronized with V-REP via the ROS interface.

Chapter seven concludes the thesis with a discussion of the results, and suggests future research directions.

CHAPTER 2 BACKGROUND AND RELATED WORK

Robotics is used in many biomedical applications, such as robotic surgery. Robotic surgery could be used for a wide range of procedures, because it provides better vision, dexterity, and precision than is possible with standard minimally invasive surgery. This chapter presents the state-of-the-art of neurosurgical robotics, with an analysis of the current situation and future trends. The proposed system architecture is constructed, and problems associated with the proposed system are stated.

2.1 REVIEW OF BONE DRILLING

The skull is a bony structure comprised of two parts, the neurocranium and the facial skeleton [17]. The neurocranium is the protective cranial cavity that houses the brain and brainstem, and supports of the face. There are typically 22 bones in the human skull [18]. Cranial bones are illustrated in Fig 2.1.

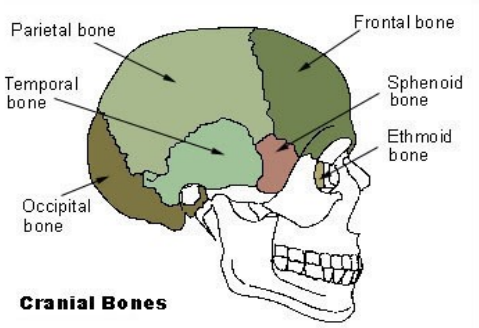

Figure 2.1 Structure of the Cranial Bones [18]

Injuries to the brain can be life-threatening, causing significant brain damage due to increased intracranial pressure. This necessitates the performance of an urgent procedure to relieve the pressure [19]. A subdural hematoma is a type of hematoma usually associated with traumatic brain injury, where blood collects between the brain and the dura mater; this causes an increase in intracranial pressure and damage to the brain tissue [20]. Subdural hematomas are often life-threatening. Their treatment depends upon the size and rate of growth, and can be managed via a catheter or a craniotomy, where a hole is drilled through the skull to remove the hematoma [21]. Surgical robotics could be used to improve health services by enabling surgeons to perform

interventions more precisely, flexibly [22], and safely, and from greater distances [23]. It could thus be used to improve health services and to minimize healthcare delivery times [24].

Skull drilling with the aid of an automated industrial robot is a very challenging task, because it is necessary to maintain the force applied so as to achieve high-quality drilling, with a minimally invasive procedure. Keeping the drilling force low minimizes the risk of overshooting or position sliding when the robot is in contact with the skull. Sliding during drilling can cause the hole not to be orthogonal to the skull, resulting in poor hole quality, and possibly damaging the cutting tool due to tangential forces that act on the cutting tool. The key aspects of skull drilling are positioning accuracy, accessibility, cycle time, and the quality of the hole. Although half a century has elapsed since initial investigations of bone drilling were carried out [25], there is still no general agreement on the parameters of bone drill design [26]. Most of the drilling tools currently used in surgery depend only upon the surgeon's manual skill to stop penetration when completing a hole. During skull drilling, a drill makes a hole in the skull [27]. Numerous studies of bone drilling have been performed to establish optimum drilling conditions and drill geometry for skull drilling. Many preliminary applications of surgical drilling systems have been demonstrated to be clinically useful [28].

The first recorded medical application of a robot, by Sakaguchi, occurred in 1985 [29]. There has been a steady increase in robotic systems and technologies applicable to surgical procedures [30]. Glauser designed a stereotactic neurosurgical robot, which is used to insert a probe with a diameter of 2 to 3 mm through a hole drilled in the skull [31]. A motor-driven drilling tool is used to perforate the bone [32]. A mechatronic drilling tool is presented for precise drilling of soft bone tissues during ear surgery. By using force and torque sensor data, the designed system can complete the breakthrough with minimum drill bit protrusion. Kaburlasos developed a two-level fuzzy controller for estimating the thickness of a bone with the aid of a force/torque pair of drilling profiles [33]. For drilling into long bones, Bouazza described a reliable, repeatable method of breakthrough detection, based on a modified Kalman filter. This manipulator allows a drill bit guide to be automatically aligned with the planned drilling trajectory [10]. The surgeon can then perform manual completion of the drilling stage. The force sensor is used as a safety enhancement. Shen developed a telerobotic skull drill for neurosurgical applications; the controller design realizes three phases during operation. A wave variable-based approach is applied to ensure the

stability of the system. Drilling depth is controlled, and the target joint angles of the robot are adjusted to place the payload in the desired position. This is based on knowledge of deflections in the arm derived from the mathematical model interpretation of the disturbance caused by the payload [34].The robot should provide enough force to counterbalance the skull reaction force. A certain amount of force is required for drilling and cutting tasks performed on the patient's skull.

A craniectomy is a neurosurgical procedure for removing cranial bone, to allow the surgeon to access the brain to treat brain tumors, cerebrovascular disease, and brain injuries. A craniotomy may be small or large, depending upon the problem. Craniotomy procedures are risky due to the complex distribution of blood vessels and nerves under the skull. A craniotomy requires a high level of expertise and coordination on the part of the surgeon, to protect the dura from injury, and to ensure safety, efficiency, and accuracy [35].

The primary challenges for a craniotomy still apply, because it is a time-consuming and laborious procedure. A craniotomy can be performed with a variety of tools for bone removal, including a drill cutter, an artificial stop cranial drill, wire saw tools, and milling tools. The surgeon directly controls only basic tool parameters such as spindle speed, and the result depends entirely on the skill of the surgeon. A bone flap procedure is common in neurosurgery; it is performed by drilling several holes in the skull [36], and linking the holes so that a complete piece can be removed from the skull. This procedure takes more than 30 minutes, subjecting the surgeon to pressure and fatigue, and possibly reducing the effectiveness of subsequent surgery. Safety and stability are the main design considerations for craniotomy devices [37]; Specifications should be determined in accordance with medical ethics, to protect the internal structures of the cranium. The intervention of the surgeon is required in the case of hardware failures, software errors, and non-planned situations, where the surgeon should be able to take control of the procedure and continue with the surgery.

2.2 STATE-OF-THE-ART NEUROSURGICAL ROBOTICS

Neurosurgical robotics is a branch of surgical robotics. A surgical robot is a controlled manipulator with artificial sensing, that can be reprogrammed to carry out a range of surgical tasks. Surgical robotics is a promising and challenging field [38]. It could be used to improve health services by saving money and ensuring the timely delivery of care [24]; and could enable surgeons to perform surgery more precisely, flexibly, and safely, from greater distances [39]. In future, surgeons may

use robotic manipulators to perform a series of surgical applications such as skull drilling and milling [40]. Surgical manipulators have many advantages, including a high level of accuracy, stability, reliability, and flexibility [41]; Since the beginning of the 1960s, robots have been available with more accurate positioning systems and high interface stiffness [42], By the 1980s, surgical robots became increasingly attractive. In accordance with the targeted anatomy, or surgical tasks, surgical robots can be classified as orthopedic, microsurgical, urology, cardiac, laparoscopy, percutaneous, or neurosurgical robots.

Neurosurgery was one of the first surgical fields where robotic surgery was introduced [43]. Neurosurgical robots are among the most technologically advanced devices in the modern operating room [44]. They have extended the dexterity and stamina of neuro surgeons, enabling them to achieve an unprecedented precision of tool manipulation [45] and to work with greater accuracy at the microscopic level [46]. Neurosurgery procedures are usually very complex, requiring an experienced surgeon with highly developed skills [47]. Many preliminary applications of neurosurgical robotic systems have been demonstrated to be clinically useful. Following the pioneering medical application of Sakaguchi's robot in 1985 [48], there has been a steady increase in robotic systems and technologies for applications in critical surgical procedures. Sakaguchi's robot was designed to perform a percutaneous nephrostomy. Localization was carried out by establishing the trajectory from an entry point on the skin to a target point in the dilated renal pelvis. The neurosurgical robot could be used to accomplish several tasks; for example, it could be used as a retractor holder, as an endoscope holder, as a holder for insertion of a biopsy cannula or electrode [49], or even as a surgical microscope [50].

Neurosurgical robotic manipulators are usually image-guided manipulators [51]; where the images can include magnetic resonance imaging (MRI) or computed tomography (CT) images [52]. In 1985, a PUMA robot was proposed to control a biopsy needle inserted into the human brain. A CT scanner was used to examine a 52-year-old man. The robot pushed a biopsy needle toward a suspected area of the brain, in accordance with CT images of the target [48]. The tissue biopsy was sucked out with a syringe and sent directly to the pathology lab. In 1987, a robot was used to perform stereotactic targeting [53]; Benabid was the first neurosurgeon to report using a robotic manipulator with six degrees of freedom (DOF), linked to a stereotactic frame. The main practical use of this robot was in image-guided robotic endoscopy. The robotic manipulator was used to

control the movement of a probe to reach a target calculated from x-rays and angiograms. By 1992, this methodology had been used in 140 cases. Benabid established the potential use of robots in endoscopic neurosurgery.

Figure 2.2 ‘Minerva’ neurosurgery robot in position [54]

The Minerva robot was one of the earliest neurosurgical robots. It was used for two operations on patients in September 1993, at the CHUV Hospital in Switzerland [55]. The Minerva system is designed for precise stereotactic brain biopsy procedures [56], and is dedicated to inserting a needle according to a CT scan map, under the supervision of a commanding surgeon. A real-time CT scan ensures high precision and increases the reliability of the procedure.

Figure 2.3 Renishaw-Mayfield NeuroMate [57]

Figure 2.2 shows a Minerva neurosurgical robot in position, with the patient adjacent to a CT scanner [64]. A Minerva system has a structure with five degrees of freedom, and a Brown-Roberts-Wells (BRW) reference frame. The robot is attached to a robot gantry, which is a horizontal carrier that moves on rails [58]. There is a stereotactic frame, coupled to a motorized CT table by two ball and socket joints arranged in series. The NeuroMate, a neurosurgical robot approved by the FDA, was the first commercially available robot with six degrees of freedom [59]; It was invented by Benabid, Lavallee and colleagues at Grenoble University Hospital in France [60]. More than 1600 procedures have been performed by NeuroMate robots since 1989, including a range of neurosurgical procedures such as tumor biopsies [61],mid-line stereotactic neurosurgery [62], functional neurosurgery of the basal ganglia, and stereo-electro-encephalographic investigations of patients with epilepsy [63]. Preoperative 3D imaging is used for navigation and manipulator localization[64].

Figure 2.4 Clinical testing of MODICAS assistance robot [65]

Figure 2.4 shows the MODICAS (Modular Interactive Computer-Assisted Surgery) robotic system which is an integrated solution for the software-based combination of a surgical planning, localization device and a haptic sensor with a robotic manipulator to support surgical interventions. MODICAS introduced by Castillo Cruces [66] which combine PA10 robot manipulator with a common surgical navigation to one integral unit. Minimally invasive neurosurgery procedures are an important field [67]. A development in the field of robot-assisted neurosurgery took place in

China in 1995 [68], The neurosurgical robot was used to locate a tumor target in the brain, making it possible to avoid radiotherapy exposure while injecting dangerous radioisotopes via remote control. This system consists of three components: An image-guided surgical planning and support system, marker-based registration with some measurement tools, and a robot with six degrees of freedom and a custom-designed surgical device. The Zeiss Mehrkoordinaten Manipulator (MKM) is a guided microscope which was developed in the late 1990s [69]. The MKM is designed for frameless stereotactic procedures. It superimposes a definition of the target on the view of the surgical field. This is similar to but more accurate than the BRW stereotactic frame, which was one of the first image-guided surgery devices.

The application of telerobotics to neurosurgery is a promising research area. There has been considerable interest in telecontrolled surgical robotics [70]. Since the 1990s. In addition to design issues associated with local surgery, the primary limitation of all of these systems is the effect of communication delays [71] and possible interruptions on the overall performance. On May 12, 2008, the NeuroArm robotic system made history when it was used to operate on a human patient at the Faculty of Medicine at the University of Calgary [72]. This landmark operation marked the first time that a robot was used to perform image-guided neurosurgery [73]. A brain tumor was removed from a 21-year-old patient.

Figure 2.5 Photographs showing NeuroArm in use [61]

Figure 2.5 shows the NeuroArm in use. The main image depicts the workstation for the OR. The surgeon controls the workstation haptic hand, and the assistant surgeon is stationed opposite the NeuroArm. The introduction of robotic surgery could significantly improve the quality of many

surgical procedures (e.g.,[74] [75] [76]). In the past two decades, many methods for designing image-guided surgical robotic systems have been developed. In 2016, real-time navigation of breast cancer surgery was performed at Queen's University, and succeeded in reducing the positive margin rate [77]. Position sensors were applied to the tumor localization needle, and 3D navigation views were generated by using real-time tracking information. Xia et al. [78] combined a NeuroMate robot with a 3D slicer to provide mechanical assistance for a human skull base drilling system. Tauscher [79] developed an interface for integrating a robot into an image-guided therapy system. Simplified integration was achieved by using only a single programming context for the implementation of the state machine, the interfaces, and the robot control. PA10 robots are used in teleoperation neurosurgical applications [80],[81][82][83].

Table 2.1 Examples of Neurosurgical Robotic Systems

System	Studies	Institution/company	Country
Minerva	Commercial use	Univ. of Lausanne	Switzerland
NeuroMate	Commercial use	Grenoble Univ. Hospital	USA
MRI compatible	Tissue samples	Univ. of Tokyo	Japan
Minerva	human	CHUV Hospital	Switzerland
NeuroArm/SYMBIS	Experimental Setup	IMRIS, Winnipeg	Canada
Rosa	Commercial use	Medtech	French
Neurobot	Experimental Setup	Dalhousie University	Canada

Neurobot is a neurosurgical robotic system developed by using multiple simulation and hardware tools such as VRML, Matlab, and V-REP. Neurobot based on PA107c robot and features a unified framework for quick and cost-effective testing of control algorithms [84]. Remote Controller implemented in the Matlab/Simulink environment. Moreover, the System physical environment has been modeled on Virtual Robotics Experimentation Platform (V-REP). Matlab / Simulink controller synchronized with the V-REP by using ROS interface. In this brief overview, it has not been possible to cover all aspects of the rapidly developing area of neurosurgical robotic systems; however, Table 2.1 summarizes some examples of these systems.

2.3 BASIC REQUIRMENTS OF NEUROSURGICAL OPERATIONS

Several criteria need to be fulfilled to make a robotic system attractive for neurosurgery clinical applications [85]. The robotic system should be accurate, image-guided, and cost-effective [86];

and should permit faster results than can be achieved by a neurosurgeon using an operating microscope [87]. Such a system should be semi-autonomous, allowing the surgeon to exercise direct control in real time, even in telerobotic surgery. An increasing number of locations have few neurosurgeons available. The selection of a robotic manipulator depends upon the design requirements of the neurosurgical applications. Important factors include the capability of handling the drilling tool, ease of assembly, the strength and durability of the parts, light weight and high joint stiffness, and the ability to eliminate backlash.

Table 2.2 Basic Design Requirements of the neurosurgical robotic System

Property	**Value**
Motion Precision	1 mm
Repeatability	1mm
IP code	≥ 57
Load weight	$\geq 6\ Kg$
End Effector Force	$\geq 60\ N$
robot total length	$\geq 90\ cm$

Table 2.2 presents the basic design requirements of robotic neurosurgical systems. The combination of all of these factors significantly influenced the design choices made in the selection of the type of robotic manipulator and the kind of controller. Because the designed system might be used in rural areas, the neurosurgical robotic manipulator should permit less specialized doctors to perform more operational tasks in a user-friendly environment, by applying human-robot interaction techniques. The use of MRI, CT scans, and x-rays in the operating room should be minimized to suit the needs of clinics and hospitals in the least developed countries. The navigation and planning algorithm should be adaptable to patient variations, and able to overcome poor decision making. Different scenarios should be discussed in detail to build an expert neurosurgical robotic system.

2.4 DESIGN SELECTION OF THE ROBOTIC MANIPULATORS

There are many design selection criteria for robotic manipulators, most of which depend upon previous experience, and on customer requests or requirements [88]. These considerations influence the shape, and choice of materials for the fabrication of the robotic arm. Other factors to be taken into account when selecting robotic manipulators include design cost, fabrication time,

the ease of manufacturing the parts and the method of manufacturing, the simplicity of assembly, and the strength and durability of the parts. The combination of these factors has significantly influenced all choices made in the design selection of the robotic arm. The principal requirements for robot power transmission depend upon the correct selection of design parameters such as:

- Low energy losses and friction for improved responsiveness of the control system
- Small size, and low weight and moment of inertia
- High effective stiffness, and elimination of backlash
- Accurate and constant transmission ratio

2.5 TELEOPERATION OF ROBOTIC MANIPULATORS

The teleoperation of robotic systems is an important research topic. In bilateral systems, information flows in two directions between the operator and the robot [89]. Remotely controlled robotic manipulators are required when robot self-decisions are difficult or in cases where human presence at the robot site is undesirable or difficult [90]. In many instances when complex operations have to be performed in the event of an emergency, surgeons may be unable to be in time to operate, possibly resulting in fatalities. In order to overcome such problems, a solution is to use teleoperated robots controlled by surgeons, via a network where the patient and the surgeon are geographically separated [41]. There are many advantages of using teleoperated robot manipulators in the biomedical field. It could increase the accuracy of doctors and save their time, save money and effort in sending patients to the doctor, strengthen medical staff experience, and reduce the shortage of skilled surgeons. There are many challenges involved in the use of remotely operated robotic manipulators, starting with the communication link, where usually there can be no guarantee concerning transmission times [91]; Therefore, many interface designs are not suitable for time-critical interactions, such as remote assembly with force feedback. Multitask robot manipulators also present difficulties, because the commands are accommodated in parallel.

2.6 SKULL DRILLING PARAMETERS

Drilling of the skull is a common procedure in neurosurgical operations. The development of automated skull drilling systems aims to minimize human error. Skull Drilling parameters can be classified as drill specifications and drilling parameters. Drill specifications include the drill point,

drill diameter, cutting face, rake angle, clearance angle, and drill wear; while drilling parameters include drilling speed, feed rate, drilling energy, cooling, drilling depth, pre-drilling and drilling time [25].

Thermal damage is a key factor in skull drilling. The increase in temperature during such a procedure increases the chances of a thermal invasion of the bone, which can cause thermal osteonecrosis, resulting in increased healing time. Therefore, drilling of the bone with a minimum temperature is a major challenge in bone drilling. Osteonecrosis is a disease arising from reduced blood supply to the bone [92]; Due to thermal damage, bone drilling can also cause microdamage to the bone. An average temperature of 47 °C for one minute is used as a threshold [93], above which it is considered likely that necrosis of the human bone will take place.

Skull drilling speed is related to friction and temperature. There is no universally accepted standard for drilling speed [25]; most previous research done on bone drilling has found that with increased drilling speed, the heat generated increases [94]. Another important parameter is the effect of applied force combined with drill speeds, which should be restricted to less than 20 N for safety reasons [95]. Increasing either the speed or the loading force causes an increase in temperature in the bone structure. However, increasing both the speed and the load together has been found to allow for more efficient cutting with no significant increase in temperature [96]. Skull thickness estimation is critical in neurosurgery, to minimize error in skull drilling. Skull thickness cannot be measured during the surgery because the skull is not regarded as a uniform layer. Estimated thickness should be incorporated into realistic geometric head models to improve resolution in skull drilling procedures. Many studies of thickness of the human skull have been performed by using physical measurements of thickness [97], and qualitative analyses of photographs and CT scans of the skull [98]. There are a variety of thickness estimation algorithms that incorporate physical properties of the human skull [99]. Some researchers have used a finite element approach. Skull localization and mapping are required to define the thickness of each part of the cranial section.

There are many methods which can be used, such as international 10-20 system EEG electrode placement, which is increasingly applied in cognitive neuroscience and psychiatric treatment studies [100]. Many researchers have developed models which represent the scalp, skull, and brain as separate layers; however, they have assumed the thickness of the layers to be uniform [101].

This assumption is not realistic, because there are local variations in skull thickness for each part of the cranial section. Moreover, various human races have physical differences in skull thickness measurements. The use of realistic human skull models could make it possible to reduce the probability of neurosurgical complications. In measuring skull thicknesses, Law [102] found that the mean width and standard deviation was 5.2 ±.8 mm. Craniometry parameters are directly related to race and sex. Such parameters are also widely used in modern forensic anthropology to determine racial affinity from human crania [103].

Based on research done by Nawrocki [104], skull thickness is measured and related to anthropological markers and international 10-20 markers. The sample size for each landmark was 76, including American black and white males and females in approximately equal proportions, who died between approximately 1900 and 1950. Adeloye [100] measured the thickness of the cranium at four different points in the sagittal plane, for a population of 300 black Americans and 200 white Americans. Adeloye concluded that there is a rapid increase in skull thickness during the first two decades of life, followed by a small uniform increase, reaching a peak in the fifth and sixth decades.

2.7 ARCHITECTURE OF THE PROPOSED SYSTEM

The objective of this research is to develop an interactive navigation and control system for a neurosurgical robot, and to design a real-time trajectory controller incorporating constraints on the robot arm such as force limits. A force sensor at the end effecter is to be designed and implemented, and signal processing algorithms are to be applied to minimize the effect of vibration and to compensate for noise effects by implementing a reliable sensor fusion technique. The proposed system could be used as a training platform to train surgeons and to verify control strategies. The following are the main components of the robotic neurosurgical robotic system.

- Mitsubishi PA10-7C robotic arm: Industrial robotic manipulator with seven degrees of freedom, manufactured by Mitsubishi Heavy Industries, Ltd.
- Joystick: Force feedback joystick produced by Logitech, Ltd.
- Force sensor: Force sensor produced by JR3, Inc., used to measure forces and torques exerted by the patient.

- Optical tracking system: Optical tracking device, i.e., kinetic sensor, produced by Microsoft, Inc.
- Client/server control architecture, used to monitor the neurosurgical robot system through the Internet by using the Winsock API.
- PC-based control and graphic user interface built by using Visual Basic.
- 3D model made by using VREP.

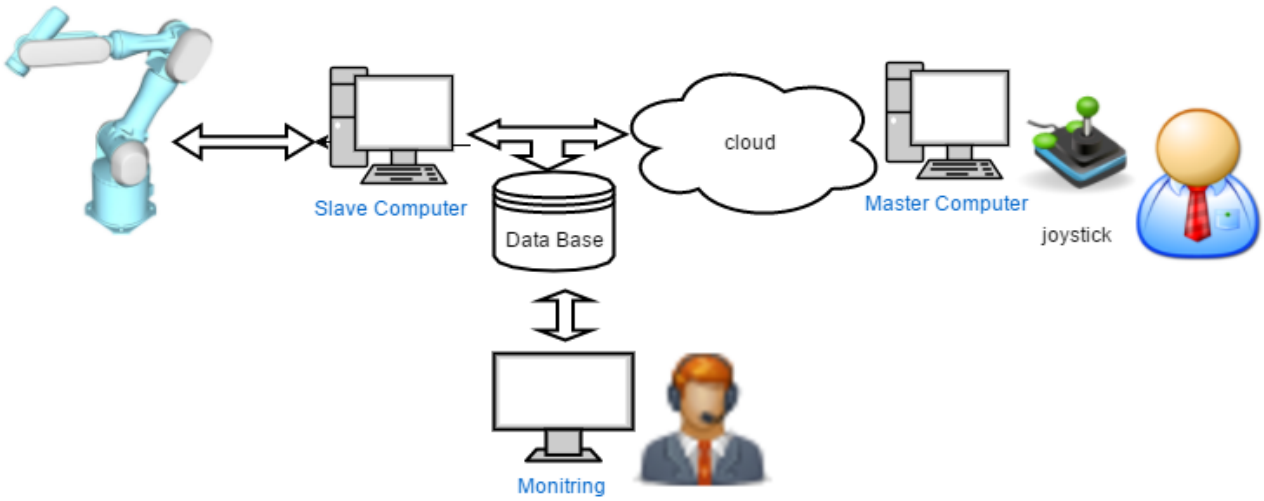

Figure 2.6 Architecture of the proposed system

The control system structure is based on server/client topology. The server program has two main functions. First, it must receive commands from the client and send feedback. Second, it must interpret the commands and execute them. The client part accepts user input and transmits user requests, and provides a user-friendly interface for remote users. The desired drilling position is input into the robot motion planner, and the drilling trajectory is calculated, taking the safety rules into account. Inverse kinematics is used to derive the joint control angles; these data are sent from the master computer to the client computer. Drilling depth and orientation controllers are used for controlling the robotic arm, and force and torque data are employed in the force controller.

2.8 ARCHITECTURE OF THE PROPOSED INTERACTIVE NAVIGATION AND CONTROLLER

The structure of the robot-assisted neurosurgical system is shown in Fig. 2.7. This demonstrates a prototype surgical robot system that uses widely available software and hardware: A 3D slicer as the planning interface, a PA107c manipulator as the robot hardware, MATLAB as the robot control

software, and VREP as a simulation environment. Figure 2.7 shows the architecture of the robot-assisted neurosurgical system. The following tasks are performed by each component:

1. The 3D slicer is used for medical image display, segmentation, procedure planning, pre- and intra-image registration, and tracker support.

2. OpenIGTLink is used as a transformation manager, while MATLAB-IGTL-Bridge is used to convert MATLAB topics to OpenIGTLink.

3. MATLAB is used to implement robot control algorithms, path planning, image libraries, sensor integration, and simulations.

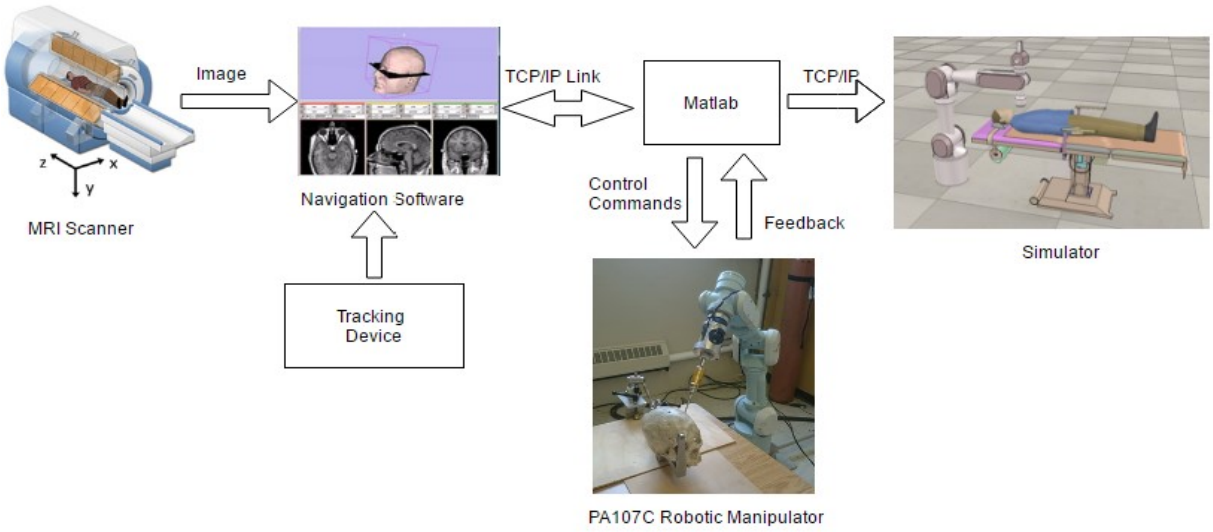

Figure 2.7 The Architecture of the Robotic Assisted neurosurgical system

The Mitsubishi PA10-7C robot arm has seven degrees of freedom, allowing it to make complex movements. The benefits of a redundant robot are its movement flexibility and dexterity, and the capability of adding additional constraints to avoid obstacles and joint limits. Figure 2.7 shows the experimental setup of the robot-assisted neurosurgical system. The robot arm moves the drill to the appropriate point on the patient's head, angles the drill perpendicular to the tangent of the skull arc, guides the drill through the skull along trajectories, and stops when the drill has perforated the skull.

CHAPTER 3 MODELING OF THE ROBOTIC MANIPULATOR

A mathematical model of a manipulator is an essential step in describing its motion and dynamic response when excited. Accurate modeling of a robot manipulator is critical in controlling tasks. Through careful modeling, the controller could be designed for the robotic manipulator to follow a desired skull drilling trajectory accurately. This chapter covers kinematic and dynamic model for serial manipulators that will serve as a foundation and a reference for the remainder of this work. The kinematic modeling of the Mitsubishi PA10-7C robot arm developed in section 3.1. Section 3.2 presents the Jacobian matrix computation. Section 3.3 describes the dynamics of serial robots. Finally, this chapter is summarized in section 3.5.

3.1 INTRODUCTION

A robot is an electromechanical device which performs automated tasks, either according to direct human supervision, a pre-defined program [105] or, a set of general guidelines, using artificial intelligence techniques. The robot is a combination of various physical and computational parts. A manipulator consists of a robot arm, and the gripper or end effector at the end of the arm [106]; A robot arm is known, manipulator. It is composed of a set of joints separated in space by the arm links. The joints are where the motion in the arm occurs. Basically, a robot arm consists of the parts: base, joints, links, and a gripper. The base is the basic part of the arm; it may be fixed or moving. Robots have been designed to serve various functions, and they, therefore, appear in a variety of forms. So as robotic systems grow in number and complexity, they are more widely used in many applications such as repetitive tasks, handling different types of loads, and the control applications.

Robots are used to perform repetitive tasks. Also; robot systems could be operated for long periods without any rest or need to take time off; for that reason; Many robots are being used in surgery systems because they are more accurate in the long term compared with human hands[107]. Also; robots can be used to handle much larger and complicated tasks than a human can such as heavy weights or dealing with tiny pieces. Robots are useful in Tele control applications; robots in many cases are reliable and more accurate than Human can do; when comparing them in an extended period. Robotic systems consist of two main parts: hardware part and software part. The hardware

part contains locomotion system, sensing system and communication system [108]. Software part contains control algorithms and reasoning methodology. In other words; these parts enable the robotic system to operate automatically with humanlike skill.

3.2 THE MITSUBISHI PA10-7C ROBOTIC MANIPULATOR

The PA10-7c robotic arm is an open kinematic structured chain with 7-Degrees of Freedom of Rigid links connected by revolute joints [109], made by Mitsubishi Heavy Industries, Ltd.(MHI), widely used for industry, research. The PA-10 is a redundant manipulator; The manipulator that has the joint degree of freedom more than the degree of freedom necessary for the aimed work is called a redundant manipulator [110]. When the degree of freedom of the robot increases, it comes to be able to do more complex work. PA10 robot is ideal for precise manipulation tasks due to the back drivability, accurate positioning capability, and zero backlash[111].

Table 3.1 The specifications of PA-107C robot [112]

Available weight of tip	10 kg
Arm body weight	35 kg
Controller weight	25 kg
Number of joints(DOF)	7 joints
Tip total speed	1550 mm/s
Arm length	950 mm
Position repeating accuracy	$\pm$0.1 mm
Operating Speed (degree/S)	Base axis (28.5); Elbow axis (57); Wrist axis (180)
Structure Dust-proof, drip-proof	Structure Dust-proof, drip-proof
System configuration	Layered-type open architecture

The challenge of modeling the robot due to that PA10 robot is a commercial product offering limited access to the low-level subsystems. The PA10 robot has four rotation axes and three pivot axes as illustrated in Fig. 3.1. The coordinate systems of the PA10-7C robot arm can be obtained and shown in Fig 3.1. The key specifications of the PA10 manipulator are that the full length of the manipulator is 1.345 m [113], the weight of the manipulator is 40 kg [114] while the payload weight is 10 kg, and the output torque of the end effector is 9.8 N.m. The maximum combined speed with all axes is 1.55 m/s [115]. It possesses seven degrees of freedom; thus, a so-called shoulder, elbow, and wrist are present. All joint encoder resolutions for the Mitsubishi robot are 0.05 degrees.

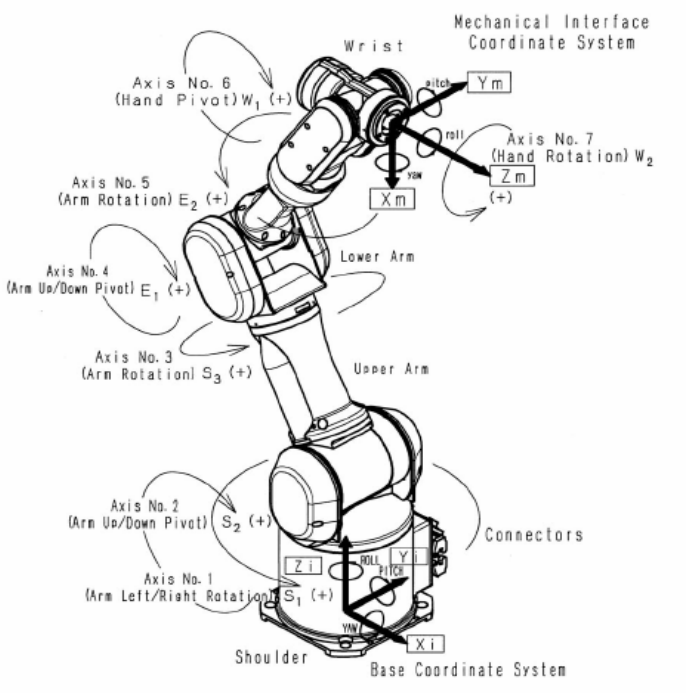

Figure 3.1 Coordinate systems of the Mitsubishi PA10-7CE robot [116]

PA-10 can transport up to 10 kg in weight and has about 1 m reach, as for industrial robots on the market. All sensors (resolvers), gears (harmonic drives) and electric brakes are mounted directly on motor shafts, and thus we can omit waste structural parts like coupling to miniaturize the PA-10 successfully. The servo drivers could be controlled via the serial communication (ARCNET). Indicated are the limits of the joint rotations and the maximum joint velocities as shown in Table 3.2.

Table 3.2 Joint description and physical Limits of PA10 robot manipulator

#	Joint	Axis	Mechanical limit (degree)	Servo limit (degree)	Software Limits (degree)	Joint Velocity Limits (degree/second)
1	Shoulder 1	Rotating	±180	±178	±177	±57
2	Shoulder 2	Pivoting	± 97	±95	±94	±57
3	Shoulder 3	Rotating	± 180	±175	±174	±114
4	Elbow 1	Pivoting	± 143	±138	±137	±114
5	Elbow 2	Rotating	± 270	±256	±255	±360
6	Wrist 1	Pivoting	± 180	±166	±165	±360
7	Wrist 2	Rotating	± 270	±256	±255	±360

3.3 DENAVIT HARDENBERG METHOD

Denavit Hardenberg (D-H) model parameters are used to describe the relationship between two consecutive frames of joints; Then the Kinematic model is derived to describe the robot motion on a fixed reference Cartesian frame by ignoring the forces and moments that cause movement of the structure. The basic idea behind DH convention is that it systematically assigns coordinate frames to the joint of each link; the main advantage of this method that it needs calculating four parameters for each joint which is better than calculating the six parameters like the geometric method.

Table 3.3 Mitsubishi PA10-7C Denavit-Hartenberg Parameters

Link	$\boldsymbol{a_i(m)}$	$\boldsymbol{d_i(m)}$	$\propto_i \boldsymbol{(rad)}$	$\boldsymbol{q_i(rad)}$
Link 1	0	0.317	$-\pi/2$	q_1
Link 2	0	0	$\pi/2$	q_2
Link 3	0	0.45	$-\pi/2$	q_3
Link 4	0	0	$\pi/2$	q_4
Link 5	0	0.480	$-\pi/2$	q_5
Link 6	0	0	$\pi/2$	q_6
Link 7	0	0.070	$-\pi/2$	q_7
Tooltip	0	0.2	0	0

The two common types of joints used in robot manipulators are revolute and prismatic joints. For a revolute joint, Qi is the angle of rotation (i), while for a prismatic joint, Qi is the joint displacement (di). There are two forms of Denavit-Hartenberg representation for Manipulator Kinematics as follows:

- Classical DH parameters: widely used since introduced in 1955 and used in textbooks such as Spong and Vidyasagar [117].
- Modified DH parameters: used in many textbooks such as Craig book [118]. These parameters are ai - link length, αi - link twist, di – link offset, and θi – joint angles shown in Table 3.3. The DH parameters solution steps:

The first thing that is necessary when using DH convention is first to assign the coordinate frames. After appropriately assigning the coordinate frames the next step is to define your DH parameters.

The third step is calculating the transformation matrix for each joint with respect the previous joint. Next, a detailed D-H algorithm step by step explanation is presented:

Algorithm. 1 Denavit-Hartenberg Algorithm

1. Numerate links beginning with 1 and Start from the bottom of link's chain and ending with n (Tooltip link). Where fixed base reference coordinate system will be numbered as link 0.
2. Find the type of each axis. Where prismatic joint will have an axis along which the displacement takes place. While revolving joint will have an axis turn around its own.
3. For n+1 of link 0 to 7; locate Zn+1 axis on the axis of articulation n.
4. Place the origin of the base reference coordinate system in any point of z0 axis. Axes z0 and y0 will be located so that they form a right-handed system with z0.

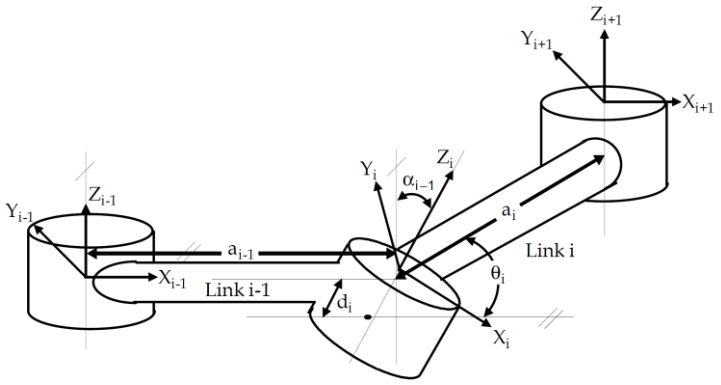

Figure 3.2 Denavit-Hartenberg Frame Assignments [119]

Table 3.3 shows DH parameter for the Mitsubishi robot. The notation is based on the "Introduction to Robotics, Mechanics, and Controls" book by John Craig. All units are SI. There are three types of twisting angles:

- Roll angle which is the rotation around Z-axis of the base coordinate
- Pitch angle which is the rotation around Y-axis of the base coordinate
- Yaw angle which is the rotation around X-axis of the base coordinate.

On the other hand; there are three types of Joint Coordinates:

- Roll coordinate: the same as the base coordinate.
- Pitch coordinate: 90 degrees diverted around X-axis of the base coordinate.
- Yaw coordinate: 90 degrees rotated around Y axis of the pitch coordinates

The homogeneous transformation matrix from Frame i to Frame $(i$ -1) is denoted by $A_i^{i-1}(q_i)$ using classical DH parameters given by:

$$A_i^{i-1}(q_i) = \begin{bmatrix} \cos(q_i) & -sin(q_i).cos(\alpha_i) & sin(q_i).sin(\alpha_i) & a_i cos(q_i) \\ sin(q_i) & cos(q_i).cos(\alpha_i) & -cos(q_i).cos(\alpha_i) & a_i sin(q_i) \\ 0 & sin(\alpha_i) & cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.3.1)$$

3.4 FORWARD KINEMATIC OF ROBOTIC MANIPULATORS

Robot kinematics is the study of the motion or kinematics of robots. In a kinematic analysis, the position, velocity, and acceleration of all the links are calculated without considering the forces that cause this motion. The relationship between motion and the associated forces and torques is studied in robot dynamics. In the kinematic analysis of manipulator position, there are two separate problems to solve: direct kinematics, and inverse kinematics. Direct kinematics involves solving the forward transformation equation to find the location of the hand regarding the angles and displacements between the links.

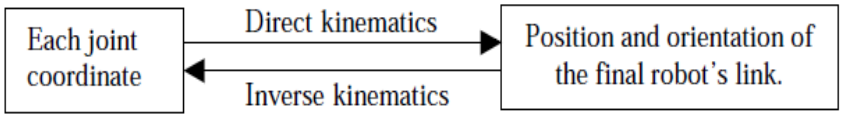

Figure 3.3 Direct and Inverse Kinematics

In forward kinematics, the length of each link and the angle of each joint is given, and the position of any point in the work volume of the robot is calculated. There are two ways for calculating the forwarded kinematics which is Analytic Geometric Method and Denavit Hardenberg Method. Analytic Geometric Method used for calculating the six parameters of each joint which is the joint angles and translation displacement. The forward kinematic analysis solved by using simple homogeneous matrices. D-H model parameters are used to describe the relationship between two consecutive frames of joints; Then the Kinematic model is derived to describe the robot motion on a fixed reference Cartesian frame by ignoring the forces and moments that cause movement of the structure. The end effector frame is transferred to the base frame as follows:

$$f_{fk}: \theta \rightarrow X, \theta \in \Re^7, X \in \Re^6 \quad (3.4.1)$$

The Forward geometric model of the robot PA107CE described by the equation:

$$T_8^0(q) = A_1^0.A_2^1.A_3^2.A_4^3.A_5^4.A_6^5.A_7^6.A_8^7 \quad (3.4.2)$$

$$T_8^0(q) = \begin{bmatrix} n & s & a & p \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.4.3)$$

The homogeneous transformation matrix T_8^0, specifies the position and orientation of the end-effector with respect to the base coordinate system, T_8^0 is the chain product of successive coordinate transformation matrices. where **n** ∈ *R*3 is the normal vector of the end-effector in the base coordinate system, **s** ∈ *R*3 is the sliding vector of the end-effector, **a** ∈ *R*3 is the approach vector of the end-effector, and $p = [x, y, z]^T$ is the position vector of the end-effector.

$$A_1^0(q_i) = \begin{bmatrix} c_1 & 0 & -s_1 & 0 \\ s_1 & 0 & c_1 & 0 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.4.4)$$

$$A_2^1(q_i) = \begin{bmatrix} c_2 & 0 & s_2 & 0 \\ s_2 & 0 & -c_2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.4.5)$$

$$A_3^2(q_i) = \begin{bmatrix} c_3 & 0 & -s_3 & 0 \\ s_3 & 0 & c_3 & 0 \\ 0 & -1 & 0 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.4.6)$$

$$A_4^3(q_i) = \begin{bmatrix} c_4 & 0 & s_4 & 0 \\ s_4 & 0 & -c_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.4.7)$$

$$A_5^4(q_i) = \begin{bmatrix} c_5 & 0 & -s_5 & 0 \\ s_5 & 0 & c_5 & 0 \\ 0 & -1 & 0 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.4.7)$$

$$A_6^5(q_i) = \begin{bmatrix} c_6 & 0 & s_6 & 0 \\ s_6 & 0 & -c_6 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.4.8)$$

$$A_7^6(q_i) = \begin{bmatrix} c_7 & -s_7 & 0 & 0 \\ s_7 & c_7 & 0 & 0 \\ 0 & 0 & 1 & d_7 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.4.9)$$

$$A_8^7(q_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_8 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4.10}$$

Where ci = cos θi , and si = sin θi. The end effecter frame is transferred to the base frame by applying same procedure. The end effector translation can be expressed as following:

$$\begin{aligned} X = [-([[(c_1c_2c_3 - s_1s_3)c_4 - c_1s_2s_4]c_5 - [-(-c_1c_2s_3 - s_1c_3)]s_5)s_6 \\ + ([c_1c_2c_3 - s_1s_3]s_4 - c_1s_2c_4)c_6]d_7 \\ + (-[c_1c_2c_3 - s_1s_3]s_4 - c_1s_2c_4]d_5 - c_1s_2d_3 \\ + (-[c_1c_2c_3 - s_1s_3]s_4 - c_1s_2c_4]d_5 - c_1s_2d_3 \end{aligned} \tag{3.4.11}$$

$$\begin{aligned} Y=[-([s_1c_2c_3 + c_1s_3)c_4 - s_1s_2s_4]c_5 - \big(-(-s_1c_2s_3 + c_1c_3)\big)s_5)s_6 \\ +(-[s_1c_2c_3 + c_1s_3]s_4 - s_1s_2c_4)c_6]d_7 \quad +[-(s_1c_2c_3 + c_1s_3)s_4 - \\ s_1s_2c_4]d_5 - s_1s_2d_3 \end{aligned} \tag{3.4.12}$$

$$\begin{aligned} Z = [-([s_2c_3c_4 + c_2s_4]c_5\text{-} \quad s_2s_3s_5)s_6 + (-s_2c_3s_4 + c_2c_4)c_6]d_7 + \\ [-s_2c_3s_4 + c_2c_4]d_5 + c_2d_3 \end{aligned} \tag{3.4.13}$$

The end effector rotation can be expressed as following:

$$\alpha = atan2(b, a) \tag{3.4.14}$$

$$\beta = atan2(-c, \frac{a}{\cos(\alpha)}) \tag{3.4.15}$$

$$\gamma = atan2(d, e) \tag{3..4.16}$$

Where:

$$\begin{aligned} a = [([(c_1c_2c_3 - s_1s_3)c_4 - c_1s_2s_4]c_5 + [-c_1c_2s_3 - s_1c_3]s_5)c_6 \\ -(-[-(c_1c_2c_3 - s_1s_3)s_4 - c_1s_2c_4])s_6]c_7 \\ +[([(s_1c_2c_3 + s_1s_3)c_4 - c_1s_2s_4)s_5 + (-c_1c_2s_3 - s_1c_3)c_5]s_7 \end{aligned} \tag{3.4.17}$$

$$\begin{aligned} b = [([(s_1s_2s_4]c_5 + [-s_1c_2s_3 + c_1c_3]s_5)c_6 \quad -(-(-(s_1c_2c_3 + c_1s_3)s_4 - \\ s_1s_2c_4))s_6]c_7 \quad +[-([s_1c_2c_3 + c_1s_3]c_4 - s_1s_2s_4)s_5 + (-s_1c_2s_3 + \\ c_1c_3)c_5]s_7 \end{aligned} \tag{3.4.18}$$

$$\begin{aligned} c = [([-s_2c_3c_4 - c_2s_4]c_5 + s_2s_3s_5)c_6 - (-[s_2c_3s_4 - c_2c_4])s_6]c_7 \\ +[-(-s_2c_3c_4 - c_2s_4)s_5 + s_2s_3c_5]s_7 \end{aligned} \tag{3.4.19}$$

$$\begin{aligned} d = -[([-s_2c_3c_4 - c_2s_4]c_5 + s_2s_3s_5)c_6 - (-[s_2c_3s_4 - c_2c_4])s_6]s_7 \\ +[-(-s_2c_3c_4 - c_2s_4)s_5 + s_2s_3c_5]c_7 \end{aligned} \tag{3.4.20}$$

$$e = -[-([-s_2c_3c_4 - c_2s_4]c_5 + s_2s_3s_5)s_6 - (-[s_2c_3s_4 - c_2c_4])c_6] \tag{3.4.21}$$

The kinematic model of a robot, can be represented by another homogeneous transformation matrices based on a calculation of the Modified D-H parameters.

$$ {}^{n-1}_{n}T = \begin{bmatrix} \cos\theta_n & -\sin\theta_n & 0 & a_{n-1} \\ \sin\theta_n\cos\alpha_{n-1} & \cos\theta_n\cos\alpha_{n-1} & -\sin\alpha_{n-1} & -d_n\sin\alpha_{n-1} \\ \sin\theta_n\sin\alpha_{n-1} & \cos\theta_n\sin\alpha_{n-1} & \cos\alpha_{n-1} & d_n\cos\alpha_{n-1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4.22} $$

The resulted homogeneous transformations are:

$$ A_1^0(q_i) = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4.23} $$

$$ A_2^1(q_i) = \begin{bmatrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_2 & -c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4.24} $$

$$ A_3^2(q_i) = \begin{bmatrix} c_3 & -s_3 & 0 & 0 \\ 0 & 0 & -1 & -d_3 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4.25} $$

$$ A_4^3(q_i) = \begin{bmatrix} c_4 & -s_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_4 & -c_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4.26} $$

$$ A_5^4(q_i) = \begin{bmatrix} c_5 & -s_5 & 0 & 0 \\ 0 & 0 & -1 & -d_5 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4.27} $$

$$ A_6^5(q_i) = \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_6 & -c_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4.28} $$

$$ A_7^6(q_i) = \begin{bmatrix} c_7 & -s_7 & 0 & 0 \\ 0 & 0 & -1 & -d_7 \\ s_7 & c_7 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4.29} $$

$$A_8^7(q_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4.30}$$

3.5 JACOBIAN MATRIX

Jacobian matrix is a matrix quantity used for velocity analysis; it consists all the first-order partial derivatives of a vector- or scalar-valued function on another vector. A Jacobian matrix specifies the mapping of angular velocities in joint space to velocities in Cartesian space. It also provides the joint torques needed for desired contact force and moment.

$$J = \begin{bmatrix} J_{11} & J_{12} & J_{13} & J_{14} & J_{15} & J_{16} & J_{17} \\ J_{21} & J_{22} & J_{23} & J_{24} & J_{25} & J_{26} & J_{27} \\ J_{31} & J_{32} & J_{33} & J_{34} & J_{35} & J_{36} & J_{37} \\ J_{41} & J_{42} & J_{43} & J_{44} & J_{45} & J_{46} & J_{47} \\ J_{51} & J_{52} & J_{53} & J_{54} & J_{55} & J_{56} & J_{57} \\ J_{61} & J_{62} & J_{63} & J_{64} & J_{65} & J_{66} & J_{67} \end{bmatrix} \tag{3.5.1}$$

The Jacobian matrix relates the joint velocity to end-effector velocity expressed in the end-effector reference frame. A differential motion can be represented by using a six-element vector with the following elements [dx dy dz drx dry drz]; where the [dx dy dz] are the differential translation, and the last three elements [drx dry drz] are a differential rotation. The order becomes unimportant when dealing with infinitesimal rotations, and the differential motion could be written as a combination of compounded transforms and rotation. Many control schemes require the inverse of the Jacobian. The Jacobian might have a singularity problem which should be tested. The manipulator's Jacobian matrix relates differential joint coordinate motion to differential Cartesian motion; for any number of joint manipulator, we could calculate the manipulator Jacobian matrix which is used in many manipulator control schemes. The Jacobian of the PA10 manipulator [120] can be expressed as shown in Equation 3.5.2. Please refer to Appendix 1. for individual parameters.

$$J = \begin{bmatrix} -s1q114 - c1q234 & c1p23 & s1s2p33 - c2p32 \\ c1q114 - s1q234 & s1p23 & c2p31 - c1s2p33 \\ 0 & -s1p22 - c1p21 & c1s2p32 - s1s2p31 \\ 0 & -s1 & C_1C_2 \\ 0 & c_1 & S_1S_2 \\ 1 & 0 & c2 \end{bmatrix} +$$

$$
+\begin{matrix}
q423p43 - q433p42 & -q422p53 + q432p52 \\
q433p41 - q413p43 & -q432p51 + q412p53 \\
q413p42 - q423p41 & -q412p52 + q422p51 \\
-c1c2c3 - s1c3 & (c1c2c3 - s1s3)s4 + c1s2c4 \\
-s1c2s3 + c1c3 & -(s1c2c3 + c1s3)s4 - s1s2c4, \\
s2s3 & -s2c3s4 - c2c4
\end{matrix}
\tag{3.5.2}
$$

$$
+\begin{bmatrix}
q522p63 - q532p62 & q522p63 - q532p62 & 0 \\
q522p63 - q532p62 & q522p63 - q532p62 & 0 \\
q512p62 - q522p61 & q512p62 - q522p61 & 0 \\
-q411s5 + q413c5 & -q411s5 + q413c5 & -q612 \\
-q421s5 + q423c5 & -q421s5 + q423c5 & -q622 \\
-q421s5 + q423c5 & -q431s5 + q433c5 & -q632
\end{bmatrix}
$$

Whitney method [121] is an alternative strategy to computing a Cartesian trajectory and solving the inverse kinematics. Whitney method is used to resolve the rate of motion where dX/dt is the desired Cartesian velocity, and dQ/dt is the required joint velocity to achieve this. However, this makes a problem when dealing with difficulty at a manipulator singularity where the Jacobian is singular. When two of the robot manipulators wrist joints area aligned this resulting in the loss of one degree of freedom and this called singularity problem; where at a manipulator singularity or degeneracy the Jacobian becomes singular. The singularity is revealed by the ranking of the Jacobian matrix. In the case of robotic manipulators which have six degrees of freedom or more with redundant manipulators, the redundancy increases the flexibility and the generality of work and minimize the singularity problem effect. However, on the other hand; six degrees of freedom makes the computing of the joint motion for the robotic manipulator to be not straightforward.

Many approaches have been suggested based on the pseudo-inverse of the Jacobian which in this case is not square or singular value decomposition of the Jacobian. Singularity calculation leads to understanding more about the manipulability of the robotic manipulator; which describes how `well-conditioned' the manipulator is for making certain motions and is referred to as `manipulability.' So, when there is no singularity in the robotic arm, we could say that the manipulability of the robot is higher. There are many scalar manipulability measures have been proposed such as Yoshikawa method which is based purely on kinematic parameters of the manipulator [122]. Moreover, there is another by Asada [123] takes into account the inertia of the manipulator which affects the acceleration achievable in different directions. This measure varies

from 0 to 1, where 1 indicates uniformity of acceleration in all directions. Both measures would show that this particular pose is not well conditioned.

3.6 INVERSES KINEMATIC OF ROBOTIC MANIPULATOR

Inverse kinematics involves solving the inverse transformation equation to find the relationships between the links of the manipulator from the location of the hand in space [124]. In inverse kinematics, the length of each link and position of the point in work volume is given, and then the angle of each joint is calculated. The inverse kinematics is so hard to solve, and it will be harder if we increase the degrees of freedom. There is a different method to solve the inverse kinematics. The analytic method and Jacobian method are well-known [125]. This problem is much more complex than forwarding kinematics. Two popular approaches are used which are Inverse Kinematic solving by Algebraic Method (Closed Form Solution), and Inverse Kinematic solving by Geometric Method. The motion of the edge of the robot arm is used as a reference to a controller that can calculate the link angles to achieve the reference motion of the edge. Therefore, inverse kinematics is important to derive the unknown link motions from the known edge motion. Closed Form Solution is a numerical solution, and the spatial geometry of manipulator is broken down into several plane problems which are easily solvable with using iterative process and requires more processing power [126].

Inverse Kinematic solving by Algebraic Method is an analytical solution and requires solving a set of non-linear equations derived from frame transformations. This returns all possible solutions, and according to Pieper's Theorem [127], there is existence for a solution for any 6 degrees of freedom robot with a spherical wrist. The closed form solutions are preferable than the numerical solutions because inverse kinematic equations must be solved at a rapid rate. However; the inverse kinematic equations, in general, have multiple solutions. So when the solution is derived faster; this allows developing rules for choosing a particular solution among several solutions. Please refer to Appendix 4. for Matlab code for calculating the inverse kinematics of the PA10 robot.

3.7 DYNAMICS OF PA10 ROBOT

Robot Dynamics is the mathematical modeling of robot motion in with the associated forces and torques. The dynamic equations of manipulator motion are a set of equations describing the

dynamic behavior of the robotic manipulator. There are various methods to formulate robot dynamics, such as the Lagrange-Euler, the Newton-Euler, the recursive Lagrange-Euler.

Table 3.4 Joint description and dynamic Limits of PA10 Robotic Manipulator

Joint	Nominal Torque (N-m)	Max Torque (N-M)	Torque Limit (N-M)	Centers of Mass			Joint Limit (degree)	
				X (m)	Y (m)	Z (m)	Lower	Upper
S 1	4.64	232	158	0	0	-0.01	-180	180
S 2	4.64	232	158	0	-0.2	0.0	-97	97
S 3	2	100	68	0	0	-0.035	-180	180
E 1	2	100	68	0	-0.115	0.0	-143	143
E 2	0.29	14.5	17	0	0	-0.084	-270	270
W 1	0.29	14.5	17	0	-0.042	0.0	-180	180
W 2	0.29	14.5	17	0	0	0.022	-270	270

Forward and Inverse Dynamics are solved using manipulator dynamics. Direct dynamics in which the equations of motion are integrated to determine the generalized coordinate response to applied generalized forces. While Inverse dynamics in which the manipulator's equations of motion are solved for given motion to determine the generalized forces. In order to determine the Euler-Lagrange equations in a specific situation, one has to form the Lagrangian of the system, which is the difference between the kinetic energy and the potential energy. When PA10 robot is performing drilling; it only uses the last three degrees of freedom. Matlab code is written for calculating the torques of the last three DOF and used to control the position of three degrees of freedom robotic manipulator, and it is used to find the torques of three joints for movement of the robotic manipulator end-effector from the start location to the goal location.

The position control system contains a PID controller where the PID controller is used to control the torque and enhance the movement of the robotic manipulator; where the cubic function is used as velocity trajectory for the robotic manipulator. The resulted dynamic model is used for simulation robot dynamics, the design of robot controller. Table 3.4 describes the kinematic, dynamic, and compliance properties as well as constraints of the PA10 robotic manipulator. Table 3.5 shows the link Center of gravity for the Mitsubishi robot. Each row specifies the coordinates of the C.G. of the link (in its local frame) and its mass. It is necessary to know the mass parameters

of each link of the PA10 robot; to model correctly the dynamic; Parameters are Link mass; Inertia matrix; Center of gravity; Joint friction coefficient.

Table 3.5 Mass parameters of the PA10 robot model [116][112][128]

Link	Mass (Kg)	Center of Mass (m)	Inertia Matrix			Center of gravity	Offset Distance (m)
1	9.78	0	0.110697 0.000005 0.000345	0.000005 0.084268 0.000518	0.000345 0.000518 0.054080	0.0 0.0 -0.166	0.115
2	8.41	0.06325	0.177079 0.000257 0.000000	0.000257 0.018440 0.000000	0.000000 0.000000 0.173903	0.0 -0.0632 0.0	0
3	3.51	0.08944	0.032979 0.000000 0.000002	0.000000 0.017031 0.000027	0.000002 0.000027 0.022417	0.0 0.0 -0.112	0.45
4	4.31	0.04609	0.051762 -0.000577 0.000000	-0.000577 0.006329 0.000000	0.000000 0.000000 0.051407	0.0 -0.046 0.0	0
5	3.45	0.1647	0.076687 0.000000 0.000960	0.000000 0.077392 0.000000	0.000960 0.000000 0.003233	0.0 -0.0632 0.0	0.5
6	1.46	-0.03	0.012500 0.000000 0.000000	0.000000 0.001431 0.000000	0.000000 0.000000 0.012500	0.0 0.003 0.0	0
7	1.46	-0.029	0.001575 0.000000 0.000000	0.000000 0.001575 0.000000	0.000000 0.000000 0.000131	0.0 0.0 0.0	0.08

Link Inertia is shown in Table 3.5. Each three rows specify the inertia tensor of a link. The first three rows are for link 1, the second three rows for link 2, etc. All units are SI. Equation 3.7.1 shows joint compliance matrix Q_j for the Mitsubishi robot [129]. The units are rad/(N-m).

$$Q_j = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.000037 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0000588 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0000909 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.000227 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0005 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.000556 \end{bmatrix} \quad (3.7.1)$$

The knowledge of the full dynamic model of the robot is very important to be used as a part of control system simulation and algorithm development. The PA107c robotic manipulator is composed of *7* joints. The dynamic equation can be described by:

$$\tau = M(q)\ddot{q} + C(q,\dot{q}) + G(q) + F(q,\dot{q}) \tag{3.7.2}$$

Where:

$$q = [\,\theta_1 \;\; \theta_2 \;\; \theta_3 \;\; \theta_4 \;\; \theta_5 \;\; \theta_6 \;\; \theta_7\,]^T, \dot{q} = \frac{dq}{dt}, \ddot{q} = \frac{d^2q}{dt^2} \tag{3.7.3}$$

The variable $q_i(t) \in R^m$ denotes the joint angular position, $M_i(q_i) \in R^{m\times m}$ is inertia matrix, $G_i(q_i) \in R^m$ is the gravitational force vector, $\tau_i(q_i) \in R^m$ denotes the input torque vector, $F(q,\dot{q})$represent Friction terms. The Inertia matrix $M_i(q_i)$ is symmetric and uniformly positive definite for all $q_i(t) \in R^m$, The resultant equations of the dynamic model are a set of second order, coupled nonlinear differential equations. So; based on [130] method; PA10 robot dynamics can be described as following:

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_\iota} - \frac{\partial L}{\partial q_i} = \tau_i\,, i = 1,\dots,7 \tag{3.7.4}$$

Where q_i is the joint angles, $\dot{q}_i$ is the joint velocities, τ_i is the joint torques and L is defined by Lagrangian mechanics as following:

$$L = T - U = \sum_{i=1}^{7} T_i - \sum_{i=1}^{7} U_i \tag{3.7.5}$$

Where T_i is the kinetic energy and U_i is the potential energy of link i. The kinetic energy is described as following:

$$\begin{aligned} T_i = {} & \frac{1}{2} m_i \dot{\boldsymbol{p}}_i^{iT} \dot{\boldsymbol{p}}_i^i + \dot{\boldsymbol{p}}_i^{iT} S(w_i^i) m_i \boldsymbol{r}_{i,C_i}^i, + \frac{1}{2} w_i^{iT} \hat{\boldsymbol{I}}_i^i \boldsymbol{w}_i^i + \\ & k_{r,i+1}\dot{q}_{i+1} I_{m_{i+1}} Z_{m_{i+1}}^{iT} w_i^i + \frac{1}{2} k^2{}_{r,i+1} \dot{q}^2{}_{i+1} I_{m_{i+1}} \end{aligned} \tag{3.7.6}$$

The Lagrangian equation could be represented as following

$$\begin{aligned} L = \sum_{i=1}^{7} & \frac{1}{2} m_i \dot{\boldsymbol{p}}_i^{iT} \dot{\boldsymbol{p}}_i^i + \dot{\boldsymbol{p}}_i^{iT} S(w_i^i) m_i \boldsymbol{r}_{i,C_i}^i, + \frac{1}{2} w_i^{iT} \hat{\boldsymbol{I}}_i^i \boldsymbol{w}_i^i + k_{r,i+1}\dot{q}_{i+1} I_{m_{i+1}} Z_{m_{i+1}}^{iT} w_i^i \\ & + \frac{1}{2} k^2{}_{r,i+1} \dot{q}^2{}_{i+1} I_{m_{i+1}} - \sum_{i=1}^{7} -g_0^{iT} (m_i p_i^i + m_i r_{i,C_i}^i) \end{aligned} \tag{3.7.7}$$

Where:

m_i: is the overall mass of the desired link i

$\boldsymbol{p}_i^i$: is the position vector of the link i referred to frame i

$\dot{\boldsymbol{p}}_i^i$: is the linear velocity of the link i referred to frame i

g_0^i: is the gravity acceleration vector of the link i with respect to frame i

w_i^i: is the angular velocity of the link i with respect to frame i

r_{i,C_i}^i: is the vector with start the reference frame i and the center of mass of the augmented link i , refereed to frame i

$$r_{i,C_i}^i = [l_{c_ix} \quad l_{c_iy} \quad l_{c_iz}]^T \tag{3.7.8}$$

$k_{r,i+1}$: is the gear reduction ratio of motor $i+1$

$I_{m_{i+1}}$: is the inertia tensor of the rotor $i+1$ relative to its center of mass

$Z_{m_{i+1}}^i$: is the unit vector along the rotor axis $i+1$ refereed to frame i

$S(.)$: is a matrix operator defined for a vector $r = [r_x \quad r_y \quad r_z]^T$

$$S(r) = \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix} \tag{3.7.9}$$

$\widehat{I}_i^i$: is the inertia tensor of the link i relative to the origin i

$$\widehat{I}_i^i = \begin{bmatrix} \hat{I}_{ixx} & -\hat{I}_{ixy} & -\hat{I}_{ixz} \\ -\hat{I}_{ixy} & \hat{I}_{iyy} & -\hat{I}_{iyz} \\ -\hat{I}_{ixz} & -\hat{I}_{iyz} & \hat{I}_{izz} \end{bmatrix} \tag{3.7.10}$$

The Lagrange equation of the PA107c robotic manipulator could be represented as follows:

$$L = \sum_{i=1}^{7} (\beta_{T_i}^T - \beta_{U_i}^T)\pi_i \tag{3.7.11}$$

Where $\beta_{T_i}^T$, $\beta_{U_i}^T$ are variables dependent on joint positions and joint velocities. π is 11×1 vector of dynamic parameters defined by:

$$\pi_i = \begin{bmatrix} m_i & m_i l_{c_ix} & m_i l_{c_iy} & m_i l_{c_iz} & \hat{I}_{ixx} & \hat{I}_{ixy} & \hat{I}_{ixz} & \hat{I}_{iyy} & \hat{I}_{iyz} & \hat{I}_{izz} & \hat{I}_{m_{i+1}} \end{bmatrix}^T \tag{3.7.12}$$

Where π_i is a p×1 vector of constant parameters. Please refer to Appendix 2. for complete modelling parameters. Inverse dynamics is used to calculate the required velocity and acceleration of the joints in the robotic manipulators. There are many mathematical methods to calculate the inverse dynamics such as the recursive Newton-Euler formulation which is an efficient matrix oriented Algorithm. The inverse dynamics calculations require identifying the kinematic parameters and the inertial and mass parameters of each link.

3.8 SUMMARY

Kinematic modeling of PA10-7C robot arm involves forward kinematics and inverse kinematics. Forward kinematics is a mapping from the joint space to the operational space; while Inverse kinematics is a mapping from the operational space to the joint space. In this chapter, the forward kinematics of PA10-7C robot arm is calculated efficiently based on the D-H model. D-H model parameters are used to describe the relationship between two consecutive frames of joints; Then the Kinematic model is derived to describe the robot motion on a fixed reference Cartesian frame by ignoring the forces and moments that cause motion of the structure. Jacobian matrix for PA10-7C robot arm is derived to describe the relationship between joint angular velocities in the joint space and the end effector's velocities in Cartesian space.

CHAPTER 4 MOTION PLANNING AND ROBOT CONTROL SCHEMES

Robot control is the spine of robotics. It consists of studying how to make a robot manipulator do what it is desired to do automatically; hence, it includes in designing robot controllers. Typically, these take the form of an equation or an algorithm which is realized via specialized computer programs. Then, controllers form part of the so-called robot control system which is physically constituted of a computer, a data acquisition unit, actuators, in this chapter; different types of motion planning algorithms studied. Various control theory for neurosurgical manipulators reviewed in this chapter.

4.1 ROBOTIC ARM CONTROLLER

A block diagram model of the robot arm control is shown in Fig 4.1. The feedback sensors are located on the robot joints which send feedback information to the controller on the robot's position.

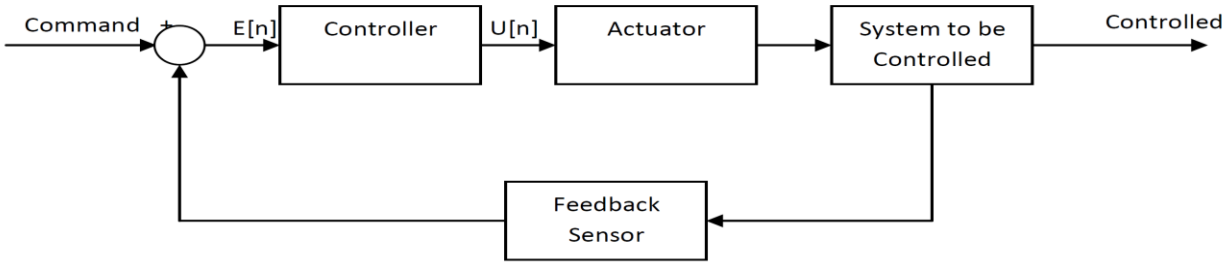

Figure 4.1 Block Diagram Model of the Robot Arm Control

Robot position control is achieved by linear control and non-linear control systems which get feedback from position and velocity sensors. Robot force control is required to create sufficient torque in the joints for a desired amount of force. In many cases; reaction forces are not present in all directions so that some directions will have position control, and others will have force control which results in hybrid control.

4.1.1 MOTION CONTROL OF THE PA10-7C ROBOT

The position and orientation of the rigid body are defined easily when using the Cartesian space trajectory generation. The user specifies the desired end-effector path, the traveling time, and tool orientations along the path. Cartesian space deals easily when there is an obstacle in the path, but it is much more complex than joint space technique. Some common techniques for Cartesian space

trajectory planning such as a parametric description of a path; straight-line path; a circular path; position planning and orientation planning technique. Optimizing the control of drilling depth while drilling skull is extremely hard, and there is a chance of brain damage due to the over drilling. However; in many cases; even after completing the drilling work, it is very difficult to measure the depth; especially for thin holes. Therefore, an automatic skull drilling system is needed to perform the function of drilling. In the first phase, the drill will follow the desired trajectory from its initial position to a certain position under position control without any interactions with the environment; Inverse kinematics is used to calculate the proper joint angles taken into account the obstacle avoidance, and robot constraints such as singularity and joint limits.

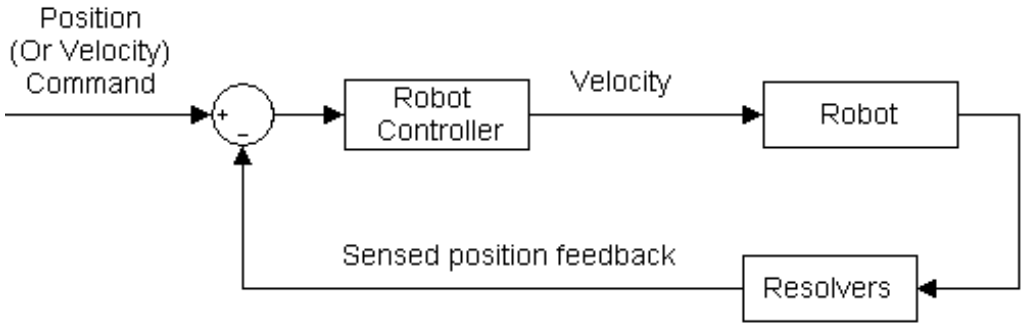

Figure 4.2 Existing PA10-7C Robot Control

Position control involves only controlling the motion (position, velocity, and acceleration) of a manipulator, i.e. determining a set of command signals that will move the robotic manipulator along the desired motion drilling trajectory. Before the techniques are discussed, it is important to understand the control inputs and outputs. The PA107c robot controller responds to position and velocity commands. The target set points are X, Y, and Z positions in the robot's base frame calculated using the joint angles. The robot controller calculates velocities from rates of change of positions and sends torque motor commands to servo-controllers that move the robot joints as shown in Fig. 4.2.

There are two types of motion controllers based on joint space and end-effector Cartesian space; The joint space motion algorithms are not implemented because it is not suitable for the neurosurgical tasks. Cartesian-Based Control applies the desired trajectory to the robot end effector regarding time histories of positions, velocities, and accelerations. Joint- based control schemes use these desired trajectories to the joint inputs. The inverse kinematics performs the

trajectory conversion. The errors in Cartesian space calculated as follows.

$$e_x(t) = x_d(t) - x(t) \quad (4.1.1.1)$$

Where: $x_d(t)$ is the desired Cartesian trajectory and x(t) is the actual robot end effector Cartesian space. The procedure of the operation of the proposed neurosurgical system consists of multiple controlled variables which are: Drilling Orientation Controller; Drilling Depth Controller; Drilling Velocity Controller; and Drilling Force Controller.

4.1.2 DRILLING ORIENTATION CONTROL

In the second phase, the drill will be angled along the normal to the tangent of the skull, that is, under orientation control without any interactions with the environment. Perpendicular drilling aims to have most of the drilling force in Z direction

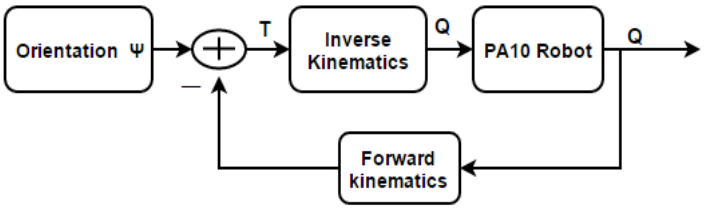

Figure 4.3 Orientation Control for Skull Drilling System

Fig 4.3 shows the block diagram for Orientation Control used for Skull Drilling System to make robot tooltip perpendicular with the skull surface. Angle interpolation done for the target direction and the velocity command interpolated to form a letter "S" shape.

4.1.3 DRILLING DEPTH CONTROL

In the third phase, drilling depth control will be applied to monitor the drill to move toward the skull along the normal to the tangent of the skull. The control of the drill's position and orientation is combined Orientation and Depth controller. The control block diagram for the first phase and the second step are shown in Fig. 4.3 and Fig. 4.4 respectively.

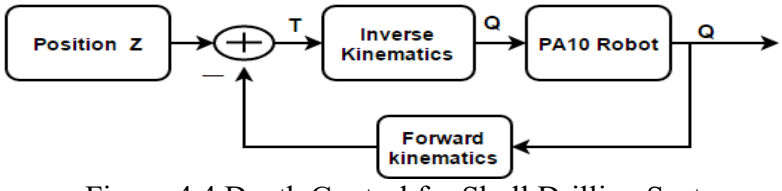

Figure 4.4 Depth Control for Skull Drilling System

4.1.4 DRILLING-DEPTH AND VELOCITY CONTROLLER

The robot system runs a cascaded controller for controlling skull drilling depth and velocity by implementing two loops; the inner loop is velocity loop while the outer loop is Drilling depth controller as shown in Figure 4.5. The velocity signal used in the velocity loop is the position signal differentiated and low-pass filtered.

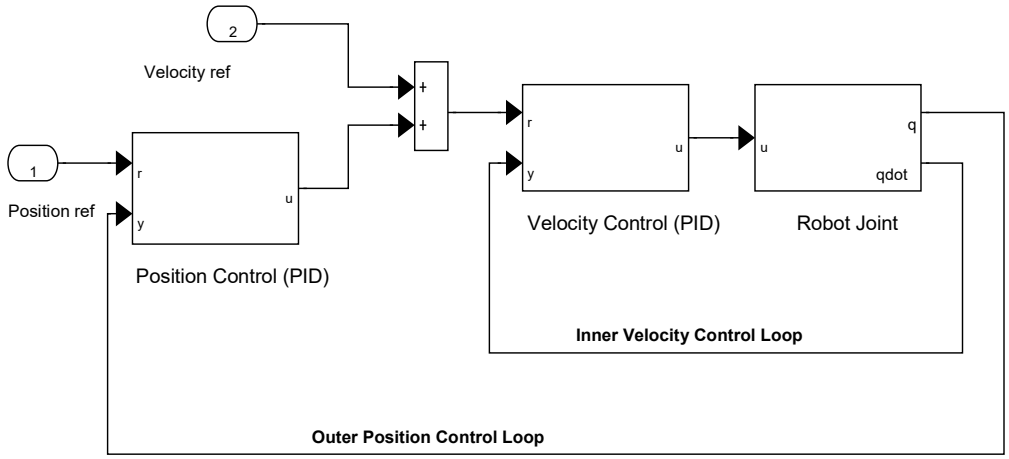

Figure 4.5 The structure of the Drilling-Depth and Velocity Controller

4.1.5 DRILLING FORCE CONTROLLER

Force signal is analyzed to determine the stage of operation. When the drill is contacted with the object initially, the changing of force is large, and the drill is activated. During drilling, the force will be maintained at a certain value by Force Controller as shown in Fig 4.6. The drill automatically stops when the drilling is finished. Otherwise, the brain will be damaged. The drill moves forward or backward according to the force signal applied to the drilling tool. However, for safety consideration when the force exceeds force threshold; the robot will not move the drill forward again.

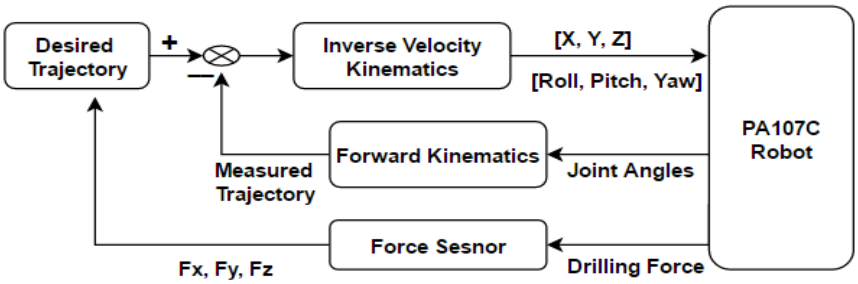

Figure 4.6 Force Control for Skull Drilling System

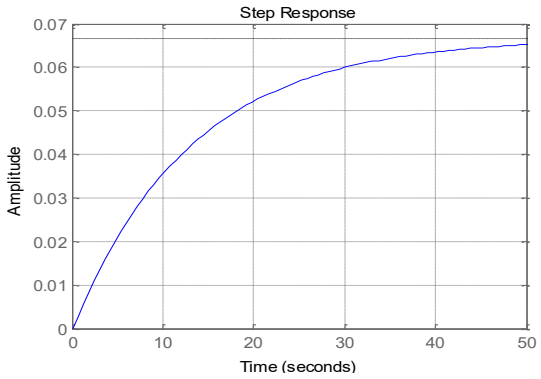

Figure 4.7 Step response of system output

Fig 4.7 shows the simulation of skull drilling depth represented by step response; where reference setpoint is selected to be 0.065 meter and drilling process is done in 47 seconds. However; Drilling force is not linear due to the nonlinear structure of skull layers.

4.2 TRAJECTORY GENERATION AND CONTROL

The trajectory is the path followed by the manipulator while moving from point A to B. The joint angles can describe it as a function of time. To smoothen this motion through the via points, Spline curves are utilized, and position and its first two derivatives must be continuous. Cubic polynomials satisfy this condition. Higher order polynomials and linear functions with parabolic blends are also compliant. Trajectory generation is done by defining the type of motion or trajectory; selecting the trajectory technique and computing the trajectory. Time sequence values are attained by the functions generated from the trajectory planning technique are calculated at a particular path update or sampling rate. Path update rate in real time lies between 20Hz to 200Hz in typical industrial manipulator systems. There are two types of trajectories used in robotic manipulators which are point-to-point motion and Continuous path motion. Point to point-to-point motion used in pick and place operation, the task is specified as initial and final end-effector location. There is no particular specification about the intermediate locations of end-effector. Continuous path motion used to describe a specific path between points is required to be traced by the end effector in Cartesian space. However; in particular cases; more than two points of the path are specified, this is done to ensure a better monitoring of executed trajectory in the event of application similar to the point-to-point motion. Joint space techniques are employed in which motion planning is done at the joint level. The Joint space planning scheme generates a time-dependent function of all joint variable and their first two derivatives to describe their motion of

manipulator. The next step is to find smooth function q(t) for each n-joints of an n-DOF manipulator. For that, we need traveling time, initial and final locations. Usually; cubic polynomial and linear function with parabolic blend are used in joint space trajectories. A cubic polynomial in which cubic connects the points of each joint smoothly. While linear function with the parabolic blend in which each segment between two successive points contains a linear function with parabolic bends near the path points. Cartesian space techniques are used for an application requiring continuous path motion; The Cartesian space planning scheme provides a time history of the location, velocity, acceleration of the end-effector on the base. The corresponding joint variables and their derivatives are computed, using inverse kinematics. The generated trajectory has a constant acceleration up to certain time and then constant velocity for the period and then the trajectory goes to a constant retardation to make velocity from v to zero. There will be a trapezoidal velocity profile. The triangular velocity profile could be used for the small value of the joint parameter. The travel time in parabolic blends is chosen and the total time of motion is fixed to calculate the value of acceleration; else if the value of acceleration is fixed, then the value of acceleration can be calculated. To make a robot perform a drilling task, the motion planner is designed to generate a smooth motion, as described in detail in [131]. A path is a geometric representation, initially in Cartesian space, that is then converted to joint space by using inverse kinematics, as described in [84].

4.2.1 FEEDFORWARD TRAJECTORY PLANNING

Feedforward control can be used very successfully to improve a control loop's response to disturbances without having to wait for a deviation in the process variable. This enables a feedforward controller to quickly and directly cancel out the effect of a disturbance. To do this, a feedforward controller produces its control action based on a measurement of the disturbance. In a feed-forward system, the control variable adjustment relies on knowledge about the process and knowledge about or measurements of the disturbances. Feedforward and feedback control is often combined with cascade control, to ensure that their control actions manipulate the physical process linearly, eliminating control valve nonlinearities and mechanical problems. Feedforward Controllers are based on system knowledge; basically, a mathematical model of the system is required, and Feedforward controller is combined with feedback control to optimize performance. Feedforward Motion Control is an algorithm for calculating point to point trajectory. These

trajectories are time-optimal in the most relevant cases. Based on Lambrechts's technique [132] different types of feedforward trajectory planner are implemented; which are:

- First Order Feedforward Control of Skull Drilling "Speed limited Trajectory."
- Second Order Feedforward Control of Skull Drilling "Acceleration limited Trajectory."
- Third Order Feedforward Control of Skull Drilling" Jerk limited Trajectory."
- Fourth Order Feedforward Control of Skull Drilling "Snap Limited Trajectory Control

Velocity is bounded in the first order feedforward motion planner; while acceleration is bounded in the second order feedforward motion planner; The specifics of motion planning based on simple rigid-body analysis as shown in Fig. 4.6 where x is the position, m is equivalent mass or inertia, b is viscous damping, F is actuator force, and F is feedforward force.

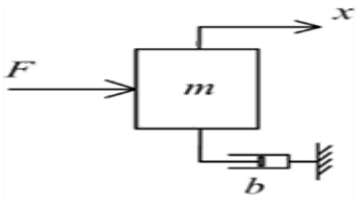

Figure 4.6 One degree of Feedforward Control of Skull Drilling

$$m\ddot{x} + b\dot{x} = F_{ff} \tag{4.2.1.1}$$

$$F_{ff} = ma + bv \tag{4.2.1.2}$$

F_{ff} is the feedforward force; $\ddot{x}$ represent the acceleration a component; and $\dot{x}$ represent the velocity v component. The shortest time path for the desired displacement distance (x) can be calculated as:

$$x = at^2 \implies t_a = \sqrt{\frac{x}{a}} \implies t_x = 2t_a \tag{4.2.1.3}$$

Where t_a and t_x are time for acceleration and position respectively. Maximum velocity could be calculated as following:

$$v = a.t_a \tag{4.2.1.4}$$

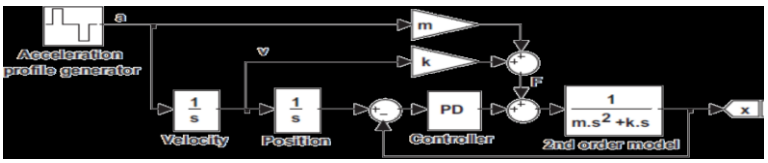

Figure 4.7 One degree of Feedforward Control of Skull Drilling

The resulted trajectory is visible and optimized in execution time. Trajectory ends at the desired end position. A derivative of acceleration signal (Jerk signal) is limited in the third order feedforward motion planner; where trajectory planned over a distance at bounded velocity, acceleration, and jerk; The bound on jerk is related to 'rise time' of the system which implemented within the robot electromechanical capabilities.

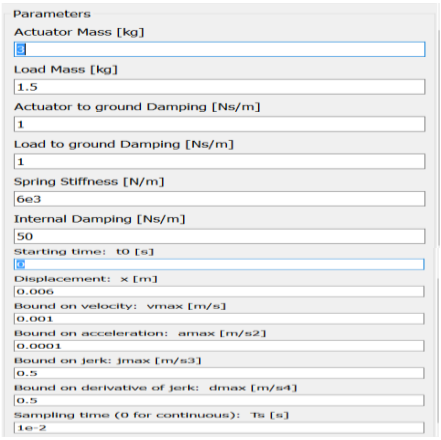

Figure 4.8 Design Parameters of Motion Planners

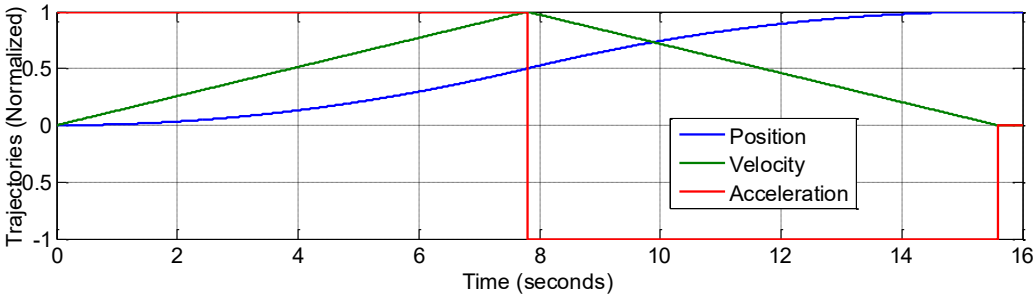

Figure 4.9 Second Order Trajectory Determination

There are eight-time instances at which the jerk could be constructed as shown in Fig 4.10.

$$t_{\bar{j}} = t_1 - t_0 = t_3 - t_2 = t_5 - t_4 = t_7 - t_6 \quad (4.2.1.5)$$

$$t_{\bar{a}} = t_2 - t_1 = t_6 - t_5 \quad (4.2.1.6)$$

$$t_{\bar{v}} = t_4 - t_3 \quad (4.2.1.7)$$

$$a(t) = j_0 t + a_0 \quad (4.2.1.8)$$

$$v(t) = \frac{1}{2} j_0 t^2 + a_0 t + v_0 \quad (4.2.1.9)$$

$$x(t) = \frac{1}{6} j_0 t^3 + \frac{1}{2} a_0 t^2 + v_0 t + x_0 \quad (4.2.1.10)$$

The third-order motion trajectories for acceleration, velocity, and position can be expressed as follows:

$$a(t_2) = a(t_1) = \bar{j} t_{\bar{j}} \quad (4.2.1.11)$$

$$v(t_2) = v(t_1) = \frac{1}{2} \bar{j} t_{\bar{j}}^2 \quad (4.2.1.12)$$

$$x(t_2) = x(t_1) = \frac{1}{6} \bar{j} t_{\bar{j}}^3 \quad (4.2.1.13)$$

Bounds on acceleration and velocity are not violated. In the second part of the jerk profile; $\bar{j} = S$ which represent snap signal and can be expressed as following:

$$a(t_4) = a(t_3) = -\bar{j} t_{\bar{j}} + a(t_2) \quad (4.2.1.14)$$

$$v(t_4) = v(t_3) = -\frac{1}{2} \bar{j} t^2_{\bar{j}} + a(t_2) t_{\bar{j}} + v(t_2) \quad (4.2.1.15)$$

$$x(t_4) = x(t_3) = -\frac{1}{6} \bar{j} t^3_{\bar{j}} + a(t_2) t^2_{\bar{j}} + v(t_2) t_{\bar{j}} + x(t_2) \quad (4.2.1.16)$$

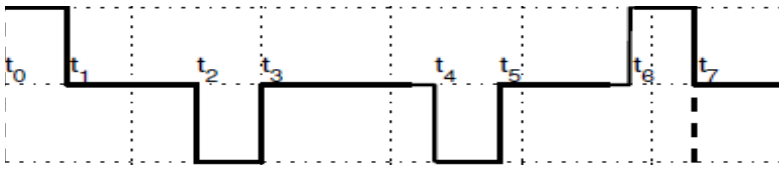

Figure 4.10 Jerk Trajectory for Third Order Motion Planner

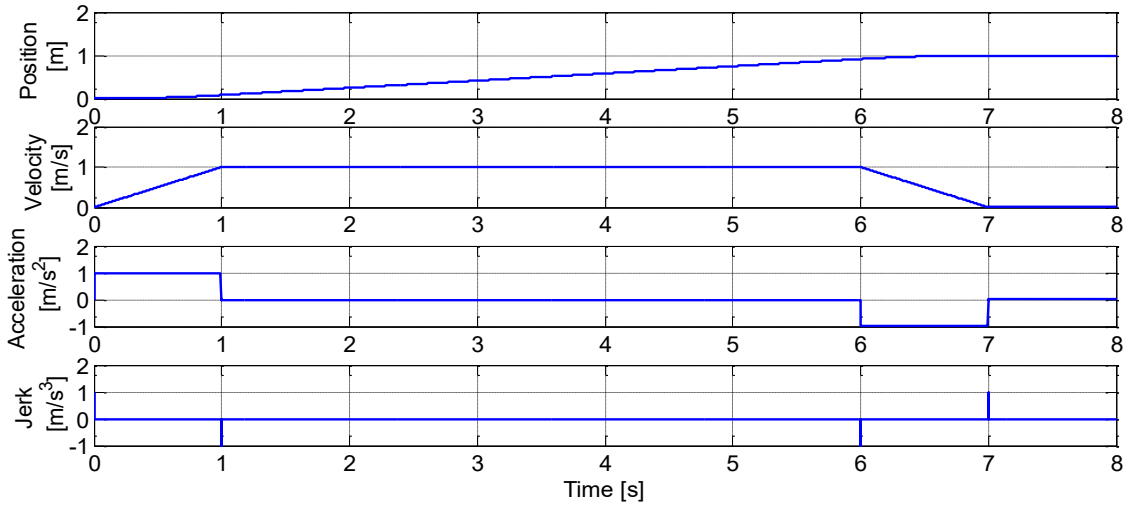

Figure 4.11 Third Order Feedforward Control of Skull Drilling" Jerk limited Trajectory

A derivative of the jerk signal (snap signal) is bounded in the fourth order feedforward motion planner; The specifics of motion planning based on rigid-body analysis of the double mass system as shown in Fig. 4.11 where x is the position, m is equivalent mass or inertia, b is viscous damping, F is actuator force, and F is feedforward force.

$$m_1\ddot{x}_1 = -K_1\dot{x}_1 - c(x_1 - x_2) - k_{12}(\dot{x}_1 - \dot{x}_2) + F \tag{4.2.1.17}$$

$$m_2\ddot{x}_2 = -K_2\dot{x}_2 + c(x_1 - x_2) + k_{12}(\dot{x}_1 - \dot{x}_2) \tag{4.2.1.18}$$

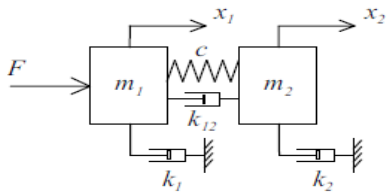

Figure 4.12 Double mass rigid body analysis

$$F = \frac{q_1 s^4 + q_2 s^3 + q_3 s^2 + q_4 s}{k_{12}s + c}.x_2 \tag{4.2.1.19}$$

$$q_1 = m_1 m_2 \tag{4.2.1.20}$$

$$q_2 = (m_1 + m_2)k_{12} + m_1 k_2 + m_2 k_1 \tag{4.2.1.21}$$

$$q_3 = (m_1 + m_2)c + k_1 k_2 + (k_1 + k_2)k_{12} \tag{4.2.1.22}$$

$$q_4 = (k_1 + k_2)c \tag{4.2.1.23}$$

$$F = \frac{1}{k_{12}s + c}.\{q_1 d + q_2 j + q_3 a + q_4 v\} \tag{4.2.1.24}$$

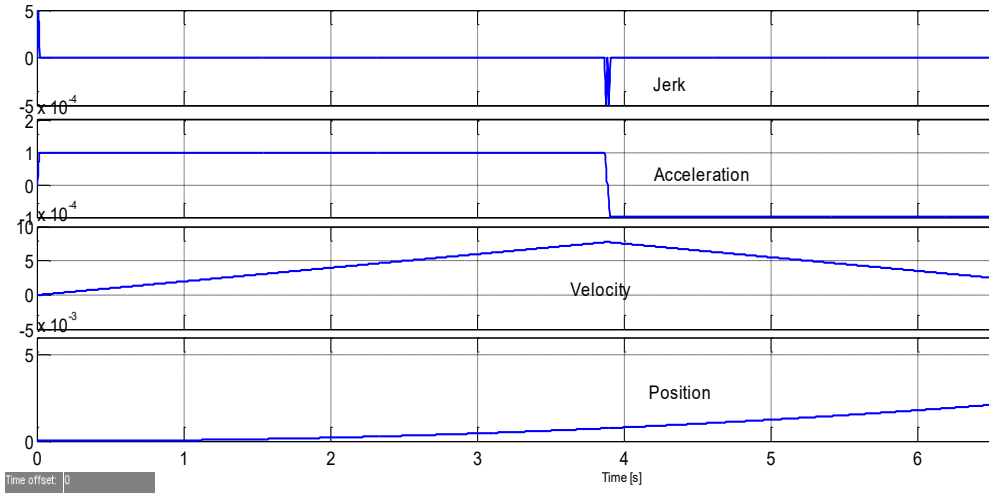

Figure 4.13 Fourth Order Feedforward Motion Planner

Fig. 4.13 shows the resulted Fourth Order Feedforward Motion Planner" where the deceleration of the mechanical system shutdown is bounded by the limit of Snap signal.

4.2.2 PID CONTROLLER FOR SKULL DRILLING

A Proportional–Integral–Derivative Controller (PID controller) is a generic control loop feedback mechanism (controller) widely used in industrial control systems. A PID controller attempts to correct the error between a measured process variable and the desired setpoint by calculating and then outputting a corrective action that can adjust the process accordingly and rapidly, to keep the error minimal [133]. PID controller is used to controlling the manipulator action, and suitable values of controller gains are chosen to avoid the oscillations in robotic manipulator and successfully removal of errors and make the movements as the desired movements.

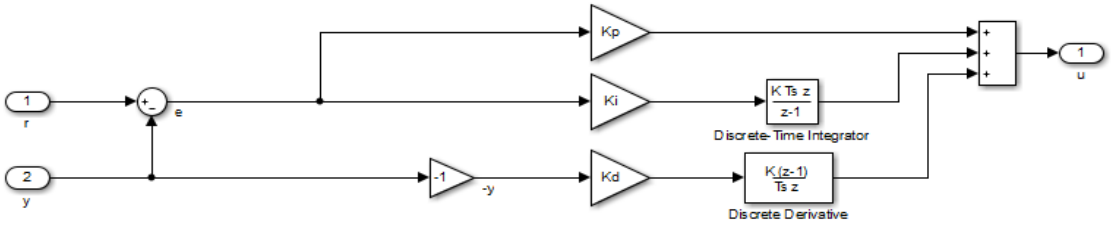

Figure 4.14 Block diagram of PID controller

PID controller is used to controlling the skull drilling. Parallel PID structure implemented in a discrete time with Backward Euler numerical integration method. The transfer Function of discrete time PID controller can be expressed as follows:

$$Kp + Ki * \frac{Ts.z}{z-1} + Kd * \frac{z-1}{Ts * z} \qquad (4.2.2.1)$$

Table 4.1 Values of PID controller

Proportional Gain 'Kp'	Integral Gain 'Ki'	Derivative Gain 'Kd'	Sample Time
2.8114	0.2484	1.1913	0.005 second

PID tuning and loop optimization techniques are used for offline tuning. Skull model developed using Matlab system identification toolbox, and optimal tuning values are calculated based on transient response parameters. Phase margin is selected to be 60 degrees, and overshoot to be zero. The PID controller calculation (algorithm) involves three separate parameters; the proportional, the integral and derivative values. The proportional value determines the reaction to the current error, the integral value determines the reaction based on the sum of recent errors, and the derivative value determines the reaction based on the rate at which the error has been changing.

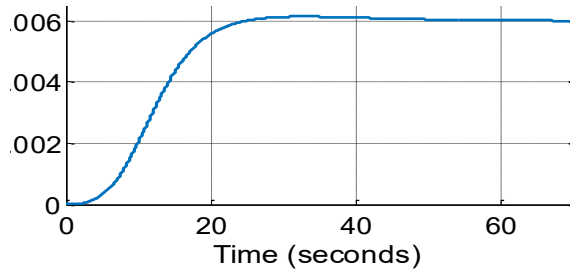

Figure 4.15 Response of PID controller for Skull Drilling

Figure 4.15 shows the time response of Drilling Depth Controller used for skull Drilling; Y axis represents the drilling Depth in meters; drilling done is 26 seconds for 6 mm. Table 4.1 shows the selected values for designed PID Controller.

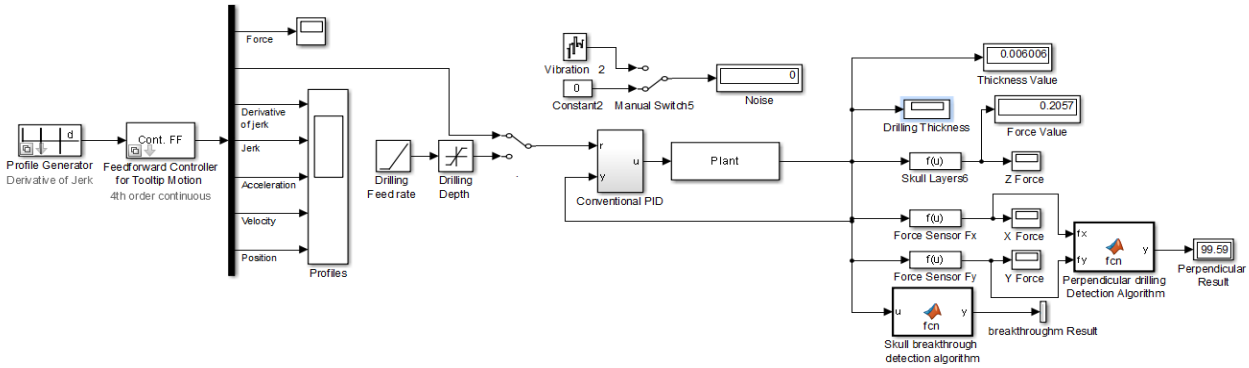

Figure 4.16 MATLAB Simulation of PID controller for Skull Drilling

Fig 4.17 shows the response of PID controller using different types of trajectory planner; fourth order feeds forward trajectory shows the best steady-state response when compared with the first, second, third order feedforward trajectories. Setpoint for skull drilling thickness is required to be 6mm.

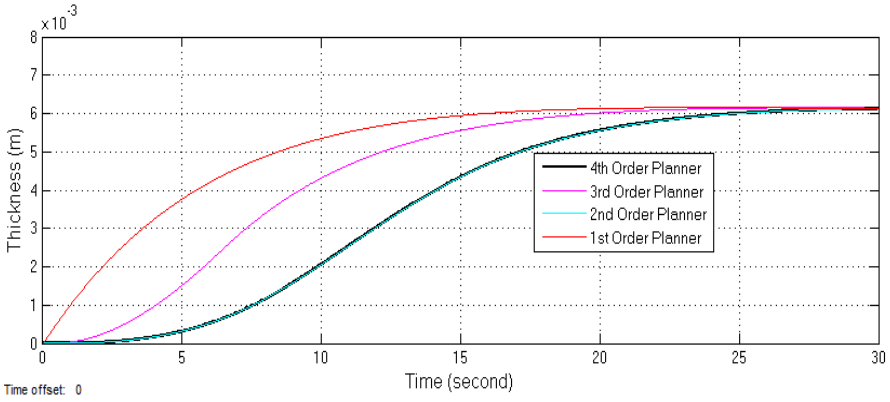

Figure 4.17 Response of PID controller using several types of trajectory planner

4.2.3 FUZZY PID CONTROLLER FOR SKULL DRILLING

A fuzzy model is developed for controlling the thrust force applied to the skull surface during drilling as a function of the drilling process parameters. A fuzzy controller analyzes the force sensor data measured by the Force sensor mounted on the robotic drill.

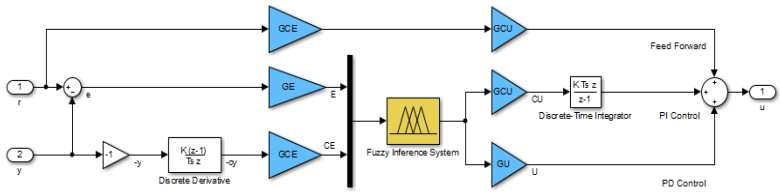

Figure 4.18 Structure of Fuzzy PID controller for Skull Drilling

Fig 4.18 shows the structure of Fuzzy PID controller used for Skull Drilling. When break-through is detected, the drilling controller automatically will stop drilling to prevent excessive protrusion of the drill bit.

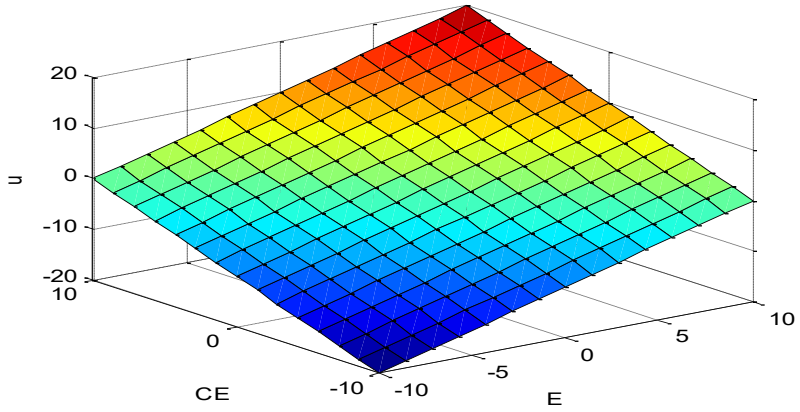

Figure 4.19 Control Surface plot of Fuzzy PID Controller

There will be no unexpected failure, and the desired overshoot of drilling limited to less than 1 mm. Fig 4.19 shows Matlab Simulation of Nonlinear Fuzzy PID controller for Skull Drilling. Fig 4.20 shows the response of Fuzzy PID controller using different types of trajectory planner; fourth order feed forward trajectory shows the best steady-state response when compared with the first, second, third order feedforward trajectories.

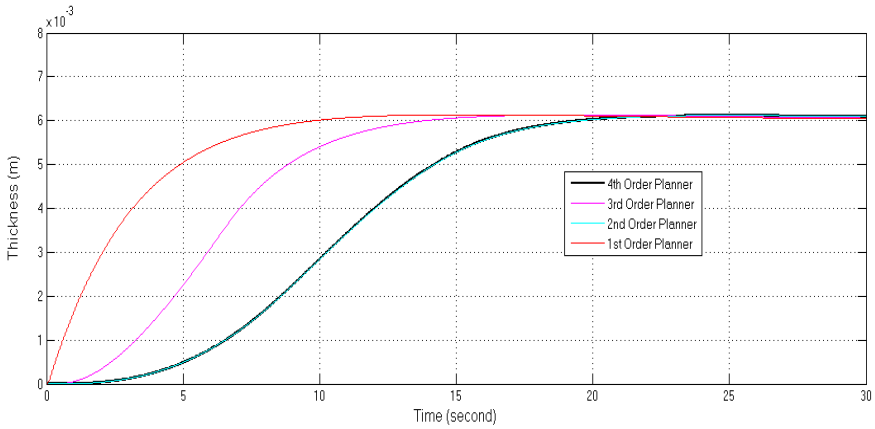

Figure 4.20 Response of Fuzzy PID controller using different types of trajectory planner

4.2.4 NONLINEAR FUZZY PID CONTROLLER

Nonlinear fuzzy PID controller is designed for a skull drilling Controller. Drilling Trajectory approximated by a single-input-single-output system in discrete time and our design goal is simply to achieve good reference tracking performance. The change of output is used to smooth the response of the derivative action. Nonlinear control surfaces can often be approximated by lookup tables to simplify the generated code and improve execution speed. Two gain blocks, GCE and GCU in the feed-forward path from r to u, are used to ensure that the error signal e is used in proportional action when the fuzzy PID controller is linear.

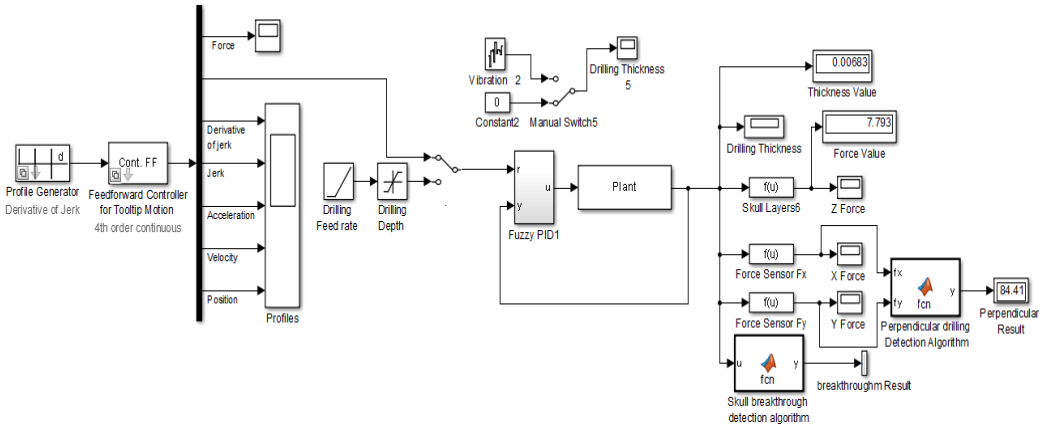

Figure 4.21 Matlab Simulation of Nonlinear Fuzzy PID controller for Skull Drilling

A fuzzy inference system (FIS) maps given inputs to outputs using fuzzy logic. Figure 4.22. Shows control surface plot which describes the mapping of a two-input one-output fuzzy controller in a 3-D plot.

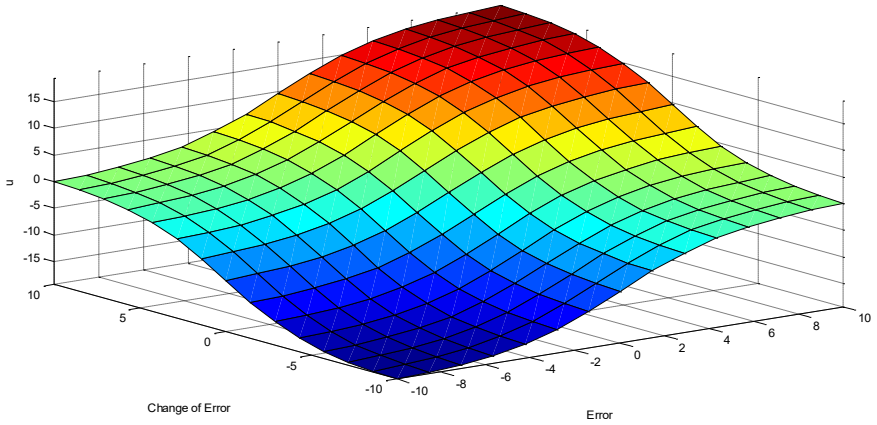

Figure 4.22 Control Surface plot of Nonlinear Fuzzy PID Controller

Fig 4.23 shows the response of nonlinear fuzzy PID controller using different types of trajectory planner; fourth order feeds forward trajectory shows the best steady-state response when compared with the first, second, third order feedforward trajectories.

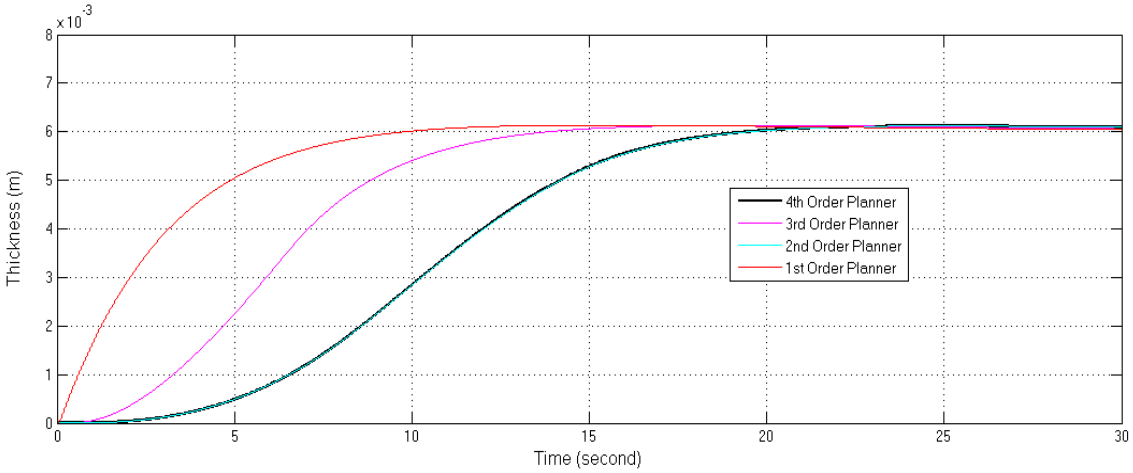

Figure 4.23 Response to Non-Linear Fuzzy PID Controller

4.2.5 APPLICATION OF LQR IN SKULL DRILLING CONTROLLER

Optimal control deals with finding a certain optimality criterion to be achieved. Optimization techniques divided into three categories: analytical and numerical procedures, and nonlinear programming [134]. The Linear quadratic regulator is a technique based on a state feedback to minimize the cost function within prescribed constraint boundaries. The main advantage of LQR is that the optimal input signal u (t) derived from full state feedback; the feedback matrix K is

obtained by solving the Ricatti equation. The Objective of the Cost function is to minimize the total energy of the closed loop system.

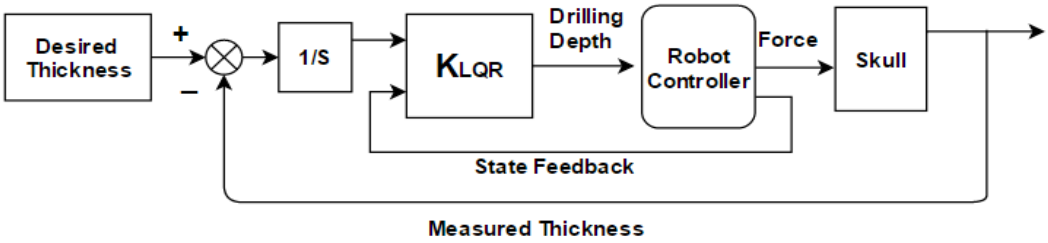

Figure 4.24 Block diagram of LQR added to the control system

Neurosurgical robotic skull drilling system aims to drill a hole in the human skull by using robotic arm holding rotating drill; where LQR is used in the state-space design approach, which is well suited to the control of multiple outputs as we have here. Disturbance rejection is a critical factor in skull drilling system to minimize the chance of brain damage due to the over drilling. So, an automatic skull drilling system with excellent noise rejection is needed to perform the function of drilling. A state-space model of the Skull drilling controller is constructed with two inputs (Thickness_ set point, Force) and one output (Thickness); System response improved by adding a linear quadratic regulator (LQR) for the feedback structure shown in Fig. 4.25. The LQR algorithm is a state-feedback controller. LQR technique uses the state vector x to synthesize the drilling thickness set point. In addition to the integral of error, the resulting thickness output is of the form:

$$\mathrm{Dp} = \mathrm{K1} * \mathrm{w} + \mathrm{K2} * \mathrm{w/s} + \mathrm{K3} * \mathrm{I} \tag{4.2.5.1}$$

For better disturbance rejection, a cost function used to penalize large integral error. The LQR gives better disturbance rejection compared with the feed forward controller. Continuous-time LQR defined on $t \in [t_0, t_1]$ can be described by

$$\dot{x} = Ax + Bu \tag{4.2.5.2}$$

with a quadratic cost function Q_c defined as

$$Q_c = \int_0^\infty (x^T Q x + u^T R u + 2x^T N u) dt \tag{4.2.5.3}$$

$$Q_c = \int_0^\infty (40q(t)^2 + w(t)^2 + 4Dp(t)^2) dt \tag{4.2.5.4}$$

For a discrete-time state-space model:

$$J = \sum_{n=0}^{\infty} (x^T Q x + u^T R u + 2x^T N u) \tag{4.2.5.5}$$

The state-feedback law u[n] = –Kx[n] minimizes the quadratic cost function. The pair (A, B) is stabilizable before applying LQR algorithm; otherwise, LQR is not applicable; The associated Riccati equation of and Optimal gain matrix K is calculated.

$$\mathrm{A^T S + SA - (SB + N)R - 1(B^T S + N^T) + Q = 0} \tag{4.2.5.6}$$

$$\mathrm{K = R - (B^T S + N^T)} \tag{4.2.5.7}$$

$$\mathrm{J = \sum_{n=0}^{\infty} (x^T Q x + u^T R u + 2x^T N u)} \tag{4.2.5.8}$$

For a discrete-time linear system described by

$$x_{k+1} = Ax_k + Bu_k \tag{4.2.5.9}$$

with a performance index defined as

$$J = x_N^T Q x_N + \sum_{k=0}^{N-1} (x_k^T Q x_k + u_k^T R u_k + 2x_k^T N u_k) \tag{4.2.5.10}$$

The optimal control sequence minimizing the performance index is given by

$$u_k = -(R + B^T P_{k+1} B)^{-1} (B^T P_{k+1} A + N^T) x_k \tag{4.2.5.11}$$

P_k is found iteratively backwards in time by the dynamic Riccati equation

$$P_{k-1} = A^T P_k A - (A^T P_k B + N)(R + B^T P_k B)^{-1} (B^T P_k A + N^T) + Q \tag{4.2.5.12}$$

LQR has better set point tracking and disturbance rejection compared with feedforward controller; and Integral feedback controller. However; filters should be used to reduce unwanted behavior like known or measured disturbances and non-linearity.

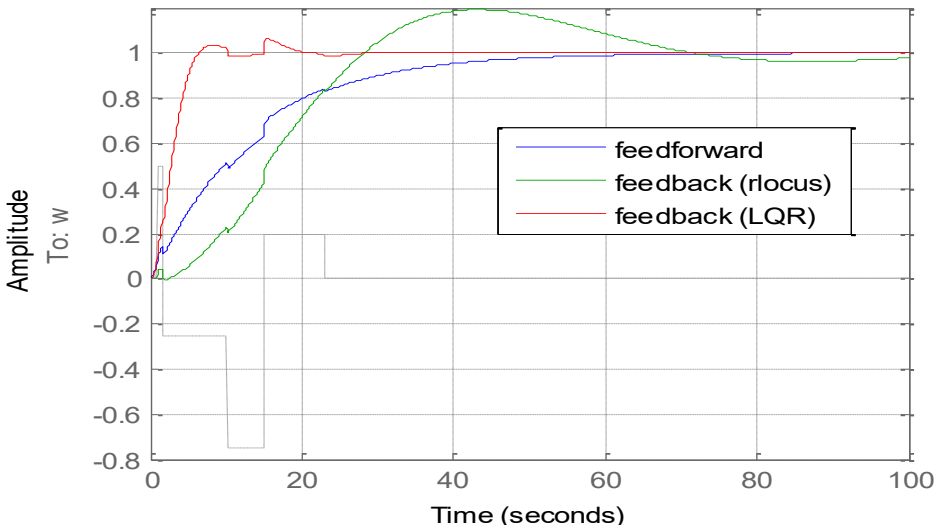

Figure 4.25 Setpoint tracking and disturbance rejection

4.2.6 VERIFICATION OF TRAJECTORY TRACKING

The end-effector trajectory is tracked with respect the required drilling trajectory to verify the performance of the proposed controller for the Mitsubishi PA-107c robotic manipulator. The chosen trajectory was a lemniscate in the y-z plane of the robot base coordinate system. This trajectory is chosen because it is smooth and easy to generate. The mathematical expression for lemniscate trajectory is given by:

$$x = 0.15, \quad y = 0.2\frac{cos\left(\frac{t}{2}\right)}{\left(1 + sin\left(\frac{t}{2}\right)^2\right)} \tag{4.2.6.1}$$

$$z = 0.1 + 0.4\frac{sin(\frac{t}{2})cos(\frac{t}{2})}{\left(1 + sin(\frac{t}{2})^2\right)} \tag{4.2.6.2}$$

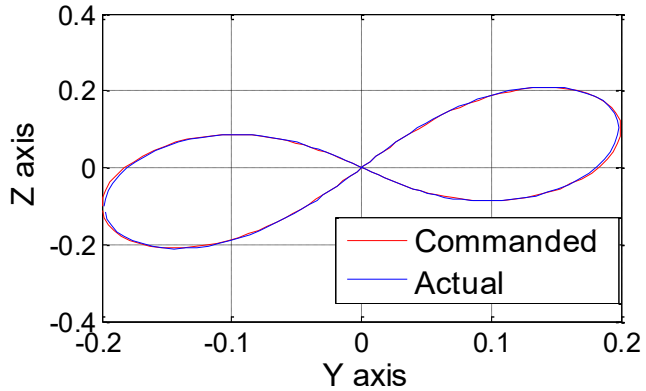

Figure 4.26 End Effector Trajectory Tracking Experiment

Fig 4.28 shows End- Effector Trajectory Tracking Experiment; to verify end effector trajectory tracking; the following discrete time trajectory is selected, and T is sampling time in millisecond.

$$S_1(k) = \frac{\pi}{2}\left(1 - e^{0.001kT^3}\right)sin(0.285KT) \quad (4.2.6.3)$$

$$S_2(k) = \frac{\pi}{3}\left(1 - e^{0.001kT^3}\right)sin(0.453KT) \quad (4.2.6.4)$$

$$S_3(k) = \frac{\pi}{2}\left(1 - e^{0.01kT^3}\right)\sin(0.285KT) \quad (4.2.6.5)$$

$$E_1(k) = \frac{\pi}{3}\left(1 - e^{0.01kT^3}\right)Sin(0.555KT) \quad (4.2.6.6)$$

$$E_2(k) = \frac{\pi}{2}\left(1 - e^{0.01kT^3}\right)Sin(0.345KT) \quad (4.2.6.7)$$

$$W_1(k) = \frac{\pi}{3}\left(1 - e^{0.01kT^3}\right)Sin(0.615KT) \quad (4.2.6.8)$$

$$W_2(k) = \frac{\pi}{2}\left(1 - e^{0.01kT^3}\right)Sin(0.465KT) \quad (4.2.6.9)$$

4.3 SUMMARY

Feedforward trajectory planner designed for optimizing the performance of neurosurgical robotic manipulator used for human skull drilling; The required trajectory planned with constrained dynamics for the robot end effector drilling system. Drilling trajectory is a point to point motion with optimal time required to minimize the friction effect

Table 4.2 Comparison of Transient Response Characteristics

Controller	Rise Time	Overshoot	Settling time	Steady-state error
Feedback	16 s	0.05mm	28 s	0.001mm
PID	14 s	0.025mm	23 s	0.0001mm
Fuzzy PID	13.2s	0.0125mm	22.5 s	0.0001mm
Non-linear Fuzzy PID	13.2s	0.0125mm	22.5 s	0.0001mm

A nonlinear fuzzy PID controller is used and compared with fuzzy inference system and PID controller. As shown in Table 4.2; A Nonlinear fuzzy controller deployed with improved execution speed. Simulation results show the smooth trajectory of fourth order feedforward planner in comparison with rigid-body feedforward trajectory planner.

CHAPTER 5 FORCE SENSING AND CONTROL

Force estimation is a critical parameter in neurosurgery; the main advantage is to minimize the error in skull drilling and could be used to optimize the skull drilling planner. This chapter presents the proposed force detection and Control algorithms. The organization of the chapter is as follows. Skull under drilling is modeled using linear and experimental model; A simplified model of the human skull under drilling by using Matlab SimMechanics toolbox presented in section 5.1. Section 5.2 describes the skull thickness measurement and estimation technique. Section 5.3 presents the Neural networks classifier for detecting of bone layers and Drilling state. Direct and indirect force control presented in section 5.6 and 5.7.

5.1 LINEAR MODEL OF THE SKULL UNDER DRILLING

To accomplish a systematic study of robotic skull-drill a model of the skull-drilling interaction forces is needed to create a mathematical model to simulate the dynamics of the drilling system. In the literature, there is little information on methods for applying such a force to a dynamic manipulator simulation. Drilling mechanism is the interaction between the robot tooltip drill and the skull; which could be modeled by the double suspended mass-spring-damper system. This simplified model could be used to analysis force and vibration acting on the drill. Two suspended masses linked by springs and dampers demonstrating basic PA10 tooltip modeling by using SimMechanics Toolbox.

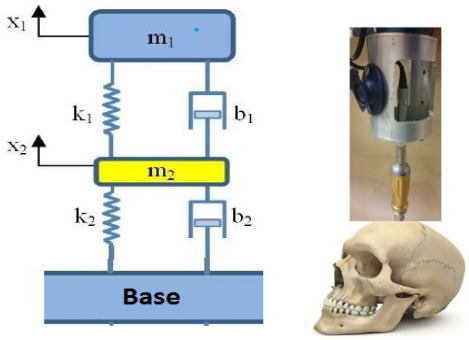

Figure 5.1 Simplified Model of the Skull Drilling Mechanism

Since the equations of the system cannot be solved easily in mathematical form. So; SimMechanics model developed with a schematic in Matlab Simulink that allows analyzing the behavior of the drilling transit response.

Table 5.1 Design Parameters of the Skull Drilling Model

Part Name	Variable	Value
Mass of Drill tool	M1	1 Kg
Damper of Drill tool	B1	2 Kg/sec
Spring of the Drill tool	K1	20 N/m
Mass of Skull	M2	1 Kg
Damper of Skull	B2	2 Kg/sec
Spring of the Skull	K2	20 N/m

The system is shown in Fig. 5.1 is a representation of the Skull Drilling Mechanism where m1 is the mass of robot drilling head, while m2 is the mass of the human skull, K1 & K2 are the stiffness coefficient of the robot drilling head and table stiffness coefficient. Suspension. B1 - is the damping coefficient of the suspension of the drilling head, b2 is the damping coefficient of the table which the head mounted on it. X2 is the vertical displacement of drilling.

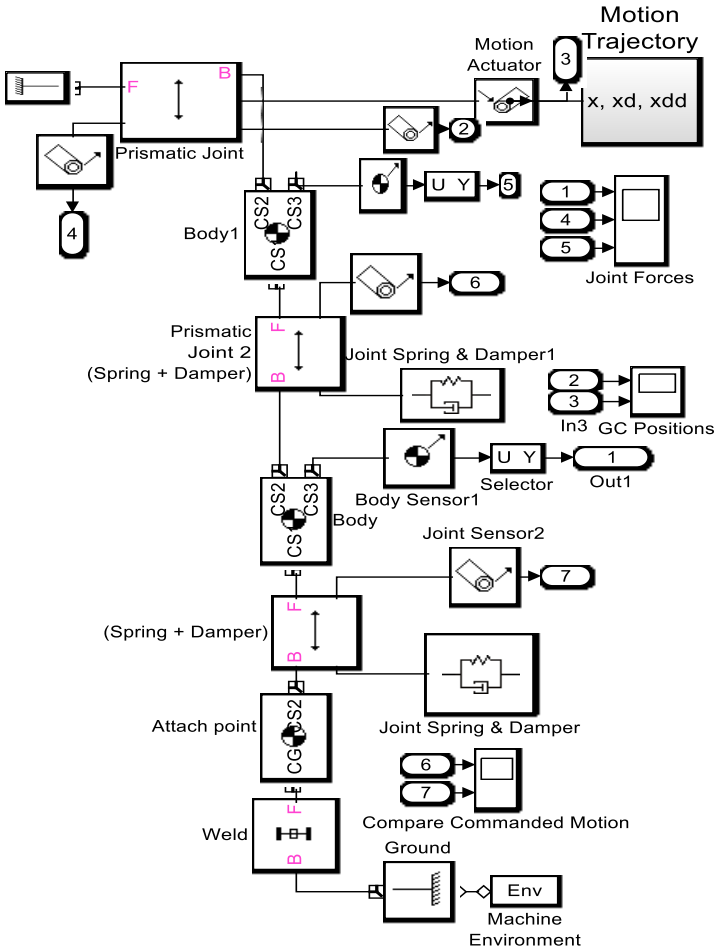

Figure 5.2 SimMechanics Model of the Skull Drilling Mechanism

Fig. 5.2 shows the SimMechanics Model of the skull drilling mechanism. This model used for both forward dynamics and inverse dynamics analysis. This model demonstrates how to specify the position, velocity, and acceleration of the drill mounted on the robot end-effector and measure the force necessary to generate drilling motion. Only mass movements on the vertical axis will be considered in this model. The response of Force modeling is shown in Fig. 5.3.

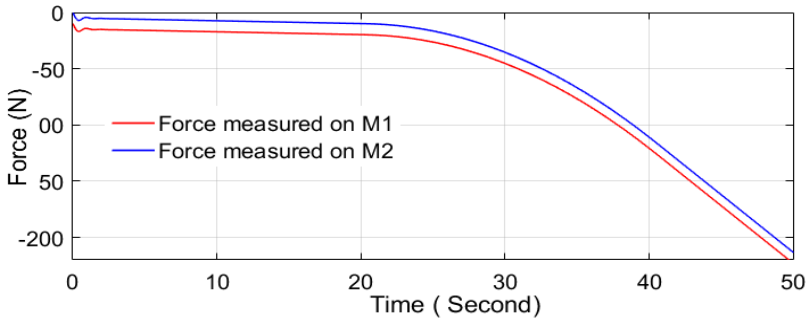

Figure 5.3 Response of Force Modelling

The displacement x1 & x2 will demonstrate the behavior of the drilling tool. This model applies a motion at the extremity of a double mass-spring-damper system; it measures the forces generating the motion which is the forces between masses. Force applied by the spring Measure the force required to produce the motion specified by the Joint actuator in motion mode. The resulted displacement in Fig.5.4 represents the drilling depth.

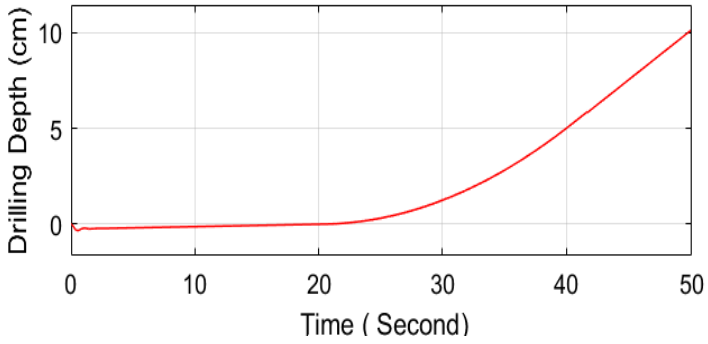

Figure 5.4 Drilling Depth in Skull

5.2 EXPERIMENTAL MODEL OF SKULL UNDER DRILLING

Due to the safety requirements of the robotic neurosurgical drill, a prediction model for the force applied on skull under drilling is proposed. Experimental data is used to derive the transfer function of the human skull under drilling.

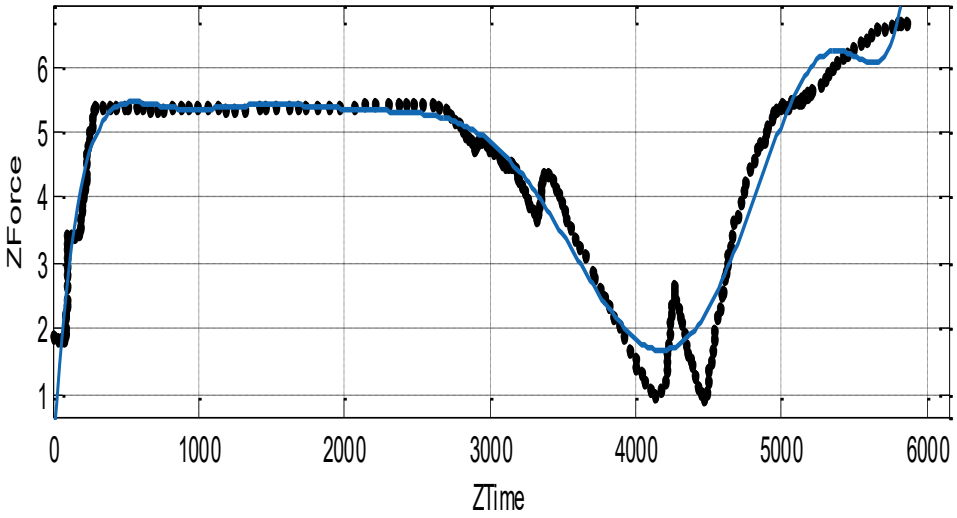

Figure 5.5 Thrust Force (Fz) Response of skull under Drilling

Matlab system identification toolbox is used to approximate the transfer function. Drilling experiments were done on the human skull by using PA107c robotic manipulator. The thrust force is measured using a JR3 sensor which mounted on the robot tooltip. The resulted force is shown in Fig.5.5.

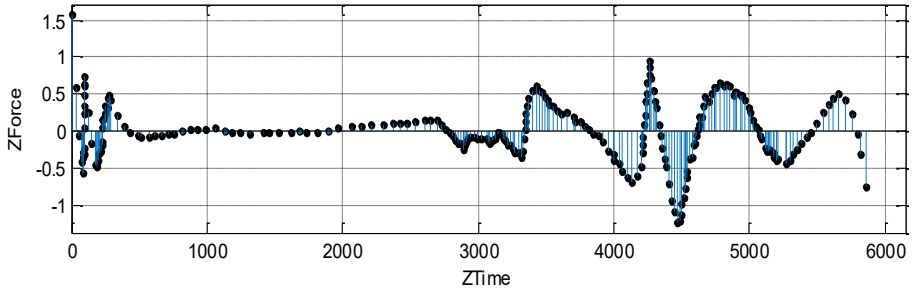

Figure 5.6 Residuals of the Thrust Force

Linear Polynomial function with 9 degrees; LAR robust algorithm is used to improve system identification; the resulted Transfer function is:

$$f(t) = P_1 * x^9 + P_2 * x^8 + P_3 * x^7 + P_4 * x^6 + P_5 * x^5 + P_6 * x^4 + P_7 * x^3 + P_8 * x^2 + P_9 * x + P_{10} \quad (5.2.1)$$

Table 5.2 Coefficients values with 95% confidence bounds

Coefficients	Value	95% confidence bounds
p1	6.696e-30	(5.323e-30, 8.068e-30)
p2	-1.718e-25	(-2.081e-25, -1.355e-25)
p3	1.843e-21	(1.44e-21, 2.247e-21)
p4	-1.078e-17	(-1.322e-17, -8.336e-18)
p5	3.761e-14	(2.89e-14, 4.632e-14)
p6	-8.081e-11	(-9.943e-11, -6.218e-11)
p7	1.061e-07	(8.305e-08, 1.292e-07)
p8	-8.136e-05	(-9.649e-05, -6.622e-05)
p9	0.03269	(0.02826, 0.03713)
p10	0.2055	(-0.1893, 0.6003)

Goodness-of-fit for the drilling force parametric model evaluated by calculating: The sum of squares due to error (SSE); R-square; Adjusted R-square; and Root mean squared error (RMSE). The resulted transfer function has a good fit as shown in Table 5.2.

Table 5.3 Goodness of fit

SSE:	R-square:	Adjusted R-square:	RMSE:
34.58	0.9378	0.9353	0.392

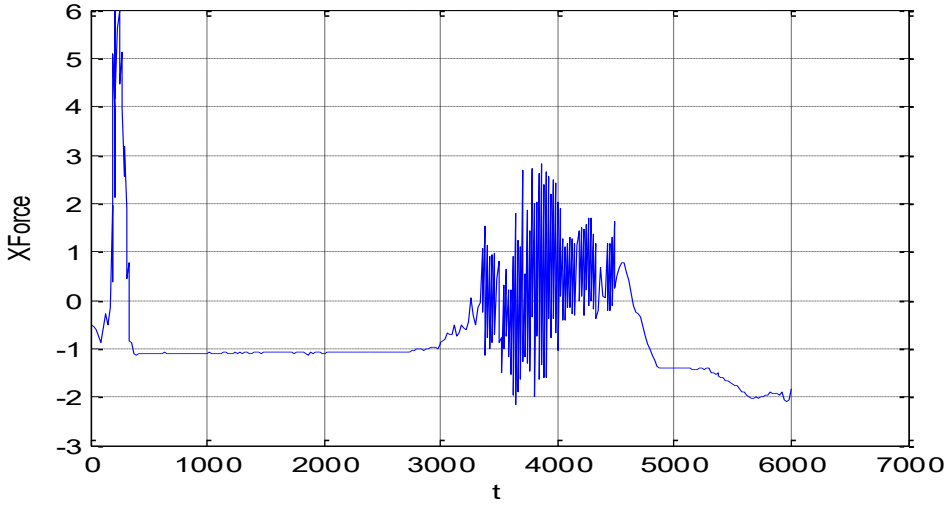

Figure 5.7 Drilling Force along x-axis

Fig. 5.7 and Fig. 5.8 show Drilling force along X-axis and Y-axis. Force difference between Fx and Fz used in obtaining a perpendicular drilling by calculating the force difference between Fx and Fy.

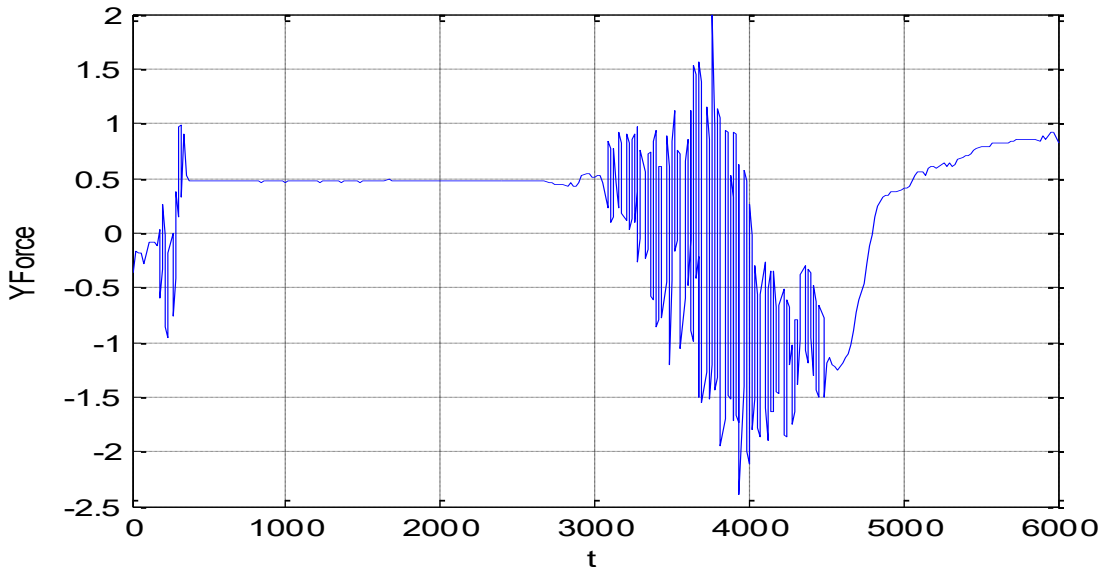

Figure 5.8 Drilling Force along y-axis

The desired prescribed drilling path is specified in the Cartesian space, as this is where the drilling specifications easily described. Then, given a table of desired points, the end effector should have the robot pass through. Finally; the inverse kinematics used to convert points from Cartesian space to joint space. The rotary speed of the Skull drilling tool is a major factor that affects the drilling efficiency. Drill bit breaks skull with light bit pressure mainly by grinding, the faster the rotating speed is, the faster the drilling speed.

Table 5.4 Skull Drilling Trajectory Profile

Drilling Depth (mm)	6
Federate (mm·s−1)	0.27
Acceleration (mm.s−2)	0.01
Jerk (mm·s−3)	0.05
Snap (mm·s−4)	0.05

Table 5.4 shows the reference parameters for the generated drilling trajectory of the skull. The recommended drill speed is less than 6000 rpm. The maximum recommended cutting speed is 0.27 mm/sec. The maximum recommended acceleration is 0.01 mm/sec^2.

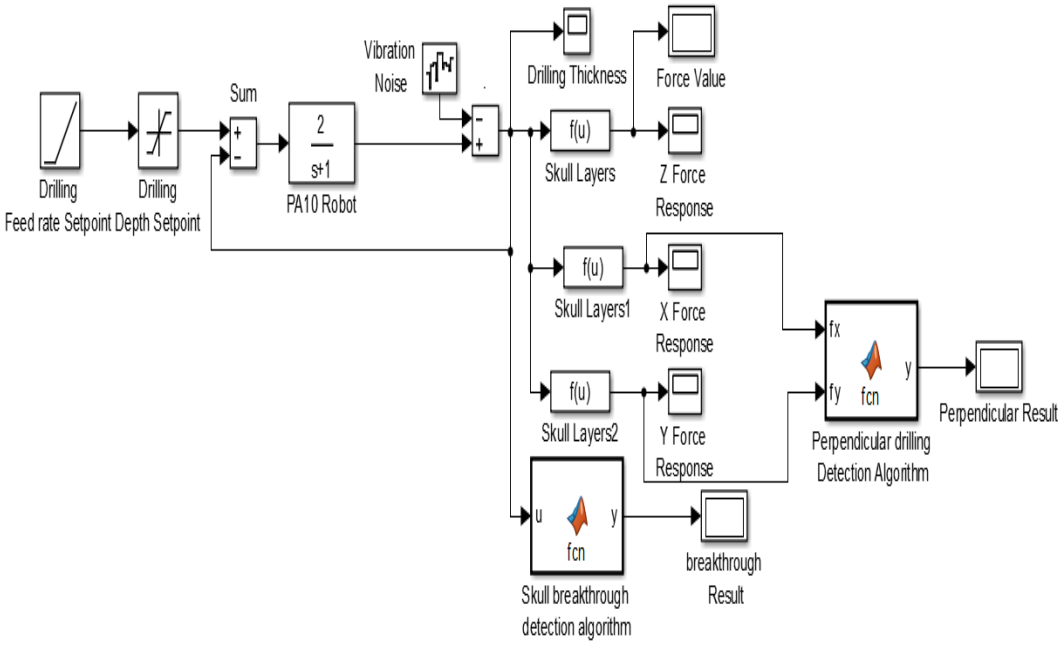

Figure 5.9 Matlab Simulation of Skull Drilling

The maximum recommended drill force 20 N. The rotating speed of bit ‘n’ can be calculated by:

$$n = (60v)/(\pi D) \qquad (5.2.2)$$

Where ‘v’ is the cutting linear velocity of the drilling tool, 'D' is the diameter of the bit. Because of small in bit pressure and shallow depth in the drilling of the skull drill, high rotating speed can be adopted to obtain higher drilling speed.

5.3 SKULL DRILLING STATE DETECTION

Thrust is a reaction force described quantitatively by Newton's second and third laws. When a robot accelerates drilling tooltip mass in one direction, the accelerated drilling tooltip mass will cause a force of equal magnitude but opposite direction on that system. Thrust force applied on a skull surface in a direction perpendicular or normal to the skull surface. A force sensor is used to measure the contact force “thrust force” between the robot tooltip and human skull; Force control is needed due to the need of an accurate movement in contact with the skull under neurosurgical operation. Without the sensor interaction, the process may damage skull. Strategies that handles this drilling task make a larger scope of use and a higher safety possibility and improves drilling performance. There are two practical implementations of force/torque sensors using equilibrium

condition between two forces, being one known and working as a reference to obtain the other. The second method is the determination of motion parameters imposed on a known mass by the unknown force. This approach used in the JR3 force/torque sensor.

Table 5.5 Force Sensor Specifications

Parameter	Force	Torque
Range in X, Y	$\pm 100N$	± 6.3
Range in Z	± 200 N	± 6.3
Accuracy	0.009N	0.01
Resolution in x, y	0.025 N	0.0016
Resolution in Z	0.05N	0.0016

The force sensor mounted on the drilling tooltip of the PA10-7c robotic manipulator; the force/torque sensor measures near the point where the robot control. An orientation transformation used for mapping force between joint coordinates and Cartesian coordinates. In the presented system, a force/torque sensor from JR3 Ltd. used as a wrist sensor on the end effector.

Figure 5.10 Force Sensor JR3

This sensor is very stiff and will therefore not affect the position accuracy of the robot system. The JR3 Force/ Torque sensor used is of type 100M40A. It is mounted on the tool tip of the robotic manipulator; JR3 instrumented with strain gauges can measure forces in the x-, y- and Z-directions as well as the corresponding torques. It is DSP-based board and has a sample rate of 8 kHz. It can measure forces up to 400N and torques up to 40Nm.

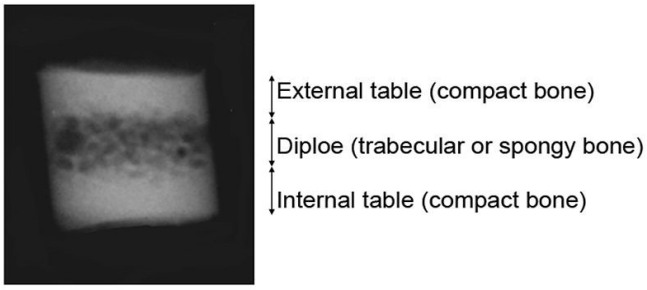

Figure 5.11 The three bone layers of the cranial vault. [135]

The skull is composed of two layers of cortical bone, the inner and outer "tables"- separated by a cancellous bone [136]. Cortical bone, synonymous with compact bone, is one of the two types of osseous tissue that form bones as shown in Fig 5.11. Each layer has different structure; that is why each layer reacts differently when robotic arm is applying drilling on the skull.

5.3.1 NEURAL NETWORK STATE CLASSIFIER

Neural Networks widely used in many applications, such as prediction, pattern recognition, and classification. Neural networks provide an alternative to algorithmic methods when logical rules do not exist or are difficult to determine; especially when the input data may be incomplete or distorted. A force sensor is mounted on the drilling tool tip to detect the sharp drop in thrust when the drill crosses the interfaces between hard and soft tissues of the bone.

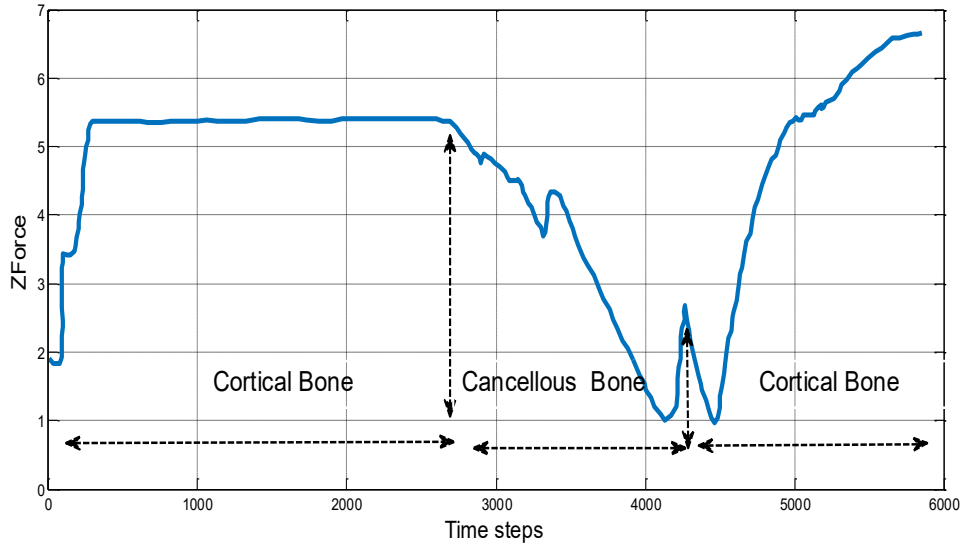

Figure 5.12 Thrust Force (Fz) Response of skull under Drilling

A neural network pattern recognition algorithm analyzes the force and torque data received from the JR3 sensor. Force data is pre-filtered using the built in JR3 filters, and force data are used in the determination of skull drilling state by using Neural Networks. Force data is represented as a force with time series as shown in Figure 5.8; skull bones consist of internal and external layers of compact bone, separated by diploë [137].

Identifying robot working States during drilling is necessary. Moreover, To determine if the perforator has perforated the hole in the skull, the robot needs to know current stopping-drilling state to avoid stressing both the robot and the patient. Many training algorithms could applied to train ANN such as Levenberg-Marquardt algorithm (LM) which is one of the fastest algorithms and works perfectly with small neural networks; but it is not suitable for very large neural networks due to the memory and computation overhead. Quasi-Newton algorithm is based on the Newton's method, which computes the update of approximated Hessian matrix as a function of the gradient. Generally, it converges in a few iterations. Nevertheless, it is not suitable for very large networks because of its computation and memory overhead. However, it is very efficient for small size networks. Gradient descent algorithm "GD" updates the weights and biases in the direction of the negative gradient of the performance function. It is slow and usually trapped in local minima of the error surface. Gradient descent by momentum algorithm "GDM" is similar to Gradient descent algorithm; but it improves the gradient by using momentum during the training. It could overcome the local minimum problem and has faster convergence than the Gradient Descent algorithm.

Table 5.6 Ranking Training Algorithms of FFN with 10 hidden layers

#	**Algorithm**	**Training Time**	**Operation Time**	**MSE**	**Regression**	**MAPE**
1	LM	27.74	0.0704	$1.17 * 10^{-7}$	1	0.0000056
2	BFG	9.1	0.0155	10.55	0.9938	0.432
3	RP	0.499	0.0159	8.75	0.9949	0.3375
4	SCG	0.416	0.0169	$1.17 * 10^{-7}$	1	0.0000056
5	GDM	0.254	0.0169	10.55	0.9938	0.423
6	OSS	0.901	0.0172	8.75	0.9949	0.3375
7	GDX	0.47	0.0166	$1.17 * 10^{-7}$	1	0.0000054

There are many training algorithms used with Feed Forward neural networks; these algorithms are trained with the experimental force data; and evaluation criteria is applied on networks result based

on: Training Time; Execution Time; Error; MSE; MAPE and regression analysis. The performance of the network measured in terms of its response as shown in Table 5.6.

Table 5.7 Network Performance in case of using 10 hidden layers

Network	Regression	MAPE	Processing Time	MSE
Net 1	0.9983	0.2726	0.0736	2.95
Net 2	0.9965	0.3375	0.0155	5.98
Net 3	1	0.000025	0.0154	$2.5 * 10^{-8}$

After the training process, the performance of the trained network evaluated by applying testing data and evaluate the fit between inputs and outputs. So; based on the performance specification priorities; three algorithms chosen as following: RP, BFG, LM. All of them are fast in training; operation time; and have low regression value which is required to be law in case of improving the performance by using different network topologies. Table 5.7 shows the performance of three different ANN in case of using 10 hidden layers using different training techniques;

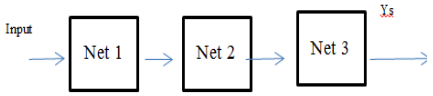

Figure 5.13 Serial Topology of ANN

Neural Networks; could be connected in many connection topologies; such as: serial topology and parallel topology. Cascade connection topology can be constructed by connecting the output of the first network to the input for the second network as shown in Fig 5.13 where Net 1 trained with ‘x’ input value and ‘t’ target value; while Net 2 will be trained with the output of Net 1 “y1” as an input to it with respect to “t” target value.

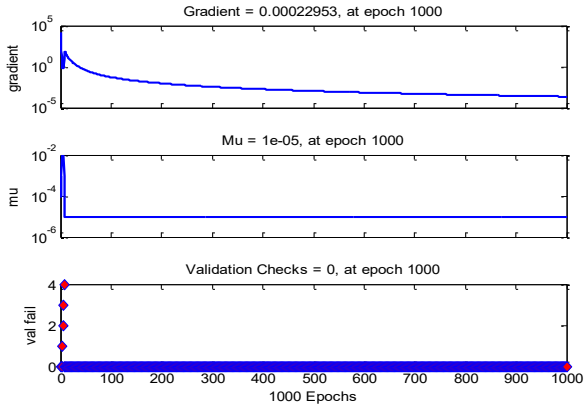

Figure 5.14 Training state of Serial ANN Topology

Connection order is very important factor in serial topology; accurate ANN should be in first stage to minimize the overall fitting error and to increase the overall performance. Fig. 5.13 shows serial connection topology of three ANN; training state and the performance of the aggregated ANN is shown in Fig. 5.14 and Fig. 5.15 respectively.

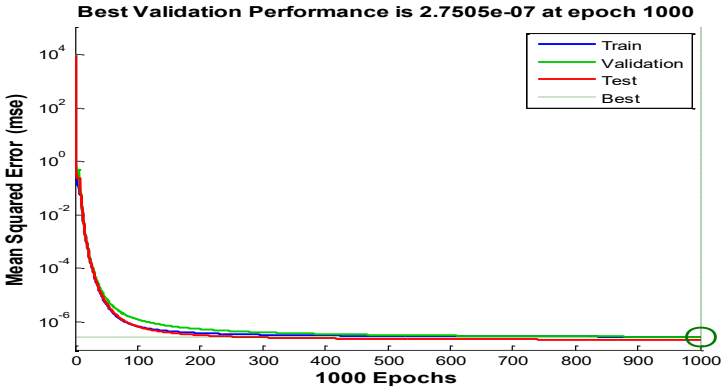

Figure 5.15 Evaluation of Serial ANN connection

Fig. 5.16 shows the Parallel topology of ANN; in the Parallel ANN Topology; same input is applied to all networks; and then the output of all networks is aggregated to single Output neuron. Different weight is assigned to each network where the best network is chosen to have higher weight.

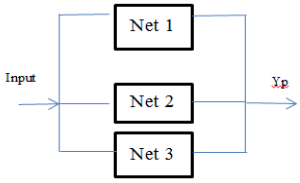

Figure 5.16 Parallel Topology of ANN

By connecting neural networks in parallel; there is no difference in connection sequence because all networks are in parallel; and the final output is improved compared with the output of each network due to the averaging mechanism in the parallel topology. Better results could be obtained by adding higher weight to the best network performance.

Table 5.8 Topology Performance in case of using 10 hidden layers

Topology	Regression	MAPE	Processing Time
Serial Ys	0.9965	0.2855	0.0157
Parallel Yp	0.9978	0.2684	0.0152

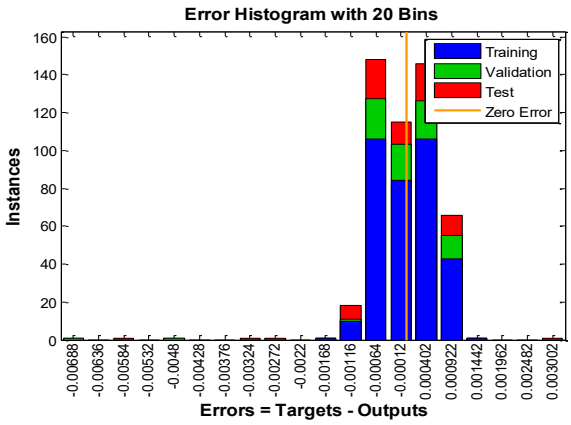

Figure 5.17 Evaluation of Parallel ANN Topology

Parallel topology is doing averaging for the output error; which mostly improve the performance in the systems with low data distribution; On the other hand; Serial topology is accumulating error from each training stage; which mostly improve the overall performance in the systems with high data distribution. Table 5.8 shows the comparison between serial and parallel topologies in case of using 10 hidden layers.

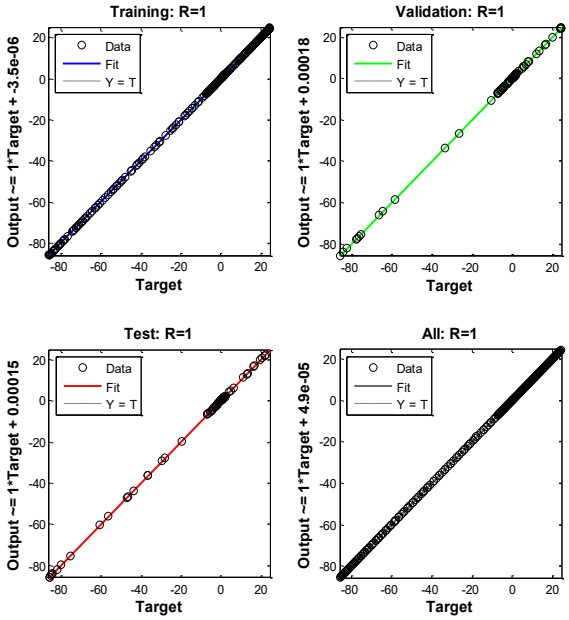

Figure 5.18 Regression Analysis of the Parallel ANN Topology

Fig. 5.18 shows performance evaluation of parallel topology; After analyzing each topology performance; it could be concluded that; If there is only one stage in the topology; then performance of parallel topology is better than the performance of serial topology.

5.3.2 DETECTION OF BONE LAYER USING ANN

The procedure of the classification based on the force signals using neural networks with 5-level decomposition as shown in Fig 5.19. Signal Thresholds defined for each drilling stage using Force signal in the Z direction. These levels are further applied to the Neural Network to classify drilling state, such as rotation without drilling, Start drilling in the first layer of skull bone; Start of drilling in the second layer; Start of drilling in the third layer; End of drilling the third layer and the detection of breakthrough. The force data comes from the JR3 force sensor. First, we collect a complete set of force data during the whole drilling process including the drilling in the three layers of the skull; 919 force samples used with neural networks; where data divided into three parts: 70% training; 15% validation and 15% testing.

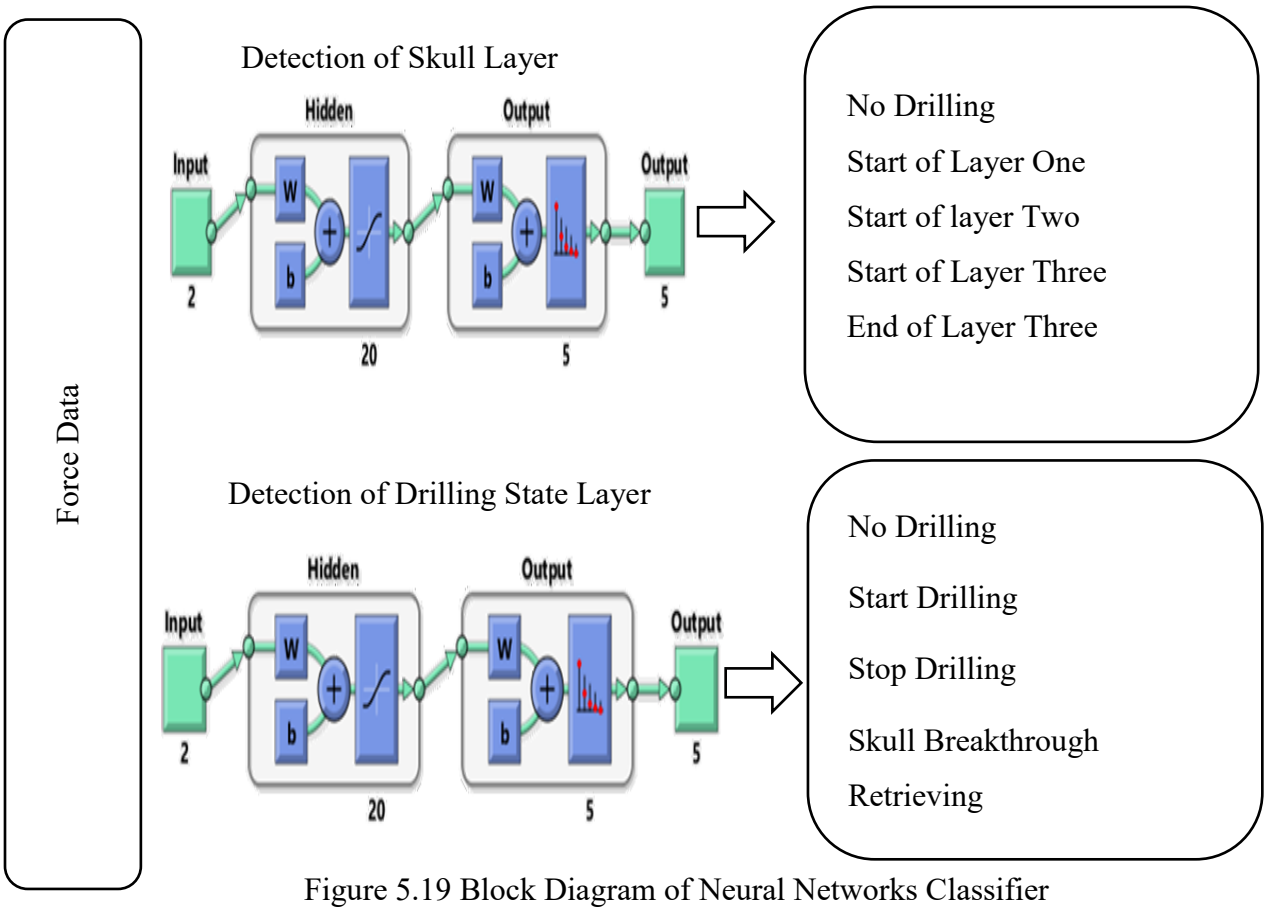

Figure 5.19 Block Diagram of Neural Networks Classifier

Following the procedure mentioned above, the force prediction experiment conducted, and the data processed offline. 641 samples used as training data and another two sets of 137 samples used in testing and validation respectively. The numbers of input and output layers in the neural

networks model are set to be 20 and 5, respectively. After training, we use other complete sets of force data to test the proposed classifier.

Table 5.9 Drilling State recognition using Neural Networks

Stage	Case One (919 Sample)		Case Two (6000 sample)		Force (N)
	Start Sample	End Sample	Start Sample	End Sample Time	
No Drilling (Rotating)	0	14	0	96	1.853
Start of Layer One	14	349	96	2695	3.2158
Start of Layer Two	349	596	2695	4263	5.364
Start of Layer Three	596	846	4263	5159	1.8535
End of Layer Three	846	846	5159	5159	5.235
Breakthrough	846	919	5159	6000	3.158

Table 5.9 shows the Drilling state recognition using Neural Networks; based on the selected thresholds; each drilling state is classified using experimental data as shown in Fig. 5.20; LM training algorithm is used in training ANN; parallel topology is used in connecting the layers of ANN to improve the overall performance.

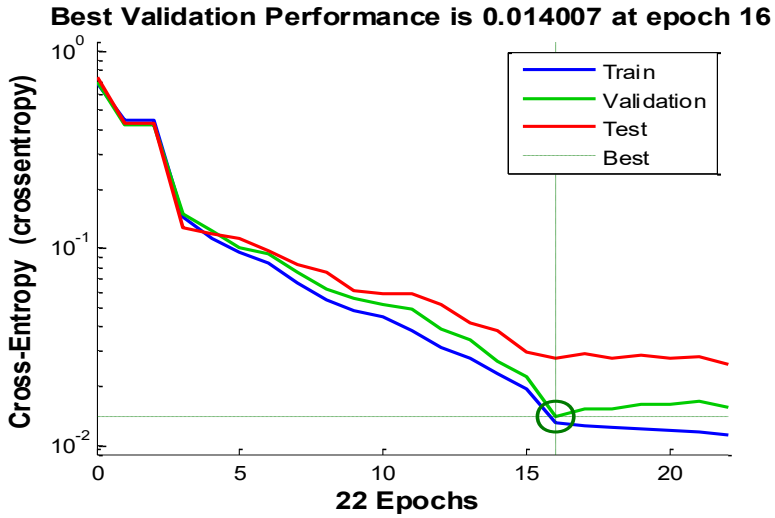

Figure 5.20 Neural Networks Training Performance

5.3.1 DETECTION OF DRILLING STATE USING ANN

Force data represented as Force with time series; Drilling experiments were done on the Human skull, and Force data is logged using Matlab routine. Force data used in Feature Extraction; when

the robot performs the drilling task, each drilling stage is defined, and the threshold is defined for five drilling stages as shown in Table 5.10.

Table 5.10 Robot State recognition using Neural Networks

Stage	**Case One (919 Sample)**		**Force (N)**
	Start Sample	**End Sample**	
No Drilling (Rotating)	0	14	1.853
waiting	14	349	1.41
Starting-drilling	349	596	5.364
Stopping drilling	846	846	1.8535
Retrieving	846	919	5.235

This classifier added to improve the overall design; where five outputs represent five states which are: rotation, waiting, starting-drilling, stopping drilling, and retrieving, the features for one complete set of force data as shown in Table 5.10. Force data used as training examples to train the NN model.

Figure 5.21 Drilling Tests on Human Skull using PA10 Robotic Manipulator

Skull thickness estimation is critical in neurosurgery; the main advantage is to minimize the error in skull drilling and could be used to optimize the skull drilling planner. Skull Drilling State Detection is a part of Neurosurgical Robotic System; the overall system has There is no golden standard for skull drilling. However, by performing drilling tests on the human skull; The maximum drilling torque is about 20 N·m; in the pressure of 20 N and rotating speed of 6000 rpm,

and the system power could satisfy the demand for drilling. Drilling pressure is one of the critical technical parameters of the skull drill. The bit pressure is 5 N/m. Robotic manipulator moves to the desired drilling position, then begin to drill the hole. When a hole completed, the robot automatically stops by using the neural networks drilling state recognition algorithm. After the data has been acquired from the first experiment, The Artificial neural network model constructed and the parameters threshold was applied. Three different criteria were chosen to evaluate the performance of the state detection, namely: relative mean squared error (RMSE), cross-correlation (CC), mean absolute error (MAE). The feed rate was fixed at 0.1mm/s and tested. In the experimental drilling tests, there was no unexpected failure, and the overshoots of all tests were well less than 1 mm. The proposed NN recognition algorithm gives the ability to perform partial Drilling tasks which could be useful in some orthopedic procedures.

5.4 INTERACTION CONTROL

Skull Drilling using an industrial robot requires keeping the contact force in such way that an excellent quality of the hole achieved and that the hole is in right position. Keeping the contact forces low will minimize the risk of having a position sliding when the robot is in contact with the skull. A sliding causing the hole to get oval and may damage the cutting tool because of the tangential forces that occur in the cutting tool. The goal of work is to set up the requirements of skull drilling task where high requirements for the drilling application exist in positioning accuracy, accessibility, cycle time and quality of the hole. The focus will be on the positioning and the contact forces, which are the most important requirements for neurosurgery system. The whole drilling process should be considered and be optimized.

5.4.1 FORCE CONTROL OF PA10 ROBOT TOOLTIP

Managing the interaction of a neurosurgical robot with the skull done by adopting motion control strategy with force control to allow the surgeon to guide the robot by leading its tool to the desired position. Force control algorithms can be roughly divided into direct methods and indirect methods. In the Direct control; The forces measured at the end-effector are used to directly calculate joint torques by using the transpose Jacobian of the robot arm. A proportional or integral term is used as a basic control law with this control methods. In the Indirect control; a position-

based control loop used to drive the robot joints based on both position error and force error; where force error is converted into a position error by using a mass-spring-damper representation; the resulting response causes the robot end effector to respond to forces in a manner like a mass-spring-damper system. In other words; The designed interaction control scheme could be one of the following:

- The control system should be able to control the robot position along the direction of the task space and the environment imposes natural force constraints
- The control system should be able to control the robot force along the direction of the task space and the environment imposes natural position constraints. In both cases; A force feedback system for a robotic neurosurgical system should be developed to fulfill the drilling requirements in skull structures and to minimize errors and uncertainties which may cause to an unstable behavior during the interaction.

5.4.2 DIRECT FORCE CONTROL

The direct force control scheme is the simplest form of force control where the controller acts on the error between the measured contact force and the desired force. The force loop is acting on the force error as shown in Fig 5. 22.

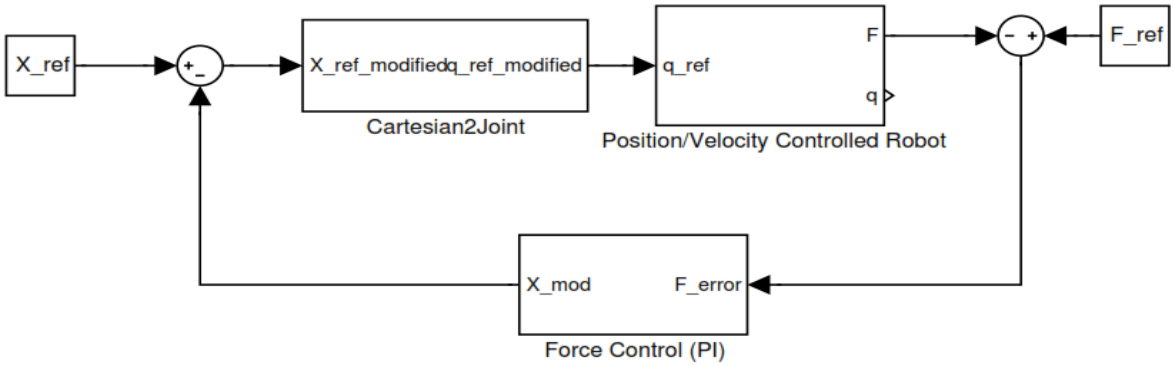

Figure 5.22 The structure of the position based direct force controller

The force controller will calculate the feedback distance error which used in the closed loop of the trajectory in Cartesian space. Inverse kinematics is used to convert the resulted Cartesian space to joint space and fed to the position controlled the robot. The force loop parameters selected to dominate over the position loop so that the force error goes to zero at the expense of the position error.

5.4.3 INDIRECT FORCE CONTROL

The force control law generated to accommodate position commands to the inner position controller. The controller input variables are the force/torque error e(k) and error variation de(k):

$$e(k) = f_a(k) - f_d(k) \quad (5.4.3.1)$$

$$de(k) = e(k) - e(k-1) \quad (5.4.3.2)$$

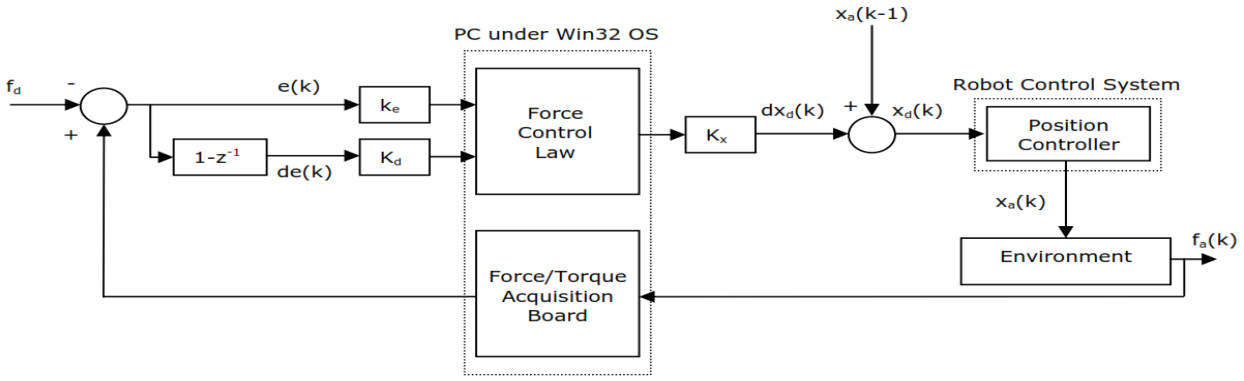

Figure 5.23 Indirect force control approach

Where fa(k) is the actual thrust force applied to the skull; and fd (k) is the desired Force which within the safety margin. The controller output is the position/orientation accommodation, assumed to be small.

5.5 SUMMARY

This chapter presents an optimized detection technique for human skull drilling system. The skull drilling device modeled by using SimMechanics Matlab toolbox; the proposed model is simulated to test the performance of skull drilling device. Neural network classifier is used for robot state recognition to optimize the performance of skull drilling control system and proofs that the system has good disturbance rejection. A neural network classifier presented which used to detect of bone layers; this technique improves the overall performance of the skull drilling and could be used in partial bone drilling.

CHAPTER 6 SIMULATIONS AND EXPERIMENTS

This chapter presents the simulation and experimental results of manipulator control algorithms: In the simulation part; The mechanism of building a modular design of a neurosurgical robotic manipulator is implemented by using multiple simulation tools such VRML, Matlab, and V-REP. This approach provides a unified framework for quick and cost-effective testing of control algorithms. A remote controller implemented in the Matlab/Simulink environment. Moreover, the System physical environment has been modeled on Virtual Robotics Experimentation Platform (V-REP). Matlab / Simulink controller synchronized with the V-REP by using ROS interface.

6.1 PA10-7C ROBOT SIMULATOR

Simulation has been recognized as an important research tool robotics. Different tools used for the analysis of kinematics and dynamics of robotic manipulators. Before we apply our ideas to a suitable control concept to the actual robot, a simulation should be done to avoid risky robot motions and perform a ground truth evaluation.

6.1.1 SIMULATOR OF PA10 ROBOT USING MATALB

A simulator is built for the PA107c robot to help in controller design. Robotic arm represented by a mathematical model using Matlab robot toolbox which helps in minimizing the number of modeling steps. The matrices obtained using robot kinematics allow to calculate the relative link's position for the PA10 robot structure and build a kinematics simulator. Fig. 6.1 shows the trajectory movement of the PA10 joints; smooth trajectory is chosen for each joint.

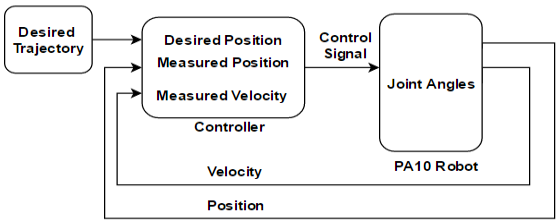

Figure 6.1 Matlab Simulation of PA10 Robotic Manipulator

Some useful functions added to the simulator such as joint limits; speed limit; acceleration limit and force modeling. This simulator is very helpful for both offline and online programming of the

robot which implemented on the real robot. This program also was able to output the signals to the PA107c robot, thus allows to prove in real time the drilling routine. The 3D simulator has many advantages such as:

- Debugging control code, simulating real robot movements
- Immersive virtual scenario for patients
- Fast development of neurosurgical training systems
- Integration of a redundant measuring system verify
- The actual position and increase the safety level

PA10-7C model is developed to define the kinematic and dynamic characteristics of a Mitsubishi PA10-7C 7 DOFs manipulator by using standard DH conventions.

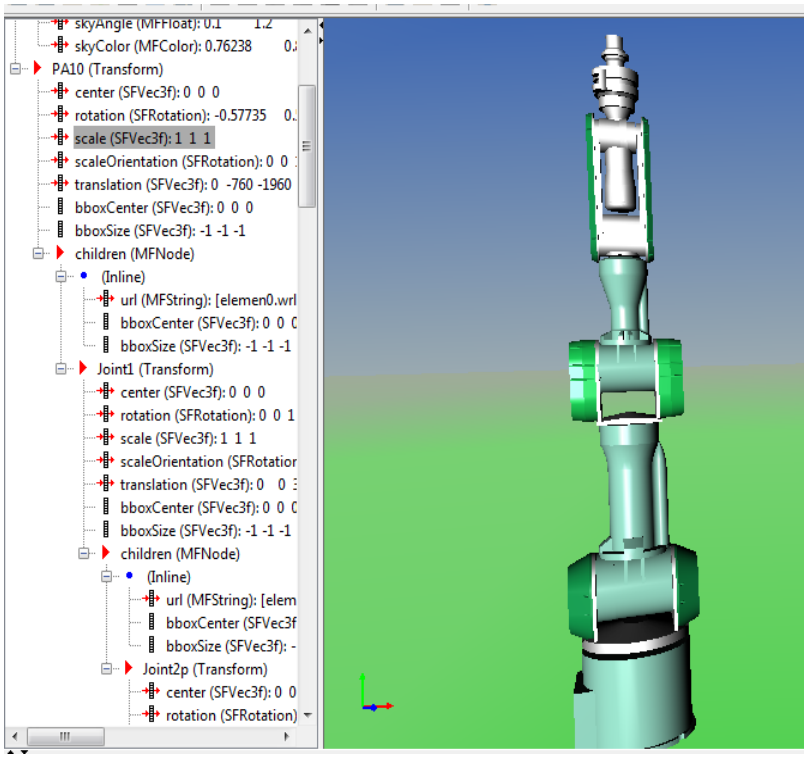

Figure 6.2 3D Simulator of the PA107c Robotic Manipulator

Kinematics provides a means to connect the operational space and joint space; where the position and orientation of the end effector are conveniently described in the operational space; while the robotic manipulator controlled in the joint space.

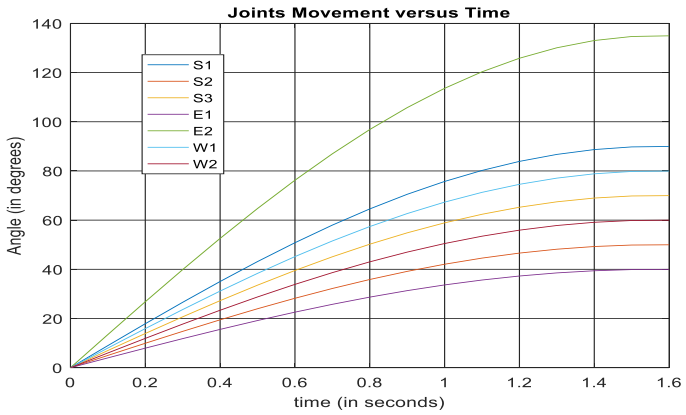

Figure 6.3 Trajectory movement for PA 10 joints

Inverse kinematics is a mapping from the operational space to the joint space; Jacobian matrix for PA10-7C robot arm is derived to describe the relationship between joint angular velocities in the joint space and the end effector's velocities in Cartesian space. Matlab model of PA10-7C built for simulating the motion behavior of the manipulator.

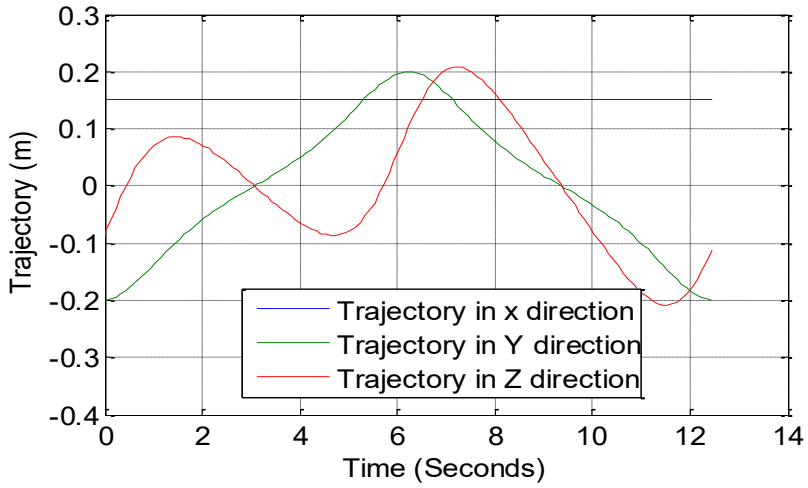

Figure 6.4 End Effector Trajectory for each direction

6.1.2 PA10 ROBOT MECHANICAL MODELLING USING SIMSCAPE

Classical control system analysis involves the use of differential equations to describe a physical system. To get the response of the system, we use the steps shown in Fig 6.5 to derive the mathematical model. Most physical systems include some combination of mechanical, electrical, and hydraulic components. Mathematical models used to predict system performance with considerable accuracy and adequate quality control of the materials used. Once differential equations describe the control system, it is desired to solve them for some control variable (output)

in response to the desired input function such as a step input, ramp input, or other function. Also, the control system may have some initial conditions. With the advent of total computer control, it is possible to use iterative techniques for the solution of control systems described by Laplace transforms.

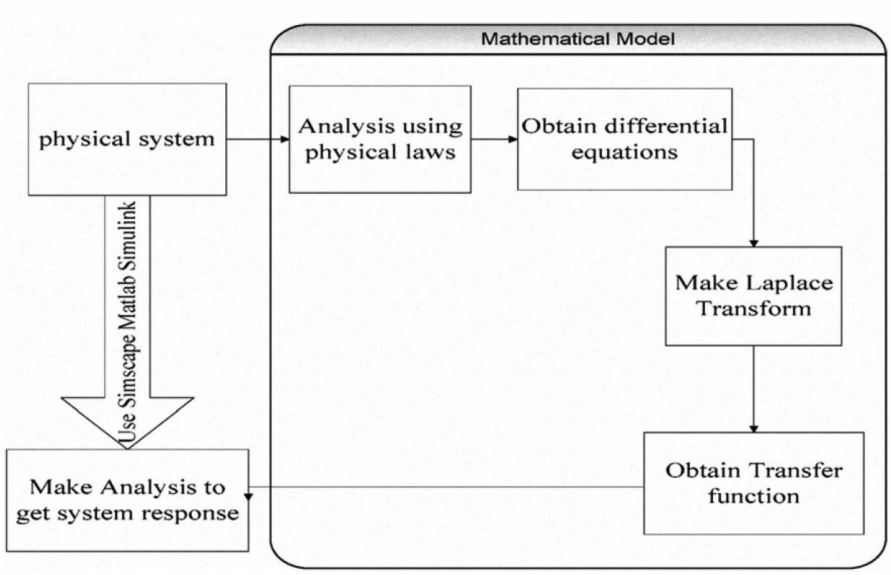

Figure 6.5 Steps of modeling mechanical system response

The physical model built using Matlab Simscape toolbox which used to get the response of the system without the need of deriving the mathematical model because the mathematical model for each physical component in the system is pre-calculated and derived, and it is easy to suit most of the real environmental conditions and restrictions such as change in initial conditions and operating conditions and fault cases [138]. The components of this library depends on using incremental model approach which is used for multi-input- multi-output system; When building a robotic model by using Simscape; two types of variables must be mentioned; [139] which are cross variables and through variables. Across variables such as angular velocity which directly connected with one another continue to share the same across variables. While through variable such as flow rate or torque are transferred along the Physical connection line is divided among the multiple components connected by the branches. For each through variable, the sum of all its values flowing into a branch point equals the sum of all its values flowing out.

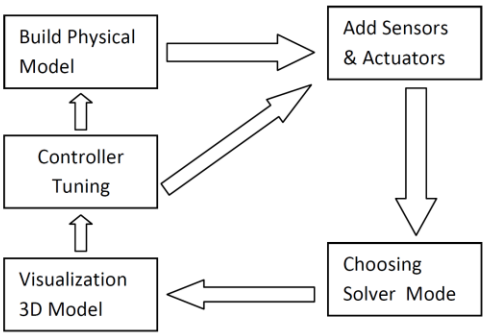

Figure 6.6 Steps for Modeling Mechanical Systems [140]

Steps needed to build and run a model representation of a robot shown in Fig 6.6 which start with building the physical Model and specifying body inertial properties, degrees of freedom, and constraints, along with coordinate systems attached to bodies to measure motions and forces. Sensors and actuators added to the system and solver mode for forces ; torques and initial conditions are chosen. Solver mode could be one forward or inverse dynamics model. The three-dimensional model of the robot is visualized and animated. The dynamics of the Mitsubishi PA-107c robotic manipulator is defined, and all unknown system parameters identified which used in the obtained dynamic model. Simscape Toolbox and Control Toolbox are used to test the robotic arm.

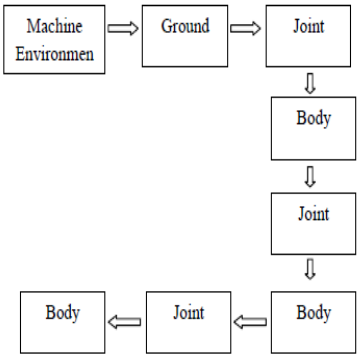

Figure 6.7 Mechanism of Building Dynamic Model using MATLAB Simscape

After drawing the mechanical model of motion platform by using computer aided design tools such as SolidWorks or AutoCAD; CAD translation tool could be used to transform geometric CAD assemblies into Simulink block diagram model. The CAD translation tool first exports the Assembly model from CAD platform into physical modeling file with XML extension. The

physical modeling file is then imported into Simulink, creating a SimMechanics model. Figure 6.8 shows the simplified Steps for generating Mechanical Model of the robot in CAD platform and model after it is converted into SimMechanics.

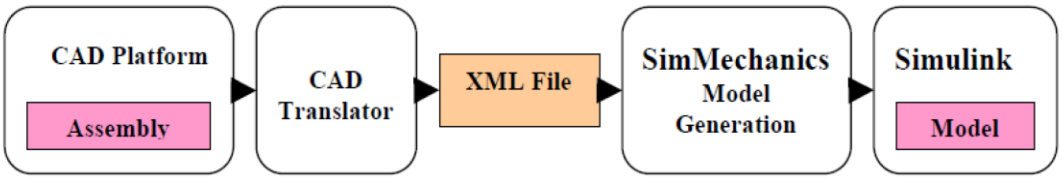

Figure 6.8 Steps for generating Mechanical Model of Motion Platform

The SimMechanics model is incorporated with inverse kinematics model for actuation control. The inverse kinematic model controls the actuators to extend and retract relatively to one another. The complete model allows the motion of the platform cues to be visualized. This also helps to test and validate the performance of inverse kinematics model. Fig 6.9 shows the construction of PA10-7c Dynamic Modeling using Matlab SimMechanics.

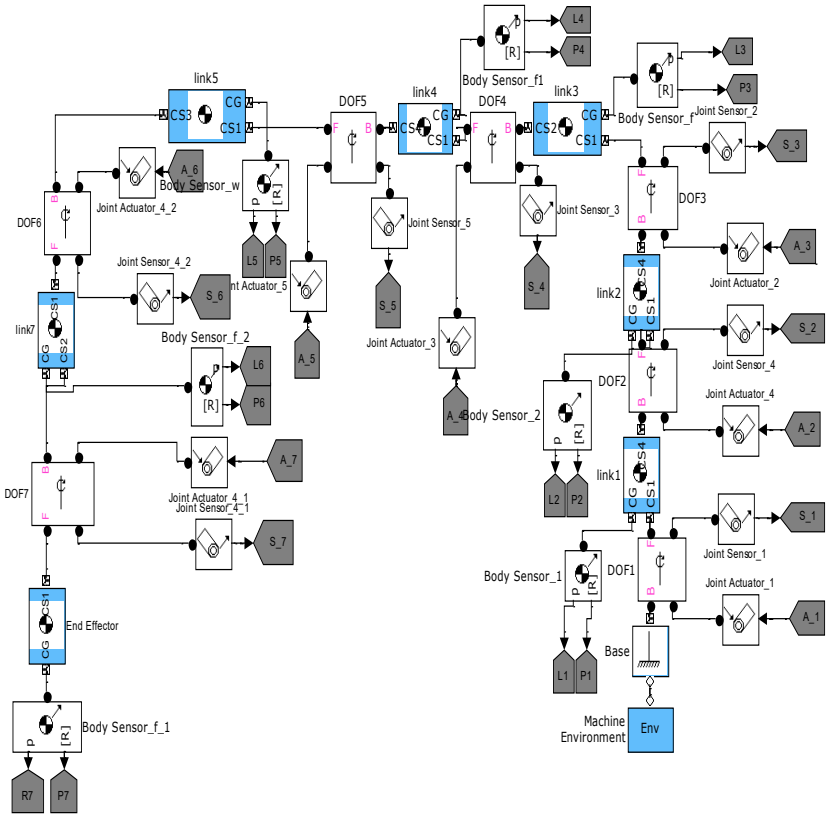

Figure 6.9 PA107c Dynamic Modeling using Matlab SimMechanics

6.1.3 DYNAMIC SIMULATION OF THE PA10 ROBOT BY USING V-REP

The software Virtual Robot Experimentation Platform (V-REP) is an open source 3D robot simulator utilizes a distributed control architecture either in threaded or non-threaded fashion [141] which makes V-REP perfect for multi-robot applications. VREP makes it possible to create 3D simulations imported from CAD design tools. V-REP have a physical engine implemented which can simulate physical properties like gravity and collision forces

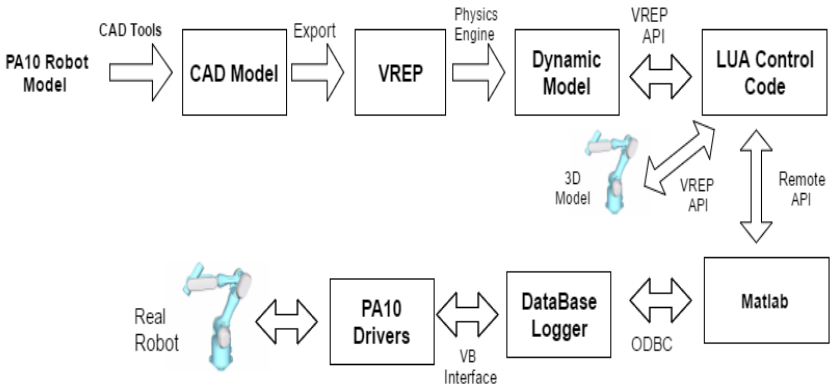

Figure 6.10 Workflow to create a dynamic model

The physics engine allows dynamic interaction between the robot and with the simulated environment and effect as friction and gravity are added to the simulation. V-REP offers a remote API allowing to control a simulation from an external application such as from Matlab [142]. In our experiment; the Remote controller is developed in Matlab Simulink while the actuation and physical interaction part created in V-REP. The general scheme of the developed simulator can be seen in Figure.6.9.

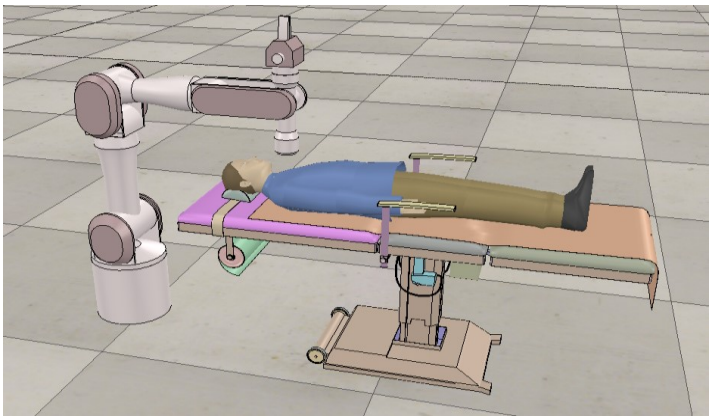

Figure 6.11 Simulation of Skull Drilling by using VREP

In the Simulink part; controller read feedback signals and send the control signals based on the required set point which sent back to V-rep. Matlab interact with V-REP via communication socket in a way that reduces lag and network load. Matlab code sends the drilling depth set point to V-REP and gets state data from it. LUA script in V-REP used to interact with Matlab and sends these values to appropriate child scripts. The visualization of processed data is done both in Matlab and V-REP application. The simulation model is controlled via LUA scripting language. Figure. 6.10 Shows the 3D representation of the PA10 robotic manipulator used in Skull Drilling procedure; Kinematic and dynamic model is interfaced with the 3D visualization model. Two major types of embedded scripts are written for PA10 robotic manipulator: Main simulation script and child scripts, child scripts handle low-level dynamic motor controllers. Matlab is an excellent tool developing and testing control algorithms, but Matlab lacks an easy to use 3D physical simulation engine. In this point, V-REP is one of the alternatives for 3D modeling and physical environment interaction testing. The interface of Matlab/Simulink with V-REP using ROS as communication middleware.

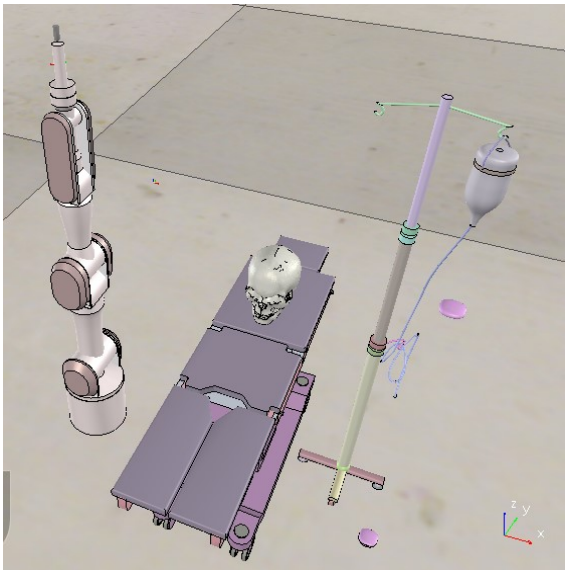

Figure 6.12 Simulation of Skull Drilling by using VREP

V-REP used to make robot simulation of the PA10 robotic arm to be used for skull drilling. In V-REP the movements are written in LUA code. In the development process of the control system, late modifications are costly and time-consuming. Virtual models could be used to find errors in the design before implementing it to the hardware. The dynamic model is used to predict the movements of the PA10 robot. To build a dynamic model of the neurosurgical drill; static CAD model exported to V-REP. by comparing the performance of both dynamic models; it could be

concluded that V-REP gives easier; faster and better visualization when compared with Matlab Simulink.

Table 6.1 Comparison between Dynamic Simulation using Matlab and V-REP

Feature	Matlab	V-REP
Dynamic Model Design	Good	Good
3D Model Design	Good	Very Good
Ability to Integrate with External Devices	Very Good	Good
Data Logging and Analysis	Excellent	Good
Learning Curve	Fast	Slow
Simulation Speed	Slow	Fast

6.2 EXPERIMENTS OF NEUROSURGICAL ROBOTIC SYSTEM

The objectives of the proposed Neurosurgical System are open source based; an easy user interface; embedded with safety algorithms; improve precision; enhance dexterity; enables work in confined spaces; enable remote participation, and eliminates hand tremor. Separated Monitoring and Control loops are used in the control architecture of the PA107c robot. The Objective of working on a Separated Monitoring and Control procedures is to minimize the sampling rate by using direct executable Matlab interface with PA107c robot implemented with separated Monitoring and Control procedures.

Matlab is connected with logging Database and sampling rate is 118m seconds; So it is better to use direct executable interfacing between Matlab and VB6; and try to read the current joint angles S1, S2, S3, W1, W2, E1, E2. Moreover, use that data in Simulink to move the 3D model of the PA10 robot. The desired prescribed drilling path is specified in the Cartesian space, as this is where the drilling specifications are easily described about the PA10 robotic manipulator environment and the skull. Then, given a table of desired points, the end effector should pass through the inverse kinematics stage to convert points from Cartesian space to joint space.

6.2.1 COMPONENTS OF PA10 ROBOTIC MANIPULATOR

The Mitsubishi PA10-7C robot arm has 7-Degree of Freedom (DOF). That can make complex movements. The benefits of a redundant robot are its flexibility and dexterity to move around, and capability of adding additional constraints to avoid obstacles and joint limits. However, to successfully, reliably, safely, and precisely control the robot, modeling and simulation are

necessary to find control strategies for the proposed system. To successfully execute tasks, the robot has to be controlled precisely, stable, and safely, either in free space or the Interactive environment. Consequently, modeling and control are the foundations of the Mitsubishi PA10-7C robot arm. Modeling and control deal with kinematic and dynamic modeling, Jacobian matrix derivation, and control of set-point and trajectory tracking.

Figure 6.13 shows the relation existing among each of the mentioned components; where the parts of PA10 are as follows:

- Level 1: robot manipulator mechanical system.
- Level 2: Servo drives to control the servo motors of each joint.
- Level 3: Motion control section (MCS); formed by the motion control and optical boards.
- Level 4: Operation control section (OCS); formed by the PC and the teaching pendant.

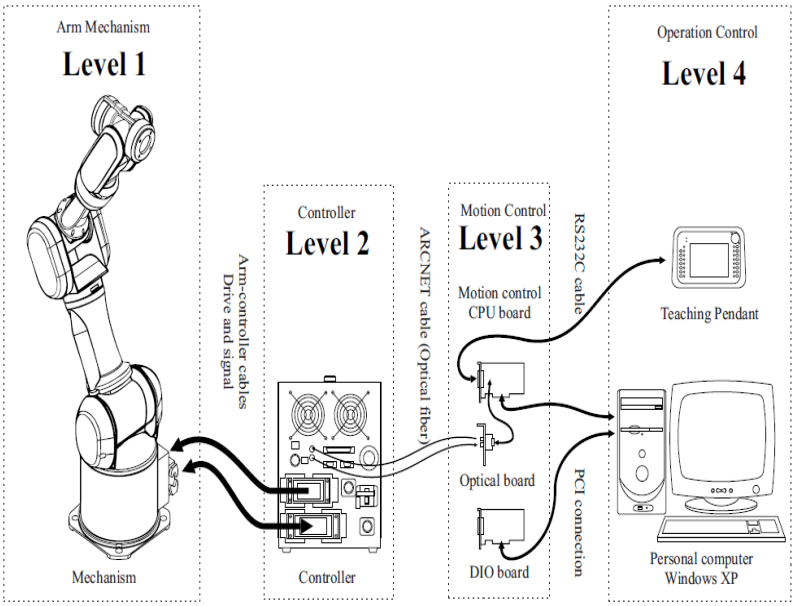

Figure 6.13 Components of PA10 Robotic Manipulator [49]

6.2.2 ARCHITECTURE OF THE NEUROSURGICAL MOTION SYSTEM

A flow chart for the implemented control system is shown in Fig 6.14. The host computer runs the Windows XP operating system, and we have been able to achieve communication rates of up to 200 Hz with the robot servo driver through the ARCNET LAN protocol.

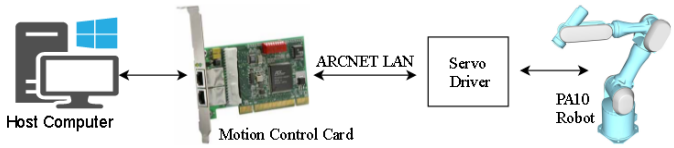

Figure 6.14 Flowchart of Mitsubishi PA-10 Four-layer control architecture

Visual Basic IDE is used for building the graphic user interface for the control system while Matlab is used for data analysis. TCP/IP protocol is used to communicate between Matlab, Visual Basic, 3D Slicer; and V-REP. The PA10 system is composed of four sections or levels, which conform a hierarchical structure. PA10-7C have 7 degrees of freedom industrial robotic manipulator manufactured by Mitsubishi by Mitsubishi Heavy Industries.

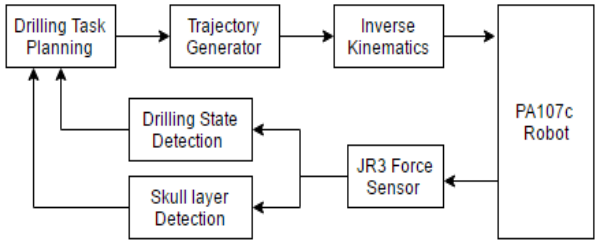

Figure 6.15 Architecture of the Neurosurgical Robotic System

The architecture of the robotic skull drill system is shown in Fig.6.15. It consists of the Mitsubishi PA-107C robotic manipulator, the robotic arm control box, master Computer; Client Computer; a JR3 force sensor; and joystick.

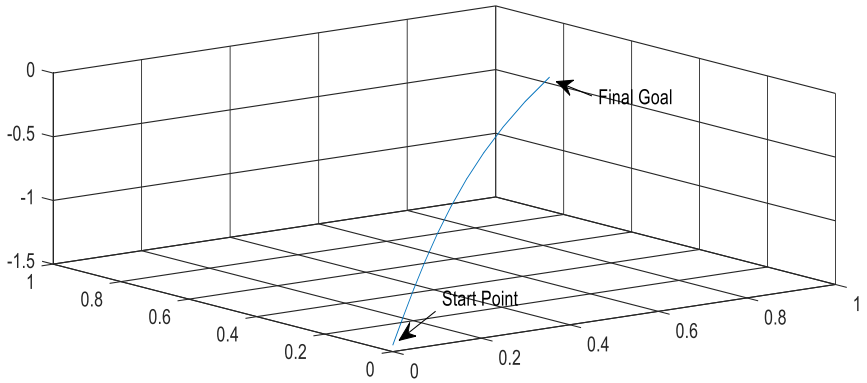

Figure 6.16 End Effector Trajectory

6.2.3 MOTION CONTROL OF THE NEUROSURGICAL ROBOTIC SYSTEM

The PA10-7c robotic arm controlled in three distinct phases. In the first step; the drill will follow the desired trajectory from its initial position to a particular position under position control without any interactions with the environment; Inverse kinematics is used to calculate the proper joint angles taken into account the robot constraints such as singularity and joint limits. In the Second step; Force steering is used to control the robot end effector to the drilling location; In the third step; Cartesian-based control apply the desired trajectory to the robot end effector regarding time histories of positions, velocities, and accelerations. Joint- based control schemes use these desired trajectories to the Joint inputs.

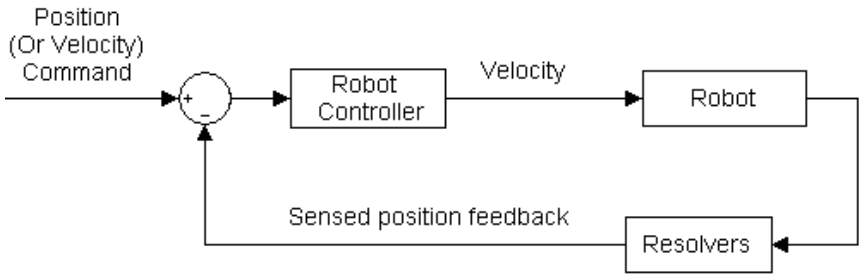

Figure 6.17 Existing PA10-7C Robot Control

The inverse kinematics performs the trajectory conversion. The errors in Cartesian space calculated as follows:

$$e_x(t) = x_d(t) - x(t) \qquad (6.2.3.1)$$

Where: $x_d(t)$ is the desired Cartesian trajectory and x(t) is the actual robot end-effector Cartesian space. The velocity command interpolated to form a letter "S" shape adjusting to the default speed.

6.2.4 NEUROSURGICAL SKULL DRILLING FORCE CONTROLLER

Skull Drilling using an industrial robot requires keeping the contact force in such way that an excellent quality of the hole achieved and that the hole is in right position. Keeping the contact forces low will minimize the risk of having a position sliding when the robot is in contact with the skull. A sliding causing the hole to get oval and may damage the cutting tool because of the tangential forces that occur in the cutting tool.

Force control algorithms can be roughly divided into direct methods and indirect methods. In the Direct control; The forces measured at the end-effector are used to directly calculate joint torques

by using the transpose Jacobian of the robot arm. A proportional or integral term used as a basic control law with this control methods.

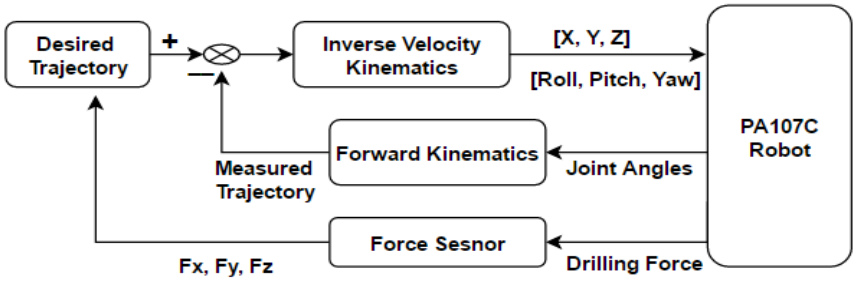

Figure 6.18 Force Control for Skull Drilling System

In the Indirect control; a position-based control loop used to drive the robot joints based on both position error and force error; where force error converted into a position error by using a mass-spring-damper representation; the resulting response causes the robot end effector to respond to forces in a manner like a mass-spring-damper system. Force signal is analyzed to determine the stage of operation. When the drill is contacted with the object initially, the changing of force is large, and the drill is activated. During drilling, the force will be maintained at a particular value by Force Controller. The drill automatically stops when the drilling finished. Otherwise, the brain will be damaged.

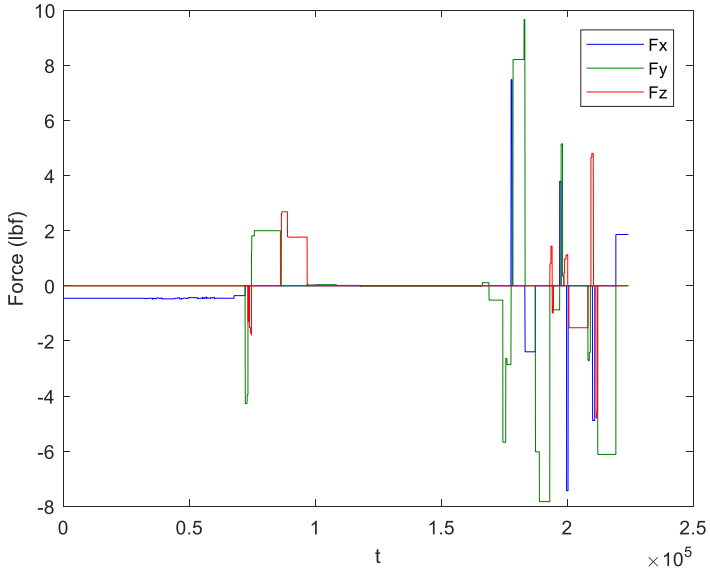

Figure 6.19 Robot Steering Force in X, Y,Z directions

Force control algorithms could also be divided into passive and active Force control. Passive force control is used when there is no force sensor. This force control needs special purpose compliant end-effectors for the desired task and can only cope with small misalignments [143].

Active force control is based on force measurements which are fed back to the controller. Impedance control or compliance control is the most common active force control approaches.

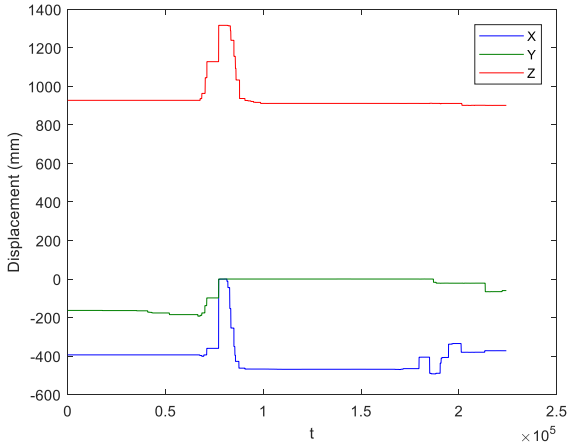

Figure 6.20 Robot End Effector Trajectory in the X, Y, Z Direction

Force controller reads the robot steering Force in X, Y, and Z directions as shown in Fig.6.19 and calculates the feedback distance error which used in the closed loop of the trajectory in Cartesian space. Inverse kinematics is used to convert the resulted Cartesian space to joint space and fed to the position controlled the robot. The force loop parameters selected to dominate over the position loop so that the force error goes to zero at the expense of the position error. The resulted robot end effector trajectory is shown in Fig. 6.20.

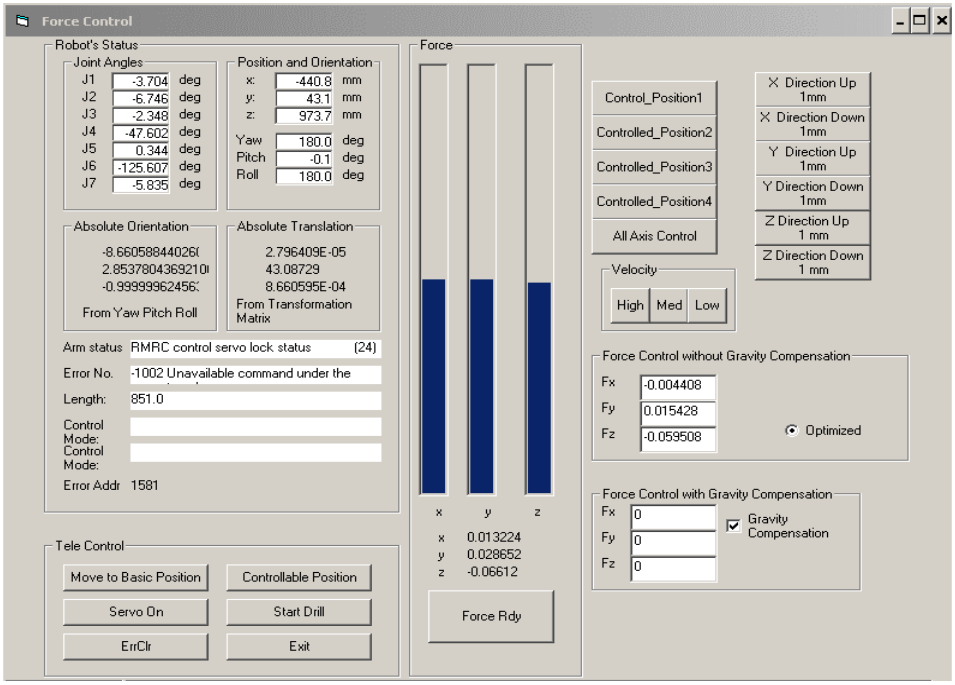

Figure 6.21 Implementation of Force Steering for PA107c Robotic Manipulator

Separated Monitoring and Control loops are used in the control architecture of the PA107c robot. The Objective of working on a Separated Monitoring and Control procedures is to minimize the sampling rate by using direct executable Matlab interface with PA107c robot implemented with separated Monitoring and Control procedures. The resulted sampling rate is 118m seconds; When Matlab connected with logging Database; So Direct interfacing between Matlab and VB6 is used to improve the system integration and overall performance; Sampling rate is minimized to 15 m seconds; Figure 6.22 shows the data monitoring and logging of PA107c Robotic Manipulator using Matlab.

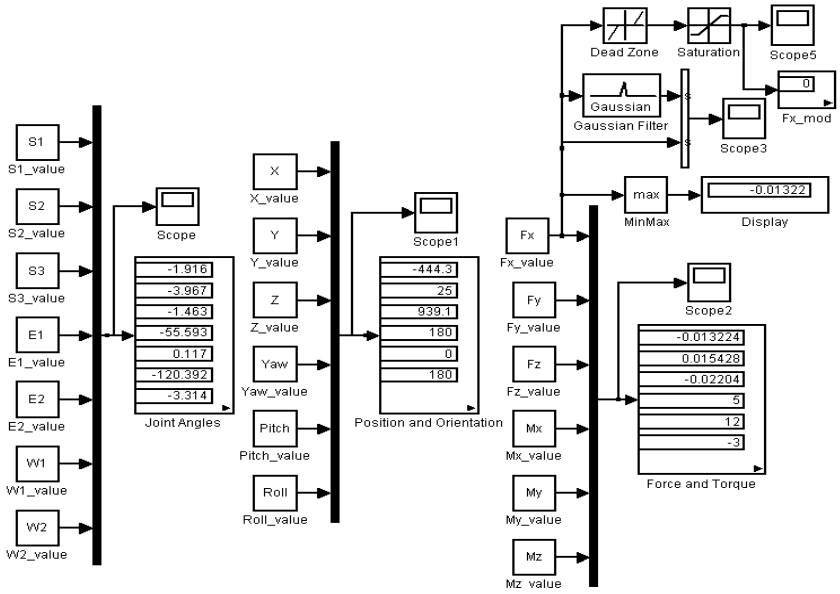

Figure 6.22 Matlab Interfacing with PA107c Robotic Manipulator

The software routines are written to synchronize the data obtained in the real robot with the simulation technique, so the simulator reads the data at the correct time. For safety consideration; the drill moves forward or backward according to the force signal applied to the drilling tool. When the force exceeds force threshold; the robot will not move the drill forward again. Different joint trajectories have been used on the real robot, and the joint positions and velocities were logged and compared with the results of robot simulator in real time.

6.2.5 MOTION CONTROLLER OF THE NEUROSURGICAL ROBOTIC MANIPULATOR

The aim of the motion controller is the determination of an allowable trajectory for all degrees of freedom of the robotic manipulator, where the desired trajectory is constrained with the design

requirement and mechanical system behavior. However; skull has three layers; So the force measured during drilling is not linear; which should be compensated to minimize the effect of the non-linearity.

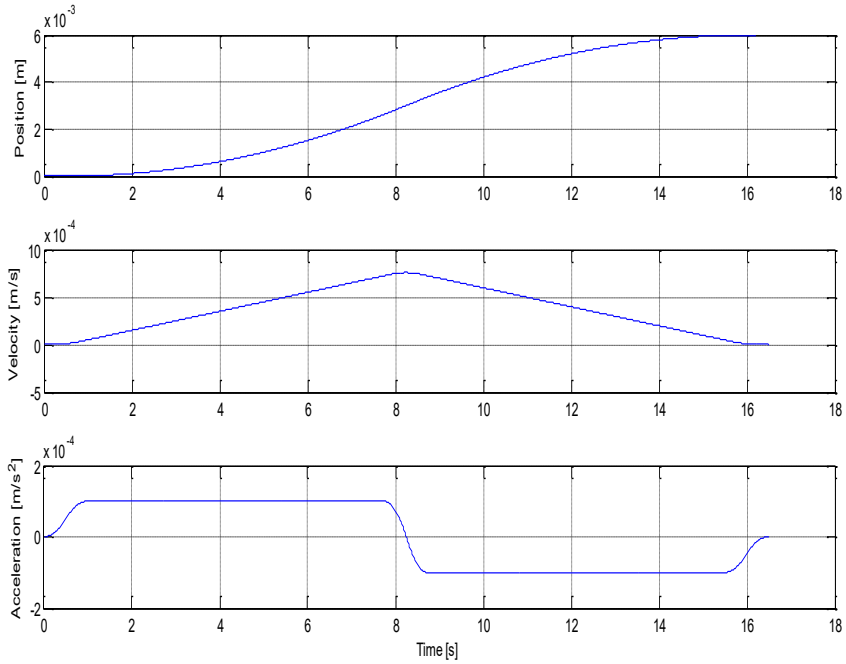

Figure 6.23 Skull Drilling Trajectory Profile

Feedforward Controller used to minimize the effect of systematic non-linearity; while the Feedback Controller used to compensate for unknown disturbances or unmodelled parameters; by using measurement to calculate the desired control signal. Drilling depth is controlled, and the target joint angles of the robot are adjusted to place the payload in the desired position based on knowing the deflections in the arm from the mathematical model's interpretation of the disturbance caused by the payload. The system is divided into the following parts:

- Drilling Task Planner: the Fuzzy logic algorithm is used to determine the allowable drilling thickness based on age; sex; race and other variables used to calculate an allowable drilling trajectory.
- Feedforward control: feedforward controller is used in defining the motion parameters which are: jerk; acceleration; velocity and position signal.
- System Compensation: filters are used to reduce unwanted behavior like known or measured disturbances and non-linearitys.

In the classical 'rigid-body feedforward controllers' the trajectory planning and feedforward control are usually done for each part separately, relying on system compensation and feedback

control to deal with interactions and non-linearity. However, the disadvantage of this approach is that position error is very large which is not safe for skull drilling; and the settling time is large which is not safe due to temperature effect on skull cells; and to overcome that many approaches could be used such as: using a more detailed model; Feedback control optimization; and Trajectory smoothing algorithms. Angle interpolation method is used to control the selected axis to the target angle; this method interpolates the velocity profile to form a letter "S" shape. The motion velocity is interpolated adjusting to the default velocity. Table 6.1 shows an example of robot movement in from initial conditions to reach the target position in Cartesian Space

Table 6.2 Settings of Initial Conditions and Cartesian Space

Robot Joint Space (degree)	**Target position in Cartesian Space**
$q_0(0)$ =0	X_d=23.2 mm
$q_1(0) = 32$	$Y_d = -10.51$ mm
$q_2(0)$=0	Z_d=730.34 mm
$q_3(0) = 50$	$Roll = 0$ degree
$q_4(0) = 0$	$Pitch$=0 degree
$q_5(0) = 45$	$Yaw = 0$ degree
$q_6(0)$=90	

6.2.6 FORCE SENSOR CALIBRATION

The force sensor is calibrated by using defined weights to derive the relationship between the sensor's reading and the weight. The original reading from the JR3 force sensor calibrated with experiment using the weight of a water bottle mounted on top of the robot, then water added gradually to the bottle, water volume measured, and finally; the average sensor's reading is calculated with respect to the corresponding water weight.

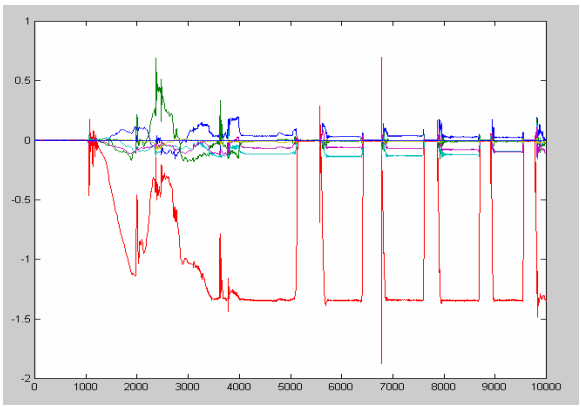

Figure 6.24 Plot of all the components of force/torque sensor readings.

After the above processing, the following result is concluded:

$$1000 \textit{ in sensor's reading} = 3.3581 \text{ lbs} = 1.5234\text{kg} \qquad (6.2.6.1)$$

Different joint trajectories have been used on the real robot, and the joint positions and velocities were logged and compared with the results of robot simulator in real time. The software routines are written to synchronize the data obtained in the real robot with the simulation technique, so the simulator reads the data at the correct time.

6.2.7 BLOCK DIAGRAM OF THE PROPOSED MOTION CONTROLLER

Fig. 6.16 shows the experimental setup of the robotic neurosurgical system. The robot arm moves the drill to the appropriate point on the patient's head, angles the drill perpendicular to the tangent of the skulls arc, moves the drill through the skull along the trajectories, and stops when the drill has perforated the skull. To make a robot do drilling task; Motion Planner is designed to generate the motion to be smooth.

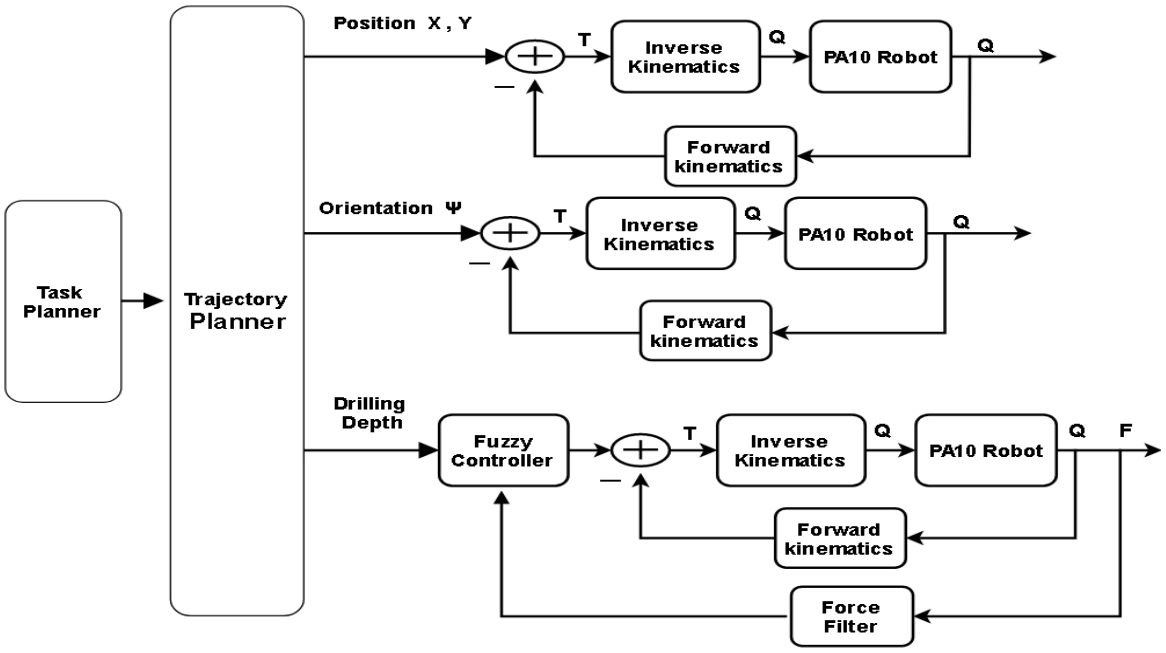

Figure 6.25 Block Diagram of the proposed Control System

A path is a geometric representation is started in Cartesian space and then converted to joint space by using inverse kinematics. During drilling, the linear velocity and acceleration of the drill have

to stay within certain limits. At the same time, the drill should be completed in the minimum possible time which is about 47 seconds to avoid temperature effect.

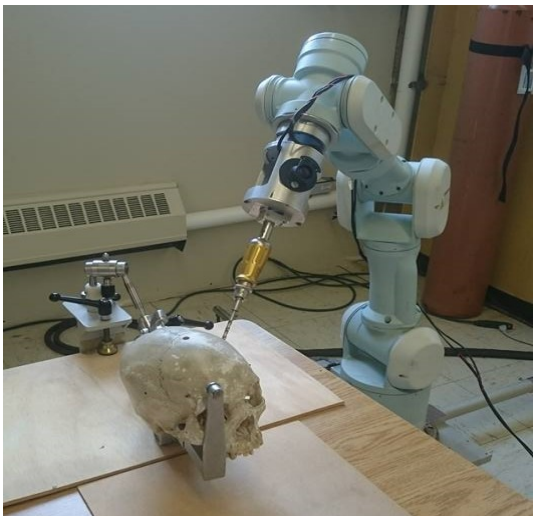

Figure 6.26 The Neurosurgical Robotic Drilling System

Drilling Task planner is used to determine the proper drilling trajectory. Where this is used in defining the motion parameters which are: jerk; acceleration; velocity and position signal. The goal of this work is to set up the requirements of skull drilling task where high requirements for the drilling application exist in positioning accuracy, accessibility, cycle time and quality of the hole.

6.3 HUMAN-ROBOT INTERACTION IN NEUROSURGICAL INTERVENTIONS

Human-robot interaction is defined as the study of humans, robots, and the ways they influence each other [144]. In the case of neurosurgical robotics; there are many considerations must be taken into account when developing human-robot interaction [57] such as Multimodality, Adaptivity, Level of autonomy, and Cooperativeness. Multimodality is the ability to allow switching seamlessly between voice to force steering, according to the changes in context. While adaptivity is the ability to perform distinct levels of human-robot interactions based on surgeon preference. Level of autonomy should be matched with health care policies and requirements; High-level perception and task planning should be developed to correct user misconceptions or reject user's request in the cause of error detection. Cooperativeness should be implemented in high-level task planner of the robotic manipulator; So, the robot can cooperate with the surgeon to achieve system goals. Reactive and Proactive control features should be added to the robot; So,

the robot can realize the safety boundaries and its capabilities to stop operation in safely way in case of unplanned situations. Figure 6.27 shows an Overview of the designed interaction system.

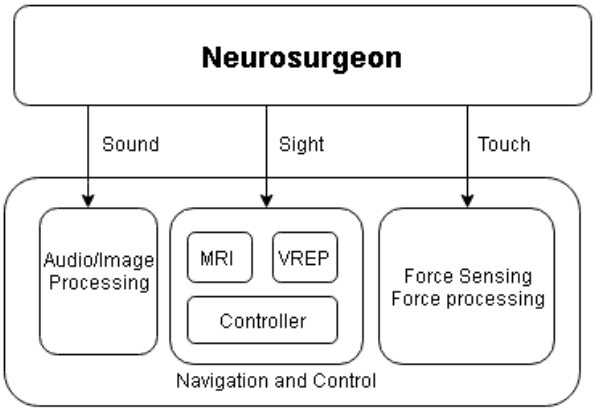

Figure 6.27 Overview of the proposed Human-robot Interaction

Kinect is a motion sensing input device by Microsoft for video game consoles. Kinect motion sensor provides researchers with a facility to do 2D and 3D motion and gesture detection and skeleton motion tracking. Kinect sensor is used to recognize the gesture to improve the human-robot interaction. Depth data is retrieved using software development kit provided by Microsoft.

Figure 6.28 Structure of Kinect Sensor

Data is analysis using Matlab to define depth measurement of the area of interest in the workspace and comparing that with safety measures defined by experimental setup. The Kinect device returns data streams shown in Table 6.3. The image stream returns color image data; raw YUV format is selected which has resolution of 640 x 480 pixels and frame rate of 15 frames per second. The depth stream returns the depth map measured in millimeters from the camera plane. The tracking ranges are a default range of 50 cm to 400 cm. Intel(R) CS431 camera mounted on the side of robot tooltip to improve the reliability of the neurosurgical system and gives the ability to visually track the robot workspace.

Table 6.3 Data streams retrieved from Kinect device

Data streams	Return Data	Data Format	Sampling
Image stream	by the color sensor	RawYUV_640x480	15 frames per second
Depth stream	by the depth sensor	Resolution of 640 x 480	30 frames per second
Skeletal stream	by the depth sensor	the position of the skeleton	15 frames per second
Audio stream	by the audio sensor	16-bit sampling size	44100 Hz sampling rate

6.3.2 MEDICAL IMAGE ANALYSIS AND VISUALIZATION TOOLS

Medical robots are increasingly popular in surgical procedures. Collaboration between the robot-assisted interventions and surgical planning software require a full range of tools and methods developed in robotics and medical image computing fields.

Table 6.4 List of Open Source Medical Image Analysis Software's

Name	Image Types	Usage
Visualization Toolkit (VTK) [145]	biomedical image	Image processing, visualization
Insight Segmentation, Registration Toolkit (ITK)	biomedical image	Registering and segmenting
FMRIB Software Library (FSL) [146]	FMRI, MRI, and DTI brain imaging	Brain Imaging
Statistical Parametric Mapping (SPM) [147]	fMRI, PET, SPECT, EEG and MEG	functional brain imaging
Graphical Interface for Medical Image Analysis and Simulation (GIMIAS) [148]	biomedical image	automatic segmentation, visualization,
3D Slicer [148]	biomedical image	Visualization and image analysis
Medical Image Analysis (MIA) [149]	biomedical image	gray scale medical image analysis
The Medical Imaging Interaction Toolkit (MITK) [150]	medical imaging	Visualization, image analysis

Image-guided neurosurgical robotics allow surgeons to improve accuracy; Enables new minimally-invasive procedures, increases the speed of surgical procedures, shortens hospital stays, decreases long-term costs. Table 6.3. Summarize the open source Medical Image Analysis software.

6.3.3 IMAGE VISUALIZATION AND NAVIGATION USING 3D SLICER

The goal of neurosurgical planning is to integrate image information from multiple sources, define the properties of the structure, and determine the best surgical approach. Medical Imaging such as CT scan, MRI, ultrasound, X-ray, fluoroscope usually used in the preoperative diagnostic imaging and the intraoperative planning.

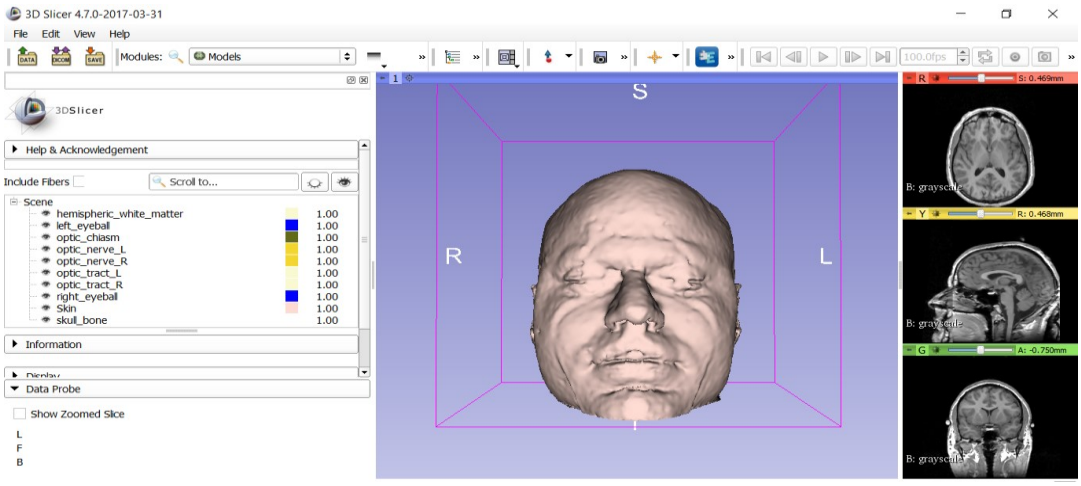

Figure 6.29 Visualization of Medical Images using 3D Slicer

Several types of MRI could be used in the preoperative diagnostic imaging such as 3D SPGR MRI, FLAIR MRI, language and motor functional MRI (fMRI) imaging, and diffusion tensor imaging (DTI). However; Ultrasound imaging mostly used in the inter-operative planning.

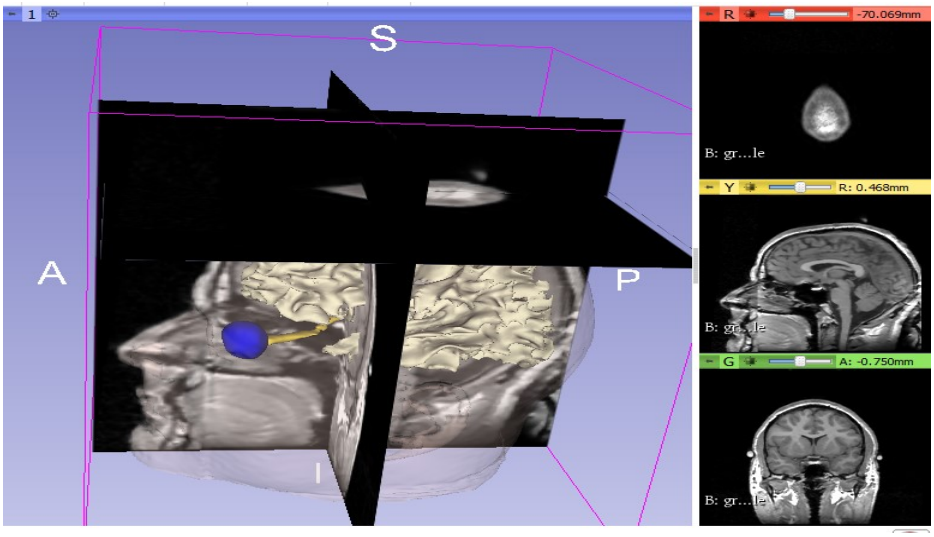

Figure 6.30 Image Visualization and Navigation using 3D Slicer

3D Slicer is a comprehensive open-source platform for Multimodality medical imaging including, MRI, CT, US, nuclear medicine, and microscopy. 3D Slicer built on top of VTK, ITK, CTK, Qt, Tcl/Tk, Teem, Python, DCMTK, JQPlot. 3D Slicer has extensive support for Image Guided Therapy including Visualization, Registration, Segmentation, Model making, Diffusion Tensor Imaging, Quantification, Filtering, and Interfacing to imaging devices. Fig 6.30. Shows the visualization of the medical images using 3D Slicer.

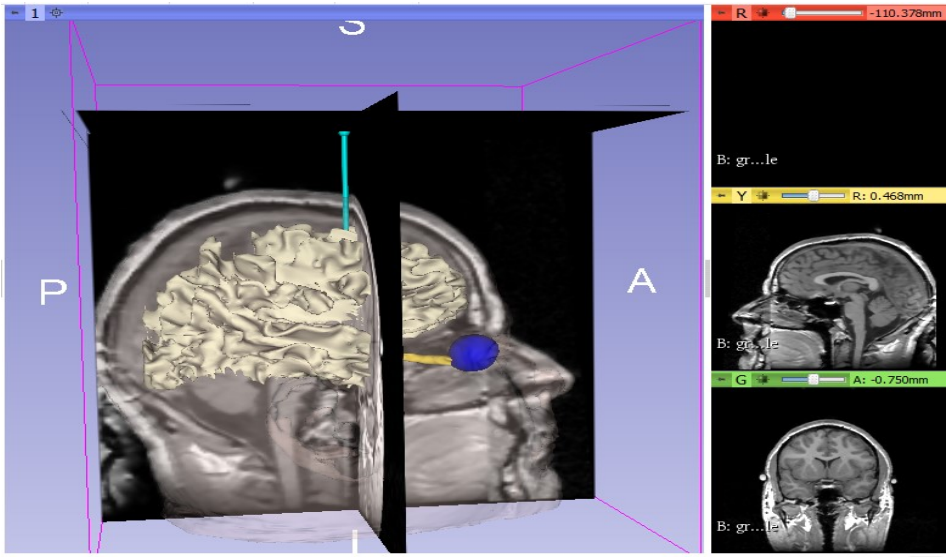

Figure 6.31 Tracking Skull Drilling using 3D Slicer

Patient MRI images data set imported to 3D Slicer for visualization and pre-planning; Fig 630 shows the three pre-computed MRML scenes Tracking data imported to the 3D Slicer software using Open-IGT Link which is an open-source network communication interface designed to provide a unified real-time communications (URTC) for sensors, surgical robots, and computers. Control commands send to the robotic manipulator using the TCP/IP protocol.

3D Slicer is used in both Pre-Surgical Planning and Intraoperative Planning which done by exploration of biomedical images to define the target selection, cutting plane; safety requirements and setting these points in the image coordinate system. Surgical Registration techniques used to transform the surgical planning from the image coordinate system (ICS) into the patient coordinate system (PCS) as shown in equation 6.3.3.1.

$$PCS_{3\times1} = R_{3\times3} \cdot ICS_{3\times1} + t_{3\times1} \quad (6.3.3.1)$$

$$\begin{bmatrix} PCS_x \\ PCS_y \\ PCS_z \\ 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} ICS_x \\ ICS_y \\ ICS_z \\ 1 \end{bmatrix} \quad (6.3.3.2)$$

3D Slicer uses Right Anterior Superior (RAS) coordinate system; while VTK image is in IJK coordinate system. This registration should define on the robot coordination system (RCS) by calculating the rotation and transformation between each coordinate system with respect an absolute coordination system calibrated at the Operation room (OR).

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & A_{13} & t_1 \\ A_{21} & A_{22} & A_{23} & t_2 \\ A_{31} & A_{32} & A_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} i \\ j \\ k \\ 1 \end{bmatrix} \quad (6.3.3.3)$$

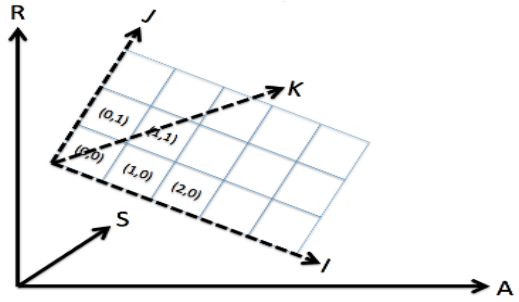

Figure 6.32 Description of (RAS) and (IJK) coordinate systems

6.3.4 COOPERATIVE POSE REGISTRATION FOR NEUROSURGICAL ROBOTIC SYSTEMS

Point-based registration approach used in this system. Human-robot interaction technique to the image coordinate system. Iterative Closest Points approach is used to determine transfer is used to automatically select the corresponding points in the patient coordinate system with respect motion matrix that related PCS with ICS. A brain atlas and image registration used in predicting the locations of brain structures. 3D Slicer used for Image Segmentation and Navigation; Navigation is an essential component of the interactive surgery control system. 3D Slicer provides visualization to the surgeon during the procedure. Operation Tracking consists of two parts; Localization and mapping which aim to track surgical tools and anatomical structures during the neurosurgical procedures; Camera i mounted on the tool-tip of the robotic arm. Communication Interface and synchronization needed to enable data exchange between different tools; Matlab integrated with 3D Slicer using OpenIGTLink which developed by Perk Lab. at Queen's

University. The imaging device could be used to track surgeon's movement during the surgery and could be used as constraints to the robot obstacle avoidance algorithm. Localization and mapping of linear surgical instruments commonly applied in multiple of medical interventions. Many surgical applications require location registration to define the targeted area of interest while minimizing damage to adjacent structures. A coordinate measuring Machines (CMM) used for measuring the physical geometrical characteristics of an object in six degrees of freedom. Basically; location registration is a method to the link between a preoperative surgical plan on the real surgical space. There are multiple registration methods used in surgical interventions; One is fiducial point-based registration method, and the other is the anatomical landmark combined with surface-based method. A position marker and pointer tools are used in almost every medical navigation systems; There are several types of position marker such as an electromagnetic sensor, or optical marker. The pointer tip is localized relative to the pointer marker. Transformation matrices used to relate between point pairs.

6.4 INTERACTIVE CONTROL OF NEUROSURGICAL ROBOTIC ARM

Force steering would be precious to the operator at the Operation Room which helps in obtaining easier environmental perception and eliminate the need for using haptic devices. The force sensor can be utilized in human-robot coordination, teleoperation, and collaborative robots. Human-robot interaction mediated by a force sensor which is the primary control input for the operator.

Table 6.5 Experimental results for measuring the Gravity Effect

Case	End Effecter Position (mm)			End Effector Rotation (degree)			Force Measurement (N)		
	X	Y	Z	Yaw	Pitch	Roll	F_x	F_y	F_z
1	-0.024	0	1317	0	0	0	-0.017	0.04	0
2	382.7	267.9	927.8	180	0	-145	0	0	2.8
3	-104.7	-70.6	814.3	34	-90	0	-0.22	-3.55	1.5
4	-395.6	-229.5	934.9	-176.5	-1.1	-145.2	0.13	-0.189	2.85
5	-437.2	-257.9	957.2	-174.9	-47	-148.9	0.055	-2.75	2.45
6	-467.4	-0.9	927.4	-180	0	-179.9	0.07	0.25	2.83

Table 6.6 Experimental results for measuring the robot end-effector joints

Case	PA107c Robot Joints						
	S1	**S2**	**S3**	**E1**	**E2**	**W1**	**W2**
1	0	0	0	0	0	0	0
2	21.8	-6.278	15.795	-54.833	-1.954	-119.024	1.532
3	28.3	53.964	0.82	-114.97	12.04	-29.5	-9.853
4	16.825	-5.311	14.857	-54.903	2.499	-119.039	-1.971
5	16.966	-5.298	14.705	-54.9	3.984	-73.085	-1.8505
6	0.07	-6.076	0.05	-54.9	0	-118.975	-0.004

The force sensor mounted at the robot end effector; Force/Position Control algorithm is developed to guide the robot using the force measurement in the X direction, Y direction, and Z direction to create straightforward and efficient human- robot interaction. Figure 6.33 shows the End Effector Force Measurement without Gravity Compensation; which is measured without any interaction or steering with the robot.

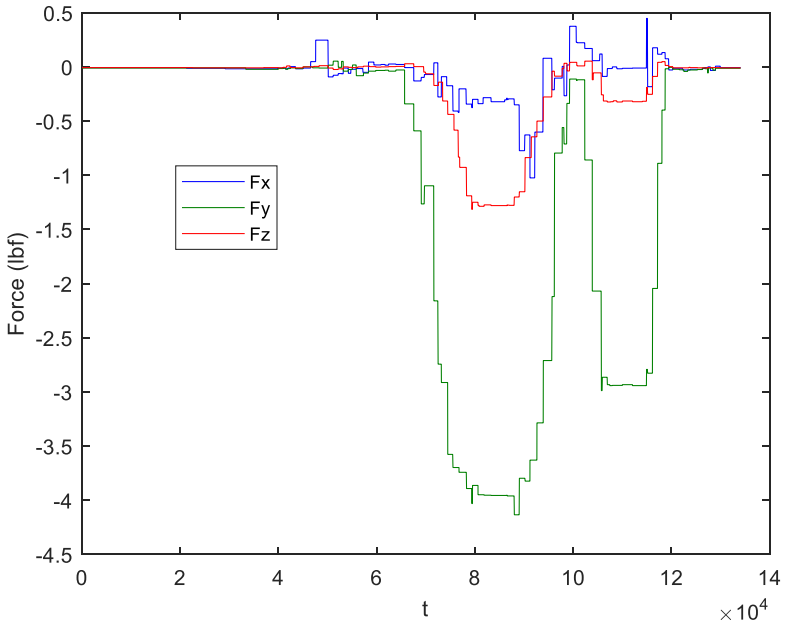

Figure 6.33 End Effector Force Measurement without Gravity Compensation

$$F = f + n + G \tag{6.4.1}$$

Where F is the total measured steering force which combined of human force f, with Gaussian noise n and non-linear offset due to the gravity effect G as an influence of changing the end effector pose as verified and listed in table 3. Multi-axis force sensor used to measure all six axes (three forces and three torques). Force sensor is mounted on the end-effecter of the PA107c robotic manipulator.

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} f_x & n_x & G_x \\ f_y & n_y & G_y \\ f_z & n_z & G_z \end{bmatrix} \tag{6.4.2}$$

Where $F_x, F_y,$ and F_z are the measured human steering force on x axis, y axis, and z axis respectively. $n_x, n_y,$ and n_z are the noise component in each direction. G_x, G_y and G_z are the estimated gravity component in each direction. Gravity effect is compensated by measuring the value of the initial force in real time and remove that as following

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} f_x & n_x & G_x - f_{x_{(inital)}} \\ f_y & n_y & G_y - f_{y_{(inital)}} \\ f_z & n_z & G_z - f_{z_{(inital)}} \end{bmatrix} \tag{6.4.3}$$

The gravity compensation is a calibration variable used to minimize the dynamic error which required to be eliminated in force steering. Moving average technique is used to calculate the gravity compensation $f_{(initial)}$ in x, y, and z direction to ensure that variations in the force mean are aligned with the variations in the data rather than being shifted in time; the experiment used average data of every 10 samples from every channel.

$$f_{(initial)} = \frac{f_M + f_{M-1} + \cdots + f_{M-(j-1)}}{n} \tag{6.4.4}$$

$$f_{(initial)} = \frac{1}{j}\sum_{i=0}^{j-1} f_{M-i} \tag{6.4.5}$$

Where $f_M, f_{M-1}, \ldots, f_{M-(j-1)}$ are the force measurement in sequence. The force value is used to define the distance Δ between the current position of the manipulator's end-effecter and the target position. The resulted robot displacement could be calculated as following

$$\Delta = D * C * F \tag{6.4.6}$$

Where Δ is the resulted displacement, D is the displacement direction matrix, C is the Compliance Factor depends on the required velocity. The displacement direction is generated according to the input steering force direction and speed to target point by the operator command as following

$$D = \begin{cases} D_x = 1, & if \quad |F_x| > |F_y| \text{ AND } |F_x| > |F_z| \\ D_y = 1, & if \quad |F_y| > |F_x| \text{ AND } |F_y| > |F_z| \\ D_z = 1, & if \quad |F_z| > |F_x| \text{ AND } |F_z| > |F_y| \end{cases} \tag{6.4.7}$$

The displacement direction matrix D is 3*3 diagonal matrix used to improve the steering accuracy by restricting the robot end effector movement in only one direction in every cycle; this simple restriction improved the overall all system performance and minimized the unwanted side force components and increased system stability.

$$D = \begin{bmatrix} D_x & 0 & 0 \\ 0 & D_y & 0 \\ 0 & 0 & D_z \end{bmatrix} \tag{6.4.8}$$

Humans and robots in shared work areas must meet high safety standards to eliminate any risk of human injury. Safety of the human-robot interaction without protective barriers is secured by determining the dynamic allowable interaction force range and represented by Compliance Factor C matrix. The maximum value of $C_x, C_y,$ and C_z depends on working conditions and determined by consulting the safety standards and designed for robots and assistance systems. There is a trade off between steering speed and accuracy; Gain Scheduling is added to change values of $C_x, C_y,$ and C_z to maximize the steering speed and maintain accuracy.

$$\begin{bmatrix} \Delta_x \\ \Delta_y \\ \Delta_z \end{bmatrix} = \begin{bmatrix} D_x & 0 & 0 \\ 0 & D_y & 0 \\ 0 & 0 & D_z \end{bmatrix} * \begin{bmatrix} C_x & 0 & 0 \\ 0 & C_y & 0 \\ 0 & 0 & C_z \end{bmatrix} * \begin{bmatrix} f_x & n_x & G_x - f_{x_{(inital)}} \\ f_y & n_y & G_y - f_{y_{(inital)}} \\ f_z & n_z & G_z - f_{z_{(inital)}} \end{bmatrix} \tag{6.4.9}$$

Measurement noise n is represented by Gaussian probability density function by

$$n_{(k)} = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(k-\mu)^2}{2\sigma^2}} \tag{6.4.10}$$

Where k represents the grey level, μ the mean value and σ the standard deviation. Due to real-time response requirements; simple noise filter is implemented using a dead zone with saturation limiter in which output becomes zero when the input crosses certain limiting value.

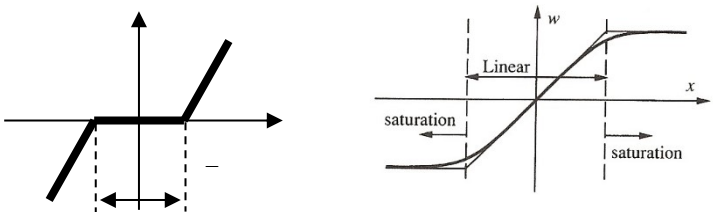

Figure 6.34 Steering Force Signal Conditioning using Dead zone and Saturation limit

Algorithm. 1 Algorithm for Force/Position Control for Human/Robot Interaction

Input $F_x, F_y,$ and F_z

Output $\Delta_x, \Delta_y,$ and Δ_z

1. Read Steering Force Measurement $F_x, F_y,$ and F_z
2. Calculate the Gravity effect using Moving Average technique for the last ten samples

$$f_{(initial)} = \frac{f_M + f_{M-1} + \cdots + f_{M-(j-1)}}{n} = \frac{1}{j}\sum_{i=0}^{j-1} f_{M-i}$$

3. Calculate Gravity compensation by subtracting $f_{(initial)}$ from Force Measurement $F_x, F_y,$ and F_z as following:

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} G_x - f_{x_{(inital)}} \\ G_y - f_{y_{(inital)}} \\ G_z - f_{z_{(inital)}} \end{bmatrix}$$

4. Apply Noise Filtering for Force measurement using Dead zone

$$\begin{cases} F_x = 0, & if \quad |F_x| < n_{(k)} * \left|f_{x_{(inital)}}\right| \quad \text{else } F_x = F_x \\ F_y = 0, & if \quad \left|F_y\right| < n_{(k)} * \left|f_{y_{(inital)}}\right| \quad \text{else } F_y = F_y \\ F_z = 0, & if \quad |F_z| < n_{(k)} * \left|f_{z_{(inital)}}\right| \quad \text{else } F_z = F_z \end{cases}$$

Where: $$n_{(k)} = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(k-\mu)^2}{2\sigma^2}} \leq 0.1$$

5. Determine the displacement direction D using the following condition

$$D = \begin{cases} D_x = 1, & if \quad |F_x| > \left|F_y\right| \text{ AND } |F_x| > |F_z| \\ D_y = 1, & if \quad \left|F_y\right| > |F_x| \text{ AND } \left|F_y\right| > |F_z| \\ D_z = 1, & if \quad |F_z| > |F_x| \text{ AND } |F_z| > \left|F_y\right| \end{cases}$$

6. Calculate Compliance factor C_x, C_y, and C_z using Gain Scheduling and saturation to improve system response and increase safety level
7. Calculate the required robot Displacement Δ based on steering force as following

$$\Delta = D * C * F$$

$$\begin{bmatrix}\Delta_x\\ \Delta_y\\ \Delta_z\end{bmatrix} = \begin{bmatrix}D_x & 0 & 0\\ 0 & D_y & 0\\ 0 & 0 & D_z\end{bmatrix} * \begin{bmatrix}C_x & 0 & 0\\ 0 & C_y & 0\\ 0 & 0 & C_z\end{bmatrix} * \begin{bmatrix}f_x & n_x & G_x - f_{x_{(inital)}}\\ f_y & n_y & G_y - f_{y_{(iital)}}\\ f_z & n_z & G_z - f_{z_{(inital)}}\end{bmatrix}$$

8. Calculate the new end effector position of the robotic manipulator

$$\begin{bmatrix}X\\ Y\\ Z\end{bmatrix} = \begin{bmatrix}X + \Delta_x\\ Y + \Delta_y\\ Z + \Delta_z\end{bmatrix}$$

9. Return to Step 1 Move robot

6.4.1 SPEECH CONTROLLED ROBOTIC SYSTEM

A speech recognition module is added to the system to make PA107c robotic manipulator function more naturally and can be considered as partners for the neurosurgeon not just as mere tools. To achieve this goal, Speech engine is configured and integrated into the robotic neurosurgical system. Speech engine contains a speech synthesizer, a speech recognizer, and grammatical knowledge base. Speech engine enables the PA107c robotic manipulator to recognize and speak in the English language. Windows Speech Application Programming Interfaces (SAPI) used to convert text into speech (TTS) with high flexibility to change the voice command speed. SAPI is a speaker-dependent tool which trained and used in Automatic Speech Recognition. SAPI allows the user to train any word(s) in any language to control the robotic manipulator.

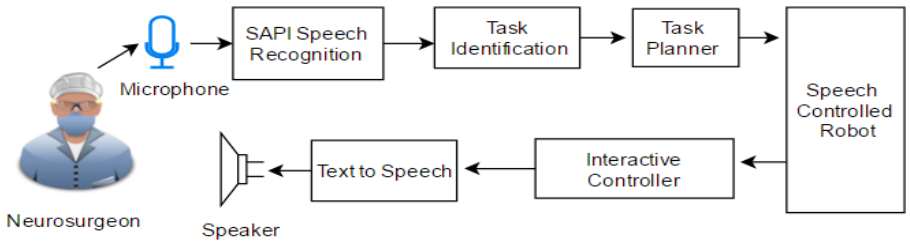

Figure 6.35 Architecture of the proposed speech controlled robotic system

An experimental study was undertaken using robotic manipulator and voice-controlled guidance allowing the surgeon to move the robotic arm in operating space. The operation surgeon registered 12 distinct voice commands using voice trainer. SAPI is employed to accurately output the crisp control signals for the robot systems, based on linguistic spoken language commands, issued by a user; a powerful speech control system is developed to control the PA107c robotic manipulator using voice commands.

Table 6.7 User spoken directives and the output control signals

User Directive	**Control Output of the SAPI classifier**
Forward	movement in X Positive Direction
Backward	movement in X Negative Direction
Left	movement in Y Positive Direction
Right	movement in Y Negative Direction
Up	movement in Z Positive Direction
Down	movement in Z Negative Direction
Stop	Stop PA107c robot
slow	Change Speed to 0.1 mm/s
fast	Change Speed to 1 mm/s
Select point A	Register End Effector Position and Rotation
Select point B	Register End Effector Position and Rotation
Select Point C	Register End Effector Position and Rotation

The proposed speech controlled robotic system is shown in Fig. 6.35. The system is composed of the microphone; a speech recognizer; Task Identification; PA107c robotic manipulator; Text to Speech converter; and interactive controller used to define the system response with respect each action.

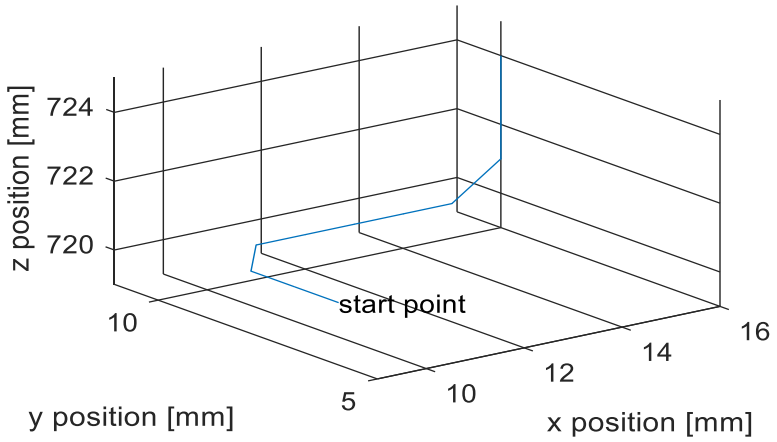

Figure 6.36 Profile of the end-effector of PA 107c.

The input to the system appears in the form of spoken commands from the neurosurgeon. The voice commands recognized and converted to a crisp desired action for the robotic manipulator. Fig. 6.36 shows the end effector trajectory resulted from voice commands. The movements of the robotic manipulator performed in the three-dimensional operational space, namely, up, down, forward, backward, left, and right directions.

6.4.2 POSE REGISTRATION IN NEUROSURGICAL INTERVENTIONS

Robotic systems are involved in the operating room to solve such problem to minimize the risk of damage to critical structures. The human skull has complex 3D anatomy and traversing critical structures such as nerves and vessels. So; Registration Points or fiducials used to define the landmarks on the skull and used in inter-operative registration; The selected registration technique in neurosurgical procedures should be reliable; precise, convenient; and fast to obtain an optimal result. Manual registration is time-consuming and limited by human dexterity. There are a variety of registration methods, including paired points and surface matching concepts, pose registration introducing a marker-based pattern, which is placed on the patient table. The registered markers form a pattern, allowing to deduce the robot pose and Skull pose by use of the projective invariant cross-ratio. Thus, the physician's view is optimized, and localization of the robot tooltip at the correct position is facilitated.

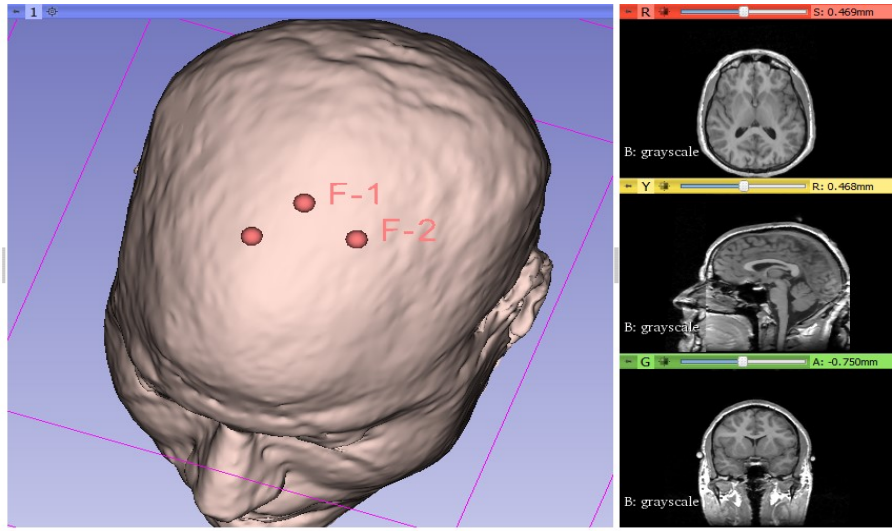

Figure 6.37 Pose Registration in 3D Slicer

At least 3-point pairs are needed for registration; more points increase accuracy. Realistically, the accuracy requirement for location's registration of skull surface should be around 0.5 mm accuracy. Therefore, applying industrial robots with a coordinate measurement technique for skull metrology is practical and feasible. For example; robot-Assisted neurosurgery could improve safety and efficiency of skull drilling by defining "drilling zone, and dynamic safety boundaries to prevent excursion outside the safe zone and respond to surgeon's commands. The safety boundaries have anticipated benefits such as Less trauma, faster recovery, fewer complications.

The PA107c robotic manipulator has seven-jointed Degree of freedom, and its geometry constraints have resulted in several measuring trajectories and strategies. When there is direct contact between the robot with the skull; system reads the X, Y, and Z coordinates of each of points to determine position and orientation of the points on the skull surface. Also; Joystick is used in the proposed system as one of the command control inputs. Fig. 6.37 shows the Pose registration in 3D Slicer. The overall workflow of the neurosurgical navigation system starts with scan imaging when the patient arrives, the acquired images of the patient are automatically transferred to the to 3D Slicer software either in real-time or offline using OpenIGT protocol. The images segmented and the 3D surface model reconstructed to simplify the Preoperative planning by defining the area of interest and defining target and entry point and creating a trajectory connecting these two points.

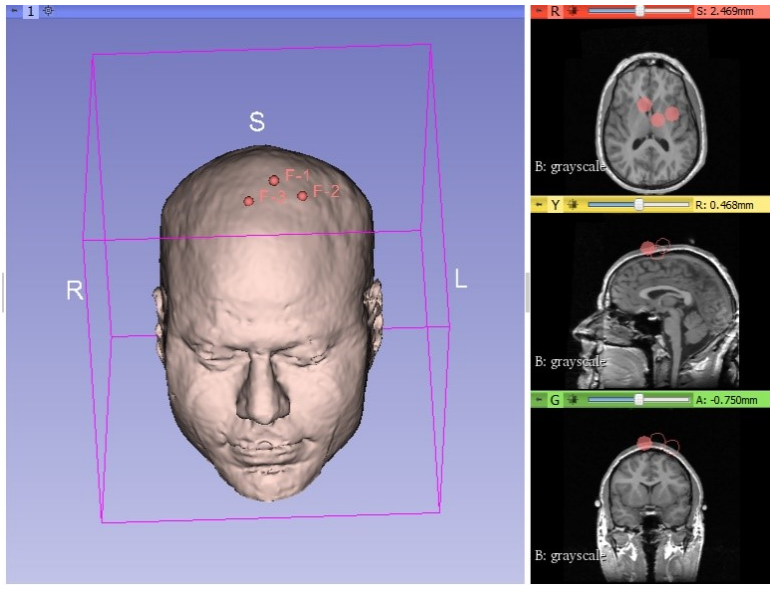

Figure 6.38 Pose Registration in 3D Slicer

The created trajectory sent to the robot controller and safety margins set up on Matlab. Calibration is done by aligning the patient position and robot position using real and virtual fiducials to determine the spatial relationship between the surgical instruments and the reference frames. The calibration matrices are registered to synchronize the preoperative virtual and real fiducials. The last step is starting skull drilling using semi-Automatic control. The feasibility of the navigation system was validated through a phantom experiment in which a point of interest is defined on the Human skull. Pose calibration of the surgical drill was conducted using three physical fiducial landmarks which placed on a human skull. These points were selected in the images as the fiducials and registered on reference coordinate system (RCS) to registration transformation matrix was generated. Then, The PA107c robot robotic manipulator commanded to reach the defined point and perform the planned drilling trajectory.

Table 6.8 Pose Registration in 3D Slicer

Registered Point	**Right,**	**Anterior**	**Superior**
F-1	-9.375	-17.306	76.196
F-2	-29.163	-9.509	67.489
F-3	8.27	3.092	72.633

The movement of the surgical drill was rendered in the preoperative image coordinate system in real time. Also, the 3D Slicer software used for visualization where the volumetric image of the skull resliced with the plane of the surgical tool and the sectional image visualized on the 2D and 3D viewers as shown in Fig. 6.38. The movement of the robotic manipulator navigated on the computer screen in real time and synchronized with the preoperatively planned drill trajectory. Measurements are taken by this contact method being logged into a computer using voice commands. Operators no longer had to touch the keyboard to register points physically. A camera is mounted on the robot tooltip and connected to All-in-one computer and used for intraoperative motion tracking. Registration points are tracked, and visualization software is running at the same time. The all-in-one computer has Celeron processor, 1GB RAM, and it runs under the Windows XP operating system.

6.4.3 EVALUATION OF ROBOT POSITIONING ACCURACY

The industrial robot needs to be calibrated before it can be used in metrology. Robot repeatability can normally reach 1 mm which is not acceptable in surface calibration techniques used in surgery.

However; accuracy calibration appears to be the major issue for a robotic-assisted neurosurgery system.

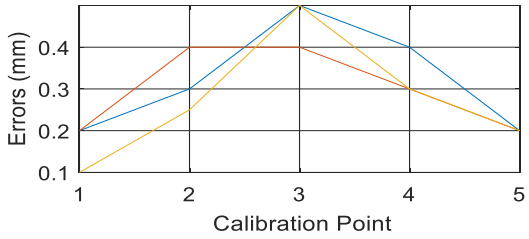

Figure 6.39 Robot Positioning Accuracy

According to the robot calibration standard ISO 9238, the PA107c robot needs to move repeatedly to five explicitly defined testing points where the robot should be moved from position P1 to P5 for five times surround the measuring workspace and to measure the deviation of positioning accuracy and repeatability errors in each point. The calibration results is shown on Fig 6.40.

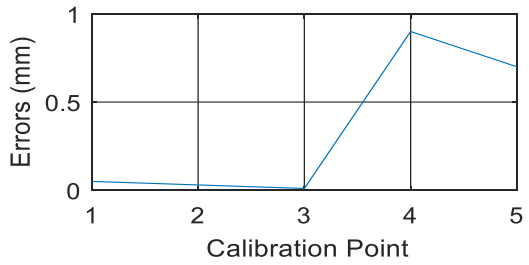

Figure 6.40 Robot Repeatability Accuracy

The maximum standard deviation of repeatability is 0.6 mm, and the maximum positioning inaccuracy is 0.5 mm A calibration sheet was designed and implemented in this study. Force Steering used to move the robot end effector; fiducials are logged via voice commands and the offset values calculated on the tipping point of the instrument. Experiments show that the robot pose estimation is reliable and accurate for translations inside an area of 50 cm x 50 cm. Mean error values are 0.33 mm in 3D space and 0.1 mm in the 2D plane. High success rates of 90% and fully satisfying execution times below 4 seconds could be achieved.

6.4.4 A VIRTUAL REALITY TRAINING SYSTEM FOR ROBOT-ASSISTED NEUROSURGERY

Neurosurgical procedures emphasis on the experience and training of the surgeons. Many surgical

practices involve the cutting or drilling of bone which can be learned only through repetitious practice using different methods such as mock surgeries on animals, dissection of cadavers, and the operating on live patients. However, there are many ethical and accessibility issues with the classical training techniques. Neurosurgical training systems based on virtual reality offer a cost-effective and efficient training tools for surgeons. The training system allows students to acquire the skills needed to perform surgeries on real patients and practice their technical skills without any risk to patients. The training system consists of the task of building the 3D virtual operation room and the interfacing techniques with the surgical robot.

Figure 6.41 Virtual Reality Smartphone Headset

The 3D model of the patient's brain is visualized using 3D Slicer software. The user can control the movement of the robotic manipulator and interact with the virtual tool inserted into the virtual brain. The interactive control increases the immersion and telepresence. The created training environment can mimic the real surgical procedure and might used for medical staff training and evaluation. Virtual reality (VR) environment enable surgeons to perform motivating surgery-like scenarios. The proposed prototype allows users to navigate naturally within and interact with the Robot to perform neurosurgical procedures and establish a robot-assisted surgical training demo system oriented clinical application for a risk-free training environment. VR headset provides virtual reality for the wearer and comprises a stereoscopic head-mounted display, stereo sound, and gyroscope sensor. Head motion tracking sensors allow the wearer to view the surroundings, with the perspective moving as his head moves, giving a deep sense of immersion. Virtual Reality Smartphone Headset manufactured by Dream Vision which is compatible with most smartphone devices up to 6" wide; it has an integrated headband and Microphone as shown in Fig 6.41.

A smartphone can be inserted into VR headset. VR content is viewed from the screen of the device itself through lenses acting as a stereoscope, the virtual Operation room is constructed on VREP simulation platform, and video is streamed from Computer to phone for virtual reality training. The interactive navigation is tested on Both Android and iOS devices with minimal lag, and head tracking. The tools needed to connect the VREP, VR, and smartphone are either a micro USB cable for Android phones or a high-speed WI-FI connection and the Pc server streaming software such as TRINUS VR software.

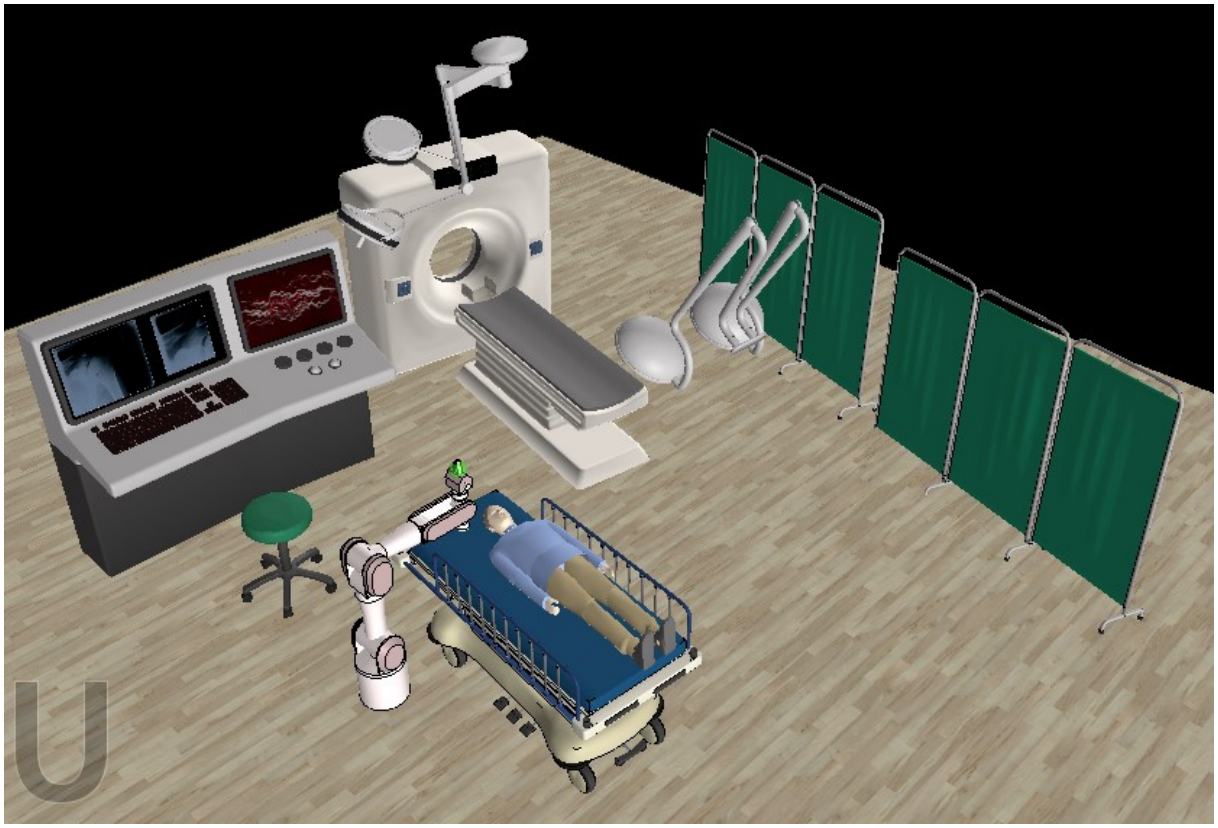

Figure 6.42 Simulation of neurosurgery by using VREP

6.5 SUMMARY

This chapter presents the design of a robotic neurosurgical system; A software-hardware integration of an image guided biomedical robotics system is introduced. The proposed architecture will ensure the seamless data flow among those components and enable a closed-loop process of planning and control. The proposed system could be integrated with an intraoperative image guidance using MRI and medical image visualization software. 3D Slicer used in image processing and visualization for surgical planning, and Matlab used in implementing control

algorithms and synchronization between the different components. Force steering algorithm is implemented and tested for different scenarios. The simulations and experiments have further verified the proposed approach. Software architecture was also developed which uses the concept of device drivers to achieve modularity in a real-time subsystem environment. The proposed Torque /position control approach has been simulated on the dynamic model of PA10-7C. Path planning control algorithm will be developed for the manipulator. Location registration is the most critical issue that must be addressed before a robot can position a tool on a patient accurately in neurosurgical procedures. Pose registration is simplified by using Human-robot interaction. The preoperatively planned space-related to intraoperative operating space to improve navigation. Experiments were carried out on a human skull, and the performance was evaluated. The proposed algorithm is very efficient and could be used for robot teaching mode. More points slightly increase the RMSE (root mean square error) and increase accuracy because random errors cancel each other out; RMSE is an unknown mix of precision and accuracy. This navigation system has been validated by multiple experiments, and the existing layout could be integrated into a clinical environment within only 30 minutes setup time. The registration time is less than 2 minutes. A three-dimensional force sensor is employed in direction detection and collision avoidance. As a result, collision-free poses of the PA10-7c robotic manipulator is reached with more safety and flexibility. Experiment results have shown successful pose registration and emulation of the actual command force. The robot movement controlled using the 3D force feedback signal which considered as an orienting vector controlling the direction of the robot end effector displacement in the Cartesian space. Common software architecture is proposed and implemented to facilitate any future integration between the developed modules. The use of voice-controlled robotic manipulator as a substitute for the joystick is feasible and more convenient and minimize the number of Sterilized tools in the operation room. Performance evaluation of pose registration and calibration system is introduced and presented. The intuitive graphical user interface is developed using Open-source environment. Average error 0.5 mm and maximum pose error is about 1 mm. Experiments demonstrated that Navigation system enables the user to align the real and virtual markers to obtain an optimized registration of the desired target.

CHAPTER 7 CONCLUSIONS

7.1 CONCLUSION

A control theory for the proposed neurosurgical robotic system was presented in this thesis. It included the modeling and control of the Mitsubishi PA10-7C robotic manipulator; the modular design of neurosurgical robotic systems; an LQR controller; an ANN drilling state detection; human-robot interaction for neurosurgical manipulators; a 3D simulator of an operating room; and a semi-automatic control architecture that makes the proposed system stable, reliable, safe, and efficient. The main contribution of this thesis is to develop a reliable and convenient neurosurgical robotic system. The interrelated issues that the proposed approach solved are as follows:

1. The approach uses systematic modeling and control of the Mitsubishi PA10-7C robot manipulator. The Mitsubishi PA10-7C robot arm is modeled systematically, includes kinematic and dynamic models, and uses a Jacobian matrix. It also incorporates general set-point control and trajectory tracking control utilizing a feedforward controller to move the perforator smoothly.
2. The ANN-based force information analysis and process applies force sampling by the JR3 sensor, which contains noise from the sensor itself and mechanic vibration when drilling. An ANN-based force state classification is proposed in order to know the state of the robot when performing tasks such as rotation, waiting, drilling starting, drilling stops, and retrieving. In the proposed approach, ANN is constructed and trained to classify various states by the force data.
3. The approach features semi-automatic control. To safely use the proposed system, the user switches from an automatic to a manual control state. Task-planning is proposed to enable the surgeon to define the area of interest and safety margins.
4. The proposed system architecture could be used as a testing and development platform for various surgical applications, such as image-guided percutaneous needle interventions, tumor biopsy, tissue ablations, and neurological pain management,
5. The designed system could be used as a training platform for surgeons to reduce the time and limitations of training under senior supervision.
6. An image-guided cooperatively controlled robot is designed to assist with neurosurgery procedures, while skull surface position registration is performed using human-robot interaction techniques. The kinetic sensor is used for position-tracking and is combined with an Augmented and Virtual Reality display.

7. Software architecture is presented for the integration of a 3D Slicer and VREP to provide access to resources developed in two communities. This integration allows quick prototyping of surgical robot systems, enabling the system to be scaled up easily without changing the system architecture. The introduced interface will hopefully foster research collaboration and incremental results toward more effective treatment in a large patient population. Since the presented method is not limited to neurosurgery brain tumors, future studies should be conducted to explore its full potentials in other surgical procedures.

7.2 FUTURE WORK

The control theory discussed and developed in this thesis has successfully solved the control problems for the proposed telerobotic neurosurgical system. However, there are still many tasks to be done to improve the current system, as follows.

1. A dedicated robot arm design is needed to make it compact, reliable, highly-loaded, and thus suitable for future medical robot applications.
2. Adding dynamic obstacle avoidance detection would help the system detect surgeons, nurses, and devices in the operation room and improve safety.
3. The proposed system needs to be tested further on a cadaver in a medical lab.
4. Enhanced 3D simulator should be added to simulate the operation room and the robot to verify the control strategies and for training surgeons
5. If the neurosurgical robotic manipulator will be used in rural areas, a low bandwidth communication channel should be integrated into the controller design to optimize the overall system performance.

Bibliography

[1] V. Chan, J. D. Pole, R. E. Mann, and A. Colantonio, "A population based perspective on children and youth with brain tumours," BMC Cancer, vol. 15, no. 1, p. 1007, Dec. 2015.

[2] T. Litman, "Transportation and Public Health," Annu. Rev. Public Health, vol. 34, no. 1, pp. 217–233, Mar. 2013.

[3] D. Hastings, "Trauma prevention and intervention around the world.," in International Trauma and Emergency Medicine Conference, 2002, pp. 13–16.

[4] N. Agarwal, I. O. Smith, K. L. Tomei, C. J. Prestigiacomo, C. D. Gandhi, and , A, "Norrmén- & Improving Medical Student Recruitment into Neurological Surgery: Institution's Experience.," World Neurosurg., vol. 80, no. 6, pp. 745–750, 2013.

[5] P. K. Dempsey, P. K. Dempsey, and S. W. Hwang, "Management of Closed Head Injury," in Surgical Intensive Care Medicine, Boston, MA: Springer US, 2010, pp. 129–136.

[6] Jongseong Jang and Young Soo Kim, "Safety management algorithm for telesurgical robot system for brain tumor surgery," in IEEE ISR 2013, 2013, pp. 1–2.

[7] S. Au, K. Ko, J. Tsang, and Y. C. Chan, "Robotic endovascular surgery," Asian Cardiovasc. Thorac. Ann., vol. 22, no. 1, pp. 110–114, Jan. 2014.

[8] "Brain Injury Statistics | Northern British Columbia | Prince George," Northern Brain Injury Association | British Columbia, 2017. [Online]. Available: http://nbia.ca/brain-injury-statistics/. [Accessed: 31-Mar-2017].

[9] M. Faul, L. Xu, M. M. Wald, and V. G. Coronado, "Traumatic brain injury in the United States: emergency department visits, hospitalizations, and deaths," Centers Dis. Control Prev. Natl. Cent. Inj. Prev. Control, pp. 891–904, 2010.

[10] Brain Trust Canada, "The Facts about Brain Injury," 2017. [Online]. Available: http://www.vistacentre.ca/_files/statistics.pdf. [Accessed: 31-Mar-2017].

[11] V. G. C. Mark Faul, Likang Xu, Marlena M. Wald, “Traumatic Brain Injury in The United States, Emergency Department Visits, Hospitalizations and Deaths 2002–2006,” U .S . Department of Health and Human Services, 2017. [Online]. Available: https://www.cdc.gov/traumaticbraininjury/pdf/blue_book.pdf.

[12] M. M. Dinh, K. Bein, S. Roncal, C. M. Byrne, J. Petchell, and J. Brennan, “Redefining the golden hour for severe head injury in an urban setting : The effect of prehospital arrival times on patient outcomes,” Injury, vol. 44, no. 5, pp. 606–610, 2013.

[13] CMA Member Dialogue, “A Canadian Approach to Assisted Dying: A CMA Member Dialogue Summary Report,” 2016. [Online]. Available: https://www.cma.ca/Assets/assets-library/document/en/advocacy/Canadian-Approach-Assisted-Dying-e.pdf.

[14] S. I. Woodrow, C. O. Kelly, S. J. Hamstra, and M. C. Wallace, “Unemployment in an Underserviced Specialty ?: The Need for Co-ordinated Workforce Planning in Canadian Neurosurgery,” Can. J. Neurol., pp. 170–174, 2006.

[15] B. Barua and F. Ren, “Waiting Your Turn: Wait Times for Health Care in Canada, 2016 report,” Fraser Institute, 2016. [Online]. Available: https://www.fraserinstitute.org/sites/default/files/waiting-your-turn-wait-times-for-health-care-in-canada-2016.pdf. [Accessed: 20-Jul-2006].

[16] B. Morris, “Robotic surgery: applications, limitations, and impact on surgical education.,” MedGenMed, vol. 7, no. 3, p. 72, Sep. 2005.

[17] T. D. White D, M. T. Black, and P. A. Folkens, Human osteology, 3rd ed. Benicia, California: Elsevier/Academic Press, 2012.

[18] L. Aiello, C. Dean, and J. Cameron, An introduction to human evolutionary anatomy, 2nd ed. London, UK: Academic Press, 2002.

[19] J. E. Risdall, D. K. Menon, and B. London, “Traumatic brain injury.,” Philos. Trans. R. Soc. Biol. Sci. Ruby Prem. Press, vol. 366, no. 1562, pp. 241–250, 2011.

[20] K. U. Schmitt, M. H. Muser, and F. Walz, “Trauma Biomechanics: Introduction to accidental injury.,” in Springer Science Business Media, 1st ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, p. 173.

[21] G. M. Giles and J. Clark-Wilson, Brain Injury Rehabilitation, 1st ed. Boston, MA: Springer US, 1993.

[22] P. Gambadauro, “Torrejón, R. . The ‘tele’ factor in surgery today and tomorrow: implications for surgical training and education.,” Surg. Today, vol. 43, no. 2, pp. 115–122, 2013.

[23] J. Arata et al., “Surgical bedside master console for neurosurgical robotic system,” Int. J. Comput. Assist. Radiol. Surg., vol. 8, no. 1, pp. 75–86, Jan. 2013.

[24] H. Marcus, D. Nandi, A. Darzi, and Guang-Zhong Yang, “Surgical Robotics Through a Keyhole: From Today’s Translational Barriers to Tomorrow’s Disappearing Robots,” IEEE Trans. Biomed. Eng., vol. 60, no. 3, pp. 674–681, Mar. 2013.

[25] R. K. Pandey and S. S. Panda, “Drilling of bone: A comprehensive review,” J. Clin. Orthop. Trauma, vol. 4, no. 1, pp. 15–30, 2013.

[26] W. W. Bertollo N, N. Bertollo, and W. R. Walsh, “Biomechanics in Applications,” Biomech. Appl., pp. 249–274, 2011.

[27] T. W. Vogel, B. J. Dlouhy, and M. A. Howard, “Don’t take the plunge: avoiding adverse events with cranial perforators,” J. Neurosurg., vol. 115, no. 3, pp. 570–575, Sep. 2011.

[28] J. Brodie and S. Eljamel, “Evaluation of a neurosurgical robotic system to make accurate burr holes,” Int. J. Med. Robot. Comput. Assist. Surg., vol. 7, no. 1, pp. 101–106, Mar. 2011.

[29] Dong-Soo Kwon et al., “The mechanism and registration method of a surgical robot for hip arthroplasty,” in IEEE International Conference on Robotics and Automation, 2002, vol. 2, pp. 1889–1894.

[30] G. Dogangil, B. L. Davies, and F. Rodriguez y Baena, “A review of medical robotics for minimally invasive soft tissue surgery,” Proc. Inst. Mech. Eng. Part H J. Eng. Med., vol. 224, no. 5, pp. 653–679, May 2010.

[31] D. Glauser, P. Flury, N. Villotte, and C. W. Burckhardt, “Conception of a robot dedicated to neurosurgical operations,” in Fifth International Conference on Advanced Robotics ’Robots in Unstructured Environments, 1991, pp. 899–904 vol.1.

[32] D. Baker, P. N. Brett, M. V. Griffiths, and L. Reyes, “A mechatronic drilling tool for ear surgery: A case study of some design characteristics,” Mechatronics, vol. 6, no. 4, pp. 461–477, Jun. 1996.

[33] V. G. Kaburlasos, V. Petridis, P. Brett, and D. Baker, “Learning a linear association of drilling profiles in stapedotomy surgery,” in Proceedings. 1998 IEEE International Conference on Robotics and Automation, 1998, vol. 1, pp. 705–710.

[34] Weimin Shen and J. Gu, “Multi-criteria kinematics control for the PA10-7C robot arm with robust singularities,” in 2007 IEEE International Conference on Robotics and Biomimetics, 2007, pp. 1242–1248.

[35] J. A. Smith, J. Jivraj, R. Wong, and V. Yang, “30 Years of Neurosurgical Robots: Review and Trends for Manipulators and Associated Navigational Systems,” Ann. Biomed. Eng., vol. 44, no. 4, pp. 836–846, 2016.

[36] R. Kapoor, C. Hoffman, I. Hussain, and P. Stieg, “Basic Neurosurgical Procedures,” in Comprehensive Guide to Neurosurgical Conditions, 1st ed., Cham: Springer International Publishing, 2015, pp. 59–71.

[37] T. Essomba, C.-T. Wu, S.-T. Lee, and C.-H. Kuo, “Mechanical Design of a Craniotomy Robotic Manipulator Based on Optimal Kinematic and Force Performance,” Springer, Cham, 2016, pp. 191–198.

[38] J. Arata et al., “Surgical bedside master console for neurosurgical robotic system,” Int. J. Comput. Assist. Radiol. Surg., vol. 8, no. 1, pp. 75–86, Jan. 2013.

[39] P. Gambadauro and R. Torrejo, “The ‘“ tele ”’ factor in surgery today and tomorrow : implications for surgical training and education,” Surg. Today, pp. 115–122, 2013.

[40] B. Morris, “Robotic surgery: applications, limitations, and impact on surgical education.,” MedGenMed, vol. 7, no. 3, p. 72, Sep. 2005.

[41] T. A. Mattei, A. H. Rodriguez, D. Sambhara, and E. Mendel, “Current state-of-the-art and future perspectives of robotic technology in neurosurgery,” Neurosurg. Rev., vol. 37, no. 3, pp. 357–366, 2014.

[42] M. B. Popovic, Biomechanics and Robotics, 1st ed. boca Raton, Fl: Pan Stanford, 2013.

[43] W. Walz, Experimental Neurosurgery in Animal Models, 1st ed., vol. 116. New York, NY: Springer New York, 2016.

[44] E. M. Bogen, K. M. Augestad, H. R. H. Patel, and R. Lindsetmo, “Telementoring in education of laparoscopic surgeons : An emerging technology,” World J Gastrointest Endosc., vol. 6, no. 5, pp. 148–155, 2014.

[45] S. Najarian, M. Fallahnezhad, and E. Afshari, “Advances in medical robotic systems with specific applications in surgery--a review.,” J. Med. Eng. Technol., vol. 35, no. 1, pp. 19–33, 2011.

[46] J. P. Cullen and M. A. Talamini, “General Surgery : Current Trends and Recent Innovations,” in General Surgery: Current Trends and Recent Innovations, 2010, pp. 781–791.

[47] P. Gélinas-Phaneuf, Nicholas MD; Del Maestro, Rolando F. MD, “Surgical Expertise in Neurosurgery: Integrating Theory Into Practice ´,” Neurosurgery, vol. 73, no. 4, pp. 30–38, 2013.

[48] T. Sakaguchi, “Percutaneous puncture with a robot,” Hinyokika Kiyo., vol. 31, no. 7, pp. 1265–8, Jul. 1985.

[49] W. Shen, “Control theory on tele-robotics with neurosurgical applications,” Doctoral dissertation, Dalhousie University, Halifax, NS, 2009.

[50] P. L. Gildenberg, “Robotic Neurosurgery,” in Textbook of Stereotactic and Functional Neurosurgery, Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 583–597.

[51] J. Zhang, “Design of Steerable Electrode Arrays and Optimal Insertion Path Planning for Robot-Assisted Cochlear Implant Surgery,” Doctoral dissertation, Columbia University, 2010.

[52] K. Cleary and T. M. Peters, “Image-guided interventions: technology review and clinical applications,” Annu Rev Biomed Eng, vol. 12, pp. 119–142, 2010.

[53] A. Benabid, P. Cinquin, S. Lavallee, J. Le Bas, J. Demongeot, and J. de Rougemont, “Computer-Driven Robot for Stereotactic Surgery Connected to CT Scan and Magnetic Resonance Imaging,” Meet. Am. Soc. Stereotact. Funct. Neurosurg., pp. 153–154, 1987.

[54] B. Davies, “A review of robotics in surgery.,” Proc. Inst. Mech. Eng. H., vol. 214, no. 1, pp. 129–140, 2000.

[55] D. Glauser, H. Fankhauser, M. Epitaux, J. Hefti, and A. Jaccottet, “Neurosurgical robot Minerva: First results and current developments,” J. Image Guid. Surg., vol. 1, no. 5, pp. 266–272, 1995.

[56] C. W. Burckhardt, P. Flury, and D. Glauser, “Stereotactic brain surgery,” IEEE Eng. Med. Biol. Mag., vol. 14, no. 3, pp. 314–317, 1995.

[57] C. Faria, W. Erlhagen, M. Rito, E. De Momi, G. Ferrigno, and E. Bicho, “Review of robotic technology for stereotactic neurosurgery,” IEEE Rev. Biomed. Eng., vol. 8, pp. 125–137, 2015.

[58] G. Savuc, D. Forna, and N. C. Forna, “The Contribution of Medical Robots to Clinical Performance: Up-To-Date,” vol. 4, no. 4, pp. 67–75, 2012.

[59] A. Gasparetto and V. Zanotto, “Toward an optimal performance index for neurosurgical robot’s design,” Robotica, vol. 28, no. 2, p. 279, 2010.

[60] S. Lavalldet, J. Troccazt, L. Gaboriti, P. Cinquint, A. L. Benabid, and D. Hoffmannt, “Image guided operating robot: a clinical application in stereotactic neurosurgery,” 1992 IEEE Int. Conf. Robot. Autom., no. 2, pp. 618–624, 1992.

[61] G. R. Sutherland, S. Lama, L. S. Gan, S. Wolfsberger, and K. Zareinia, “Merging machines with microsurgery: clinical experience with neuroArm,” J. Neurosurg., vol. 118, no. 3, pp. 521–529, Mar. 2013.

[62] A. R. Asthagiri, N. Pouratian, J. Sherman, G. Ahmed, and M. E. Shaffrey, “Advances in Brain Tumor Surgery,” neurlogic Clin., vol. 25, pp. 975–1003, 2007.

[63] K. Cleary and C. Nguyen, “State of the art in surgical robotics: Clinical applications and technology challenges,” Comput. Aided Surg., vol. 6, no. 6, pp. 312–328, 2001.

[64] Y.-C. Chung, “Path Control for NeuroMate Robot in a Skull Drilling System,” J. Korean Soc. Manuf. Technol. Eng., vol. 22, no. 2, pp. 256–262, Apr. 2013.

[65] H.-C. Schneider and J. Wahrburg, “Simulation Model for the Dynamics Analysis of a Surgical Assistance Robot,” in Robot Surgery, 1st ed., InTech, 2010.

[66] S. K. Nandi, A. Mahato, B. Kundu, and P. Mukherjee, “Doped Bioactive Glass Materials in Bone Regeneration,” in Advanced Techniques in Bone Regeneration, 1st ed., Vanja Bozovic, Ed. InTech, 2016.

[67] D. Liu and T. Wang, “A Workflow for Robot Assisted Neurosurgery,” in 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006, pp. 2870–2875.

[68] T. W. Da Liu, “Study On Robot -Assisted Minimally Invasive Neurosugery,” in Study On Robot -Assisted Minimally Invasive Neurosugery, 2001.

[69] P. W. A. Willems, H. J. Noordmans, J. W. B. van der Sprenkel, M. A. Viergever, and C. A. F. Tulleken, “An MKM-mounted instrument holder for frameless point-stereotactic procedures: a phantom-based accuracy evaluation,” J. Neurosurg., vol. 95, no. 6, pp. 1067–1074, Dec. 2001.

[70] A. Masih, W. Building, and P. Road, “Towards Requirements Engineering for a Tumour Removing Robot : Work-practice Observation of Surgical Teams Performing Brain Tumour Surgery,” in Proceedings of the ACM 2011 conference on Computer supported cooperative work., pp. 677–680.

[71] D. LaRose et al., “A Telerobotic Assistant for Laparoscopic Surgery,” IEEE Eng. Med. Biol. Mag., vol. 14, no. 3, pp. 279–288, 1995.

[72] G. R. Sutherland, P. B. Mcbeth, and D. F. Louw, “NeuroArm : an MR compatible robot for microsurgery,” Int. Congr. Ser., vol. 1256, pp. 504–508, 2003.

[73] H. Lang, Y. Wang, and C. W. de Silva, “Visual servoing with LQR control for mobile robots,” Control Autom. (ICCA), 2010 8th IEEE Int. Conf., pp. 317–321, 2010.

[74] R. Muradore et al., “Development of a Cognitive Robotic System for Simple Surgical Tasks,” Int. J. Adv. Robot. Syst., vol. 12, no. 4, p. 37, Apr. 2015.

[75] P. Gomes, “Surgical robotics: Reviewing the past, analysing the present, imagining the future,” Robot. Comput. Integr. Manuf., vol. 27, pp. 261–266, 2010.

[76] C. Bergeles and G. Z. Yang, “From passive tool holders to microsurgeons: Safer, smaller, smarter surgical robots,” IEEE Trans. Biomed. Eng., vol. 61, no. 5, pp. 1565–1576, 2014.

[77] T. Ungi et al., “Navigated Breast Tumor Excision Using Electromagnetically Tracked Ultrasound and Surgical Instruments,” IEEE Trans. Biomed. Eng., vol. 63, no. 3, pp. 600–606, Mar. 2016.

[78] T. Xia et al., “An integrated system for planning, navigation and robotic assistance for skull base surgery,” Int. J. Med. Robot. Comput. Assist. Surg., vol. 4, no. 4, pp. 321–330, Dec. 2008.

[79] S. Tauscher, J. Tokuda, G. Schreiber, T. Neff, N. Hata, and T. Ortmaier, “OpenIGTLink interface for state control and visualisation of a robot for image-guided therapy systems,” Int. J. Comput. Assist. Radiol. Surg., vol. 10, no. 3, pp. 285–292, Mar. 2015.

[80] W. Shen, J. Gu, and Y. Shen, “Trajectory planning for tele-robotic skull drill system,” Proc. IEEE ICIA 2006 - 2006 IEEE Int. Conf. Inf. Acquis., pp. 1497–1501, 2006.

[81] W. Shen, J. Gu, and E. Milios, “Robotic neurosurgery and clinical applications,” in International Conference on Intelligent Mechatronics and Automation 2004 Proceedings, 2004, no. August, pp. 114–119.

[82] W. Shen, J. Gu, and Y. Shen, “Using Tele-robotic Skull Drill for Neurosurgical Applications,” in 2006 International Conference on Mechatronics and Automation, 2006, pp. 334–338.

[83] W. Shen, J. Gu, and Zuren Feng, “A Stable tele-robotic neurosurgical system based on SMC,” in 2007 IEEE International Conference on Robotics and Biomimetics (ROBIO), 2007, pp. 150–155.

[84] M. M. Mohamed and J. Gu, “Modular Design of Neurosurgical Robotic System,” Int. J. Robot. Autom., 2017.

[85] B. Davies, “Robotic Surgery – A Personal View of the Past, Present and Future,” Int. J. Adv. Robot. Syst., vol. 12, no. 5, p. 54, May 2015.

[86] P. L. Gildenberg, “Robotic Neurosurgery,” in Textbook of Stereotactic and Functional Neurosurgery, R. R. T. Andres M. Lozano, Philip L. Gildenberg, Ed. 2009, pp. 583–597.

[87] D. F. Louw et al., “Surgical robotics: A review and neurosurgical prototype development,” Neurosurgery, vol. 54, no. 3, pp. 525–537, 2004.

[88] R. J. Hendrick, C. R. Mitchell, S. D. Herrell, and R. J. Webster, “Hand-held transendoscopic robotic manipulators: A transurethral laser prostate surgery case study,” Int. J. Rob. Res., vol. 34, no. 13, pp. 1559–1572, 2015.

[89] G. Niemeyer, C. Preusche, S. Stramigioli, and D. Lee, “Telerobotics,” in Springer Handbook of Robotics, 1st ed., Cham: Springer International Publishing, 2016, pp. 1085–1108.

[90] Y. Deng, P.-P. J. Beaujean, E. An, and E. Carlson, “Task Allocation and Path Planning for Collaborative Autonomous Underwater Vehicles Operating through an Underwater Acoustic Network,” J. Robot., vol. 2013, pp. 1–15, 2013.

[91] R. Wirz et al., “An experimental feasibility study on robotic endonasal telesurgery.,” Neurosurgery, vol. 76, no. 4, p. 479–84; discussion 484, Apr. 2015.

[92] W. Shen, J. Gu, and E. E. Milios, “Self-configuration fuzzy system for inverse kinematics of robot manipulators,” Annu. Conf. North Am. Fuzzy Inf. Process. Soc. - NAFIPS, pp. 41–45, 2006.

[93] O. Maizza Neto, “Modal analysis and control of flexible manipulator arms.,” Doctoral dissertation, Massachusetts Institute of Technology, 1975.

[94] G. Augustin et al., “Cortical bone drilling and thermal osteonecrosis,” Clin. Biomech., vol. 27, no. 4, pp. 313–325, May 2012.

[95] B. Noble, “Bone microdamage and cell apoptosis.,” Eur. Cell. Mater., vol. 6, p. 46–55; discusssion 55, Dec. 2003.

[96] R. Baron, W. C. Horne, F. Bronner, M. C. Farach-Carson, and J. Rubin, Bone resorption., 1st ed. London, UK: London Springer, 2005.

[97] I. Clabaugh, “An evaluation of thermal changes during insertion of self-drilling miniscrew implants as measured by infrared thermography,” Doctoral dissertation, Saint Louis University, 2013.

[98] N. M. da Silva, V. E. Rozanski, and J. P. S. Cunha, “A 3D multimodal approach to precisely locate DBS electrodes in the basal ganglia brain region,” in 2015 7th International IEEE/EMBS Conference on Neural Engineering (NER), 2015, pp. 292–295.

[99] B. B. G. M. Franssen, P. J. van Diest, A. H. Schuurman, and M. Kon, “Drilling k-wires, what about the osteocytes? An experimental study in rabbits,” Arch. Orthop. Trauma Surg., vol. 128, no. 1, pp. 83–87, Jan. 2008.

[100] A. Adeloye, K. R. Kattan, and F. N. Silverman, “Thickness of the normal skull in the American blacks and whites,” Am. J. Phys. Anthropol., vol. 43, no. 1, pp. 23–30, Jul. 1975.

[101] G. Kuhn, R. Evison, and M. Schultz, “Diagnostic value of micro-CT in comparison with histology in the qualitative assessment of historical human skull bone pathologies.,” Am. J. Phys. Anthropol., vol. 133, no. 4, pp. 1099–1111, 2007.

[102] S. K. Law, “Thickness and resistivity variations over the upper surface of the human skull.,” Brain Topogr., vol. 6, no. 2, pp. 99–109, 1993.

[103] M. Okamoto et al., “Three-dimensional probabilistic anatomical cranio-cerebral correlation via the international 10-20 system oriented for transcranial functional brain mapping.,” Neuroimage, vol. 21, no. 1, pp. 99–111, Jan. 2004.

[104] B. N. Cuffin et al., “Tests of EEG localization accuracy using implanted sources in the human brain.,” Ann Neurol, vol. 29, pp. 132–138, 1991.

[105] R. Kiran and V. V. P. Babu, “Design and Implementation of Portable Pick and Place Robotic,” Int. J. Mag. Eng. Technol. Manag. Res., vol. 4, pp. 435–439, 2017.

[106] L. Sciavicco and B. Siciliano, Modelling and Control of Robot Manipulators, 1st ed. London, UK: Springer London, 2000.

[107] D. B. Camarillo, T. M. Krummel, and J. K. Salisbury, “Robotic technology in surgery: Past, present, and future,” Am. J. Surg., vol. 188, no. 4, pp. 2–15, Oct. 2004.

[108] M. Patil, T. Abukhalil, and T. Sobh, “Hardware Architecture Review of Swarm Robotics System: Self-Reconfigurability, Self-Reassembly, and Self-Replication,” ISRN Robot., vol. 2013, pp. 1–11, 2013.

[109] R. Campa, C. Ramírez, and K. Camarillo, “Motion Control of Industrial Robots in Operational Space: Analysis and Experiments with the PA10 Arm,” Intechopen.Com, pp. 417–443, 2010.

[110] K. M. Ben-Gharbia, A. A. Maciejewski, and R. G. Roberts, “An example of a seven joint manipulator optimized for kinematic fault tolerance,” in 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2014, vol. 2014–Janua, no. January, pp. 802–807.

[111] F. Wang, “Design and Control of Robotic Systems for Upper Extremity Rehabilitation Following Stroke,” Doctoral dissertation, Vanderbilt University, 2011.

[112] Mitsubishi Heavy Industry, “General Purpose Robot PA10 Series , Operating Manual,” Mitsubishi Heavy Industry, Tokyo, 2001.

[113] K. Oonishi, N. Oonishi, and K. Shimoyama, “Producing and the latest development programs of the portable general purpose intelligent arm ‘Mitsubishi PA-10,’” Adv. Robot., vol. 15, no. 3, pp. 333–337, 2001.

[114] K. Oonishi, N. Oonishi, and K. Shimoyama, “Producing and the latest development programs of the portable general purpose intelligent arm ‘Mitsubishi PA-10,’” Adv. Robot., vol. 15, no. 3, pp. 333–337, Jan. 2001.

[115] Y. Zhang and Z. Zhang, “PA10 Examples,” in Repetitive Motion Planning and Control of Redundant Robot Manipulators, 1st ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 137–148.

[116] C. Kapoor, “Task-Based Decision Making and Control of Robotic Manipulators,” Doctoral dissertation, The University of Texas at Austin, 2004.

[117] M. Spong and M. Vidyasagar, Robot dynamics and control, 2nd ed. New York, NY: John Wiley & Sons, 1989.

[118] J. J. Craig, Introduction to Robotics: Mechanics and Control, 3rd ed., vol. 1, no. 3. Upper Saddle River: Pearson Education International, 2004.

[119] M. Toz and S. Kucuk, “Dynamics simulation toolbox for industrial robot manipulators,” Comput. Appl. Eng. Educ., vol. 18, no. 2, pp. 319–330, 2009.

[120] Y. Zhang and Z. Zhang, Repetitive motion planning and control of redundant robot manipulators, 1st ed. Springer Science & Business Media, 2014.

[121] D. Whitney, “Resolved Motion Rate Control of Manipulators and Human Prostheses,” IEEE Trans. Man Mach. Syst., vol. 10, no. 2, pp. 47–53, Jun. 1969.

[122] L. Jin, S. Li, H. M. La, and X. Luo, “Manipulability Optimization of Redundant Manipulators Using Dynamic Neural Networks,” IEEE Trans. Ind. Electron., vol. 64, no. 6, pp. 1–1, 2017.

[123] M. Katayama, K. Asada, X.-Z. Zheng, M. Yamakita, and K. Ito, “Self-Organization of a Task Oriented Visuo-Motor map for a Redundant arm,” in The 5th International Conference on Emerging Technologies and Factory Automation, 1996, pp. 302–308.

[124] R. Murray, Z. Li, and S. Sastry, A Mathematical Introduction to Robotic Manipulation, 1st ed., vol. 29. Boca Raton, Florida: CRC press, 1994.

[125] F. L. Lewis, D. M. Dawson, and C. T. Abdallah, Manipulator Control Theory and Practice, 2nd ed. New York, NY: Marcel Dekker, 2004.

[126] P. H. Chang, “A Closed-Form Solution for Inverse Kinematics of Robot Manipulators with Redundancy,” IEEE J. Robot. Autom., vol. 3, no. 5, pp. 393–403, 1987.

[127] K. Deshmukh, J. L. Rickli, and A. Djuric, “Kinematic Modeling of an Automated Laser Line Point Cloud Scanning System,” Procedia Manuf., vol. 5, pp. 1075–1091, 2016.

[128] Mitsubishi Heavy Industry, “General Purpose Robot PA10 Series Programming Manual,” Tokyo, 2001.

[129] T. Tsumugiwa, R. Yokogawa, and K. Hara, “Measurement method for compliance of vertical-multi-articulated robot application to 7-DOF robot PA-10,” Robot. Autom. 2003. Proceedings. ICRA ’03. IEEE Int. Conf., vol. 2, pp. 2741–2746 vol.2, 2003.

[130] N. A. Bompos, P. K. Artemiadis, A. S. Oikonomopoulos, and K. J. Kyriakopoulos, “Modeling, full identification and control of the mitsubishi PA-10 robot arm,” in 2007 IEEE/ASME international conference on advanced intelligent mechatronics, 2007, pp. 1–6.

[131] M. M. Mohamed, J. Gu, and J. Luo, “LQR controller for robotic skull drilling system,” in 2017 29th Chinese Control And Decision Conference (CCDC), 2017, pp. 7533–7538.

[132] P. Lambrechts, M. Boerlage, and M. Steinbuch, “Trajectory planning and feedforward design for electromechanical motion systems,” Control Eng. Pract., vol. 13, no. 2, pp. 145–157, Feb. 2005.

[133] M. M. Mohamed and J. Gu, “CMOS based single active element PID controllers,” in 2015 IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE), 2015, vol. 2015–June, no. June, pp. 932–936.

[134] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, Nonlinear Programming Theory and Algorithms, 1st ed. Hoboken, New Jersey: John Wiley & Sons, 2013.

[135] N. Lynnerup, J. G. Astrup, and B. Sejrsen, “Thickness of the human cranial diploe in relation to age, sex and general body build,” Head Face Med., vol. 1, no. 1, p. 13, Dec. 2005.

[136] J. A. Motherway, P. Verschueren, G. Van der Perre, J. Vander Sloten, and M. D. Gilchrist, “The mechanical properties of cranial bone: The effect of loading rate and cranial sampling position,” J. Biomech., vol. 42, no. 13, pp. 2129–2135, Sep. 2009.

[137] C. Gunn, Bones and joints : a guide for students, 6th ed. Elsevier, 2012.

[138] M. M. Mohamed and J. Gu, “PLC controller for hydraulic pressing machine,” in Proceedings of the 2015 27th Chinese Control and Decision Conference, CCDC 2015, 2015.

[139] M. M. Mohamed and M. A. Hamdan, “Development of control system for two degree of freedom hydraulic motion base,” in 2010 2nd International Conference on Mechanical and Electronics Engineering, 2010, vol. 2, no. Icmee, pp. V2-166-V2-170.

[140] Mathworks, “User’s Guide for Matlab Simscape.” MathWorks, Natick, MA, 2017.

[141] E. Rohmer, S. P. N. Singh, and M. Freese, “V-REP: A versatile and scalable robot simulation framework,” in IEEE International Conference on Intelligent Robots and Systems, 2013, pp. 1321–1326.

[142] J. KHALILOV, “Interfacing Simulinkmatlab With V-Rep for Analysis and Control Synthesis of a Quadrotor,” Doctoral dissertation,Middle East Technical University, 2016.

[143] M. Prats, Á. P. del Pobil, and P. J. Sanz, Robot Physical Interaction through the combination of Vision, Tactile and Force Feedback, vol. 84. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[144] B. Siciliano and O. Khatib, Springer handbook of robotics, 1st ed. New York, NY: Springer, 2016.

[145] S. Pieper, B. Lorensen, W. Schroeder, and R. Kikinis, "The NA-MIC Kit: ITK, VTK, Pipelines, Grids and 3D Slicer as An Open Platform for the Medical Image Computing Community," 3rd IEEE Int. Symp. Biomed. Imaging Macro to Nano, 2006., pp. 698–701, 2006.

[146] G. FAGIOLO, A. WALDMAN, and J. V HAJNAL, "A simple procedure to improve FMRIb Software Library Brain Extraction Tool performance," Br. J. Radiol., vol. 81, no. 963, pp. 250–251, Mar. 2008.

[147] F. Kurth, C. Gaser, and E. Luders, "A 12-step user guide for analyzing voxel-wise gray matter asymmetries in statistical parametric mapping (SPM)," Nat. Protoc., vol. 10, no. 2, pp. 293–304, Jan. 2015.

[148] M. Nolden et al., "The medical imaging interaction toolkit: Challenges and advances: 10 years of open-source development," Int. J. Comput. Assist. Radiol. Surg., vol. 8, no. 4, pp. 607–620, 2013.

[149] G. Wollny, P. Kellman, M.-J. Ledesma-Carbayo, M. M. Skinner, J.-J. Hublin, and T. Hierl, "MIA - A free and open source software for gray scale medical image analysis," Source Code Biol. Med., vol. 8, no. 1, p. 20, 2013.

[150] I. Wolf et al., "The Medical Imaging Interaction Toolkit," Med. Image Anal., vol. 9, no. 6, pp. 594–604, Dec. 2005.

APPENDICES

APPENDIX 1. INDIVIDUAL PARAMETERS OF PA10 ROBOT JACOBIAN MATRIX

$$q414 = c5s6d7 \tag{3.5.3}$$

$$q424 = -c6d7 - d5 \tag{3.5.4}$$

$$q434 = s5s6d7 \tag{3.5.5}$$

$$q314 = c4q414 - s4q424, \tag{3.5.6}$$

$$q324 = q434 \tag{3.5.7}$$

$$q334 = -s4q414 - c4q424 \tag{3.5.8}$$

$$q214 = c3q314 - s3q434 \tag{3.5.9}$$

$$q224 = -q334 - d3 \tag{3.5.10}$$

$$q234 = s3q314 + c3q434, \tag{3.5.11}$$

$$q114 = c2q214 - s2q224, \tag{3.5.12}$$

$$q134 = -s2q214 - c2q224, \tag{3.5.13}$$

$$q411 = (c1c2c3 - s1s3)c4 - c1s2s4, \tag{3.5.14}$$

$$q412 = -(c1c2c3 - s1s3)s4 - c1s2c4, \tag{3.5.15}$$

$$q413 = -c1c2c3 - s1c3, \tag{3.5.16}$$

$$q421 = (s1c2c3 + c1s3)c4 - s1s2s4, \tag{3.5.17}$$

$$q422 = -(s1c2c3 + c1s3)s4 - s1s2c4, \tag{3.5.18}$$

$$q423 = -s1c2s3 + c1c3, \tag{3.5.19}$$

$$q431 = -s2c3c4 - c2s4, \tag{3.5.20}$$

$$q432 = s2c3s4 - c2c4, \tag{3.5.21}$$

$$q433 = s2s3, \tag{3.5.22}$$

$$q511 = q411c5 + q413s5, \tag{3.5.23}$$

$$q512 = -q411s5 + q413c5, \tag{3.5.24}$$

$$q513 = -q412, \tag{3.5.25}$$

$$q521 = q421c5 + q423s5, \tag{3.5.26}$$

$$q522 = -q421s5 + q423c5, \tag{3.5.27}$$

$$q523 = -q422, \tag{3.5.28}$$

$$q531 = q431c5 + q433s5, \tag{3.5.29}$$

$$q532 = -q431s5 + q433c5, \tag{3.5.30}$$

$$q533 = -q432, \tag{3.5.31}$$

$$q612 = -q511s6 + q412c6, \tag{3.5.32}$$

$$q622 = -q521s6 + q422c6, \tag{3.5.33}$$

$$q632 = -q531s6 + q432c6, \tag{3.5.34}$$

$$p21 = c1c2q214 - c1s2q224 - s1q234, \tag{3.5.35}$$

$$p22 = s1c2q214 - c1s2q224 + c1q234, \tag{3.5.36}$$

$$p23 = -s2q214 - c2q224, \tag{3.5.37}$$

$$p31 = (c1c2c3 - s1s3)q314 - (c1c2c3 + s1c3)q434 + c1s2q334, \tag{3.5.38}$$

$$p32 = (s1c2c3 + c1s3)q314 + (-s1c2s3 + c1c3)q434 + s1s2q334, \tag{3.5.39}$$

$$p33 = -s2c3q314 + s2s3q434 + c2q334, \tag{3.5.40}$$

$$p41 = q411q414 + q412q424 + q413q434, \tag{3.5.41}$$

$$p42 = q421q414 + q422q424 + q423q434, \tag{3.5.42}$$

$$p43 = q431q414 + q432q424 + q433q434, \tag{3.5.43}$$

$$p51 = q511s6d7 + q513c6d7, \tag{3.5.44}$$

$$p52 = q521s6d7 + q523c6d7, \tag{3.5.45}$$

$$p53 = q531s6d7 + q533c6d7, \tag{3.5.46}$$

$$p61 = -q612d7, \tag{3.5.47}$$

$$p62 = -q622d7, \tag{3.5.48}$$

$p63 = -q632d7.$ (3.5.49)

APPENDIX 2. DYNAMIC MODEL OF PA10 ROBOT

From equation 3.7.11 ; The mathematical model of PA10 robot could be simplified as following:

$$\tau = Y(q, \dot{q}, \ddot{q})\,\pi \tag{3.7.13}$$

where τ a 7×1 vector of joint torques, Y a 7×p matrix which is a function of joint positions, velocities and accelerations.

$$l_{c_1x} = l_{c_1y} = l_{c_2x} = l_{c_2z} = l_{c_3x} = l_{c_3y} = l_{c_4x} = l_{c_4z} = l_{c_5x} = l_{c_5y} = l_{c_6x} = l_{c_6z} = 0 \tag{3.7.14}$$

The following simplifications made to minimize system complicity.

$$\hat{I}_{1xz} = \hat{I}_{1xy} = \hat{I}_{1yz} = \hat{I}_{2xy} = \hat{I}_{2xz} = \hat{I}_{3xz} = \hat{I}_{3xy} = \hat{I}_{3yz} = \hat{I}_{4xy} = \hat{I}_{4xz} = \hat{I}_{5xz} = \hat{I}_{5xy} = \hat{I}_{5yz} = \hat{I}_{6xz} = \hat{I}_{6xy} = \hat{I}_{6yz} = 0 \tag{3.7.15}$$

$$\hat{I}_{jxxR} = \hat{I}_{jxx} = \hat{I}_{jyy} \tag{3.7.16}$$

$$\hat{I}_{(j-1)xxR} = \hat{I}_{(j-1)xx} + \hat{I}_{jyy} + 2d_jM_{jz} + d^2{}_jM_j \tag{3.7.17}$$

$$\hat{I}_{(j-1)xyR} = \hat{I}_{(j-1)xy} + a_{j-1}S\alpha_{j-1}M_{jz} + a_{j-1}d_jS\alpha_{j-1}M_j \tag{3.7.18}$$

$$\hat{I}_{(j-1)xzR} = \hat{I}_{(j-1)xz} - a_{j-1}C\alpha_{j-1}M_{jz} - a_{j-1}d_jC\alpha_{j-1}M_j \tag{3.7.19}$$

$$\hat{I}_{(j-1)yyR} = \hat{I}_{(j-1)yy} + CC\alpha_{j-1}I_{jyy} + 2d_jCC\alpha_{j-1}M_{jz} + (a^2{}_{j-1} + d^2{}_jCC\alpha_{j-1})M_j \tag{3.7.20}$$

$$\hat{I}_{(j-1)yzR} = \hat{I}_{(j-1)yz} + CS\alpha_{j-1}I_{jyy} + 2d_jCS\alpha_{j-1}M_{jz} + d^2{}_jCS\alpha_{j-1}M_j \tag{3.7.21}$$

$$\hat{I}_{(j-1)zzR} = \hat{I}_{(j-1)zz} + SS\alpha_{j-1}I_{jyy} + 2d_jSS\alpha_{j-1}M_{jz} + (a^2{}_{j-1} + d^2{}_jSS\alpha_{j-1})M_j \tag{3.7.22}$$

$$M_{(j-1)xR} = M_{(j-1)x} + a_{j-1}M_j \tag{3.7.23}$$

$$M_{(j-1)yR} = M_{(j-1)y} - S\alpha_{j-1}M_{jz} - d_jS\alpha_jM_j \tag{3.7.24}$$

$$M_{(j-1)zR} = M_{(j-1)z} + C\alpha_{j-1}N_{jz} + d_jC\alpha_{j-1}M_j \tag{3.7.25}$$

$$M_{(j-1)R} = M_{(j-1)} + M_j \tag{3.7.26}$$

$$\pi = \begin{bmatrix} \hat{I}_{1zzR} & M_{2yR} & \hat{I}_{2xxR} & \hat{I}_{2yzR} & \hat{I}_{2zzR} & \hat{I}_{m_3} & \hat{I}_{3xxR} \\ \hat{I}_{3zzR} & I_{m_4} & M_{4yR} & \hat{I}_{4xxR} & \hat{I}_{4yzR} & \hat{I}_{4zzR} & I_{m_5} \\ \hat{I}_{5xxR} & \hat{I}_{5zzR} & \hat{I}_{m_6} & M_{6yR} & \hat{I}_{6xxR} & \hat{I}_{6zzR} & \hat{I}_{m_7} \end{bmatrix}^T \tag{3.7.27}$$

$$\pi_1 = \hat{I}_{1zzR} = \hat{I}_{1zz} + \hat{I}_{2yy} + K_r^2 \hat{I}_{rni} = 6.3376 gm^2 \quad (3.7.28)$$

$$\pi_2 = M_{2yR} = L_{C_{2y}m_2} - L_{C_3z} m_3 - d_3(m_3 + m_4 + m_5 + m_6) \quad (3.7.29)$$
$$= -7.3424\ Kgm$$

$$\pi_3 = \hat{I}_{2xxR} = \hat{I}_{2xx} + \hat{I}_{3yy} + 2d_3 L_{C_{3z}} m_3 \quad (3.7.30)$$
$$+ d_3^2(m_3 + m_4 + m_5 + m_6) - \hat{I}_{2yy}$$
$$= 2.9419\ Kgm^2$$

$$\pi_4 = \hat{I}_{2yzR} = \hat{I}_{2yz} = 3.7152\ kg.m^2 \quad (3.7.31)$$

$$\pi_5 = \hat{I}_{2zzR} = \hat{I}_{2zz} + \hat{I}_{3yy} + 2d_3 lC_{3z} m_3 \quad (3.7.32)$$
$$+ d_3^2(m_3 + m_4 + m_5 + m_6) + K_r^2 \hat{I}_{rn2}$$
$$= 4.2823\ kg.m^2$$

$$\pi_6 = \hat{I}_{rn3} = 4.7890.10^{-4}\ kg.m^2 \quad (3.7.33)$$

$$\pi_7 = \hat{I}_{3xxR} = \hat{I}_{3xx} + \hat{I}_{4yy} - \hat{I}_{3yy} = -1.5248\ kg.m^2 \quad (3.7.34)$$

$$\pi_8 = \hat{I}_{3zzR} = \hat{I}_{3zz} + \hat{I}_{4yy} = 3.8092\ kg.m^2 \quad (3.7.35)$$

$$\pi_9 = I_{m_4} = 2.0595.10^{-4}\ kg.m^2 \quad (3.7.36)$$

$$\pi_{10} = M_{4yR} = lC_{4y} m_4 - lC_{5z} m_5 - d_5(m_5 + m_6) \quad (3.7.37)$$
$$= -3.2317\ kg.m^2$$

$$\pi_8 = \hat{I}_{4xxR} = \hat{I}_{4xx} + \hat{I}_{5yy} + 2d_5 lC_{5z} m_5 + d_5^2(m_5 + m_6) - \hat{I}_{4yy} \quad (3.7.38)$$
$$= 0.7116\ kg.m^2$$

$$\pi_{12} = \hat{I}_{4yzR} = \hat{I}_{4yz} = 0.4189\ kg.m^2 \quad (3.7.39)$$

$$\pi_{13} = \hat{I}_{4zzR} = \hat{I}_{4zz} + \hat{I}_{5yy} + 2d_5 lC_{5z} m_5 + d_5^2(m_5 + m_6) \quad (3.7.40)$$
$$= 0.4961\ kg.m^2$$

$$\pi_{14} = I_{m_5} = 3.2303.10^{-4}\ kg.m^2 \quad (3.7.41)$$

$$\pi_{15} = \hat{I}_{4xzR} = \hat{I}_{5xx} + \hat{I}_{5yy} = 0.2917 kg.m^2 \quad (3.7.42)$$

$$\pi_{16} = \hat{I}_{5zzR} = \hat{I}_{5zz} + \hat{I}_{6yy} = 0.1309 kg.m^2 \quad (3.7.43)$$

$$\pi_{17} = \hat{I}_{m_6} = 2.5371.10^{-4} kg.m^2 \quad (3.7.44)$$

$$\pi_{18} = M_{6yR} = lC_{6y} m_6 = 0.05377 kg.m^2 \quad (3.7.45)$$

$$\pi_{19} = \hat{I}_{6xzR} = \hat{I}_{6xx} - \hat{I}_{6yy} = 0.1861\ kg.m^2 \quad (3.7.46)$$

$$\pi_{20} = \hat{I}_{6zzR} = \hat{I}_{6zz} = 0.0747\ kg.m^2 \quad (3.7.47)$$

$$\pi_{21} = \hat{I}_{m_7} = 9.2458.10^{-6} kg.m^2 \quad (3.7.48)$$

APPENDIX 3. MATLAB CODE FOR MODELLING PA107C ROBOTIC ARM

```
clear all
close all
clc
clear L
% Here, SI unit of length is meter (m)
% D-H parameters
% L1 = Link('d', 0, 'a', 1, 'alpha', pi/2)
L1 = Link('d', 0.317, 'a', 0, 'alpha', -pi/2)
%L{1} = Link([ -pi/2     0       0       .317], 'standard');
L2 = Link('d', 0, 'a', 0, 'alpha', pi/2)
```

```
%L{2} = Link([  pi/2     0       0          0], 'standard');
%L{3} = Link([ -pi/2     0       0       .450], 'standard');
L3 = Link('d', 0.450, 'a', 0, 'alpha', -pi/2)
%L{4} = Link([  pi/2     0       0          0], 'standard');
L4 = Link('d', 0, 'a', 0, 'alpha', pi/2)
%L{5} = Link([ -pi/2     0       0       .480], 'standard');
L5 = Link('d', 0.480, 'a', 0, 'alpha', -pi/2)
%L{6} = Link([  pi/2     0       0          0], 'standard');
L6 = Link('d', 0, 'a', 0, 'alpha', pi/2)
L{7} = Link([     0     0       0       .215], 'standard');
L7 = Link('d', 0.215, 'a', 0, 'alpha', 0)
% Mass is kilogram kg
%L{1}.m = 9.78;
L1.m = 9.78;
%L{2}.m = 8.41;
L2.m = 8.41;
%L{3}.m = 3.51;
L3.m = 3.51;
%L{4}.m = 4.31;
L4.m = 4.31;
%L{5}.m = 3.45;
L5.m = 3.45;
%L{6}.m = 1.46;
L6.m = 1.46;
%L{7}.m = 0.24;
L7.m = 0.24;
%#########################################################################
% not sure
%L{1}.r = [ 0    0     0 ];
%L{2}.r = [ -.3638  .006    .2275];
%L{3}.r = [ -.0203  -.0141  .070];
% L{4}.r = [ 0    .019    0];
% L{5}.r = [ 0    0    0];
% L{6}.r = [ 0    0    .032];
% L{7}.r = [ 0    0    .032];
L1.r = [ 0    0    0 ];
L2.r = [ -.3638  .006    .2275];
L3.r = [ -.0203  -.0141  .070];
L4.r = [ 0    .019    0];
L5.r = [ 0    0    0];
L6.r = [ 0    0    .032];
L7.r = [ 0    0    .032];
% L{1}.I = [  0  0.35    0   0   0   0];
% L{2}.I = [  .13    .524    .539    0   0   0];
% L{3}.I = [   .066  .086    .0125   0   0   0];
% L{4}.I = [  1.8e-3  1.3e-3  1.8e-3  0  0   0];
```

```
% L{5}.I = [  .3e-3   .4e-3   .3e-3   0  0   0];
% L{6}.I = [  .15e-3  .15e-3  .04e-3  0  0   0];
% L{7}.I = [  .15e-3  .15e-3  .04e-3  0  0   0];
L1.I = [  0  0.35    0   0   0   0];
L2.I = [  .13    .524    .539    0   0   0];
L3.I = [   .066  .086    .0125   0   0   0];
L4.I = [  1.8e-3  1.3e-3  1.8e-3  0  0   0];
L5.I = [  .3e-3   .4e-3   .3e-3   0  0   0];
L6.I = [  .15e-3  .15e-3  .04e-3  0  0   0];
L7.I = [  .15e-3  .15e-3  .04e-3  0  0   0];
% L{1}.Jm =  200e-6;
% L{2}.Jm =  200e-6;
% L{3}.Jm =  200e-6;
% L{4}.Jm =  33e-6;
% L{5}.Jm =  33e-6;
% L{6}.Jm =  33e-6;
% L{7}.Jm =  33e-6;
L1.Jm =  200e-6;
L2.Jm =  200e-6;
L3.Jm =  200e-6;
L4.Jm =  33e-6;
L5.Jm =  33e-6;
L6.Jm =  33e-6;
L7.Jm =  33e-6;
% L{1}.G =  -62.6111;
% L{2}.G =  107.815;
% L{3}.G =  -53.7063;
% L{4}.G =  76.0364;
% L{5}.G =  71.923;
% L{6}.G =  76.686;
% L{7}.G =  76.686;
L1.G =  -62.6111;
L2.G =  107.815;
L3.G =  -53.7063;
L4.G =  76.0364;
L5.G =  71.923;
L6.G =  76.686;
L7.G =  76.686;
% viscous friction (motor referenced)
% L{1}.B =   1.48e-3;
% L{2}.B =   .817e-3;
% L{3}.B =   1.38e-3;
% L{4}.B =   71.2e-6;
% L{5}.B =   82.6e-6;
% L{6}.B =   36.7e-6;
% L{7}.B =   36.7e-6;
```

```
L1.B =   1.48e-3;
L2.B =   .817e-3;
L3.B =   1.38e-3;
L4.B =   71.2e-6;
L5.B =   82.6e-6;
L6.B =   36.7e-6;
L7.B =   36.7e-6;
% Coulomb friction (motor referenced)
% L{1}.Tc = [ .395  -.435];
% L{2}.Tc = [ .126  -.071];
% L{3}.Tc = [ .132  -.105];
% L{4}.Tc = [ 11.2e-3 -16.9e-3];
% L{5}.Tc = [ 9.26e-3 -14.5e-3];
% L{6}.Tc = [ 3.96e-3 -10.5e-3];
% L{7}.Tc = [ 3.96e-3 -10.5e-3];
L1.Tc = [ .395  -.435];
L2.Tc = [ .126  -.071];
L3.Tc = [ .132  -.105];
L4.Tc = [ 11.2e-3 -16.9e-3];
L5.Tc = [ 9.26e-3 -14.5e-3];
L6.Tc = [ 3.96e-3 -10.5e-3];
L7.Tc = [ 3.96e-3 -10.5e-3];
Q0=[ 0 0 0 0 0 0 0]
%###########################################################################
%
% some useful poses
%
global qesc qhome qsafe
qesc = [0 pi/4 0 pi/2 0 pi/4 0]; % Escape angles for PA10-7C, arm up
qhome = [0 0 0 0 0 0 0]; % Home angles for PA10-7C,
qsafe = [0 pi/4 0 pi/2 0 -pi/4 0]; % Safety position,
%bot = SerialLink([L1 L2], 'name', 'my robot')
% pa107c = robot(L, 'PA10-7C', 'Mitsubishi', '');
%bot = SerialLink(L, 'PA10-7C', 'Mitsubishi', 'my robot');
% global pa10_7c
pa10_7c = SerialLink([L1 L2 L3 L4 L5 L6 L7])
Qinitial2.signals.values= [0 0 0 0 0 0 0 ]
Qinitial.signals.values= [0 0 0 0 0 0 0 ]
X=0
Y=0
Z=0
%% Extra ===============
% clear L1 L2 L3 L4 L5 L6 L7
% pa10_7c.plot([0 0 0 0 1 0.3 0.5] )
% pause(3)a
% pa10_7c.plot(qsafe )
```

```
% pause(3)
% pa10_7c.plot(qhome )
%
% %t = fkinepa10(robot, q)
pa10_7c =
noname (7 axis, RRRRRRR, stdDH, fastRNE)
+---+-----------+-----------+-----------+-----------+-----------+
| j |     theta |         d |         a |     alpha |    offset |
+---+-----------+-----------+-----------+-----------+-----------+
|  1|         q1|      0.317|          0|     -1.571|          0|
|  2|         q2|          0|          0|      1.571|          0|
|  3|         q3|       0.45|          0|     -1.571|          0|
|  4|         q4|          0|          0|      1.571|          0|
|  5|         q5|       0.48|          0|     -1.571|          0|
|  6|         q6|          0|          0|      1.571|          0|
|  7|         q7|      0.215|          0|          0|          0|
+---+-----------+-----------+-----------+-----------+-----------+
  grav =    0  base = 1  0  0  0   tool =  1  0  0  0
         0         0  1  0  0           0  1  0  0
      9.81         0  0  1  0           0  0  1  0
                   0  0  0  1           0  0  0  1
 qesc = [0 pi/4 0 pi/2 0 pi/4 0]

>> pa10_7c.fkine(qesc)
ans =
   -1.0000   -0.0000    0.0000    0.6576
    0.0000    1.0000    0.0000    0.0000
   -0.0000    0.0000   -1.0000    0.0808
         0         0         0    1.0000
```

APPENDIX 4. MATLAB CODE FOR CALCULATING THE INVERSE KINEMATICS OF THE PA10 ROBOT

```
% %Inverse kinematics development to the robot PA - 10
Clear all ; clc
syms teheta1 teheta2 teheta3 teheta4 teheta5 teheta6 L1 L2 L3 L4 nx ny nz ox oy oz ax ay az px
py pz
%The following are the DH parameters obtained from the robot shown in
dh = [teheta1+(pi/2) L1 0 pi/2 ;
teheta2+(pi/2) 0 L2 0 ;
teheta3-(pi/2) 0 0 -pi/2 ;
teheta4 L3 0 pi/2 ;
teheta5 0 0 -pi/2 ;
teheta6 L4 0 0 ; ]
TO6 = simple(T01*T12*T23*T34*T45*T56)
```

```
T03 = [ sin(teheta1)*sin(teheta2)*sin(teheta3)-sin(teheta1)*cos(teheta2)*cos(teheta3), -cos(teheta1), sin(teheta1)*sin(teheta2)*cos(teheta3)+sin(teheta1)*cos(teheta2)*sin(teheta3), sin(teheta1)*L2*sin(teheta2)]
[ -cos(teheta1)*sin(teheta2)*sin(teheta3)+cos(teheta1)*cos(teheta2)*cos(teheta3), -sin(teheta1), -cos(teheta1)*sin(teheta2)*cos(teheta3)-cos(teheta1)*cos(teheta2)*sin(teheta3), -cos(teheta1)*L2*sin(teheta2)]
[ cos(teheta2)*sin(teheta3)+sin(teheta2)*cos(teheta3), 0, cos(teheta2)*cos(teheta3)-sin(teheta2)*sin(teheta3), L2*cos(teheta2)+L1]
[ 0, 0, 0, 1]
%Where the rotation matrix is:
R03 = [ sin(teheta1)*sin(teheta2)*sin(teheta3)-sin(teheta1)*cos(teheta2)*cos(teheta3) -cos(teheta1) sin(teheta1)*sin(teheta2)*cos(teheta3)+sin(teheta1)*cos(teheta2)*sin(teheta3)
cos(teheta1)*sin(teheta2)*sin(teheta3)+cos(teheta1)*cos(teheta2)*cos(teheta3) -sin(teheta1) -cos(teheta1)*sin(teheta2)*cos(teheta3)-cos(teheta1)*cos(teheta2)*sin(teheta3)
cos(teheta2)*sin(teheta3)+sin(teheta2)*cos(teheta3) 0 cos(teheta2)*cos(teheta3)-sin(teheta2)*sin(teheta3)]
% Now we determine the inverse of this rotation matrix:
R03i = simple(inv(R03))
T36 = [ cos(teheta4)*cos(teheta5)*cos(teheta6)-sin(teheta4)*sin(teheta6), -cos(teheta4)*cos(teheta5)*sin(teheta6)-sin(teheta4)*cos(teheta6), -cos(teheta4)*sin(teheta5), -cos(teheta4)*sin(teheta5)*L4]
[ sin(teheta4)*cos(teheta5)*cos(teheta6)+cos(teheta4)*sin(teheta6), -sin(teheta4)*cos(teheta5)*sin(teheta6)+cos(teheta4)*cos(teheta6), -sin(teheta4)*sin(teheta5), -sin(teheta4)*sin(teheta5)*L4]
[ sin(teheta5)*cos(teheta6), -sin(teheta5)*sin(teheta6), cos(teheta5), cos(teheta5)*L4+L3]
[ 0, 0, 0, 1]
% Where the rotation matrix is:
R36 = [cos(teheta4)*cos(teheta5)*cos(teheta6)-sin(teheta4)*sin(teheta6) -cos(teheta4)*cos(teheta5)*sin(teheta6)-sin(teheta4)*cos(teheta6) -cos(teheta4)*sin(teheta5);
sin(teheta4)*cos(teheta5)*cos(teheta6)+cos(teheta4)*sin(teheta6) -sin(teheta4)*cos(teheta5)*sin(teheta6)+cos(teheta4)*cos(teheta6) -sin(teheta4)*sin(teheta5);
sin(teheta5)*cos(teheta6) -sin(teheta5)*sin(teheta6) cos(teheta5); ]
% The rotation matrix from base to tip is the following
R06 = [nx ox ax; nyoy ay; nzozaz;]
% R03i*R06= R36
R03i_R06 = R03i*R06
% Getting the following result
% Of Equalization, the third row element third column, we obtain
teheta5 = solve('(1/2*cos(teheta1-teheta2-teheta3)-1/2*cos(teheta1+teheta2+teheta3))*ax+(-1/2*sin(teheta1+teheta2+teheta3)+1/2*sin(teheta1-teheta2-
teheta3))*ay+cos(teheta2+teheta3)*az=cos(teheta5)',teheta5);
teheta5 = simple(teheta5)
teheta4 = solve('-cos(teheta1)*ax-sin(teheta1)*ay = -sin(teheta4)*sin(teheta5)',teheta4);
teheta4 = simple(teheta4)
% Now we will calculate teheta6, with the element of the third row first column
teheta6 = solve('(1/2*cos(teheta1-teheta2-teheta3)-1/2*cos(teheta1+teheta2+teheta3))*nx+(-
```

```
cos(teheta2).*sin(teheta3)+sin(teheta2).*cos(teheta3)).*cos(teheta4).*sin(teheta5)+(cos(teheta2).
*cos(teheta3)-sin(teheta2).*sin(teheta3)).*cos(teheta5)).*L4+(cos(teheta2).*cos(teheta3)-
sin(teheta2).*sin(teheta3)).*L3+L2.*cos(teheta2)+L1
1/2*sin(teheta1+teheta2+teheta3)+1/2*sin(teheta1-teheta2-
teheta3))*ny+cos(teheta2+teheta3)*nz = sin(teheta5)*cos(teheta6)',teheta6);
teheta6 = simple(teheta6)
% Now to obtain the given path knowing that:
L1 = 317; L2 = 450; L3 = 480; L4 = 70;
Px=-400:10:400; Py=400*ones(1,101);
Px1 = 400*ones(1,41); Py1 = 400:-10:100;
Px2 = 400:-10:-400; Py2 = 100*ones(1,101);
Px3 = -400*ones(1,41); Py3 = 0:10:400;
px=[Px Px1 Px2 Px3]; py=[Py Py1 Py2 Py3];
pz=200*ones(1,284); q0 = zeros(1,284);
nx=1; ny=0; nz=0; ox=0; oy=1; oz=0; ax=0; ay=0; az=1;
Pmx = px - (ax*L4)
Pmy = py - (ay*L4)
Pmz = pz - (az*L4)
teheta1 = atan(-Pmx./Pmy);
teheta3 = -acos((Pmx.^2+Pmy.^2+(Pmz-L1).^2-L3.^2-L2.^2)./(2.*L2.*L3));
teheta2 = -acos(((Pmz-L1).*(L3.*cos(teheta3)+L2)+(sin(teheta1).*Pmx-
cos(teheta1).*Pmy).*(L3.*sin(teheta3)))./((sin(teheta1).*Pmx-cos(teheta1).*Pmy).^2+(Pmz-
L1).^2));
teheta5 =acos(1/2.*ax.*cos(teheta1-teheta2-teheta3)-1/2.*ax.*cos(teheta1+teheta2+teheta3)-
1/2.*ay.*sin(teheta1+teheta2+teheta3)+1/2.*ay.*sin(teheta1-teheta2-
teheta3)+cos(teheta2+teheta3).*az)
teheta4 =asin((cos(teheta1).*ax+sin(teheta1).*ay)./sin(teheta5))
teheta6 =-acos(1/2.*(nx.*cos(teheta1-teheta2-teheta3)-nx.*cos(teheta1+teheta2+teheta3)-
ny.*sin(teheta1+teheta2+teheta3)+ny.*sin(teheta1-teheta2-
teheta3)+2.*cos(teheta2+teheta3).*nz)./sin(teheta5))
x = (-((sin(teheta1).*sin(teheta2).*sin(teheta3)-
sin(teheta1).*cos(teheta2).*cos(teheta3)).*cos(teheta4)-
cos(teheta1).*sin(teheta4)).*sin(teheta5)+(sin(teheta1).*sin(teheta2).*cos(teheta3)+sin(teheta1).*
cos(teheta2).*sin(teheta3)).*cos(teheta5)).*L4+(sin(teheta1).*sin(teheta2).*cos(teheta3)+sin(tehe
ta1).*cos(teheta2).*sin(teheta3)).*L3+sin(teheta1).*L2.*sin(teheta2)
y = (-((cos(teheta1).*cos(teheta2).*cos(teheta3)-
cos(teheta1).*sin(teheta2).*sin(teheta3)).*cos(teheta4)-
sin(teheta1).*sin(teheta4)).*sin(teheta5)+(-cos(teheta1).*sin(teheta2).*cos(teheta3)-
cos(teheta1).*cos(teheta2).*sin(teheta3)).*cos(teheta5)).*L4+(-
cos(teheta1).*sin(teheta2).*cos(teheta3)-cos(teheta1).*cos(teheta2).*sin(teheta3)).*L3-
cos(teheta1).*L2.*sin(teheta2)
%% Matlab code for generating the cubic trajectories
function q = q_cubic_fn(qs, qg, t)
a=qs; c=3*(qg-qs)/(t*t);d=-2*(qg-qs)/(t*t*t);tt= ceil(20*t);
for i=1:tt
q(i)=(a+(c*i*i/200)+(d*i*i*i/4000)); end
```

```
function qd = qd_cubic_fn(qs, qg, t)
a=qs; c=3*(qg-qs)/(t*t); d=-2*(qg-qs)/(t^3); tt= ceil(20*t);
for i=1:tt qd(i)=((2*c*i/20)+(3*d*i*i/200)); end
function qdd = qdd_cubic_fn(qs, qg, t)
a=qs; c=3*(qg-qs)/(t*t); d=-2*(qg-qs)/(t*t*t); tt= ceil(20*t);
for i=1:tt qdd(i)=(2*c)+(6*d*i/20); end
```

APPENDIX 5. VB CODE FOR PA10 ROBOT DATA LOGGING TO THE DATABASE

```
==================================Data Base Recording========
Data1.Recordset.AddNew
Data1.Recordset.Fields("S1c") = Me.Label2(0).Caption
Data1.Recordset.Fields("S2c") = Label2(1)
Data1.Recordset.Fields("S3c") = Label2(2)
Data1.Recordset.Fields("E1c") = Label2(3)
Data1.Recordset.Fields("E2c") = Label2(4)
Data1.Recordset.Fields("W1c") = Label2(5)
Data1.Recordset.Fields("W2c") = Label2(6)
'========
Data1.Recordset.Fields("S1t") = Label4(0).Caption
Data1.Recordset.Fields("S2t") = Label4(1).Caption
Data1.Recordset.Fields("S3t") = Label4(2).Caption
Data1.Recordset.Fields("E1t") = Label4(3).Caption
Data1.Recordset.Fields("E2t") = Label4(4).Caption
Data1.Recordset.Fields("W1t") = Label4(5).Caption
Data1.Recordset.Fields("W2t") = Label4(6).Caption
'=========
Data1.Recordset.Fields("Xc") = 1
Data1.Recordset.Fields("Yc") = 2
Data1.Recordset.Fields("Zc") = Label8(2)
Data1.Recordset.Fields("length") = Label8(3)
Data1.Recordset.Fields("Yawc") = Label8(4)
Data1.Recordset.Fields("Pitchc") = Label8(5)
Data1.Recordset.Fields("Rollc") = Label8(6)
Data1.Recordset.Fields("Xt") = Label9(0)
Data1.Recordset.Fields("Yt") = Label9(1)
Data1.Recordset.Fields("Zt") = Label9(2)
Data1.Recordset.Fields("Yawt") = Label9(4)
Data1.Recordset.Fields("Pitcht") = Label9(5)
Data1.Recordset.Fields("Rollt") = Label9(6)
Data1.Recordset.Update
Private Sub Command1_Click()
'Dim Matlab As Object
Dim server_version As String
'Matlab = CreateObject("matlab.application")
server_version = Matlab.Execute("version")
Text1.Text = server_version
```

```
'Dim h As Object
'h = GetObject(, "MLApp.MLApp")
' Handle h should be valid now. Test it by calling Execute.
'h.Execute ("plot([0 18], [7 23])")
End Sub
Private Sub Command2_Click()
result = Matlab.Execute("cd ('C:\Program Files\MATLAB\R2006a\Work\2015')")
End Sub
Private Sub Form_Load()
Dim result As String
Set Matlab = New MLApp.MLApp
result = Matlab.Execute("cd ('C:\Program Files\MATLAB\R2006a\Work\2015')")
End Sub
Private Sub Matlab_hallo_Click()
Dim result As String
'result = Matlab.Execute("strcat", 1, "hello", " world")
'result = Matlab.Execute("5*2")
'result = Matlab.PutCharArray("str", "He jests at scars that never felt a wound.")
Text1.Text = 1
Dim XReal(4, 4) As Double
Dim XImag(4, 4) As Double
Dim ZReal(4, 4) As Double
Dim ZImag(4, 4) As Double
Dim i, j As Integer
For i = 0 To 4
  For j = 0 To 4
  XReal(i, j) = Rnd() * 6
  XImag(i, j) = 0
  Next j
Next i
'result = Matlab.PutFullMatrix("M", "base", XReal, XImag)
Text1.Text = result
End Sub

Private Sub RunFile_Click()
Dim result As String
result = Matlab.Execute("cd ('C:\Program Files\MATLAB\R2006a\Work\2015')")
'result = Matlab.Execute("5+5")
'Text1.Text = result
result = Matlab.Execute("plotSinwave")
End Sub
Private Sub StartMatlab_Click()
Dim result As String
result = Matlab.Execute("cd ('C:\Program Files\MATLAB\R2011b\Work')")
result = Matlab.Execute("5*2")
Text1.Text = result
```

```
'result = Matlab.PutWorkspaceData('A', 'base', rand(6))
End Sub
Dim MatLab As New MLApp.MLApp
Dim Result As String
Result = MatLab.Execute("cd ('C:\Program Files\MATLAB\R2006a\Work\2015')")
'result = Matlab.Execute("5+5")
'Text1.Text = result
Result = MatLab.Execute("plotSinwave")
```

APPENDIX 6. MATLAB CODE FOR DATA LOGGING TO MATLAB

```
%Database2
% Program Written by Meftah Mohamed to draw database data in graph and
% logg data in Matalb array
clear all
close all
clc
% Define the data base connection
setdbprefs({'DataReturnFormat','ErrorHandling','NullNumberRead','NullNumberWrite','NullStri
ngRead','NullStringWrite','JDBCDataSourceFile'},{'cellarray','store','NaN','NaN','null','null',''});
conn = database('PA10dataBase','','');
delay = .01;                % make sure sample faster than resolution
time = 0;
data = 0;
count = 0;
disp('Close Plot to End Session');
a=1;
tic
while (a>0)
e = exec(conn,'SELECT LAST (S2t)  FROM ANGLE');
e = fetch(e);
dat=e.data;
count = count + 1;
time(count) = toc;    %Extract Elapsed Time
data(count) =cell2mat(e.data);% dat(1); %Extract 1st Data Element
pause(delay);
end
disp('Session Terminated...');
%=====================================================
%Database3
% Program Written by Meftah Mohamed to draw database data in graph and
% logg data in Matalb array
clear all
close all
clc
% Define the data base connection
```

```
setdbprefs({'DataReturnFormat','ErrorHandling','NullNumberRead','NullNumberWrite','NullStri
ngRead','NullStringWrite','JDBCDataSourceFile'},{'cellarray','store','NaN','NaN','null','null',''});
conn = database('PA10dataBase','','');
% prompt = {'Enter joint name:'};
% dlg_title = 'Input';
% num_lines = 1;
% defaultans = {'Sc1'};
% DBvariable= inputdlg(prompt,dlg_title,num_lines,defaultans)
prompt = 'What is the joint you want plot ? ';
DBvariable = input(prompt,'s')
% Graph parameters
plotTitle = 'Robot joint  Data Log';  % plot title

xLabel = 'Elapsed Time (s)';    % x-axis label
yLabel = 'Data';                % y-axis label
plotGrid = 'on';                % 'off' to turn off grid
min = -1.5;                     % set y-min
max = 1.5;                      % set y-max
scrollWidth = 10;               % display period in plot, plot entire data log if <= 0
delay = .01;                    % make sure sample faster than resolution
%Define Function Variables
time = 0;
data = 0;
count = 0;
 %Set up Plot
plotGraph = plot(time,data,'-mo',...
            'LineWidth',1,...
            'MarkerEdgeColor','k',...
            'MarkerFaceColor',[.49 1 .63],...
            'MarkerSize',2);

title(plotTitle,'FontSize',25);
xlabel(xLabel,'FontSize',15);
ylabel(yLabel,'FontSize',15);
axis([0 10 min max]);
grid(plotGrid);
%Open Database
disp('Close Plot to End Session');
a=1;
tic
 while  ishandle(plotGraph) %Loop when Plot is Active
    %  Read the last value in the record
e = exec(conn,'SELECT LAST (DBvariable)  FROM ANGLE');
e = fetch(e);
%dat=e.data;
%dat=1.5;
```

```
%dat = fscanf(s,'%f'); %Read Data from Serial as Float

   %if(~isempty(dat) && isfloat(dat)) %Make sure Data Type is Correct
      count = count + 1;
      time(count) = toc;    %Extract Elapsed Time
      data(count) = e.data %cell2mat(e.data); %Extract 1st Data Element
           %Set Axis according to Scroll Width
      if(scrollWidth > 0)
      set(plotGraph,'XData',time(time       >       time(count)-scrollWidth),'YData',data(time       >
time(count)-scrollWidth));
      axis([time(count)-scrollWidth time(count) min max]);
      else
      set(plotGraph,'XData',time,'YData',data);
      axis([0 time(count) min max]);
      end
        %Allow MATLAB to Update Plot
      pause(delay);
   %end  %% end of if check
end
 %Close Serial COM Port and Delete useless Variables
clear count dat delay max min plotGraph plotGrid plotTitle s ...
      scrollWidth  xLabel yLabel;
 close(e)
close(conn)
disp('Session Terminated...');
```

APPENDIX 7. SAMPLE OF LUA CODE WRITTEN TO SIMULATE PA10 ROBOT IN V-REP

```
if (sim_call_type==sim_childscriptcall_initialization) then
simExtRemoteApiStart(19999)
simSetScriptAttribute(sim_handle_self,sim_childscriptattribute_automaticcascadingcalls,false)
end
if (sim_call_type==sim_childscriptcall_cleanup) then

end
if (sim_call_type==sim_childscriptcall_sensing) then
  simHandleChildScripts(sim_call_type)
end
      jHandles={-1,-1,-1,-1,-1,-1,-1}
      jHandles[1]=simGetObjectHandle("Joint0")
      jHandles[2]=simGetObjectHandle("Joint1")
      jHandles[3]=simGetObjectHandle("Joint2")
      jHandles[4]=simGetObjectHandle("Joint3")
      jHandles[5]=simGetObjectHandle("Joint4")
      jHandles[6]=simGetObjectHandle("Joint5")
      jHandles[7]=simGetObjectHandle("Joint6")
```

```
    gripperJ=simGetObjectHandle("gripper_joint")
    fingerJ=simGetObjectHandle("finger_joint")
    target=simGetObjectHandle("M_target")
    tip=simGetObjectHandle("M_tip")
    ui=simGetUIHandle("manipulatorUi")
    ik1=simGetIkGroupHandle("M_IK_Group")
    ik2=simGetIkGroupHandle("M_IK_Group2")
    ik3=simGetIkGroupHandle("M_IK_Group3")
    selfColl=simGetCollisionHandle("SelfCollision")
    environmentColl=simGetCollisionHandle("CollisionWithEnvironment")
    selfCollReportHandle=-1
    envCollReportHandle=-1
    ikPosAndOrientReportHandle=-1
    ikOrientReportHandle=-1
    ikPosAndOrientReportHandle=-1
    robotBase=simGetObjectHandle("RedundantManipulator")
desiredJ={0,0,0,math.pi/2,0,-math.pi/2,0} -- when in FK mode
    for i=1,7,1 do
        simSetJointPosition(jHandles[i],desiredJ[i]) -- Make sure we start in the default configuration
    end


if ikMode then
    -- We are in IK mode
    maxVariationAllowed=maxPosVelocity*simGetSimulationTimeStep()
    deltaX={0,0,0,0,0,0}
    -- position:
    for i=1,3,1 do
        delta=desiredPos[i]-currentPos[i]
        if (math.abs(delta)>maxVariationAllowed) then
            delta=maxVariationAllowed*delta/math.abs(delta) -- we limit the variation to the maximum allowed
        end
        deltaX[i]=delta
    end
    -- orientation:
    maxVariationAllowed=maxOrVelocity*simGetSimulationTimeStep()
    delta=desiredPos[4]-currentPos[4]
    -- Normalized delta to be between -pi and +pi:
    delta=math.fmod(delta,math.pi*2)
    if (delta<-math.pi) then
        delta=delta+math.pi*2
    else
        if (delta>math.pi) then
            delta=delta-math.pi*2
```

```
    end
  end

  if (math.abs(delta)>maxVariationAllowed) then
    delta=maxVariationAllowed*delta/math.abs(delta) -- we limit the variation to the maximum allowed
  end
  deltaX[4]=delta

  delta=desiredPos[5]-currentPos[5]
  -- Normalized delta to be between -pi/2 and +pi/2:
  delta=math.fmod(delta,math.pi)
  if (delta<-math.pi/2) then
    delta=delta+math.pi
  else
    if (delta>math.pi/2) then
      delta=delta-math.pi
    end
  end

  if (math.abs(delta)>maxVariationAllowed) then
    delta=maxVariationAllowed*delta/math.abs(delta) -- we limit the variation to the maximum allowed
  end
  deltaX[5]=delta

  delta=desiredPos[6]-currentPos[6]
  -- Normalized delta to be between -pi and +pi:
  delta=math.fmod(delta,math.pi*2)
  if (delta<-math.pi) then
    delta=delta+math.pi*2
  else
    if (delta>math.pi) then
      delta=delta-math.pi*2
    end
  end

  if (math.abs(delta)>maxVariationAllowed) then
    delta=maxVariationAllowed*delta/math.abs(delta) -- we limit the variation to the maximum allowed
  end
  deltaX[6]=delta


currentPos={currentPos[1]+deltaX[1],currentPos[2]+deltaX[2],currentPos[3]+deltaX[3],currentPos[4]+deltaX[4],currentPos[5]+deltaX[5],currentPos[6]+deltaX[6]}
```

```
    -- Normalize the orientation part to display normalized values:
    for i=1,3,1 do
        f=1
        if i==2 then f=0.5 end
        currentPos[3+i]=math.fmod(currentPos[3+i],math.pi*2*f)
        if (currentPos[3+i]<-math.pi*f) then
            currentPos[3+i]=currentPos[3+i]+math.pi*2*f
        else
            if (currentPos[3+i]>math.pi*f) then
                currentPos[3+i]=currentPos[3+i]-math.pi*2*f
            end
        end
    end


pos={initialTipPosRelative[1]+currentPos[1],initialTipPosRelative[2]+currentPos[2],initialTipP
osRelative[3]+currentPos[3]}

orient={initialTipOrRelative[1]+currentPos[4],initialTipOrRelative[2]+currentPos[5],initialTipO
rRelative[3]+currentPos[6]}
    -- We set the desired position and orientation
    simSetObjectPosition(target,robotBase,pos)
    simSetObjectOrientation(target,robotBase,orient)

    if (simHandleIkGroup(ik1)==sim_ikresult_fail) then
        -- the position AND orientation could not be reached.
        -- We try to reach just the position:
        if (simHandleIkGroup(ik2)==sim_ikresult_fail) then
            simHandleIkGroup(ik3) -- Even the position could not be reached!
            if (ikOrientReportHandle>=0) then
                simEndDialog(ikOrientReportHandle) -- We close any report message about IK
failure in orientaion
                ikOrientReportHandle=-1
            end
            if (ikPosAndOrientReportHandle==-1) then -- We display a IK failure (in pos and
orientation) report message
                ikPosAndOrientReportHandle,elementHandle=simDisplayDialog("IK failure
report","IK solver failed for position AND
orientation.",sim_dlgstyle_message,false,"",nil,{0.8,0,0,0,0,0})
                ep=simGetUIPosition(elementHandle)
                ep[2]=ep[2]+100 -- we shift the dialog down by 100 pixels
                simSetUIPosition(elementHandle,ep)
            end
        else
            if (ikPosAndOrientReportHandle>=0) then
```

```
            simEndDialog(ikPosAndOrientReportHandle) -- We close any report message about
IK failure in orientaion AND position
            ikPosAndOrientReportHandle=-1
          end
          if (ikOrientReportHandle==-1) then -- We display a IK failure (in orientation only)
report message
            ikOrientReportHandle,elementHandle=simDisplayDialog("IK failure report","IK
solver failed for orientation only.",sim_dlgstyle_message,false,"",nil,{1,0.7,0,0,0,0})
            ep=simGetUIPosition(elementHandle)
            ep[2]=ep[2]+100 -- we shift the dialog down by 100 pixels
            simSetUIPosition(elementHandle,ep)
          end
        end
      else
        if (ikPosAndOrientReportHandle>=0) then
          simEndDialog(ikPosAndOrientReportHandle) -- We close any report message about IK
failure in position and orientaion
          ikPosAndOrientReportHandle=-1
        end
        if (ikOrientReportHandle>=0) then
          simEndDialog(ikOrientReportHandle) -- We close any report message about IK failure
in orientaion
          ikOrientReportHandle=-1
        end
      end
      -- Now update the desiredJ in case we switch back to FK mode:
      for i=1,7,1 do
        desiredJ[i]=simGetJointPosition(jHandles[i])
      end

    else
      -- We are in FK mode
      currentJ={0,0,0,0,0,0,0}
      for i=1,7,1 do
        currentJ[i]=simGetJointPosition(jHandles[i])
      end
      maxVariationAllowed=maxJointVelocity*simGetSimulationTimeStep()
      for i=1,7,1 do
        delta=desiredJ[i]-currentJ[i]
        if (math.abs(delta)>maxVariationAllowed) then
          delta=maxVariationAllowed*delta/math.abs(delta) -- we limit the variation to the
maximum allowed
        end
        simSetJointPosition(jHandles[i],currentJ[i]+delta)
      end
      -- Now make sure that everything is ok if we switch to IK mode:
```

```
        simSetObjectPosition(target,-1,simGetObjectPosition(tip,-1))
        simSetObjectOrientation(target,-1,simGetObjectOrientation(tip,-1))
        mbase=simGetObjectMatrix(robotBase,-1)
        m=simGetObjectMatrix(target,-1)
        m=simMultiplyMatrices(simGetInvertedMatrix(mbase),m)
        -- m is now the tip position relative to the base.
        -- We now want m to be the tip position relative to the initial position:
        m=simMultiplyMatrices(simGetInvertedMatrix(initialTipMatrixRelative),m)
        orient=simGetEulerAnglesFromMatrix(m)
        desiredPos={m[4],m[8],m[12],orient[1],orient[2],orient[3]}
        for i=1,6,1 do
            currentPos[i]=desiredPos[i]
        end
        -- Close any IK warning dialogs:
        if (ikPosAndOrientReportHandle>=0) then
            simEndDialog(ikPosAndOrientReportHandle) -- We close any report message about IK
failure in position and orientaion
            ikPosAndOrientReportHandle=-1
        end
        if (ikOrientReportHandle>=0) then
            simEndDialog(ikOrientReportHandle) -- We close any report message about IK failure in
orientaion
            ikOrientReportHandle=-1
        end

    end
    -- Check for robot self-collisions:
    if (simReadCollision(selfColl)~=0) then
        if (selfCollReportHandle==-1) then -- We display a collision report message
            selfCollReportHandle,elementHandle=simDisplayDialog("Collision report","Robot self-
collision detected.",sim_dlgstyle_message,false,"",nil,{0.8,0,0,0,0,0,0})
            ep=simGetUIPosition(elementHandle)
        end
    else
        if (selfCollReportHandle>=0) then
            simEndDialog(selfCollReportHandle) -- We close the report message about collision
            selfCollReportHandle=-1
        end
    end
        -- Check for robot-environment collisions:
    if (simReadCollision(environmentColl)~=0) then
        if (envCollReportHandle==-1) then -- We display a collision report message
            envCollReportHandle,elementHandle=simDisplayDialog("Collision report","Robot-
environment collision detected.",sim_dlgstyle_message,false,"",nil,{0.8,0,0,0,0,0,0})
            ep=simGetUIPosition(elementHandle)
            ep[2]=ep[2]-100 -- we shift the dialog up by 100 pixels
```

```
        simSetUIPosition(elementHandle,ep)
    end
else
    if (envCollReportHandle>=0) then
        simEndDialog(envCollReportHandle) -- We close the report message about collision
        envCollReportHandle=-1
    end
end
```

APPENDIX 8. MATLAB CODE FOR CALCULATING THE FORWARD KINEMATICS OF PA10 ROBOT

```
Clear all ; clc
Syms q1 q2 q3 q4 q5 q6 L1 L2 L3 L4 nx ny nz ox oy oz ax ay az px py pz
%The following are the DH parameters obtained from the robot: dh =
[q1+(pi/2) L1 0 pi/2 ;
q2+(pi/2) 0 L2 0 ;
q3-(pi/2) 0 0 -pi/2 ;
q4 L3 0 pi/2 ;
q5 0 0 -pi/2 ;
q6 L4 0 0 ; ]
T01 = [ -sin(q1), 0, cos(q1), 0]
[ cos(q1), 0, sin(q1), 0]
[ 0, 1, 0, L1]
[ 0, 0, 0, 1]
T12 = [ -sin(q2), -cos(q2), 0, -L2*sin(q2)]
[ cos(q2), -sin(q2), 0, L2*cos(q2)]
[ 0, 0, 1, 0]
[ 0, 0, 0, ]
T23 = [ sin(q3), 0, cos(q3), 0]
[ -cos(q3), 0, sin(q3), 0]
[ 0, -1, 0, 0]
[ 0, 0, 0, 1]
T34 = [ cos(q4), 0, sin(q4), 0]
[ sin(q4), 0, -cos(q4), 0]
[ 0, 1, 0, L3]
[ 0, 0, 0, 1]
T45 = [ cos(q5), 0, -sin(q5), 0]
[ sin(q5), 0, cos(q5), 0]
[ 0, -1, 0, 0]
[ 0, 0, 0, 1]
T56 = [ cos(q6), -sin(q6), 0, 0]
[ sin(q6), cos(q6), 0, 0]
[ 0, 0, 1, L4]
[ 0, 0, 0, 1]
% The forward Kinematic of the PA10 robot
TO6 = simple(T01*T12*T23*T34*T45*T56)
```

```
X=T06(1,4)
Y=T06(2,4)
Z=(T06(3,4)
% Finally we plot the movement of the robot with the line
Figure
plot3(px,py,pz)
grid
```

APPENDIX 9. MATLAB CODE FOR INTERFACING BETWEEN MATLAB AND 3D SLICER

```
% Draw profile
% 3 translations ; X, Y fixed ; Z is variable
open('slicer_Example1')
%https://www.mathworks.com/help/simulink/slref/get_param.html
%open('vdp')
%BlockParameterValue = get_param('vdp/Fcn','Expression')
%Z = get_param('slicer_Example1/Constant','value')
%Z=str2double(Z)
%%
%x = get_param('slicer_Example1/x_value','value')
%x=str2double(x)
%y = get_param('slicer_Example1/y_value','value')
%y=str2double(y)
%z = get_param('slicer_Example1/z_value','value')
%z=str2double(z)
Z=7;
igtlConnection = igtlConnect('127.0.0.1',18944);
transform.name = 'NeedleToTracker';
startTime = igtlTimestampNow();
for t=1:10000
  t=igtlTimestampNow()-startTime;
  %transform.matrix = [ 1 0 0 12; 0 1 0 -5; 0 0 1 30*sin(t*0.5); 0 0 0 1 ];
x = get_param('slicer_Example1/x_value','value')
x=str2double(x)
y = get_param('slicer_Example1/y_value','value')
y=str2double(y)
z = get_param('slicer_Example1/z_value','value')
z=str2double(z)
 transform.matrix = [ 1 0 0 x; 0 1 0 y; 0 0 1 z; 0 0 0 1 ];
transform.timestamp = igtlTimestampNow();
  transform
  igtlSendTransform(igtlConnection, transform);
  pause(0.1)
end
 igtlDisconnect(igtlConnection);
```

APPENDIX 10. MATLAB CODE FOR VOICE RECOGNITION USING CORRELATION TECHNIQUE

```
%Voice04
% Tested on Matlab 2017a
clc
clear all
close all
%Speech Recognition Using Correlation Method
%Write Following Command On Command Window
%speechrecognition('test.wav');
%[voice,Fs] = audioread('test.wav');
%x=voice;
x = audioread('test2.wav');
x=x';
x=x(1,:);
x=x';
y1=audioread('one.wav');
y1=y1';
y1=y1(1,:);
y1=y1';
z1=xcorr(x,y1);
m1=max(z1);
l1=length(z1);
t1=-((l1-1)/2):1:((l1-1)/2);
t1=t1';
%subplot(3,2,1);
%plot(t1,z1);
y2=audioread('two.wav');
y2=y2';
y2=y2(1,:);
y2=y2';
z2=xcorr(x,y2);
m2=max(z2);
l2=length(z2);
t2=-((l2-1)/2):1:((l2-1)/2);
t2=t2';
%subplot(3,2,2);
%figure
%plot(t2,z2);
y3=audioread('three.wav');
y3=y3';
y3=y3(1,:);
y3=y3';
z3=xcorr(x,y3);
m3=max(z3);
l3=length(z3);
```

```
t3=-((l3-1)/2):1:((l3-1)/2);
t3=t3';
%subplot(3,2,3);
%figure
%plot(t3,z3);
y4=audioread('four.wav');
y4=y4';
y4=y4(1,:);
y4=y4';
z4=xcorr(x,y4);
m4=max(z4);
l4=length(z4);
t4=-((l4-1)/2):1:((l4-1)/2);
t4=t4';
%subplot(3,2,4);
%figure
%plot(t4,z4);
y5=audioread('five.wav');
y5=y5';
y5=y5(1,:);
y5=y5';
z5=xcorr(x,y5);
m5=max(z5);
l5=length(z5);
t5=-((l5-1)/2):1:((l5-1)/2);
t5=t5';
%subplot(3,2,5);
%figure
%plot(t5,z5);
m6=300;
a=[m1 m2 m3 m4 m5 m6];
m=max(a);
h=audioread('allow.wav');
if m<=m1
   soundsc(audioread('one.wav'),50000)
      soundsc(h,50000)
elseif m<=m2
   soundsc(audioread('two.wav'),50000)
      soundsc(h,50000)
elseif m<=m3
   soundsc(audioread('three.wav'),50000)
      soundsc(h,50000)
elseif m<=m4
   soundsc(audioread('four.wav'),50000)
      soundsc(h,50000)
elseif m<m5
```

```
   soundsc(audioread('five.wav'),50000)
      soundsc(h,50000)
else
  {soundsc(audioread('denied.wav'),50000)}
end
```

APPENDIX 11. SAMPLE OF MATLAB CODE FOR CONNECTING MATLAB WITH V-REP

```
% Last Test: June 17 2017
% Author: Meftah Mohamed
% Objectives:  This file connects Matlab with VREP  V3.4.0 rev. 1 on April 5th 2017
% Make sure to have the server side running in V-REP:
% in a child script of a V-REP scene, add following command
% to be executed just once, at simulation start:
% simExtRemoteApiStart(19999)
% then start simulation, and run this program.
% please check website for more commands
% http://www.coppeliarobotics.com/helpFiles/en/remoteApiFunctionsMatlab.htm
clear all
close all
clc
vrep=remApi('remoteApi');
vrep.simxFinish(-1);
clientID=vrep.simxStart('127.0.0.1',19999,true,true,5000,5);
% Now send some data to V-REP in a non-blocking fashion:
vrep.simxAddStatusbarMessage(clientID,'Start Comunication between Matlab and
VREP',vrep.simx_opmode_oneshot);
  %%
  pause(2);
    if (clientID>-1)
disp('Connected to remote API server');
 % Now try to retrieve data in a blocking fashion (i.e. a service call):
[res,objs]=vrep.simxGetObjects(clientID,vrep.sim_handle_all,vrep.simx_opmode_blocking);
      if (res==vrep.simx_return_ok)
         fprintf('Number of objects in the scene: %d\n',length(objs));
      else
         fprintf('Remote API function call returned with error code: %d\n',res);
      end
%%

[returnCode,target]=vrep.simxGetObjectHandle(clientID,'redundantRob_manipSphere',vrep.sim
x_opmode_blocking);
 [returnCode,array_position]=vrep.simxGetObjectPosition(clientID,target,-
1,vrep.simx_opmode_oneshot_wait)
 array_position=array_position-0.1
```

```
 [returnCode]=vrep.simxSetObjectPosition(clientID,target,-
1,array_position,vrep.simx_opmode_oneshot)
[returnCode,array_Orientation]=vrep.simxGetObjectOrientation(clientID,target,-
1,vrep.simx_opmode_oneshot_wait)
disp('array_Orientation')
array_Orientation=array_Orientation*180/pi
array_Orientation= array_Orientation-0.1
 [returnCode]=vrep.simxSetObjectOrientation(clientID,target,-
1,array_position,vrep.simx_opmode_oneshot)

%[returnCode,joint1]=vrep.simxGetObjectHandle(clientID,'redundantRob_joint1',vrep.simx_op
mode_blocking);
%[returnCode]=vrep.simxSetJointPosition(clientID,joint1,1,vrep.simx_opmode_oneshot)
%%
%  Now retrieve streaming data (i.e. in a non-blocking fashion):
%        t=clock;
%        startTime=t(6);
%        currentTime=t(6);
%
vrep.simxGetIntegerParameter(clientID,vrep.sim_intparam_mouse_x,vrep.simx_opmode_stream
ing); % Initialize streaming
%        while (currentTime-startTime < 60)
%
[returnCode,data]=vrep.simxGetIntegerParameter(clientID,vrep.sim_intparam_mouse_x,vrep.si
mx_opmode_buffer); % Try to retrieve the streamed data
%            if (returnCode==vrep.simx_return_ok) % After initialization of streaming, it will take
a few ms before the first value arrives, so check the return code
%                fprintf('Mouse position x: %d\n',data); % Mouse position x is actualized when the
cursor is over V-REP's window
%            end
%            t=clock;
%            currentTime=t(6);
%        end
 % Before closing the connection to V-REP, make sure that the last command sent out had time
to arrive. You can guarantee this with (for example):
vrep.simxGetPingTime(clientID);
% Now close the connection to V-REP:
vrep.simxFinish(clientID);
%%
else
fprintf('Remote API function call returned with error code: %d\n',res);
    end
vrep.delete(); % call the destructor!
disp('Program ended');
%plot(x(:,1),x(:,2))
```

APPENDIX 12. MATLAB CODE USED TO GENERATE TRAJECTORY

```
function [Y,T]=GenTraj(A,V,P,Tj,Ts)
%GenTraj Trajectory generation for point to point motion with
velocity,
% acceleration, jerk and snap (second time derivative of
acceleration)
% constraints
% Example:[Y,T]=GenTraj(A,V,P,Tj,Ts) returns the position,
velocity
% and acceleration profiles for a snap controlled law from the
specified
% constraints on maximum velocity V, maximum acceleration A,
desired
% travelling distance P, Jerk time Tj and Snap time Ts.
% Y is a 3 row matrix containing the position, velocity and
acceleration
% profile associated to the time vector T.
%
% If Tj and Ts are not given, Tj=Ts=0 is assumed. The resulting
mouvement is
% acceleration limited. If Ts is not given, Ts=0 and P contains
the points
% of the corresponding jerk limited law
%%-----------------------------------------------------------------
------------
if nargin<3
   error('At Least Three Input Arguments are Required.')
end
if nargin==3
   type=0;
   Tj=0;
   Ts=0;
elseif nargin==4
   type=1;
   Ts=0;
elseif nargin==5
   type=2;
end
Te=1e-4; % interpolation time
% Verification of the acceleration and velocity constraints
Ta=V/A; % Acceleration time
Tv=(P-A*Ta^2)/(V); % Constant velocity time
if P<=Ta*V % Triangular velocity profile
    Tv=0;Ta=sqrt(P/A);
end
Tf=2*Ta+Tv+Tj+Ts; % Mouvement time
```

```
 % Elaboration of the limited acceleration profile
T=0:Te:Tf;
t(1)=0;t(2)=Ta;t(3)=Ta+Tv;t(4)=2*Ta+Tv;
s(1)=1;s(2)=-1;s(3)=-1;s(4)=1;
P=zeros(3,length(T));
% Ech=zeros(4);
for k=1:3
    u=zeros(1,k+1);u(1,1)=1;
for i=1:4
    Ech = tf(1, u,'inputdelay',t(i));
    law(i,:)=impulse(s(i)*A*(Ech),T);
end
Y(k,:)=sum(law);
end

if (type==1 || type==2)
% Average Filter for Jerk limitation
a = 1;      % Filter coefficients
b = (1/(Tj/Te))*ones(1,(Tj/Te)); % Filter duration equal to jerk
time
Y(3,:)= filter(b,a,Y(3,:));
Y(2,1:length(T)-1)=diff(Y(3,:),1)/Te;
Y(1,1:length(T)-1)=diff(Y(2,:),1)/Te;
if type==2
% Average Filter for snap limitation
a = 1;      % Filter coefficients
b = (1/(Ts/Te))*ones(1,(Ts/Te)); % Filter duration equal to snap
time
Y(3,:)= filter(b,a,Y(3,:));
Y(2,1:length(T)-1)=diff(Y(3,:),1)/Te;
Y(1,1:length(T)-1)=diff(Y(2,:),1)/Te;
end
end
%%%%%%%%%%%%%
figure;
sp(1)=subplot(3,1,1);plot(T,Y(3,:))
grid
sp(2)=subplot(3,1,2);plot(T,Y(2,:))
grid
sp(3)=subplot(3,1,3);plot(T,Y(1,:))
grid
linkaxes(sp,'x');
ylabel(sp(1),'Position [m]');ylabel(sp(2),'Velocity
[m/s]');ylabel(sp(3),'Acceleration [m/s^2]');xlabel(sp(3),'Time
[s]')
end
```

APPENDIX 13. MATLAB CODE FOR FORCE DATA FITTING

```
%file name: Modeling_Force
clear all
close all
clc
%% Force data obtained from experiments
Array=csvread('Force_data.csv');
Time = Array(:, 1);
Voltage = Array(:, 2);
plot(Time, Voltage)
grid
title('Voltage vs. Time')
Force = 20/1.5*Array(:, 2); % convert voltage to force
Thickness=Time/(max(Time)-min(Time))/1.667 % convert drilling time to skull thickness depth
plot(Time, Force)
grid
xlabel('Time seconds')
ylabel('y-axis')
title('Force vs. Time')
plot(Time, Thickness)
grid
plot(Thickness, Force)
grid
title('Force vs. Thickness')
 cftool
%%
%Coefficients (with 95% confidence bounds):
      x=Thickness(1)
       x=0.3
       fx = p1*x^9 + p2*x^8 + p3*x^7 + p4*x^6 +p5*x^5 + p6*x^4 + p7*x^3 + p8*x^2 + p9*x +
p10

Drilling = iddata(Force,Time,0.1880);
Drilling
get(Drilling)
Drilling.InputName = 'Time  ';
Drilling.OutputName = 'Force ';
Drilling.TimeUnit = 'seconds';
dry.InputUnit = 'Time';
dry.OutputUnit = 'N';
%Now that we have the data set ready, we choose the first 300 data points for model estimation.
ze = Drilling
plot(ze);
ze1 = ze;
%The same data set after it has been detrended:
plot(ze1) %show samples from 200 to 300 of detrended data
```

```
%Now that the dataset has been detrended and there are no obvious outliers, let us first estimate
the impulse response of the system by correlation analysis to get some idea of time constants and
the like:
clf
mi = impulseest(ze); % non-parametric (FIR) model
showConfidence(impulseplot(mi),3); %impulse response with 3 standard
                        %deviations confidence region
%The simplest way to get started on a parametric estimation routine is to build a state-space
model where the model-order is automatically determined, using a prediction error method. Let
us estimate a model using the ssest estimation technique:
m1 = ssest(ze);
%m1 is a continuous-time identified state-space model, represented by an idss object. The
estimation algorithm chooses 3 as the optimal order of the model. To inspect the properties of the
estimated model, just enter the model name at the command window:
m1
get(m1)
[A,B,C,D,K,~,dA,dB,dC,dD,dK] = idssdata(m1)
%Analyzing the Estimated Model
h = bodeplot(m1);
%Right-click on the plot and pick Characteristics->Confidence Region. Or, use the
showConfidence command to view the variance of the response.
showConfidence(h,3) % 3 std (99.7%) confidence region
showConfidence(stepplot(m1,'b',mi,'r',3),3)
%We can also consider the Nyquist plot, and mark uncertainty regions at certain frequencies
with ellipses, corresponding to 3 standard deviations:
Opt = nyquistoptions;
Opt.ShowFullContour = 'off';
showConfidence(nyquistplot(m1,Opt),3)
%The response plots show that the estimated model m1 is quite reliable.
%% Estimating Models with a Prescribed Structure
%%System Identification Toolbox can also be used to obtain a model with a prescribed
structure. For example, a difference equation model with 2 poles, 1 zero and 3 sample delays can
be obtained using the arx function as shown below:
%m2 = arx(ze,[2 2 3]);
m2 = nlarx(ze,[4 4 1]) % na=nb=4 and nk=1
m2
%A continuous time transfer function with 2 poles, one zero and 0.2 second transport delay can
be estimated using the tfest command:
m3 = tfest(ze, 2, 1, 0.2)
%Validating the Estimated Model to Experimental Output
zv = Drilling;   % select an independent data set for validation
zv = detrend(zv);    % preprocess the validation data
set(gcf,'DefaultLegendLocation','best')
compare(zv,m1);      % perform comparison of simulated output
%Comparing Estimated Models
compare(zv,m1,'b',m2,'r',m3,'c');
```

```
%The pole-zero plots for the models can be obtained using iopzplot:
h = iopzplot(m1,'b',m2,'r',m3,'c');
showConfidence(h,3);
%The frequency functions above that are obtained from the models can be compared with one
that is obtained using a non-parametric spectral analysis method (spa):
gs = spa(ze);
%The spa command produces an IDFRD model. The bode function can again be used for a
comparison with the transfer functions of the models obtained.
w = linspace(0.4,pi/m2.Ts,200);
opt = bodeoptions; opt.PhaseMatching = 'on';
bodeplot(m1,'b',m2,'r',m3,'c',gs,'g',w,opt);
legend('m1','m2','m3','gs')
%Also, a Nyquist plot can be analyzed with the uncertainty regions marked at certain
frequencies:
showConfidence(nyquistplot(m1,'b',m2,'r',m3,'c',gs,'g'),3)
```

APPENDIX 14. MATLAB CODE FOR RETRIEVING DEPTH DATA USING KINETIC SENSOR

```
% Depth Sensor
clear all
close all
clc
% vid2 = videoinput('kinect', 2);
vid2 = videoinput('kinect',2,'Depth_640x480');
%Look at the device-specific properties
src2 = getselectedsource(vid2);
src2
src2.SkeletonsToTrack = [1]
src2.TrackingMode = 'Skeleton'
src2
%Access body tracking data as metadata on the depth stream using getdata.
% Get the data on the object.
preview(vid2);
%Start the second video input object (the depth stream).
start(vid2);
[frame2, ts2, metaData2] = getdata(vid2);
% Look at the metadata to see the parameters in the body data.
metaData2.IsSkeletonTracked
a =metaData2(1).JointWorldCoordinates(:,:,1)
closepreview(vid2);
%Look at any individual property by drilling into the metadata.
getselectedsource(vid2)
% View the segmentation data as an image.
imagesc(metaData2(1).JointWorldCoordinates(:,:,1));
% Set the color map to jet to color code the people detected.
colormap(jet);
```

```
src.BacklightCompensation = 'LowLightsPriority';
preview(vid);
closepreview(vid);
vid2 = videoinput('kinect',2,'Depth_640x480');
src = getselectedsource(vid2);
src
start(vid2);
% Get the data on the object.
[frame, ts, metaData] = getdata(vid2);
metaData
metaData.IsSkeletonTracked
%Get the joint locations for the first person in world coordinates using the
JointWorldCoordinates property. Since this is the person in position 1, the index uses 1.
metaData.JointWorldCoordinates(:,:,1)
% View the segmentation data as an image.
imagesc(metaDataDepth.SegmentationData);
% Set the color map to jet to color code the people detected.
colormap(jet);
```