2010

A study of haptic interactions with an under actuated robot in three dimensions

Melissa Mae Wickham
Iowa State University

A study of haptic interactions with an under actuated robot in three dimensions

by

Melissa Mae Wickham

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Mechanical Engineering

Program of Study Committee:
Greg Luecke, Major Professor
LeAnn Faidley
Yan-Bin Jia

Iowa State University

Ames, Iowa

2010

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

As haptic devices become more common, the afford-ability, complexity, and size of the device are all factors in considering appropriate applications. A popular haptic device is the PHANTOM Omni, a six degree of freedom (6 DOF) positioning manipulator with 3 DOF force feedback. To increase the usability of the class of under actuated devices that include the common PHANTOM Omni, an impedance requiring full 6 DOF force feedback control law for under actuated devices based on a virtual mechanism (VM) was developed. The goal of this control law is to mask the un-actuated joints of a physical robot manipulator (RM) using force feedback on the actuated joints. The control theory includes using a VM as a constraint for the RM. Weighted matrices and the pseudo inverse Jacobian control law place the VM as to minimize the error between the current RM position and somewhere on the path of the VM. A pseudo inverse Jacobian control law is used to generate a force to send to user of the RM. The theory is tested using the PHANTOM Omni with six cases where the importance of the position error and orientation error for the manipulator are varied.

CHAPTER 1. OVERVIEW

Haptic feedback is sent to the user as touch. The forces for haptic interaction can be created using a computer and a robotic device and generated by a computer along with graphics to create an interactive virtual environment. Within a virtual environment haptics can be used to create, manipulate, or explore. Haptics can also be used in teleoperation and simulation environments. Medicine is a common place for teleoperation and simulation, as surgeries can be simulated for surgeons to practice, and minimal invasive surgery is now possible through teleoperation. Haptics are being used in consumer goods such as cell phones that give a physical response to users as they type on a touch screen, and on video game controllers to allow users to feel effects of the virtual environment they are in. While haptics are becoming more common in many fields and consumer goods there are still limitations on the devices such as cost, functionality, and complexity.

1.1 Motivation

As haptic devices become more common, the afford-ability, complexity, and size of the device are all factors in considering appropriate applications. In order to fully sense a three dimensional environment a haptic device is required to have six degree of freedom (6 DOF) both in positioning and force feedback. To accomplish this the device would need to have a minimum of six sensors and six actuators, increasing the size, complexity, and cost of any haptic device. 6 DOF devices with 6 DOF positional with 3 DOF force feedback allow the user to see the position and orientation of the device, but only feel tactile feedback in 3 DOF. The PHANTOM Omni is a 6 DOF sensing haptic device, and a popular choice of haptic device due to its ease of use, small footprint, and inexpensive price. The drawback to this device is the missing torque

feedback; only three Cartesian forces can be sent back to the user. Work has been done [3] to study the need for combined torque and force feedback. Some applications, like drawing and tracing require only force feedback [2], while in other cases like surgical simulators, a torque sensation is required. These 6 DOF devices with sensing and force feedback at every joint are often not affordable. To increase the usability of the class of under actuated devices that include the common phantom design applications, an impedance requiring full 6 DOF force feedback control law for under actuated devices based on a virtual mechanism was developed [1, 11]. To continue this work, six cases are explored using this theory in two and three dimensional space.

1.2 Preview of Report

In Chapter 2, a general overview of robotics will be presented. This includes manipulator kinematics including: forward kinematics, inverse kinematics, and rate kinematics; and a definition of a Jacobian matrix are given. This will later be used to aid in the development of control theory and implementation. A brief overview of the PHANTOM Omni will be given, the PHANTOM Omni is the haptic device that will be used in experiments for this report. In Chapter 3, the control theory, including specific implementation techniques, is described. In Chapter 4, experimental results from testing the theory in one degree of freedom, then testing in three degrees of freedom are presented with a brief discussion of each case. In Chapter 5, an overall discussion of the control law application will be given. This report will cover theory and results of the control approach applied to the PHANTOM Omni and will include only C++ code developed for implementing theory on the haptic device. The C++ code is located in Appendix D. For a better understanding of how to render the device in C++ code see [8, 9]. An important part of using haptic devices in a virtual environment are the graphics for the user to gain more channels of feedback. Implementation of graphics for each case are not included, for more information on graphics programming see [10].

CHAPTER 2. REVIEW

The control approach presented in this report depends on the details of the theoretical development of the robotic positions and rate kinematics, thus a brief review of robotics will be presented in the following chapter. A brief overview of the PHANTOM Omni, the haptic device used for implementing control theory, is also given.

2.1 Review of Robotics

This section presents a general overview of robotics. First parameters of the manipulator need to be defined, Denavit Hartenberg notation will be used. Next, general manipulator kinematics including: forward kinematics, inverse kinematics and rate kinematics are described. A Jacobian matrix will be defined, and finally a brief overview of force and torque relations. Spacial descriptions are used to relate one robotic frame to another, a review is given in Appendix A. For further understanding of general robotics in terms of kinematics and control see [6]. A general overview of linear algebra may be needed to understand matrix properties and basic mathematical operations, see [5]. Also trigonometric identities are used for simplification, a list of trigonometric identities is included in Appendix C.

2.1.1 Forward kinematics

Kinematics tells how motion will occur in a manipulator, without including forces or dynamics of the manipulator. Forward kinematics (fk) finds the relations between each joint that makes up the robotic manipulator. If all parameters of every joint are known for a certain manipulator, the fk will find the placement of the end-effector of that manipulator.

2.1.1.1 Denavit-Hartenberg Notation

A link, i, can be described in four parameters (α_i, a_i, d_i, and θ_i). Three of these four are fixed link parameters while one is a joint variable. For a revolute joint, the joint variable is considered to be θ_i, while for a prismatic joint it is d_i. Assigning of the four parameters for each link is done using Denavit-Hartenberg (DH) notation.

Denavit-Hartenberg notation

a_i, link length, distance from z_i to z_{i+1} measured along x_i

α_i, link twist, angle from z_i to z_{i+1} measured about x_i

Description of joint

d_i, link offset, distance x_{i-1} to x_i measured along z_i

θ_i, joint angle, angle x_{i-1} to x_i measured about z_i

The DH frame assignment for link and joint parameters are described in Figure 2.1.

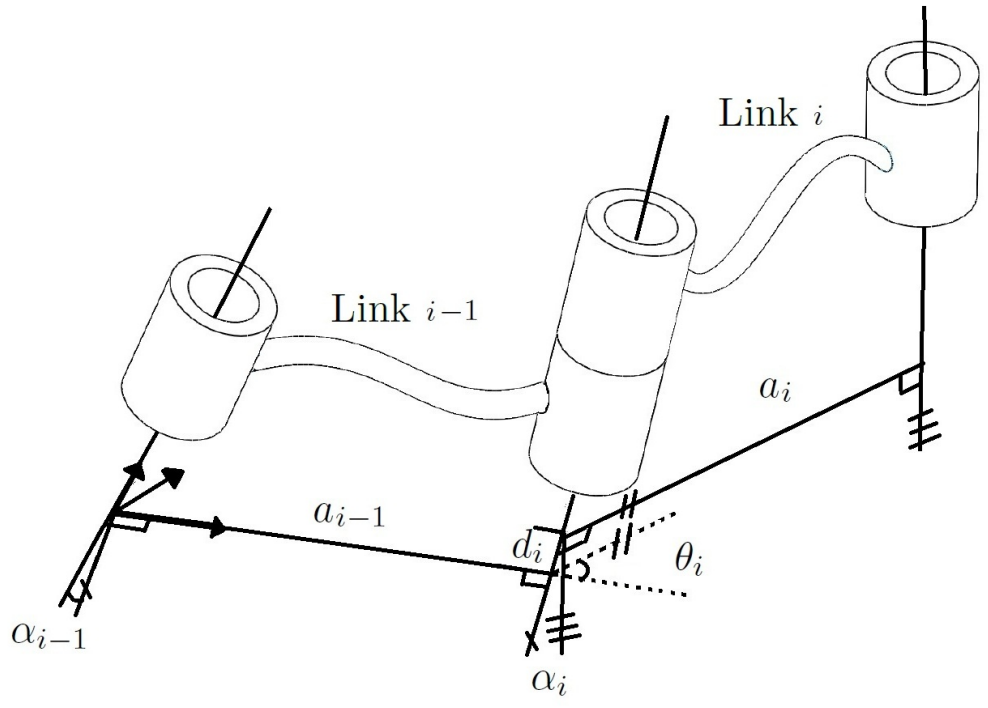


Figure 2.1 Link Description

To better understand using the DH parameters, a solution of fk for a two link manipulator, shown in Figure 2.2 will be presented. The DH Parameter assignment for the two link manipulator can be seen in Table 2.1.

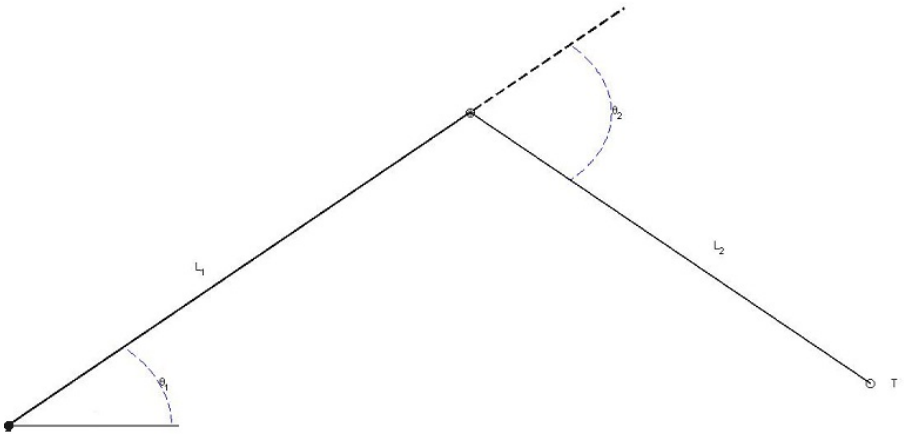


Figure 2.2 Two Link Manipulator

Table 2.1 Denavit Hartenberg Parameters of Two Link Manipulator

i	a_{i-1}	α_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	L_1	0	0	θ_2
3	L_2	0	0	0

The next step in defining a manipulator is to determine the transforms between joint frames.

2.1.1.2 Transforms

Using the assigned DH parameters, the space between one joint to another joint attached by a link can be defined by a transformation matrix. Every transformation can be described using $R_X(\alpha i - 1)$, $D_x(a_{i-1})$, $R_Z(\theta_i)$, and $D_Z(d_i)$. A general transformation from one joint

frame to the next joint frame can be found using the assigned DH parameters in

$$ {}^{i}_{j}T = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & a_{i-1} \\ \sin(\theta_i)\cos(\alpha_{i-1}) & \cos(\theta_i)\cos(\alpha_{i-1} & -\sin(\alpha_{i-1}) & -\sin(\alpha_{i-1})d_i \\ \sin(\theta_i)\sin(\alpha_{i-1}) & \cos(\theta_i)\sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & \sin(\alpha_{i-1})d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.1} $$

After DH parameters are defined for the manipulator the transformations between each joint are found. In the example of the two link manipulator, the transformations between joints are as follows.

$$ {}^{0}_{1}T = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2} $$

$$ {}^{1}_{2}T = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & L_1 \\ \sin(\theta_2) & \cos(\theta_2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3} $$

$$ {}^{0}_{3}T = \begin{bmatrix} 1 & 0 & 0 & L_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4} $$

Thus, the fk of the two link manipulator is,

$$ {}^{0}_{3}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T \tag{2.5} $$

Due to space limitations in writing the solution, $sin(\theta)$ will be replaced with sθ and cos(θ) with cθ.

$$ {}^{0}_{3}T = \begin{bmatrix} c\theta_1c\theta_2 - s\theta_1s\theta_2 & -c\theta_1s\theta_2 - c\theta_2s\theta_1 & 0 & L_2(c\theta_1c\theta_2 - s\theta_1s\theta_2) + L_1c\theta_1 \\ c\theta_1s\theta_2 + c\theta_2s\theta_1 & c\theta_1c\theta_2 - s\theta_1s\theta_2 & 0 & L_2(c\theta_1s\theta_2 + c\theta_2s\theta_1) + L_1 * s\theta_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.6} $$

2.1.2 Inverse kinematics

From the previous section, it was shown that fk of a manipulator finds the final transformation and position using the joint variables and fixed link parameters. Inverse kinematics (ik) uses the final transformation (goal frame) and fixed link parameters to find all joint variables for a manipulator. There is no generic solution of inverse kinematics that works for all manipulators, as it depends on the geometry of the manipulator. A goal frame contains the desired orientation and position of the end-effector of the manipulator. There may not be a unique solution, or any solution at all for a given goal frame. If the goal frame lies outside the workspace of the manipulator, there will be no joint angles that could cause the end-effector to arrive at the goal frame, thus no solution is possible. A unique solution only occurs if there is only one possible set of joint angles to reach the goal frame. When programming the trajectory of a manipulator it is important to choose the best, or shortest overall solution for ik.

2.2 Rate kinematics

To solve for rate kinematics the velocity starts at the base frame, and propagates, or moves through each frame based on the previous frames linear and angular velocity. Once the final frame is reached, a rotation matrix from the base frame to the final frame may by used to find the velocity of the final frame in reference to the base frame. The equations for finding velocity propagation through links are now presented. For finding a vector dot product, a skew symmetric matrix is used.

$$S = \begin{bmatrix} 0 & -\Omega_Z & \Omega_Y \\ \Omega_Z & 0 & -\Omega_X \\ -\Omega_Y & \Omega_X & 0 \end{bmatrix} \tag{2.7}$$

Using DH notation, rotation of a frame happens about the Z axis. A Z axis in it's own frame will always be a unit vector along the Z axis

$${}^{i+1}\hat{(Z)}_{i+1} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.8}$$

Using Eq(2.7) to represent a dot product, the rotational velocity can be written as a 3 X 3 matrix.

$$ {}^{i}\omega_i = \begin{bmatrix} \omega_X \\ \omega_Y \\ \omega_Z \end{bmatrix} = \begin{bmatrix} 0 & -\omega_Z & \omega_Y \\ \omega_Z & 0 & -\omega_X \\ -\omega_Y & \omega_X & 0 \end{bmatrix} \tag{2.9} $$

For a revolute joint, angular velocity is given by

$$ {}^{i+1}\omega_{i+1} = {}^{i+1}_{i} R\, {}^{i}\omega_i + \dot{\theta}^{i+1}_{i+1}\hat{(Z)}_{i+1} \tag{2.10} $$

and linear velocity is

$$ {}^{i+1}\nu_{i+1} = {}^{i+1}_{i} R({}^{i}\nu_i + {}^{1}\,\omega_i \times {}^{i}\,P_{i+1}) \tag{2.11} $$

Rate propagation does not consider dynamics or the manipulator structure, but is dependent on the geometry of the structure. Rate kinematics can be used to find the Jacobian of the manipulator.

2.2.1 Jacobian

A Jacobian matrix contains the partial differentiable equations of each joint stored in a matrix. Consider Y to contain n functions of m independent variables X.

$$ \begin{aligned} y_1 &= f_1(x_1, x_2, \ldots, x_m) \\ y_2 &= f_2(x_1, x_2, \ldots, x_m) \\ &\vdots \\ y_n &= f_n(x_1, x_2, \ldots, x_m) \end{aligned} $$

The derivative of each function can be written as

$$ \begin{aligned} \delta y_1 &= \frac{\delta f_1}{\delta x_1}\delta x_1 + \frac{\delta f_1}{\delta x_2}\delta x_2 + \ldots + \frac{\delta f_1}{\delta x_m}\delta x_m \\ \delta y_2 &= \frac{\delta f_2}{\delta x_1}\delta x_1 + \frac{\delta f_2}{\delta x_2}\delta x_2 + \ldots + \frac{\delta f_2}{\delta x_m}\delta x_m \\ &\vdots \\ \delta y_n &= \frac{\delta f_n}{\delta x_1}\delta x_1 + \frac{\delta f_n}{\delta x_2}\delta x_2 + \ldots + \frac{\delta f_n}{\delta x_m}\delta x_m \end{aligned} $$

This can be rewritten, defining the Jacabian $J(X)$ as a nxm matrix of partial differential equations.

$$\delta Y = \frac{\delta F}{\delta X}\delta X = J(X)\delta X \tag{2.12}$$

For the robotic manipulator the Jacobian can be found from rate kinematics, a Jacobian written in base frame $\{0\}$ would be

$$^0J(\Theta)\dot{\Theta} = ^0 v = \begin{bmatrix} ^0\nu \\ \\ ^0\omega \end{bmatrix} \tag{2.13}$$

2.2.1.1 Force

To describe the Jacobian in the force domain

$$F^T\delta X = \tau^T\delta\theta \tag{2.14}$$

Using Eq(2.12), Eq(2.14) can be rewritten as

$$F^TJ\delta\theta = \tau^T\delta\theta \tag{2.15}$$

Which simplifies to

$$\tau = J^TF \tag{2.16}$$

Using error force control, force can be defined as

$$F = GAIN\delta X \tag{2.17}$$

The transpose-Jacobian control is given by

$$\tau = J^T\delta X = J^TGAIN\delta X \tag{2.18}$$

while the inverse-Jacobian control is defined by

$$\tau = GAIN\delta\theta = GAINJ^{-1}\delta X \tag{2.19}$$

These are considered intuitive schemes of Cartesian control [6].

2.3 PHANTOM Omni

The PHANTOM Omni is an inexpensive portable six degree of freedom positional sensing haptic device made by SensAble Technologies. It is considered an under actuated six degree of freedom (DOF) robot, it is classified as such as it has six revolute joints. The first three joints define the position of the end effector, consider Cartesian space (x, y, z). The last three joints are attached at a point, and define the orientation of the end effector(roll, pitch, yaw). The PHANTOM Omni is considered under actuated due to that only the first three joints (x, y, z) are actuated. This means that although both position and orientation can be manipulated by the user and sensed by the manipulator, force can only be sent back to the first three joints, or position. To program the PHANTOM Omni, Visual Studio with C++ language was used. Analysis of the phantom involved find the DH parameters, forward kinematics, inverse kinematics, and rate kinematics, this is included in Appendix B.

CHAPTER 3. CONTROL THEORY

The PHANTOM Omni is an under actuated robotic manipulator, the user can freely change the position and orientation of the end effector, although it is not possible to apply force directly to the last three joints: torque around the three orientation joints. This limits the effective use of the device in applications where torque feedback is required. In order to overcome this limitation, it is necessary to generate feedback forces in the three actuated linear joints in such a way as to mask the missing rotational joint torques. This will give the user a feel similar to moment forces being applied based on the orientation of the end effector. To accomplish this, a control approach using a virtual manipulator (VM) for a constraint force will be studied. First a general overview of the control theory is presented, next specific theory for a VM constrained in two dimensional space, and finally the VM constrained in three dimensional space will be presented. For each of these cases it will be shown that the VM approach can be used to mask the missing actuation of the under actuated device.

3.1 Virtual Manipulator For Constraint Force

The VM is a computer generated manipulator that moves in virtual 6 degree of freedom (DOF) space without full motion kinematics. The physical robot manipulator (RM) is located in real 6 DOF space, and has full motion kinematics. For this haptic feedback approach, the VM kinematics will be used to control the position of the RM. For full 6 DOF haptic interaction, the VM will be modeled as a 5 DOF manipulator with two prismatic joints and three revolute joints. In a physical sense this can be thought of as a probe on a surface with the two prismatic joints representing friction, and the three revolute joints representing the orientation of the probe [1]. First, the forward kinematics (fk) of the VM are developed in 6 DOF space. The fk

yield a transformation matrix,

$$ {}^{0}_{t}T = \begin{bmatrix} {}^{0}\hat{X}_t & {}^{0}\hat{Y}_t & {}^{0}\hat{Z}_t & {}^{0}P_{tORG} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1} $$

that contains the position ${}^{0}P_{tORG}$ as a point represented in Cartesian coordinates, and orientation about each axis represented by ${}^{0}\hat{X}_t$, ${}^{0}\hat{Y}_t$, and ${}^{0}\hat{Z}_t$.

The position of the end effector as a function of fk for given joint angles can be represented in general 6 DOF space as

$$ X = \Phi(\theta) = \begin{bmatrix} x \\ y \\ z \\ \theta_x \\ \theta_y \\ \theta_z \end{bmatrix} \tag{3.2} $$

The general equation for force control generates a torque for each joint based on the Jacobian of that manipulator and the specified force vector. The relation between torque required by the VM τ_v, the Jacobian of the VM J_v, and the force F_H can be found using the Jacobian relation:

$$ \tau_v = J_v^T F_H \tag{3.3} $$

Because the VM has reduced DOF kinematics, there are only five joint torques. There are still six generalized forces applied, three linear forces and three rotational torques. To get a generalized force for a given set of joint torques, the equation must be rearranged and the inverse of the Jacobian must be found. A matrix must be square and non-singular to be invertible. The size and rank of the VM Jacobian vary by the particular constraint it is describing, so it is impossible to guarantee a Jacobian will be invertible. Thus a Moore-Penrose pseudo inversion will be used,

$$ (J_v^T)^{-1} = J_v(J_v^T J_v)^{-1} \tag{3.4} $$

Now that the VM Jacobian can be inverted, the force at the end effector, corresponding to a

"given set" of VM joint torques, can be described in a general sense by F_H^*

$$F_H^* = J_v(J_v^T J_v)^{-1}\tau_v \tag{3.5}$$

Substituting Eq(3.3) into Eq(3.5) yields

$$F_H^* = J_v(J_v^T J_v)^{-1}J_v^T F_H \tag{3.6}$$

which gives a general force representation of the VM based on the force applied to the end effector by the user of the RM.

To find how much force needs to be applied to oppose the motion of the RM, the difference of the force applied by the user and the general force from the VM is found,

$$\begin{aligned} f_{motion} &= F_H - F_H^* \\ &= F_H - J_v(J_v^T J_v)^{-1}J_v^T F_H \\ &= (I - J_v(J_v^T J_v)^{-1}J_v^T)F_H \end{aligned} \tag{3.7}$$

The force can be described by a general error feedback law show as

$$F_H = (K_p e + K_v \dot{e}) \tag{3.8}$$

using the placement of the VM as the desired origin.

The position error, e, is the difference between the VM position , X_d and the RM position, X_R.

$$e = X_d - X = X_V - X_R \tag{3.9}$$

and the velocity error, $\dot{e}$, is defined as the difference between the VM and RM velocities.

$$\dot{e} = \dot{X}_d - \dot{X} = \dot{X}_V - \dot{X}_R \tag{3.10}$$

The VM is virtual, so the velocity can be set to any desired value. Setting the desired velocity to zero would increase the damping in the system. Assuming the desired velocity is equal to the current velocity will cause the velocity error to be zero, $\dot{e} \approx 0$, Eq(3.7) then becomes

$$f_{motion} = (I - J_v(J_v^T J_v)^{-1}J_v^T) * (K_p e + K_v \dot{e}) \tag{3.11}$$

The joint torque feedback for the RM can then be found using the Jacobian relationship between the RM and the applied hand force using

$$\tau_r = J_r^T F_{motion} = J_r^T (I - J_v (J_v^T J_v)^{-1} J_v^T) * (K_p e + K_v \dot{e}) \tag{3.12}$$

Note that the Jacobian and applied forces must be in the same frame of reference, usually the base frame.

3.2 Positioning the Virtual Manipulator

Recall that the VM is imaginary, in order to find the error the VM must be placed in some position. In general, the end effector position of the VM does not necessarily have to coincide with the end effector position of the RM. In this control approach forces will be generated to move both the VM and the RM to the same position. The RM has higher degrees of freedom than the VM, thus the VM will not be able to reach all positions the RM can. For simplicity consider a VM as a probe constrained to a point in a plane as shown in Figure 3.2. The VM is able to move only in a circular trajectory of radius the link length, while the RM has the ability to move anywhere in the plane.

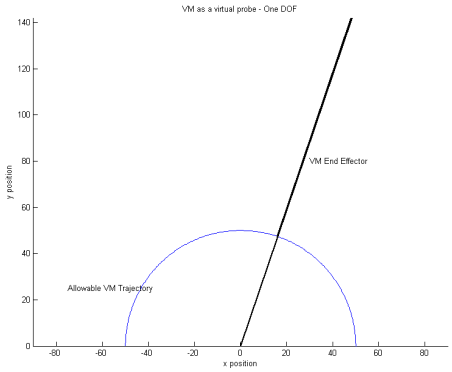
One way to position the VM is to place VM as close as possible to the RM, regardless of the end effector orientation. In Figure 3.3 this is shown as the perpendicular distance between the VM trajectory and the position of the RM in the plane.

Another possibility for positioning the VM is to align the VM end effector with the RM end effector based on orientation as shown in Figure 3.2.

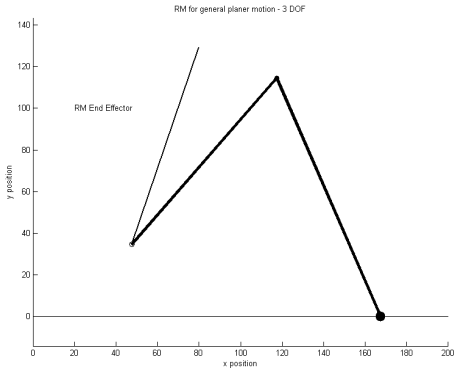
It is possible to place the desired position of the VM anywhere on the VM trajectory, and desired to place the VM somewhere between X(θ_R) and X(θ_V) to minimize orientation and position errors. The error must be examined to determine how to place the VM. The position error is defined as in Eq(3.9). Using the definition of the Jacobian the location of the VM can be found to minimize this error.

$$\delta X_V = J_V \delta \theta_V \tag{3.13}$$

Using the desired position for the VM to be as close as possible to the current position



(a) VM as virtual probe in planer in 1 DOF case



(b) RM for general planer motion (x,y,θ)

Figure 3.1 Defining the VM

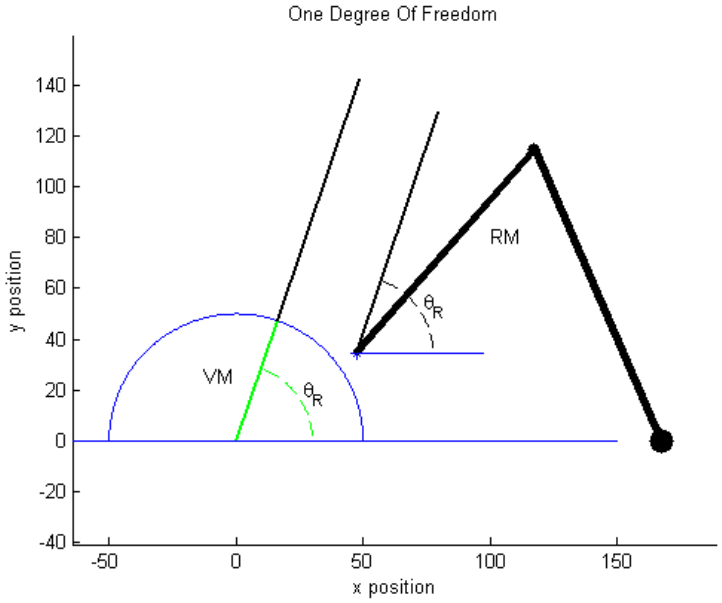


Figure 3.2 Placing the VM based on orientation of RM

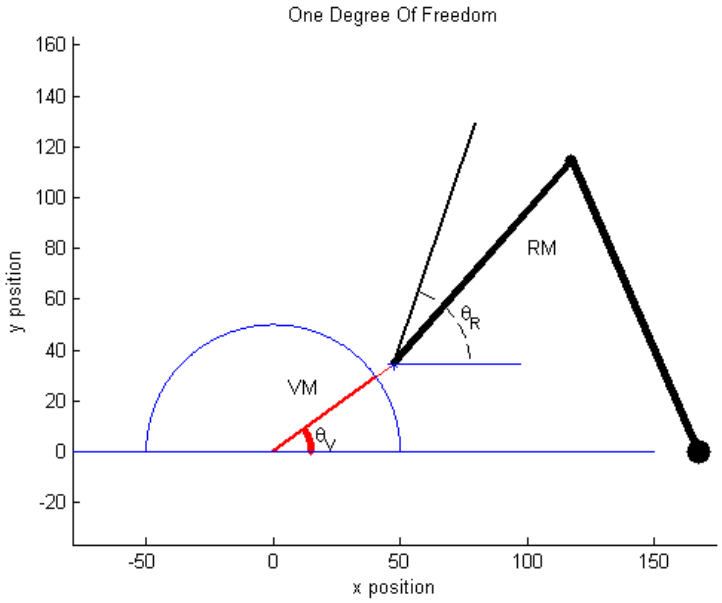


Figure 3.3 Placing the VM based on position of RM

of the RM, $X_{V\,desired}$ is set to be equal to X_R, the current position of the RM. The VM can initially be placed anywhere on the trajectory of the VM. Using the pseudo inverse of the VM Jacobian, $\delta\theta_V$, the amount the angle should change to decrease the error in X_V and X_R is found. Using an arbitrary value for θ_{V0}, an initial Jacobian J_{V0} can be computed and the following relation can be found,

$$\delta\theta_V* = ({J_{V0}}^T J_{V0})^{-1} {J_{V0}}^T (X_R - X_V) \tag{3.14}$$

A weight matrix, W, can be used to vary the amount the RM position or orientation influence the positioning of the VM.

$$\delta\theta_V* = ({J_{V0}}^T W J_{V0})^{-1} {J_{V0}}^T W (X_R - X_V) \tag{3.15}$$

$$\theta_V = \theta_{V0} + \delta\theta_V* \tag{3.16}$$

This process can be repeated until the change in $\delta\theta_V$ is small in an iterative loop.

The weight matrix, W, can be used to describe only un-actuated joints,

$$W = W_o = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{\theta_x} & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{\theta_y} & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{\theta_z} \end{bmatrix} \tag{3.17}$$

For the planer 1 DOF case applying Eq(3.17) the error for orientation can be found to be:

$$\delta\theta_V* = ({J_{V0}}^T W_o J_{V0})^{-1} {J_{V0}}^T W_o (X_R - X_V) \tag{3.18}$$

Which reduces to

$$\delta\theta_V* = \theta_R - \theta_{V0} = 0 \tag{3.19}$$

θ_V is then found to be equal to the orientation angles of the RM

$$\theta_V = \theta_R \tag{3.20}$$

The weight matrix, W, can also be used to describe only actuated joints,

$$W = W_p = \begin{bmatrix} w_x & 0 & 0 & 0 & 0 & 0 \\ 0 & w_y & 0 & 0 & 0 & 0 \\ 0 & 0 & w_z & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.21}$$

For the planer 1 DOF case applying this the error for orientation can be found to be:

$$\delta\theta_V* = \left({J_{V0}}^T W_p J_{V0}\right)^{-1} {J_{V0}}^T W_p (X_R - X_V) \tag{3.22}$$

Multiplying out the equation reduces to,

$$\delta\theta_V* = \frac{e_y \cos(\theta_V)}{L} - \frac{e_x \sin(\theta_V)}{L} = 0 \tag{3.23}$$

θ_V is then found to be the nearest position on the path of the VM based on the current position of the RM

$$\theta_V = \tan^{-1} \frac{e_y}{e_x} \tag{3.24}$$

When describing angles, θ_V will be described as the angles to cause the VM to go to the nearest position on the path of the VM based on the current position of the RM, while θ_R will be described as the angles that cause the VM to align with the current orientation of the RM.

First the VM will be considered as a 1 DOF manipulator, or probe with sufficient friction to be fixed to a point. Later, unilateral constraints can be added to the VM to allow it to move along the surface as if friction has been overcome. The second case will consider the VM as a 3 DOF manipulator and the RM as a 6 DOF manipulator with three un-actuated joints. This is similar to a probe in free space constrained to a point. For each case, the forward kinematics

and Jacobian of the VM are found and implemented in code using C++ programming language. The code for implementation of control in C++ can be seen in Appendix D.

3.3 One Degree of Freedom

The VM is considered as one joint connected to one link. To accomplish this, the 5 DOF manipulator constrained to a plane to mimic a 1 DOF manipulator as shown in Figure 3.4. The VM is constrained to a point with motion of the VM only allowed for movement in a circular path with radius equal to the VM link length. The PHANTOM Omni is considered a 3 DOF RM with one un-actuated joint. The objective of this control approach is to mask the missing RM joint actuator. The VM and RM will be moved regardless of the user applied forces to allow the user to feel a torque like force. To simulate this, the first joint is held constant and only rotation about the z axis in the wrist is considered. To position the virtual manipulator in the 1 DOF case the fk of the VM are found using the homogeneous transformation matrix relating the motion of the VM handle to the base frame

$$ {}^{0}_{2}T = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & L\cos(\theta) \\ \sin(\theta) & \cos(\theta) & 0 & L\sin(\theta) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.25} $$

The VM can move in three planar dimensions, but there are constraints in the motion. The forward kinematics can be written as:

$$ X_V = \Phi_v(\theta_V) = \begin{bmatrix} X_0 + L\cos(\theta_V) \\ Y_0 + L\sin(\theta_V) \\ Z_0 \\ 0 \\ 0 \\ \theta_v \end{bmatrix} \tag{3.26} $$

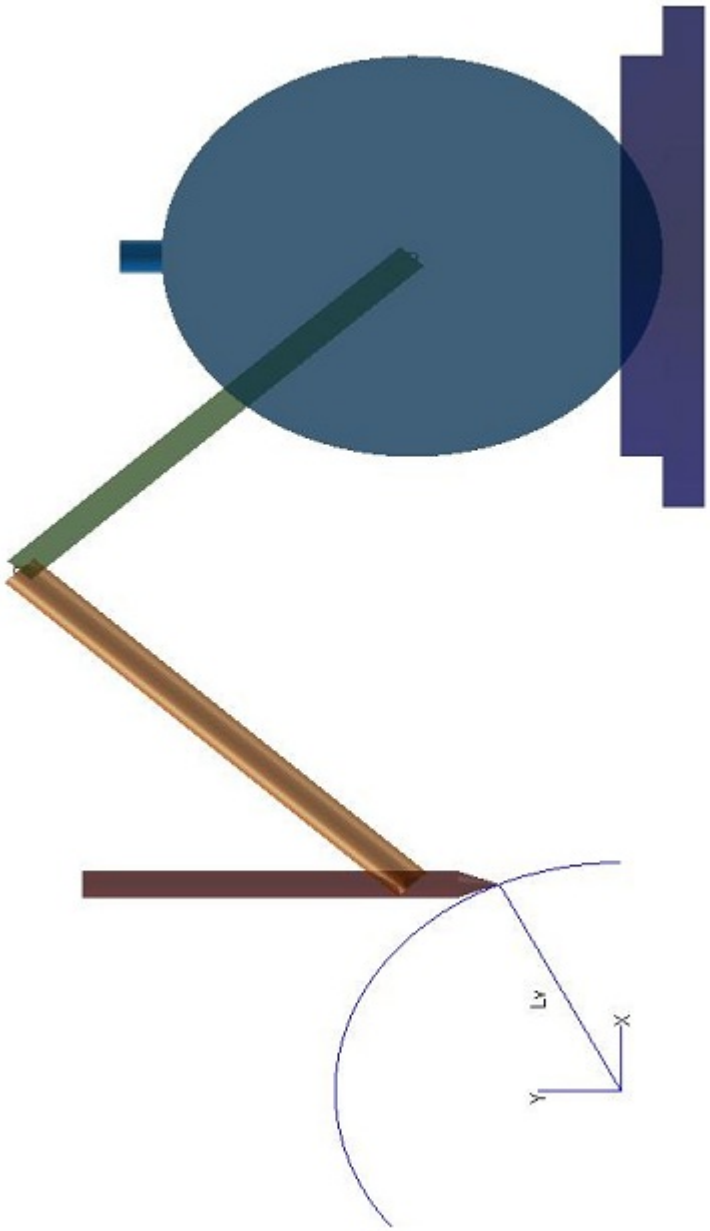


Figure 3.4 One degree of freedom Case

The Jacobian is then found to be

$$J_V = \begin{bmatrix} -L\sin(\theta_V) \\ L\cos(\theta_V) \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.27}$$

Note that in many planar robotics examples, the Jacobian described as a $3x1$ matrix. However, general 6 DOF motion still requires a $6x1$ Jacobian. To keep the results more general for the 1 DOF and 3 DOF cases, the Jacobian is shown with all six rows.

3.3.1 Find Error

The position angle is given by $\theta_V = \tan^{-1}\frac{Y-Y_0}{X-X_0}$, while the orientation angle is given by $\theta_R = \tan^{-1}\frac{{}^0\hat{Z}_{t_{yr}}}{{}^0\hat{Z}_{t_{x_R}}}$. The $\tan^{-1}$ or atan2 function, as previously defined, allows the angle to be in the correct quadrant using both the sine and cosine of the function. Using the fk of VM, desired center of the circle, and current position of RM position error and orientation error are found and shown below.

$$e_x = X_V - X_R$$

$$e_y = Y_V - Y_R$$

$$e_z = Z_V - Z_R$$

$$e_{\theta x} = 0$$

$$e_{\theta y} = 0$$

$$e_{\theta z} = \theta_V - \theta_R$$

Now that the error and control law are known the gains must be set. Orientation error is an angle in radians, to get this in the same order of magnitude it must be multiplied by

the circumference C of the circle, $C = 2\pi r$, where r is the radius of the circle. K_p is then represented by

$$K_p = \begin{bmatrix} k_p & 0 & 0 & 0 & 0 & 0 \\ 0 & k_p & 0 & 0 & 0 & 0 \\ 0 & 0 & k_p & 0 & 0 & 0 \\ 0 & 0 & 0 & k_p 2\pi L & 0 & 0 \\ 0 & 0 & 0 & 0 & k_p 2\pi L & 0 \\ 0 & 0 & 0 & 0 & 0 & k_p 2\pi L \end{bmatrix} \tag{3.28}$$

The control can now be implemented as in Appendix D, and tested on the PHANTOM Omni.

3.4 Three Degree of Freedom

The VM is considered as three joints connected to one link, as shown in Figure 3.5. When constrained to a point this allows for movement of the RM in a spherical path with radius equal to the VM link length. The PHANTOM Omni is considered a 6 DOF RM. The first three joints are actuated while the last three are not. . The fk given for the 3 DOF VM are

$${}^0_4T = \begin{bmatrix} r11 & r12 & r13 & P_x \\ r21 & r22 & r23 & P_y \\ r31 & r32 & r33 & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.29}$$

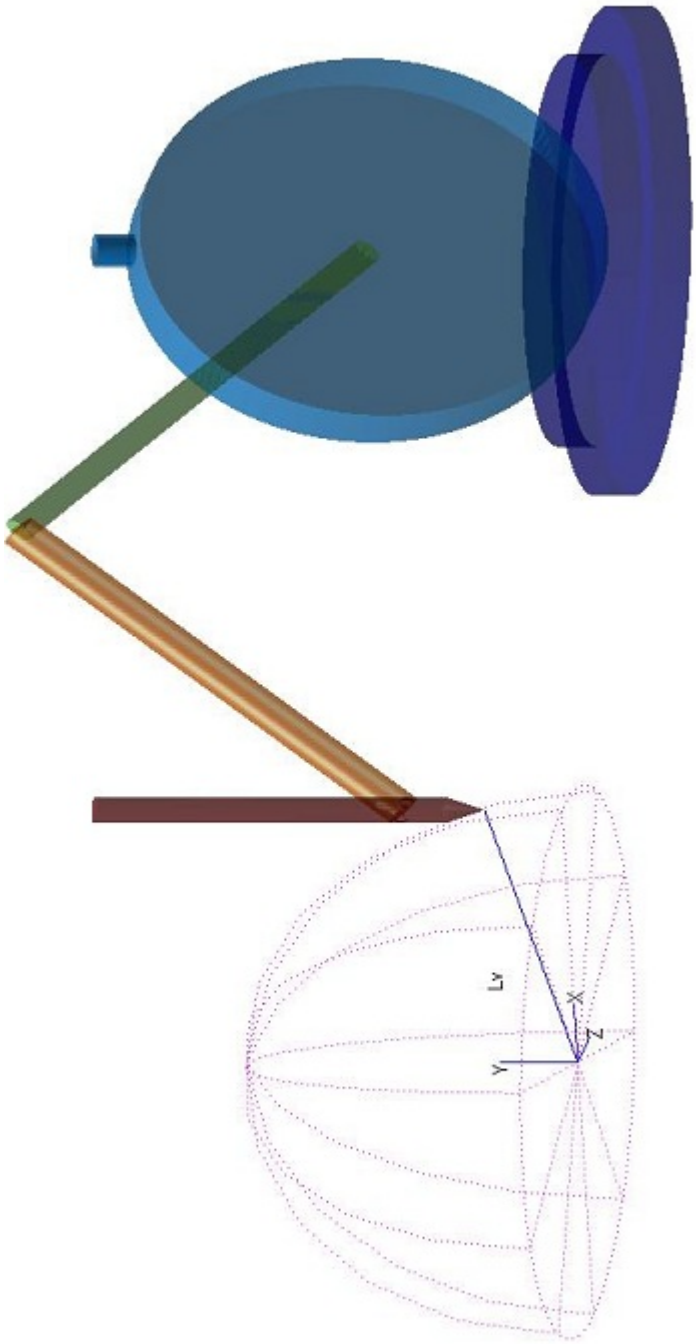


Figure 3.5 Three degree of freedom Case

where,

$$r11 = \sin(\theta_1)\sin(\theta_3) + \cos(\theta_1)\cos(\theta_2)\cos(\theta_3)$$

$$r21 = \sin(\theta_1)\cos(\theta_2)\cos(\theta_3) - \cos(\theta_1)\sin(\theta_3)$$

$$r31 = \sin(\theta_2)\cos(\theta_3)$$

$$r12 = \sin(\theta_1)\cos(\theta_3) - \cos(\theta_1)\cos(\theta_2)\sin(\theta_3)$$

$$r22 = -\cos(\theta_1)\cos(\theta_3) - \sin(\theta_1)\cos(\theta_2)\sin(\theta_3)$$

$$r32 = -\sin(\theta_2)\sin(\theta_3)$$

$$r13 = \cos(\theta_1)\sin(\theta_2)$$

$$r23 = \sin(\theta_1)\sin(\theta_2)$$

$$r33 = -\cos(\theta_2)$$

$$P_x = L\cos(\theta_1)\sin(\theta_2)$$

$$P_y = L\sin(\theta_1)\sin(\theta_2)$$

$$P_z = -L\cos(\theta_2)$$

The position of the end effector can be given as:

$$X(\theta_V) = \Phi_v(\theta_V) = \begin{bmatrix} X_0 + L\cos(\theta_{V1})\sin(\theta_{V2}) \\ Y_0 + L\sin(\theta_{V1})\sin(\theta_{V2}) \\ Z_0 - L\cos(\theta_{V2}) \\ 0 \\ \theta_{V2} \\ \theta_{V1} \end{bmatrix} \tag{3.30}$$

In this case it was found that the third angle does not affect position of the VM, thus it will not be included in the Jacobian of the VM. Consider the example of twisting a pencil verses

rotating a pencil on a table, the pencil will twist freely but when we try to rotate or bend the pencil a force trying to move the pencil will be felt. The Jacobian J_v for the VM is found to be

$$J_v = \begin{bmatrix} -L\sin(\theta_{V1})\sin(\theta_{V2}) & L\cos(\theta_{V1})\cos(\theta_{V2}) & 0 \\ L\cos(\theta_{V1})\sin(\theta_{V2}) & L\sin(\theta_{V1})\cos(\theta_{V2}) & 0 \\ 0 & L\sin\theta_{V2}) & 0 \\ 0 & \sin(\theta_{V1}) & \cos(\theta_{V1})\sin(\theta_{V2}) \\ 0 & -\cos(\theta_{V1}) & \sin(\theta_{V1})\sin(\theta_{V2}) \\ 1 & 0 & -\cos(\theta_{V2}) \end{bmatrix} \tag{3.31}$$

3.4.1 Find Error

The first position angle is given just as in the 1 DOF case θ_{V1} =$\tan^{-1}\frac{Y-Y_0}{X-X_0}$. For the second angle the inverse cosine function will be used. The VM is a specified length, but the distance of the RM from the desired sphere center may not be accurate, so a variable equal to the distance of the RM to the desired center of the sphere is created LL = $\sqrt{(X-X_0)^2+(Y-Y_0)^2+(Z-Z_0)^2}$. Now the second position angle is found to be θ_{V2} = π - $\cos^{-1}\frac{Z-Z_0}{LL}$. The orientation angles are found based on orientation using the forward kinematics of the RM. the first orientation angle is $\theta_{R1} = \tan^{-1}\frac{{}^0\hat{Z}_{t_{y_R}}}{{}^0\hat{Z}_{t_{x_R}}}$, and the second θ_{R2} = π - $\cos^{-1}{}^0\hat{Z}_{t_{z_R}}$. The third orientation angle is the twist about the x axis. If the desired angle is always set to equal the current angle, the angle error will always be zero. Now the fk for the VM is defined, so the error for position and orientation is defined as below.

$$e_x = x_V - x_R \tag{3.32}$$

$$e_y = y_V - y_R \tag{3.33}$$

$$e_z = z_V - z_R \tag{3.34}$$

$$e_{\theta x} = \theta_{V3} - \theta_{R3} = 0 \tag{3.35}$$

$$e_{\theta y} = \theta_{V2} - \theta_{R2} \tag{3.36}$$

$$e_{\theta z} = \theta_{V1} - \theta_{R1} \tag{3.37}$$

The gain Kp is defined as in Eq(3.28). All components of control are found, thus it can be implemented as on PHANTOM Omni. The code for implementing this code is shown in Appendix D.

CHAPTER 4. RESULTS

This chapter includes results for the 1 DOF and 3 DOF cases. Error based on position is considered as finding the nearest position on the path of the VM based on the current position of the RM, or error in the x, y and z directions. This error will be multiplied by K_{pos}. Orientation error, or the difference in orientation of the VM and the RM will be multiplied by K_{ornt}. The values of gains for the position, K_{pos}, and orientation,K_{ornt} can be adjusted as to make the position or orientation of the RM more or less important on generating force to be felt by the user. The following cases were selected to test the theory on the phantom:

Case 1: One DOF, $K_{pos} = 0.5$, $K_{ornt} = 0$

Case 2: One DOF, $K_{pos} = 0.5$, $K_{ornt} = 0.5$*circumference

Case 3: One DOF, $K_{pos} = 0.6$, $K_{ornt} = 0.5$*circumference

Case 4: One DOF, $K_{pos} = 0.5$, $K_{ornt} = 1$*circumference

Case 5: Three DOF, $K_{pos} = 0.5$, $K_{ornt} = 0$

Case 6: Three DOF, $K_{pos} = 0.5$, $K_{ornt} = 0.5$*circumference

4.1 One Degree of Freedom

4.1.1 Case 1

The 1 DOF case considers the VM as one link connected by a revolute joint to the base. When $K_{ornt} = 0$, the error ignores orientation error (error in θ_x, θ_y, and θ_z), and the RM is constrained to the circular path of the VM. For the 1 DOF VM, only rotation about the z axis is considered,thus setting the error of θ_z equal to zero should result in the RM being constrained to anywhere on the circular path of the VM. Figure 4.1 shows the RM located outside the path of the VM,while Figure 4.2 shows the RM located on the path of the VM.

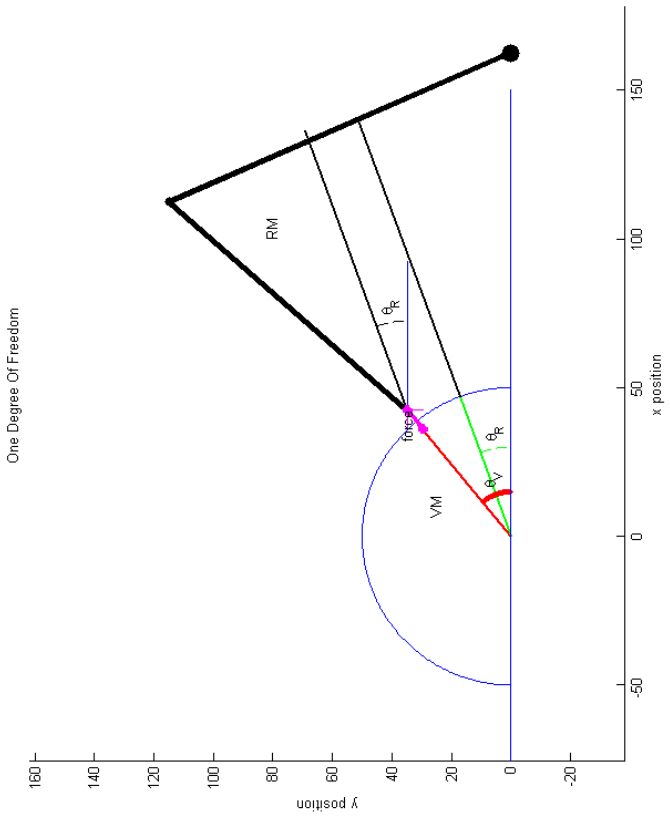


Figure 4.1 Case 1: error outside path of VM

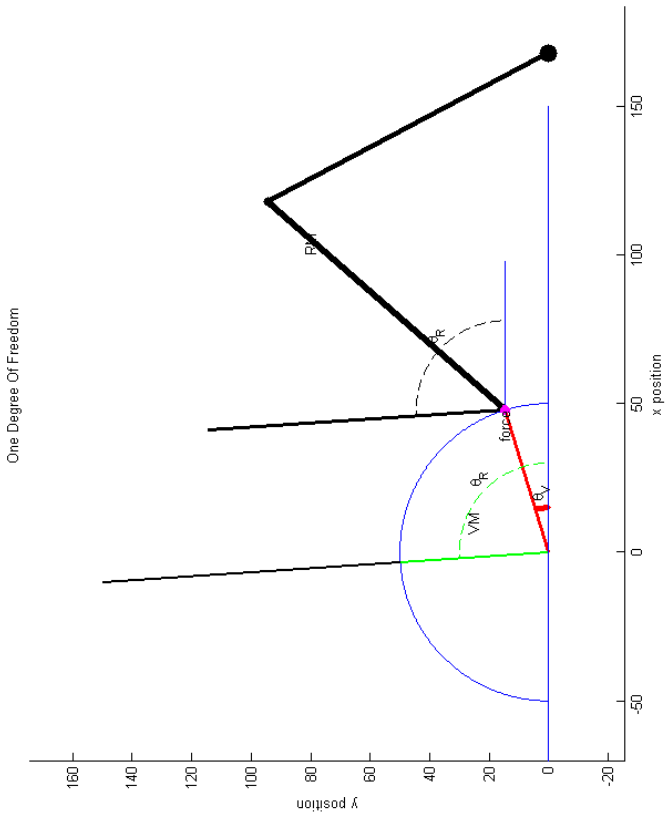


Figure 4.2 Case 1: error on path of VM

The force sent to the RM for the user to feel is shown as a magenta line. The force is shown in two thiner lines that show f_x and f_y, while the thicker magenta line is the two dimensional force vector. The force magnitude has been multiplied by a constant to be easier to see, but contains the same directional information. The red line inside the path of the VM, is the VM based on X(θ_V), the nearest position of the RM. Force in this direction is normal to the surface of the circle. The green line located inside the path of the VM is the VM location based on θ_R, the orientation of the RM. The figure also contains a representation of the RM as a 3 DOF manipulator. The following cases in this chapter will represent data in the same way.

The time history of the position of the RM, VM(θ_R), and VM(θ_V) in the x and y directions is shown in Figure 4.3. This figure also includes a time history of force both in the x and y directions. For the case where orientation error is set to zero $K_{ornt} = 0$, when the RM is located on the path of the VM there is no position error and thus no force. When the RM is not of the path of the VM the force is felt by the is normal to the surface of the circle, it will generate a force to take the user directly to the closest point on the path of the VM. When the force is dependent only on current position or X(θ_V), force will only be created when there is a difference in the position of the RM and X(θ_V). It is clearly shown in Figure 4.3 that when there is no error between the current position of the RM and X(θ_V), the force created is zero, when the current position of RM is greater than X(θ_V) a negative force is created pulling the RM back to the VM. The force generated is independent of X(θ_R).

4.1.2 Case 2

In this case, force is computed based on both orientation and position error. The gain matrix can be adjusted to make error of position or orientation of the RM more important. This case considers orientation error and position error weighted equally. The RM located outside the path of the VM can be seen in Figure 4.4, while the RM on the path of the VM is shown in Figure 4.5. The data is shown in the same manor as in Figure 4.1.

A time history is also included for this set of data shown in Figure 4.6. When the RM is not on the path of the VM, the force acting on the RM causes the RM to go to a location

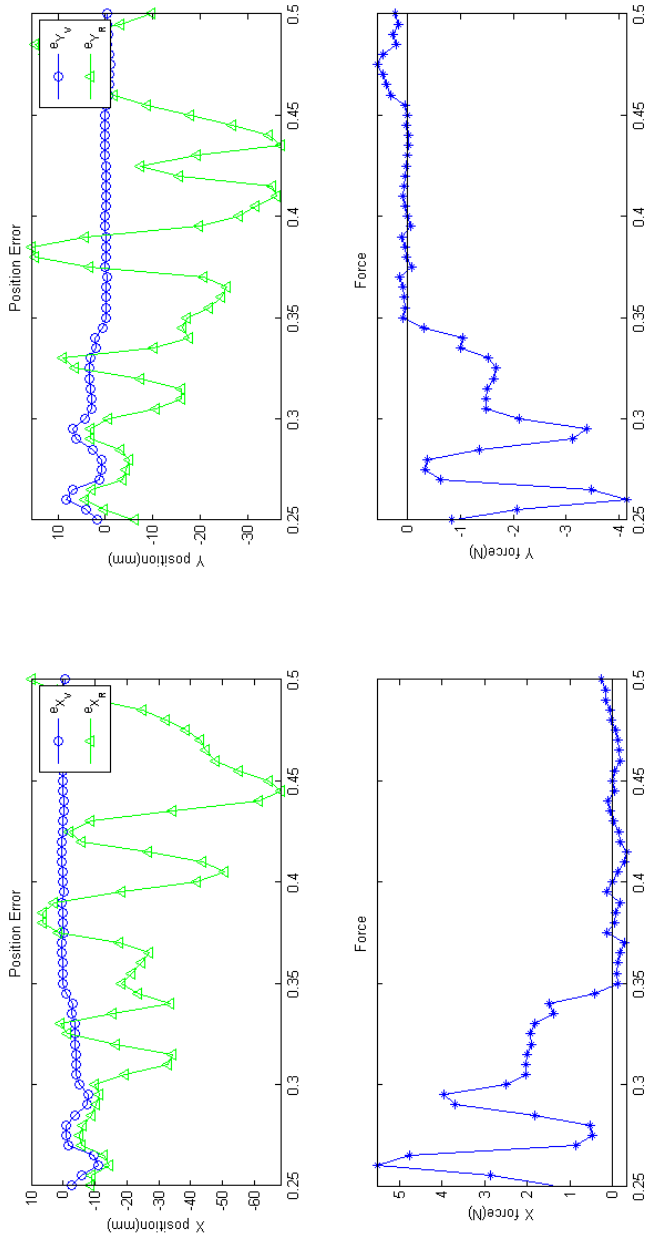


Figure 4.3 Case 1: time history of error

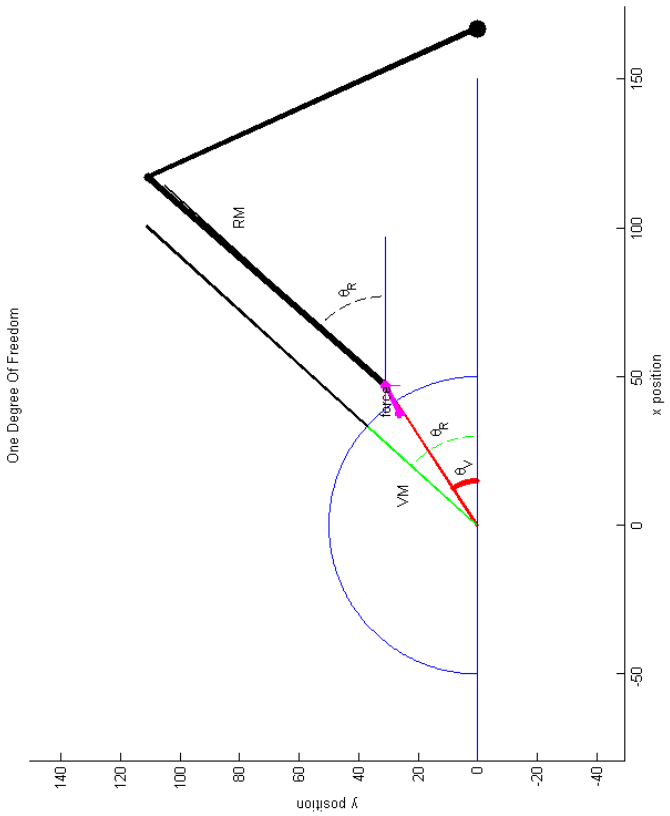


Figure 4.4 Case 2: error outside path of VM

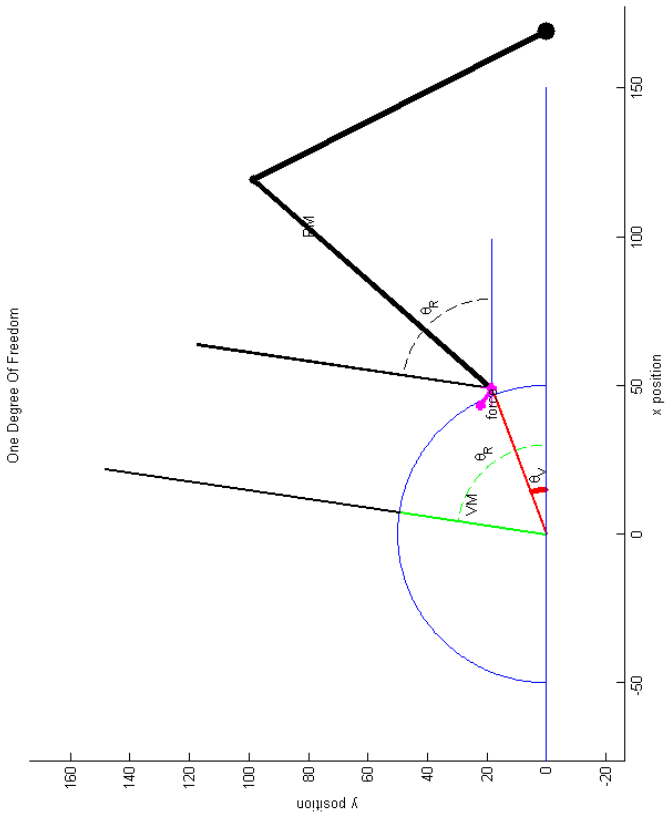


Figure 4.5 Case 2: error on path of VM

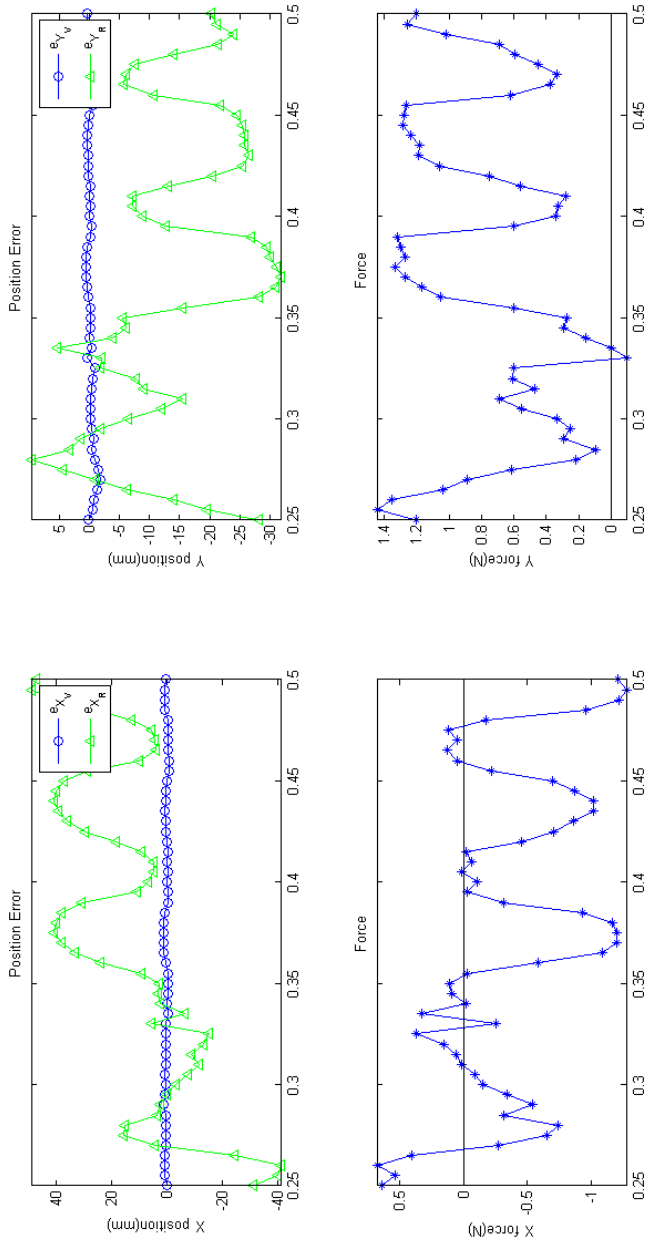


Figure 4.6 Case 2: time history of error

on the VM path between X(θ_V) and X(θ_R). It appears the force acting on the RM will force the user closer to the position on the path generated by X(θ_V), or the current position of the VM. When the RM is on the path of the VM, a force is generated to the user to cause the user to go towards the position based on the orientation of the RM. This force generated gives the user a feeling similar to a bending force. The time history data in Figure 4.6 shows when there is small error based on X(θ_V) the force generated is strongly based on the error in orientation, or X(θ_R).

4.2 One Degree of Freedom Adjust Kp

4.2.1 Case 3

By adjusting the error gains the amount of force generated can cause the RM to want to go to the nearest position on the path of the VM, or have the VM align with the current orientation of the RM. Case 3 has a greater the gain based on position error increased while the orientation error remains the same as in Case 2. The RM outside the path of the VM is shown in Figure 4.7.The RM on the path of the VM is shown in Figure 4.8. The representation of lines on the diagram are similar to Figure 4.1.

A time history is also included for this set of data shown in Figure 4.9. In this case, position error is weighted more heavily that orientation error. In Figure 4.7 it appears that the force acting on the RM is very similar to the case were equal position and orientation weights were used when calculating Kp. When looking at Figure 4.9 it can be seen that there is some force created due to orientation error, but less than that based on position error as expected.

4.2.2 Case 4

This case generates Kp to more heavily weight orientation error than position error. This put a greater importance on aligning the VM with the orientation of the RM. The RM outside the path of the VM is shown in Figure 4.10, while the RM on the path of the VM is shown in Figure 4.11.

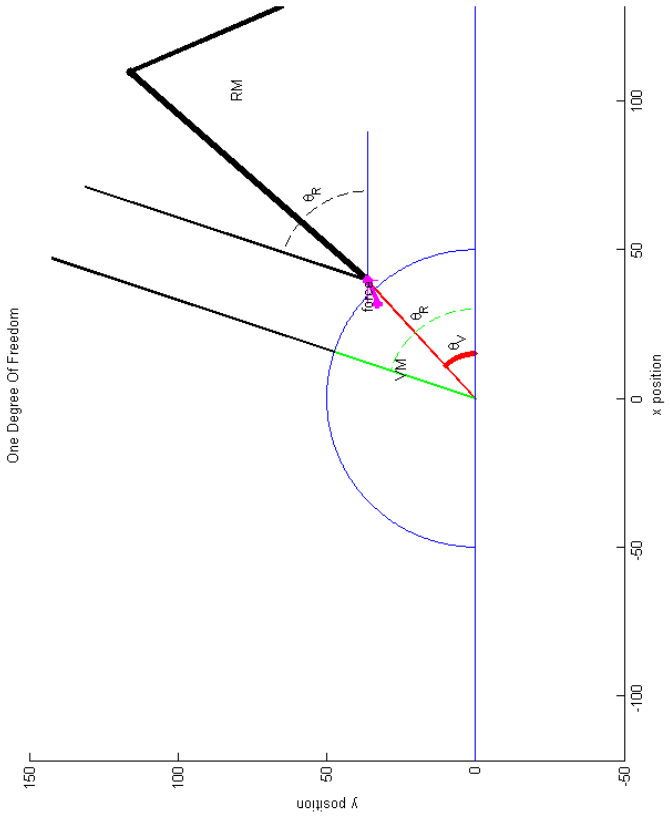


Figure 4.7 Case 3: error outside path of VM

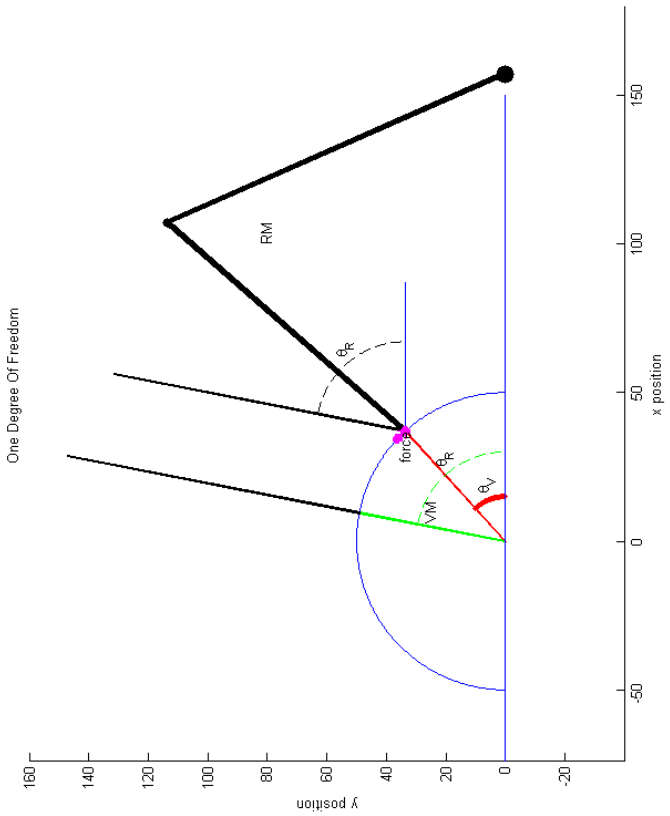


Figure 4.8 Case 3: error on path of VM

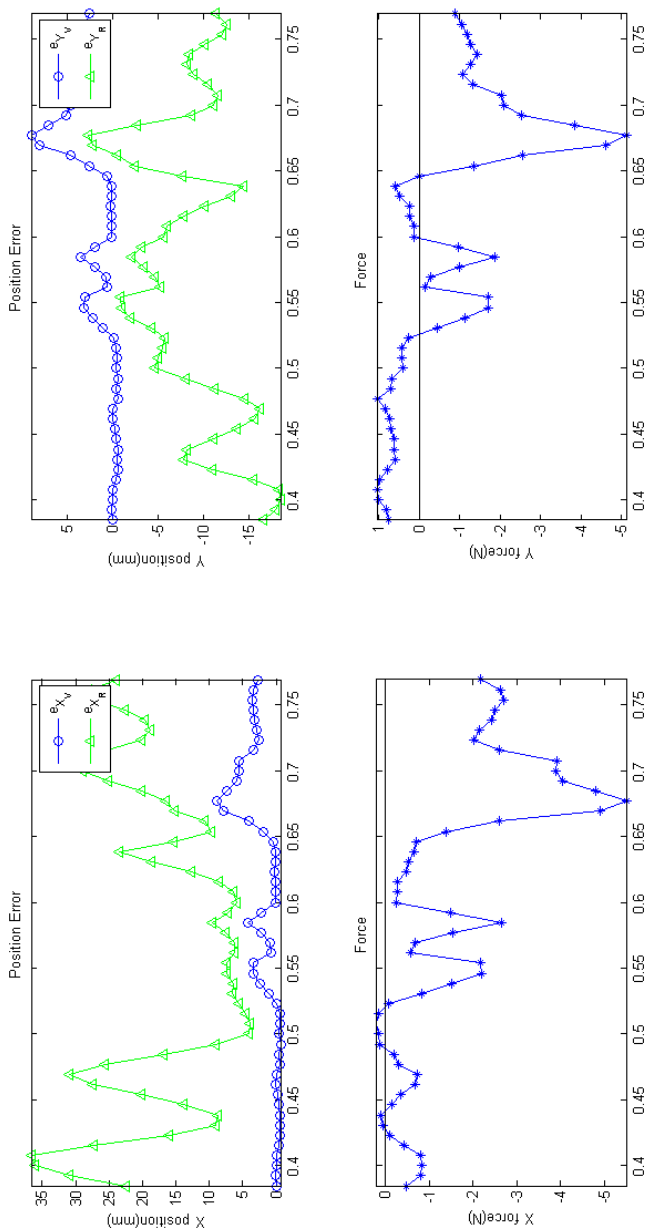


Figure 4.9 Case 3: time history or error

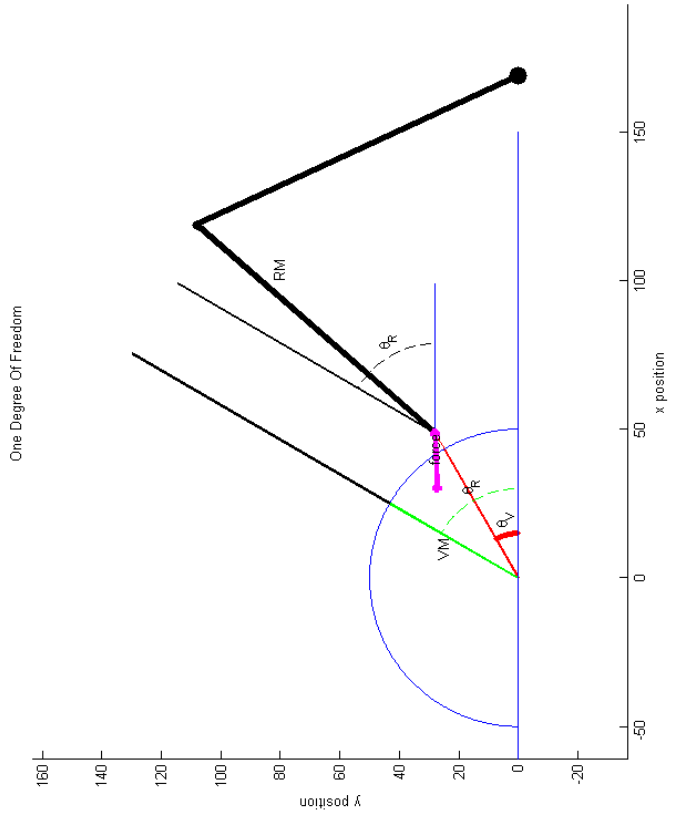


Figure 4.10 Case 4: error outside path of VM

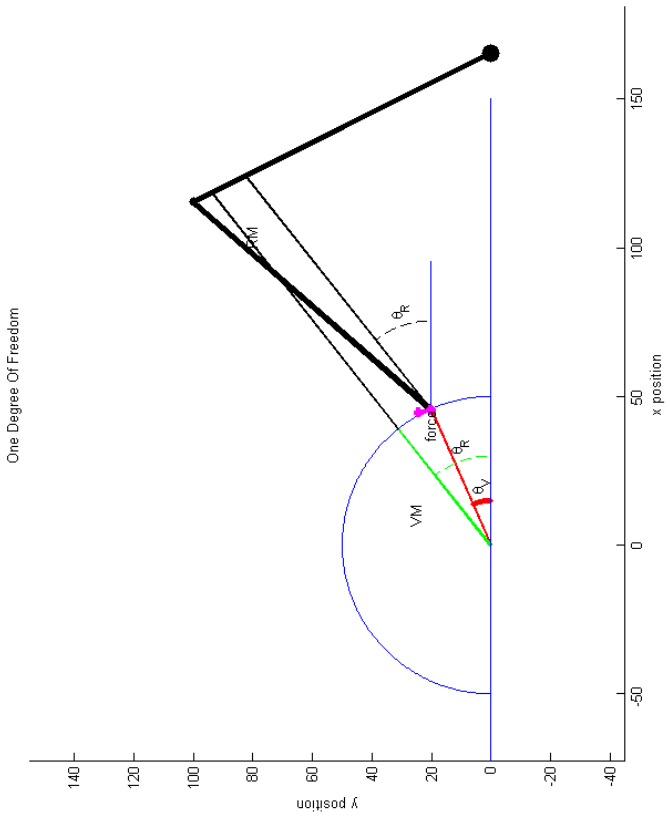


Figure 4.11 Case 4: error on path of VM

The representation of lines on the diagram are similar to Figure 4.1, it is seen in Figure 4.10 that the force is pushing the RM towards the path of the VM between X(θ_V) and X(θ_R). The force acting in the negative x direction is much greater than the force acting in the negative y direction; while pulling the RM closer to the path of the VM the orientation is being heavily weighted. The x axis of the time history graph in Figure 4.12 shows that as the RM gets farther away from the path of the VM the force is much greater than the force felt when the RM is on the path of the VM.

4.3 Three Degrees of Freedom

4.3.1 Case 5

The 3 DOF case considers the probe as one link connected to the base by three revolute joints. As in the 1 DOF Case 1, the error in orientation can be considered zero and the RM is constrained to the path of the VM. For the 3 DOF the RM will be constrained to a spherical path of the VM. Figure 4.13 shows the RM located outside the path of the VM. Figure 4.14 and Figure 4.15 show multiple views of the RM located outside the path of the VM. The VM located on the path of the VM is shown in Figure 4.16 with Figure 4.17 and Figure 4.18 showing multiple views.

The force is shown in three thiner lines that show f_x, f_y, and f_z, while the thicker magenta line is the three dimensional force vector. The force magnitude has been multiplied by a constant to be easier to see, but contains the same directional information. The red line inside the path of the VM, is the VM based on X(θ_V), the nearest position of the RM. The black line located inside the path of the VM is the VM location based on θ_R, the orientation of the RM. The black line located outside the path of the VM is the handle of the RM, a 6 DOF manipulator.The time history of position error of the RM in relations to X(θ_R) and X(θ_V) in the x, y, and z directions are shown in Figure 4.19. The error based on orientation is much greater than that based on position, Figure 4.20 shows the same time history with the error plots zoomed in on position error based on the closest position.

The force generated by the position error forces the RM to the VM based on X(θ_V), or

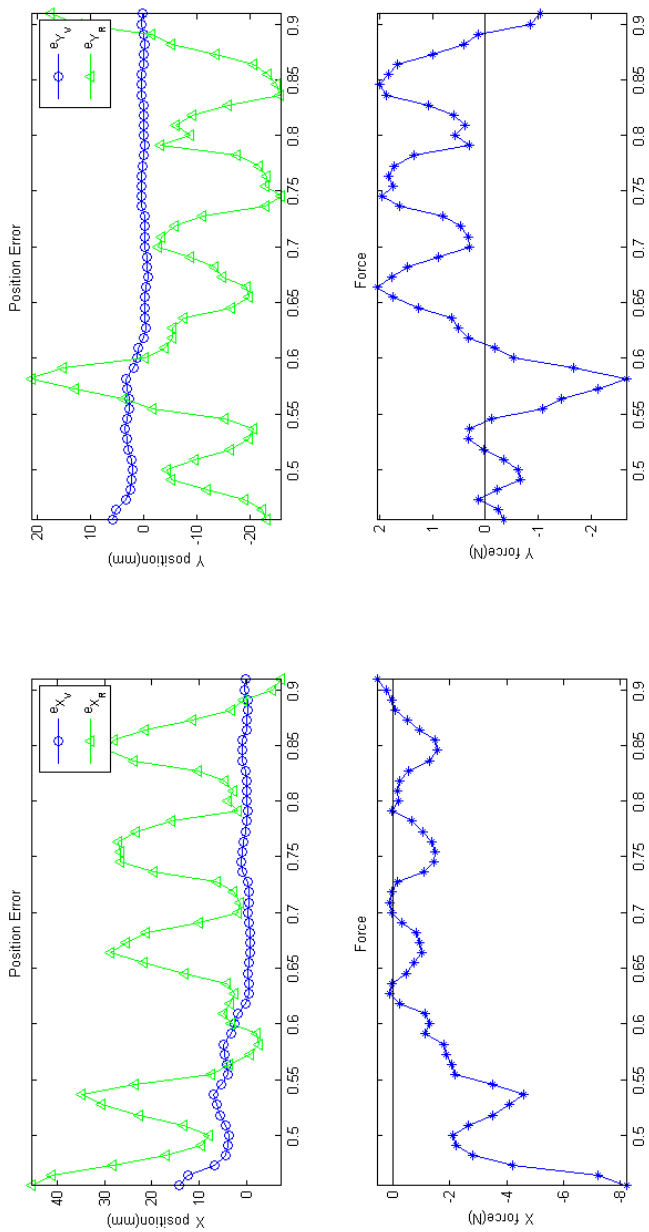


Figure 4.12 Case 4: time history of error

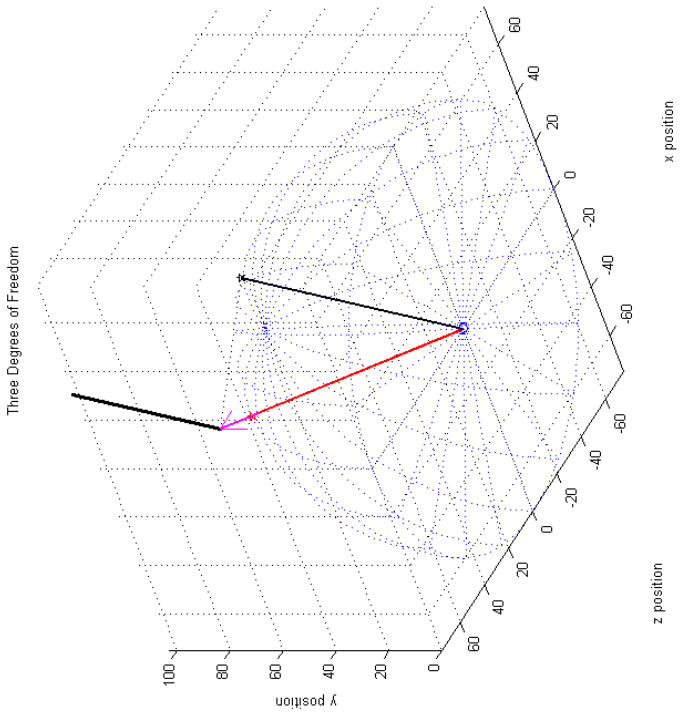


Figure 4.13 Case 5: error outside path of VM

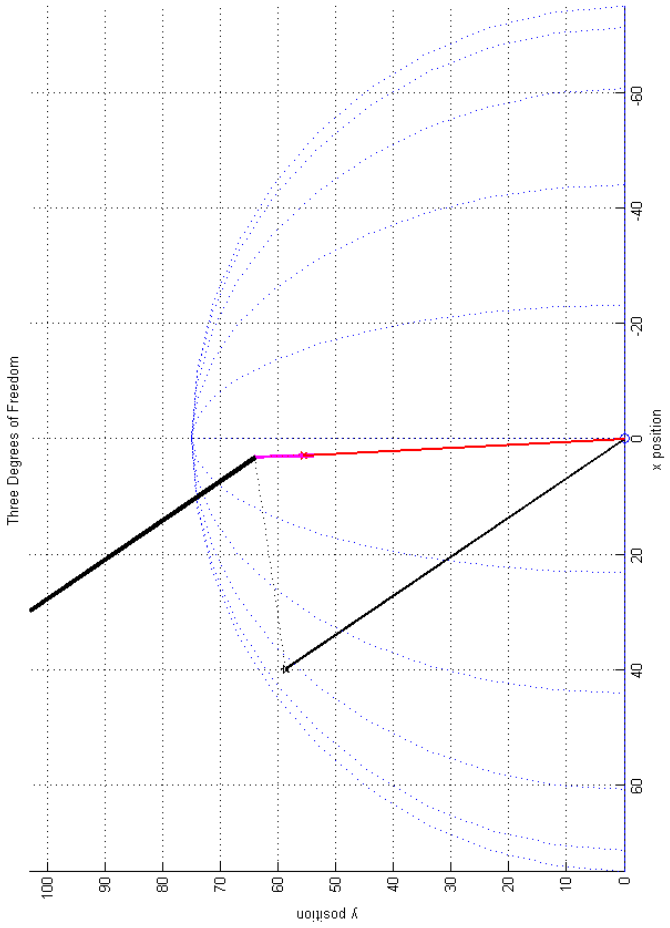


Figure 4.14 Case 5: error outside path of VM yx view

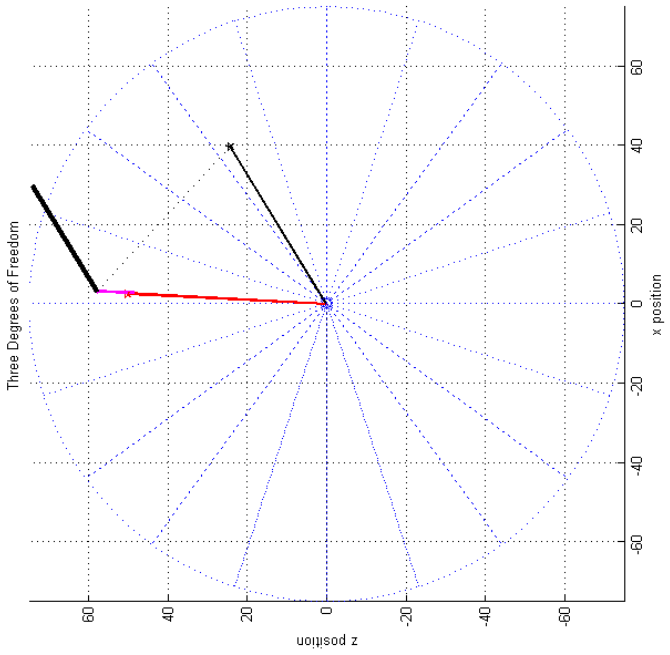


Figure 4.15 Case 5: error outside path of VM zx view

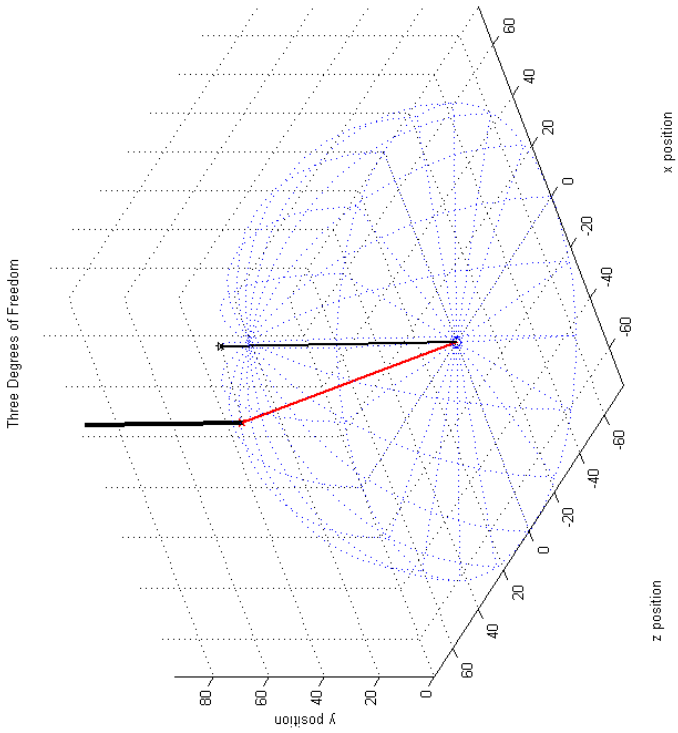


Figure 4.16 Case 5:error on path of VM

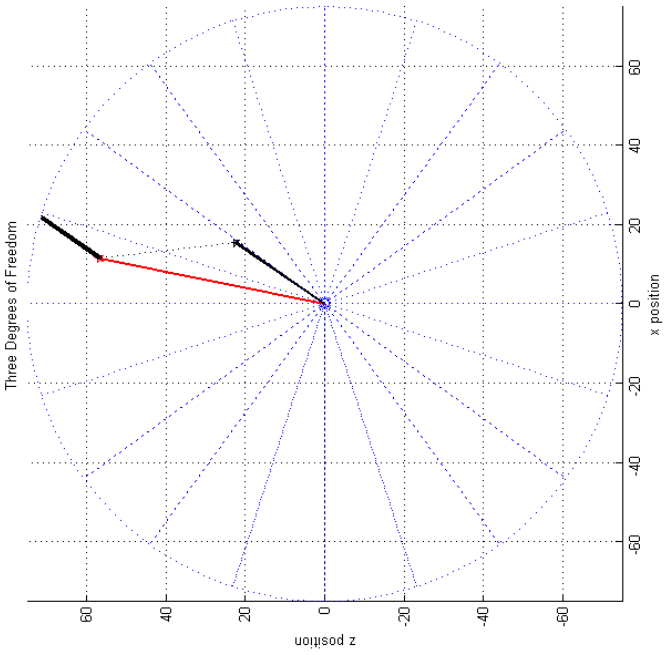


Figure 4.17 Case 5: error on path of VM zx view

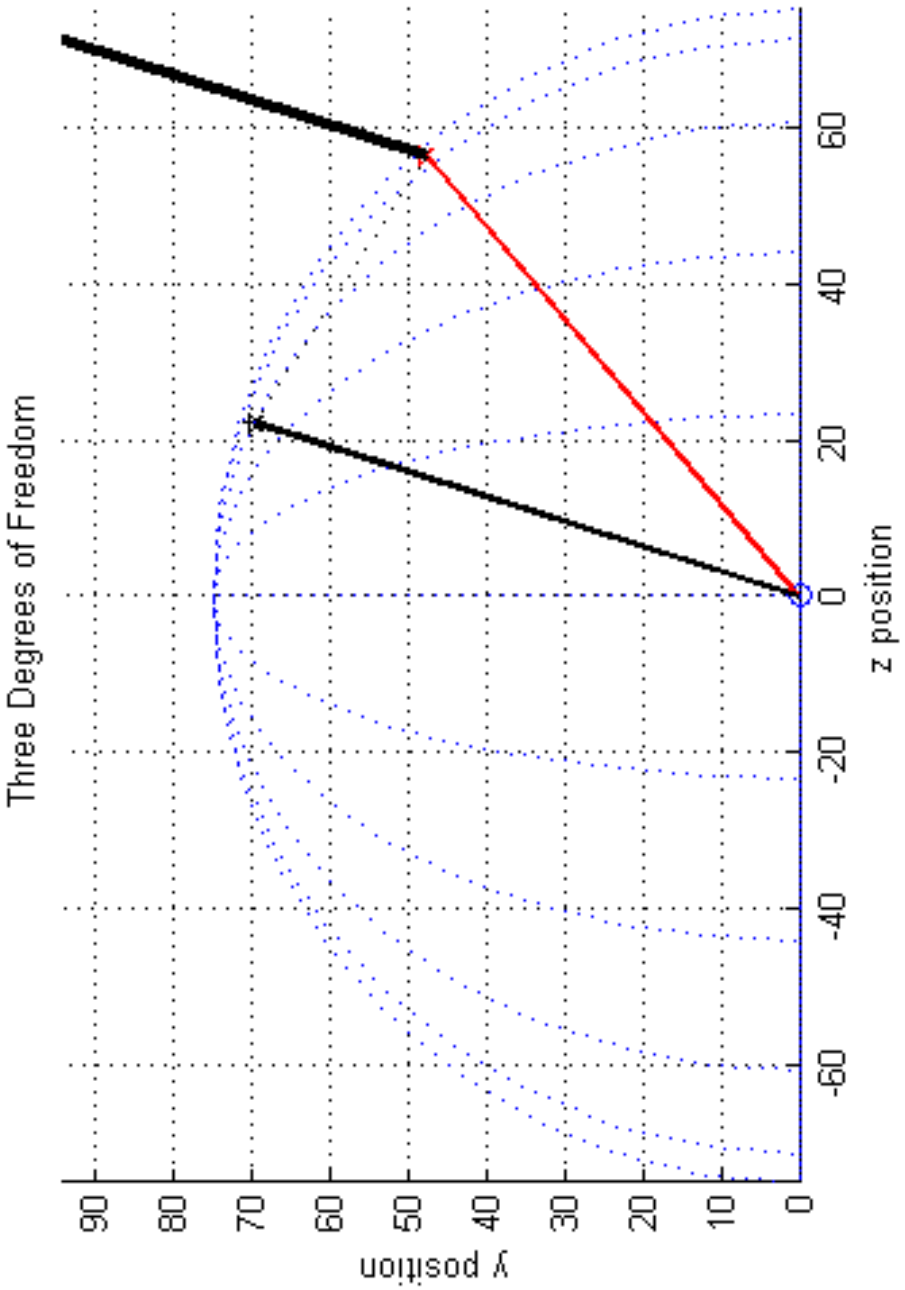


Figure 4.18 Case 5: error on path of VM zy view

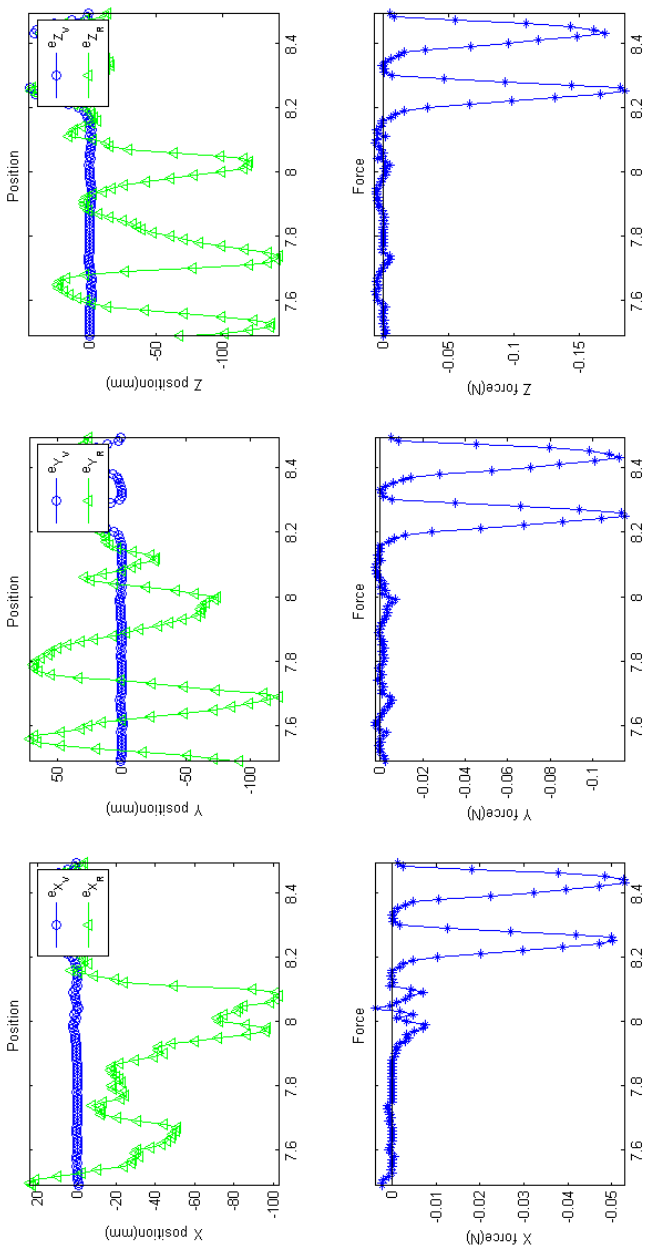


Figure 4.19 Case 5: time history or error

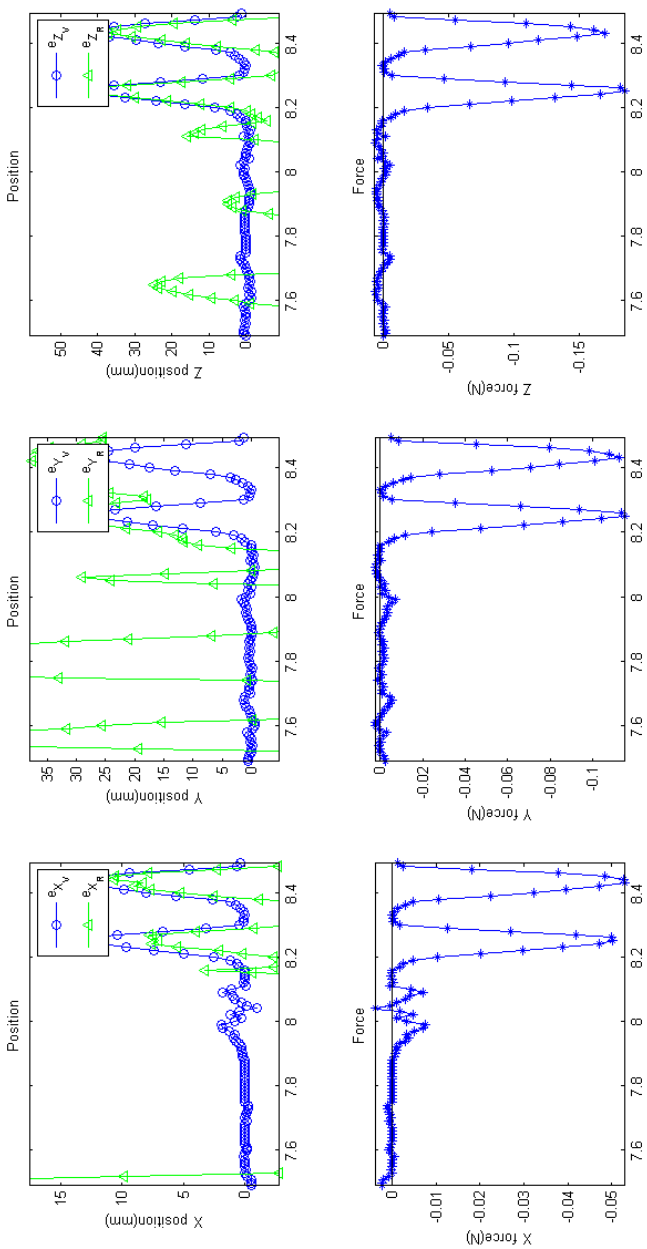


Figure 4.20 Case 5: time history of position error

the closest position. The position error of X(θ_R) is shown, but this does not affect the force generated by the control law. The error and force for each axis shown as a time history shows the error between the currant position of the RM and X(θ_V) remains similar throughout the first half of the run while the orientation angle placing X(θ_R) is varied. When the error in X(θ_V) is small almost no force is generated, when the RM is moved away from the path of the VM the force generated is in the direction that causes the RM to move towards the position of X(θ_V) on the path the of VM. When the RM is on the path of the VM no force is generated. The response in this case is similar to the response in Case 1.

4.3.2 Case 6

For the last case in three dimensions, position and orientation error will be considered. The position error and orientation error are weighted equally. The forces generated when the RM is outside the pathe of the VM are shown in Figure 4.21,Figure 4.22, and Figure 4.23.

The representation of the lines is similar to Figure 4.13. All three figures show the force acting on the RM causing the RM to move towards the sphere in a direction between X(θ_V) and X(θ_R). When the RM is farther away from the spherical path of the VM the force generated to pull the RM directly onto the path are much greater than those to pull the RM to the position that would match the orientation. The response in this case is found to be similar to the response in Case 2. The forces generated when the RM is located on the path of the VM are shown in Figure 4.24,Figure 4.25,and Figure 4.26.

When the RM is on the path of the VM, the forces generated will cause the RM to move towards the location of the VM based on orientation. A time history of force and position error is shown in figure Figure 4.27. As in Case 5 the error pased on posistion is much smaller than that based on orientation, so a zoomed in portion of error based on closest position to spherical path is shown in Figure 4.28.

The time history shows that when the RM is on the path of the VM, the force generated is dependent on the orientation error, but as the error between the location of the RM and the path of the VM gets larger the force generated is dependent on the error between the RM and

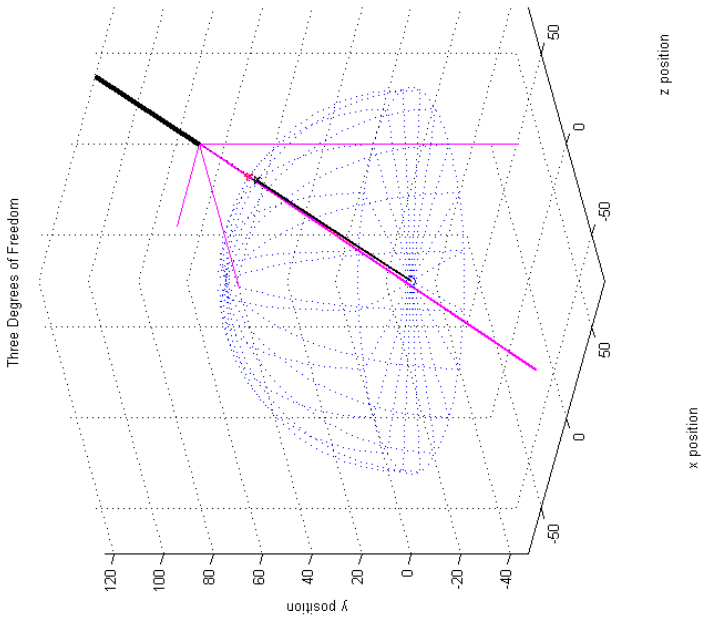


Figure 4.21 Case 6: error outside path of VM

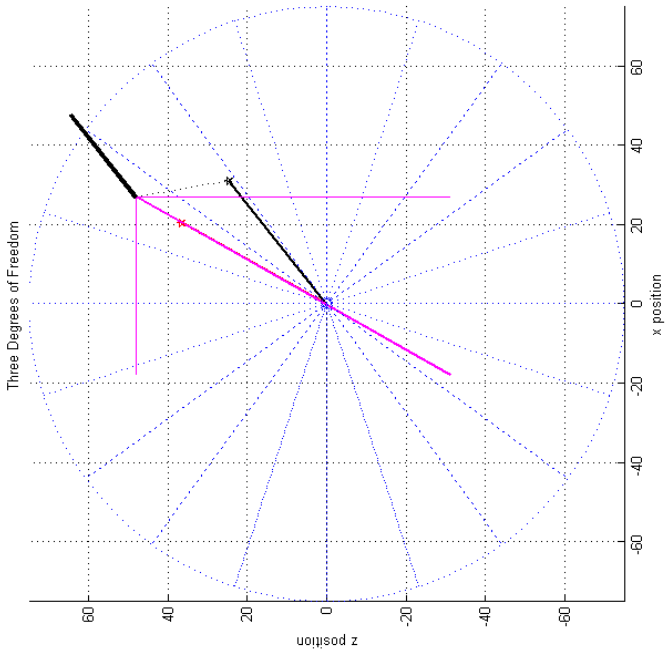


Figure 4.22 Case 6: error outside path of VM xz view

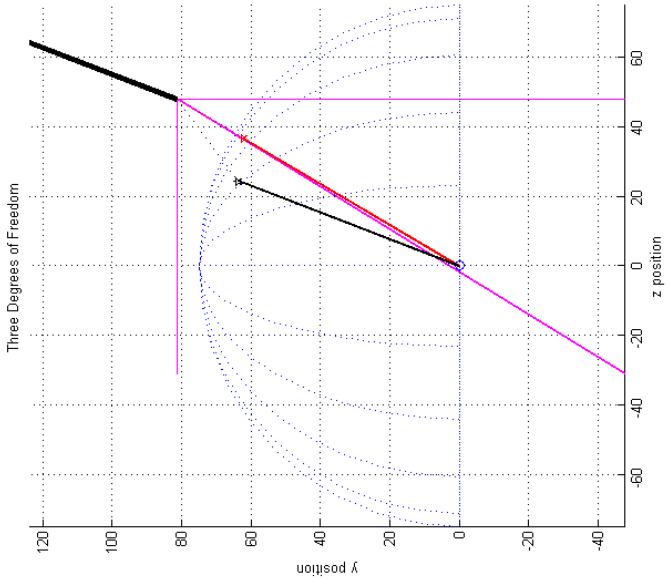


Figure 4.23 Case 6: error outside path of VM zy view

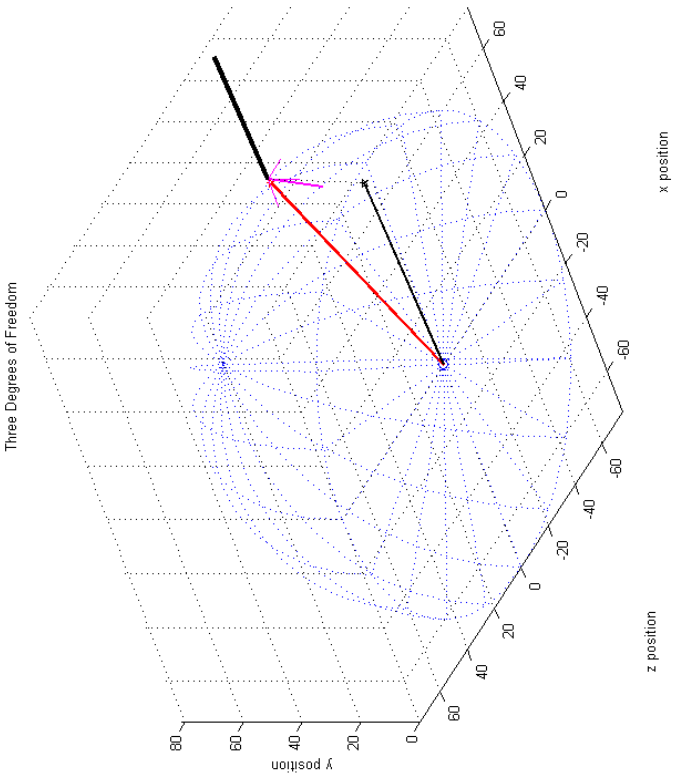


Figure 4.24 Case 6: error on path of VM

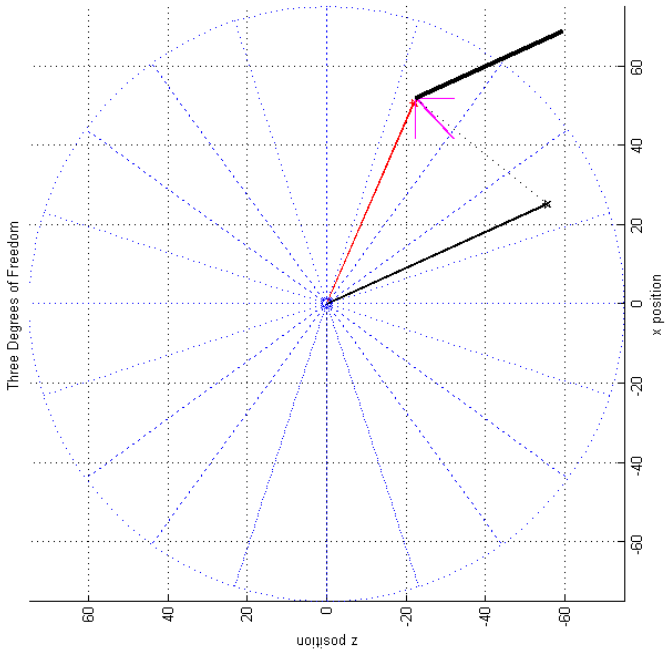


Figure 4.25 Case 6: error on path of VM xz view

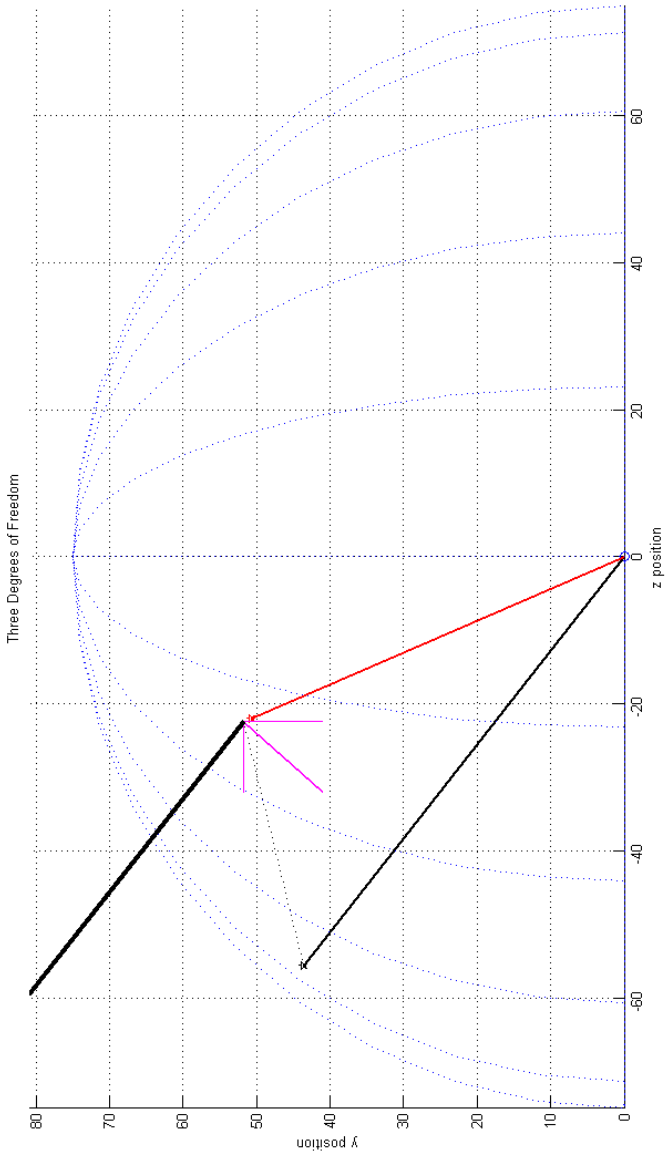


Figure 4.26 Case 6: error on path of VM zy view

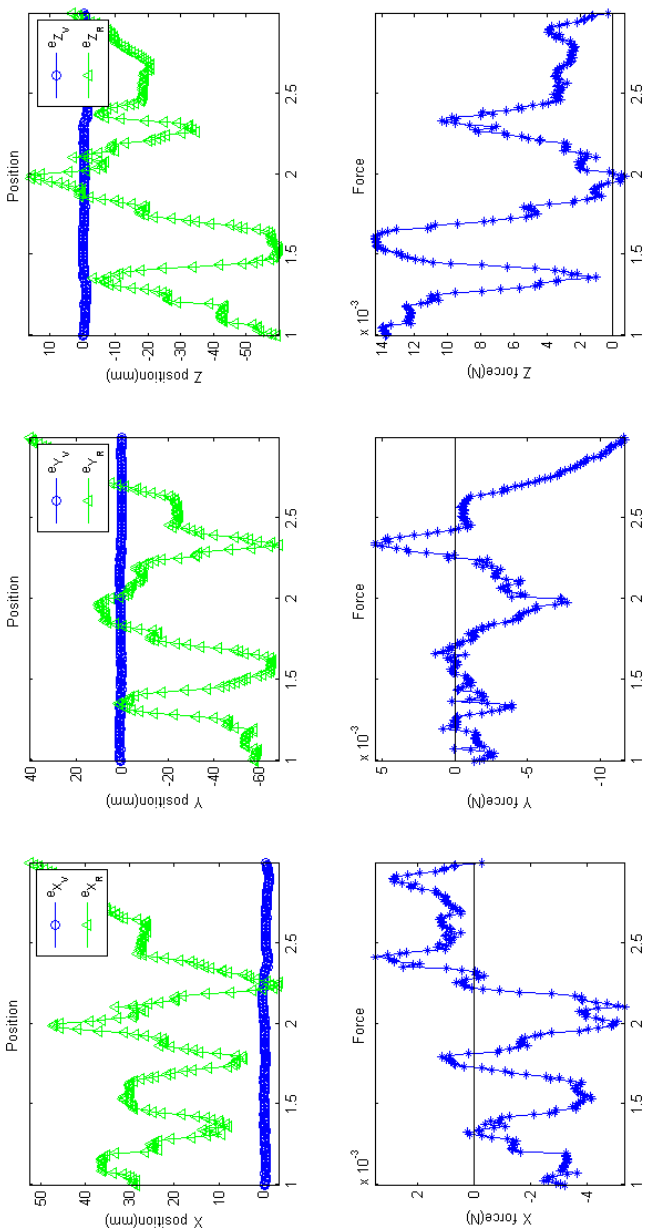


Figure 4.27 Case 6: time history of error

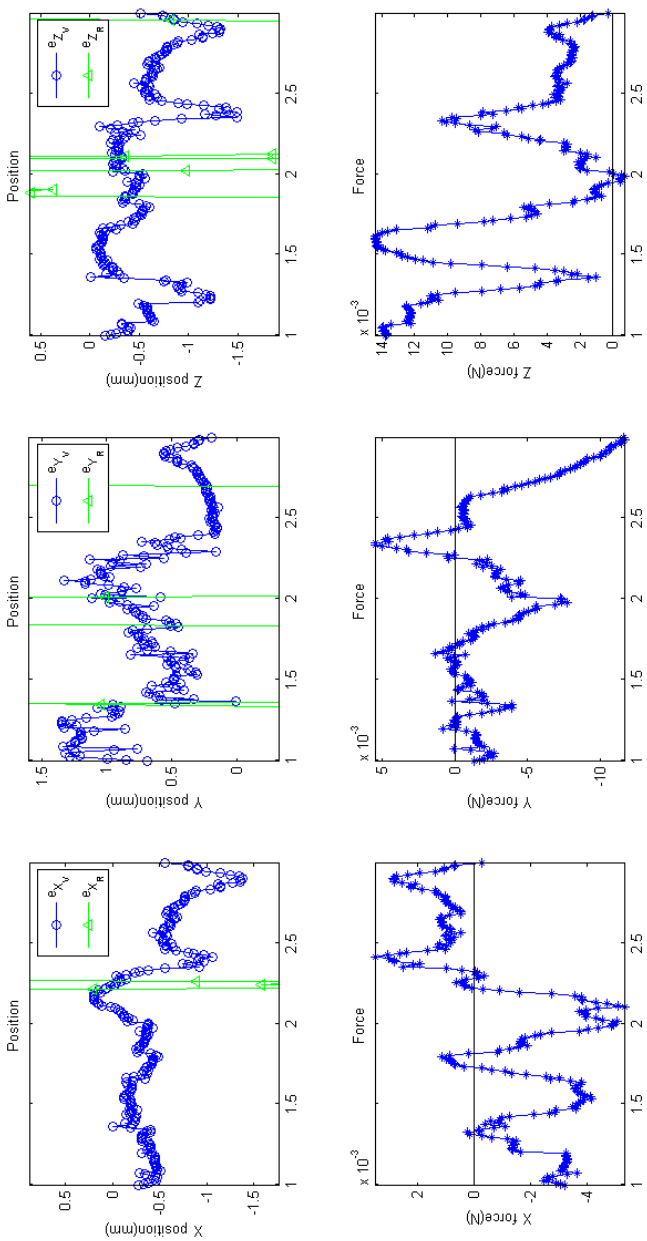


Figure 4.28 Case 6: time history of position error

closest position on the sphere.

Due to the complications with viewing the three dimensional cases and the similarities between cases 1 and 5 and the similarities between cases 2 and 6 it is assumed that the method as applied in 1 DOF will work similarly in higher DOF, thus more cases in 3 DOF and higher DOF are not part of this project.

CHAPTER 5. DISCUSSION AND CONCLUSION

This report discussed developing haptic constraints with a virtual manipulator (VM) for use with under actuated robots, the control was then applied to physical robot manipulator (RM), specifically the PHANTOM Omni. The goal of the control was to mask the missing joint torques by generating feedback forces to the actuated joints. The control theory included using weighted matrices to place the VM and the pseudo inverse Jacobian control law to place the VM to minimize the error between the current RM position and orientation somewhere on the path of the VM. A pseudo inverse Jacobian control law was then used again to generate a force to send to user if the RM. The theory was then tested with six cases which varied in the importance of the position error and orientation error when generating the force feedback.

All the cases show the control theory is able to apply force based on both position and orientation error as desired. The VM can be placed to be in the nearest position on the path based on the current position of the RM. When only position error is considered, the RM is constrained to anywhere on the path of the VM, due to there being no error when the closest current position is equal to the current position. The amount of torque like force felt by the user is able to be varied based on the gain applied to the orientation and position error. The toque force can be increased by increasing the gain of orientation error. In Case 4, the higher orientation gain caused the user to notice the virtual manipulator to be much more responsive to changes in orientation. When the RM was outside the path of the VM, all cases generated a force that pulled the RM onto the path of the VM. The control proved to place the VM somewhere on or between the location produced by using the and closest position angle (θ_V) and current orientation angle (θ_R). When the RM was located on the path of the VM and orientation error was considered the force generated caused the RM to move towards the

location based on orientation angle along the path of the VM.

When finding the error, the weighted matrix to find closest position to the RM was used. To continue with this work experiments using different weighted matrix to find the error could be used. Also this theory could be tested using a VM with higher degrees of freedom. These experiments used a RM that is a haptic, although the theory does not require the use of a haptic device, similar experiments could be performed on a different RM.

APPENDIX A. SPACIAL DESCRIPTION

To define a robotic manipulator, first the space in which that robotic manipulator is located needs to be defined. Spacial descriptions are also a way to reference the physical space the user may be using in a virtual environment. In addition it may be necessary to define more than one space, as a tool frame may be different from the control frame. In order to define a location in the frame, two sets of information must be given: the position and orientation in reference to the origin.

Position and Orientation

In a defined coordinate system the location of a point in the three dimensional space of the coordinate system can be defined by a 3 X 1 position vector. The position vector gives the distance in the x, y, and z direction from the origin.

$$ {}^{A}P = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} \tag{A.1} $$

A general position vector, Eq(A.1), can be seen in Figure A.1. A position vector can be mapped into different frames and retain positional information. One type of mapping is translation mapping. Two translated frames have the same orientation, but different origins. There exists a translational position vector describing the origin of one frame in reference to the other. Translated frames are shown in Figure A.2, with the dashed line containing data of the position vector ${}^{A}P_{BORG}$ that described the translation. A position in frame {B} can be described in frame {A} by ${}^{A}P = {}^{B}P + {}^{A}P_{BORG}$, While location in a frame is given by a position vector, orientation given by rotation matrix, Eq(A.2), which is made up of three unit

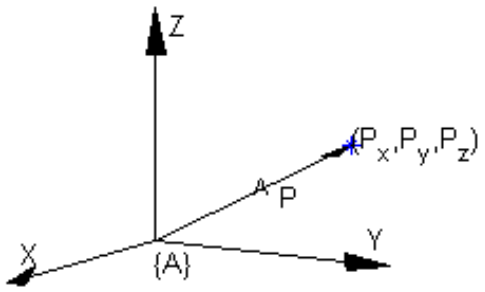


Figure A.1 Position of a point in Cartesian space

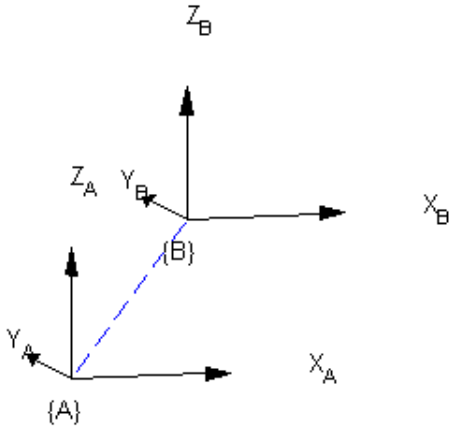


Figure A.2 Translated frames

vectors $\hat{X}$, $\hat{Y}$, and $\hat{Z}$. Each unit vector describes the rotation about that axis. For example ${}^{A}\hat{X}_{B}$ would map $\hat{X}$ in frame {B} into frame {A}. A position vector can be mapped from frame {A} to frame {B} by ${}^{A}P = {}^{A}_{B}R{}^{B}P$. A property of rotation matrices is they are orthogonal. This means the transpose of the matrix is equal to the inverse, or ${}^{A}_{B}R^{-1} = {}^{A}_{B}R^{T} = {}^{B}_{A}R$.

$$ {}^{A}_{B}R = \begin{bmatrix} {}^{A}\hat{X}_{B} & {}^{A}\hat{Y}_{B} & {}^{A}\hat{Z}_{B} \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{A.2} $$

Rotations can happen about the X, Y, or Z axis. The three rotation matrices are shown below.

$$ R_{X}(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{A.3} $$

$$ R_{Y}(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{A.4} $$

$$ R_{Z}(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.5} $$

A simple example of rotating a frame about the Z axis by θ is shown in Figure A.3, the Z axis remains the same, but the X and Y axis in the new frame are rotated by θ

A mapping from one frame to another that includes both translational and rotational mapping can be described with a transformation matrix. A transformation matrix is not orthogonal, but is invertible.

$$ {}^{A}_{B}T = \begin{bmatrix} & {}^{A}_{B}R & & {}^{A}P_{BORG} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.6} $$

A simple example of mapping one frame to another is shown in Figure A.4. The dashed line is the mapping of ${}^{A}P_{B}$, while the rotation mapping is ${}^{A}_{B}R$

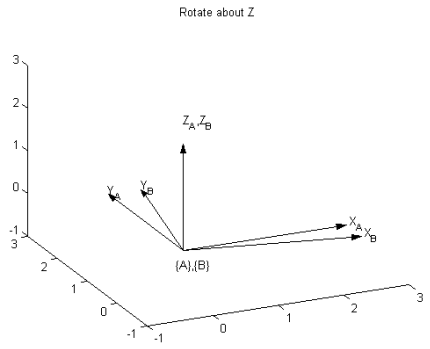


Figure A.3 Frames rotated about the Z axis

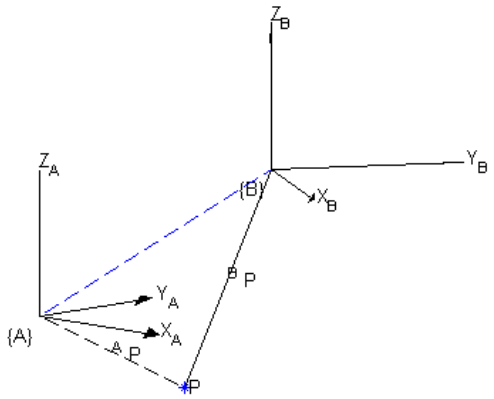


Figure A.4 Position vector mapped from one frame to another

Compound transformations, or transformations through multiple frames are given by

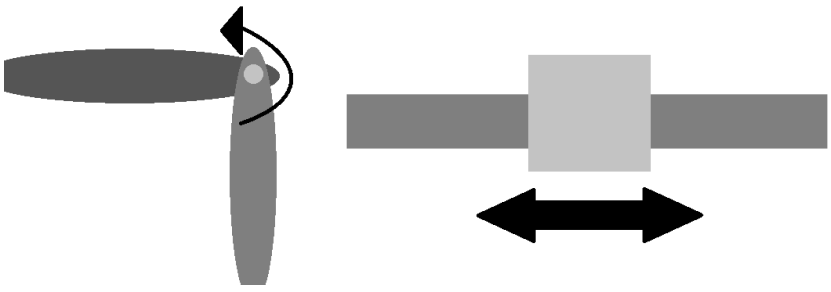
$$ {}^{A}_{C}T = \begin{bmatrix} {}^{A}_{B}R{}^{B}_{C}R & {}^{A}_{B}R^{B}P_{CORG} + {}^{A}P_{BORG} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix} \tag{A.7} $$

While the inverse of a transform, mapping from frame{B} to frame {A} instead of from {A} to frame {B} is given by

$$ {}^{A}_{B}T^{-1} = {}^{B}_{A}T = \begin{bmatrix} {}^{A}_{B}R^{T} & -{}^{A}_{B}R^{T\,A}P_{BORG} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix} \tag{A.8} $$

Joints and Links

Most manipulators are constructed from joints having one degree of freedom. The two main types of joints used in manipulators are revolute joints Figure A.5(a) and prismatic joints Figure A.5(b). A revolute joint provides rotation motion in single axis, while a prismatic joint allows for sliding motion on a single axis. A classic example of the two joints is an inverted pendulum on a cart. The cart would be a prismatic joint, which allows for sliding movement. The inverted pendulum would be a revolute joint attached to a link which allows for rotation about the joints axis.



(a) Revolute Joint (b) Prismatic Joint

Figure A.5 Joint Definition

When defining spacial descriptions of a manipulator the distance and offset of joints are considered. A link is thus described as a rigid body that defines the relationship between two neighboring joint axes of a manipulator [6].

APPENDIX B. PHANTOM OMNI KINEMATICS

Omni Forward Kinematics

The frame attachment for the PHANTOM Omni is shown in Figure B.1, the DH parameters based on this frame assignment are described as in Table B.1.

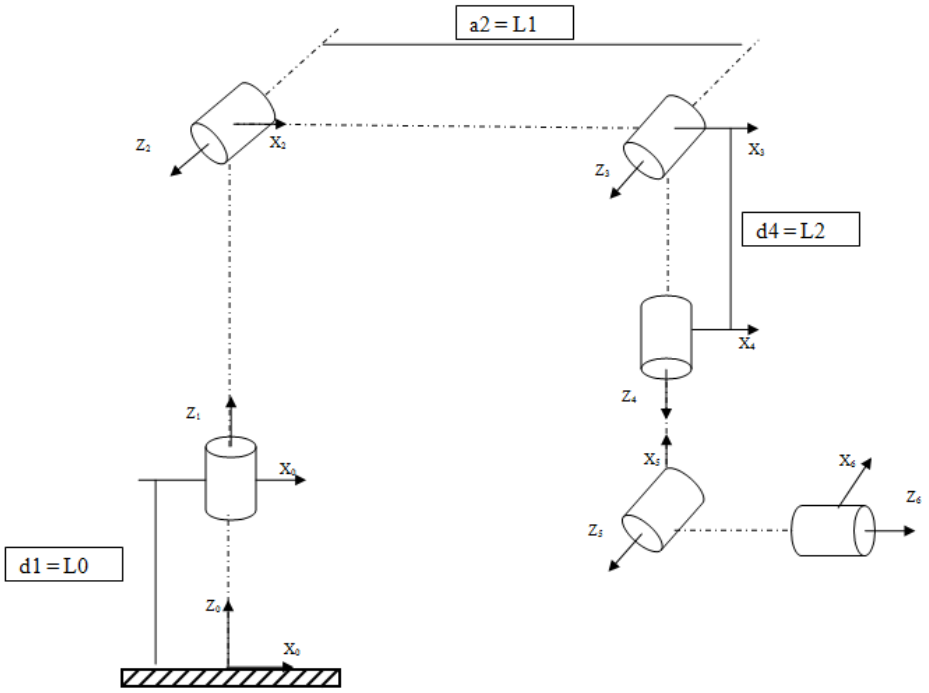


Figure B.1 Frame attachment of PHANTOM Omni

Transforms

Using the DH parameters and the transformation matrix equation, the equations below show the transformations describing each frame in reference to the previous frame in terms of

Table B.1 Denavit Hartenberg Parameters

i	a_{i-1}	α_{i-1}	d_i	θ_i
1	0	0	L_0	θ_1
2	0	90	0	θ_2
3	L_1	0	0	θ_3
4	0	90	L_2	θ_4
5	0	-90	0	$-90+\theta_5$
6	0	-90	0	$90+\theta_6$

the joint angle.Base frame to the first frame

$$ {}^{0}_{1}T = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & 0 \\ s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & L_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.1} $$

First frame to the second frame

$$ {}^{1}_{2}T = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s\theta_2 & c\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.2} $$

Second frame to the third frame

$$ {}^{2}_{3}T = \begin{bmatrix} c\theta_3 & -s\theta_3 & 0 & L_1 \\ s\theta_3 & c\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.3} $$

Third frame to the fourth frame

$$ {}^{3}_{4}T = \begin{bmatrix} c\theta_4 & -s\theta_4 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s\theta_4 & c\theta_4 & 0 & -L_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.4} $$

Fourth frame to the fifth frame

$$ {}^{4}_{5}T = \begin{bmatrix} s\theta_5 & -c\theta_5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ c\theta_5 & s\theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.5} $$

Fifth frame to the sixth frame

$$ {}^{5}_{6}T = \begin{bmatrix} -s\theta_6 & -c\theta_6 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ -c\theta_6 & s\theta_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.6} $$

The end effector can be offset from the last joint due to the length of a link attached to the last joint. This transformation can happen after the kinematics for the system have been described.

$$ {}^{6}_{t}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -L_t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.7} $$

The transformation from the base to the third frame

$$ {}^{0}_{3}T = {}^{0}_{1}T^{1}_{2}T^{2}_{3}T = \begin{bmatrix} c\theta_1(c\theta_2c\theta_3 - s\theta_2s\theta_3) & -c\theta_1(s\theta_2c\theta_3 + c\theta_2s\theta_3) & s\theta_1 & c\theta_1c\theta_2L_1 \\ s\theta_1(c\theta_2c\theta_3 - s\theta_2s\theta_3) & -s\theta_1(s\theta_2c\theta_3 + c\theta_2s\theta_3) & -c\theta_1 & s\theta_1c\theta_2L_1 \\ s\theta_2c\theta_3 + c\theta_2s\theta_3 & c\theta_2c\theta_3 + s\theta_2s\theta_3 & 0 & s\theta_2L_1 + L_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.8} $$

$$ \cos(\theta_1 + \theta_2) = c\theta_{1+2} \tag{B.9} $$

$$ \sin(\theta_1 + \theta_2) = s\theta_{1+2} \tag{B.10} $$

Using trigonometric identities and substituting Eq(B.9) and Eq(B.10) into Eq(B.8) yields

$$ {}^{0}_{3}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T = \begin{bmatrix} c\theta_1 c\theta_{2+3} & -c\theta_1 s\theta_{2+3} & s\theta_1 & c\theta_1 c\theta_2 L_1 \\ s\theta_1 c\theta_{2+3} & -s\theta_1 s\theta_{2+3} & -c\theta_1 & s\theta_1 c\theta_2 L_1 \\ s\theta_{2+3} & c\theta_{2+3} & 0 & s\theta_2 L_1 + L_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.11} $$

The wrist frame is defined as the last three joints which are connected at a point, this defines the orientation of the end effector. The wrist frame in relation to frame three is given by

$$ {}^{3}_{6}T = {}^{3}_{4}T\,{}^{4}_{5}T\,{}^{5}_{6}T = \begin{bmatrix} -c\theta_4 s\theta_5 s\theta_6 + s\theta_4 c\theta_6 & -c\theta_4 s\theta_5 c\theta_6 - s\theta_4 s\theta_6 & c\theta_4 c\theta_5 & 0 \\ c\theta_5 s\theta_6 & c\theta_5 c\theta_6 & s\theta_5 & -L_2 \\ -s\theta_4 s\theta_5 s\theta_6 - c\theta_4 c\theta_6 & -s\theta_4 s\theta_5 c\theta_6 + c\theta_4 s\theta_6 & s\theta_4 c\theta_5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.12} $$

The forward kinematics, transformation of end effector in base frame given from joint angles, is given by

$$ {}^{0}_{6}T = {}^{0}_{3}T\,{}^{3}_{6}T = \begin{bmatrix} r11 & r22 & r33 & P_x \\ r21 & r22 & r23 & P_y \\ r31 & r32 & r33 & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.13} $$

where,

$$
\begin{aligned}
r11 &= c\theta_1\left(-c\theta_{2+3}\left(-c\theta_4 s\theta_5 s\theta_6 + s\theta_4 c\theta_6\right) - s\theta_{2+3}c\theta_5 s\theta_6\right) + \\
&\quad s\theta_1\left(-s\theta_4 s\theta_5 s\theta_6 - c\theta_4 c\theta_6\right) \\
r21 &= s\theta_1\left(-c\theta_{2+3}\left(-c\theta_4 s\theta_5 s\theta_6 + s\theta_4 c\theta_6\right) - s\theta_{2+3}c\theta_5 s\theta_6\right) + \\
&\quad c\theta_1\left(-s\theta_4 s\theta_5 s\theta_6 - c\theta_4 c\theta_6\right) \\
r31 &= s\theta_{2+3}\left(-c\theta_4 s\theta_5 s\theta_6 + s\theta_4 c\theta_6\right) + c\theta_{2+3}c\theta_5 s\theta_6
\end{aligned}
\tag{B.14}
$$

$$
\begin{aligned}
r12 &= c\theta_1\left(-c\theta_{2+3}\left(-c\theta_4 s\theta_5 c\theta_6 + s\theta_4 s\theta_6\right) - s\theta_{2+3}c\theta_5 c\theta_6\right) + \\
&\quad s\theta_1\left(-s\theta_4 s\theta_5 c\theta_6 - c\theta_4 s\theta_6\right) \\
r22 &= s\theta_1\left(-c\theta_{2+3}\left(-c\theta_4 s\theta_5 c\theta_6 + s\theta_4 s\theta_6\right) - s\theta_{2+3}c\theta_5 c\theta_6\right) + \\
&\quad c\theta_1\left(-s\theta_4 s\theta_5 c\theta_6 - c\theta_4 s\theta_6\right) \\
r32 &= s\theta_{2+3}\left(-c\theta_4 s\theta_5 c\theta_6 + s\theta_4 s\theta_6\right) + c\theta_{2+3}c\theta_5 c\theta_6
\end{aligned}
$$

$$
\begin{aligned}
r13 &= c\theta_1\left(c\theta_{2+3}c\theta_4 c\theta_5 - s\theta_{2+3}s\theta_5\right) + s\theta_1 s\theta_4 c\theta_5 \\
r23 &= s\theta_1\left(c\theta_{2+3}c\theta_4 c\theta_5 - s\theta_{2+3}s\theta_5\right) - c\theta_1 s\theta_4 c\theta_5 \\
r33 &= s\theta_{2+3}c\theta_4 c\theta_5 + c\theta_{2+3}s\theta_5
\end{aligned}
$$

$$
\begin{aligned}
P_x &= c\theta_1 s\theta_{2+3}L_2 + c\theta_1 c\theta_2 L_1 \\
P_y &= s\theta_1 s\theta_{2+3}L_2 + s\theta_1 c\theta_2 L_1 \\
P_z &= -c\theta_{2+3}L_2 + s\theta_2 L_1 + L_0
\end{aligned}
$$

Omni Inverse kinematics

The solution of the PHANTOM Omni inverse kinematics finds the joint angles based on the final transformation matrix. Piepers Solution for a 6 DOF manipulator with spherical joint 'wrist' is used to solve for a solution. The location of the final position does not include any orientation information. Thus finding the solution for the transformation to the position of the wrist is useful.

$$ {}^{0}P_{4ORG} = {}^{0}_{1}T^{1}_{2}T^{2}_{3}T^{3}P_{4ORG} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{B.15} $$

From the transformation matrix from the third to fourth frame Eq(B.4), the position is given as

$$ {}^{3}P_{4ORG} = \begin{bmatrix} 0 \\ -L_2 \\ 0 \\ 1 \end{bmatrix} \tag{B.16} $$

The next step it to continue going backwards through the frames and simplify

$$ {}^{2}_{3}T^{3}P_{4ORG} = \begin{bmatrix} c\theta_3 & -s\theta_3 & 0 & L_1 \\ s\theta_3 & c\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ -L_2 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} s\theta_3 L_2 + L_1 \\ -c\theta_3 L_2 \\ 0 \\ 1 \end{bmatrix} \tag{B.17} $$

New variables f_1, f_2, and f_3 are created to further simplify solution

$$ \begin{aligned} f_1 &= s\theta_3 L_2 + L_1 \\ f_2 &= -c\theta_3 L_2 \\ f_3 &= 0 \end{aligned} \tag{B.18} $$

The position vector from the base to the location of the end effector can then be written and

simplified as

$$ {}^{0}P_{4ORG} \begin{bmatrix} c\theta_1 c\theta_2 f_1 - c\theta_1 s\theta_2 f_2 + s\theta_1 f_3 \\ s\theta_1 c\theta_2 - s\theta_1 s\theta_2 f_2 - c\theta_1 f_3 \\ s\theta_2 f_1 + c\theta_2 f_2 + L_0 \\ 1 \end{bmatrix} = \begin{bmatrix} c\theta_1 g_1 - s\theta_1 g_2 \\ s\theta_1 g_1 + c\theta_1 g_2 \\ g_3 \\ 1 \end{bmatrix} \tag{B.19} $$

Where g_1, g_2, and g_3 are defined as

$$ g_1 = c\theta_2 f_1 - s\theta_2 f_2 \tag{B.20} $$

$$ g_2 = -f_3 $$

$$ g_3 = s\theta_2 f_1 + c\theta_2 f_2 + L_0 $$

The magnitude r is found, this can be written as

$$ r = x^2 + y^2 + x^2 \tag{B.21} $$

$$ r = {g_1}^2 + {g_2}^2 + {g_3}^2 \tag{B.22} $$

$$ r = (c\theta_2 f_1 - s\theta_2 f_2)^2 + (-f_3)^2 + (s\theta_2 f_1 + c\theta_2 f_2 + L_0)^2 \tag{B.23} $$

Using Eq(B.18) in Eq(B.23) results in

$$ \frac{r}{2d_1} - z = \tag{B.24} $$
$$ \frac{2a_1(k_1 c\theta_2 + k_2 s\theta_2) + 2d_1 k_4 + d_1 s\alpha_1(k_1 s\theta_2 - k_2 c\theta_2) + K_3}{2d_1} $$
$$ -(s\alpha_1(k1 s\theta_2 - k_2 c\theta_2) + d_1 + k_4) $$

where r and x are given as

$$ r = 2L_1(k_1 c\theta_2 + k_2 s\theta_2) + k_3 \tag{B.25} $$

$$ z = (k_1 s\theta_2 - k_2 c\theta_2) + L_0 \tag{B.26} $$

and k_1, k_2,k_3 are defined as

$$k_1 = f_1 \tag{B.27}$$
$$k_2 = -f_2$$
$$k_3 = {f_1}^2 + {f_2}^2 + {L_0}^2$$
$$k_4 = 0$$

First θ_3 is solved:

$$\frac{r}{2d_1} - z = \frac{2L_0(k_1 c\theta_2 + k_2 s\theta_2) + K_3}{2L_0} - (k1 s\theta_2 - k_2 c\theta_2) - L_0 \tag{B.28}$$

$$\frac{r}{2d_1} - z = \frac{K_3}{2L_0} - L_0 \tag{B.29}$$

$$k_3 = r - 2L_0 z + 2{L_0}^2 = {f_1}^2 + {f_2}^2 + {L_)}^2 \tag{B.30}$$

$$r = {f_1}^2 + {f_2}^2 + {L_)}^2 - 2{L_0}^2 + 2L_0 z \tag{B.31}$$

$$r = (s\theta_3 L_2 + L_1)^2 + (c\theta_3 L_2)^2 - {L_0}^2 + 2L_0 z \tag{B.32}$$

Substitute an algebraic solution by reduction to polynomial given in Eq(B.33) into equaiton (B.32) to get

$$u = \tan\frac{\theta}{2} \tag{B.33}$$
$$\cos\theta = \frac{1-u^2}{1+u^2}$$
$$\sin\theta = \frac{2u}{1+u^2}$$

$$r = \left(\left(\frac{2u_3}{1+u_3^2}\right)L_2 + L_1\right)^2 + \left(\frac{1-u_3^2}{1+u_3^2}\right)^2 L_2^2 - {L_0}^2 + 2L_0 z \tag{B.34}$$

where

$$u_3 = \tan\frac{\theta_3}{2} \tag{B.35}$$

Solving for u_3:

$$u_3 = \frac{2L_1L_2}{r - 2L_0z - {L_1}^2 + {L_0}^2 - {L_2}^2}{}^{+}_{-} \frac{\sqrt{2{L_1}^2{L_2}^2 - 4L_0z{L_1}^2 - r^2 + 2r{L_1}^2 - {L_0}^2 + {L_2}^2 - 4{L_0}^2z^2 + 4{L_0}^3z}}{r - 2L_0z - {L_1}^2 + {L_0}^2 - {L_2}^2} \ldots \ldots \frac{+2{L_1}^2{L_0}^2 - 4L_0z{L_2}^2 + 4rL_0z - L_{14} - {L_0}^4 - {L_2}^4 + 2{L_0}^2{L_2}^2}{r - 2L_0z - {L_1}^2 + {L_0}^2 - {L_2}^2} \tag{B.36}$$

There are two possible solutions for u_3,

$$\theta_3 = 2\tan^{-1}(u_1) \tag{B.37}$$

There are two possible angles for θ_3

Next solve for θ_2:

$$z = ((s\theta_3L_2 + L_1)\, s\theta_2 + (-c\theta_3L_2)\, c\theta_2) + L_0 \tag{B.38}$$

Using the substitution from Eq(B.33), Eq(B.38) becomes

$$z = \left(\frac{2u_3}{1+u_3^2}\right)(s\theta_3L_2 + L_1) - \left(\frac{1-u_3^2}{1+u_3^2}\right) L_2c\theta_3 + L_0 \tag{B.39}$$

Solve for u_2

$$u_2 = \frac{L_1 + s\theta_3 + L_2{}^{+}_{-}\sqrt{{L_1}^2 + 2L_1s\theta_3L_2 + {s\theta_3}^2{L_2}^2 - z^2 + 2zL_0 + {L_2}^2{c\theta_3}^2 - L_0^2}}{z - L_2c\theta_3 - L_0} \tag{B.40}$$

There are four possible solutions for u_2, thus there are four possible solutions for $theta_2$

$$\theta_2 = 2\tan^{-1}(u_2) \tag{B.41}$$

Solve for θ_1

$${}^0P_{4ORG} = \begin{bmatrix} c\theta_1g_1 - s\theta_1g_2 \\ s\theta_1g_1 - c\theta_1g_2 \\ g_3 \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{B.42}$$

$$g_1 = c\theta_2(s\theta_3 L_2 + L_1) - s\theta_2 L_2 c\theta_3 \tag{B.43}$$

$$g_2 = 0;$$

Looking back at the position vector Eq(B.42), using Eq(B.43 and solutions of θ_2 and θ_3, θ_1 is solved.

The Atan2 shown below is a function in C++ that allows for one solution of an angle based on the cosine and sine of that angle.

$$Atan2(\sin, \cos) = \tan^{-1} \frac{\sin(\theta)}{\cos(\theta)} \tag{B.44}$$

θ_1 is then solved

$$\theta_1 = A\tan 2(\sin, \cos) \tag{B.45}$$

$$c\theta_1 = x/g_1 \tag{B.46}$$

$$s\theta_1 = y/g1 \tag{B.47}$$

$$\theta_1 = Atan2(y/g_1, x/g_1) \tag{B.48}$$

There are two possible solutions for θ_3, and four possible solutions for θ_2. Based on this there will be eight possible solutions for θ_1.

Now it is possible to solve for θ_4 , θ_5 , and θ_6. The wrist orientation is shown in Figure B.2. Where the equation for a ZYX Euler rotation matrix is given as

$$R_{Z'Y'X'}(\alpha, \beta, \gamma) = \begin{matrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ s\alpha c\beta & -s\alpha s\beta s\gamma + c\alpha c\gamma & -s\alpha s\beta c\gamma - c\alpha s\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{matrix} \tag{B.49}$$

The inverse of the rotation matrix of the wrist is

$${}^6_4R^{-1}|_{\theta_4=0} *^0_6 R = R_{Z'Y'X'}(\alpha, \beta, \gamma) = \begin{bmatrix} r11 & r12 & r13 \\ r21 & r22 & r23 \\ r31 & r32 & r33 \end{bmatrix} \tag{B.50}$$

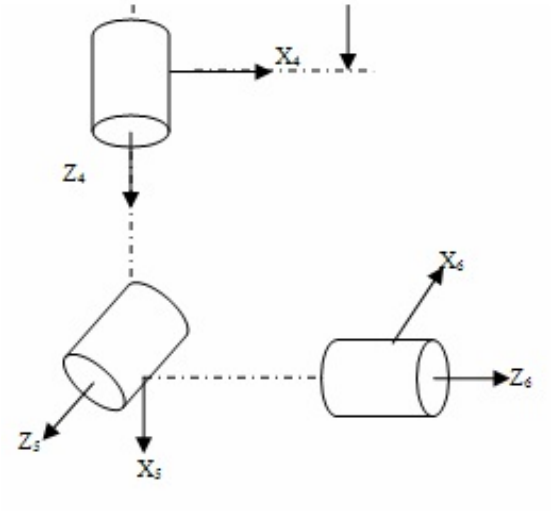


Figure B.2 Euler Wrist

The solution of the three angles are found using the function Atan2. This gives the solution

$$\theta_5 = -\beta = -Atan2(-r31, \sqrt{r32^2 + r33}) \tag{B.51}$$

$$\theta_4 = -\alpha = -Atan2(-r21/c\beta, r11/c\beta) \tag{B.52}$$

$$\theta_6 = \gamma = Atan2(r32/c\beta, r33/c\beta \tag{B.53}$$

Rate kinematics

The rate kinematics are found using the method described in the robotics review section, following velocity propagation joint to joint from the base frame. The Jacobian is then defined based on the solution to the rate kinematics.

Jacobian

$$
{}^{0}_{6}J = \begin{bmatrix} j11 & j12 & j13 & j14 & j15 & j16 \\ j21 & j22 & j23 & j24 & j25 & j26 \\ j31 & j32 & j33 & j34 & j35 & j36 \\ j41 & j42 & j43 & j44 & j45 & j46 \\ j51 & j52 & j53 & j54 & j55 & j56 \\ j61 & j62 & j63 & j64 & j65 & j66 \end{bmatrix} \tag{B.54}
$$

Where

$$j11 = s(\theta_1) * (c(\theta_2)L_1 + L2s(\theta_2)c(\theta_3) + L2c(\theta_2)s(\theta_3))$$

$$j12 = c(\theta_1)(-s(\theta_2)s(\theta_3)L_2 + c(\theta_2)c(\theta_3)L_2 - s(\theta_2)L_1)$$

$$j13 = L_2c(\theta_1)(c(\theta_2)c(\theta_3) - s(\theta_2)s(\theta_3))$$

$$j14 = 0$$

$$j15 = 0$$

$$j16 = 0$$

$$j21 = c(\theta_1)(c(\theta_2)L_1 + L_2s(\theta_2)c(\theta_3) + L_2c(\theta_2)s(\theta_3))$$

$$j22 = s(\theta_1)(-s(\theta_2)s(\theta_3)L_2 + c(\theta_2)c(\theta_3)L_2 - s(\theta_2)L_1)$$

$$j23 = L_2s(\theta_1)(c(\theta_2)c(\theta_3) - s(\theta_2)s(\theta_3))$$

$$j24 = 0$$

$$j25 = 0$$

$$j26 = 0$$

$$j31 = 0$$

$$j32 = c(\theta_2)L_1 + L_2s(\theta_2)c(\theta_3) + L_2c(\theta_2)s(\theta_3)$$

$$j33 = (s(\theta_2)c(\theta_3) + c(\theta_2)s(\theta_3))L_2$$

$$j34 = 0$$

$$j35 = 0$$

$$j36 = 0$$

$$j41 = 0$$

$$j42 = s(\theta_1)$$

$$j43 = s(\theta_1)$$

$$j44 = c(\theta_1)(s(\theta_2)c(\theta_3) + c(\theta_2)s(\theta_3))$$

$$j45 = c(\theta_1)s(\theta_4)c(\theta_2)c(\theta_3) + c(\theta_1)s(\theta_4)s(th_2)s(\theta_3) + s(\theta_1)c(\theta_4)$$

$$j46 = c(\theta_5)c(\theta_4)c(\theta_1)c(\theta_2)c(\theta_3) - c(\theta_5)c(\theta_4)c(\theta_1)s(\theta_2)s(\theta_3) +$$

$$c(\theta_5)s(\theta_1) * s(\theta_4) - s(\theta_5)c(\theta_1)c(\theta_2)s(\theta_3) - s(\theta_5)c(\theta_1)s(\theta_2)c(\theta_3)$$

$$j51 = 0$$

$$j52 = -c(\theta_1)$$

$$j53 = -c(\theta_1)$$

$$j54 = s(\theta_1)(s(\theta_2)c(\theta_3) + c(\theta_2)s(\theta_3))$$

$$j55 = s(\theta_1)s(\theta_4)s(\theta_2)s(\theta_3) - s(\theta_1)s(\theta_4)c(\theta_2)c(\theta_3) - c(\theta_1)c(\theta_4)$$

$$j56 = -c(\theta_5)c(\theta_4)s(\theta_1)s(\theta_2)s(\theta_3) + c(\theta_5)c(\theta_4)s(\theta_1)c(\theta_2)c(\theta_3) -$$

$$c(\theta_5)c(\theta_1)s(\theta_4) - s(\theta_5)s(\theta_1)c(\theta_2)s(\theta_3) - s(\theta_5)s(\theta_1)s(\theta_2)c(\theta_3)$$

$$j61 = 1$$

$$j62 = 0$$

$$j63 = 0$$

$$j64 = s(\theta_2)s(\theta_3) - c(\theta_2)c(\theta_3)$$

$$j65 = -s(\theta_4)(s(\theta_2)c(\theta_3) + c(\theta_2)s(\theta_3))$$

$$j66 = c(\theta_5)c(\theta_4)s(\theta_2)c(\theta_3) + c(\theta_5)c(\theta_4)c(\theta_2)s(\theta_3) - s(\theta_5)s(\theta_2)s(\theta_3) + s(\theta_5)c(\theta_2) * c(\theta_3)$$

APPENDIX C. TRIGONOMETRIC IDENTITIES

Trigonometric identities will be used to simplify mathematical models of the system, some common identities are shown

$$\cos(\theta_1 + \theta_2) = \cos(\theta_1)\cos(\theta_2) - \sin(\theta_1)\sin(\theta_2) \tag{C.1}$$

$$\sin(\theta_1 + \theta_2) = \cos(\theta_1)\sin(\theta_2) + \sin(\theta_1)\cos(\theta_2) \tag{C.2}$$

$$\sin(-90 + \theta) = -\cos(\theta) \tag{C.3}$$

$$\cos(-90 + \theta) = \sin(\theta) \tag{C.4}$$

$$\sin(90 + \theta) = \cos(\theta) \tag{C.5}$$

$$\cos(90 + \theta) = -\sin(\theta) \tag{C.6}$$

$$\cos(\theta_1) = c\theta_1 \tag{C.7}$$

$$\sin(\theta_1) = s\theta_1 \tag{C.8}$$

$$\cos(\theta_1 + \theta_2) = c\theta_{1+2} \tag{C.9}$$

$$\sin(\theta_1 + \theta_2) = s\theta_{1+2} \tag{C.10}$$

APPENDIX D. CODE

C++ code necessary for programming PHANTOM Omni.

Implementation of Theory

Listing D.1 Code: Define Variables

```
1  //Center of VM- state variable
2  pivotLoc[0] = 0;pivotLoc[1] = 0;pivotLoc[2] = 0;
3
4  //length of VM- state variable
5  Lv[0]=50;
6
7  //GAIN FOR POSITION ERROR
8  float Kp1 = 1;
9
10 //GAIN FOR ANGLE ERROR
11 float gain = Kp1*3.14*Lv[0];
12
13 //initialize vectors for matrix operations
14 #define n 6
15 #define m 1
16 #define p 1
17
18 float JvT[m][n];
19 float JvTWA[m][n];
20 float JvTWAJv[m][m];
21 float JvTWAJvinv[m][m];
22 float JvTWAJvinv_JvT[m][n];
23 float Jv_JvTWAJvinv_JvT[n][n];
24 float WAJv_JvTWAJvinv_JvT[n][n];
25 float IminusWAJv_JvTWAJvinv_JvT[n][n];
26 float Kpe[n][p];
27 float fmotion[n][p];
28 float eerror[n];
29 float JV[n];
30 float fx;
31 float fy;
32 float fz;
33
34 //initialize vectors for system values
```

```
float perror[n][p] = {{eerror[0]},{eerror[1]},{eerror[2]},{eerror[3]},{eerror[4]},{eerror[5]}};
//Jacobian is a different size for each 1 DOF and 2 DOF, needs to be initialized based on size
float I[n][n] = {{1,0,0,0,0,0},{0,1,0,0,0,0},{0,0,1,0,0,0},{0,0,0,1,0,0},{0,0,0,0,1,0},{0,0,0,0,0,1}};
float WA[n][n] = {{1,0,0,0,0,0},{0,1,0,0,0,0},{0,0,1,0,0,0},{0,0,0,1,0,0},{0,0,0,0,1,0},{0,0,0,0,0,1}};
float kkp[n][n] = {{Kp1,0,0,0,0,0},{0,Kp1,0,0,0,0},{0,0,Kp1,0,0,0},
 {0,0,0,gain,0,0},{0,0,0,0,gain,0},{0,0,0,0,0,gain}};
```

Listing D.2 Code: Matrix Operations

```
//Use Matrix Operations from ``Kalman Filtering'' by Dan Simon
 matrix_transpose((float*) Jv, n, m, (float*) JvT);
 matrix_multiply((float*) JvT,(float*) WA, m, n, n,(float*) JvTWA);
 matrix_multiply((float*) JvTWA, (float*) Jv, m, n, m, (float*) JvTWAJv);
 matrix_inversion((float*) JvTWAJv, m, (float*) JvTWAJvinv);
 matrix_multiply((float*) JvTWAJvinv, (float*) JvT, m, m,n,(float*) JvTWAJvinv_JvT);
 matrix_multiply((float*) Jv, (float*) JvTWAJvinv_JvT, n, m,n,(float*) Jv_JvTWAJvinv_JvT);
 matrix_multiply((float*) WA, (float*) Jv_JvTWAJvinv_JvT, n, n,n,(float*) WAJv_JvTWAJvinv_JvT);
 matrix_subtraction((float*) I, (float*) WAJv_JvTWAJvinv_JvT, n,n,(float*) IminusWAJv_JvTWAJvinv_JvT);
 matrix_multiply((float*) kkp, (float*) perror, n, n, p, (float*) Kpe);
 matrix_multiply((float*) IminusWAJv_JvTWAJvinv_JvT, (float*)Kpe,n, n, p, (float*) fmotion);
```

Listing D.3 Code: Appl force to haptic device

```
//get force from matrix operations
fx = fmotion[0][0];
fy = fmotion[1][0];
fz = fmotion[2][0];

//send force to state variable for haptic device
force[2] = fz;  //keep in the x y plane
force[0] = fx;
force[1] = fy;

//Save all components into a state variable to export
state.compforce[0] = fmotion[0][0];
state.compforce[1] = fmotion[1][0];
state.compforce[2] = fmotion[2][0];
state.compforce[3] = fmotion[3][0];
state.compforce[4] = fmotion[4][0];
state.compforce[5] = fmotion[5][0];
```

Code for One Degree of Freedom

Listing D.4 Code: One Degree of Freedom

```
29  //  Virtual Manipulator in XY plane fixed to a point
30
31 //define angles
32 float thetaV1=atan2(state.position[1]-pivotLoc[1],state.position[0]-pivotLoc[0]); //angle about z
33 float thetaR1=atan2(state.transform[9],state.transform[8]); //Orientation angle about z axis
34
35 //define error
36 eerror[0]=pivotLoc[0]+Lv[0]*cos(thetaV1)-state.position[0]; //error in x
37 eerror[1]=pivotLoc[1]+Lv[0]*sin(thetaV1)-state.position[1]; //error in y
38 eerror[2]=pivotLoc[2]-state.position[2];     //error in z
39 eerror[3]=0;          //error in thetax
40 eerror[4]=0;          //error in thetay
41 eerror[5]=thetaV1-thetaR1;       //error in thetaz
42
43 //define Jacobian
44 float Jv[n][m] = {{JV[0]},{JV[1]},{JV[2]},{JV[3]},{JV[4]},{JV[5]}};
45 JV[0] = -Lv[0]*sin(thetaV1);
46 JV[1] = Lv[0]*cos(thetaV1);
47 JV[2] = 0;
48 JV[3] = 0;
49 JV[4] = 0;
50 JV[5] = 1;
51
52 //Save angles to State variable for use in graphics
53 state.thetar[0] = thetaR1;
54 state.thetav[0] = thetaV1;
55 fnow[0] = force[1];
```

Code for Three Degrees of Freedom

Listing D.5 Code: Three Degrees of Freedom

```
// Virtual Manipulator in XYZ plane fixed to a point

//Create length variable for finding angles
float LL = (sqrt((state.position[0]-pivotLoc[0])*(state.position[0]-pivotLoc[0])
+ (state.position[1]-pivotLoc[1])*(state.position[1]-pivotLoc[1])
+ (state.position[2]-pivotLoc[2])*(state.position[2]-pivotLoc[2])));

//define angles
float thetaV1 = (atan2(state.position[1]-pivotLoc[1],state.position[0]-pivotLoc[0]));  //angle about z
float thetaV2 = -PI-acos((state.position[2]-pivotLoc[2])/(LL));     //angle about x
float thetaV3 = 0;           //rotation about y

float thetaR1 = atan2(state.transform[9],state.transform[8]);    //Orientation angle about z axis
float thetaR2 = -PI-acos((state.transform[10]));      //Orientation angle about x axis
float thetaR3 = 0;          //Orientation angle about y axis

float eerror[6];//position error
eerror[0] = pivotLoc[0]+Lv[0]*sin(thetaV2)*cos(thetaV1)-state.position[0];  //error in x
eerror[1] =pivotLoc[1]+Lv[0]*sin(thetaV2)*sin(thetaV1)+pivotLoc[1]- state.position[1]; //error in y
eerror[2] = pivotLoc[2]-Lv[0]*cos(thetaV2)- state.position[2];      //error in z
eerror[3] = 0;            //error in thetax
eerror[4] = thetaV2-thetaR2;          //error in thetay
eerror[5] = thetaV1-thetaR1;          //error in thetaz

////Compute Jacobian Based on Initial Angles found above
float Jva[9];
Jva[0] = -Lv[0]*(sin(thetaV1))*(sin(thetaV2));
Jva[1] = Lv[0]*(cos(thetaV1))*(cos(thetaV2));
Jva[2] = 0;
Jva[3] = Lv[0]*(cos(thetaV1)*sin(thetaV2));
Jva[4] = Lv[0]*(sin(thetaV1))*(cos(thetaV2));
Jva[5] = 0;
Jva[6] = 0;
Jva[7] = Lv[0]*sin(thetaV2);
Jva[8] = 0;

float Jvb[9];
Jvb[0] = 0;
Jvb[1] = sin(thetaV1); //check the sign on this!!!
Jvb[2] = cos(thetaV1)*sin(thetaV2);
Jvb[3] = 0;
Jvb[4] = -cos(thetaV1);
Jvb[5] = sin(thetaV1)*sin(thetaV2);
Jvb[6] = 1;
Jvb[7] = 0;
Jvb[8] = -cos(thetaV2);

float Jv[n][m] = {{Jva[0], Jva[1], Jva[2]},{Jva[3], Jva[4],Jva[5]}, {Jva[6], Jva[7], Jva[8]},
 {Jvb[0], Jvb[1],Jvb[2]},{Jvb[3],Jvb[4], Jvb[5]},{Jvb[6],Jvb[7], Jvb[8]}};
```

```
50
51 //Save angles into global variable to be used in graphics
52 state.thetar[0] = thetaR1;
53 state.thetar[1] = thetaR2;
54 state.thetav[0] = thetaV1;
55 state.thetav[1] = thetaV2;
```

BIBLIOGRAPHY

[1] Greg R. Luecke John A. Beckman, "Haptic Interactions With Under-Actuated Robots Using Virtual Mechanisms," *2008 IEEE International Conference on Robotics and Automation*, Pasadena, CA, USA, May 19-23, 2008.

[2] Lawton N. Verner Allison M. Okamura, "Force & Torque Feedback vs Force Only Feedback," *Third Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, March 2009.

[3] S. Wang and M. A. Srinivasan, "The role of torque in haptic perception of object location in virtual ennvironments," *In Proceedings of 11th Symposium on Haptic Interfaces for Virtual Environments and Teleoperator Systems*, pages 302-309, 2003.

[4] Alejandro Jarillo-Silva, "PHANToM OMNI Haptic Device: Kinematic and Manipulability," *2009 Electronics, Robotics and Automotive Mechanics Conference*, 2009.

[5] David C. Lay, *Linear Algebra and It's Applications*, Pearson Education, third edition, update, 2006.

[6] John J. Craig, *Introduction to Robotics Mechanics and Control*, New Jersey: Pearson Prentice Hall, third edition, 2005.

[7] Katsuhiko Ogata, *Modern Control Engineering*, New Jersey: Prentice Hall, fourth edition, 2002.

[8] SensAble, *OPENHAPTICS TOOLKIT version 3.0 PROGRAMMERS GUIDE*, SensAble Technologies, Inc, 2009.

[9] SensAble, *OPENHAPTICS TOOLKIT version 3.0 API REFERENCE MANUAL*, SensAble Technologies, Inc, 2008.

[10] Edward Angel, *Interactive computer graphics : a top-down approach with OpenGL*, Boston : Addison Wesley, 2003.

[11] John Albert Beckman, *The PHANTOM omni as an under-actuated robot*, MS thesis, Iowa State University, Ames, IA, 2007.

[12] Dan Simon, "Kalman Filtering," *Embedded Systems Programming*, pages 72-79, 2001.