UNIVERSITY OF CINCINNATI

Date: 20-May-2010

I, Mark P McCrate **,**

hereby submit this original work as part of the requirements for the degree of:

Master of Science

in Mechanical Engineering

It is entitled:

Modern Mechanical Automata

Student Signature: Mark P McCrate

This work and its defense approved by:

Committee Chair: Manish Kumar, PhD
Manish Kumar, PhD

Ernest Hall, PhD
Ernest Hall, PhD

Ronald Huston, PhD
Ronald Huston, PhD

Kelly Cohen, PhD
Kelly Cohen, PhD

Karen Davis, PhD
Karen Davis, PhD

MODERN MECHANICAL AUTOMATA

A thesis submitted to the

Department of Mechanical Engineering

College of Engineering and Applied Science

UNIVERSITY OF CINCINNATI

Division of Research and Advanced Studies
in partial fulfillment of the requirements for the degree of

Master of Science

2010

by

Mark P McCrate

BSME University of Cincinnati 2006

Committee Chair: Manish Kumar, PhD.

Abstract

Robots are becoming evermore ubiquitous. Their design requires combining many engineering specializations. Initially, robots were tele-operated arms. Then they advanced into automated assembly and material handing machines. However, more recent efforts are focused on making companion robots that look and act like humans. This thesis presents an effort to make a Modern Mechanical Automaton capable of drawing pictures that can be used to teach kinematics and dynamics and also for K-12 outreach. These functions are accomplished by adding a touch sensor and microphone to an existing five degree of freedom robot, then writing a fuzzy logic controller to direct movements based on Graphical User Interface input. This work has been demonstrated in many trials. Experience from these trials suggests people of all ages find the user interface easy to understand. Beyond the engineering accomplishments of solving equations, building a fuzzy controller and having a physically realized system, the broader impacts are creating an easily expandable system and successful outreach to bolster the National Science Foundation's Science, Technology, Engineering and Mathematics initiatives.

Acknowledgments

I would like to thank Dr. Manish Kumar for agreeing to be my committee chair. His advice has helped me finish this thesis. I would like to thank my committee members Drs. Ron Huston, Kelly Cohen and especially Karen Davis. They have taught me a great deal and Dr. Davis has been a constant source of inspiration. Thanks also to Dr. Ernest Hall for allowing me work in the robot lab. I have met many good people there over the years.

I need to thank many family and friends for their years of love and support.

I want to thank to Mark Aull and Adam Gerlach and for their help with MATLAB and Dr. Herbert Halpern for his help with *Mathematica*. Many thanks to Hema Kasture for her help with the manuscript. Thanks to Jennifer Baldwin for her ideas related to this work and to Toyota for their support.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

Motivation:

Robots are becoming evermore ubiquitous. Initially, robots were tele-operated arms. Then designs advanced into automated assembly and material handling machines. However, more recent efforts are focused on making companion robots that look and act like humans.

Many humanoid type devices have been developed. The earliest were cloth and bone, were powered by wooden gear trains and served tea, mimed archers or performed other functions. Automata of the late 19th and early 20th centuries were mostly metal, were driven by springs, cams and followers and could pen a few lines of prose or draw a few pictures, at most. Modern humanoid robots are all computer controlled and powered by hydraulic, pneumatic or most often electric elements; elements are sometimes combined in the effort to make machines that look, act and "think" like humans.

MIT is making robots that think like humans (C. Breazeal, 2000; C. Breazeal & Scassellati, 2002; C. Breazeal, 2004). Nexi is one of their newest robot designs. It can read people's expressions and respond appropriately (Caplan, 2008). Toyota's human-like Partner robots can perform acts requiring high levels of coordination such as playing instruments. They have publicly demonstrated trumpeting and playing all four strings of a violin (Kaneko, Harada, Kanehiro, Miyamori, & Akachi, 2008; Kusuda, 2008). Honda produces one of the most advanced humanoid bipeds. ASIMO can walk, run, climb up/down stairs and simultaneously balance a tray of coffee or tea (Sakagami et al., 2002). Creating machines that are gauged on their humanness, are able to compliment our motor skills and are accepted in places beyond the factory floor is quite a mission for any engineer, scientist or psychologist.

Another mission for robot enthusiasts is to create a machine with artistic ability. Artistic ability here refers to the ability to reproduce a work of art – from something as simple as a geometric shape to something as intricate as a famous Henri Matisse or Picasso. During the 2007-08 academic year David Clark, Stephen Collins, Austin Ellis and Scot Watson, all from the former Electrical and Computer Engineering and Computer Science department at the University of Cincinnati, worked to build "The Picasso Printer," a humanoid drawing robot, to fulfill their senior design requirements; this thesis represents a continuation of and a divergence from their

work. Both projects share a common goal, "To make a digital machine that draws pictures, much like the one described in the book The Invention of Hugo Cabret by Brian Selznick" (Clark, Collins, Ellis, & Watson, 2008). The Picasso team also outlined several sub goals, see Table 1-1. Much of their effort was spent building hardware from scratch. They generated wonderful system design diagrams, technology specifications/standards, interface specifications and test plans however before they could realize the goal time expired.

1. Design a control system that uses a microprocessor to control motors that move the arm across the page and lift the pen to draw the picture.
2. Build the machine (control system and physical device) during the Spring Quarter in order to implement and test our design.
3. Test and produce a final working art machine for use as a fundraiser.

Table 1-1 2007-08 Senior Design Subgoals (Clark et al., 2008)**.**

This thesis serves to document a new approach to the Hugo/Picasso robot. Work begins after identifying a robot that meets most of the physical requirements of a drawing robot. Construction and programming details are provided. Results, in the form of equations, pictures and demo data are given as well.

General objective:

Produce a robust robot capable of drawing pictures, that is easy to modify and can be used as a collegiate or K-12 teaching tool.

Specific objectives:

A. Design hardware and an end effecter with the dexterity needed to draw pictures or write.
B. Solve the equations of motion necessary to command end effecter positioning and generalize them based on widely used industrial and academic standards.
C. Write software to make a robot draw both stored pictures and images created via user input.
D. Stabilize all operations by building a flexible fuzzy controller to augment the stepper motor technology.

E. Work with art students to make a costume or other covering to personify the robot thereby making it look more like Hugo Cabret's mechanical man.

Research Methodology:

All specific objectives require completing the set of tasks listed below:

A. Design an end effecter capable of providing feedback while grasping a drawing utensil that is compatible with a Microbot Teachmover, the chosen robot base platform.
B. Using trigonometry, geometry and kinematic decoupling to solve forward and inverse kinematics analytically. Generalize these equations using matrices, Denavit-Hartenberg parameterization and a fuzzy inference system.
C. Write functions to test communication, motion and feedback. Write software that utilizes the analytic equations. Build a simple to understand GUI to benefit/aid user experience/interaction.
D. Tune membership functions and define a rule base for a fuzzy controller.
E. Teach Jennifer Baldwin's art students how to program Microbot Teachmover arms; then, work with them to design a humanoid figure and costume to cover the arm.

Contributions:

Upon successfully completing the tasks listed these contributions are expected:

A. Having robust hardware with enough dexterity to draw. This should open the door to many applications beyond the picture drawing detailed herein.
B. Closed form analytic solutions for the forward and inverse kinematics. These are transformed into standardized matrix notation to provide a good base for future work.
C. A natural user interface providing an experience similar to connect-the-dots.
D. A new control method for Microbot Teachmovers.
E. Successful outreach, especially toward K-12 students, in order to bolster the National Science Foundation's Science, Technology, Engineering and Mathematics (STEM) initiatives.

Overview

Chapter 2 provides an overview of some entertainment robots both from antiquity and modern times. Chapter 3 introduces the hardware structure that will be utilized. In Chapter 4 equations necessary to command a Teachmover are solved. Chapter 5 presents a brief overview of factory Microbot Teachmover capabilities and user interfaces. Chapter 6 introduces the new user interface for picture generation. Chapter 7 gives details of a fuzzy logic controller designed to stabilize the robot's speed. Chapter 8 is a summary of the contributions and some recommended future work.

Chapter 2 Literature Review

Automata are defined as mechanisms that are relatively self-operating (Mish, 1995). In this chapter, several previous generations of humanoid automata are examined. Focus will be on current field robots, their end effectors, feedback/control systems and the task of translating an image acquired through a GUI into a penned image. These topics capture the challenges associated with building a drawing/writing robot.

Early Robots

Shannon Jaeger compiled a timeline listing significant robots (Jaeger, 2005) and (*Robot.*2010) contains a slightly different timeline. They both trace the origin of robotic movement back to the first centuries BC and AD as detailed in (Mansfeld, 1998). Over a millennia later, Japanese Karakuri masters began crafting Karakuri Ninjyo – or person shaped mechanism meant to trick or surprise – from cloth, bone and wood (Boyle, 2008a) (Boyle, 2008c) (Boyle, 2008b) (Law, 1997). These "robots" served tea, Figure 2-1, mimed archers, Figure 2-2, or performed other functions (Kittel, 2008) (shinjiredfield, 2008).

Figure 2-1 Chahakobi Ningyo (Tea Serving Doll) (Boyle, 2008c) **used with permission** (McCrate & Boyle, 2010)

Figure 2-2 Yumihiki Doji' (archer doll). (SHOBEI Tamaya IX, 2008) **used with permission** (McCrate & Boyle, 2010)**.**

Starting in the late 19th and early 20th centuries, automata were made from metal, were driven by springs, cams and followers and could pen a few lines of prose or course line art, Figure 2-3. Gaby Wood has written extensively on these century-old "robots" (Wood, 2002).

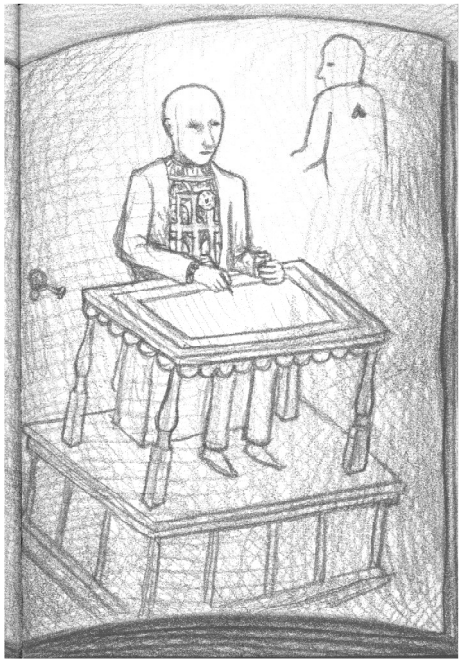


Figure 2-3 Automaton in The Invention of Hugo Cabret.

Current Challenges

Humanoid robot research is branching into emotional, voice and facial detection, recognition and response as well as natural locomotion (C. Breazeal, 2000). Simultaneously, robot researchers are trying to conquer the challenges related to locomotion, drawing, writing and signature replication.

In order to replicate signatures robots need dexterous end effectors. End effectors and other grippers are tools attempting to replicate the dexterity of a human hand. The aforementioned Karakuri tea serving dolls only had a single degree of freedom "platform" end effecter. These days, many end effecter designs are available. Bos catalogs recent efforts in robotic hand design (Bos, 2010). The report shows only four documented cases of grippers exceeding 20DOF. However, a human hand has 25DOF and also provides valuable sensory feedback.

Feedback is the transmission of evaluative or corrective information about an action, event or process to the original controlling source (Mish, 1995). The reaction force in all simple machines is similar to feedback and every electronic sensor can be used for feedback as well. Raymond Goertz patented the first electrical force feedback system (Goertz & Uecker, 1951). Today, end effectors are equipped with force/torque, tactical, proximity and/or many other sensors with resolutions approaching that of a human hand. The best writing and drawing robots utilize force feedback in combination with advanced controllers.

Controllers are built to change the behavior of a physical system while maintaining stability. There are many controller types ranging from error reducing Proportional- Integral-Derivative to advanced neuro-fuzzy logic controllers. A particular family of controllers, known as "compliance" or "accommodation" controllers, is specifically designed for applications where an end effecter must physically contact the environment (Mason, 1981), (Whitney, 1977). Adibhatla explains compliance controller applications span from on-orbit rendezvous to automobile body welding (Adibhatla, 2007). Wang et al. distinguishes between active and passive compliance (W. Wang, Loh, & Gu, 1998). The demarcation they propose is:

> *Active compliance* is controlling individual joint-servo stiffness by directly controlling joint torque.
>
> *Passive compliance* is the intrinsic mechanical structural compliance due to the finite stiffness of the robot base, links, joint drive mechanism, grippers as well as the assembled parts.

In this work, a spring-loaded "touch sensor" pen and approach function together form a passively compliant system.

Hogan introduces impedance control by further refining accommodation control. Impedance control and techniques for control of manipulator behavior result in a unified approach to kinematically constrained motion, dynamic interaction, target acquisition and obstacle avoidance (Hogan, 1985a; Hogan, 1985b; Hogan, 1985c). Adibhatla neatly summarizes Hogan's three-part paper on the control of dynamic interaction between a manipulator and its environment by writing: In part I, [Hogan] presents the theory of the mechanics of interaction. In part II, he describes impedance techniques for computing the relationship between external forces and displacements or velocities of a tool tip. In part III, Hogan presents an example to illustrate the usefulness of impedance control. Spong et al. covers compliance, impedance and

other techniques for controlling joints individually in Robot Modeling and Control (Spong, 2006).

When a sophisticated end effecter is instrumented with feedback sensors attached to a properly tuned controller, the results can appear very life-like (Goldsmith & Worsdall, 2010) (Eastern Cranial Affiliates LLC, 2009) even when operated remotely (Page, 2007). When the goal is to copy an image or signature, the forward and inverse kinematics must also be solved. Chapter 4 covers the derivation of inverse kinematics for the Microbot Teachmover and Appendix A covers solving inverse kinematics in general. The next chapter starts to describe a new attempt at a drawing/writing robot.

Chapter 3 Hardware Description

While many robot simulation environments exist, this work involves hardware because the objective is to actually draw a physical picture. The first group at the University of Cincinnati to attempt this project assumed commercial hardware did not exist that could meet the required demands. Therefore, a major part of their effort was to design/build/test a custom robot. Since that time a commercial option has been found. The present work utilizes Microbot Teachmovers as the foundation on which locomotion and picture drawing is based.

This chapter will focus entirely on hardware. A very detailed hardware description exists in the form of a user manual so this chapter will highlight key areas of the design and all modifications necessary to ready the Microbot for drawing. After reading this chapter, experimentalists should feel comfortable enough to modify and/or repair the hardware.

Teachmovers are cleverly constructed. Mechanically they are just a few pulleys and some aircraft cable; electrically, stepper motors spin pulleys once commanded by a Motorola 6502 microprocessor. Out of the box, Teachmovers, Figure 3-1, – or they will sometimes be referred to as, robots, arms, Microbots or occasionally Auto – look like small Unimation PUMA 560s. In fact both 560s and Teachmovers are 5DOF arms and both use two fingers as the end effecter. On Teachmovers a four-bar mechanism ensures the fingers remain parallel upon approach in order to maximize gripping area. Small rubber pads glued to the fingers also improve gripping ability. Above the gripper is a differential wrist joint. Using this gearing scheme for the wrist makes pitching and rolling the fingers very easy though it requires coordinating the L and R wrist gears. Between the wrist and elbow joint several sets of thin aircraft cables are encased inside the sheet metal body. Constructing the body out of sheet metal saves on cost and weight. Between the elbow and shoulder joint there is more cabling and the only feedback sensor. The sensor is just a microswitch that acts as a tension indicator. Another clever and interesting trait is that all motors and gear trains are mounted within the body. By moving all motors into the body individual link load is lessened and lighter construction is possible; also, balancing is easier and the moment of inertia is lessened since the second term of the parallel axis theorem, $I = I_{cm} + mr^2$, is reduced as r is reduced. A modern example of a "wire-operated system" with internally mounted actuators is demonstrated by the fourth Toyota Partner

Robot (Toyota Motor Corporation, 2003). One disadvantage of cable driven systems is they typically have a limited payload capacity.

Since all the arm's joints are revolute it is very important to keep cables properly tensioned. Loss of proper adjustment is most commonly caused by one of two events: when a user crashes the arm into a stationary object or by the cables tending to lengthen over time. Fortunately, the engineers were perspicacious and built in user accessible tuning knobs.

Figure 3-1 Robot arm.

The black rectangular box underneath the body is the Teachmover's base, where the power and Teach Control cables are located. Two DB-25 serial interface connectors provide a means of communicating via a computer. Having two connectors gives users the ability to daisy chain Microbots. Inside the base is where the motherboard is located, Figure 3-2.

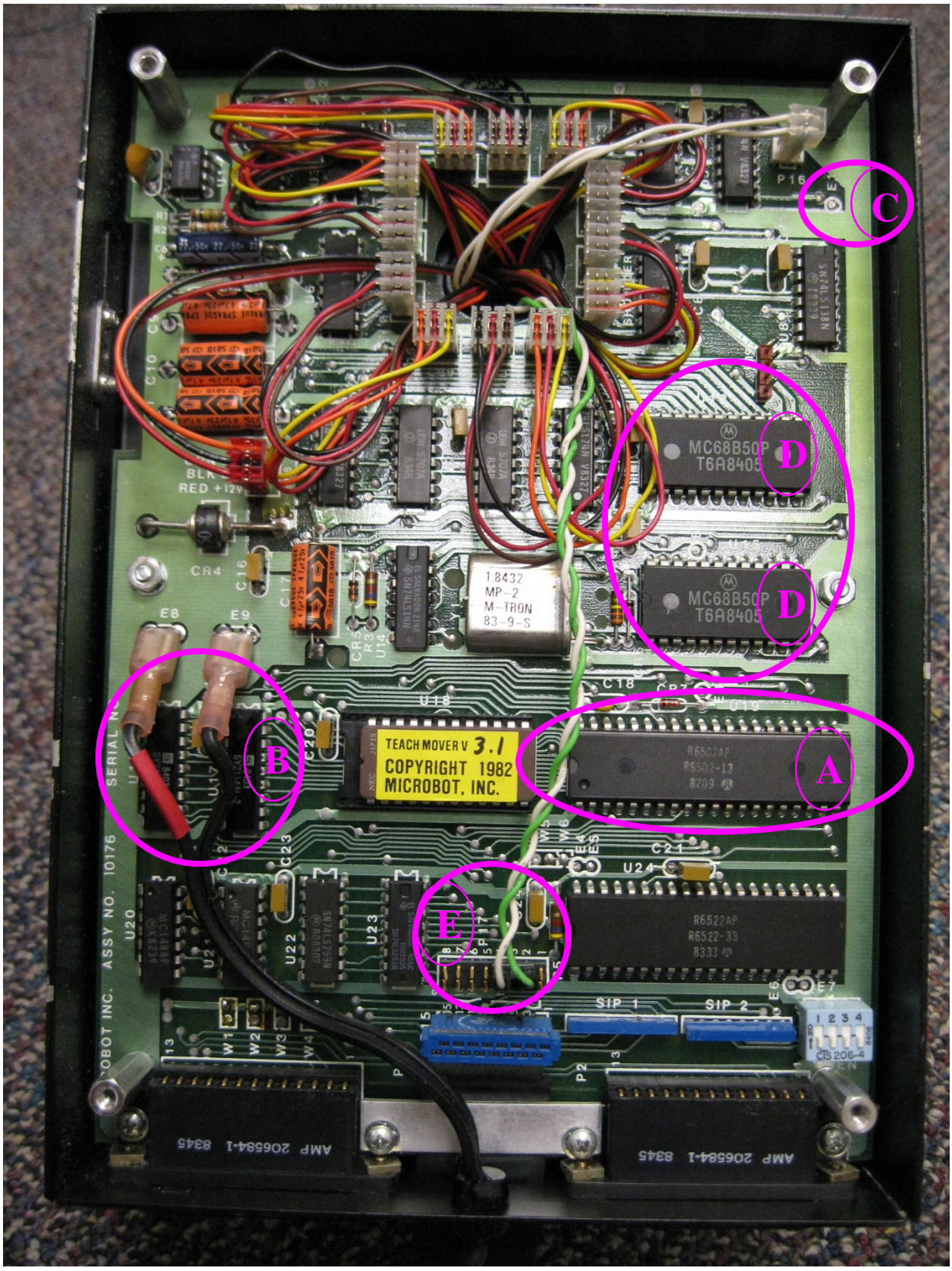


Figure 3-2 Motherboard, where points of interest are highlighted.

The biggest chip, on the motherboard, Figure 3-2, is the micro controller (A): an 8-bit Motorola 6502 common in such devices as the original Nintendo NES, Commodore 64, several Atari models and all Apple][computers. Aiding processing is external RAM (B). The manual describes an adventurous memory hack that can double system memory if needed (C shows memory mod addresser pin). Teachmovers are serial communication ready. Two chips enable this RS-232 serial communication (D). Enhancing the platform even more is a set of built-in I/O pins (E). Using these I/O pins, users can connect up to seven additional sensors and output up to five bits of information. Here green and white wires connect to the "touch sensor." The touch sensor is a specially built spring-loaded pen, Figure 3-3, which runs simultaneously with the approach function that is described in Chapter 6.

This succinct chapter introduces readers to the hardware platform. The information herein provides enough detail that someone should be able to care for the hardware. Using this chapter as a starting point one can modify Teachmovers and greatly increase functionality beyond that which is described within these pages. In the next chapter, equations necessary to effectively position the end effecter are derived and formatted such that they can be easily programmed.

Figure 3-3 Pen with embedded touch sensor/limit switch.

Chapter 4 Special Kinematics

Understanding kinematics is a necessary condition for commanding a robot in an intelligent and useful way. Kinematics is formally defined in Appendix A on Generalized Kinematics. In the context of robotics, forward kinematics is defined as a process to determine where in space a robot's end effecter is given the robot configuration. Inverse kinematics can then be defined as deriving the robot configuration based on where, in space, the end effecter is.

In this chapter, forward and inverse kinematics for the Microbot Teachmover Arm are derived. Symbols used to describe joints, links and angles are based on those found in original Microbot Teachmover manual.

In addition to notation, defining a coordinate system is necessary. Many coordinate systems exist, the most common coordinate systems include: Cartesian, polar, cylindrical and spherical. In this derivation two coordinate systems are used: Joint Coordinates and Cartesian. Joint Coordinates are simply joint angles and include base, shoulder, elbow, pitch and roll. The gripper is controlled by a motor but is not a positional DOF. Cartesian Coordinates are defined by Microbot's Initialization and Calibration Grid, Figure 4-1 and include X, Y, Z, Pitch (P) and Roll (R). Using Figure 4-1, the direction for Z is defined by the right-hand rule. All of the arm's joints are revolute, therefore it is known as an articulated, revolute, elbow or anthropomorphic manipulator (Spong, 2006).

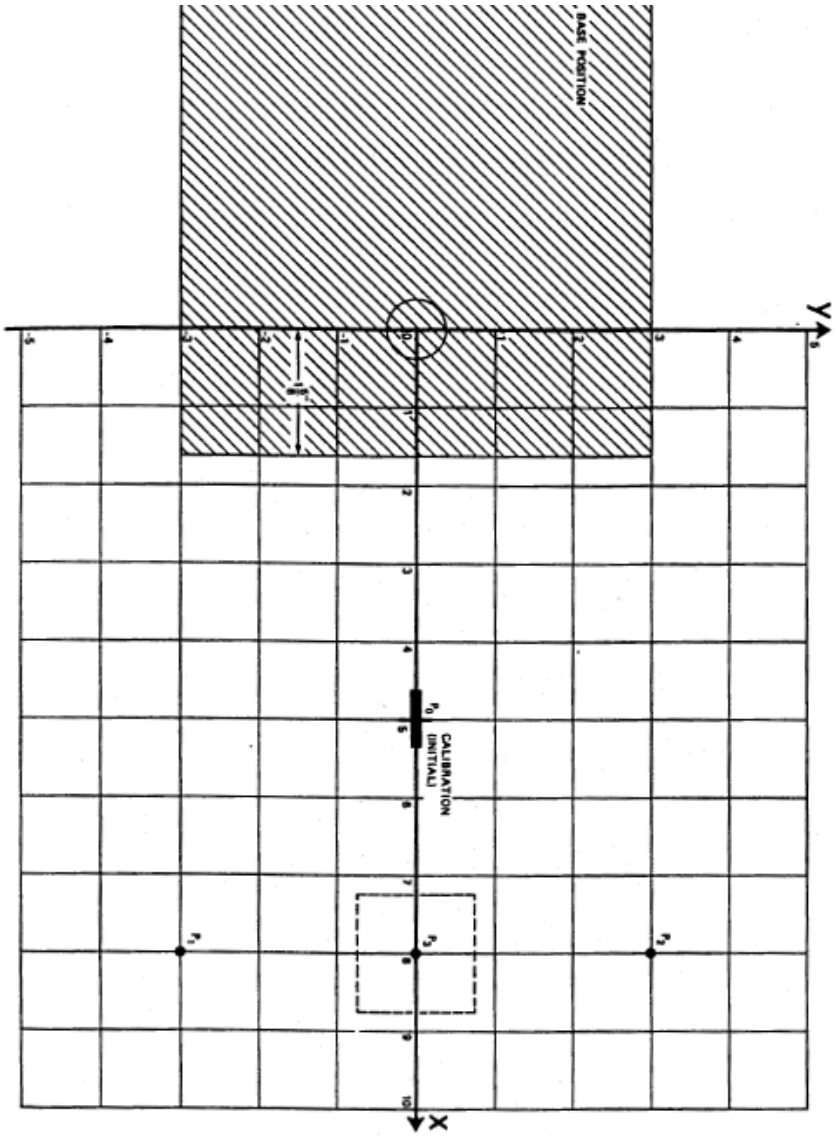


Figure 4-1 Initialization and Calibration Grid from Microbot.

The relationship between different parts of the arm must be specified. In the following explanation all specific information, concerning how each joint is articulated, how the joint angles are measured and the distances between joints, comes directly from the manual.

The Greek letter theta, θ, is used to indicate joint angles. All θs, measured in degrees, radians or revolutions, are proportional to motor steps, J, as shown in Table 4-1 (motor steps are sent via computer commands). Two procedures ensure accurate conversions: 1) RESET the position registers at an initial calibration position where all angles, θ, are zero 2) avoid crashing into hard stops.

Motor	Joint	Steps/Revolution	Steps per Radian	Steps per Degree
1	Base	7072	1125	19.64
2	Shoulder	7072	1125	19.64
3	Elbow	4158	661.2	11.55
4	Right Wrist	1536	241	4.27
5	Left Wrist	1536	241	4.27

Table 4-1 Conversion factors between motor steps and revolute joint angles.

Formulae to convert steps to angles are as follows. These account for the couple joints as well.

$$\text{Base} = \theta_1 = J_1 \qquad \textbf{Eqn 1}$$

$$\text{Shoulder} = \theta_2 = -J_2 \qquad \textbf{Eqn 2}$$

$$\text{Elbow} = \theta_3 = -J_3 \qquad \textbf{Eqn 3}$$

$$\text{Pitch, P} = -0.5(J_5 + J_4) = -0.5(\theta_5 + \theta_4) \qquad \textbf{Eqn 4}$$

$$\text{Roll, R} = 0.5(J_5 - J_4) = 0.5(\theta_5 - \theta_4) \qquad \textbf{Eqn 5}$$

Where J is an internal position register value. Negative signs, while absent in the manual, maintain the sign convention established in Figure 4-2 and Figure 4-3. Figure 4-3 shows the definition of pitch and roll.

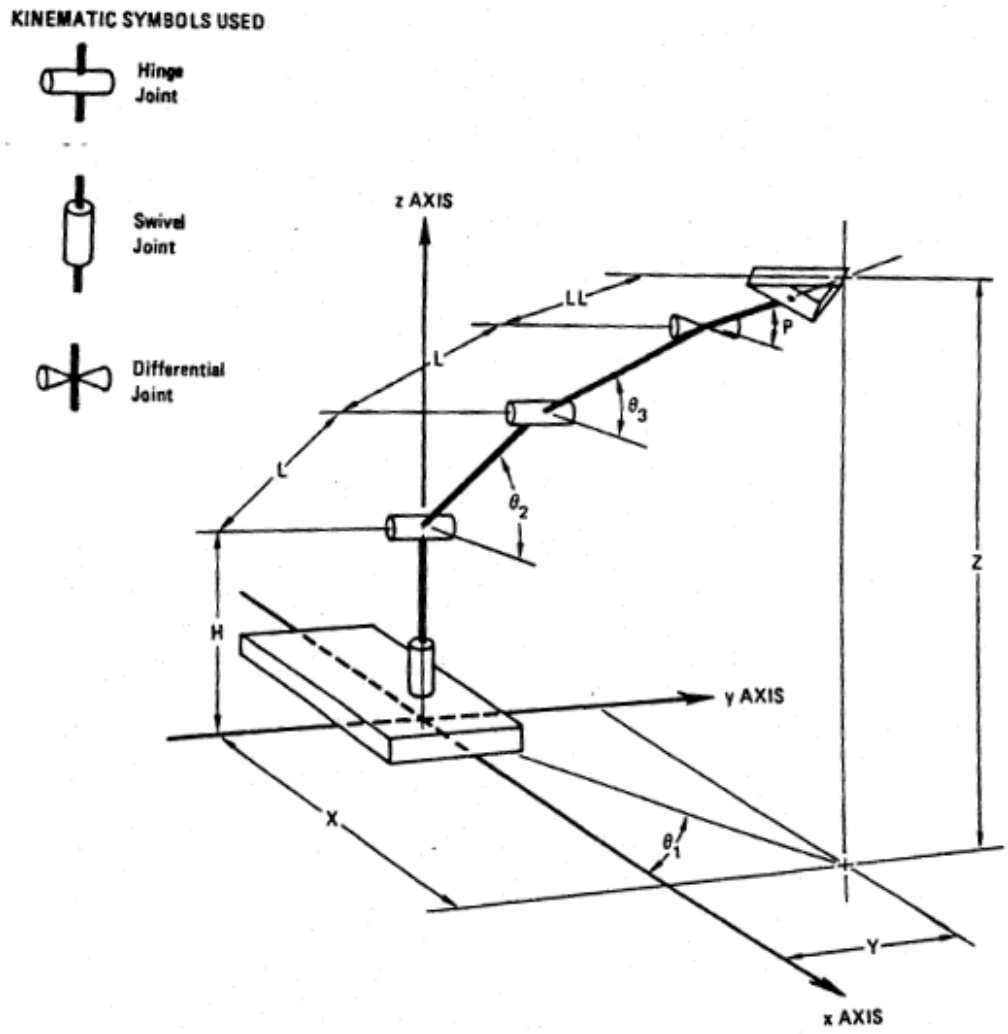


Figure 4-2 (above) Microbot supplied Kinematic Model of the Teachmover Arm.

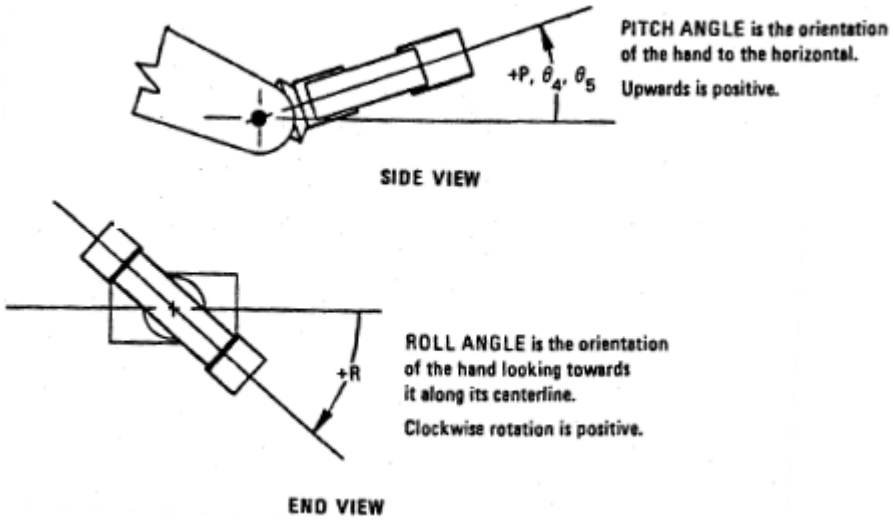


Figure 4-3 Microbot definition of pitch and roll angles.

Link lengths, distance between each joint, are indicated by the constants H, L and LL as shown in Figure 4-2 and Table 4-2. H is the distance from the table top to shoulder joint centerline; L is the distance from the shoulder joint to elbow and elbow to wrist; LL is the distance from the wrist to finger center, when the fingers are separated by 1.5 inches or are roughly parallel.

Segment	Length (inch)	Length (mm)
H	7.68	195.0
L	7.00	177.8
LL	3.80	96.5

Table 4-2 Lengths of Teachmover Arm members.

Forward Kinematics

Forward kinematics implies determining X, Y, Z, pitch and roll of the end point given joint angles $\theta_{1 \rightarrow 5}$. The end point is the center point between the two fingers and will hence forth be the location of interest. Solving the forward kinematics is easy once all physical relationships are defined. Start the forward solution by finding the height of the end point above the table, Z, a visual aid is provided in Figure 4-4.

$$Z = H + L\sin\theta_2 + L\sin\theta_3 + LL\sin(P) \qquad \textbf{Eqn 6}$$

Next, find the horizontal distance of the end point from the base, defined as the intermediate value, RR. Again, a visual aid is provided in Figure 4-4 and Figure 4-5.

$$RR = L\cos\theta_2 + L\cos\theta_3 + LL\cos(P) \qquad \textbf{Eqn 7}$$

Finally, calculate X and Y using the intermediate value RR.

$$X = RR\cos\theta_1 \qquad \textbf{Eqn 8}$$

$$Y = RR\sin\theta_1 \qquad \textbf{Eqn 9}$$

P and R are defined in Eqn 4 and Eqn 5.

This concludes the forward kinematic solution derivation. It is based on basic geometry and trigonometry. All notation is consistent with the manual. Eqn 1 → Eqn 9 in this thesis are similar to Eqn 1 → Eqn 7 in the manual, however, in the next section, on inverse kinematics, the derivation differs markedly.

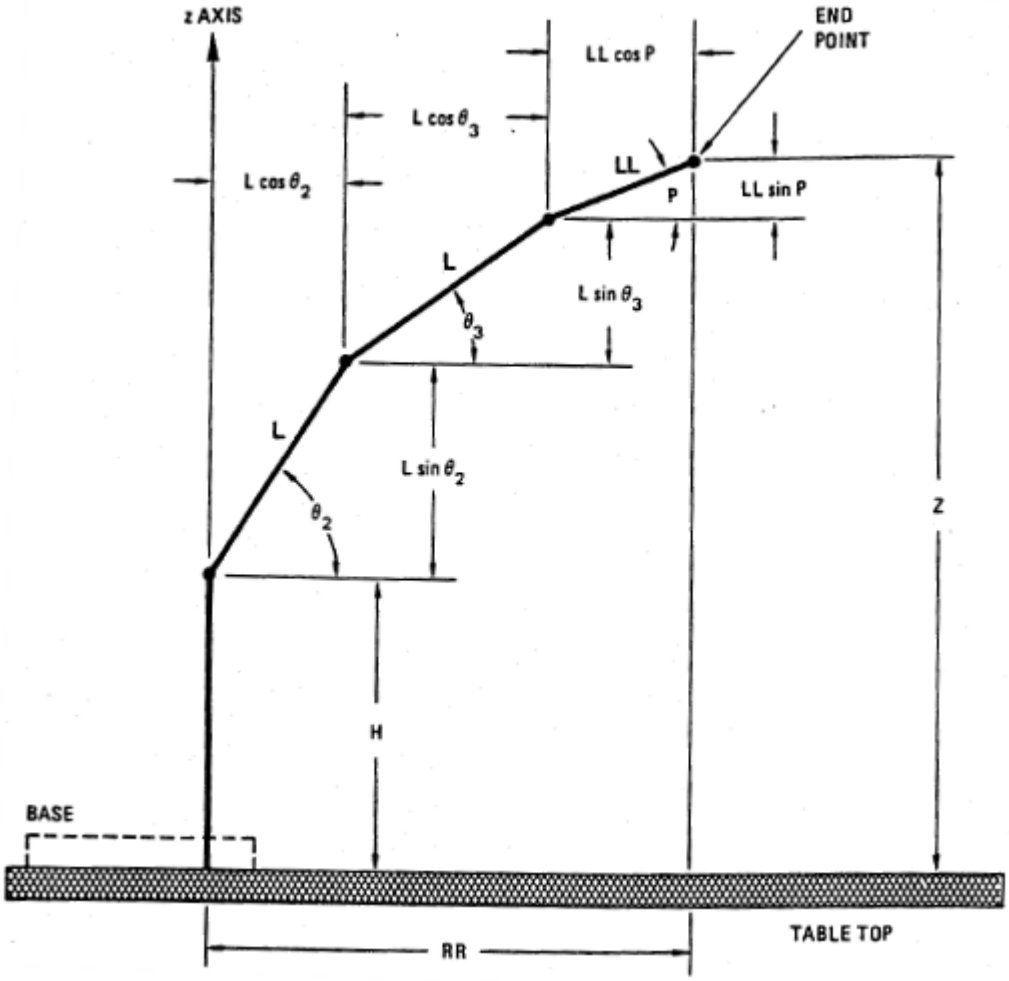


Figure 4-4 (above) Microbot supplied side view of kinematic model.

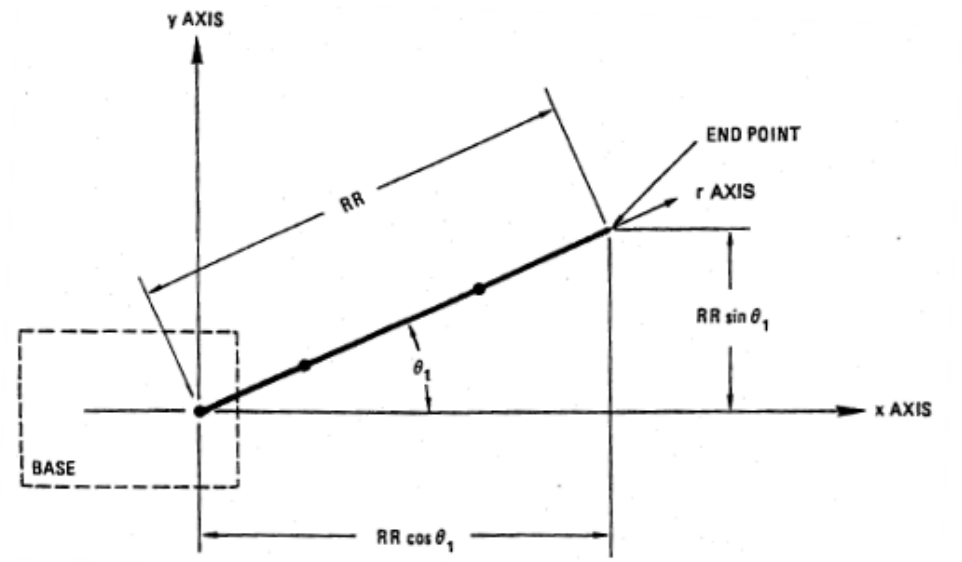


Figure 4-5 Microbot supplied top view of kinematic model.

Inverse Kinematics

One definition for inverse kinematics is – the process of determining the parameters of a jointed flexible object (a kinematic chain) in order to achieve a desired pose (*Inverse kinematics.*2010). This section will demonstrate one method for calculating the inverse kinematics. However, before beginning, a few more physical relations need to be defined.

"In practice it is difficult to distinguish between positive and negative roll angles (i.e. ±90º) by looking at the hand" according to Appendix D page 12 (Microbot, 1984). To eliminate ambiguity, the direction of the roll vector is marked with a bull's-eye, according to the convention commonly used to indicate a vector emanating from a surface. Figure 4-6 shows the bull's eye, on top of the hand, which is visible when the wrist is oriented at 0º. 0º corresponds to the orientation when the wrist cable turnbuckles are aligned, Figure 4-7.

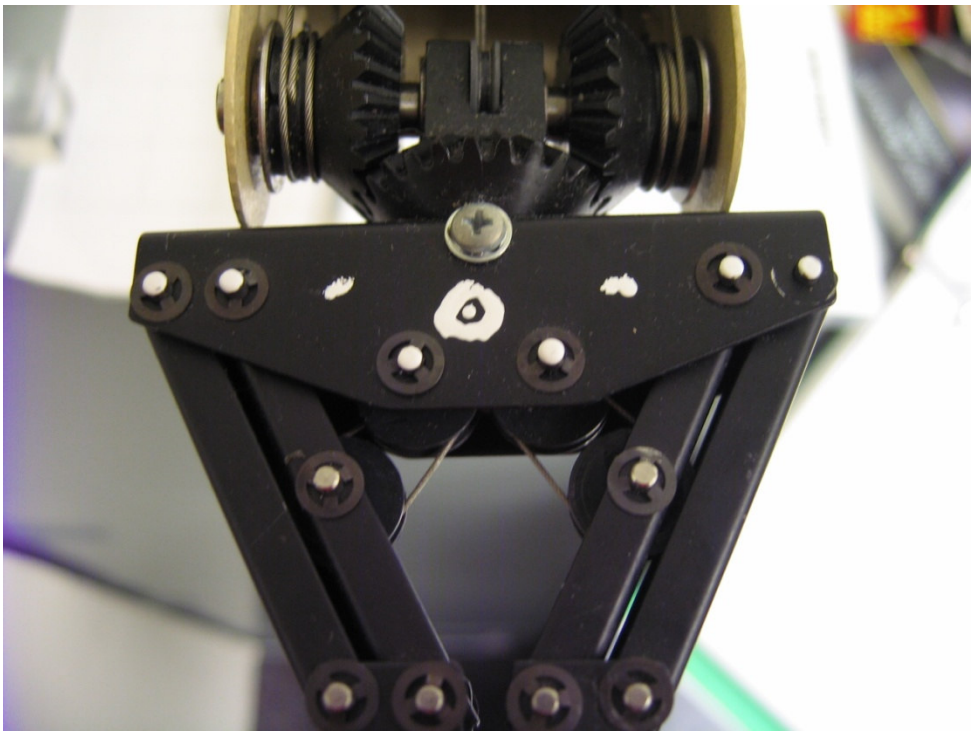


Figure 4-6 (above) Roll vector identifying wrist/hand at 0º.

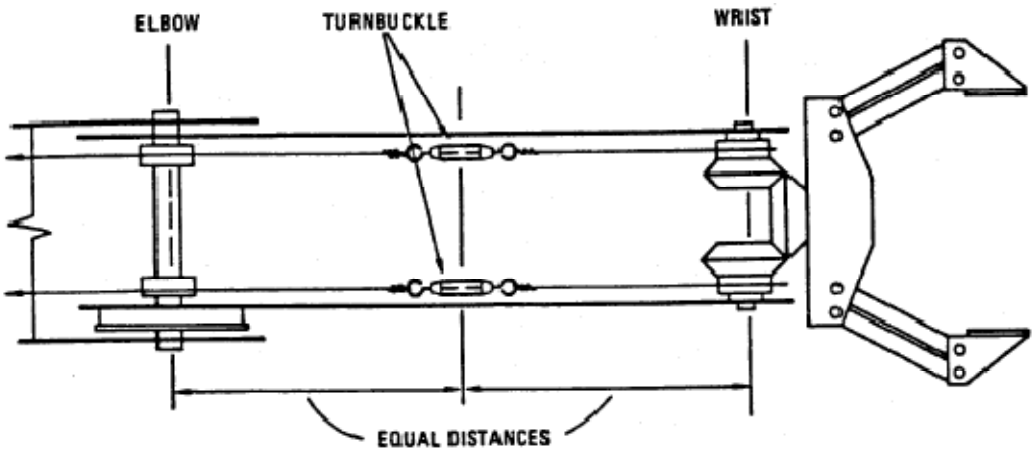


Figure 4-7 Microbot supplied images showing turnbuckle orientation at 0º.

Certain applications, such as playing games, drawing, sliding objects, assembling structures or removing a peg from a hole, require precise and often linear movements. In these cases it is "sometimes [...] useful to express "roll" with respect to [the] Cartesian frame rather than with respect to the arm" as suggested in Appendix D page 14 (Microbot, 1984). The manual suggests setting P = –90º (hand pointed down) as a reference position and measuring "Cartesian roll" with respect to the x-axis, as is usual. The Cartesian roll convention used here

differs from that described in the manual, the current convention is shown in Figure 4-8. In the figure: pitch = –90º, roll in Cartesian frame ≡ R' and roll with respect to the arm ≡ R.

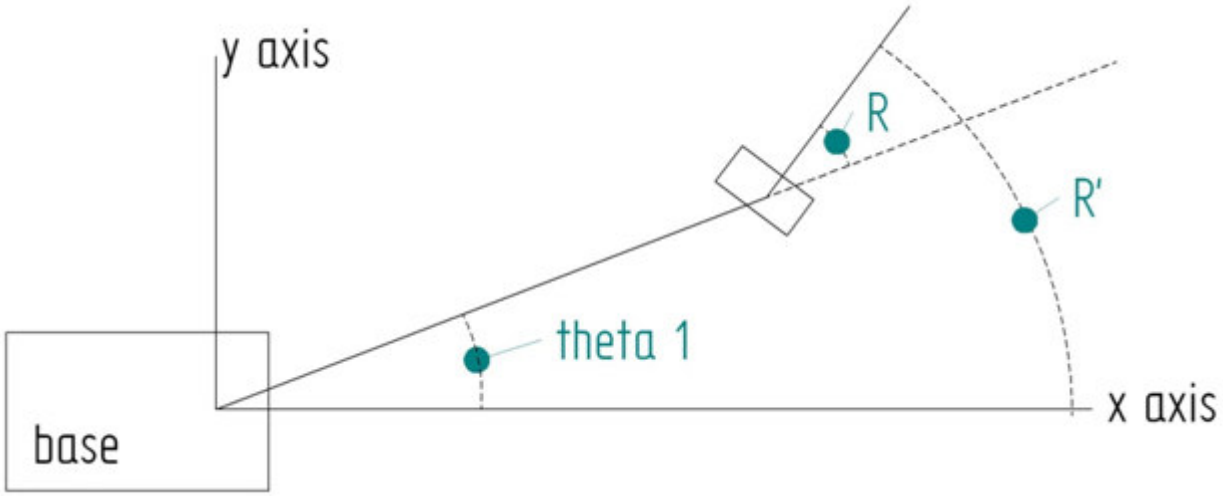


Figure 4-8 New top view of the arm.

Writing a single roll equation, valid for both the arm and Cartesian frames, is made possible by introducing "special" variable R1 in Eqn 10.

$$\text{roll} = R + R1 * \theta_1 \qquad \textbf{Eqn 10}$$

Where R1 = 1 if roll is with respect to Cartesian frame
R1 = 0 if roll is with respect to the arm frame

When R1 = 1 the Cartesian roll reported by Eqn 10 is visualized by projecting the vector in Figure 4-6 into the X-Y plane and measuring X to Y as is usual. Unfortunately, at high pitch and roll angles the simplicity of Eqn 10 breaks down.

These new physical relations aid in solving the inverse kinematics. Solving the inverse kinematics is determining joint angles $\theta_{1\rightarrow5}$ given X, Y, Z, pitch and roll of the end point.

Start the inverse solution by finding the base angle, θ_1 and radius vector, RR, Figure 4-9.

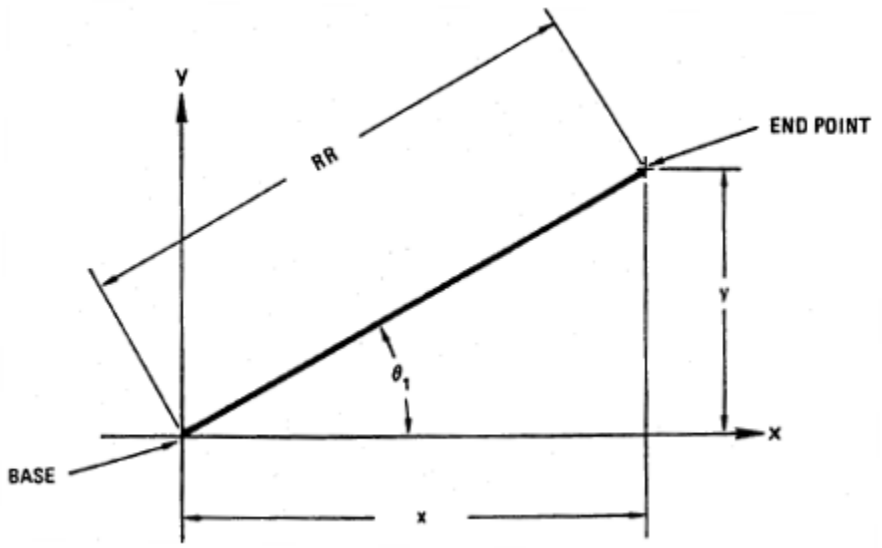


Figure 4-9 Microbot supplied top view of arm.

$$RR = \sqrt{x^2 + y^2} \qquad \textbf{Eqn 11}$$

$$\theta_1 = \tan^{-1}\left(\frac{y}{x}\right) \qquad \textbf{Eqn 12}$$

Next, find θ_4 and θ_5 from P and R, pitch and roll, respectively by using Eqn 4 and Eqn 5. Then substituting in Eqn 10 makes:

$$\theta_4 = -P - R + R1 * \theta_1 \qquad \textbf{Eqn 13}$$

$$\theta_5 = -P + R - R1 * \theta_1 \qquad \textbf{Eqn 14}$$

Then work back from the coordinates of the end point to those of the wrist. Letting R_e and Z_e be the end point coordinates, calculate the wrist coordinates, R_w and Z_w respectively, using Eqn 15 and Eqn 16 or Figure 4-10. Distances in Figure 4-10 are measured vertically along the z-axis and horizontally along the radius from the base (r-axis).

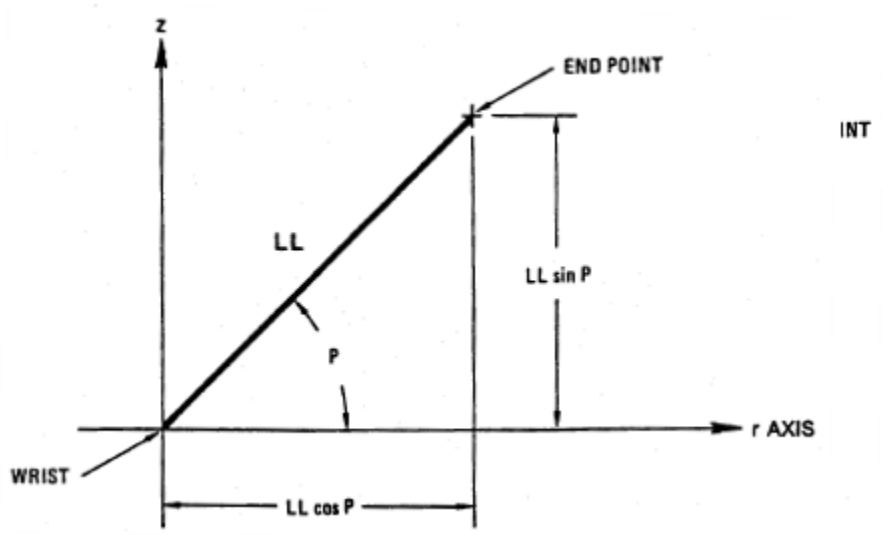


Figure 4-10 Microbot supplied side view of hand triangle used to find R_w and Z_w.

$$R_w = R_e - LL \cos P \qquad \text{Eqn 15}$$

$$Z_w = Z_e - LL \sin P \qquad \text{Eqn 16}$$

Finally, find θ_2 and θ_3, the shoulder and elbow angles, respectively. Steps for this differ markedly from the manual's derivation. Many new variables are defined to aid the derivation. A single picture, Figure 4-11, explains the new variables and how to obtain answers graphically.

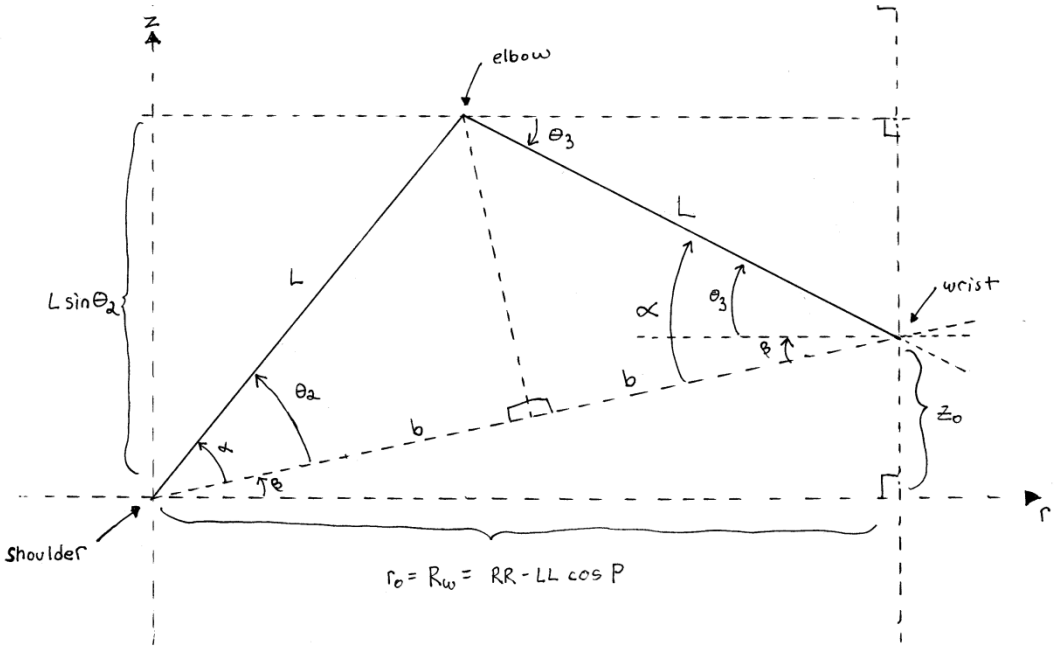


Figure 4-11 Shoulder-Elbow-Wrist triangle.

New variables introduced in Figure 4-11 include:

r_0 – distance from the shoulder to wrist, same as r_w

z_0 – height of wrist above the shoulder, Z_w – H

b – base of congruent right triangles formed by bisecting the isosceles shoulder-elbow-wrist triangle

α – one angle of the congruent right triangles with shoulder-elbow as hypotenuse

β – angle of the shoulder-elbow-wrist isosceles triangle above the horizontal

Using these new variables solve for θ_2 and θ_3 as follows. Step one, solve for b noting it is part of the hypotenuse of a right triangle.

$$b = \frac{\sqrt{{r_0}^2 + {z_0}^2}}{2} \qquad \textbf{Eqn 17}$$

Step two, solve for β noting it is part of the same right triangle as b.

$$\beta = \tan^{-1}\left(\frac{z_0}{r_0}\right) \qquad \textbf{Eqn 18}$$

Step three, solve for α, having solved for b and knowing l, a.k.a. L from Table 4-2.

$$\alpha = \cos^{-1}\left(\frac{b}{L}\right) \qquad \textbf{Eqn 19}$$

Careful inspection of Figure 4-11 shows θ_2 and θ_3 are composed of α and β

$$\theta_2 = \alpha + \beta \qquad \textbf{Eqn 20}$$

$$\theta_3 = \alpha - \beta \qquad \textbf{Eqn 21}$$

Note that θ_3 was previously defined as the angle of the elbow above the horizontal hence the sign must be changed when coding.

Table 4-3 contains a summary of this derivation. Solving the inverse kinematics this way is simple if not general. The next section covers one additional consideration used when doing high precision work.

Θ_1	$\theta_1 = \tan^{-1}\left(\frac{y}{x}\right)$
Θ_2	$\theta_2 = \alpha + \beta$
Θ_3	$\theta_3 = \alpha - \beta$
Θ_4	$\theta_4 = -p - r + r1 * \theta_1$
Θ_5	$\theta_5 = -p + r - r1 * \theta_1$

Table 4-3 Summary of θ calculations.

Further Consideration

Hand length, LL, actually varies slightly with hand opening, see Figure 4-6, so the value given in Table 4-2 is not a constant. Though the variation is small, it may be critical. "The hand length, LL, may be expressed as the sum of a fixed length, L_1, and a varying length that depends on hand opening, G" according to the manual Appendix D page 26 (Microbot, 1984).

$$LL = L_1 + \sqrt{{L_2}^2 + \frac{(G - G_0)^2}{5}} \qquad \textbf{Eqn 22}$$

Where hand opening, G, may be converted to motor steps and vice-versa by using proportionality constant: 371 steps/inch or (14.6 steps/mm).

Constant	Empirical (inch)	Metric (mm)
L_1	2.097	53.3
L_2	1.7	43.2
G_0	1.52	38.6

Table 4-4 Constants of the varying hand length equation.

In Eqn 22 and Table 4-4 the value of L_1 and the denominator of 5 are different than those in the manual. Some adjustments are needed to ensure maximum possible reach and realistic length variation. The geometry that produced the variation is shown in Figure 4-12. Results of the heuristically arrived at modified values are shown in Figure 4-13.

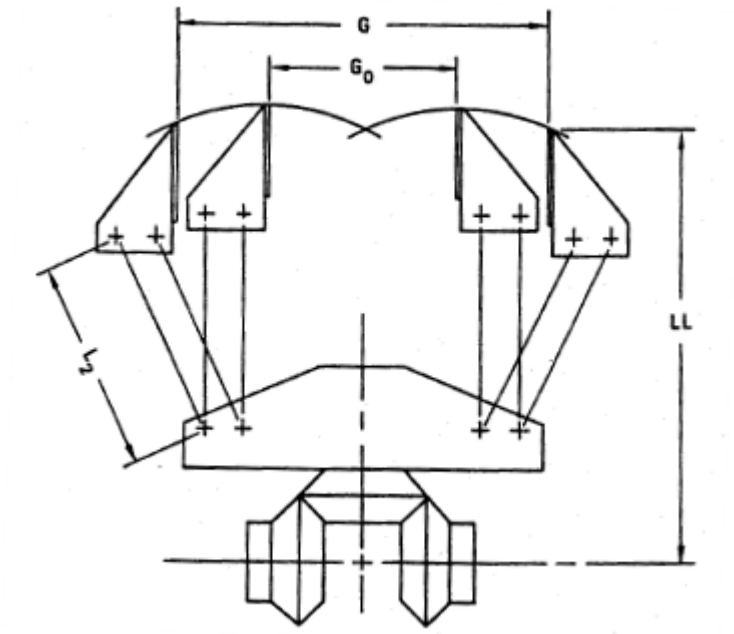


Figure 4-12 Microbot picture of the geometry that produces a variable hand length.

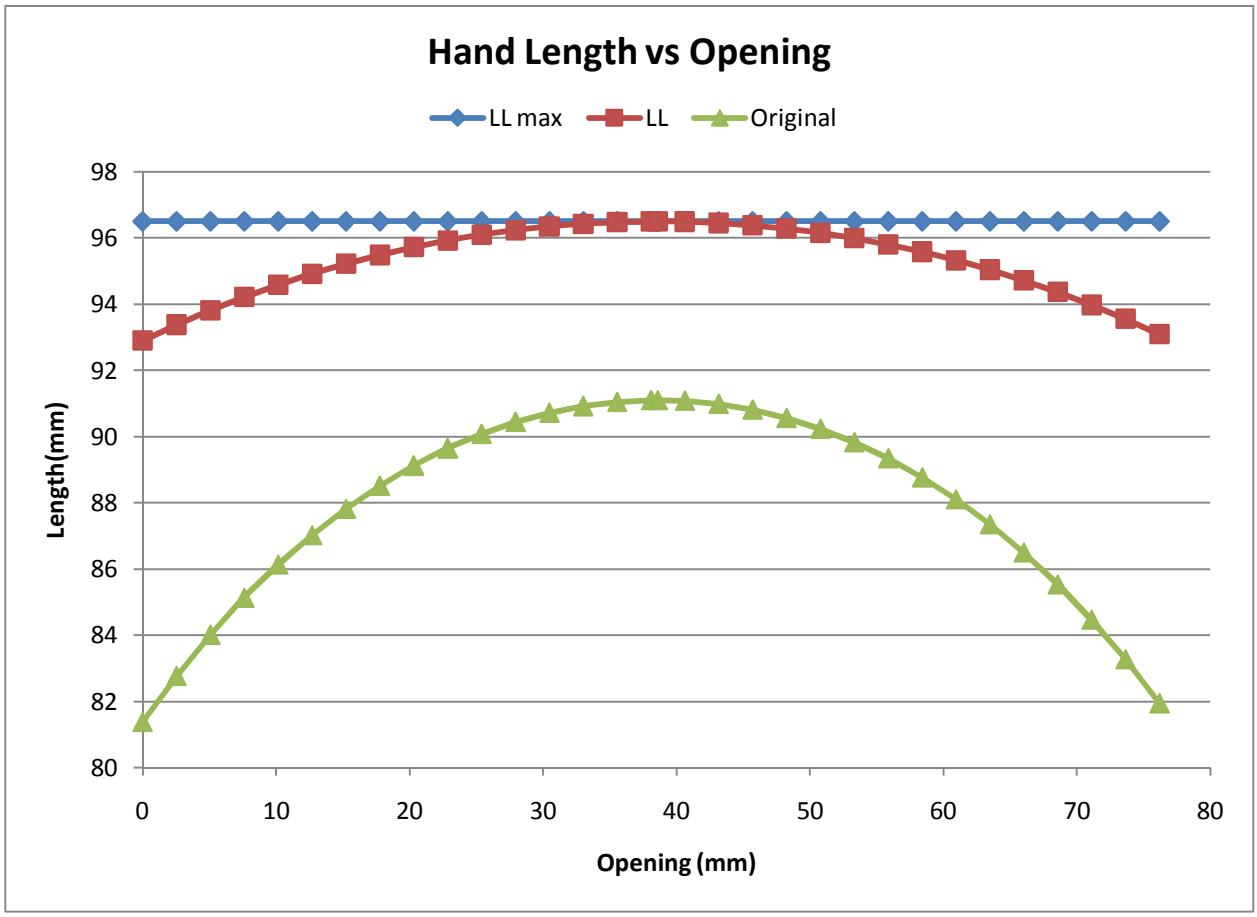


Figure 4-13 Hand Length vs Opening

Table 4-3 contains all the equations necessary for inverse kinematics. The derivation is based on the principles of kinematic decoupling covered in (Spong, 2006), simplified in the Teachmover manual and further simplified by the author. Figures and explanations have been included to aid the readers understanding. In Appendix A, these equations are derived using matrix math and formatted using Denavit-Hartenberg parameterization, which is understood by a large group of mechanical engineers. The communication protocols and factory software will be dealt with in the next chapter.

Chapter 5 Communication Protocols and Factory Software

This chapter provides background into a Microbot Teachmover's communication protocol and factory software. It touches on all the commands necessary to interface a basic drawing program.

Generally, each robot company creates their own robot communication protocol or language; for example industrial robot manufactures such as Epson, Fanuc, Kuka and Panasonic have their proprietary control software. Machine tool makers such as Fanuc and Siemens have packages that work well across many physical platforms but are often black boxes, which cannot talk to each other. Printers are much closer to universality. Manipulating paper, toner and ink in most modern printers is the direct result of obeying industry standard software definitions such as post-script (PS) or printer-command-language (PCL). Teachmovers come standard with two protocols: one that responds to Teach Control buttons presses, and another for RS-232 serial linking. After finishing this chapter, users will be familiar with both.

Teach Control

Most users interface with Teachmovers using the hand-held Teach Control, as shown in Figure 5-1. Users simply press buttons to move the joints others activate "mode" to enable the control functions and then write simple programs. Available control functions are listed and described in Table 5-1.

Figure 5-1 Handheld teach control. (Almost actual size.)

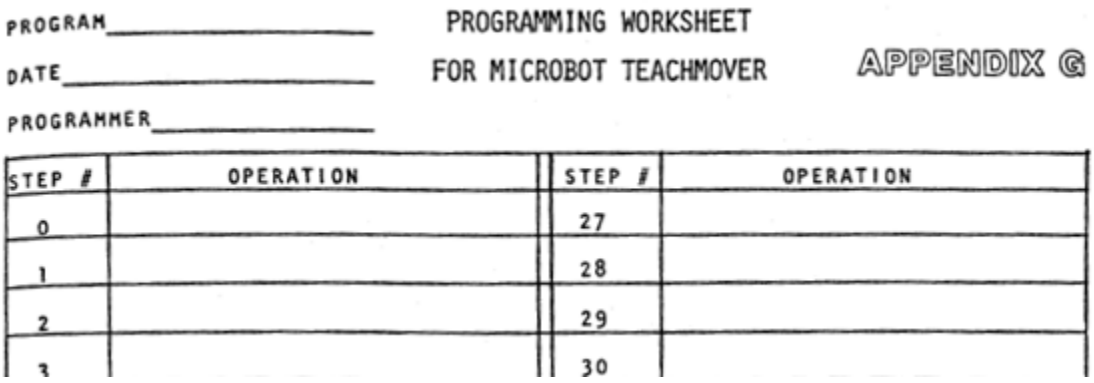
COMMAND	FUNCTION	SYNTAX/DETAILS
CLEAR	Erased entire program	To activate, hold down MODE key & press CLEAR.
FREE	Turns off all motor currents.	Enables manual positioning. No program steps are created.
GRIP	Closes the gripper.	Builds up 1 pound of grip force. Moves the hand motor 32 half-steps past the point where grip switch closes.
JUMP* * Should not be used.	Conditional (or unconditional) branching.	Two numerical entries (press MODE key b/w): 1st entry-jump condition: 0: grip switch open 1 → 7: user input bit 8: never OR 9: always 2nd entry: step to jump to.
MOVE	Activates joint-control (arm motion) keys.	Like train only does NOT: change internal position registers; allow positions to be recorded; make program steps.
MODE	Stops arm.	Exits: TRAIN, MOVE and ENTER modes.
OUT	Designates which output bit to turn on.	Two numerical entries (press MODE key b/w): 1st entry: 0: MODE light 1 → 5 user output bits 6: TRAIN light 7: RUN light 8: ENTER light 2nd entry: 0 or 1 for lights off or on, respectively.
PAUSE	Pauses arm for # seconds	Numerical entry 0 → 255.
POINT	Sets program pointer to # step	Numerical entry.
RUN	Runs current program.	If running, stops at end of current step.
SPEED	Sets speed of arm motion.	0 (slowest) to 15 (fastest).
STEP	Do current prgm, step by step.	Moves arm to next position.
TRAIN	Activates joint-control (arm motion) keys.	Press REC for ea position to be saved. REC overwrites current step and increments pointer. Power cycle = train, pointer and internal position registers = 0.
ZERO	Zeros sequence pointer and internal position registers.	To activate, hold down MODE key & press ZERO.

Table 5-1 Teachmover teach control command summary, via the manual F.3.

Writing trial programs using the commands listed in Table 5-1 is easy. For example, the program shown below moves each joint a few degrees to give the impression the robot is waving. Upon completing Step 7 the green RUN light should be lit and Auto will repeat oscillations in each joint until interrupted or powered off.

1. Press TRAIN key
2. Swivel the base by pressing one of the B keys.
3. Press the REC key.
4. Swivel the base back past the starting point by pressing the other B key.
5. Press the REC key.
6. Repeat steps 2 → 5 for each remaining joint S,E,P,R,G.
7. Press the MODE key and then the RUN key. (The RUN key is physically the same as the REC key once the mode modifier is activated.

While the trial program is useful, more involved programs often need to be written out or recorded on paper prior to entry. Microbot includes a convenient "Programming Worksheet" where someone can write out their programs, Figure 5-2. More complex programs, such as Microbot's Block Stacking program, require flow charts, Figure 5-3. Program complexity and therefore size can quickly out-grow the available onboard RAM. At this point, users are forced to interface Teachmovers with a computer. By interfacing to a computer a whole new set of command and features are made available as will be explored in the following pages.



PROGRAM_____

DATE_____

PROGRAMMER_____

PROGRAMMING WORKSHEET
FOR MICROBOT TEACHMOVER

APPENDIX G

STEP #	OPERATION	STEP #	OPERATION
0		27	
1		28	
2		29	
3		30	

Figure 5-2 Microbot provided Programming Worksheet, via the manual F.13.

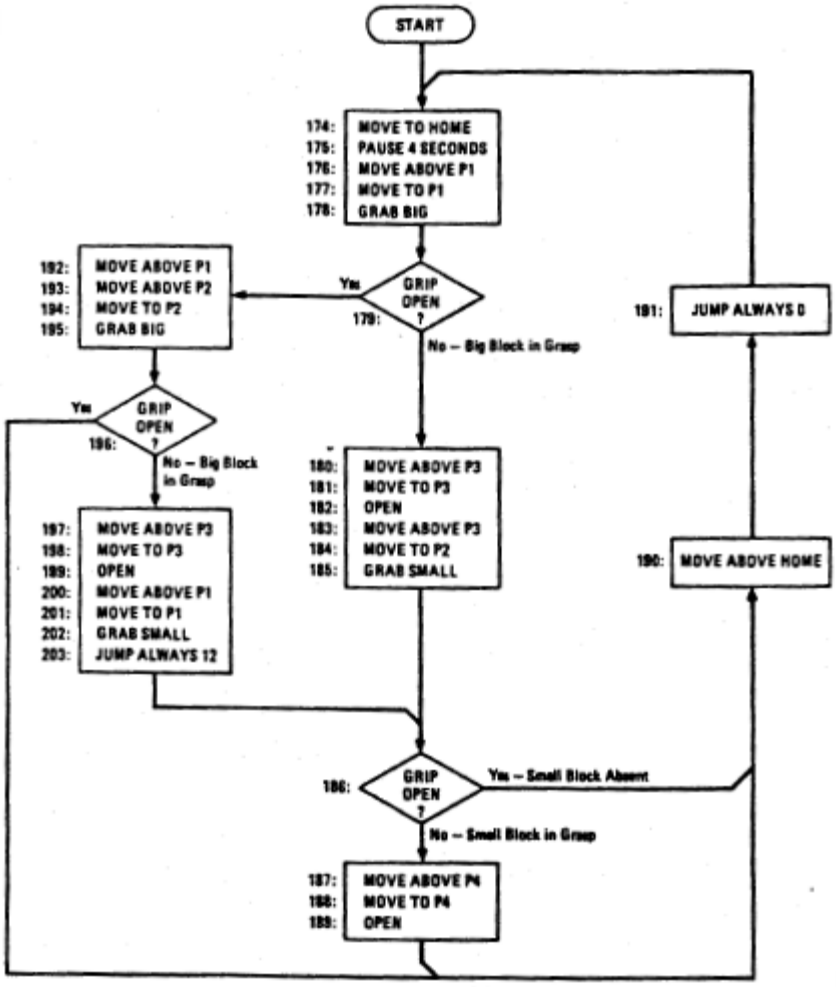


Figure 5-3 Microbot flowchart of their Block Stacking Program, via the manual 6.36.

Serial Communication

Communicating with this robot via serial requires some setup as serial devices are not generally plug-and-play. Exact communication protocol is covered by the Teachmover manual in Chapter 5 on "Electronics and Interfacing" and Chapter 7 on "Operation from a Host Computer" (Microbot, 1984). One starts by setting a baud rate and deciding if multiple robots are to be linked. Settings such as data format, interface signals, opening ports and testing the configuration are static and well documented. For this thesis, one Teachmover with the max baud rate of 9600bps and no flow control is used. To test the configuration an @CLOSE command is sent down the pipeline via HyperTerminal then a response will be issued: 0 = bad syntax, 1 = ok and 2 = stopped

Wiring is a critical part of proper setup. A null modem is vital to successful serial communication interfacing. Figure 5-4 shows the serial cabling. Standard DB-9 from a host converts to DB-25, passes through a surge protector then the null modem.

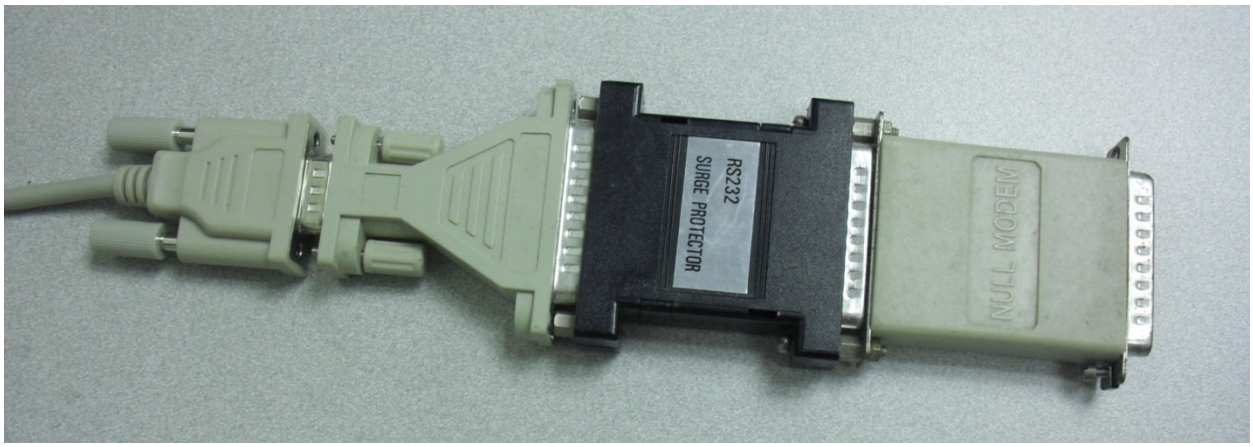


Figure 5-4 Serial communication solution.

Table 5-2 provides a listing the serial interface commands.

COMMAND	FUNCTION	SYNTAX/DETAILS
1) <CR> = Carriage Return 2) Arm returns [0<CR>] if command has syntax error, [1<CR>] after command is exe and [2<CR>] if STOP button is pressed before completing exe (STEP and CLOSE only)		
@ARM	Can change recognition character from @ to anything	@ARM <CHAR> <CR> where char is any but <CR>
@CLOSE	Close gripper until grip switch is on.	@CLOSE <SP> <CR> where SP = optional speed value
@DELAY	Inserts a delay b/w TX chars	@DELAY <N> <CR> where N is heuristically determined.
@QDUMP	Reads current program from RAM	@QDUMP <CR> Returns char string for ea program step. See Ch 7 Table 9.
@QWRITE	Writes a program step to RAM	@QWRITE <N>,<L1>,...<L7> <CR> where N is line number.
@READ	Reads internal position register values, gives last key pressed and input bit values.	@READ <CR> returns <K1>...<K6>,<I> <CR> where K = pos register and I = Last Key *256 + Input Byte, see App F.5
@RESET	Zeros the internal position registers and turns off motor current.	@RESET <CR>
@SET	Sets arm speed and activates joint control keys and teach control.	@SET <SP> <CR> where SP = optional speed value. Ctrl returned to host when REC or MODE key pressed.
@STEP	Sets arm speed, moves joints and sets output bits.	@STEP <SP>, <J1>...<J6>,<OUT><CR> where SP = optional speed value. J1-J6 = Motor half-steps base, shoulder, elbow, R wrist, L wrist and hand respectively, see App F.5 for more details. OUT = Opt decimal # whose binary equivalent specifies output bits (App F item G).

Table 5-2 Serial interface command summary, via the manual F.5.

With the new set of serial interface commands writing more sophisticated programs is possible. To demonstrate the basics of serial programming parts of an enhanced block stacking program, written in a MATLAB script, are shown in Figure 5-5. Notice in each step uses one or

several of the commands listed in Table 5-2. Figure 5-6 shows the `grip` function, which also uses several of the commands listed in Table 5-2. `grip` is integral to block stacking and serves a dual purpose: it measures and grasps blocks. By combining commands and utilizing the computer's memory, actions such as measuring, stacking blocks and graceful error handling are possible. Coding like this is only possible once users have graduated to the software programming environment and are talking to Teachmovers via serial commands. A video showing the enhanced block stacking program in action is available online (McCrate, 2009).

```
fopen(micro);      % Open COM port

% zeros the position registers
cmd = sprintf('%cRESET',arm);
fprintf(micro,cmd);
[tline,count,msg] = fgetl(micro);

% visit each block and measure
for ii = 1:3  % 3 blocks will be measured
      [...]
      % measure block size
      [tline,count,msg,b(ii)] = grip(arm);
      if tline == '2', break, end;
      [...]
end

[s,index] = sort(b,'descend');

for ii = 1:length(s)
    stack(ii) = sum(s(1:ii));   % height of n blocks
end

ii = 2; % start stacking from the 2nd largest block

% visit ~each block in decending order
while (ii <= length(s)) && (s(ii) > 10)  % blocks exist & are > 10 mm
      [...]
end

% move to home
[tline,count,msg] = pabs(arm,home);
if tline == '2', break, end;

fclose(micro);
```

Figure 5-5 Enhanced block stacking program.

```
function [tline,count,msg,mm] = grip(arm);

% grip(arm)
% mimics teach control grip functionality = CLOSE - 32 hsteps
% returns std comments AND block size

global micro;   % make global micro available to this function
global serialsp % make global serialsp available to this func

% close the gripper
cmd = sprintf('%cCLOSE',arm);
fprintf(micro,cmd);
[tline,count,msg] = fgetl(micro);
if tline == '2', return, end;

% build up 1 lb of force by squeezing 32 hsteps
[cmd, errmsg] = sprintf('%cSTEP %i %i,%i,%i,%i,%i,%i,
0',arm,serialsp,0,0,0,0,0,0);
fprintf(micro,cmd);
[tline,count,msg] = fgetl(micro);

% read position registers
cmd = sprintf('%cREAD',arm);
fprintf(micro,cmd);
[tline,count,msg] = fgetl(micro);
if tline == '2', return, end;
% reads and formats data sent after issuing read cmd
[c,count,msg] = fscanf(micro,'%i,%i,%i,%i,%i,%i,%i');

% use hand and elbow registers to calc object width
mm = (c(6)-c(3)) / 371 * 25.4;
```

Figure 5-6 Expansion of the grip() function used in Figure 5-5.

This concludes the chapter on a Teachmover's communication protocols and factory software. It opened with an explanation of the hardware interface and pseudo code for a short joint oscillating trial program. Focus then shifted to serial interfacing. Code for an advanced block stacking program is shown and this program is demoed in a short video. This code and block stacking demo are important because they showcase all the commands necessary to interface a basic drawing program. The next chapter explores how to build the drawing HMI/GUI using MATLAB.

Chapter 6 New User Interface and Software

To make Microbot Teachmovers pen images requires hardware and software capabilities beyond those on the factory platform. Improvements to the hardware are covered in other chapters. This chapter covers improvements to the software, specifically the user interface (UI). The new software-based UI, or HMI/GUI, allows users to make Teachmovers behave similar to the automaton in The Invention of Hugo Cabret, however some improvements are made that enable users to pen random images as well. This chapter introduces the capabilities and improvements of the new UI system by presenting them in a demonstration program.

Demo Program

Users begin the demo, and therefore the act of drawing, by running a program coded with MATLAB. When the demo program starts, it presents users with a popup menu, Figure 6-1. The menu contains a list of preprogrammed images. To initiate drawing, users pick what image they want from the menu. This action's analogue from a century ago would be replacing a set of cams and/or gear trains and winding springs. Like many drawing automata of old, this Microbot based automaton can draw several images. Only a few elementary shapes are used for in the demonstration; these include a: circle, square, fish, heart and house, as shown in Figure 6-2 to Figure 6-11, respectively. Images in the left column were created via the sixth and last option listed in the menu below. Images in the right column were drawn by a microbot.

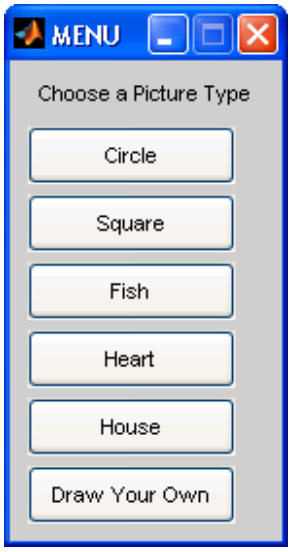


Figure 6-1 Menu used to determine images or drawing type.

Commanded via MATLAB

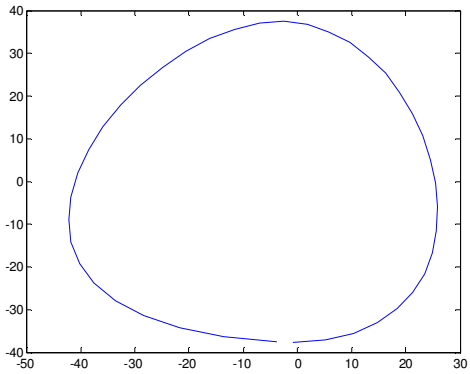


Figure 6-2 Circle

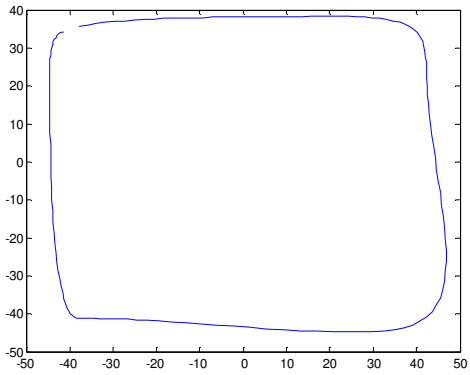


Figure 6-3 Square

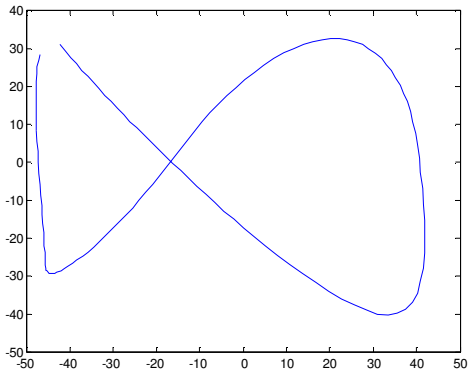


Figure 6-4 Fish

Drawn by a Teachmover

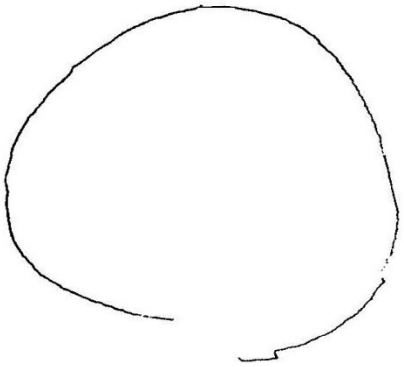


Figure 6-5 Circle

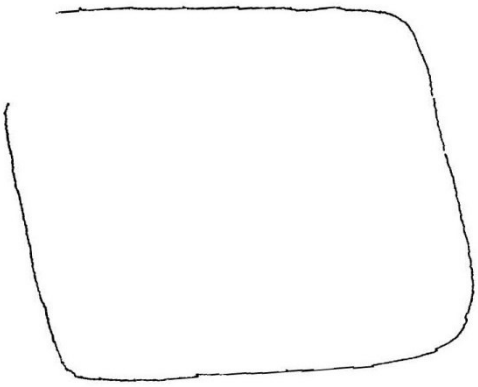


Figure 6-6 Square

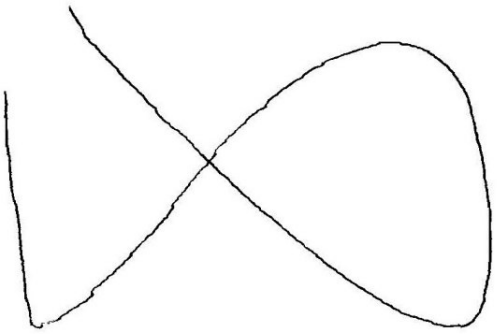


Figure 6-7 Fish

Commanded via MATLAB

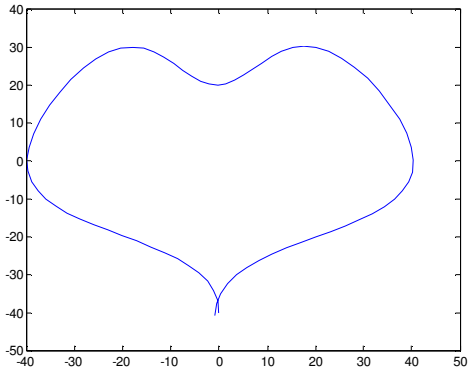


Figure 6-8 Heart

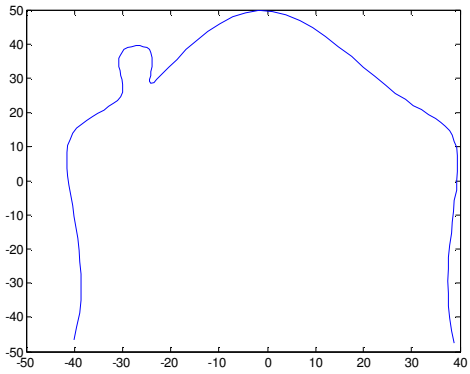


Figure 6-9 House

Drawn by a Teachmover

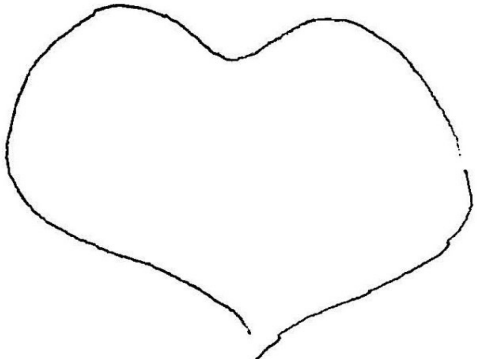


Figure 6-10 Heart

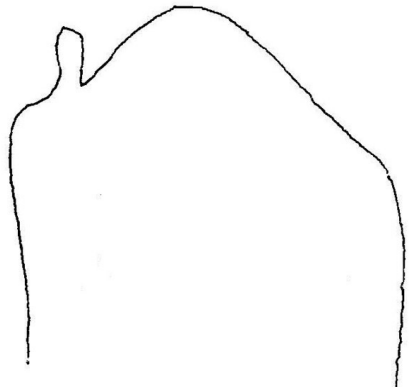


Figure 6-11 House

Unlike automata built around the turn of the 20th century, users of this Teachmover-based automaton can create their own drawing, using a connect-the-dots interface. Clicking "Create Your Own" from the first menu, Figure 6-1, activates the connect-the-dots menus. The second menu, Figure 6-12, is used to establish units, giving users a quantitative feel for size. The third menu, Figure 6-13, is used to establish roll frame. Roll frame is described in Chapter 4 and it has little bearing on penned images.

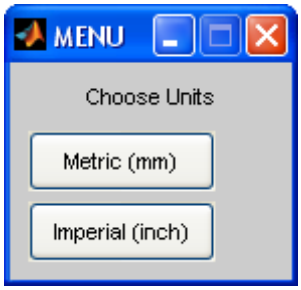


Figure 6-12 Menu to establish units.

Figure 6-13 Menu to choose roll frame

After answering the questions in menus two and three, above, a new much larger figure window opens, Figure 6-14, below, which has been made to look like MATLAB's Interactive Plotting example. People are charged with creating their desired image using the large figure window. Directions are given at the top of the screen and simultaneously in the MATLAB Command Window. The directions read, users left-click a mouse to create points and right-click to complete the process. Figure 6-15 shows an example image.

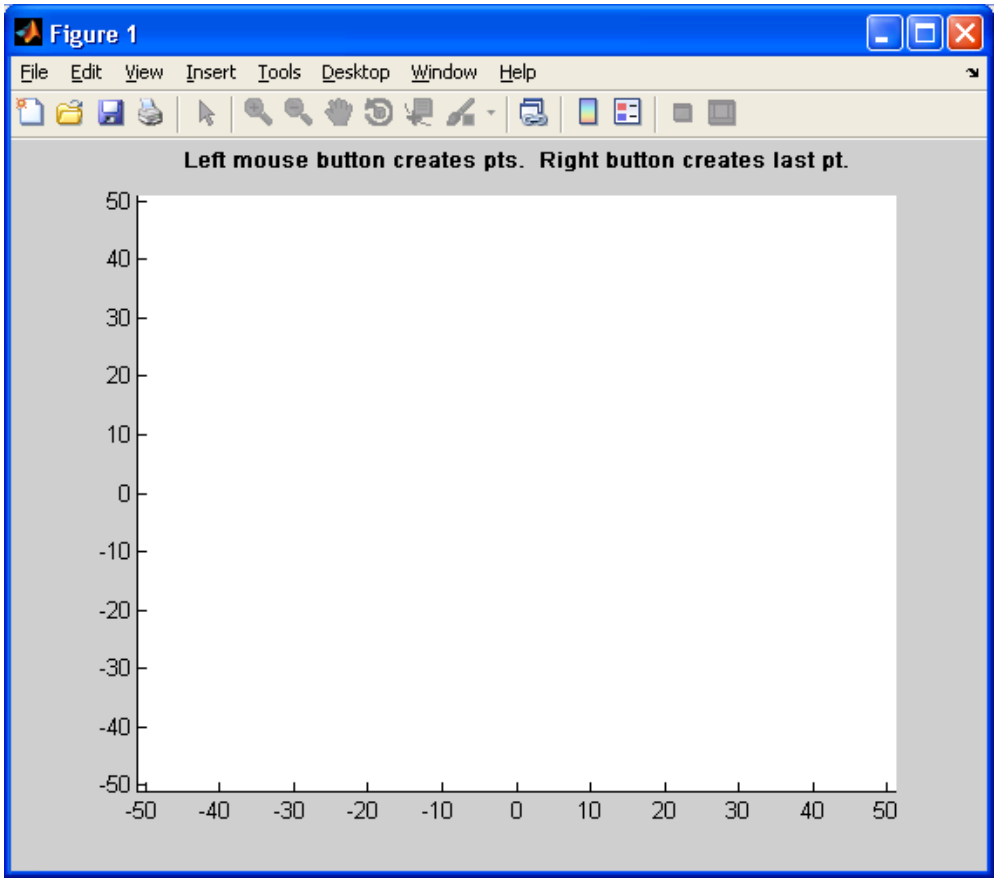


Figure 6-14 Connect-the-dots interface window.

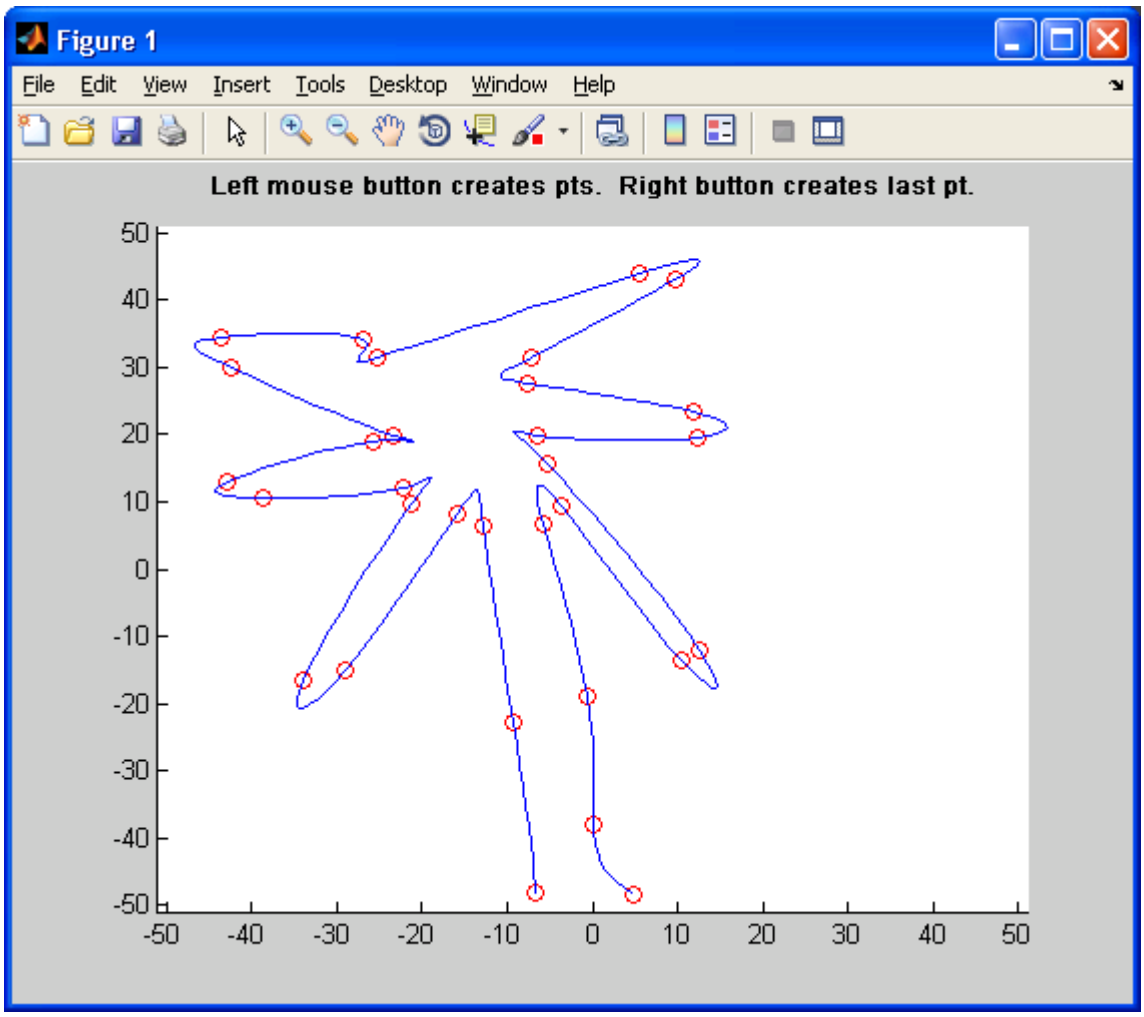


Figure 6-15 Example image, a palm tree!

Before the Teachmover will commence drawing its current position or pose must be entered into the program. Generally this is accomplished by selecting a location from one last menu, Figure 6-16. Three positions are very common: Cal Pt, Cal Pt-- and Home. Cal Pt is the calibration point as shown in Figure 4-1. This is the best place to reset the position registers after major slipping or a hard crash. Cal Pt-- is the Cartesian point (X,Y,Z) = (2,0,3) inch. In this position the pen is removed from the canvass and the arm is retracted, giving any audience a clear view of all freshly finished art work. Home is where all joint angles are zero: at this point the arm is fully extended marking a good opportunity to exchange the drawing utensil. Users are tasked with ensuring the robot is in one of these three positions prior to pushing the proper menu button. The next section contains a description of some of the demo program code.

Figure 6-16 Menu to define current position.

Demo Program Functions

Here, in this section, is a list of functions that directly support the drawing portion of the demo program.

Grip.m

Grip is written to mimic the Teach Control function named grip. First, it closes the gripper until the close switch is activated. Then it measures the block. Finally, it builds up one pound of grip force by squeezing an additional 32 half steps.

```
function [tline,count,msg,mm] = grip(arm);
% grip(arm)
% mimics teach control grip functionality = CLOSE - 32 hsteps
% returns std comments AND block size
```

Prel.m

Prel basically duplicates the Teach Control buttons. It will move the motors a commanded number of half-steps.

```
function  [tline,count,msg] = prel(arm,p)
% Position Relative (uncouples joints)
% prel(arm,[p])
% where p = B,S,E,P,R,G
% motor steps NOT angles
% make sure ONLY integers are sent
```

Pabs.m

Pabs is a function written to "force" the motors to turn until the internal position registers match the commanded values.

```
function  [tline,count,msg] = pabs(arm,p)
% Position ABSolute
% pabs(arm,p)
% p = [base,shoulder,elbow,R wrist,L wrist,grip]
% motor steps NOT angles
```

Cart_disp.m

Cart_disp stands for Cartesian Display. This function reads the Teachmovers internal position registers then calculates the forward kinematics before returning the robots current configuration.

```
function [disp] = cart_disp(arm)
% [disp] = cart_disp(arm);
% Displays current X,Y,Z,P,R,G
% X,Y,Z,P,R,G are Cartesian Coordinates then Pitch, Roll and grip width
```

Prel_cart.m

Prel_cart stands for Position Relative Cartesian. As the name implies, this function repositions the arm in the Cartesian frame by taking Cartesian inputs, e.g. move +Y by 3mm.

```
function  [tline,count,msg] = prel_cart(arm,pos)
% Cartesian coordinate based Relative positioning
% prel_cart(arm,[pos])
% where pos = B,S,E,P,R,G as shown on the Teach Control
```

Cart_cmd.m

Cart_cmd stands for Cartesian Command. This function calculates the inverse kinematics and required motor steps to move from home (where all angles and registers are zeroed) to the desired pose.

```
function [cmd] = cart_cmd(pos)
% Commanded motor steps needed to go from home (encoders = 0) to pos
% [J1,J2,J3,J4,J5,J6] = cart_cmd([pos]);
% pos = [x,y,z,p,r,g];
% J = [base,shoulder,elbow,R wrist, L wrist,grip]
```

Apporach.m

Approach is a written to trace unknown manifolds using the “touch sensor” shown in Figure 3-3.

```
function [tline,count,msg] = approach(arm);
% approach(arm)
% v.1 removes pen from paper then moves -z until the "touch sensor" trips
% v.3 rise until not touching, then touch, then back off one “step”
% returns std comments
```

Decode.m

The decode function interprets the last integer returned after issuing a READ command. This integer contains two types of information: the state of the Input Pins and the last Teach Control button that was pressed.

```
function [bits,currentkey] = decode(c);

% decodes the last integer of READ
% things differ from the manual e.g. inputs are grounded not pushed high

% their eqn I = LastKey*256 + Input Byte
% my eqn    I = CurrentKey*256 + 255 - 2^Input Bit
```

```
% bits are NORMALLY HIGH
% bits = [0,1,2,3,4,5,6,7]
% where 0 = closed indicator and 1-7 = user inputs

% current key, see table below
1 = train    8 = step
2 = pause    9 = point
3 = grip     10 = jump
4 = out      11 = clear
5 = free     12 = zero
6 = move     13 = speed
7 = stop     14 = run
```

This section shows the quantity and format of each function directly responsible for drawing. The theory supporting these functions can be found in Chapter 4 and Appendix A. All these functions are only called once the menus and other image acquisition portions of the demo program are completed.

Chapter Summary

In this chapter the new UI is introduced, as a demo program. The MATLAB based interface, software written to generate the motion commands and functions implemented in the process are all introduced as well. This chapter can serve as a stand-alone user manual for the thesis demonstration. The next chapter outlines how to build a fuzzy logic controller for the Microbot Teachmover. The fuzzy controller serves a dual role: it is vital to increasing the mean time between errant drawings and increases real-world throughput.

Chapter 7 Fuzzy Logic Control

Teachmover by Microbot is an economical, mechanically sound academic and industrial teaching tool however its 1980's era motion control system goes unstable quite often at high speeds. To eliminate instability a new controller was designed in MATLAB® using their Fuzzy Logic Toolbox™. To aid the fuzzy controller with observability, several sensors are added to Auto. Using input from these sensors, membership functions and a control rule base are tuned heuristically. When enabled the new fuzzy controller augments speed commands resulting in very fluid motion.

The University of Cincinnati has, in their Robotics laboratory, several Microbot made Teachmover robots. They are very light, being constructed mostly from sheet metal and aero-space cabling actuated by stepper motors. Unfortunately, the robots' Achilles' heel is power and stability. At high speed or under load/resistance, motors vibrate violently and the pulse counted position registers lose track of position. Of course, upgrading to powerful servo motors can solve these problems however the cost, power and time involved are all unwanted. Fortunately, any trained observer can detect the unstable and marginally stable states biologically using audio or tactical input. One can quantify these states using a microphone or accelerometer. Realizing and recreating our inherent ability to detect and attenuate speed based on incomplete qualitative data inspired and guided the current work.

Methodology and Materials

Implementing the fuzzy controller required 9 phases:

Part 1 Explore MATLAB's Data Acquisition Toolbox

Part 2 Acquire accelerometer/microphone

Part 3 Record accelerometer/microphone data via MATLAB

Part 4 Prepare a serial communication script

Part 5 Send the Microbot Teachmover a serial command

Part 6 Explore MATLAB's Fuzzy Logic Toolbox

Part 7 Form anecdotal/heuristic control functions

Part 8 Code control functions using Fuzzy logic

Part 9 Combine sensor data + fuzzy functions/rules = completed fuzzy controller

The following subsections explore the audio setup, first controller and fuzzy controller such that anyone "skilled in the art" can reproduce the results (United States Patent and Trademark Office, 2005).

Audio: Setup and Acquisition

Figure 7-1 Representative audio hardware.

A high level schematic of the audio capturing equipment is shown in Figure 7-1. The schematic includes a wireless microphone transmitter (left), its receiver (center) and a Behringer mixing box (right) for preprocessing/filtering. The mixer feeds directly into a computer sound card. A microphone is used in lieu of accelerometers for though they both capture similar vibration information the microphone is less expensive. Code written to invoke the microphone is shown below.

```
%%%%%  Create analog input object via a sound card and config
ai = analoginput('winsound');
addchannel(ai, 1);

Fs = 11025;                         % Sampling freq
ai.SampleRate = Fs;                 % Samplin freq again
ai.SamplesPerTrigger = Inf;         % no recording limit
ai.TriggerType = 'Immediate';       % start on start(ia)
ai.TriggerRepeat = Inf;             % no repeat limit


% Start analoginput obj w/ predefined properties
start(ai)
stop(ai)     % when finished

% gather all available data
data = getdata(ai,ai.SamplesAvailable);
```

First Pass Controller

Before building the fuzzy controller all proposed rules were tested using a simple bang-bang controller. Bang-bang controllers have – at least – two distinct advantages: they are time-optimal for a certain class of problems such as spacecraft-satellite attitude adjustment and have simple designs (Nagi, Perumal, & Nagi, 2009). The code for this simple controller is shown below.

```
if RMS > .007, serialsp = 239 else serialsp = 245, end;
```

A more graduated control law can be written that readily adjusts to microphone amplitude levels. The single line conditional would read:

```
if audio_amplitude >= large, serialsp-- else serialsp++, end;
```

Fuzzy Controller

The fuzzy controller is built using lessons learnt while working with the bang-bang controller. This fuzzy controller is designed as a MISO system (multi-input single-output): audio RMS and current serial speed are fed in and a new serial speed is spit out. To simplify fuzzification only two membership curves are used for each input; it is important to properly tune them. Tuned values can be seen in Figure 7-2 and Figure 7-3.

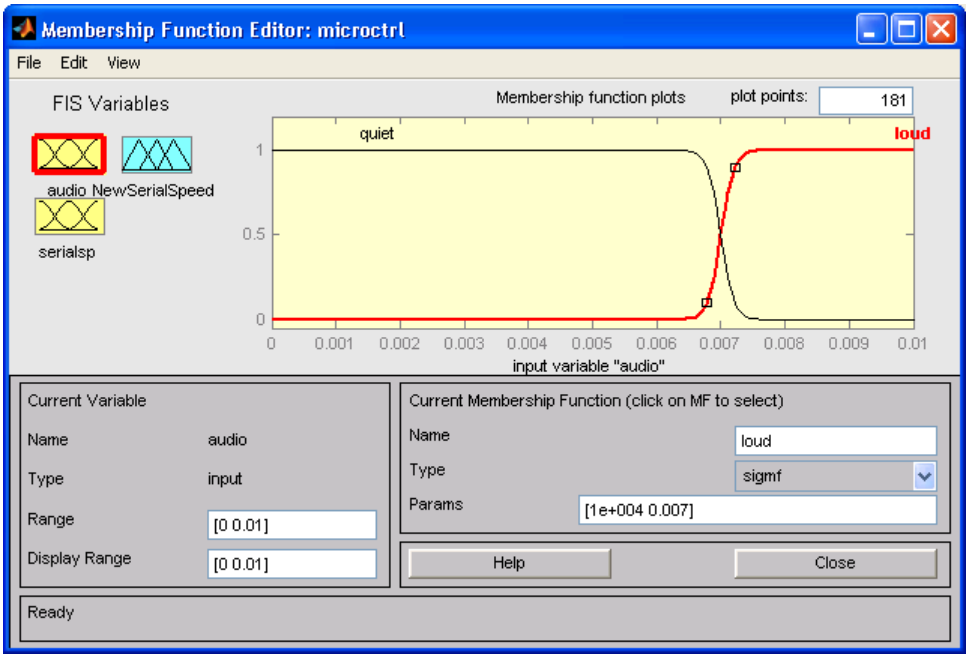


Figure 7-2 Audio, input variable, membership functions.

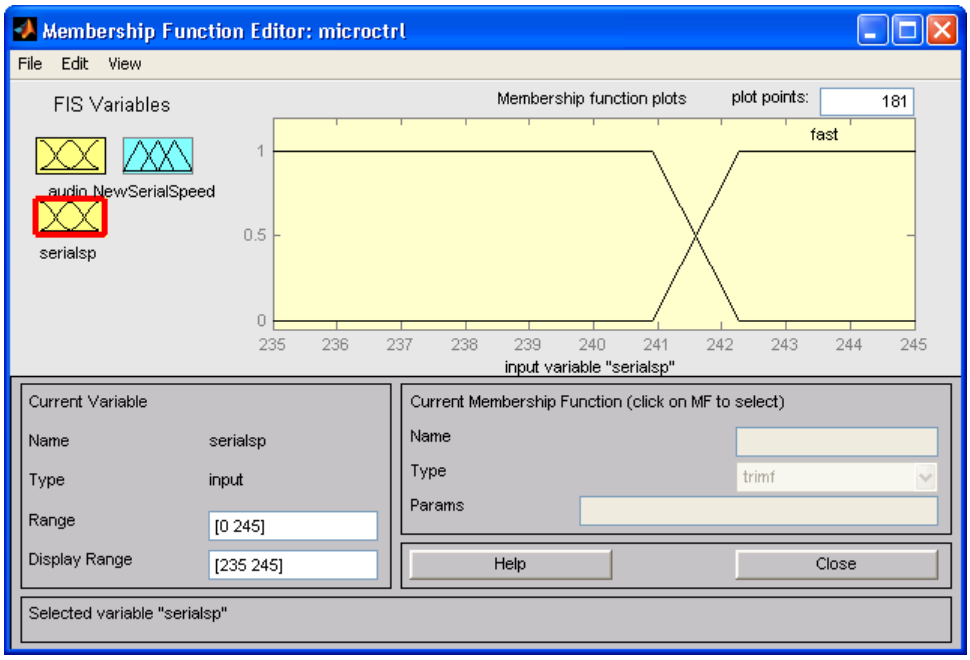


Figure 7-3 Serial speed, input variable, membership functions.

Only two membership curves are used for the output variable NewSerialSpeed. Their shapes can be seen in Figure 7-4.

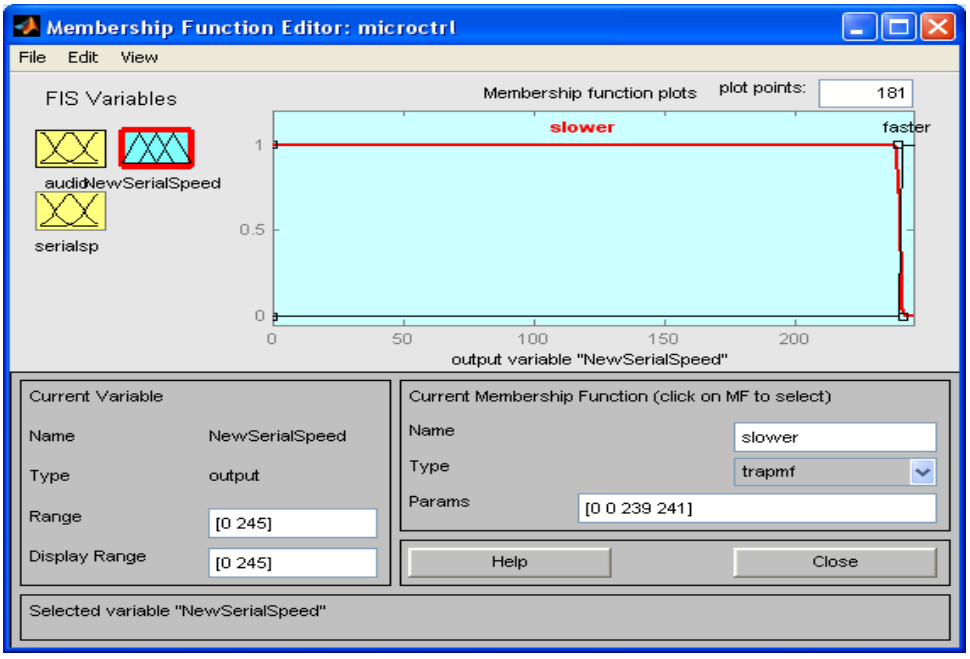


Figure 7-4 NewSerialSpeed, output, membership functions.

To develop a good rule base recall the rules previously implemented in the bang-bang controller. It had two simple rules: loud = slower; quiet = faster. These same two rules are used in the fuzzy controller. A third "performance enhancing" rule – of less weight – is added to the fuzzy controller to bias the speed upward without sacrificing stability. All rules are visible in Figure 7-5.

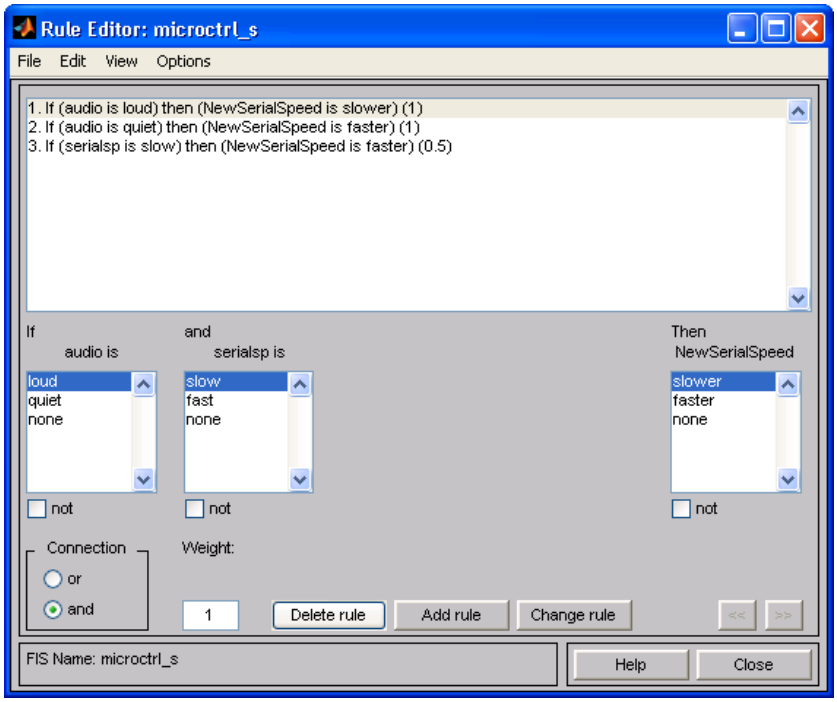


Figure 7-5 Fuzzy Rule base. Note number three is weighted less (0.5).

Having only three rules accelerates inferencing and defuzzifying. Execution can also be sped up by making the fuzzy system global, while by-passing all `evalfis()` argument checks. Code for by-passing all `evalfis()` argument checks is shown below.

```
%% Sugeno Fuzzy Speed;

fluid = readfis('microctrl_s');

serialsp = ceil(evalfismex([RMS serialsp],fluid,101))
```

Once tuned, the fuzzy controller(s) performed well. The Sugeno Fuzzy Inference System (FIS) produces a more graduated response surface than the Mamdani FIS. The Mambani surface resembles a pure bang-bang controller. See Figure 7-6 and Figure 7-7.

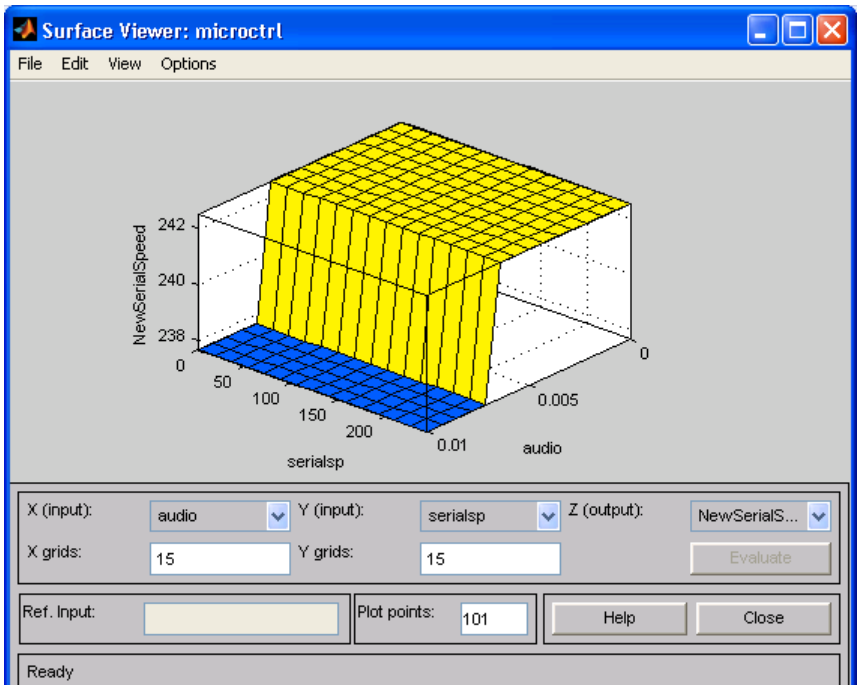


Figure 7-6 A Mamdani controller produces a nearly binary response surface.

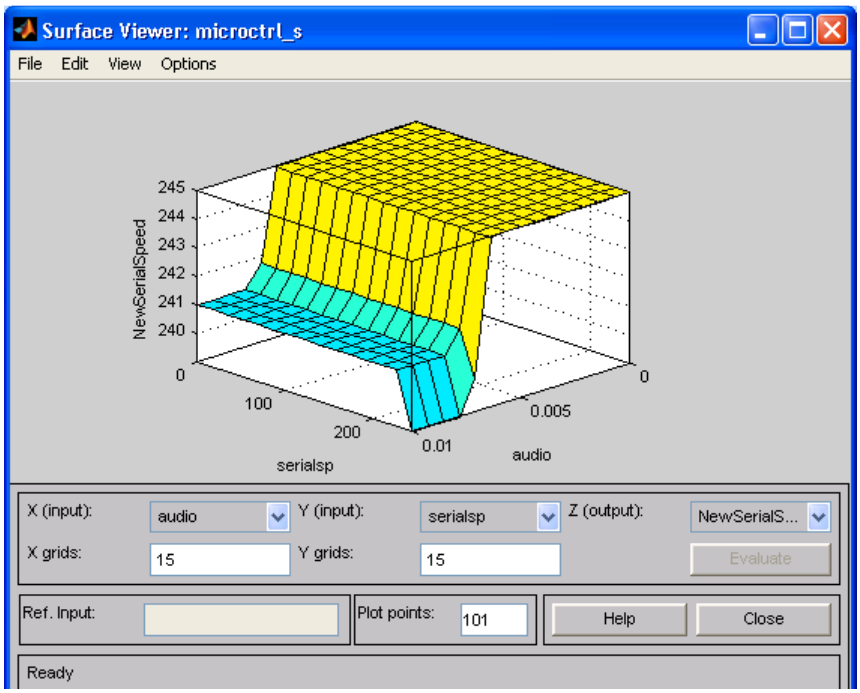


Figure 7-7 A Sugeno controller produces a partially graduated response surface.

The fuzzy controller runs well and yields predictable/steady behavior. One can see this in Figure 7-8 where three consecutive runs from a demo run are aggregated for easy viewing. Recall the goal is to use time histories of the audio signal to modify subsequent speed commands. Table 7-1 shows the controller inputs and outputs numerically.

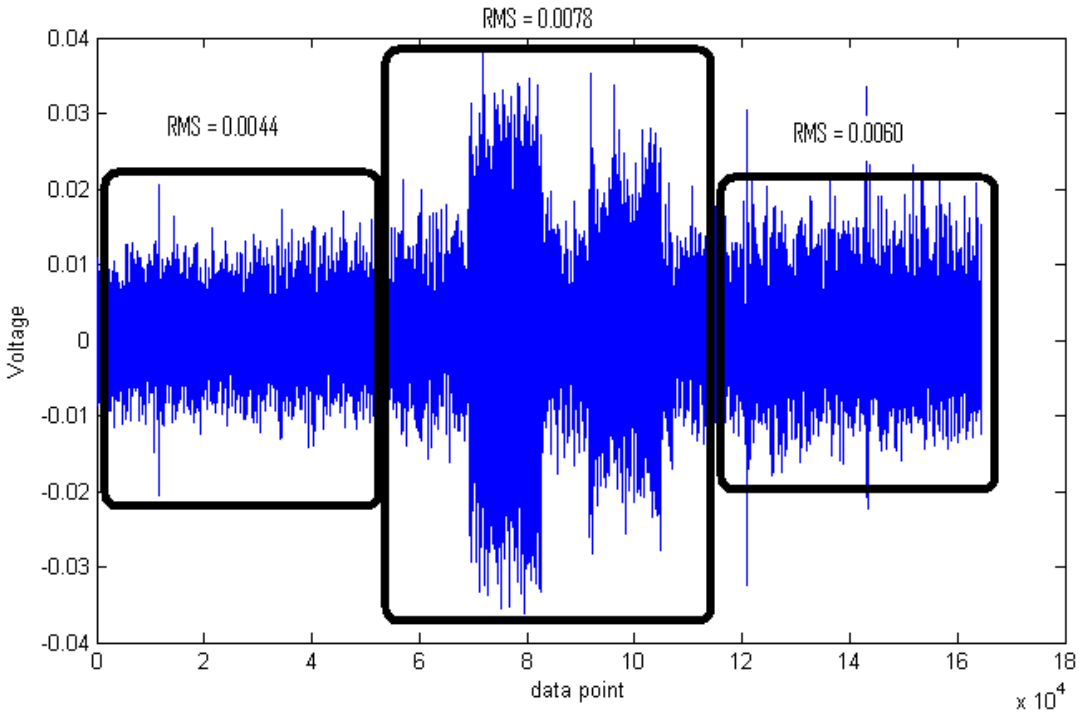


Figure 7-8 Consecutive Fuzzy runs oscillate about an RMS threshold.

Run	RMS	Resulting serial speed
Init	–	239
1	0.0044	245
2	0.0078	240
3	0.0060	245

Table 7-1 Tabular form of RMS data found in Figure 7-8.

The fuzzy controller performs more favorably than the graduated bang-bang controller and the uncontrolled system. The graduated bang-bang controller, used here, has two shortcomings: users must choose an initial speed and the controller migrates from this initial value slowly. Therefore, an incorrect initial speed value will result in poor performance,

initially. However, the bang-bang controller does outperform the uncontrolled system. When running without a controller users have two options: slow and stable or fast and unstable. There is no scope for adjusting on the fly meaning either quality or time will be lost.

Chapter Summary

Teachmover by Microbot, is a cheap and mechanically robust teaching tool. It has held up well over the years however its 1980's era motion controller needs help because it goes unstable at high speeds. This chapter showed how to solve the instability problem. It starts with describing a way to increase observability, transitions to designing a fuzzy logic controller and ultimately shows data from a demo run.

There are several places where this fuzzy work can be improved. Transforming measured data into the frequency-domain and then characterizing the broader-band noise of faster runs should be more robust since it is immune to audio amplitude. Figure 7-9 and Figure 7-10 show the frequency-domain contrast that gives this proposed method validity. Next, the current fuzzy controller, with only two membership functions curves per input, is a crude controller. More membership functions will widen the performance envelop. Finally, this work should be compared to other control methods. In the next and final chapter, conclusions and suggestions for future work will be discussed.

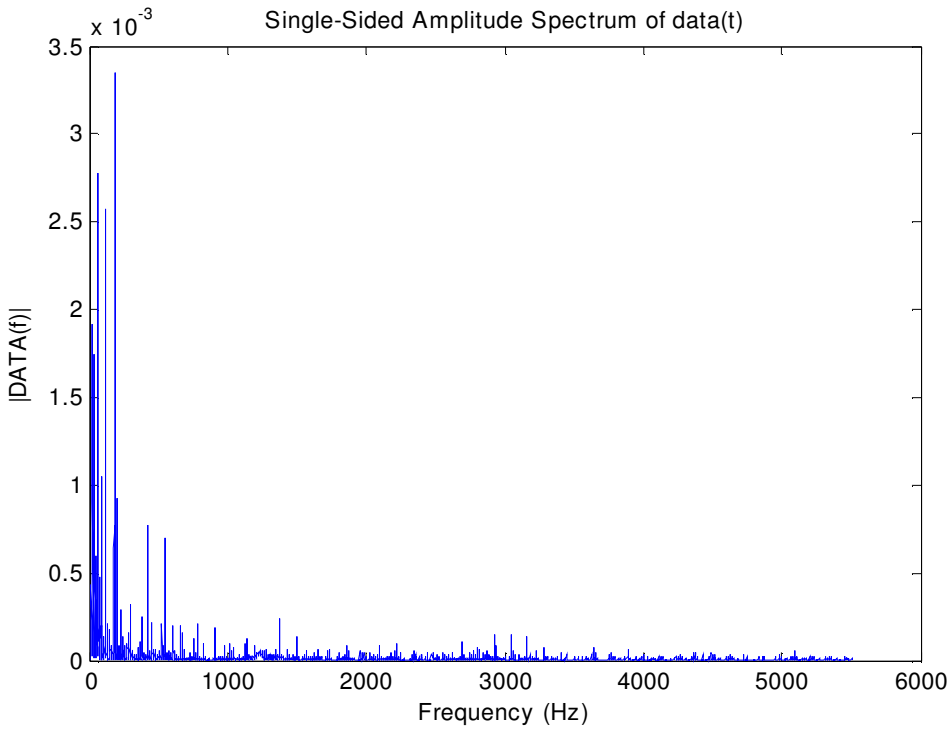


Figure 7-9 Typical FFT of a slow run.

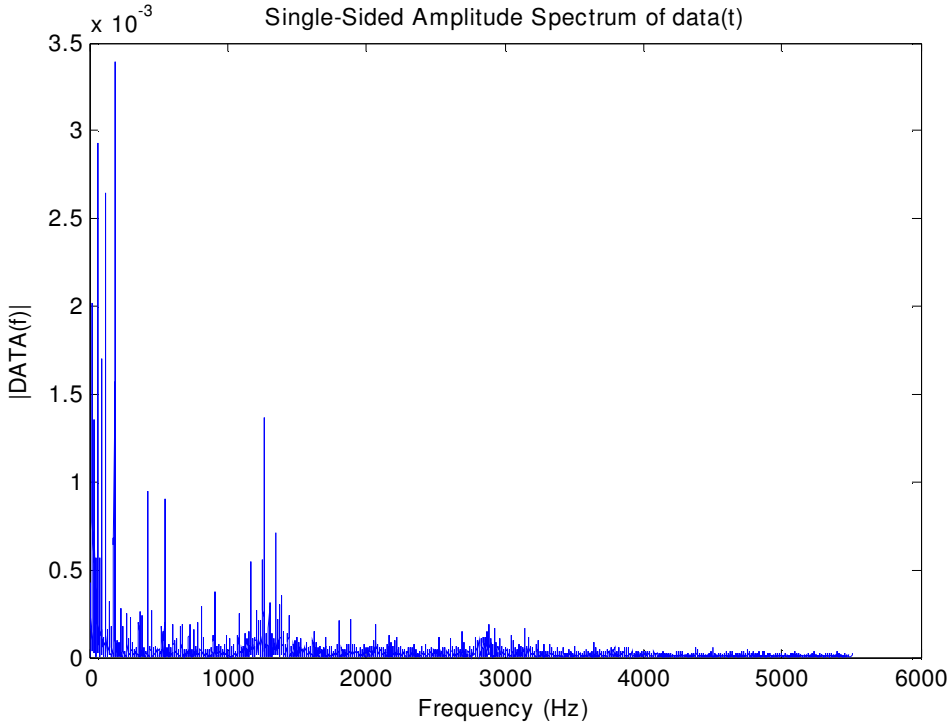


Figure 7-10 Typical FFT of a fast run. Notice the broad-band noise and spike ±1200 Hz.

Chapter 8 Contributions and Recommendations

This chapter contains a summary of the contributions made in this thesis work and includes recommendations for future work.

Contributions

Results of this thesis work include:

A. Building a combination ball point pen/touch sensor that is used as the drawing utensil. The pen provides feedback necessary for rudementary compliance control.

B. Solving the equations of motion, including forward and inverse kinematics, needed to control the robot arm's position.

C. Programming software, and a GUI, that allows people to make the robot draw both stored pictures and images created via user input. The software includes functions for positioning with step counts, Cartesian positioning, grip width measuring and random surface tracing.

D. Developing a fuzzy logic controller to improve the robot's drawing ability. The fuzzy controller uses a microphone for feedback and yields more rapid yet more stable movements, when compared to a bang-bang or uncontrolled system.

These results are described in this document in the following way. In Chapter 1, the focus is on understanding the "The Picasso Printer" approach and surveying different hardware options. Chapter 2 presents an overview of humanoid robots, dating from the 15th to the 21st Century. Construction methodologies and capabilities are compared. From this subset of robots a platform is selected based on its merits and availability.

Chapter 3 contains a hardware description. The description highlights key areas of the Microbot Teachmover design and all modifications necessary to ready it for drawing. Proper maintenance is addressed as well.

In Chapter 4 equations necessary to effectively position the end effecter are derived. This derivation is based in geometry and trigonometry. Starting from forward kinematics a framework is established and used to solve the inverse kinematics. All conventions and assumptions are laid out. Ultimately two tables contain the essential equations. They are formatted such that they can be easily programmed.

Chapter 5 provides background into a Microbot Teachmover's communication protocol and factory software. Two communication protocols are introduced: teach control protocol and serial protocol. The teach control protocol is included because it is the primary choice for most users. The serial protocol approach is necessary to generate advance movements such as those used when drawing pictures. Background into the factory software includes a table listing all available software based commands. This chapter culminates with a short block stacking program that touches on all the commands necessary to interface a basic drawing program.

Chapter 6 introduces the capabilities and improvements of a new User Interface (UI). A demonstration shows how users interact with the UI from beginning to end. By presenting the demo in this way, the chapter serves as a user manual. Each function, called by demo program, is presented as well to show how user input is converted to output commands and ultimately, a drawing.

In Chapter 7, a fuzzy logic controller is evaluated. The fuzzy controller is implemented to ameliorate high speed instabilities. Instabilities are detected using newly installed sensors, the data is processed in several ways and new speed commands are issued accordingly. Results and data from test runs show the controller can effectively mitigate excessive vibration. The controller therefore increases the mean time between failure and throughput.

In summary, this thesis shows how to produce a robust robot capable of drawing pictures, that is easy to modify and can be used as a collegiate or K-12 teaching tool.

Recommendations

Several recommendations for future work have been identified. Broadly speaking, applications beyond those described here, applications that utilize the robot's dexterity, should be explored. As of this writing, a "Towers of Hanoi" program has been written based on the functions described in Part C above. Other recommendations/suggestions/applications include:

- Having a multi-color pallet
- Making a more intuitive click-and-drag interface, as in (Fig, 2010)
- Adding provisions for 3D sculpting
- Fitting the robot onto a suck-puff system where it can be used for tasks like to elevator button pushing

Next, are several specific technical recommendations needed to achieve the proposed future work:

- Design and build a better end-effecter
- Add more sensors e.g a 6 DOF force-torque sensor
- Code a multi-input/multi-output Artificial Neural Network Fuzzy Inference solver

References

Adibhatla, G. (2007). Design and implementation of a compliance controller for the PA10-7CE seven degree of freedom dexterous robot. (Masters of Aerospace Engineering and Engineering Mechanics, University of Cincinnati).

Balestrino, A., De Maria, G., & Sciavicco, L. (1984). Robust control of robotic manipulators. *Proceedings of the 9th IFAC World Congress,* 2435-2440.

Bos, H. D. (2010). *Evolution of robotic hand.* University of Twente.

Bottema, O., & Roth, B. (1979). *Theoretical kinematics*. New York; Amsterdam: North-Holland Pub. Co.

Boyle, K. (2008a). *Karakuri origins.* Retrieved 8 April, 2010, from http://www.karakuri.info/origins/index.html

Boyle, K. (2008b). *Karakuri.info.* Retrieved 8 April, 2010, from http://www.karakuri.info/

Boyle, K. (2008c). *Zashiki karakuri.* Retrieved 8 April, 2010, from http://www.karakuri.info/zashiki/index.html

Breazeal, C. (2000). *Sociable machines: Expressive social exchange between humans and robots*. (Sc. D, MIT).

Breazeal, C. (2004). Function meets style: Insights from emotion theory applied to HRI. *IEEE Transactions on Systems, Man and Cybernetics, Part C: Appications and Reviews, 34*(2), 187-194.

Breazeal, C., & Scassellati, B. (2002). Robots that imitate humans. *Trends in Cognative Science,* (6), 481-487.

Buss, S. (2004). *Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped lease squares methods* University of California, San Diego:

Caplan, J. (2008, 10 Nov). The 50 best inventions of 2008. *Time, 67*

Clark, D., Collins, S., Ellis, A., & Watson, S. (2008). *The invention of hugo cabret "the picasso printer"*

Craig, J. J. (1986). *Introduction to robotics : Mechanics & control*. Reading, MA: Addison-Wesley Pub. Co.

Denavit, J., & Hartenberg, R. S. (1955). A kinematic notation for lower-pair mechanisms based on matrices. *Journal of Applied Mechanics, 22*, 215-221.

Denavit-hartenberg parameters. (2009). Retrieved 11/24, 2009, from http://en.wikipedia.org/wiki/Denavit-Hartenberg_Parameters

Deo, A. S., & Walker, I. D. (1993). Adaptive non-linear least squares for inverse kinematics. *Proceedings of the IEEE International Conference on Robotics and Automation,* 186-193.

D'Souza, A., Vijayakumar, S., & Schaal, S. (2001). Learning inverse kinematics. *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems,* 298-303.

Eastern Cranial Affiliates LLC. (2009). *Upper extremity prosthetic devices: Lower limb prosthetics.* Retrieved 10/10, 2010, from http://www.infinitetech.org/prosthetics_upper_limb_extremity_prosthetic_devices.aspx

Fig, M. (2010). *41 complete GUI examples.* Retrieved 9/8, 2010, from http://www.mathworks.com/matlabcentral/fileexchange/24861-41-complete-gui-examples

Goertz, R. C., & Uecker, D. F. (1951). *Electrical manipulator*

Goldsmith, J., & Worsdall, M. (2010). *Shadow robot company: The hand overview.* Retrieved 10/10, 2010, from http://www.shadowrobot.com/hand/overview.shtml

Grzeszczuk, R., & Terzopoulos, D. (1995). Automated learning of muscle-actuated locomotion through control abstraction. *Proceedings of ACM SIGGRAPH'95,* 63-70.

Grzeszczuk, R., Terzopoulos, D., & Hinton, G. (1998). NeuroAnimator: Fast neural network emulation and control of physics-based models. *Proceedings of ACM SIGGRAPH'98,* New York. 9-20.

Hogan, N. (1985a). Impedance control: An approach to manipulation: Part I - theory. *Transactions of the ASME Journal of Dynamic Systems, Measurement and Control, 107*(1), 1-7.

Hogan, N. (1985b). Impedance control: An approach to manipulation: Part II - implementation. *Transactions of the ASME Journal of Dynamic Systems, Measurement and Control, 107*(1), 8-16.

Hogan, N. (1985c). Impedance control: An approach to manipulation: Part III - aplications. *Transactions of the ASME Journal of Dynamic Systems, Measurement and Control, 107*(1), 17-24.

Huston, R. L. (2001). In Liu C. Q. (Ed.), *Formulas for dynamic analysis*. New York: Marcel Dekker.

Inverse kinematics. (2010). Retrieved 5/10, 2010, from http://en.wikipedia.org/w/index.php?title=Inverse_kinematics&action=history

Jaeger, S. (2005). *Visual journal - robot history.* Retrieved 10/10, 2010, from http://pages.cpsc.ucalgary.ca/~jaeger/visualMedia/robotHistory.html

Jordan, M. I., & Rumelhart, D. E. (1992). Forward models: Supervised learning with a distal teacher. *Cognitive Science, 16*, 307-354.

Kaneko, K., Harada, K., Kanehiro, F., Miyamori, G., & Akachi, K. (2008). Humanoid robot HRP-3. Nice, France. 2471-2478.

Kittel, N. (2008). *Karakuri ningyo: Yesterday's robotics.* Retrieved 8 April, 2010, from http://www.abc.net.au/local/videos/2008/11/12/2417728.htm

Kusuda, Y. (2008). Toyota's violin-playing robot. *Industrial Robot: An International Journal, 35*(6), 504-506.

Law, J. M. (1997). *In the shape of a person: The varieties of ritual uses of effigy in japan*. Princeton, NJ, USA: Princeton University Press.

Lendaris, G. G., Mathia, K., & Sacks, R. (1999). Linear hopfield networks and constrained optimization. *IEEE Transactions on Systems, Man, and Cybernetics - Part B Cybernetics, 29*, 114-118.

Mansfeld, J. (1998). In Runia D. T., van Winden J. C. M. (Eds.), *Chapter VI heron of alexandria*. Leiden, Netherlands: Brill.

Mason, M. T. (1981). Compliance and force control for computer controlled manipulators. *IEEE Transactions on Systems, Man and Cybernetics, SMC-11*(6)

McCrate, M. (2009). *Microbot teachmover - enhanced block stacking program.* Retrieved 10/10, 2009, from http://www.youtube.com/watch?v=bnhDGNO71PQ

McCrate, M., & Boyle, K. (2010). *Karakuri request*

Microbot. (1984). *Operation of the five-axis robot model TCM*. Mountain View, CA, USA: Microbot.

Mish, F. (Ed.). (1995). *Merriam webster's collegiate dictionary* (10th ed.). Springfield, Massachusetts, USA: Merriam-Webster, Inc.

Nagi, F., Perumal, L., & Nagi, J. (2009). A new integrated fuzzy bang-bang relay control system. *Mechatronics, 19*(5), 748-760.

Nakamura, Y., & Hanafusa, H. (1986). Inverse kinematics solutions with singularity robustness for robot manipulator control. *Transactions of the ASME Journal of Dynamic Systems, Measurement and Control, 108*(3), 163-171.

Oyama, E., Chong, N. Y., Agah, A., Maeda, T., & Tachi, S. (2001). Inverse kinematics learning by modular architecture neural networks with performance prediction networks. *Proceedings of the IEEE International Conference on Robotics and Automation,* 1006-1012.

Page, E. (2007, 1/2007). Remote control: Mechatronic device extends author's reach over the internet. *Design Engineering,* , 30.

Paul, R. P. (1981). *Robot manipulators : Mathematics, programming, and control : The computer control of robot manipulators*. Cambridge, Mass: MIT Press.

Ramdane-Cherif, A., Daachi, B., Benallegue, A., & Levy, N. (2002). Kinematic inversion. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems,* 1904-1909.

Reuleaux, F. (1876). *Kinematics of Machinery*(A. B. W. Kennedy Trans.). London, England: Macmilan and Company, Ltd.

Robot. (2010). Retrieved 4/8, 2010, from http://en.wikipedia.org/wiki/Robot

Sakagami, Y., Watanabe, R., Aoyama, C., Matsunaga, S., Higaki, N., & Fijimura, K. (2002). The intelligent ASIMO: System overview and integration. 2478-2483.

shinjiredfield. (2008). *karakuri02pv7.jpg.* Retrieved 13 April, 2010, from http://nubikaraokeparty.files.wordpress.com/2008/06/karakuri02pv7.jpg

SHOBEI Tamaya IX. (2008). *Yumihiki doji (archer doll).* Retrieved 15 Apr, 2010, from http://www.karakuri.info/zashiki/index.html

Spong, M. W. (2006). In Hutchinson S., Vidyasagar M. (Eds.), *Robot modeling and control.* Hoboken, NJ: John Wiley & Sons.

Tevatia, G., & Schaal, S. (2000). Inverse kinematics for humanoid robots. *Proceedings of the IEEE International Conference on Robotics and Automation,* 294-299.

The Mathworks. (1998). *Help file on anfis.* Natick, MA, USA:

The Mathworks. (2010). *Anfis and the ANFIS editor GUI: Tutorial (fuzzy logic toolbox): Constraints of anfis.* Retrieved 1/15, 2010, from http://www.mathworks.com/access/helpdesk/help/toolbox/fuzzy/fp715dup12.html

The MathWorks. (2010). *Modeling inverse kinematics in a robotic arm demo.* Retrieved 11/17, 2009, from http://www.mathworks.com/products/fuzzylogic/demos.html?file=/products/demos/shipping/fuzzy/invkine_codepad.html

Toyota Motor Corporation. (2003). *TOYOTA.CO.JP -toyota partner robot.* Retrieved 10/24, 2010, from http://www.toyota.co.jp/en/special/robot/

United States Patent and Trademark Office. (2005). *General information concerning patents.* Retrieved 7/17, 2009, from http://www.uspto.gov/web/offices/pac/doc/general/index.html#functions

Wampler, C. W. (1986). Manipulator inverse kinematic solutions based on vector formations and damped least squares method. *IEEE Transactions on Systems, Man, and Cybernetics, 16*, 93-101.

Wang, L. C. T., & Chen, C. C. (1991). A combined optimization method for solving the inverse kinematics problem of mechanical manipulators. *IEEE Transactions on Robotics and Automation, 7*, 489-499.

Wang, W., Loh, R. N. K., & Gu, E. Y. (1998). Passive compliance versus active compliance in robot-based automated assembly systems. *Industrial Robot: An International Journal, 25*(1), 48-57.

Whitney, D. E. (1969). Resolved motion rate control of manipulators and human prosetheses. *IEEE Transactions on Man-Machine Systems, 10*, 47-53.

Whitney, D. E. (1977). Force feedback control of manipulator fine motions. *Transactions of the ASME Journal of Dynamic Systems, Measurement and Control, 99*(2), 91-97.

Wolovich, W. A., & Elliot, H. (1984). A computational technique for inverse kinematics. *IEEE Conference on Decision and Control, , 23* 1359-1363.

Wood, G. (2002). *Edison's eve*. New York: Alfred A. Knopf.

Zhao, J., & Badler, N. I. (1994). Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Transactions on Graphics, 13*, 313-336.

Appendix A Generalized Kinematics

"Formally, kinematics is that branch of mechanics that treats the phenomena of motion without regard to the cause of the motion. In kinematics there is no reference to mass or force; the concern is only with relative positions and their changes." (Bottema & Roth, 1979) As Microbot Teachmovers are light, slow moving robot with minimal payload capabilities, focusing on kinematics makes sense. In Chapter 4 Special Kinematics, for Teachmovers, both forward and inverse kinematics have been solved. Those equations are valid for all Teachmovers, however the derivation methods lack generality and utilizes many simplifying assumptions and constraints. Here, the forward and inverse kinematics are solved using generalized conventions and modern methods; specifically, forward kinematics are formulated based on Denavit-Hartenberg's convention while an artificial neural-network fuzzy inference system (ANFIS) is used for the inverse kinematic solution.

Forward Kinematics based on Denavit-Hartenberg Parameterization

Many conventions can be used to describe manipulator kinematics, there are Euler Angles; roll, pitch and yaw angles; axis/angle representations to name but a few. One convention/method, known at the Denavit-Hartenberg parameterization, stands above the rest as being the most popular among robotists. Denavit and Hartenberg build on the work of Reuleaux (Reuleaux, 1876). All these works utilize homogeneous transformations, which are referred to as Special Euclidean Groups or dyadics in (Huston, 2001). Denavit and Hartenberg's abstract explains:

> A symbolic notation devised by Reuleaux to describe mechanisms did not recognize the necessary number of variables needed for complete description. A reconsideration of the problem leads to a symbolic notation which permits the complete description of the kinematic properties of all lower-pair mechanisms by means of equations. The symbolic notation also yields a method for studying lower-pair mechanisms by means of matrix algebra; two examples of application to space mechanisms are given. (Denavit & Hartenberg, 1955)

It would seem that in a 6DOF world six parameters/coefficients "constitutes the necessary and sufficient number" for a complete description of position and pose (Denavit & Hartenberg, 1955). Using matrix notation, the special cases of pure translations or rotations can be represented as follows (note: standard right-hand coordinate systems are used, n_1 = x-axis, n_2 = y-axis, n_3 = z-axis, T ≡ translation, R ≡ rotation and finally the symbol inside the ( ) represents the angle or distance of action.

$$T_x(a) = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & Cos(\alpha) & -Sin(\alpha) & 0 \\ 0 & Sin(\alpha) & Cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_y(b) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad R_y(\beta) = \begin{bmatrix} Cos(\beta) & 0 & Sin(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ -Sin(\beta) & 0 & Cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_z(c) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad R_z(\gamma) = \begin{bmatrix} Cos(\gamma) & -Sin(\gamma) & 0 & 0 \\ Sin(\gamma) & Cos(\gamma) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

However, by careful selection of coordinate frame assignments "four coefficients suffice" (Denavit & Hartenberg, 1955). Spong has this to say regarding existence and uniqueness, "Clearly it is not possible to represent any arbitrary homogeneous transformation using only four parameters (Spong, 2006). Given two reference frames, 0 and 1, two conditions must be met:

(DH1) axis x_1 is perpendicular to axis z_0

(DH2) axis x_1 intersects the z_0

Historically, the matrix result, after packaging each of the four parameters into one matrix, has been label "**A**." A systematic method for building **A** matrices is given in (Paul, 1981). (Craig, 1986) expands upon the explanation and provides some useful examples. A chapter in (Spong, 2006) quite literally contains a tutorial on assigning coordinate frames. Using

D-H parameters one can compactly represent transformations imparted by individual links, then quickly do the matrix multiplication required to completely solve serial-link kinematics.

Each **A** matrix represents the transformation imparted by a link of an n-link manipulator (special care is taken when assigning frame 0 and frame n or the first and last frames, respectively). **A** matrices are the direct result of multiplying four 4x4 matrices where each 4x4 matrix represents the homogeneous transformation of one parameter. To clarify, assume the desired operations are: $R_{z,\theta}\ T_{z,d}\ T_{x,a}\ R_{x,\alpha}$; where θ_i, d_i, a_i and α_i are unique for each link. Multiplying in the given order yields Eqn 23. Post-multiplying operates on the subsequent "current frames."

$$A_i = \mathrm{Rot}_{z,\theta_i}.\mathrm{Trans}_{z,d_i}.\mathrm{Trans}_{x,a_i}.\mathrm{Rot}_{x,\alpha_i}$$

$$= \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha_i) & -\sin(\alpha_i) & 0 \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_i = \begin{matrix} \mathrm{Cos}[\theta_i] & -\mathrm{Cos}[\alpha_i]\mathrm{Sin}[\theta_i] & \mathrm{Sin}[\alpha_i]\mathrm{Sin}[\theta_i] & \mathrm{Cos}[\theta_i]a_i \\ \mathrm{Sin}[\theta_i] & \mathrm{Cos}[\alpha_i]\mathrm{Cos}[\theta_i] & -\mathrm{Cos}[\theta_i]\mathrm{Sin}[\alpha_i] & \mathrm{Sin}[\theta_i]a_i \\ 0 & \mathrm{Sin}[\alpha_i] & \mathrm{Cos}[\alpha_i] & d_i \\ 0 & 0 & 0 & 1 \end{matrix} \qquad \textbf{Eqn 23}$$

Generally, the symbols used in Eqn 23 are defined as follows: $a_i \equiv$ link length, $\alpha_i \equiv$ link twist, $d_i \equiv$ link offset, $\theta_i \equiv$ joint angle. Craig (Craig, 1986) gives the definitions in an easy to read bulleted list:

- a_i = distance from Z_i to Z_{i+1} measured along X_i
- α_i = angle between Z_i and Z_{i+1} measured about X_i
- d_i = distance from X_{i-1} to X_i measured along Z_i
- θ_i = angle between X_{i-1} and X_i measured about Z_i

Where three of the four parameters (a, α and either d or θ) are fixed for any given link (d varies in a prismatic link and θ in a revolute link). Throughout the various literature parameter definitions will vary depending on how frames have been attached to links. In the original paper,

symbols a, α, s and θ are used and most recently symbols r, α, d and θ are popular (*Denavit-hartenberg parameters.*2009).

Now that the basic math has been introduced, attention turns to assigning parameters to the Teachmover. Frames are assigned according to Figure A-1; the global frame is fixed to the floor; sequentially there are: Base (hidden), Shoulder, Elbow, Pitch, Roll and grip. Table A-1, summarizes: a, α, d and θ parameters; they are similar to that of a PUMA 560. Links 1 → 5 correspond to Base, Shoulder, Elbow, Pitch and Roll in that order, which, for clarity, is the Teach Control order. In Table A-1, all d(s) are fixed because all Teachmover joints are revolute.

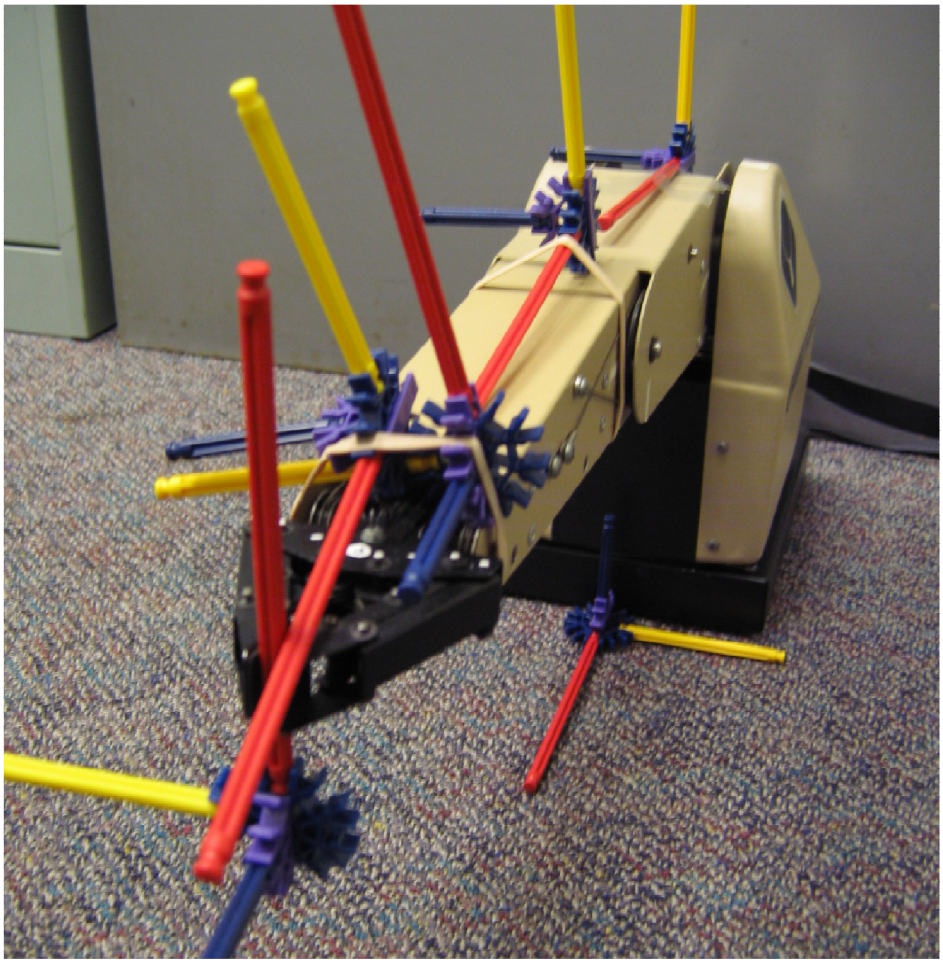


Figure A-1 Link Coordinate Frames; all are right-hand; x = red, y = yellow & z = blue.

	a_i	α_i	d_i	θ_i
Link $1(Base)$	0	$\frac{\pi}{2}$	h	θ_1
Link $2(Shoulder)$	L	0	0	θ_2
Link $3(Elbow)$	L	0	0	θ_3
Link $4(Pitch)$	0	$\frac{\pi}{2}$	0	$\frac{\pi}{2}+\theta_{44}$
Link $5(Roll)$	0	0	LL	θ_5

Table A-1 Summary of Microbot Teachmover D-H parameters.

Placing each row of Table A-1 into an A matrix yields:

$$A_1 = \begin{bmatrix} \mathrm{Cos}[\theta_1] & 0 & \mathrm{Sin}[\theta_1] & 0 \\ \mathrm{Sin}[\theta_1] & 0 & -\mathrm{Cos}[\theta_1] & 0 \\ 0 & 1 & 0 & h \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} \mathrm{Cos}[\theta_2] & -\mathrm{Sin}[\theta_2] & 0 & L\mathrm{Cos}[\theta_2] \\ \mathrm{Sin}[\theta_2] & \mathrm{Cos}[\theta_2] & 0 & L\mathrm{Sin}[\theta_2] \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} \mathrm{Cos}[\theta_3] & -\mathrm{Sin}[\theta_3] & 0 & L\mathrm{Cos}[\theta_3] \\ \mathrm{Sin}[\theta_3] & \mathrm{Cos}[\theta_3] & 0 & L\mathrm{Sin}[\theta_3] \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_4 = \begin{bmatrix} -\mathrm{Sin}[\theta_{44}] & 0 & \mathrm{Cos}[\theta_{44}] & 0 \\ \mathrm{Cos}[\theta_{44}] & 0 & \mathrm{Sin}[\theta_{44}] & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \textbf{Eqn 24}$$

$$A_5 = \begin{bmatrix} \mathrm{Cos}[\theta_5] & -\mathrm{Sin}[\theta_5] & 0 & 0 \\ \mathrm{Sin}[\theta_5] & \mathrm{Cos}[\theta_5] & 0 & 0 \\ 0 & 0 & 1 & \mathrm{LL} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Building T, where T ≡ transformation from base to point of interest (often considered end effecter tip), simply involves matrix multiplication, see Eqn 25.

$$T = A_1 A_2 \ldots A_{n-1} = \begin{matrix} r_{1,1} & r_{1,2} & r_{1,3} & r_{1,4} \\ r_{2,1} & r_{2,2} & r_{2,3} & r_{2,4} \\ r_{3,1} & r_{3,2} & r_{3,3} & r_{3,4} \\ r_{4,1} & r_{4,2} & r_{4,3} & r_{4,4} \end{matrix} \qquad \textbf{Eqn 25}$$

Where $r_{\#,\#}$ are given in Eqn 26.

$$r_{1,1} = \text{Sin}[\theta_1]\text{Sin}[\theta_5] - \text{Cos}[\theta_1]\text{Cos}[\theta_5]\text{Sin}[\theta_2 + \theta_3 + \theta_{44}]$$
$$r_{1,2} = \text{Cos}[\theta_5]\text{Sin}[\theta_1] + \text{Cos}[\theta_1]\text{Sin}[\theta_5]\text{Sin}[\theta_2 + \theta_3 + \theta_{44}]$$
$$r_{1,3} = \text{Cos}[\theta_1]\text{Cos}[\theta_2 + \theta_3 + \theta_{44}]$$
$$r_{1,4} = \text{Cos}[\theta_1](L\text{Cos}[\theta_2] + L\text{Cos}[\theta_2 + \theta_3] + \text{LLCos}[\theta_2 + \theta_3 + \theta_{44}])$$
$$r_{2,1} = -\text{Cos}[\theta_1]\text{Sin}[\theta_5] - \text{Cos}[\theta_5]\text{Sin}[\theta_1]\text{Sin}[\theta_2 + \theta_3 + \theta_{44}]$$
$$r_{2,2} = -\text{Cos}[\theta_1]\text{Cos}[\theta_5] + \text{Sin}[\theta_1]\text{Sin}[\theta_5]\text{Sin}[\theta_2 + \theta_3 + \theta_{44}]$$
$$r_{2,3} = \text{Cos}[\theta_2 + \theta_3 + \theta_{44}]\text{Sin}[\theta_1]$$
$$r_{2,4} = (L\text{Cos}[\theta_2] + L\text{Cos}[\theta_2 + \theta_3] + \text{LLCos}[\theta_2 + \theta_3 + \theta_{44}])\text{Sin}[\theta_1]$$
$$r_{3,1} = \text{Cos}[\theta_5]\text{Cos}[\theta_2 + \theta_3 + \theta_{44}]$$
$$r_{3,2} = -\text{Cos}[\theta_2 + \theta_3 + \theta_{44}]\text{Sin}[\theta_5]$$
$$r_{3,3} = \text{Sin}[\theta_2 + \theta_3 + \theta_{44}]$$
$$r_{3,4} = h + L\text{Sin}[\theta_2] + L\text{Sin}[\theta_2 + \theta_3] + \text{LLSin}[\theta_2 + \theta_3 + \theta_{44}]$$
$$r_{4,1} = 0$$
$$r_{4,2} = 0$$
$$r_{4,3} = 0$$
$$r_{4,4} = 1$$

Eqn 26

The DH parameters above represent an idealized Teachmovers with independent joints. Production Teachmovers are equipped with several coupled joints because it offers favorable characteristics for Teach Control operation and because "Designing cabling to prevent [coupling] from happening mechanically would have added undesirable complexity" (Microbot, 1984). Coupling joints also graduates kinematic analysis beyond serially linked lower-pair mechanism design to something resembling a parallel manipulator.

Mathematica is used for numeric and symbolic trigonometric simplification; code for this part is greatly enhanced since interviewing Dr. Herbert Halpern. His suggestions include the following commands:

```
in = {{a1, α1, d1, θ1}, {a2, α2, d2, θ2}, {a3, α3, d3, θ3}, {a4, α4, d4, θ4}, {a5, α5, d5, θ5}, {a6, α6, d6, θ6}}
```

```
tableMaker[x_]
:= Module[{aa, ss}, aa = Dimensions[x]; ss
= Table[SequenceForm[Link, k], {k, 1, First[aa]}]; TableForm[x, TableHeadings
→ {ss, {ai, αi, di, θi}}]]
```

```
tableMaker[in]
```

```
A[{a, α, d, θ}] := {{Cos[θ], −Sin[θ] * Cos[α], Sin[θ] * Sin[α], a * Cos[θ]}, {Sin[θ], Cos[θ]
    * Cos[α], −Cos[θ] * Sin[α], a * Sin[θ]}, {0, Sin[α], Cos[α], d}, {0,0,0,1}}
```

```
T = Dot@@(A/@in)
```

```
T = Simplify[T, Trig → True]
```

```
mat[1] = Array[Subscript[r, #1, #2]&, {4,4}]
```

```
List@@LogicalExpand[mat[1](*template*) == T(*calculation*)]//TableForm
(*use//TableForm todisplayinMathematica, omitwhencopyingtoWord*)
```

To fully appreciate why D-H parameterization is valuable consider the following. Whereas 6DOF can be described by four parameters this alone is not enough. A 6DOF manipulator having all parameters populated by non-zero terms produces a very long total transformation matrix T, $T = A_1A_2...A_{n-1}$. Mathematica produces a warning while trying to compute the matrix multiplication:

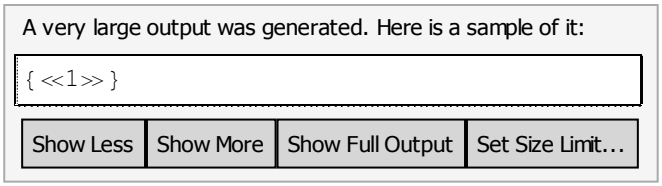


Figure A-2 Mathematica supressing the output T for a fully populated 6DOF manipultor.

T's first term is shown next; the whole 4x4 matrix requires ~70 pages to display in textual form!

$$
\begin{aligned}
T(1,1) = \ & \cos(\theta_5)(\cos(\theta_4)(\cos(\theta_3)(\cos(\theta_1)\cos(\theta_2) - \cos(\alpha_1)\sin(\theta_1)\sin(\theta_2)) \\
& + (-\cos(\alpha_1)\cos(\alpha_2)\cos(\theta_2)\sin(\theta_1) + \sin(\alpha_1)\sin(\alpha_2)\sin(\theta_1) \\
& - \cos(\alpha_2)\cos(\theta_1)\sin(\theta_2))\sin(\theta_3)) \\
& + (\cos(\alpha_3)\cos(\theta_3)(-\cos(\alpha_1)\cos(\alpha_2)\cos(\theta_2)\sin(\theta_1) \\
& + \sin(\alpha_1)\sin(\alpha_2)\sin(\theta_1) - \cos(\alpha_2)\cos(\theta_1)\sin(\theta_2)) \\
& + \sin(\alpha_3)(\cos(\alpha_2)\sin(\alpha_1)\sin(\theta_1) + \cos(\alpha_1)\cos(\theta_2)\sin(\alpha_2)\sin(\theta_1) \\
& + \cos(\theta_1)\sin(\alpha_2)\sin(\theta_2)) - \cos(\alpha_3)(\cos(\theta_1)\cos(\theta_2) \\
& - \cos(\alpha_1)\sin(\theta_1)\sin(\theta_2))\sin(\theta_3))\sin(\theta_4)) \\
& + (\cos(\alpha_4)\cos(\theta_4)(\cos(\alpha_3)\cos(\theta_3)(-\cos(\alpha_1)\cos(\alpha_2)\cos(\theta_2)\sin(\theta_1) \\
& + \sin(\alpha_1)\sin(\alpha_2)\sin(\theta_1) - \cos(\alpha_2)\cos(\theta_1)\sin(\theta_2)) \\
& + \sin(\alpha_3)(\cos(\alpha_2)\sin(\alpha_1)\sin(\theta_1) + \cos(\alpha_1)\cos(\theta_2)\sin(\alpha_2)\sin(\theta_1) \\
& + \cos(\theta_1)\sin(\alpha_2)\sin(\theta_2)) - \cos(\alpha_3)(\cos(\theta_1)\cos(\theta_2) \\
& - \cos(\alpha_1)\sin(\theta_1)\sin(\theta_2))\sin(\theta_3)) \\
& + \sin(\alpha_4)(-\cos(\theta_3)\sin(\alpha_3)(-\cos(\alpha_1)\cos(\alpha_2)\cos(\theta_2)\sin(\theta_1) \\
& + \sin(\alpha_1)\sin(\alpha_2)\sin(\theta_1) - \cos(\alpha_2)\cos(\theta_1)\sin(\theta_2)) \\
& + \cos(\alpha_3)(\cos(\alpha_2)\sin(\alpha_1)\sin(\theta_1) + \cos(\alpha_1)\cos(\theta_2)\sin(\alpha_2)\sin(\theta_1) \\
& + \cos(\theta_1)\sin(\alpha_2)\sin(\theta_2)) + \sin(\alpha_3)(\cos(\theta_1)\cos(\theta_2) \\
& - \cos(\alpha_1)\sin(\theta_1)\sin(\theta_2))\sin(\theta_3)) - \cos(\alpha_4)(\cos(\theta_3)(\cos(\theta_1)\cos(\theta_2) \\
& - \cos(\alpha_1)\sin(\theta_1)\sin(\theta_2)) + (-\cos(\alpha_1)\cos(\alpha_2)\cos(\theta_2)\sin(\theta_1) \\
& + \sin(\alpha_1)\sin(\alpha_2)\sin(\theta_1) \\
& - \cos(\alpha_2)\cos(\theta_1)\sin(\theta_2))\sin(\theta_3))\sin(\theta_4))\sin(\theta_5)
\end{aligned}
$$

Trying to simplify T yields another error:

> Simplify::time: Time spent on a transformation exceeded 300 seconds, and the transformation was aborted. Increasing the value of TimeConstraint option may improve the result of simplification. >>

Figure A-3 Mathematica warning produced while trying to simplilfy a "full T."

It is rather ingenious that attaching frames to links following DH1 and DH2 listed above often results in one or more parameter(s) equating to zero, which greatly simplifies the math.

Inverse Kinematics derived using Artificial Neural Network Fuzzy Inference

In the previous section, forward kinematics are derived using a standard convention known as Denavit-Hartenberg parameterization. Solving inverse kinematics of very large systems can be intractable. Using a more modern method that discovers kinematic relations in lieu of geometry and trigonometry is useful and is the topic of this section.

While it is possible to solve Eqn 26 iteratively, those 16 equations are nonlinear and almost one-way; that is, it is infinitely easier to check a candidate solution for satisfiability than to produce a satisfying solution. For simple systems, traditional closed formed solutions can be found geometrically that are solvable in a few clock cycles on any mediocre computer. Upgrading to a modest computer opens the door to computationally discovering inverse kinematics.

"There are several methods for solving [inverse kinematics] [computationally], coming originally from robotic applications" writes Samual Buss in his survey (Buss, 2004). "These include cyclic coordinate descent methods (L. C. T. Wang & Chen, 1991), pseudoinverse methods (Whitney, 1969), Jacobian transpose methods [(Balestrino, De Maria, & Sciavicco, 1984),(Wolovich & Elliot, 1984)], the Levenberg Marquardt damped least squares methods [(Wampler, 1986), (Nakamura & Hanafusa, 1986)], quasi-Newton and conjugate gradient methods [(L. C. T. Wang & Chen, 1991),(Zhao & Badler, 1994),(Deo & Walker, 1993)], and neural net and artificial intelligence methods [(Grzeszczuk & Terzopoulos, 1995),(Lendaris, Mathia, & Sacks, 1999), (Oyama, Chong, Agah, Maeda, & Tachi, 2001), (Ramdane-Cherif, Daachi, Benallegue, & Levy, 2002), (Grzeszczuk, Terzopoulos, & Hinton, 1998), (Jordan & Rumelhart, 1992), (Tevatia & Schaal, 2000), (D'Souza, Vijayakumar, & Schaal, 2001)]." An artificial neural network fuzzy inference system is used here.

Neural Networks are: a computer architecture in which a number of processors are interconnected in a manner suggestive of the connections between neurons in a human brain and which is able to learn by a process of trial and error (Mish, 1995). In this thesis, a neural net is treated as a black box.

The procedure to use the Neural Network black box is as follows: start with feeding in reference/training data; let the box train itself; then query. Ideally interpolated responses will accurately represent the training data. The concentration of this section will be on generating training data, training/testing of artificial neural net parameters and optimizing.

Teachmover arms have 5DOF, plus a gripper, so they can draw on many arbitrary surfaces. Since it is easier to draw on planes and it costs nothing to markup recycled paper, the canvas will be always be recycled paper taped to a table. This table will be the XY plane.

The 1st step in using a neural net is to acquire/generate reference/training data. Here the domain is limited to a rectangle. Limiting the scope to a 2D rectangle is a necessary condition because MATLAB's `anfis` function is zero or first order only (The Mathworks, 2010). Training data is generated using a function written for this thesis named `cart_cmd`, which outputs motor encoder values given a desired position. `cart_cmd` is called many times, with different parameters, and each input/output combination is saved in a training matrix. The philosophy is: forward kinematics are much easier to solve than inverse kinematics, so generate a mesh of training points using forward kinematics, and use the artificial neural network fuzzy inference system to solve the inverse kinematics. The syntax of `cart_cmd` is shown next.

```
% Command needed to go from home to pos
% [J1,J2,J3,J4,J5,J6] = cart_cmd([pos]);
% pos = [x,y,z,p,r,g];
% J = [base,shoulder,elbow,R wrist, L wrist,grip]
```

A sample of the training matrix is shown in Table A-2. It is a mesh of points on the XY-plane inside a 2"x2" rectangle centered about point (0,7) on the calibration sheet. On the left is the commanded position and pose and to the right are the motor encoder values. All together the training matrix is 81x12.

x (inch)	y (inch)	z (inch)	pitch (deg)	roll (deg)	grip	base	shoulder	elbow	R wrist	L wrist	grip
5	1.5	0	-90	0	0	327	-477	1149	455	312	1149
5	2	0	-90	0	0	428	-482	1132	477	291	1132
5.5	-2	0	-90	0	0	-393	-491	1086	298	469	1086

Table A-2 A sample of the training matrix.

The 2nd step, to train the artificial neural network fuzzy inference system, requires only one MATLAB command, `anfis`. Given as:

```
[fis,error,stepsize,chkFis,chkErr] = ...
anfis(trnData,numMFs,trnOpt,dispOpt,chkData,optMethod)
```

The command has many options. Here: `fis, error` plus `trnData, numMFs, trnOpt` and `dispOpt` are used for training the picture drawing robot. Each of the options is well documented in the help index. Syntax for one call, using this truncated set of options, is:

```
[anfisC,error(:,mf,C)] = anfis([cmd(:,1) cmd(:,2) data(:,C)], mf, epochmax);
```

Where, according to the help index (The Mathworks, 1998):

- `anfisC` is the FIS structure whose parameters are set according to a minimum training error criterion.
- `error(:,mf,C)` is an array of root mean squared errors representing the training data error signal and the checking data error signal, respectively. The function only returns chkErr when you supply chkData as an input argument.
- ([cmd(:,1) cmd(:,2) data(:,C)] is the training data set. This matrix contains data input in all but the last column. The last column contains a single vector of output data.
- `mf` is the number of membership functions. Use numMFs, an integer scalar value, as the second argument to anfis when you do not already have a FIS to train, and you want anfis to build a default initial FIS using your data. Each input and output to this FIS is characterized by one or more membership functions. Specify the number of membership functions in numMFs.
- `epochmax` is trnOpt(1) training epoch number
- `dispOpt(1:4)` are all true.

The number of membership functions (MFs) and training epoch number are analogous to the number of people tasked with a mission and the amount of time they have to complete it, respectively. A big team and more time are helpful however these two parameters compete for computational resources. Another concern is how to test when MF have saturated their learning potential. Plots demonstrating these ideas are presented in the next few pages.

Training the base motor exemplifies the typical four step procedure. Step 1, run `anfis`. Running `anfis` and looking at `anfisBASE` reported error, Figure A-4, there is no apparent end to its "goodness," more members and time appear to decrease error indefinitely. While this may be true, the intention is to decrease encoder count and therefore positional error.

In step 2, a new test compares `anfis` to the analytic solution. `anfisBASE` is queried to predict points that interpolate the training data, the interpolated data set differs from the training data in that it is three times denser. The test is repeated several times, each time the number of membership functions is increased by one, while the number of epochs is kept constantly high. Test results are giving graphically in Figure A-5 to Figure A-10.

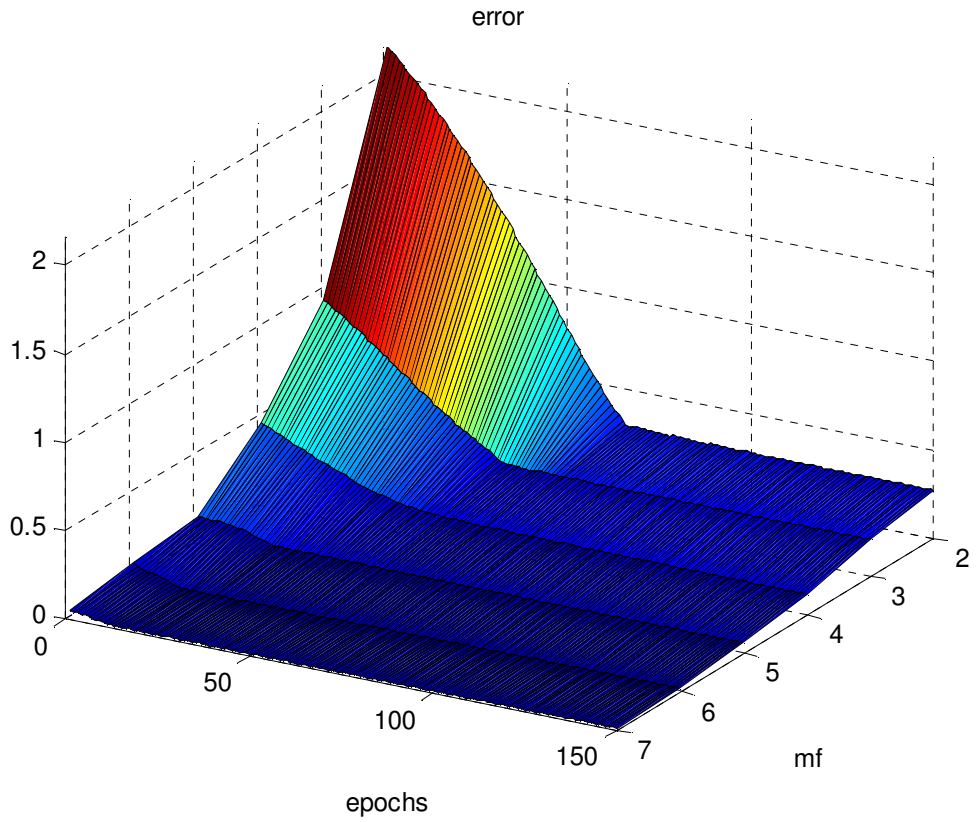


Figure A-4 anfisBASE error as reported by anfis.

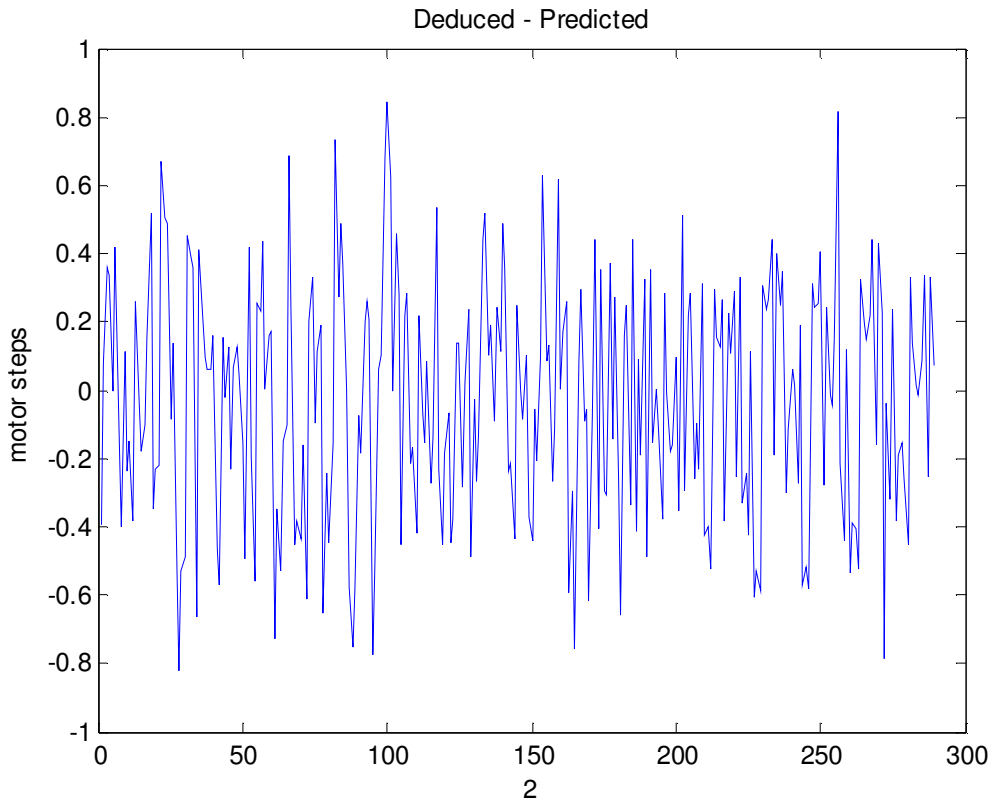


Figure A-5 (above) Motor step error at membership funcs = 2 and epochs = 150.

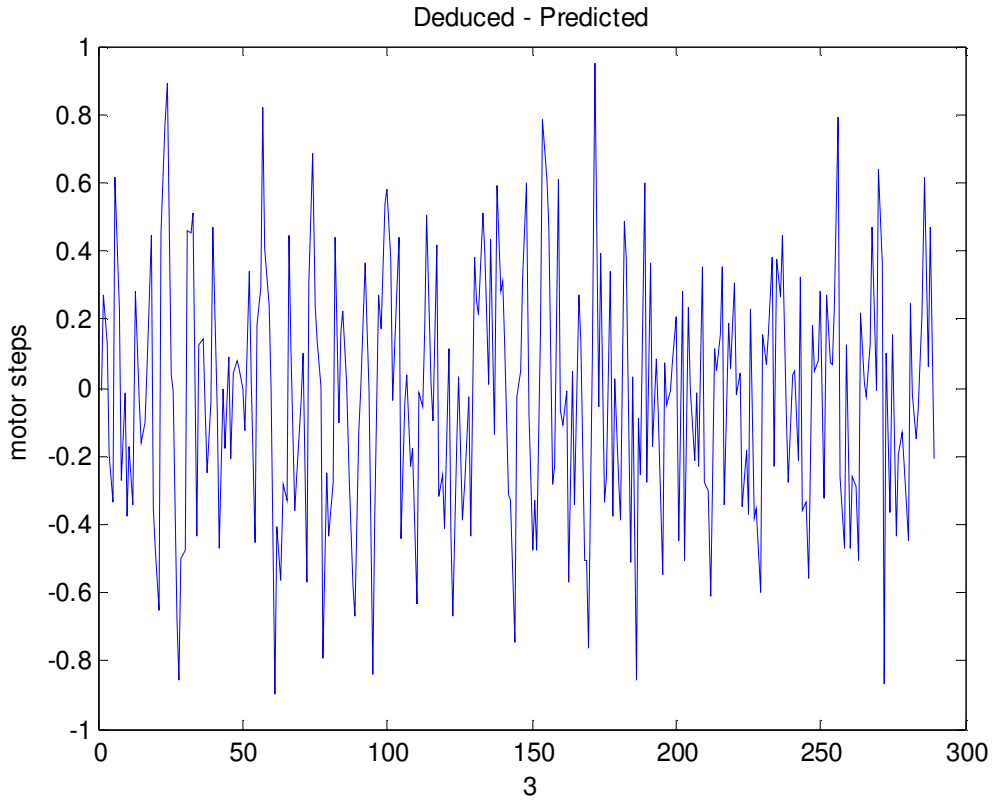


Figure A-6 (above) Motor step error at membership funcs = 3 and epochs = 150.

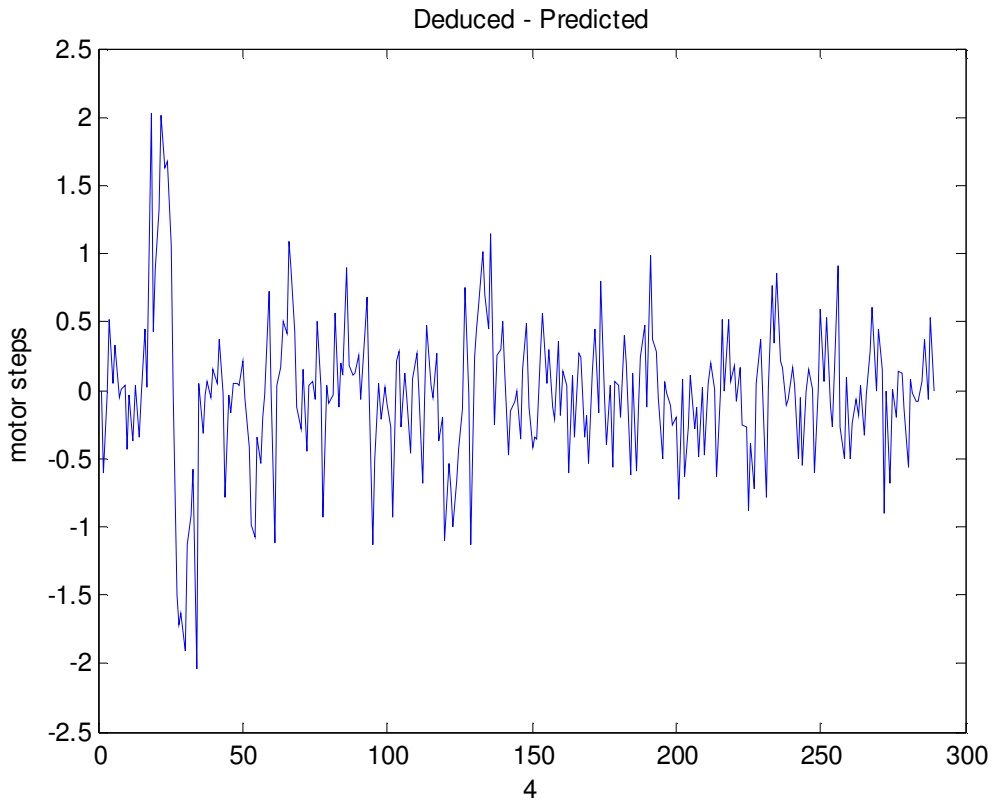


Figure A-7 (above) Motor step error at membership funcs = 4 and epochs = 150.

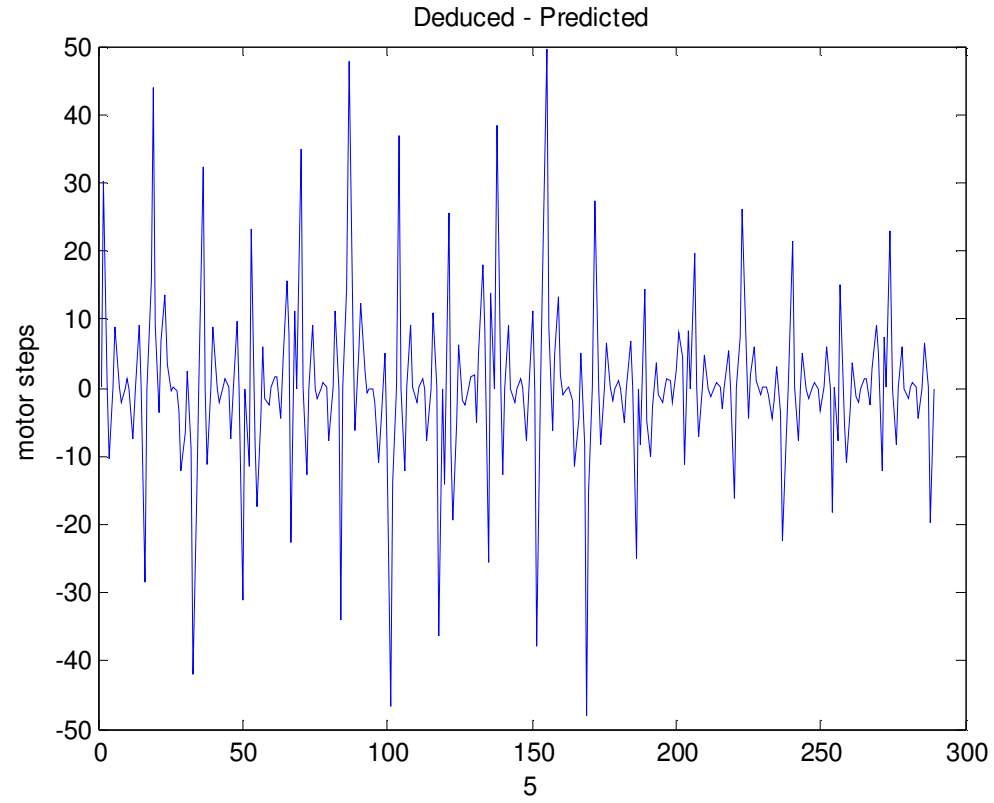


Figure A-8 (above) Motor step error at membership funcs = 5 and epochs = 150.

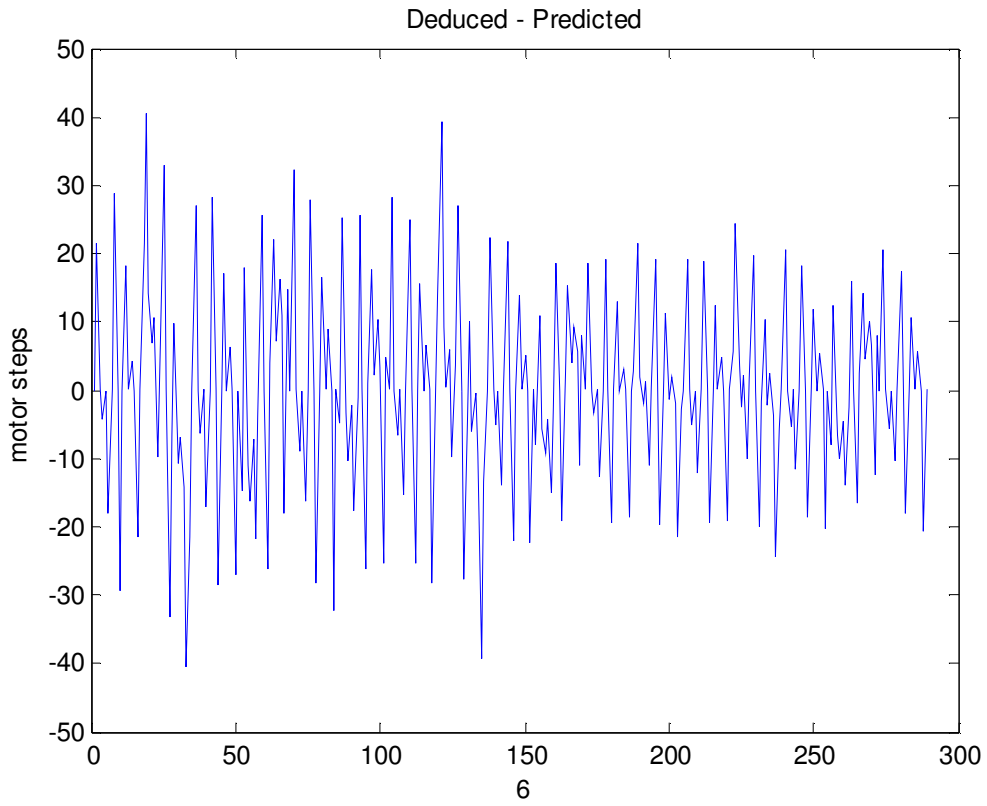


Figure A-9 (above) Motor step error at membership funcs = 6 and epochs = 150.

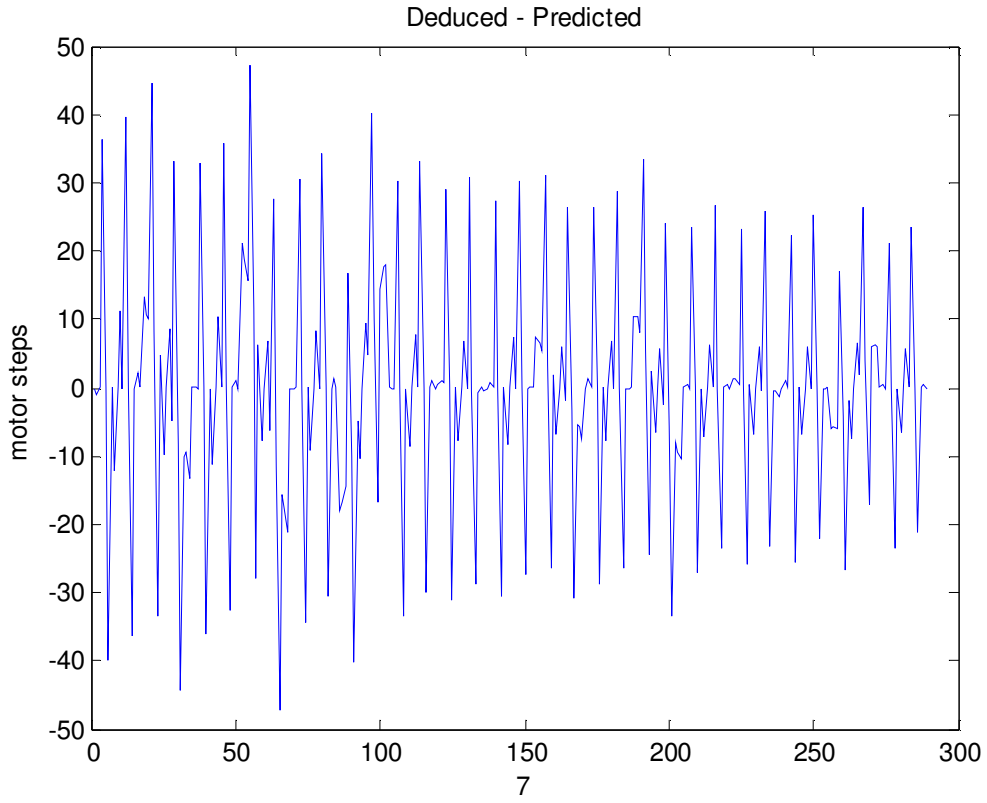


Figure A-10 (above) Motor step error at membership funcs = 7 and epochs = 150.

Scale alone demonstrates two or three membership functions are best. Comparing standard deviation proves two membership functions is best for the plotted/observed graphs.

In step 3, computation time also demonstrates training two membership functions is faster than training three. Each new member appears to increase training time exponentially, Figure A-11.

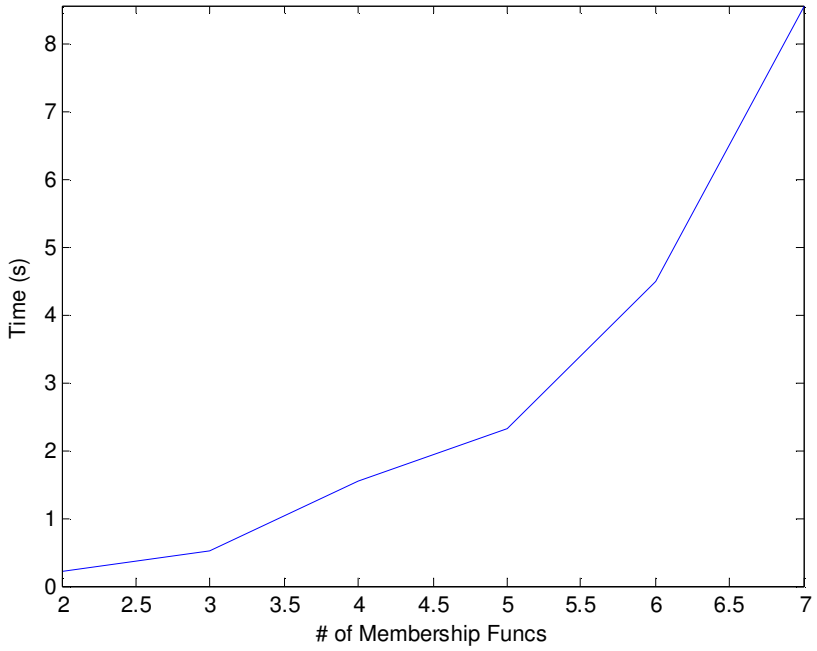


Figure A-11 Time vs # of membership functions.

Optimizing is the 4th and final step. Thus far an arbitrarily large number of epochs have been used for training. Previously, this was set to 150 epochs. More detailed inspection of Figure A-4 is required to determine exactly where the knee is and saturation occurs. In Figure A-12 it is clear knee is at 68 and error turns into a flat line around 100 epochs.

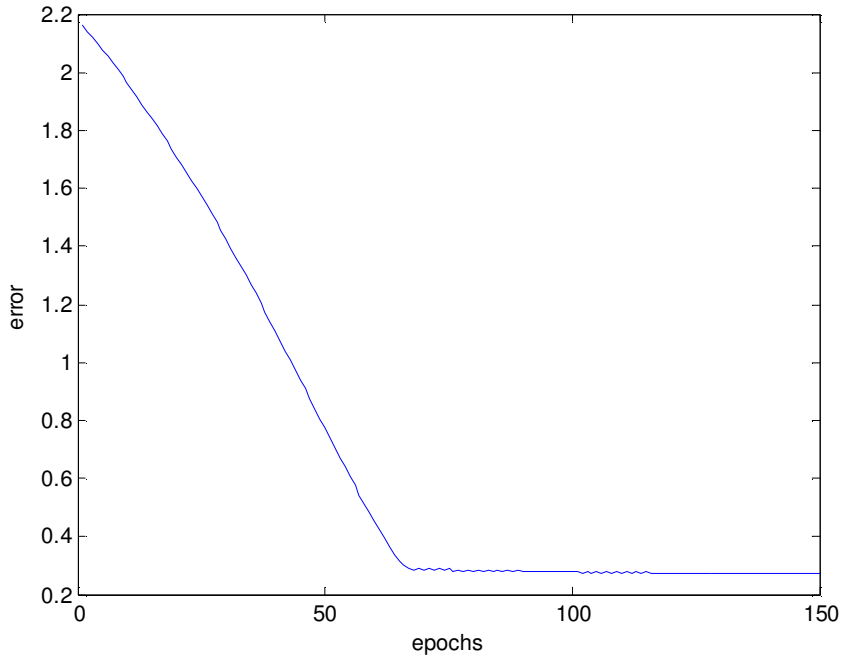


Figure A-12 Error vs # of epochs given 2 membership functions.

Most `anfis` 'reported error' surface plots monotonically decrease as membership function and epochs increase as shown in Figure A-4. Exceptions to this 'rule' occur twice. Both elbow and wrist plots contain a local maximum whereas none should occur, Figure A-13 and Figure A-14. A possible explanation for this behavior could be the fact that Teachmovers are 5DOF systems and `anfis` cannot handle such a highly non-linear systems.

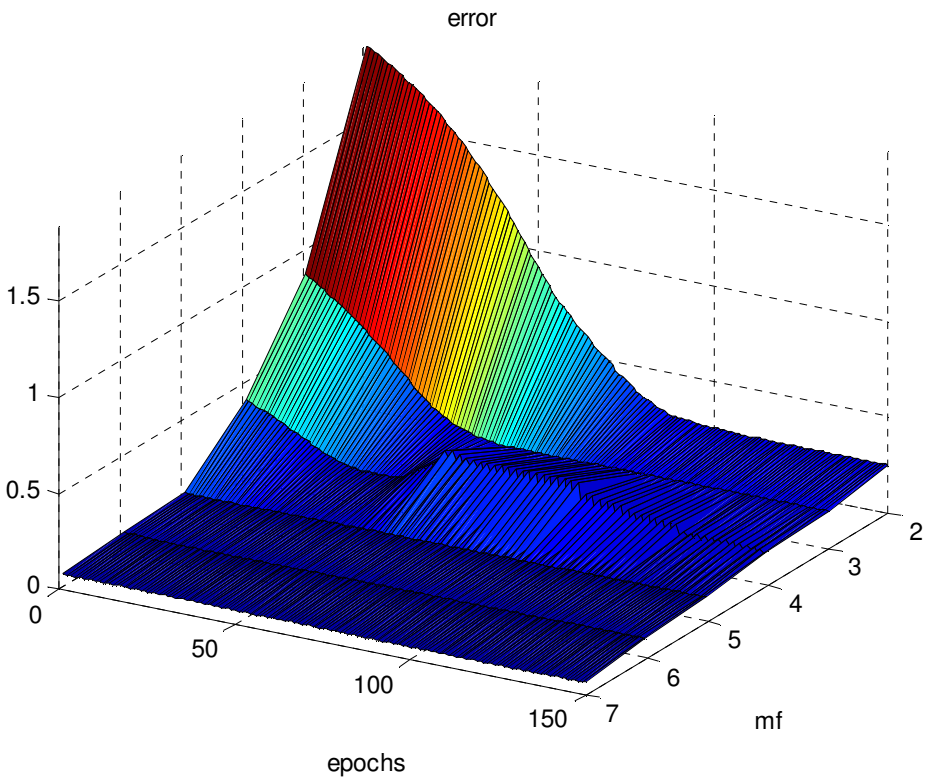


Figure A-13 anfisELBOW error reported by anfis. Notice the bump.

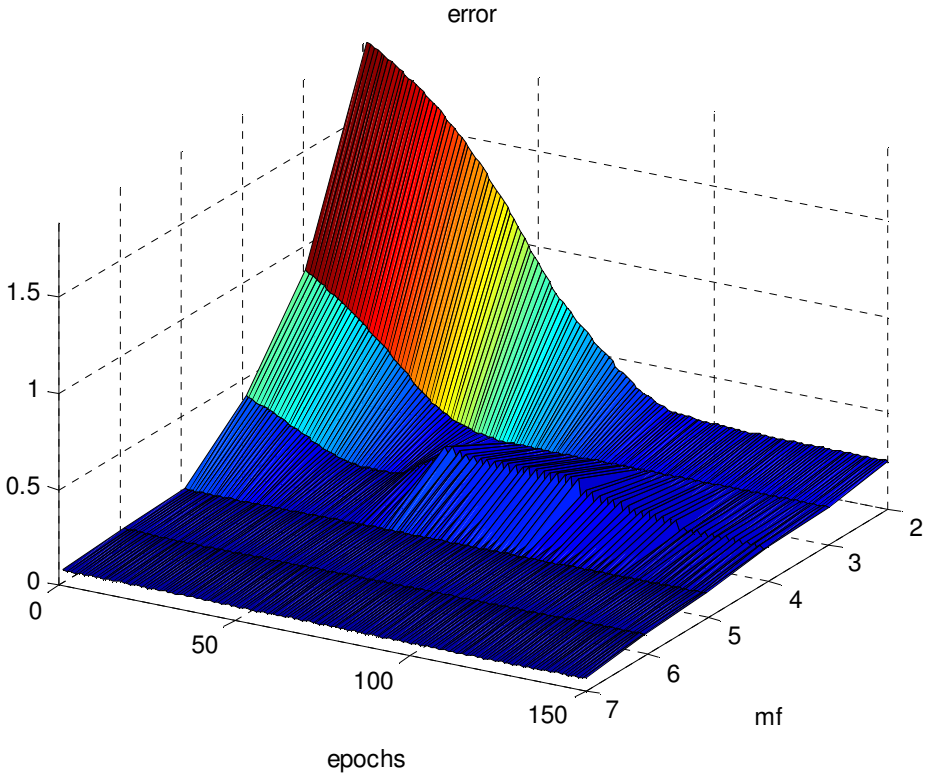


Figure A-14 anfisWRIST error reported by anfis. Notice the bump.

Steps one through four are repeated for each joint motor. The optimal number of membership functions and epochs for each joint motor are tabulated in Table A-3.

Motor	Membership function count	Epochs
base	2	100
shoulder	2	100
elbow	2	125
pitch	2	100
roll	3	150
grip	2	250

Table A-3 Optimal ANFIS parameters.

Summary

This concludes Appendix A on Generalized Kinematics. The math covered in this section is important for several reasons. First, it frames the kinematics and inverse kinematics within a standardized notation. Taking the equations derived in Chapter 4 and reformulating them in this way enables the work to be quickly compared and implemented. Second, using the compact Denavit-Hartenberg parameterization technique reduces the computation time by using matrix operations. Third, the ANFIS method shown here can be used to solve other, larger systems where analytic solutions are intractable. Some code for this modern ANFIS approach to kinematics can be found at (The MathWorks, 2010).