# DESIGN AND CONSTRUCTION OF 9-DOF HYPER-REDUNDANT ROBOTIC ARM

Thesis

Submitted to

The School of Engineering of the

UNIVERSITY OF DAYTON

In Partial Fulfillment of the Requirements for

The Degree of

Master of Science in Electrical Engineering

By

Xingsheng Xu

Dayton, Ohio

December 2013

# DESIGN AND CONSTRUCTION OF 9-DOF HYPER-REDUNDANT ROBOTIC ARM

Name: Xingsheng, Xu

APPROVED BY:

# ABSTRACT

DESIGN AND CONSTRUCTION OF 9-DOF HYPER-REDUNDANT ROBOTIC ARM

Name: Xingsheng, Xu
University of Dayton

Advisor: Dr. Raúl Ordóñez

Hyper-redundant robotics is a branch of advanced robotic technology recognized as a method to improve manipulator performance in complex and unstructured environments. Research in both kinematic and dynamic control of hyper-redundant manipulator plays an import role in high-tech field like modern industry, military and space applications. The kinematic redundancy considered in this thesis means the total degrees of freedom (DOF) of robot is more than the degrees of freedom required for the task to be executed. The redundancy provides infinite solutions to achieve the same position and orientation of the end-effector. Therefore, the efficacy of kinematic algorithm affects the accuracy and stability of both motion control and path tracking. In this thesis, we mainly focus on constructing an application robotic platform based on kinematic modeling of a 9-DOF hyper-redundant manipulator.

We firstly take a brief introduction of the background, related work, significance and objective of this thesis. Then the kinematic model of 9-DOF manipulator is established along with its home position configuration. The next work is divided into two parts: first is the construction of hardware platform, and the second one is to design an application software with user interface (UI). In addition, the result of proposed thesis design is demonstrated in a number of experiments. In the end, conclusion and future work are presented.

*To my beloved parents, professors and friends,*

*Live long and prosper!*

# ACKNOWLEDGEMENTS

Time passes quickly like a white pony's shadow across a crevice. The closer winter approaches, the more sunshine I miss. Thankfully, Lady Luck always shines upon me. Through the ages, I always feel like living in my own colourful and fantastic dream.

When I was a kid, I enjoyed painting imaginary robots or aircrafts with cannons on canvas while other children were drawing beautiful flowers and cute animals. After studying in school, I was addicted to reading science fiction secretly during homework time, lunch time ,even bed time under my mothor's keen "sense of smell". All the way to my college life, things got worse. Sometimes, I liked to peek into the small window of our robotic lab hoping someday I was the one, who was lucky enough, to be chosen to live inside this holy palace.

I do not know what I may appear to the world during my whole life. But to myself, I seem to be like a boy who enjoys playing on the seashore and entertaining himself. In now and then I may lose myself seeking a smoother pebble or a prettier starfish than ordinary until someone shows me the great ocean of truth laying all undiscovered before me. This wizard of my life is Dr. Raúl Ordóñez. Because of him, I never stop chasing my dream; because of him, my dream finally comes true.

There are certainly pains for chasing my dreams as well as balms for all my pain. Thanks my friends and workfellows in University of Dayton, Hariharan Ananthanarayanan, Temesgen Kebede, Abdelbaset Elhangari and Bo Li, to make me feel I am not fighting alone. Also, a special thanks to a girl who brought inner peace to me with all her support and prayer during the time I myself was immersed in darkness. Hence, I give my robotic arm her name “Teresa” to thank this angel. It seems that all these joyfulness in the past were part of our dreams while the prime youth seemed to leave us only fragments of memories.

Words cannot describe my gratitude to my beloved parents for giving me all this colourful and fantastic dream.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# SYMBOLS AND ABBREVIATIONS

| | |
|---|---|
| $A_i$ | *HTM of frame $o_i x_i y_i z_i$ with respect to frame $o_{i-1} x_{i-1} y_{i-1} z_{i-1}$* |
| AC | *alternating current* |
| $a_i$ | *link length* |
| $\alpha_i$ | *link twist* |
| DC | *direct current* |
| DH | *Denavit-Hartenberg* |
| DOF | *degrees of freedom* |
| $d_i$ | *link offset* |
| H | *HTM of the end-effector with respect to base frame* |
| HRR | *hyper-redundant robots* |
| $HTM$ | *homogeneous transformation matrix* |
| $h_i$ | *physical link length of 9-DOF arm* |
| MFC | *microsoft foundation classes* |
| $o_n^0$ | *position matrix of the end-effector with respect to base frame* |
| $O_i$ | *origin of joint i+1* |
| PC | *personal computer* |
| $\theta_i$ | *joint variable* |
| RPY | *roll-pitch-yaw* |
| $R_n^0$ | *rotation matrix of the end-effector with respect to base frame* |
| SDK | *software development kit* |

| | |
|---|---|
| $T_j^i$ | *HTM of frame $o_jx_jy_jz_j$ with respect to $o_ix_iy_iz_i$ frame* |
| UART | *universal asynchronous receiver/transmitter* |
| UI | *user interface* |
| WMRA | *wheelchair-mounted robotic arm* |
| $Z_i$ | *actuating shaft of joint i+1* |

# CHAPTER 1

# INTRODUCTION

With the development of science and technology, robots have increasingly penetrated into military, aerospace, medical science and other fields, mostly performing the risky or precise work for human beings. According to [4], if the total degrees of freedom (DOF) of a robot is more than the degrees of freedom required for the task to be executed, the robot is termed as “kinematically redundant” or simply “redundant”. Hyper-redundant robots (HRR) are a kind of robots that have a very large degree of kinematic redundancy. Redundancy in robotic design has been used to improve robotic performance in complex and multiple barrier environments. HRR, analogous in morphology and operation to snakes [5], elephant trunks [6], or tentacles [7], are used to deal with a number of important applications where such robots would be advantageous. Once the number of DOF of the robot exceeds the number of task coordinates, it becomes a great challenge to solve the inverse kinematics and generate a path to make its motion robust and dexterous. Therefore, the research on HRR, has become one of the hot topics of robotic research for its practical significance and theoretical value. In modern days, robot manufacturers provide variety of industrial robots with their own forward and inverse kinematics and control methods (e.g.,PID control). However, these kinematics and control approaches are regarded as black boxes to the researchers who are using them. In order to further independently research and study HRR, building an open HRR is imperatively the main task of this thesis.

The objective of this thesis is to build a hyper-redundant 9-DOF robotic arm with its forward kinematic model and control software with user interface (UI), also implement trajectory tracking of the end-effector. The open robotic arm is constructed using nine orthogonal DC servo motor joints with detachable physical links. The RS-485 protocol is used for communicating from PC direct to the arm. In addition, we develop our own library of Visual C++ functions to control the manipulator's motion and path operation. Its main job is to satisfy different control and motion requests and experiments, e.g., teaching positions, tracking trajectory or testing inverse kinematics in the future.

The thesis is organized as follows:

Chapter 2 discusses the background of thesis and related work.

Chapter 3 introduces the forward kinematic modeling and operation of 9-DOF robotic arm.

The construction of 9-DOF hardware platform with its mechanical analysis and the development of control software with UI are explained in Chapter 4 and Chapter 5 respectively.

Chapter 6 shows all the experiment results and demonstrations of 9-DOF arm.

Finally, the thesis experience, difficulties, deficiencies and future improvement are summarized in Chapter 7 as a conclusion.

# CHAPTER 2

# BACKGROUND AND RELATED WORK

In this chapter, all the fundamental background, relevant knowledge and previous work corresponding to this thesis are discussed. The 9-DOF manipulator in this thesis is constituted of orthogonal DC servo motors which will be introduced in Section 2.1. Section 2.2 describes a typical protocol that allows the PC to communicate with the servo motors directly. Fundamental background theory on robot kinematics and control that is used throughout the thesis termed forward kinematics will be explained in Section 2.3. As explained in Chapter 1, HRR are used in wide range of applications and experiments. Section 2.4 presents a brief literature review on previous work on HRR.

## 2.1 DC SERVO MOTOR

A DC servo motor is a rotary actuator, powered from direct current, that converts the received electrical signal into angular displacement or velocity of motor output. It contains a suitable motor embedded with sensors for position and velocity feedback, which allows for accurate closed-loop control of angular position. Nowadays, servo motors are generally used as a high performance alternative to the stepper motor both in industrial manufacturing and institutional research. As there are abundant choices of servo motors in the market, motor selection becomes significant for building

Figure 2.1: DC servo motor products called “Dynamixels” of *ROBOTIS* (source: [1]).

specific robotic arm. The foremost step is to check the required specifications such as positioning accuracy, torque requirements, speed range, operating voltage and other environmental resistances for the manipulator. The final determination of motors should also meet required specifications of different arm parts (i.e., wrist, elbow and shoulder). Figure 2.1 presents a series of DC servo motors (called Dynamiexls) from *ROBOTIS*, which will be used to construct the arm in this thesis.

## 2.2 RS-485 PROTOCOL

RS-485 described in [8] is a standard electrical characteristic protocol used for computers and devices. RS-485 is widely used for the configuration of inexpensive local networks and multidrop communications links as it is fast, efficient and robust [9]. In this thesis, the signal of main controller universal asynchronous receiver/transmitter (UART) should be converted into RS-485 type signal to control “MX series” servo motors [1] with a personally made main controller. According to [3], RS-485 instruction packet used here is command data that is sent from the PC to the servo motors. The structure of instruction packet is as in Figure 2.2, where the first two addresses (i.e., “*OXFF OXFF*”) notify the beginning of the packet. The instruction packet contains ID of motor which will be commanded, length of the packet[1], different

[1]It equals the number of parameters plus two.

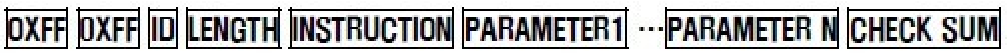


Figure 2.2: Structure of instruction packet by RS-485 protocol.

types of instructions and parameters that are used when instruction requires ancillary data. At last, the function of "*CHECK SUM*" address is to check if packet is damaged during communication. In addition, servo motor receives instruction packet to perform a command and returns the result as status packet to the PC.

## 2.3 FORWARD KINEMATICS

Forward kinematics introduced in [2, 10, 11] is an arithmetic operation, which determines the position and orientation of the end-effector given the values of each joint parameter of the robot using the kinematic equations. A robot manipulator is composed of a set of links connected together by joints. Also, a manipulator with $n$ joints will have $n + 1$ links, since each joint connects two links.

With the $i^{th}$ joint, we define a joint variable, denoted by $q_i$. In the case of a revolute joint, $\theta_i$ is the angle of rotation, and as to a prismatic joint, $d_i$ is the joint displacement:

$$q_i = \begin{cases} \theta_i & \text{if joint } i \text{ is revolute,} \\ \\ d_i & \text{if joint } i \text{ is prismatic.} \end{cases} \tag{2.1}$$

To proceed with the kinematic analysis, we associate a corresponding coordinate frame rigidly to each link. Furthermore, when joint $i$ is actuated, link $i$ and its attached frame, $o_i x_i y_i z_i$, experiences the resulting motion. The frame $o_0 x_0 y_0 z_0$, which is attached to the robot base, is labeled as the base frame. Figure 2.3 illustrates an example of attaching frames rigidly to links in the case of an elbow manipulator. We suppose that $A_i$ is the Homogeneous Transformation Matrix (HTM) that represents the position and orientation of $o_i x_i y_i z_i$ with respect to $o_{i-1} x_{i-1} y_{i-1} z_{i-1}$. Now it can be inferred the HTM that expresses the position and orientation of $o_j x_j y_j z_j$ with respect to $o_i x_i y_i z_i$ is termed, by convention, a Transformation Matrix, and is denoted by $T_j^i$. Moreover, $T_j^i$ is the product of each HTM post multiplied together as

$$T_j^i = \begin{cases} A_{i+1} A_{i+2} \cdots A_{j-1} A_j & \text{if } i < j, \\ I & \text{if } i = j, \\ (T_j^i)^{-1} & \text{if } j > i. \end{cases} \tag{2.2}$$

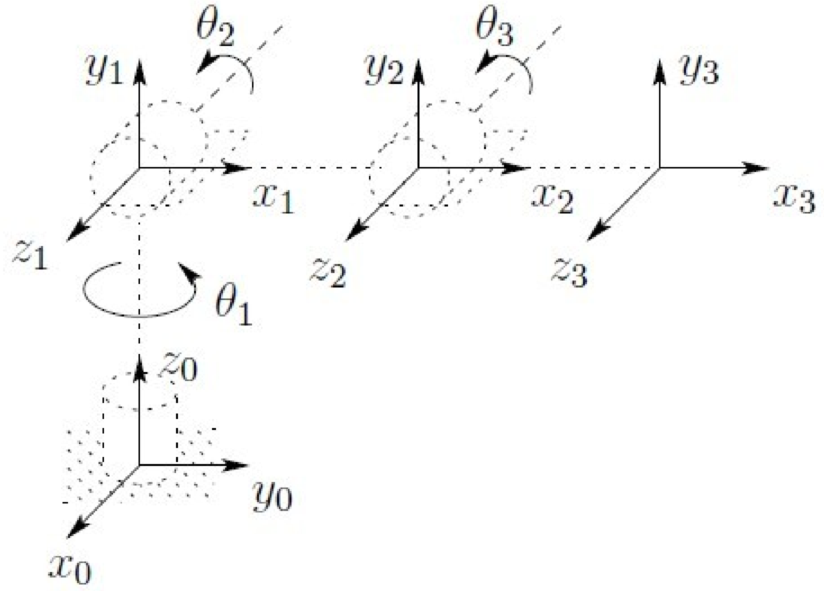


Figure 2.3: Coordinate frames attached to elbow manipulator (source: [2]).

Denoting the position and orientation of the end-effector with respect to the base frame by a three-vector $o_n^0$ which gives the spatial coordinates of the position and a $3\times 3$ rotation matrix $R_n^0$, and define the homogeneous transformation matrix respectively as

$$H = \begin{bmatrix} R_n^0 & o_n^0 \\ 0 & 1 \end{bmatrix}. \tag{2.3}$$

Based on (2.1), (2.2) and (2.3), the position and orientation of the end-effector with respect to the base frame can be given as

$$H = T_n^0 = A_1(q_1)\cdots A_n(q_n).$$

Hence, we rewrite (2.3) as

$$T_j^i = A_{i+1}\cdots A_j = \begin{bmatrix} R_j^i & o_j^i \\ 0 & 1 \end{bmatrix}. \tag{2.4}$$

A widely used conventional method [12] for attaching reference frames to the links of a spatial kinematic chain is the Denavit-Hartenberg (DH), or DH convention:

$$
\begin{aligned}
A_i &= Rot_{z,\theta_i} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,\alpha_i} \\
&= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\times
\begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} c_{\alpha_i} & s_{\theta_i} s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i} c_{\alpha_i} & -c_{\theta_i} s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix},
\end{aligned}
\tag{2.5}
$$

where $c_i$ means $\cos(\theta_i)$ and $s_i$ means $\sin(\theta_i)$. Equation (2.5) indicates that each homogeneous transformation $A_i$ is characterized as a product of four basic transformations. The four parameters $a_i$, $\alpha_i$, $d_i$, and $\theta_i$ in (2.5) are generally called link length, link twist, link offset and joint angle.

Finally, the combination of (2.4) and (2.5) gives us the position and orientation of the tool frame expressed in base coordinates. The last column of resultant matrix $T$ (i.e., $o_n^0$) describes the Cartesian coordinates of the origin of the tool frame. The upper left $3 \times 3$ matrix of matrix $T$ (i.e., $R_n^0$) presents the orientation of the tool frame with respect to the base frame. Those nine elements can in turn be represented using only three independent quantities: Euler-angles, Roll-Pitch-Yaw (RPY) angles or angle representation yielding a total end result of six quantities (i.e., three for positioning and three for orientation description).

## 2.4 LITERATURE REVIEW

As explained in the previous chapter, redundant robot manipulators are widely used in various applications. Redundancy introduced to robot manipulators gives high kinematic flexibility which enables the manipulator to achieve the same attitude though infinite number of configurations, and this preponderant property is the main motivation behind this thesis. Although, HRR have been investigated for more than 25 years, they have still remained a laboratory curiousity. Previous kinematic modeling techniques have not been particularly efficient or robust to the needs of HRR task. Also, the mechanical design and implementation of HRR has been considered as unnecessarily complex. This section will briefly review the previous work on kinematic modeling and control of redundant robot manipulators.

Several recent techniques [13–16] of redundant robot are reported in the literature. These techniques can be sorted into two main categories. The first one contains different methods that solve and optimize the forward or inverse kinematic modeling for HRR. In the early stages, [13] researchers figured out a complete and general solution dealing with the forward kinematics problem of platform-type robotic manipulators. It showed that the equations for the forward kinematics of redundant robot were highly nonlinear, however, closed-form solutions to the forward rate and acceleration kinematics could be obtained by solving a system of linear equations based on the data of three spatial positions, velocities and accelerations of the end-effector.

After solving forward kinematics, inverse kinematics, which solves each joint variable by given position and orientation of end-effector, became a point of hot debate in the robotic field. Recently, a valid analytical inverse kinematics computation for redundant manipulators was proposed by [14]. The method used a virtual joint instead of the joint offset from the wrist, and attached the virtual manipulator with a spherical wrist which means the inverse kinematics solution of real manipulator could be obtained by solving the inverse kinematics problems of virtual manipulator with less complexity and nonlinearity.

The second category consists of techniques that deal with the motion and path control of hype-redundant robot with different dynamic control approaches. Typically, [15] developed a dynamic control procedure based on analysis in the defined posture space where three parameters were used to determine the manipulator posture. The manipulator dynamics was modeled by the parameters of the space path and the path-tracking feedforward controller was formulated on the basis of the spatial dynamic equations. As a result, it was proven that the workspace path of hyper-redundant manipulator could be tracked precisely.

Other emerging technology on dynamic control laws for redundant manipulators includes the work of [16], a wheelchair-mounted robotic arm (WMRA) system, based on the dynamic model of a manipulator in Cartesian space, was designed and built to meet the needs of disabled person. An optimized control method that guarantees asymptotic tracking of a desired end-effector trajectory was adopted.

# CHAPTER 3

# 9-DOF ROBOTIC ARM MODELING AND OPERATION

A kinematic model of 9-DOF manipulator is designed in this chapter that explains the geometric motions without considering forces and torques. The main part of kinematic modeling is dealing with the assignment of coordinate frames to represent the position and orientation of the end-effector with respect to the base, and with transformations among the coordinate systems. Also, forward kinematic analysis is carried out in Section 3.2 according to DH convention to get both position and orientation of the end-effector. Section 3.3 states the configuration of the home position of 9-DOF manipulator defined for kinematic control.

## 3.1 KINEMATIC MODEL OF 9-DOF ARM

The proposed 9-DOF manipulator is composed of a set of revolute joints which are arranged orthogonal to each other while making the shoulder and wrist joint spherical. Thus, this kinematic model can be easily decoupled into inverse kinematics problem into two simpler problems, i.e., inverse position kinematics, and inverse orientation kinematics for future research.

### 3.1.1 Labeling of Links and Joints

In order to establish the kinematic model, all the links and joints of the manipulator are labeled starting from the bottom as shown in Figure 3.1 where $\theta_i$ is joint variable of each motor and $h_i$ is the length of each link. This particular configuration is based on the home position defined in Section 3.3. The arm with nine joints (from label 1 to 9) has ten links (from label 0 to 9) since one joint connects two links. Therefore, frame $(OXYZ)_i$ can be attached to link $i$, i.e., each point on link $i$ is constant when expressed in the $i^{th}$ coordinate frame.

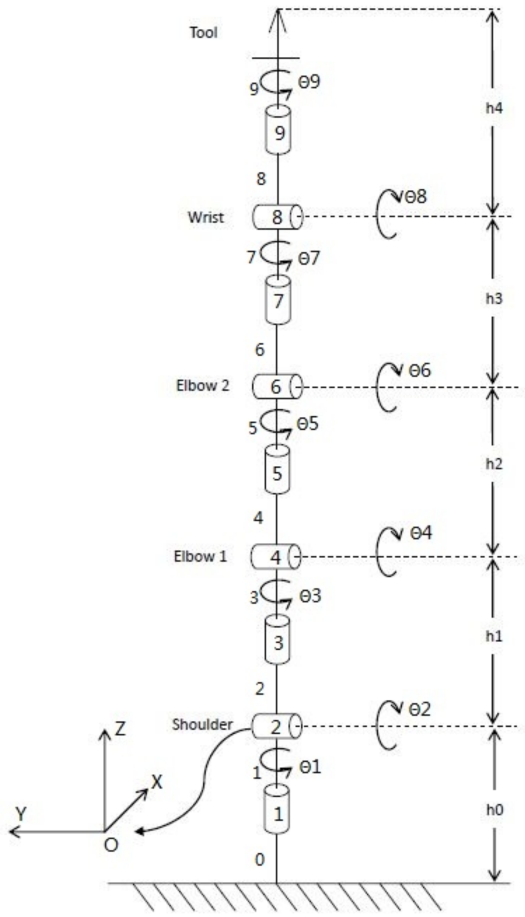


Figure 3.1: Joint schematic of 9-DOF manipulator.

### 3.1.2 Assignment of Coordinate Frames

Figure 3.2 presents frame assignment of 9-DOF manipulator where $\theta_i$ is joint variable of each motor and $h_i$ measures the physical lenth of each links. First, we assign each $Z_i$ axis along the motor shaft of joint $i+1$ (i.e., $Z_1$, $Z_2$, $\cdots$, $Z_9$). Next we establish the base frame $(OXYZ)_0$. Technically, the origin $O_i$ of each joint can be chose as any point along $Z_i$. In this thesis, we make the origin $O_0$ and $O_2$ to coincide at center of the motor shaft of joint 2 (i.e., $O_1$) to form a spherical shoulder as $h_0$ stays constant while the joints are actuating which makes DH parameter much simpler. Therefore, the origin $O_4$, $O_6$, $O_8$ also can be fixed using this method to form elbows and wrist. Then we choose $X_0$ and $Y_0$ to establish a right-handed frame. Since the axes $Z_1$ and $Z_2$ intersect which means they are co-planar, we choose $X_1$ be vertical axis of the $(Z_1\ ,Z_2)$ plane. Thus, in the same way, other $X_i$ can be assigned. Meanwhile, $Y_i$ is established to complete a right-handed frame.

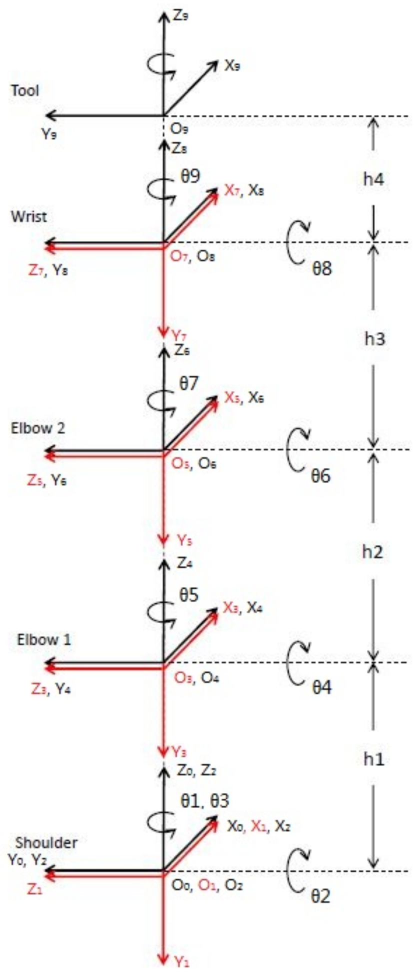


Figure 3.2: Frame assignment of 9-DOF manipulator.

### 3.1.3 Establishment of the End-Effector Frame

Finally, the tool frame $(OXYZ)_9$ is established to represent the end-effector. The unit vectors along the $X_9$, $Y_9$ and $Z_9$ are labeled as $n$[1], $s$[2] and $a$[3]. This frame is usually fixed at the end of the last joint.

[1]The direction normal to the plane formed by $a$ and $s$.
[2]The sliding direction.
[3]The approach direction.

## 3.2 FORWARD KINEMATIC ANALYSIS

The objective of forward kinematic analysis is to measure the cumulative effect of the entire set of joint variables on the end-effector, that is, to calculate the position and orientation of the end-effector for a given set of joint angles.

### 3.2.1 Table of Denavit-Hartenberg (DH) Parameters

According to Section 2.3, DH parameters are assigned in Table 3.1 based on proposed manipulator where $a_i$ is the length of common normal between $Z_{i-1}$ and $Z_i$ along $X_i$, $\alpha_i$ represents the angle from $Z_{i-1}$ and $Z_i$ measured about $X_i$, $d_i$ is the distance between $O_{i-1}$ and $O_i$ along $Z_{i-1}$ axis, and $\theta_i$ (i.e., the joint variable) is the angle from $X_{i-1}$ and $X_i$ measured about $Z_{i-1}$. The HTM $A_i$ between each frame can be determined by substituting the parameters of Table 3.1 into Equation (2.5).

Table 3.1: Denavit-Hartenberg parameters for 9-DOF manipulator where $\theta^*$ is joint angle

| $Frame$ | $a_i$ $(mm)$ | $\alpha_i$ $(rad)$ | $d_i$ $(mm)$ | $\theta_i$ $(degree)$ |
|---|---|---|---|---|
| 1 | 0 | $-\pi/2$ | 0 | $\theta_1^*$ |
| 2 | 0 | $\pi/2$ | 0 | $\theta_2^*$ |
| 3 | 0 | $-\pi/2$ | 156.55 | $\theta_3^*$ |
| 4 | 0 | $\pi/2$ | 0 | $\theta_4^*$ |
| 5 | 0 | $-\pi/2$ | 152.15 | $\theta_5^*$ |
| 6 | 0 | $\pi/2$ | 0 | $\theta_6^*$ |
| 7 | 0 | $-\pi/2$ | 119.85 | $\theta_7^*$ |
| 8 | 0 | $\pi/2$ | 0 | $\theta_8^*$ |
| 9 | 0 | 0 | 73.9 | $\theta_9^*$ |

### 3.2.2 Formation of Transfer Matrix

It is straightforward to compute the individual matrices $A_i$ from Equation (2.5) as

$$A_1 = \begin{bmatrix} c_1 & 0 & -s_1 & 0 \\ s_1 & 0 & c_1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad A_2 = \begin{bmatrix} c_2 & 0 & s_2 & 0 \\ s_2 & 0 & -c_2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$A_3 = \begin{bmatrix} c_3 & 0 & -s_3 & 0 \\ s_3 & 0 & c_3 & 0 \\ 0 & -1 & 0 & h_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad A_4 = \begin{bmatrix} c_4 & 0 & s_4 & 0 \\ s_4 & 0 & -c_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$A_5 = \begin{bmatrix} c_5 & 0 & -s_5 & 0 \\ s_5 & 0 & c_5 & 0 \\ 0 & -1 & 0 & h_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad A_6 = \begin{bmatrix} c_6 & 0 & s_6 & 0 \\ s_6 & 0 & -c_6 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$A_7 = \begin{bmatrix} c_7 & 0 & -s_7 & 0 \\ s_7 & 0 & c_7 & 0 \\ 0 & -1 & 0 & h_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad A_8 = \begin{bmatrix} c_8 & 0 & s_8 & 0 \\ s_8 & 0 & -c_8 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$A_9 = \begin{bmatrix} c_9 & -s_9 & 0 & 0 \\ s_9 & c_9 & 0 & 0 \\ 0 & 0 & 1 & h_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.1}$$

where $c_i$ means $\cos(\theta_i)$ and $s_i$ means $\sin(\theta_i)$. According to Equation (2.4) and Equation (3.1), $T_9^0$ is then given as

$$\begin{aligned} T_9^0 &= A_1 \cdots A_9 \\ &= \begin{bmatrix} n_x & s_x & a_x & o_x \\ n_y & s_y & a_y & o_y \\ n_z & s_z & a_z & o_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \end{aligned} \tag{3.2}$$

which gives the position and orientation of the end-effector with respect to the base coordinate frame. The last column of matrix $T_9^0$ (i.e., $o_x, o_y, o_z$) gives the Cartesian coordinates of the origin of the end-effector $O_9$. The upper left $3 \times 3$ matrix of matrix $T_9^0$ presents the orientation of the end-effector with respect to the base frame. This $3 \times 3$ Euler-angles matrix can be algorithmically transferred into Roll-Pitch-Yaw angles as

$$\begin{cases} R_z = tan^{-1}(\frac{n_y}{n_x}) \\ R_y = tan^{-1}(\frac{-n_z}{n_x cos(R_z) + n_y sin(R_z)}) \\ R_x = tan^{-1}(\frac{a_x sin(R_z) - a_y cos(R_z)}{o_y cos(R_z) - o_x sin(R_z)}) \end{cases}, \tag{3.3}$$

where $R_x$, $R_y$ and $R_z$ denote the parameters of RPY angles.

## 3.3 HOME POSITION CONFIGURATION

In order to determine the motion of the end-effector, home position of the manipulator must be defined. In this thesis, the proposed home position is to make the whole arm straight up at a singularity position (i.e., there will be infinity many solutions to inverse kinematics for this end-effector position and orientation) as Figure 3.1. Moreover, all the joint variables are made zero at home position. Thus, we substitute $\theta_i = 0$ $(i = 1, 2, \cdots, 9)$ into Equation (2.4) and get the transfer matrix $T_9^0$ at home position as $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 502.79 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ where $502.79mm$ is exactly the entire length of the manipulator.

# CHAPTER 4

# THE CONSTRUCTION OF HARDWARE PLATFORM

We build an open prototype of the 9-DOF kinematic model developed in Chapter 3 to study its robotic redundancy and kinematic control. This hyper-redundant robotic arm constructed with nine DC servo motors is attached to a fixed base frame. The motors, are designed orthogonal to each other based on the model in Chapter 3. The arm consists of four main joints, one shoulder, two elbows and one wrist. Each motor carries a micro-controller with a RS-485 interface. Thus, the RS-485 protocol is used for communication from PC direct to the arm. Also, a specific AC-DC power adapter is chosen to satisfy the maximum recommended working voltage and current for servo motors. Section 4.1 will first detail the configuration of hardware platform. Section 4.2 represents the structure of servo motor connection with PC using RS-485 protocol. Finally, Section 4.3 will provide a mechanical analysis of the 9-DOF manipulator.

## 4.1 FUNDAMENTAL STRUCTURE OF HARDWARE PLATFORM

Based on the kinematic model of the robotic arm, a series of high-performance servo motors corresponding to different needs of joints in this thesis. Figure 4.1 demonstrates the fundamental configuration of the arm at its home position introduced in Section 3.3. The first motor (i.e., motor 1) is fixed as the base with its axes along the straight-up $Z_0$ axis in such a way that the axis of rotation is orthogonal to previous. Other motors are assembled successively upon the base along the $Z_0$ axis. The kinematic motion is transmitted within the physical links between these motors.

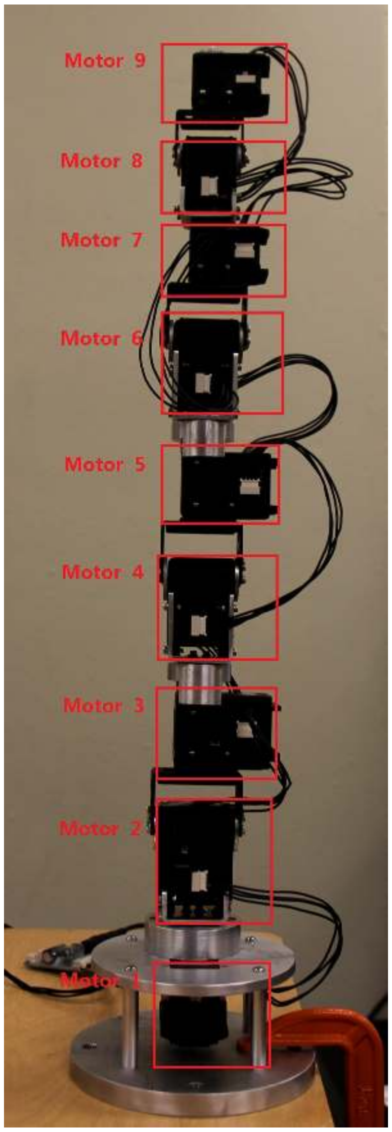


Figure 4.1: Structure of 9-DOF redundant manipulator at its home position.

### 4.1.1 Motor Features

In this thesis, the proposed motors we have chosen so as to satisfy both kinematic and dynamic control requirement, e.g., torque specification, speed range, etc. As mentioned in Section 2.1, a “MX series” high-performing “Dynamixels” motors as shown in Figure 2.1 from *ROBOTIS* are adopted. These servo motors embedded with sensors can feedback position, speed, load, temperature, etc, in real time. Meanwhile, the micro-controller inside each motor allows user to achieve PID control, upgrade its firmware version and update the goal position, speed, torque and acceleration. Table 4.1 gives the specification of different type of “MX series” motors. From the specification, we learn that “MX series” motor provides strong dynamic power with light weight which meets the requirements of the arm design. According to the stall Torque, we assign $MX-28R$, $MX-64R$ and $MX-106R$ to form the wrist, elbow and shoulder respectively. Moreover, these motors have very small holding current.

Table 4.1: Specification of “MX series” servo motors

| $Model$ | $Weight$ $(g)$ | $Dimension$ $(mm \times mm \times mm)$ | $Stall\,Torque$ $(N \cdot m)$ | $No\,load\,speed$ $(rpm)$ | $Voltage$ $(V)$ |
|---|---|---|---|---|---|
| $MX-28R$ | 72 | $35.6 \times 50.6 \times 35.5$ | 3.1 | 67 | 14.8 |
| $MX-64R$ | 126 | $40.2 \times 61.1 \times 41$ | 7.3 | 78 | 14.8 |
| $MX-106R$ | 153 | $40.2 \times 65.1 \times 46$ | 10.0 | 55 | 14.8 |

### 4.1.2 Home Position Calibration

As illustrated in Section 3.3, we require to achieve a perfect physical home position for the manipulator platform as the initial position. Therefore, a specific calibration tool as shown in Figure 4.2 is used to make the arm close to its theoretical home position which means all the angles of motors are set close to their center position. This is in order to provide maximum range of motion on either side of the home position.

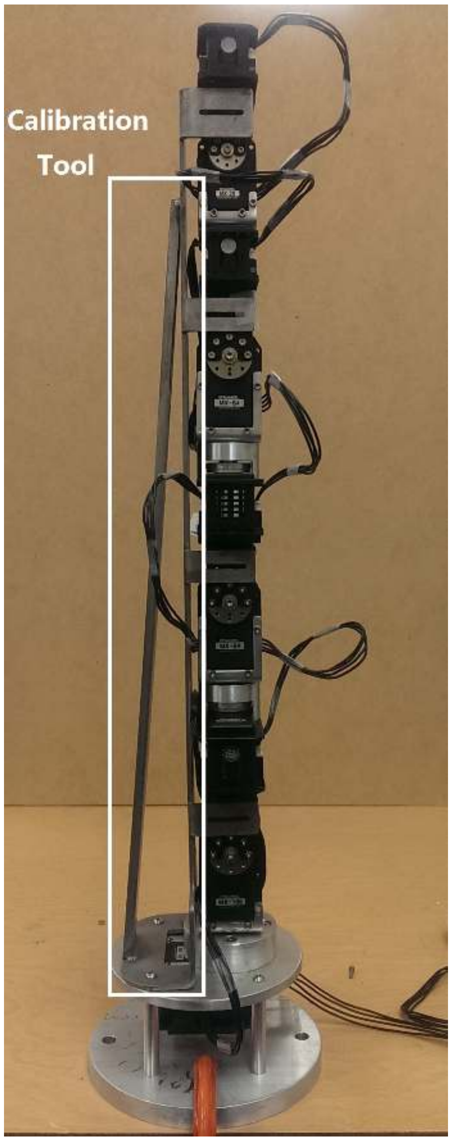


Figure 4.2: 9-DOF redundant manipulator installed with the calibration tool as shown in white box.

## 4.2 BASIC STRUCTURE OF MOTOR CONNECTION ON A BUS

According to [3], all the “MX series” motors we used are connected though RS-485 communication BUS. Figure 4.3 shows the pin assignment of a connector. The “MX series” motor has two 4 pin connectors on it and those two connectors have the same function. As shown in Figure 4.4, all the servo motors constructed the arm can be controlled on a BUS.

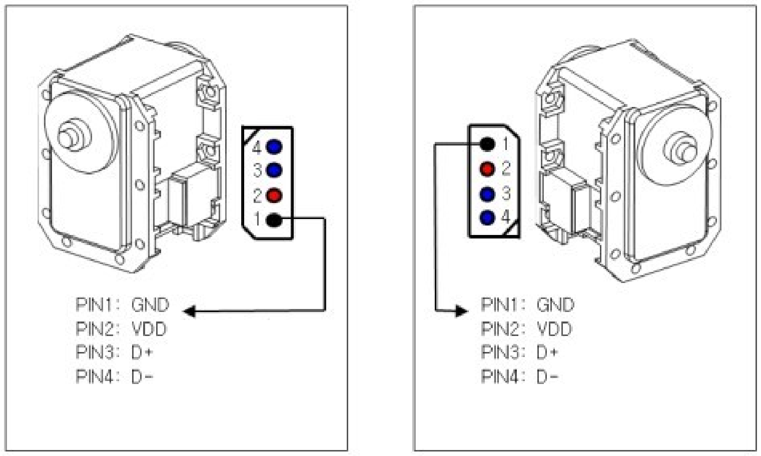


Figure 4.3: Pin assignment of “MX series” servo motors with RS-485 protocol (source: [3]).

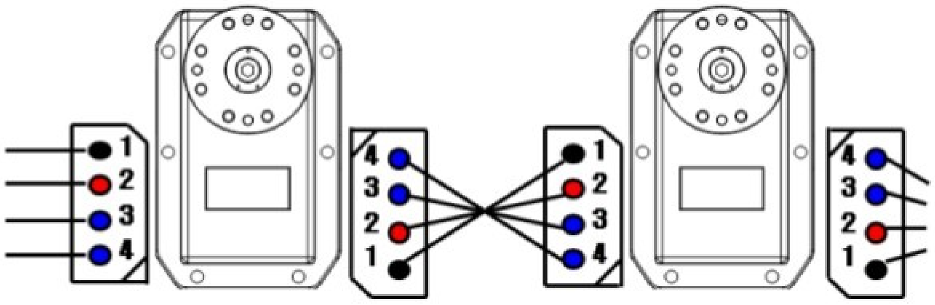


Figure 4.4: Pin-to-pin writing method of “MX series” servo motors with RS-485 protocol (source: [3]).

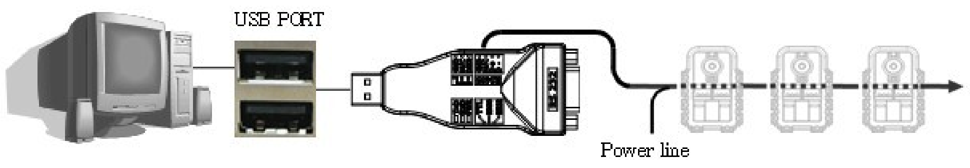


Figure 4.5: Connection of "MX series" servo motors on BUS with power source using "USB2Dynamixel" (source: [3]).

The objective of this thesis is using PC to command the manipulator composed of "MX series" motors directly on the RS-485 BUS. Therefore, "USB2Dynamixel", as shown in Figure 4.6, is used to operate motors directly from PC. Table 4.2 introduces the function of each part of "USB2Dynamixel". The 4 pin connector is used for this thesis particularly to link "MX series" motors though RS-485. Figure 4.5 describes the structure of a series of servo motors connected with PC though "USB2Dynamixel".

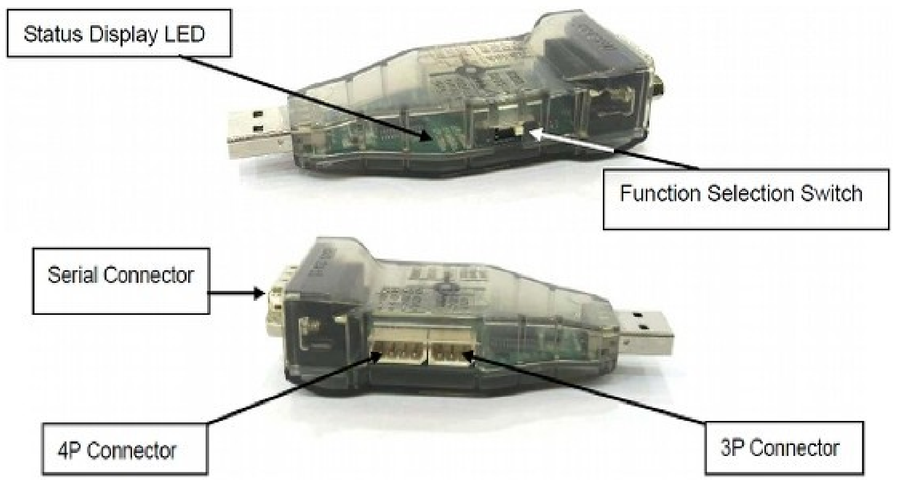


Figure 4.6: Instruction of the device used to operate "MX series" servo motors directly from PC, i.e., "USB2Dynamixel" (source: [3]).

Table 4.2: Instruction of “USB2Dynamixel” for the usage of each part

| **Name** | **Description** |
|---|---|
| Status Display LED | *Display power supply and data writing status.* |
| Function Selection Switch | *Communication method Selection.* |
| 3P Connector | *Connect “AX Series” through TTL communication.* |
| 4P Connector | *Connect “ MX, DX, RX Series” through RS-485 protocol.* |
| Serial Connector | *Change from USB port to Serial port.* |

## 4.3 MECHANICAL ANALYSIS

The 9-DOF manipulator constructed by “MX series” servo motors provides high-performance in dynamic motion and mechanical strength as shown in Table 4.3. It also has capability to afford a high torque requirement with payload.

Table 4.3: Mechanical specification of 9-DOF manipulator

| **Mechanical Table** | |
|---|---|
| Configuration | $9DOF$ |
| Total weight | $1.2kg$ |
| Payload at full reach | $792g$ |
| Payload at mid reach | $924g$ |
| Arm length | $64.6cm$ |
| Arm reach | $50.2cm$ |
| Maximum joint speed | $55rpm$ |
| Repeatability | $\pm 0.5mm$ |
| Input voltage | $14.8VDC$ |
| Ambient temperature | from $20^{\circ}C$ to $35^{\circ}C$ |

Table 4.4 refers to the limitation of each joint angle to avoid the motor to hit each other or the link. All the range of roll-axis angles stay the same (i.e., $-180^{o} \backsim +180^{o}$), and all the range of pitch-axis are the same (i.e., $-110^{o} \backsim +110^{o}$). However, since all the angles of motors are set close to their center position during calibration, there will be a few sliding difference from designed starting and terminal position while all the range stay the same.

Table 4.4: Axis range of each joint variable of 9-DOF manipulator

| **Axis Range** | |
|---|---|
| Shoulder Roll $(\theta_1, \theta_3)$ | $-180^o \backsim +180^o$ |
| Shoulder Pitch $(\theta_2)$ | $-110^o \backsim +110^o$ |
| Elbow Roll $(\theta_5, \theta_7)$ | $-180^o \backsim +180^o$ |
| Elbow Pitch $(\theta_4, \theta_6)$ | $-110^o \backsim +110^o$ |
| Wrist Roll $(\theta_9)$ | $-180^o \backsim +180^o$ |
| Wrist Pitch $(\theta_8)$ | $-110^o \backsim +110^o$ |

# CHAPTER 5

# 9-DOF ROBOTIC ARM CONTROL UI

An application software platform is designed in this thesis to provide motion control for the 9-DOF manipulator. We develop a function library which can be used for constructing various applications based on the different requirements. In addition, an user interface is designed for convenient and intuitive control of manipulator.

## 5.1 FUNCTION LIBRARY FOR APPLICATION

An original static function library, named “DynamixelFuncsLib”, is developed based on a standard programming library, called “Dynamixel SDK”, provided by *ROBOTIS*. The “MX series” motor obeys and returns the instructions by pulse signal. [3] claims that “Dynamixel SDK” is explained based on C language calling method. It offers five groups of calling methods, i.e., *Device Control Method, Set/Get Packet Method, Packet Communication Method, High Communication Method* and *Utility Method*, to send or receive pulses to or from the motor. Our function library is designed upon this specific architecture. Table 5.1 includes all the original library functions within their respective classes. It is noteworthy that since all the angles of motors are set close to their center position during calibration, the position pulse may increase beyond the maximum pulse (i.e, 4095). Thus, the function *PositionToPulses( )* automatically subtract 4095 pulses when it reaches beyond the maximum pulse to fix this problem. Users should use the “DynamixelFuncsLib” as C++ reference library to create their application projects by calling functions from library.

Table 5.1: Function list of original class library

| **Class** | **Function** | **Description** |
|---|---|---|
| *CommStatus* | *PrintCommStatus( )* | print communication result |
| *ErrorCode* | *PrintErrorCode( )* | print error bit of status packet |
| *MotorMotion* | *SetSpeed( )* | set speed to motor |
| | *GetSpeed( )* | get speed from motor |
| | *SetGoalPosition( )* | set goal position to motor |
| | *GetPresPosition( )* | get present position from motor |
| | *SyncWrite( )* | send syncwrite packets |
| | *GetPresLoad( )* | get present load from motor |
| *Transform* | *PulsesToSpeed( )* | transform pulses to speed |
| | *SpeedToPulses( )* | transform speed to pulses |
| | *PulsesToPosition( )* | transform pulses to position |
| | *PositionToPulses( )* | transform position to pulses |
| | *CalculateSpeed( )* | calculate speed |
| | *DegreeToRadian( )* | transform degree to radian |
| | *RadianToDegree( )* | transform radian to degree |
| | *ForwardKinematics( )* | use forward kinematics to find the end effector |

## 5.2 APPLICATION SOFTWARE PLATFORM

This section focuses on the main functions and structure of application software platform with its UI. The objective is to achieve initialization of packet communication between PC and manipulator, monitoring motor status, configuration of home position and sleep position, manual and sync control, motion teach and play, etc. This interactive software allows communication between PC and each servo motors go through in real time. It is capable to send instruction signal packet to command motors and receive status signal packet from them to get feedback. Also, all the kinematic and dynamic algorithm of this thesis design can be calculated and achieved by this software.

### 5.2.1 Function and Structure of Application Software

All the functional modules of software platform are built upon the original static library as shown in Table 5.1. Table 5.2 lists the main functional classes developed for the application software.

Table 5.2: Instruction of main classes in application software system

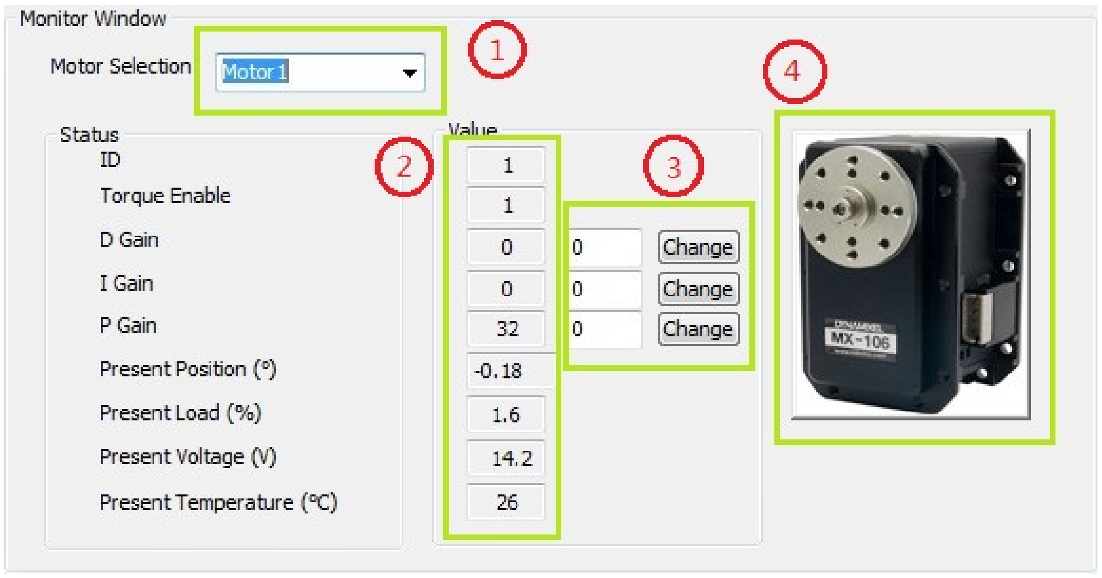
| **Class** | **Description** |
|---|---|
| *CAoutdlg* | create a software statement diagram. |
| *CUINineDOFArmDlg* | create windows of user interface. |
| *Cmonitor* | monitor motor status. |
| *Cmanual* | include manual control functions. |
| *Csync* | include sync control functions. |
| *Cforwardkinematics* | include forward kinematics functions. |

### 5.2.2 UI Design Using C++ Microsoft Foundation Classes (MFC)

The UI developed for this software platform is based on the C++ MFC. It is divided into three parts in general, that are, operation monitor, manual control and sync control as shown in Figure 5.1-5.3. These three interfaces can be switched as bookmarks using the "Tab Control" of C++ MFC which allows user to set parameters and observe motor status simultaneously. This subsection will introduce the function and application method of UI in detail for each part.

#### Operation Monitor Interface

The operation monitor interface, included major status of motors, is shown as Figure 5.1 where *label 1* is a motor selection drop-down list, *label 2* is a monitor window of motor status, *label 3* is a PID tuning interface and *label 4* is a display of selected motor appearance.


Monitor Window

Motor Selection: Motor 1

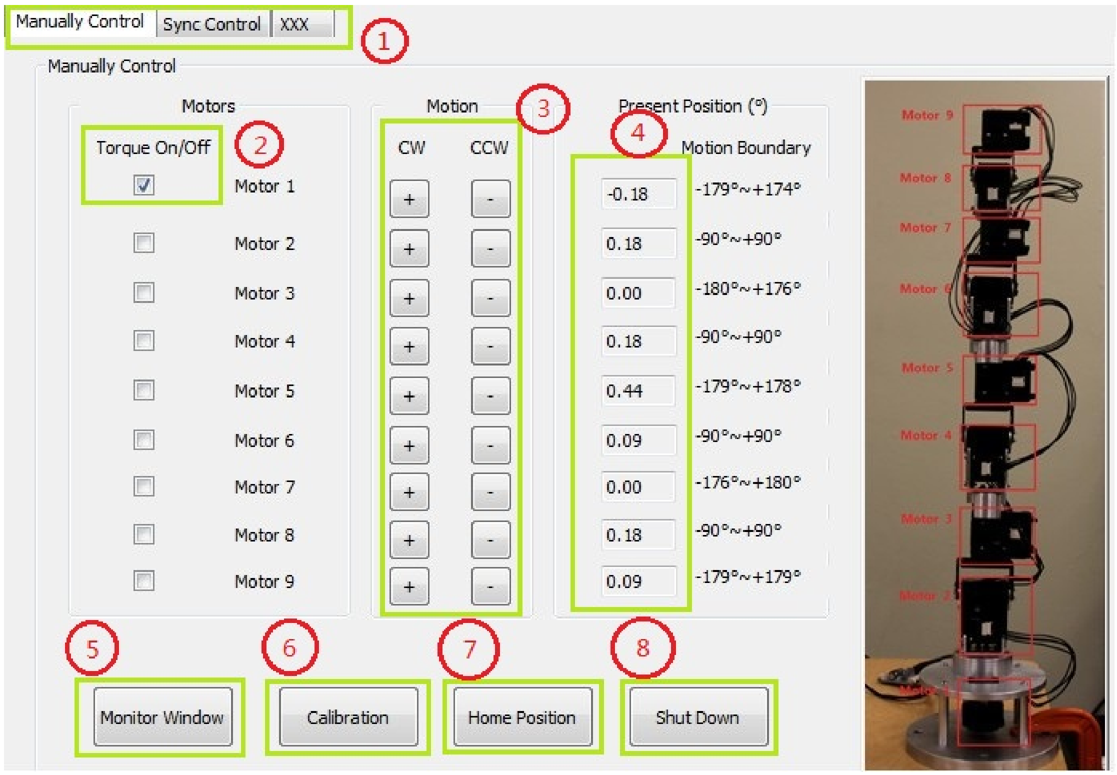
| Status | Value | | |
|---|---|---|---|
| ID | 1 | | |
| Torque Enable | 1 | | |
| D Gain | 0 | 0 | Change |
| I Gain | 0 | 0 | Change |
| P Gain | 32 | 0 | Change |
| Present Position (°) | -0.18 | | |
| Present Load (%) | 1.6 | | |
| Present Voltage (V) | 14.2 | | |
| Present Temperature (°C) | 26 | | |

Figure 5.1: Operation monitor interface diagram.

### Manual Control Interface

The manual control interface, which controls the torque switch and manual motion of motors, is shown as Figure 5.2 where *label 1* is a tab control for switching between interfaces, *label 2* is torque switches for each motor, *label 3* is manual motion control for two actuation directions (i.e., clockwise and counterclockwise), *label 4* is a monitor window of present position, *label 5* is for calling the monitor interface (i.e., 5.1), *label 6* is for recording calibration data of manipulator, *label 7* is a home position motion activated button, and *label 8* is a sleep position motion[1] activated button.

[1]A position which makes the manipulator to get close to the ground of platform and then lose all the torque of motors to shut down safely.

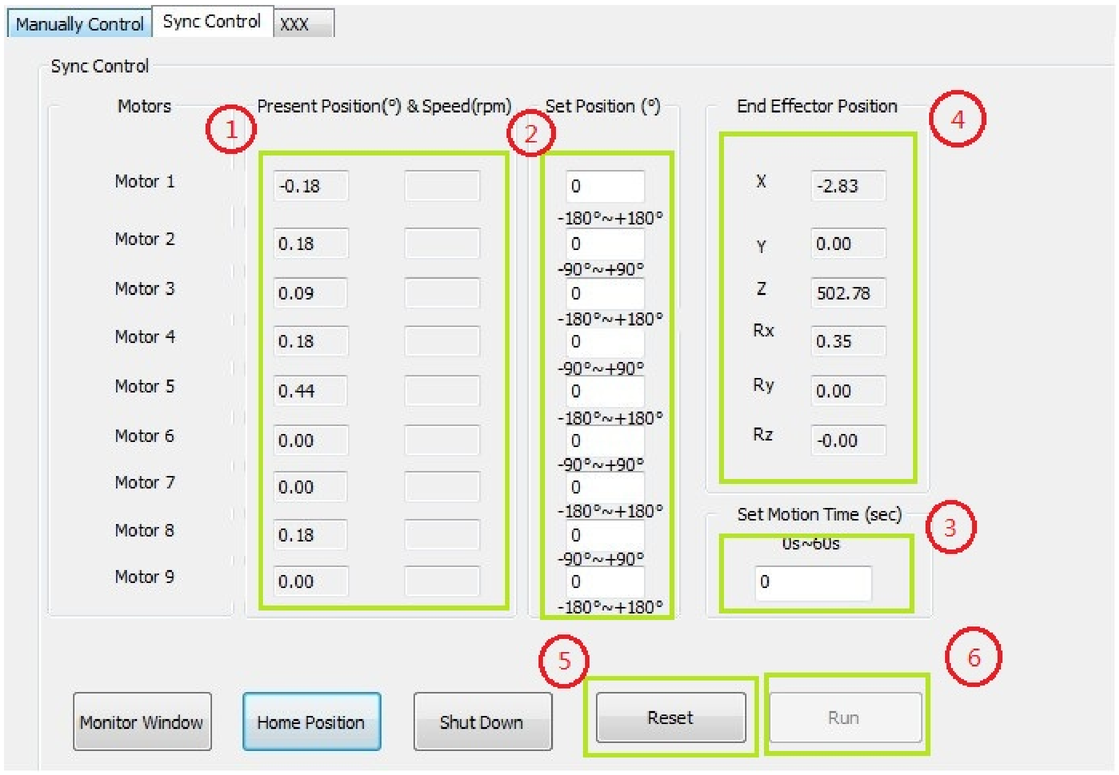


| Torque On/Off | Motors | Present Position (°) | Motion Boundary |
|---|---|---|---|
| ✓ | Motor 1 | -0.18 | -179°~+174° |
| | Motor 2 | 0.18 | -90°~+90° |
| | Motor 3 | 0.00 | -180°~+176° |
| | Motor 4 | 0.18 | -90°~+90° |
| | Motor 5 | 0.44 | -179°~+178° |
| | Motor 6 | 0.09 | -90°~+90° |
| | Motor 7 | 0.00 | -176°~+180° |
| | Motor 8 | 0.18 | -90°~+90° |
| | Motor 9 | 0.09 | -179°~+179° |

Figure 5.2: Manual control interface diagram.

### Sync Control Interface

The sync control interface, which allows all the motors to run synchronically, is shown as Figure 5.3 where *label 1* is a monitor window of present position and speed, *label 2* is for setting the positions for all the motors, *label 3* is for setting sync motion time for all the motors, *label 4* is a monitor window of position and orientation of the end-effector, *label 5* is for resetting all the input data, *label 6* is a sync motion activated button.



Sync Control

| Motors | Present Position(°) & Speed(rpm) | | Set Position (°) |
|---|---|---|---|
| Motor 1 | -0.18 | | 0 -180°~+180° |
| Motor 2 | 0.18 | | 0 -90°~+90° |
| Motor 3 | 0.09 | | 0 -180°~+180° |
| Motor 4 | 0.18 | | 0 -90°~+90° |
| Motor 5 | 0.44 | | 0 -180°~+180° |
| Motor 6 | 0.00 | | 0 -90°~+90° |
| Motor 7 | 0.00 | | 0 -180°~+180° |
| Motor 8 | 0.18 | | 0 -90°~+90° |
| Motor 9 | 0.00 | | 0 -180°~+180° |

End Effector Position

| | |
|---|---|
| X | -2.83 |
| Y | 0.00 |
| Z | 502.78 |
| Rx | 0.35 |
| Ry | 0.00 |
| Rz | -0.00 |

Set Motion Time (sec)
0s~60s
0

Figure 5.3: Sync control interface diagram.

### Safety Measures Used in UI

Safety incident of manipulator may be caused by misoperation of UI. There are several safety measures designed to avoid accidents. First, all the activated buttons and input windows (except monitor interface 5.1) are disable before the manipulator gets to its home position as initialization. Second, the manual control button is disable before the torque of corresponding motor is on. Third, all the input windows are disable during the sync motion of manipulator. Fourth, the sync motion activated button is available unless the motion time is set. Finally, UI closure leads to the sleep position motion to shut down the manipulator safely.

# CHAPTER 6

# RESULTS OF THESIS

In this chapter, we present the result of tracking different paths using the 9-DOF arm described earlier and report error analysis. Three sample paths as shown respectively in Section 6.1, 6.2 and 6.3 are designed to test accuracy and stability of the 9-DOF arm. We input all the sample joint variables and make the arm go though the path. During the real time tracking, We record the angles of each joint and calculate corresponding coordinates of the end-effector though forward kinematics (3.2) then provide spacial plot of trajectory, coordinates error analysis and velocity error analysis for each tracking experiment. However, the process of creating these proposed paths using inverse kinematics, will not be discussed in this thesis.

## 6.1 TRAJECTORY TRACKING OF A LINE ALONG THE X-AXIS

In this section, a line along the $X$-axis, which starts from coordinate (200, -236.9, -50) to (-200, -236.9, -50), is tracked by 9-DOF arm. The corresponding tracking-trajectory of the end-effector and reference path are shown in Figure 6.1. Also, Figure 6.2 shows different angles of view on $X$-$Y$ and $X$-$Z$ planes.

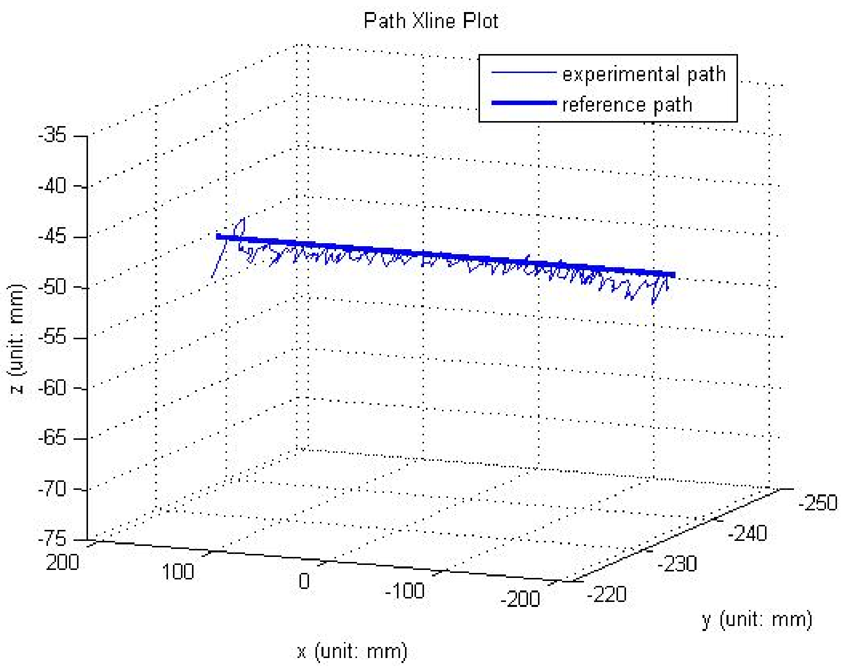
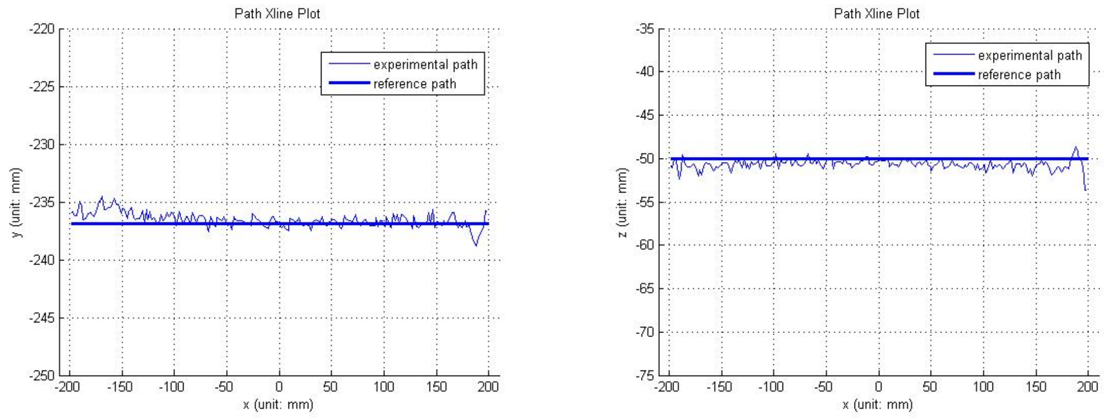


Figure 6.1: Tracking-trajectory of a line along $X$-axis is compared with designed reference path in space coordinates.



(a) $X$-$Y$ Plane (b) $X$-$Z$ Plane

Figure 6.2: Tracking-trajectory of a line along $X$-axis is compared with designed reference path on $X$-$Y$ and $X$-$Z$ planes.

The total motion time is 59.7 seconds. Figure 6.3 presents that the end-effector of 9-DOF arm tracks the line path accurately within its space distance error. The mean space distance error is 1.5295 *mm* and the maximum space distance error is 4.1948 *mm*.

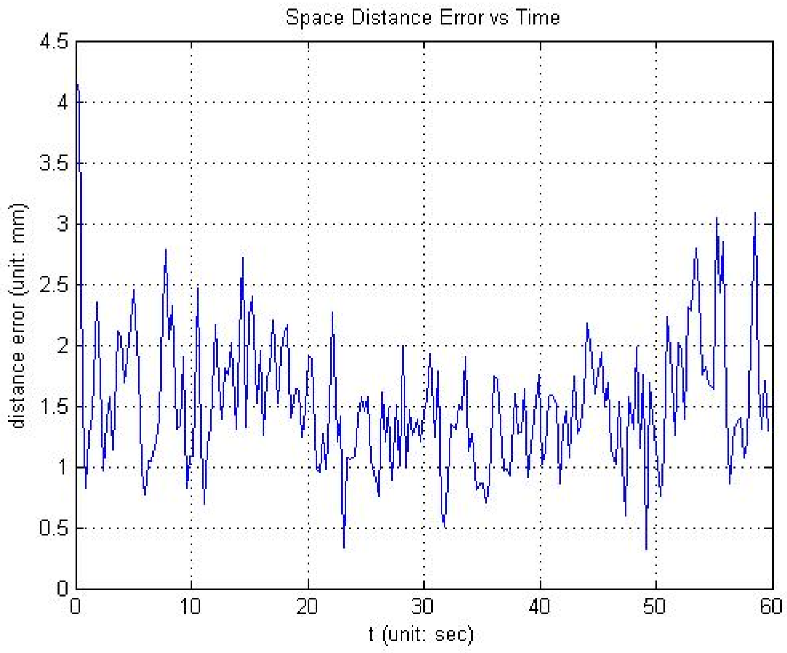


Figure 6.3: Space distance error of tracking-trajectory of a line along $X$-axis.

Figure 6.4 plots the tracking-trajectory and reference path in real time along $X$-axis, $Y$-axis and $Z$-axis. During the tracking, the mean position error and the maximum position error respectively along $X$-axis, $Y$-axis and $Z$-axis are as shown in Table 6.1.

Table 6.1: Position error analysis of tracking a line along $X$-axis

| **Axis** | **Mean Position Error (mm)** | **Maximum Position Error (mm)** |
|---|---|---|
| $X$ | 1.0763 | 2.0473 |
| $Y$ | 0.5337 | 2.3427 |
| $Z$ | 0.7345 | 3.6320 |

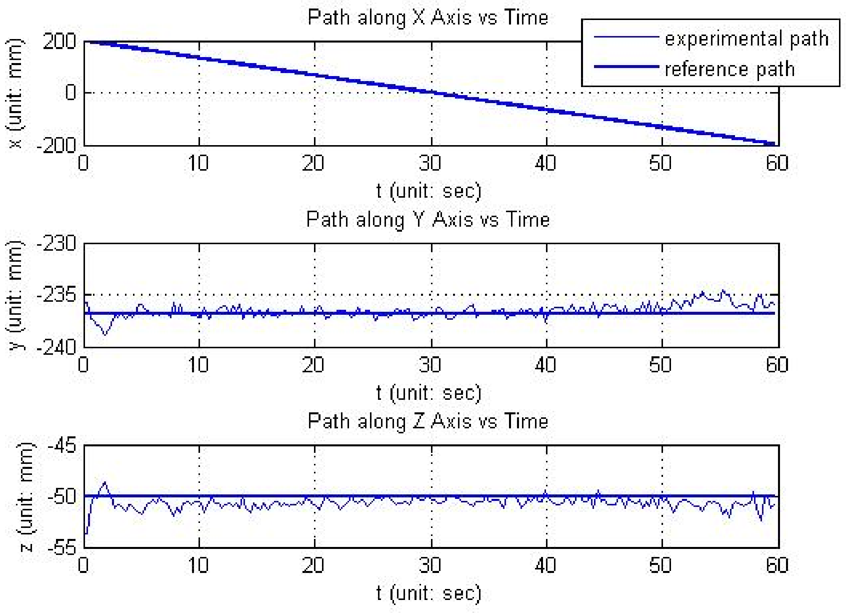


Figure 6.4: Tracking-trajectory of a line along $X$-axis versus motion time is compared with designed reference path versus motion time respectively along $X$-axis, $Y$-axis and $Z$-axis.

In order to get real-time velocity plot as shown in Figure 6.5, we take the derivative of tracking-position with respect to motion time as shown in Figure 6.4. The designed velocity respectively along $X$-axis, Y-axis and $Z$-axis are constantly -2 $mm/second$, 0 $mm/second$, 0 $mm/second$. The end-effector tracks the line path smoothly within the velocity error as shown in Figure 6.5. The mean velocity error and the maximum velocity error respectively along $X$-axis, $Y$-axis and $Z$-axis are as shown in Table 6.2.

Table 6.2: Velocity error analysis of tracking a line along $X$-axis

| **Axis** | **Mean Velocity Error** *(mm/second)* | **Maximum Velocity Error** *(mm/second)* |
|---|---|---|
| $X$ | 0.3762 | 1.3935 |
| $Y$ | 0.4020 | 1.5224 |
| $Z$ | 0.4547 | 2.6138 |

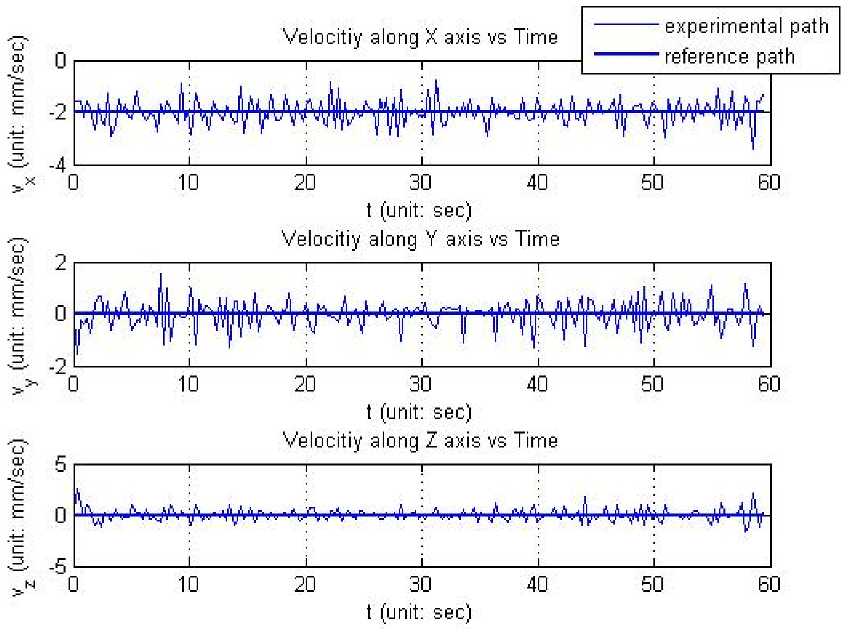


Figure 6.5: Velocity of tracking-trajectory of a line along $X$-axis is compared with velocity of designed reference path respectively along $X$-axis, $Y$-axis and $Z$-axis.

## 6.2 TRAJECTORY TRACKING OF A RECTANGLE ON X-Y PLANE

In this section, a path of a rectangle on $X$-$Y$ plane with four endpoints (100, -400, -50), (-100, -400, -50), (-100, -200, -50), (100, -200, -50) is tracked by 9-DOF arm. The corresponding tracking-trajectory of the end-effector and reference path are shown in Figure 6.6. Also, Figure 6.7 illustrates different angles of view on $X$-$Y$ and $X$-$Z$ planes.

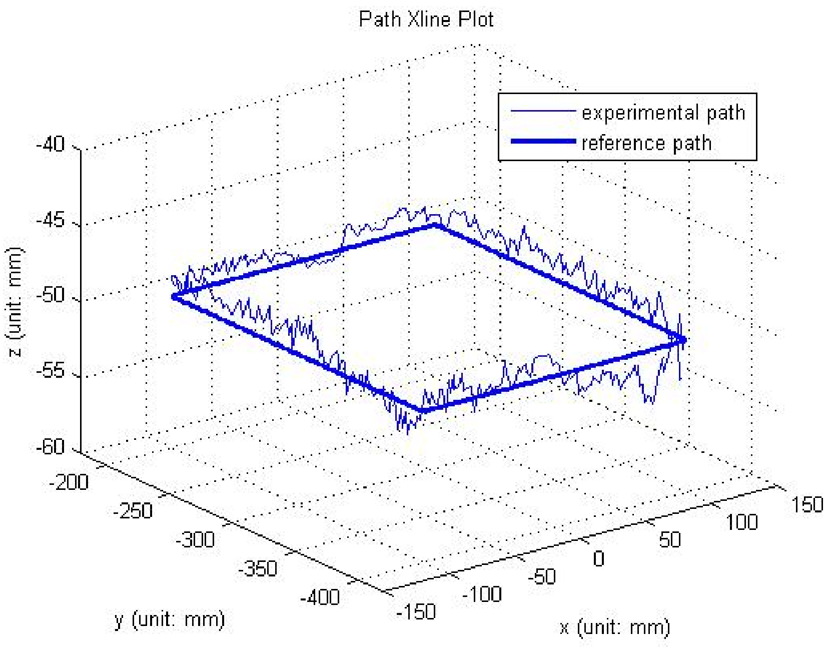
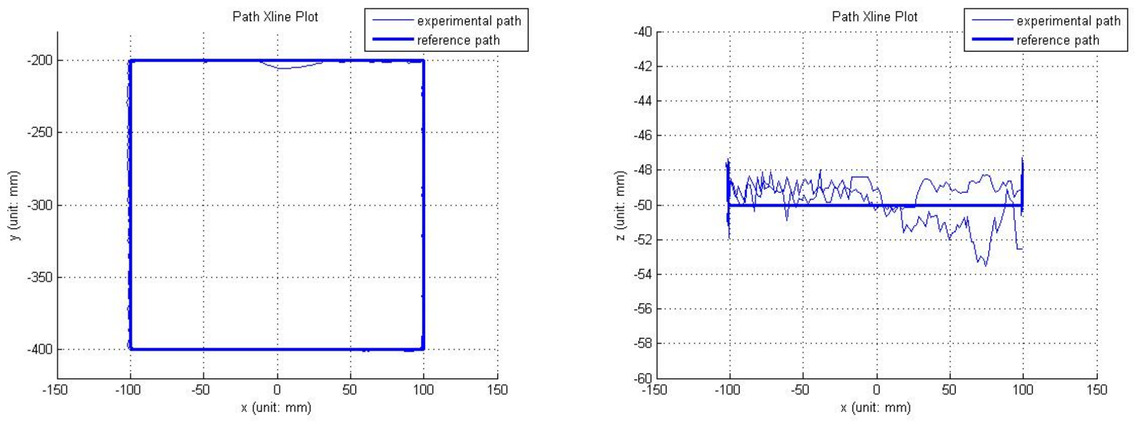


Figure 6.6: Tracking-trajectory of a rectangle on $X$-$Y$ plane is compared with designed reference path in space coordinates.



(a) $X$-$Y$ Plane (b) $X$-$Z$ Plane

Figure 6.7: Tracking-trajectory of a rectangle on $X$-$Y$ plane is compared with designed reference path on $X$-$Y$ and $X$-$Z$ planes.

The total motion time is 199.5 seconds. In Figure 6.8, we observe the end-effector of 9-DOF arm tracks the rectangle path accurately within its space distance error.

The mean space distance error is 1.9704 $mm$ and the maximum space distance error is 5.8564 $mm$.

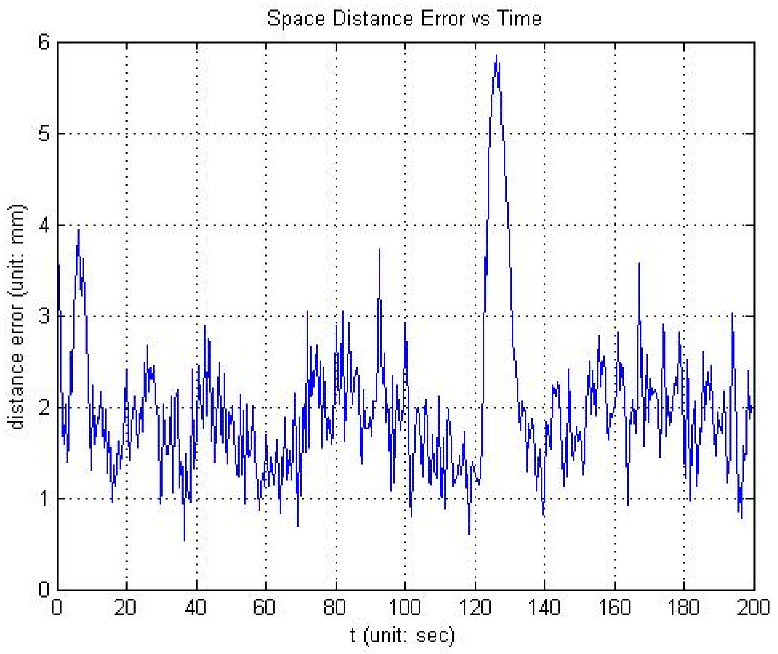


Figure 6.8: Space distance error of tracking-trajectory of a rectangle on $X$-$Y$ plane.

In Figure 6.9, we plot the tracking-trajectory and reference path in real time along $X$-axis, $Y$-axis and $Z$-axis. During the tracking, the mean position error and the maximum position error respectively along $X$-axis, $Y$-axis and $Z$-axis are as shown in Table 6.3.

Table 6.3: Position error analysis of tracking a rectangle on $X$-$Y$ plane

| **Axis** | **Mean Position Error (mm)** | **Maximum Position Error (mm)** |
|---|---|---|
| $X$ | 1.0010 | 2.7734 |
| $Y$ | 0.9316 | 5.7913 |
| $Z$ | 1.0648 | 3.5614 |

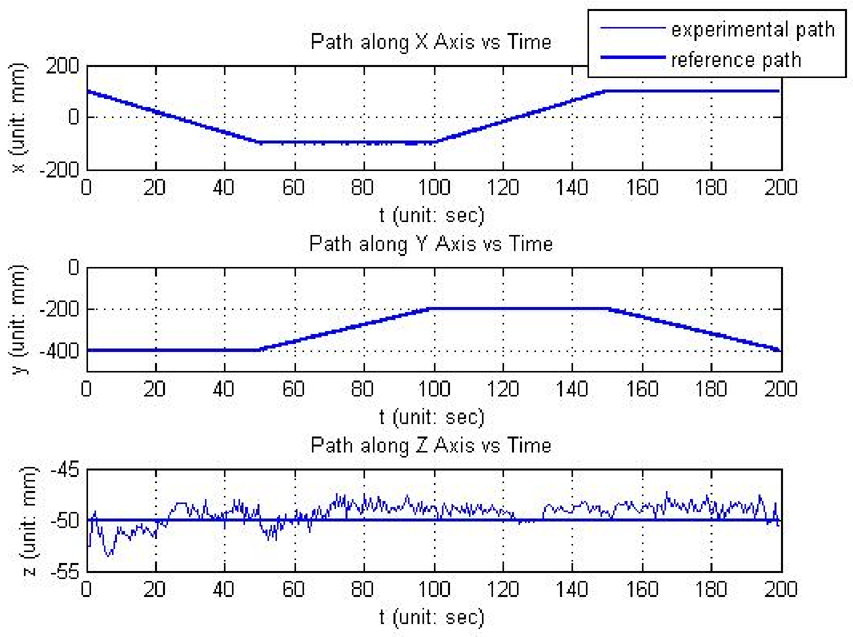


Figure 6.9: Tracking-trajectory of a rectangle on $X$-$Y$ plane versus motion time is compared with designed reference path versus motion time respectively along $X$-axis, $Y$-axis and $Z$-axis.

In order to get real-time velocity plot as shown in Figure 6.10, we take the derivative of tracking-position with respect to motion time as shown in Figure 6.9. The designed velocity respectively along $X$-axis, $Y$-axis and $Z$-axis are range from -2 $mm/second$ to +2 $mm/second$, range from +2 $mm/second$ to -2 $mm/second$ and constantly 0 $mm/second$. The end-effector tracks the rectangle path smoothly within the velocity error as shown in Figure 6.10. The mean velocity error and the maximum velocity error respectively along $X$-axis, $Y$-axis and $Z$-axis are as shown in Table 6.4.

Table 6.4: Velocity error analysis of tracking a rectangle on $X$-$Y$ plane

| **Axis** | **Mean Velocity Error** *(mm/second)* | **Maximum Velocity Error** *(mm/second)* |
|---|---|---|
| $X$ | 0.3885 | 2.0000 |
| $Y$ | 0.3411 | 2.3485 |
| $Z$ | 0.5026 | 2.6881 |

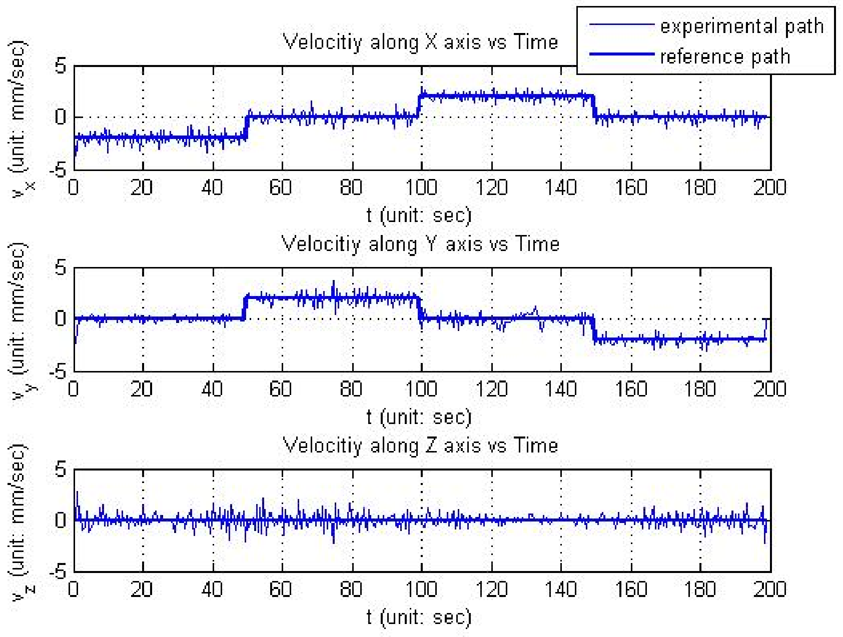


Figure 6.10: Velocity of tracking-trajectory of a rectangle on $X$-$Y$ plane is compared with velocity of designed reference path respectively along $X$-axis, $Y$-axis and $Z$-axis.

## 6.3 TRAJECTORY TRACKING OF AN INCLINED CIRCLE

In this section, a path of an inclined circle with its center coordinate (-315, 0, 10) and radius of 100 $mm$ is tracked by 9-DOF arm. The corresponding tracking-trajectory of the end-effector and reference path are shown in Figure 6.11. Also, Figure 6.12 shows different angles of view on $X$-$Y$ and $X$-$Z$ planes.

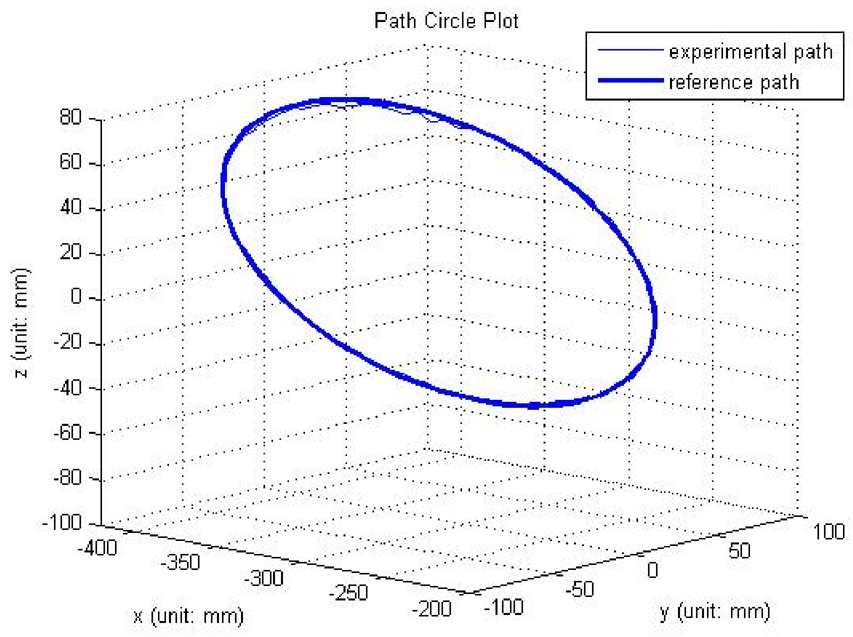
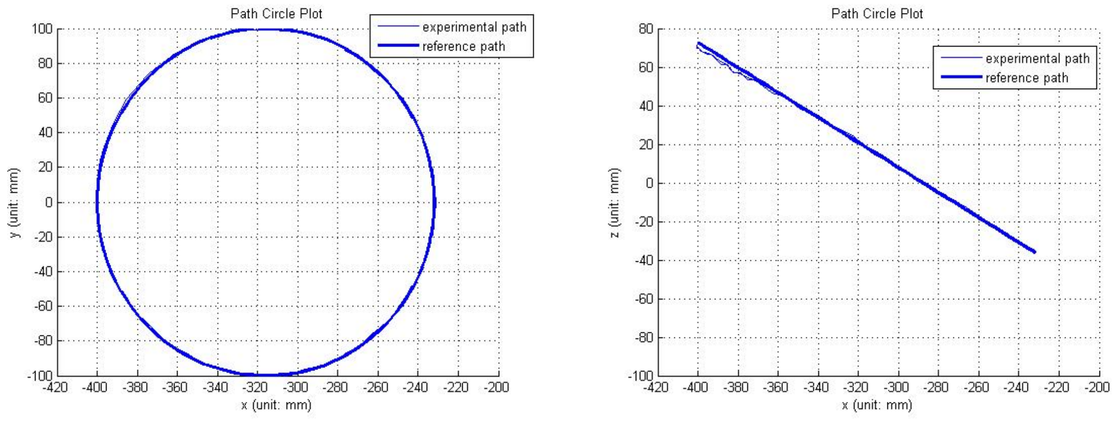


Figure 6.11: Tracking-trajectory of an inclined circle is compared with designed reference path in space coordinates.



(a) $X$-$Y$ Plane

(b) $X$-$Z$ Plane

Figure 6.12: Tracking-trajectory of an inclined circle is compared with designed reference path on $X$-$Y$ and $X$-$Z$ planes.

The total motion time is 49.5 seconds. Figure 6.13 presents the end-effector of 9-DOF arm tracks the circle path accurately within its space distance error. The mean space distance error is 1.5007 $mm$ and the maximum space distance error is 4.2913 $mm$.

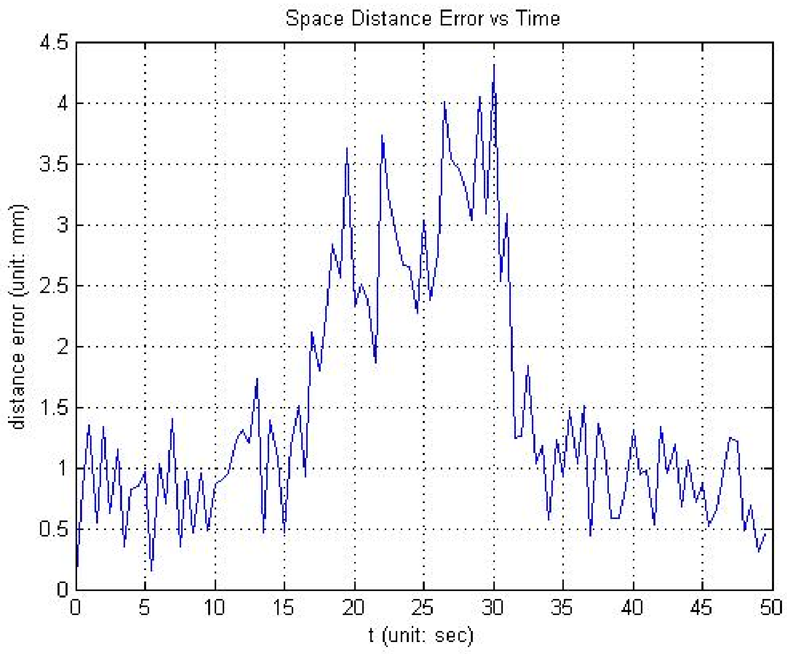


Figure 6.13: Space distance error of tracking-trajectory of an inclined circle.

Figure 6.14 shows the tracking-trajectory and reference path in real time along $X$-axis, $Y$-axis and $Z$-axis. During the tracking, the mean position error and the maximum position error respectively along $X$-axis, $Y$-axis and $Z$-axis are as shown in Table 6.5.

Table 6.5: Position error analysis of tracking an inclined circle

| **Axis** | **Mean Position Error (mm)** | **Maximum Position Error (mm)** |
|---|---|---|
| $X$ | 0.4017 | 1.0141 |
| $Y$ | 0.7819 | 2.5465 |
| $Z$ | 1.0889 | 4.0433 |

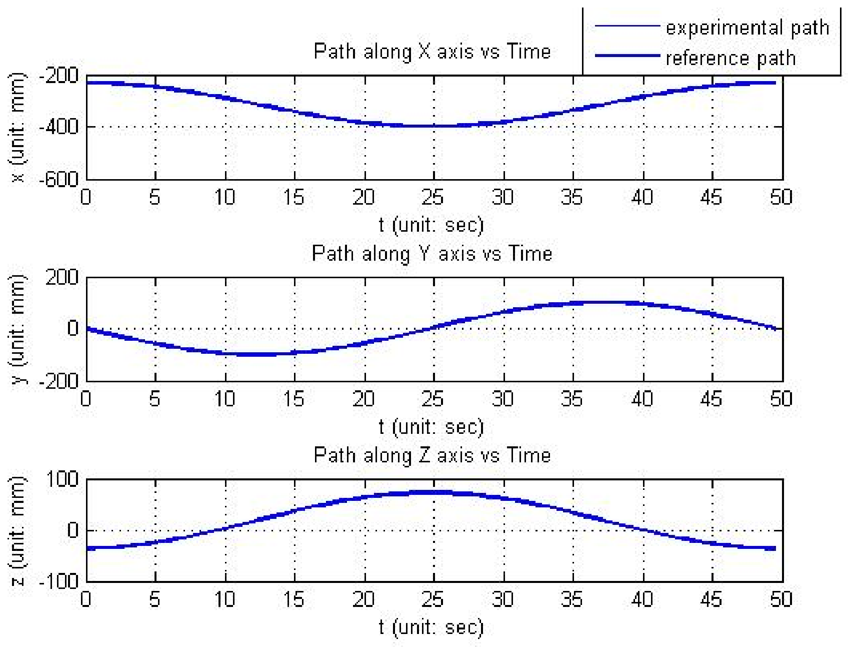


Figure 6.14: Tracking-trajectory of an inclined circle versus motion time is compared with designed reference path versus motion time respectively along $X$-axis, $Y$-axis and $Z$-axis.

In order to get real-time velocity plot as shown in Figure 6.15, we take the derivative of tracking-position with respect to motion time as shown in Figure 6.14. The designed velocity respectively along $X$-axis, $Y$-axis and $Z$-axis are $-5.3sin(2\pi t/49.5)$ $mm/second$, $-6.4cos(2\pi t/49.5)$ $mm/second$ and $3.5sin(2\pi t/49.5)$ $mm/second$ where $t$ is motion time. The end-effector tracks the circle path smoothly within the velocity error as shown in Figure 6.15. The mean velocity error and the maximum velocity error respectively along $X$-axis, $Y$-axis and $Z$-axis are as shown in Table 6.6.

Table 6.6: Velocity error analysis of tracking an inclined circle

| **Axis** | **Mean Velocity Error** *(mm/second)* | **Maximum Velocity Error** *(mm/second)* |
|---|---|---|
| $X$ | 0.2492 | 1.0220 |
| $Y$ | 0.5322 | 1.7197 |
| $Z$ | 0.6133 | 2.2843 |

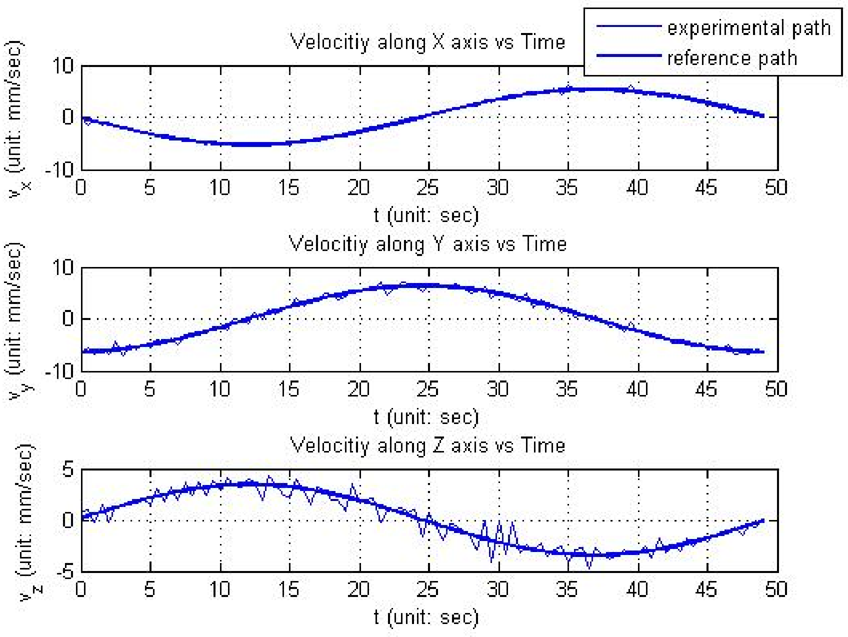


Figure 6.15: Velocity of tracking-trajectory of an inclined circle is compared with velocity of designed reference path respectively along $X$-axis, $Y$-axis and $Z$-axis.

## 6.4 CONCLUSION OF THESIS RESULTS

In general, the 9-DOF arm tracks the three designed path smoothly with certain accuracy. However, the performance can be significantly improved by proper tuning of the control parameters and reducing the delay in communication. The results shows that the 9-DOF manipulator works functionally and satisfies the requirement of this thesis design.

# CHAPTER 7

# CONCLUSION

Along this thesis, we have constructed our own 9-DOF hyper-redundant manipulator with open architecture control based on kinematic modeling and dynamic analysis. Also, we have developed corresponding original function library and application software with UI. Meanwhile, we have obtained a lot of research experience and experimental ability during this thesis. First of all, we acquire knowledge of robotics and related work, e.g., forward and inverse kinematics, servo motor, C++ programming, etc. Second, we get familiar with the design and construction of hardware platform. Third, we succeed to develop our own software to achieve manually control, sync control, tracking the trajectory using forward kinematics, etc. Chapter 6 validates the performance of designed manipulator platform and testifies that the 9-DOF manipulator satisfies the requirement of this thesis design and establish a foundation for future research.

## 7.1 DIFFICULTIES AND SOLUTIONS

A few difficulties appear along with this thesis design. First is the safety problem of the hardware platform. The “MX series” servo motors used in this thesis only have basic safety precautions, like overload warning, communication error warning, etc. These precautions are not sufficient to ensure the experimental safety. Thus, we embed a few actuation limitation of each motor into the software function library. It means that the motors are unavailable to reach the position or to achieve the speed beyond the safety boundary. In addition, we set a series of safety measures 5.2.2 inside UI to avoid artificial mistake.

Another one is that the motors actuate in the horizontal plane are physically swapped during the installation to match co-ordinate system of kinematic model. As a result, the positive direction of motor rotation is opposite from which in the kinematic model. We need either change the direction and make a special notation in the function code or change the DH table to fix this problem.

In practical use, Equation (3.3) stands unstable when calculating orientation of the end-effector due to the round off error and truncation error of computer. Orientation of the end-effector(i.e., $R_x$, $R_y$, $R_z$) changes discontinuously when approach to the critical point. Therefore, we rewrite the function code of Equation (3.3) to set up some transitional domain eliminating the computer error to make the orientation to change smoothly.

The servo motors without dynamic control are incapable to move synchronously while executing sync write. This problem will affect the smoothness and synchronism of robotic motion. As a solution, we put different scaler to the transition between speed and signal pulse for different type of motor in the function code to coordinate their speed under different situations. In the future, we can develop PID control or achieve torque control to avoid this problem.

## 7.2 FUTURE WORK

The 9-DOF manipulator designed in this thesis is an open-oriented platform to continuous robotic and control research. It can be extended to test and verify the inverse kinematic algorithm. We can experiment and observe the motion trajectory of the end-effector calculated though inverse kinematics. The path planning of the end-effector can also be proposed based on the velocity kinematics. We get dynamic profile of each joint from the Manipulator Jacobian [2, 10, 11] to optimize and control the path. In addition, we can implement independent joint control by control the torque of each motor. The adaptive control [17] can optimize and control the multiple inputs and outputs of a non-linear system.

# APPENDIX A

## Visual C++ Function Library Based on Dynamixel SDK

**PrintCommStatus.cpp**

```
void PrintCommStatus(int CommStatus)
    {
        switch(CommStatus)
        {
        case COMM_TXFAIL:
        printf("COMM_TXFAIL: Failed transmit instruction 
           packet!\n");
        break;

        case COMM_TXERROR:
        printf("COMM_TXERROR: Incorrect instruction packet!\n
           ");
        break;

        case COMM_RXFAIL:
        printf("COMM_RXFAIL: Failed get status packet from 
           device!\n");
        break;
        case COMM_RXWAITING:
        printf("COMM_RXWAITING: Now recieving status packet!\
            n");
        break;

        case COMM_RXTIMEOUT:
        printf("COMM_RXTIMEOUT: There is no status packet!\n"
            );
        break;

        case COMM_RXCORRUPT:
        printf("COMM_RXCORRUPT: Incorrect status packet!\n");
        break;

        default:
        printf("This is unknown error code!\n");
        break;
    }
        }
```

**PrintErrorCode.cpp**

```
void PrintErrorCode()
    {
        if(dxl_get_rxpacket_error(ERRBIT_VOLTAGE) == 1)
        printf("Input voltage error!\n");
        if(dxl_get_rxpacket_error(ERRBIT_ANGLE) == 1)
        printf("Angle limit error!\n");

        if(dxl_get_rxpacket_error(ERRBIT_OVERHEAT) == 1)
        printf("Overheat error!\n");

        if(dxl_get_rxpacket_error(ERRBIT_RANGE) == 1)
        printf("Out of range error!\n");

        if(dxl_get_rxpacket_error(ERRBIT_CHECKSUM) == 1)
        printf("Checksum error!\n");

        if(dxl_get_rxpacket_error(ERRBIT_OVERLOAD) == 1)
        printf("Overload error!\n");

        if(dxl_get_rxpacket_error(ERRBIT_INSTRUCTION) == 1)
        printf("Instruction code error!\n");
    }
```

**SetSpeed.cpp**

```
void SetSpeed(int id, float speedset)
    {
        dxl_write_word(id, 32,MotorData::Transform::
           SpeedToPulses(id, speedset));
        //#define P_GOAL_SPEED_L                32
    };
```

**GetSpeed.cpp**

```
float GetSpeed(int id)
    {
        return MotorData::Transform::PulsesToSpeed(
           dxl_read_word(id, 38));
        //#define P_PRESENT_SPEED_L            38
        };
```

**SetGoalPosition.cpp**

```
void SetGoalPosition(int id, float positionset1)
    {
        dxl_write_word(id, 30, MotorData::Transform::
           PositionToPulses(id, positionset1));
        //#define P_GOAL_POSITION_L            30
        };
```

**GetPresPosition.cpp**

```
float GetPresPosition(int id)
    {
        return float (MotorData::Transform::PulsesToPosition(
           id, dxl_read_word(id, 36)));
        //#define P_PRESENT_POSITION_L         36
        };
```

**GetPresLoad.cpp**

```
float GetPresLoad(int id)
    {
        float presload;
        int load = dxl_read_word( id, 40);
        if (load>=0 && load<=1023)
        {
                presload = 100.0*load/1023.0;
        }
        else
        {
                presload = 100.0*(load-1024)/1023.0;
        }
        return presload;
        };
```

**SyncWrite.cpp**

```
// Control table address
#define P_GOAL_POSITION_L       30
#define P_GOAL_POSITION_H       31

// Defulat setting
#define DEFAULT_PORTNUM         4 // COM4
#define DEFAULT_BAUDNUM         1 // 1Mbps

void SyncWrite(int number, float *pos, float *speed)
    {
```

```
int id[9];
int GoalPos[9];
int i;
// Initialize id
for( i=0; i<number; i++ )
{
        id[i] = i+1;
}
// Make syncwrite packet
dxl_set_txpacket_id(BROADCAST_ID);
dxl_set_txpacket_length((4+1)*number+4);
dxl_set_txpacket_instruction(INST_SYNC_WRITE);
dxl_set_txpacket_parameter(0, P_GOAL_POSITION_L);
dxl_set_txpacket_parameter(1, 4);

for( i=0; i<number; i++ )
{
        dxl_set_txpacket_parameter(2+5*i, id[i]);
        GoalPos[i] = MotorData::Transform::
            PositionToPulses(id[i], pos[i]);
        dxl_set_txpacket_parameter(2+5*i+1,
            dxl_get_lowbyte(GoalPos[i]));
        dxl_set_txpacket_parameter(2+5*i+2,
            dxl_get_highbyte(GoalPos[i]));
            dxl_set_txpacket_parameter(2+5*i+3,
                dxl_get_lowbyte(MotorData::Transform::
                SpeedToPulses(id[i], speed[i])));
            dxl_set_txpacket_parameter(2+5*i+4,
                dxl_get_highbyte(MotorData::Transform::
                SpeedToPulses(id[i], speed[i])));
                    }
            dxl_txrx_packet();
    }
    }
```

**PulsesToSpeed.cpp**

```
float PulsesToSpeed(int pulsestrans1)
    {
        if (pulsestrans1==0)
        {
            return 0.0;
        }
        else if (pulsestrans1==1023)
        {
            return -117.07;
        }
        else if ((pulsestrans1>0)&&(pulsestrans1<1023))
        {
            return (117.07/1023)*pulsestrans1;
        }
        else
                return (-117.07/1023)*(pulsestrans1 -1024);
        };
```

**SpeedToPulses.cpp**

```
float thres1 = 109.5, thres2 = 117.07;
int SpeedToPulses(int id, float speedtrans)
    {
        if ((id==7)||(id==8)||(id==9))
        {
                if ((speedtrans >0.0) && (speedtrans<=thres1))
                {
                        return (max(1,speedtrans/(thres1
                           /1023.0)));
                }
                else if (speedtrans>thres1)
                {
                        return 1023;
                }
                else return 1;
            }
        else
        {
                if ((speedtrans >0.0) && (speedtrans<=thres2))
                {
                    return (max(1,speedtrans/(thres2
                       /1023.0)));
                }
                else if (speedtrans>thres2)
                {
                    return 1023;
                }
                else return 1;
        }
        }
```

**PulsesToPosition.cpp**

```
float PulsesToPosition(int id, int pulsestrans2)
    {
        float HomePosition[9]={2074.0, 2040.0, 2001.0,
           2038.0, 2028.0, 1987.0, 2226.0, 2028.0, 2051.0};

        if(id==2 || id==4 || id==6 || id==8 )
        {
                return (pulsestrans2-HomePosition[(id-1)])
                   *(180.0/2048.0);
        }
        else
        {
                return (pulsestrans2-HomePosition[(id-1)])
                   *(180.0/2048.0);
        }
        }
```

**PositionToPulses.cpp**

```
int PositionToPulses(int id, float positiontrans)
    {
        float HomePosition[9]={2074.0, 2040.0, 2001.0,
           2038.0, 2028.0, 1987.0, 2226.0, 2028.0, 2051.0};

        if(id==2 || id==4 || id==6 || id==8 )
        {
                if(positiontrans>=0.0)
                {
                        return HomePosition[id-1]-(max
                           (-110.0,-positiontrans)
                           /(180.0/2048.0));
                }
                else
                {
                        return HomePosition[id-1]-(min
                           (110.0,-positiontrans)
                           /(180.0/2048.0));
                }
        }
        else if(id==1 || id==3 || id==5 || id==7 || id==9)
        {
```

```
        if(positiontrans>=0.0 )
        {
                return HomePosition[id-1]-(max
                    (-180.0,-positiontrans)
                    /(180.0/2048.0));
        }
        else
        {
                return HomePosition[id-1]-(min
                    (180.0,-positiontrans)
                    /(180.0/2048.0));
        }
}

if (pulse >= 4096)
        {
                return (pulse-4096);
        }
        else
        {
                return pulse;
        }
}
```

**CalculateSpeed.cpp**

```
float CalculateSpeed(int number, float time, float *pos,
   float *speed)
    {
        float position[9];
        int id[9];
        int i;
        // Initialize id

        for( i=0; i<number; i++ )
        {
                id[i] = i+1;
        }
        for( i=0; i<number; i++ )
        {
                position[i] = MotorStatus::MotorMotion::
                   GetPresPosition(id[i]);
                if((pos[i]-position[i])>=0.0)
                {
                        speed[i] = ((pos[i]-position[i])
                           /360.0*60.0)/time;
                }
                else
                {
                        speed[i] = ((position[i]-pos[i])
                           /360.0*60.0)/time;
                }
        }
        return *speed;
        }
```

**DegreeToRadian.cpp**

```
float DegreeToRadian(float degree)
        {
                return degree*pi/180.0;
        };
```

**RadianToDegree.cpp**

```
float RadianToDegree(float radian)
    {
                return radian*180.0/pi;
        };
```

**ForwardKinematics.cpp**

```
float ForwardKinematics(float *endeffector, float *pos)
    {
        float pos1= MotorData::Transform::DegreeToRadian(pos
            [0]);
        float pos2= MotorData::Transform::DegreeToRadian(pos
            [1]);
        float pos3= MotorData::Transform::DegreeToRadian(pos
            [2]);
```

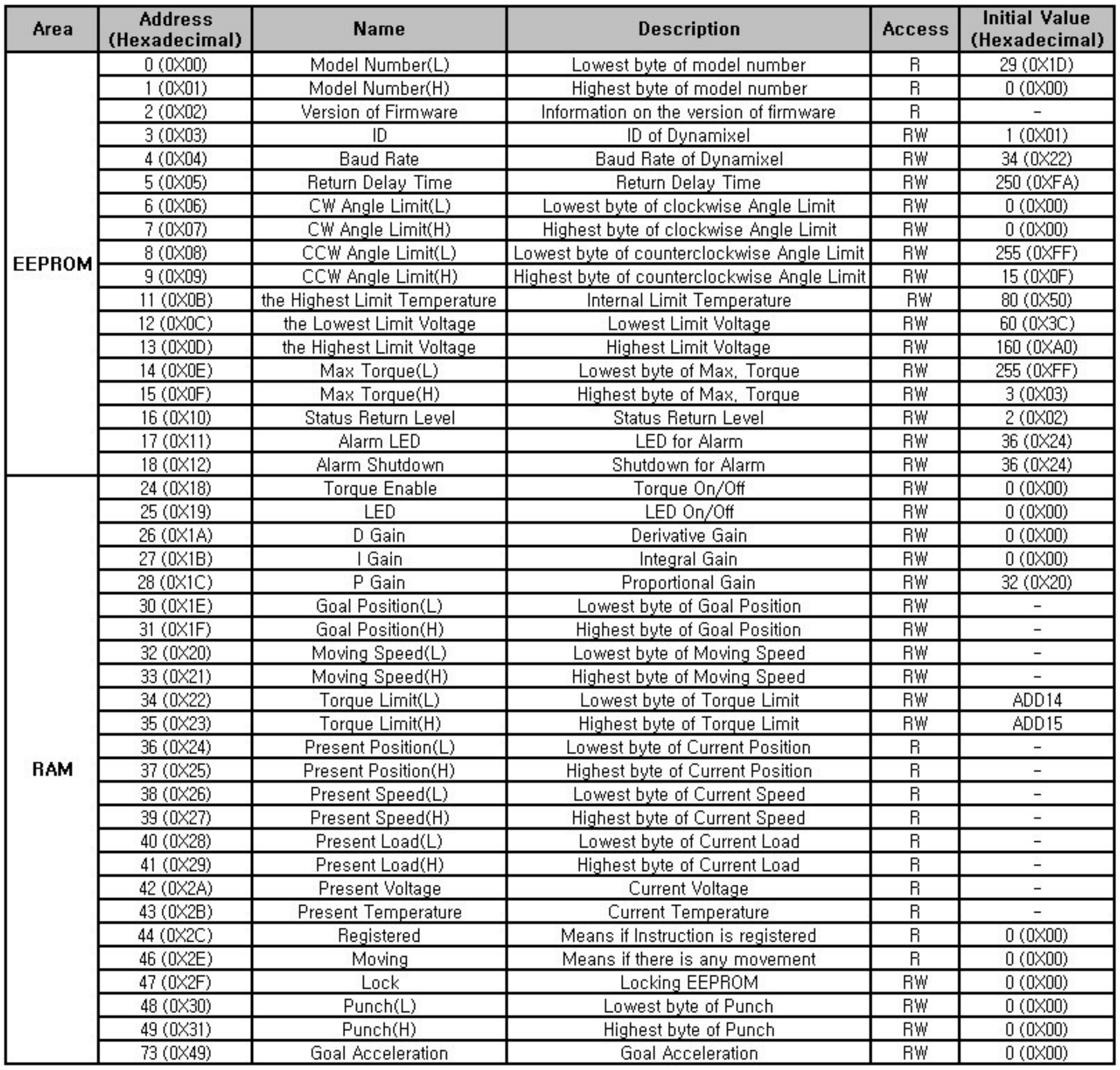
```
float pos4= MotorData::Transform::DegreeToRadian(pos
    [3]);
float pos5= MotorData::Transform::DegreeToRadian(pos
    [4]);
float pos6= MotorData::Transform::DegreeToRadian(pos
    [5]);
float pos7= MotorData::Transform::DegreeToRadian(pos
    [6]);
float pos8= MotorData::Transform::DegreeToRadian(pos
    [7]);
float pos9= MotorData::Transform::DegreeToRadian(pos
    [8]);

endeffector[4] = atan2(-nz, sqrt(nx*nx+ny*ny));
endeffector[4] = (float)((int)(endeffector[4] *
    100.0)) / 100.0;

if ((endeffector[4] <= 3.14/2.0 + 0.10) && (
    endeffector[4] >= 3.14/2.0 - 0.10))
{
        endeffector[3] = 0;
        endeffector[5] = atan2(ox, oy);
}
else if ((endeffector[4] >= -3.14/2.0 - 0.10) && (
    endeffector[4] <= -3.14/2.0 + 0.10))
```

```
{
        endeffector[3] = 0;
        endeffector[5] = -atan2(ox, oy);
}
else
{
        endeffector[3] = atan2(ny/cos(endeffector[4])
            , nx/cos(endeffector[4]));
        endeffector[5] = atan2(oz/cos(endeffector[4])
            , az/cos(endeffector[4]));
}

//Rotation radian to degree

endeffector[3] = MotorData::Transform::RadianToDegree
    (endeffector[3]);
endeffector[4] = MotorData::Transform::RadianToDegree
    (endeffector[4]);
endeffector[5] = MotorData::Transform::RadianToDegree
    (endeffector[5]);

return *endeffector;
}
```

# APPENDIX B

## Dynamixel Servo Motor Control Address Table

Table B.1: Dynamixel servo motor control command address (source: [3])



| Area | Address (Hexadecimal) | Name | Description | Access | Initial Value (Hexadecimal) |
|---|---|---|---|---|---|
| EEPROM | 0 (0X00) | Model Number(L) | Lowest byte of model number | R | 29 (0X1D) |
| | 1 (0X01) | Model Number(H) | Highest byte of model number | R | 0 (0X00) |
| | 2 (0X02) | Version of Firmware | Information on the version of firmware | R | - |
| | 3 (0X03) | ID | ID of Dynamixel | RW | 1 (0X01) |
| | 4 (0X04) | Baud Rate | Baud Rate of Dynamixel | RW | 34 (0X22) |
| | 5 (0X05) | Return Delay Time | Return Delay Time | RW | 250 (0XFA) |
| | 6 (0X06) | CW Angle Limit(L) | Lowest byte of clockwise Angle Limit | RW | 0 (0X00) |
| | 7 (0X07) | CW Angle Limit(H) | Highest byte of clockwise Angle Limit | RW | 0 (0X00) |
| | 8 (0X08) | CCW Angle Limit(L) | Lowest byte of counterclockwise Angle Limit | RW | 255 (0XFF) |
| | 9 (0X09) | CCW Angle Limit(H) | Highest byte of counterclockwise Angle Limit | RW | 15 (0X0F) |
| | 11 (0X0B) | the Highest Limit Temperature | Internal Limit Temperature | RW | 80 (0X50) |
| | 12 (0X0C) | the Lowest Limit Voltage | Lowest Limit Voltage | RW | 60 (0X3C) |
| | 13 (0X0D) | the Highest Limit Voltage | Highest Limit Voltage | RW | 160 (0XA0) |
| | 14 (0X0E) | Max Torque(L) | Lowest byte of Max. Torque | RW | 255 (0XFF) |
| | 15 (0X0F) | Max Torque(H) | Highest byte of Max. Torque | RW | 3 (0X03) |
| | 16 (0X10) | Status Return Level | Status Return Level | RW | 2 (0X02) |
| | 17 (0X11) | Alarm LED | LED for Alarm | RW | 36 (0X24) |
| | 18 (0X12) | Alarm Shutdown | Shutdown for Alarm | RW | 36 (0X24) |
| RAM | 24 (0X18) | Torque Enable | Torque On/Off | RW | 0 (0X00) |
| | 25 (0X19) | LED | LED On/Off | RW | 0 (0X00) |
| | 26 (0X1A) | D Gain | Derivative Gain | RW | 0 (0X00) |
| | 27 (0X1B) | I Gain | Integral Gain | RW | 0 (0X00) |
| | 28 (0X1C) | P Gain | Proportional Gain | RW | 32 (0X20) |
| | 30 (0X1E) | Goal Position(L) | Lowest byte of Goal Position | RW | - |
| | 31 (0X1F) | Goal Position(H) | Highest byte of Goal Position | RW | - |
| | 32 (0X20) | Moving Speed(L) | Lowest byte of Moving Speed | RW | - |
| | 33 (0X21) | Moving Speed(H) | Highest byte of Moving Speed | RW | - |
| | 34 (0X22) | Torque Limit(L) | Lowest byte of Torque Limit | RW | ADD14 |
| | 35 (0X23) | Torque Limit(H) | Highest byte of Torque Limit | RW | ADD15 |
| | 36 (0X24) | Present Position(L) | Lowest byte of Current Position | R | - |
| | 37 (0X25) | Present Position(H) | Highest byte of Current Position | R | - |
| | 38 (0X26) | Present Speed(L) | Lowest byte of Current Speed | R | - |
| | 39 (0X27) | Present Speed(H) | Highest byte of Current Speed | R | - |
| | 40 (0X28) | Present Load(L) | Lowest byte of Current Load | R | - |
| | 41 (0X29) | Present Load(H) | Highest byte of Current Load | R | - |
| | 42 (0X2A) | Present Voltage | Current Voltage | R | - |
| | 43 (0X2B) | Present Temperature | Current Temperature | R | - |
| | 44 (0X2C) | Registered | Means if Instruction is registered | R | 0 (0X00) |
| | 46 (0X2E) | Moving | Means if there is any movement | R | 0 (0X00) |
| | 47 (0X2F) | Lock | Locking EEPROM | RW | 0 (0X00) |
| | 48 (0X30) | Punch(L) | Lowest byte of Punch | RW | 0 (0X00) |
| | 49 (0X31) | Punch(H) | Highest byte of Punch | RW | 0 (0X00) |
| | 73 (0X49) | Goal Acceleration | Goal Acceleration | RW | 0 (0X00) |

# APPENDIX C

## Mechanical Drawings

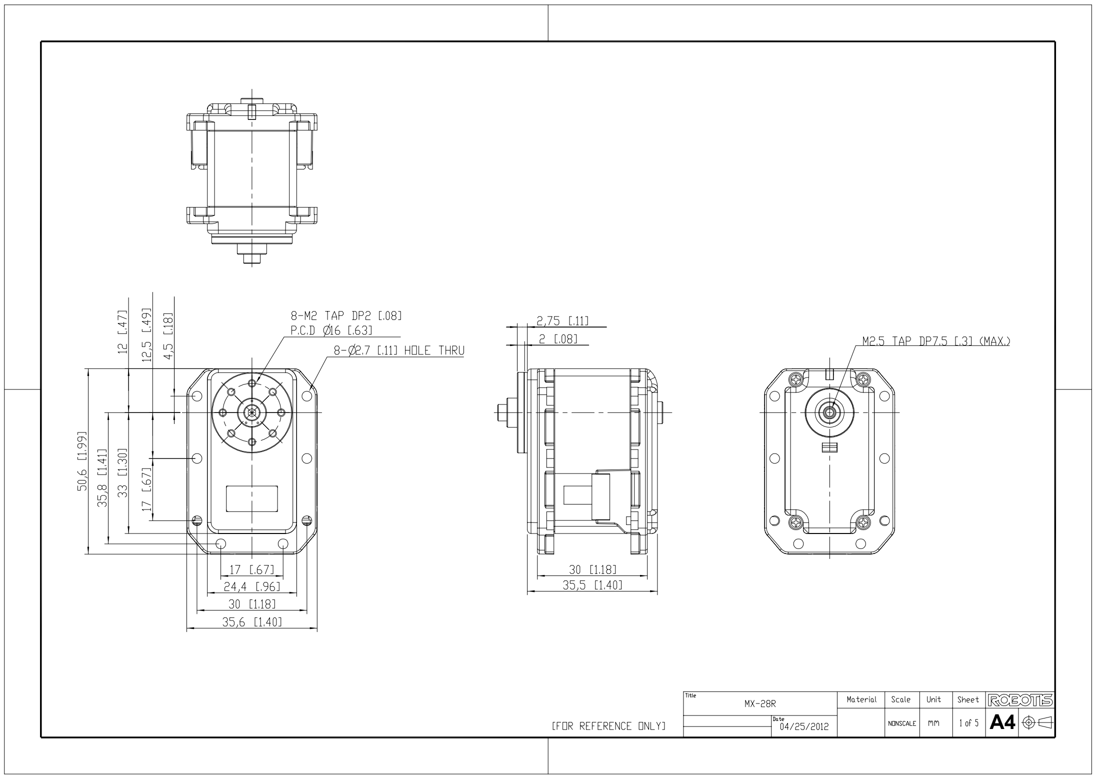


Figure C.1: MX-28R servo motor mechanical drawing (source: [1]).

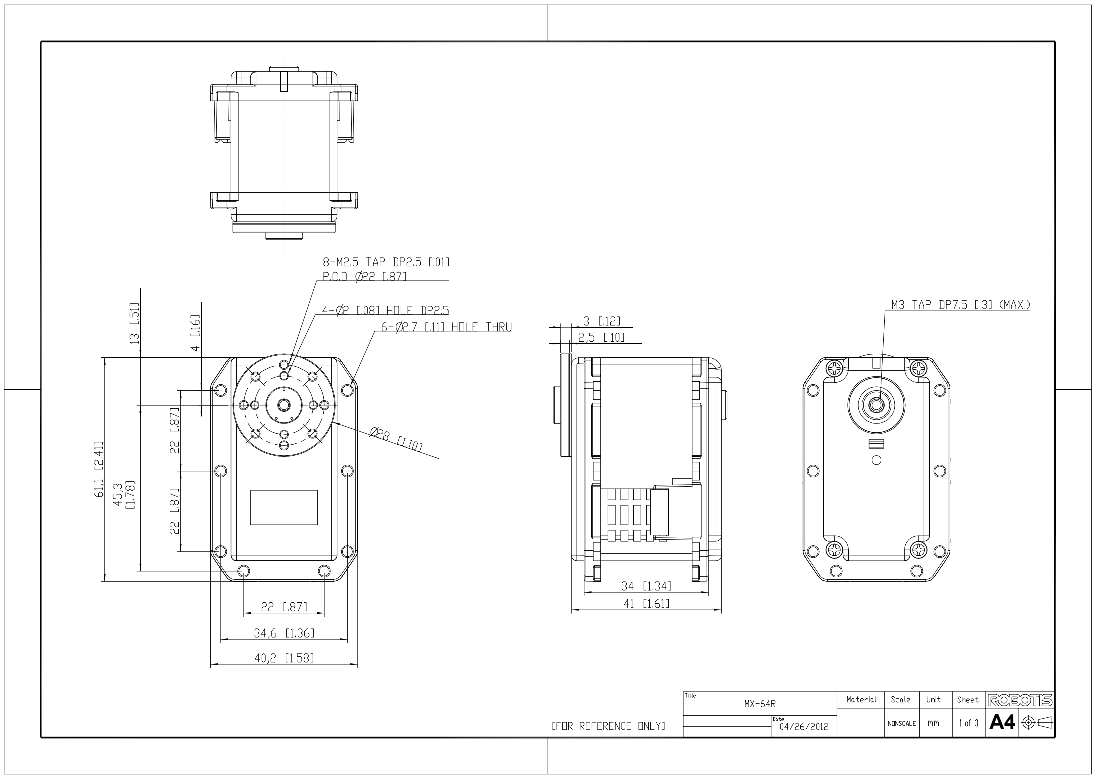


Figure C.2: MX-64R servo motor mechanical drawing (source: [1]).

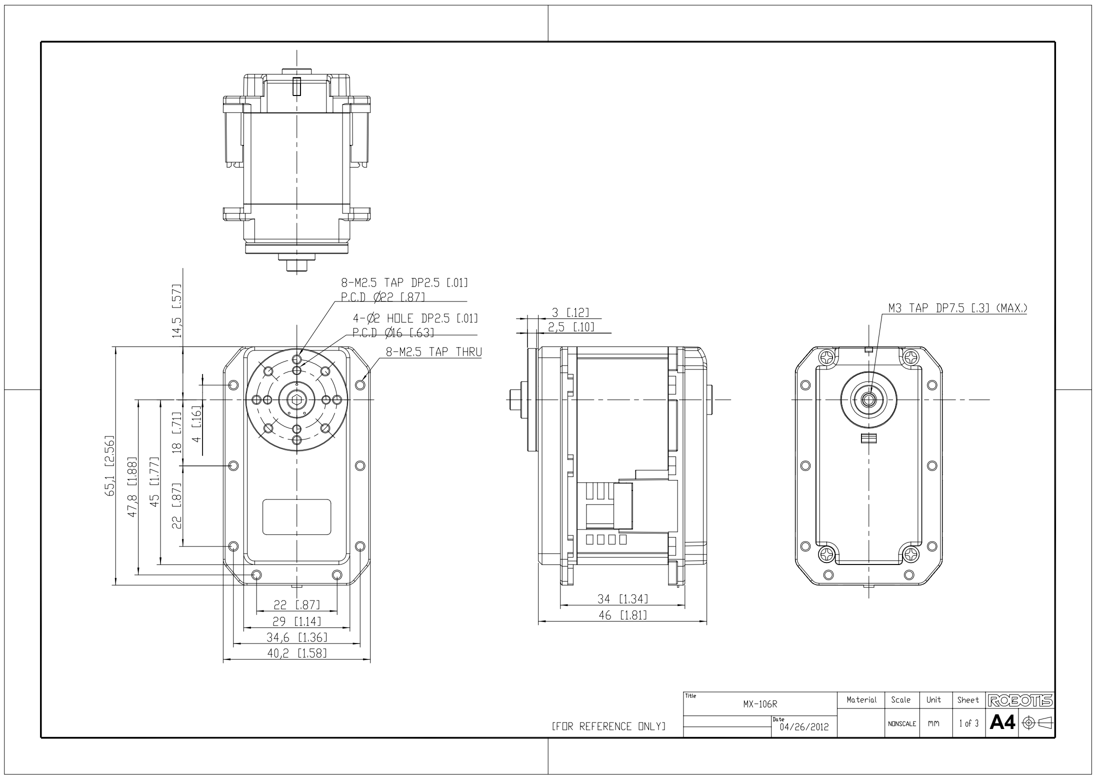


Figure C.3: MX-106R servo motor mechanical drawing (source: [1]).

# BIBLIOGRAPHY

[1] R. INC, "High-performance networked actuators." [Online]. Available: `http://www.robotis.com/xe/dynamixel_en`, 2013.

[2] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. NY, U.S.A.: John Wiley & Sons, Inc., first ed., 2005.

[3] R. INC, "Robotis e-manual v1.14.00." [Online]. Available: `http://support.robotis.com/en/`, 2013.

[4] E. S. Conkur and R. Buckingham, "Clarifying the definition of redundancy as used in robotics," *Robotica*, vol. 15, pp. 583–586, 9 1997.

[5] A. Transeth, N. Van de Wouw, A. Pavlov, J. Hespanha, and K. Pettersen, "Tracking control for snake robot joints," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pp. 3539–3546, 2007.

[6] M. Hannan and I. Walker, "The 'elephant trunk' manipulator, design and implementation," in *Advanced Intelligent Mechatronics, 2001. Proceedings. 2001 IEEE/ASME International Conference on*, vol. 1, pp. 14–19 vol.1, 2001.

[7] D. Cojocaru, M. Ivanescu, R. T. Tanasie, S. Dumitru, and F. Manta, "Experiments with tentacle robots," in *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, pp. 1–6, 2010.

[8] Advantech, “Rs-485, a proud legacy,” tech. rep., Mar. 2012.

[9] B. Lammert, “Practical information about implementing rs485.” [Online]. Available: `http://www.lammertbies.nl/comm/info/RS-485.html`, Aug, 2012.

[10] B. Siciliano, L. Villani, L. Sciavicco, and G. Oriolo, *Robotics Modelling, Planning and Control.* MA, U.S.A.: Springer-Verlag London Limited, second ed., 2009.

[11] J. J. Craig, *Introduction to Robotics, Mechanics and Control.* NJ, U.S.A.: Pearson Education, Inc., third ed., 2005.

[12] J. Denavit and R. S. Hartenberg, “A kinematic notation for lower-pair mechanisms based on matrices,” *Applied Mechanics*, p. 215–221, 1955.

[13] X. Shi and N. Fenton, “A complete and general solution to the forward kinematics problem of platform-type robotic manipulators,” in *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pp. 3055–3062 vol.4, 1994.

[14] C. Yu, M. Jin, and H. Liu, “An analytical solution for inverse kinematic of 7-dof redundant manipulators with offset-wrist,” in *Mechatronics and Automation (ICMA), 2012 International Conference on*, pp. 92–97, 2012.

[15] S. Ma, M. Watanabe, and H. Kondo, “Dynamic control of curve-constrained hyper-redundant manipulators,” in *Computational Intelligence in Robotics and Automation, 2001. Proceedings 2001 IEEE International Symposium on*, pp. 83–88, 2001.

[16] R. Alqasemi and R. Dubey, “Kinematics, control and redundancy resolution of a 9-DOF wheelchair-mounted robotic arm system for ADL tasks,” in *Mechatronics*

*and its Applications, 2009. ISMA '09. 6th International Symposium on*, pp. 1–7, 2009.

[17] R. Ordonez and K. Passino, "Stable multi-input multi-output adaptive fuzzy/neural control," *Fuzzy Systems, IEEE Transactions on*, vol. 7, no. 3, pp. 345–353, 1999.