ABSTRACT

MULTI-DEGREE OF FREEDOM TELEMANIPULATION IN AN UNSTRUCTURED ENVIRONMENT

by
Diego Ramirez

Two approaches to 6-degrees-of-freedom telemanipulation that will accommodate different needs and skills of potential users with congenital amputation, arthrogryposis, muscular dystrophy and cerebral palsy were developed. One method uses scalable movements (i.e. position and orientation), and the second employs isometric forces and torques without movement. The scalable position approach employs a 6-degress-of-freedom electromechanical stylus whose joint orientations are used by the controlling computer to determine the position and orientation of the robot's end effector via inverse kinematics or by one-to-one matching of the stylus joint angles with those of the manipulator. The isometric method uses the measured forces and torques to define velocities in X-Y-Z and in pitch, roll and yaw of the end effector. The latter is accomplished using the pseudo-inverse Jacobian to define a rate resolved controller, with a novel form of damping to minimize instabilities at singularities. Both forms of telemanipulation have been implemented using Matlab and Simulink. A fully interactive, stereo VRML model of a commercial robot has been developed using the Matlab VRML Toolbox. This robot model is driven in real-time to allow evaluation of the telemanipulation methods, and serve as an eventual user-training environment.

MULTI-DEGREE OF FREEDOM TELEMANIPULATION IN AN UNSTRUCTURED ENVIRONMENT

by
Diego Ramirez

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Biomedical Engineering

Department of Biomedical Engineering

January 2007

APPROVAL PAGE

MULTI-DEGREE OF FREEDOM TELEMANIPULATION IN AN UNSTRUCTURED ENVIRONMENT

Diego Ramirez

Dr. Richard Foulds, Thesis Advisor Date
Associate Professor of Biomedical Engineering, NJIT

Dr. Sergei Adamovich, Committee Member Date
Assistant Professor of Biomedical Engineering, NJIT

Dr. Lisa Simone, Committee Member Date
Assistant Research Professor of Biomedical Engineering, NJIT

BIOGRAPHICAL SKETCH

Author:	Diego Alejandro Ramirez
Degree:	Master of Science
Date:	January 2007

Undergraduate and Graduate Education:

- Master of Science in Biomedical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2006
- Bachelor of Science in Electrical Engineering,
  Pontificia Universidad Javeriana, Cali, Colombia, 2000

Major:	Biomedical Engineering

Para vos… porque no lo has pensado dos veces para caminar conmigo.
(For you… since you haven't had second thoughts to walk with me.)

ACKNOWLEDGMENT

I would like to express my deepest appreciation to Dr. Richard Foulds, who not only served as my research supervisor, providing valuable and countless resources, insight, and intuition, but also constantly gave me support, encouragement, and reassurance. Special thanks are given to Dr. Sergei Adamovich and Dr. Lisa Simone for actively participating in my committee.

Many of my fellow graduate students in the Neuromuscular Engineering Laboratory are deserving of recognition for their support.

I also wish to give special recognition to the Rehabilitation Engineering Research Center, which provided the resources to develop this work successfully.

TABLE OF CONTENTS

Chapter **Page**

1 INTRODUCTION.. 1

1.1 Organization of this Document.. 3

1.2 Objective.. 4

2 BACKGROUND.. 6

2.1 RERC.. 6

2.2 Robotics Theory.. 7

2.2.1 Overview.. 7

2.2.2 Definitions.. 7

2.2.3 Spatial Description of a Robotic Manipulator.. 8

2.3 Kinematics.. 13

2.3.1 Forward Kinematics.. 15

2.3.2 Inverse Kinematics.. 16

3 ELEMENTS OF THE CONTROL SYSTEM.. 21

3.1 Model Schematics.. 21

3.2 Controllers.. 22

3.2.1 Immersion Probe and Personal Digitizer.. 22

3.2.2 Spaceball 5000.. 24

3.3 Software Applications.. 24

3.3.1 Virtual Reality Viewer.. 24

TABLE OF CONTENTS
(Continued)

Chapter	Page
3.3.2 Robotics Toolbox – Matlab	27
3.3.3 Virtual Reality Toolbox – Matlab	29
3.3.4 Simulink	29
4 IMPLEMENTATION	30
4.1 Communication Protocol	30
4.1.1 Immersion Probe and Personal Digitizer	30
4.1.2 Spaceball 5000	31
4.2 Design of the Virtual Model	33
4.3 Control Models	38
4.3.1 Immersion Probe - Position Control	38
4.3.2 Spaceball – Force Control	38
4.4 Computational Application	40
4.4.1 Immersion Probe – Scalable Control of Position	40
4.4.2 Spaceball – Control of Isometric Force	44
5 RESULTS	55
5.1 Scalable Position – Immersion Probe	55
5.2 Isometric Forces – Spaceball 5000	56
6 CONCLUSIONS AND FUTURE WORK	60
REFERENCES	62

LIST OF FIGURES

Figure		Page
2.1	Model of a manipulator designed in VRML	11
2.2	Schematic of two attached joints in a robot	12
2.3	Difference form of Denavit-Hartenberg notation	13
3.1	Conceptual Sketch of a robot manipulator control system	21
3.2	Immersion Probe ™. Developed by Immersion Corporation	23
3.3	Spaceball® 5000. Developed by 3D Connexion	24
3.4	Cartesian, right-handed, three-dimensional coordinate system	26
3.5	Typical screen of VRealm Builder™, the VRNL editor used to make the model of the manipulator	27
3.6	Schematic model of a robotic manipulator built in Matlab's Robotics Toolbox	28
4.1	Control block of the Spaceball in the Virtual Reality Toolbox	33
4.2	Example of rotation for a geometric shape in VRealm Builder	34
4.3	Drawing of the ARM Manipulator. Source: Exact Dynamics	35
4.4	Drawing of the ARM Manipulator. Source: Exact Dynamics	36
4.5	Model of the ARM Manipulator built in VRealm Builder, the VRML editor used for this work	37
4.6	Control Model for the stylus controller. Scalable position approach	38
4.7	Control model for the Spaceball controller. Isometric approach	39
4.8	Schematic control system for the Spaceball controller, made in Simulink	44
4.9	Model of the ARM Manipulator built in the Matlab's Robotics Toolbox	46

CHAPTER 1

INTRODUCTION

In response to the necessities detected by the Biomedical Engineering Department at NJIT, it joined, in conjunction with the Children's Specialized Hospital, the Rehabilitation Engineering Research Center (RERC) on Technology for Children with Orthopedic Disabilities in 2005, with the objective of implementing a series of projects and developing state-of-the-art technology to allow improvements in the rehabilitation processes and independent living of children with physical impairments.

Six projects are being implemented, three of them focused on research and the other three in the development of existing technologies. These projects intend to fulfill some of the necessities to improve quality of life for children with cerebral palsy, arthrogryposis, contracture due to burns, congenital and traumatic amputations, polio and bone diseases.

The first stage of one of the projects presented by RERC is being addressed by this work; this project is a multi-degree control of a wheelchair-mounted manipulator that provides the user with increased independent living in an unstructured environment. This first stage comprises the development of new human-machine interfaces as well as a virtual reality-based training environment. It means that, before implementing the control mechanisms in the actual manipulator, these mechanisms will be designed and tested in a 3D virtual environment in a scaled model from the actual one. The control strategy to be implemented takes some characteristics of the methodology known as Extended

Physiological Proprioception (EPP)[1], which intends to give the user the sensation of using the manipulator as an extension of the body and a replacement of the impaired limb. Based on this concept, two types of controllers are used according to the differences in the necessities and limitations of the users, mainly children with cerebral palsy, arthrogryposis, muscular dystrophy and congenital amputations. These controllers are intended to operate the manipulator by replication of the movements or forces generated by the user and applied to them in direct correspondence with their directions and orientations. The first one uses a 6-degree-of-freedom stylus-like scaled manipulator, which uses joint angles as control input (i.e., scalable position approach). The second one, also with six degrees of freedom, responds to applied torques and forces and does not require displacement to execute the required tasks (i.e., isometric force and torque approach).

The mathematical computation is based on the Denavit-Hartenberg (D-H) model, which will be explained later, and brings as outputs, the joint angles that feed the model of the manipulator, built in Virtual Reality Modeling Language (VRML); these are the angles that will allow positioning the tip of the manipulator in the desired position and orientation in space.

1.1 Organization of this Document

The first chapter of this document introduces a brief picture of the total work and exposes its objectives, showing the goals that are intended to be reached. Some terms enounced will be explained later in the next chapters.

Chapter two addresses the background required to create a reference frame for the implementation of the work. First, the RERC creation and foundations will be presented, as the entity supporting the development of the whole project. Specific definitions and technical theory, necessary for the implementation of the model, are summarized in the second part of this chapter.

Chapter three focuses in the exposition of the control system and the different elements to be integrated into it. A block diagram shows schematically what is intended to control the manipulator and both, the hardware and software, are pointed out, as well as their role within the whole system.

Chapter four is dedicated to the explanation of the implementation procedure and the integration of software and hardware to comply with the specified tasks. Key parts of the code are introduced for better understanding of the implementation.

Chapters five and six brings the results, conclusions and futures work. The results are presented to show how the system behaves and which limitations were detected. Conclusions and future work are presented together to show the potentialities of the system and its future applications.Primary Objective

The primary aim of this work is to control, in real time, a 3D model of a 6-degree-of-freedom (6 dof) robotic manipulator, featuring independently position and force as control variables. In other words, two different and independent controllers will be used to drive in real time a model of a robotic arm, by using either position or force applied to the respective controller.

Several intermediate tasks must be achieved to reach the goal of the creation of this controlled environment:

1.2 Objective

To create a virtual 3D model of a 6-dof robotic manipulator according to the specifications of the Assistive Robotic Manipulator (ARM), designed by Exact Dynamics from Netherlands. (http://www.exactdynamics.nl/). (Note: The ARM Manipulator is a redesigned version of another manipulator, the Manus, a wheelchair-mounted seven-axis robot designed, for people with special needs and controlled in unstructured environments[2]. The ARM incorporates a number of improvements including a new computer interface that allows the development of new manipulator control strategies).

To create independent communication protocols between the proposed controllers:

- Immersion Probe® and Personal Digitizer® by Immersion Corporation, for the scalable position approach.
- Spaceball 5000® by 3Dconnexion, for the isometric force approach,

the application tool for computation:

- Matalb™ by Mathworks,

and the virtual reality builder tool:

- VRealm™ Builder by Data Systems

To use the kinematics properties of robotic models to determine the angles of the joints for the virtual model of the 6-dof robotic manipulator.

To direct the end-effector of the virtual model to reach a determined point in its workspace by controlling either the position or the force applied to determined controllers.

CHAPTER 2

BACKGROUND

2.1 RERC

Independent living for a person requires the development of daily tasks whether they are simple or complex. Even the simplest tasks may become extremely difficult if a subject does not have the required skills or specific tools that facilitate their development. This is the case of people with special needs. In conjunction with the Children's Specialized Hospital, the Biomedical Engineering Department at NJIT was granted with the Rehabilitation Engineering Research Center (RERC) on Technology for Children with Orthopedic Disabilities. This RERC assemblies a set of projects intended for fulfilling the fields of research and development using state-of-the-art technology.

This application includes six projects, three of them focused in research and the other three addressing development as well as training projects serving the needs of children, families, students and professionals.

The RERC was granted in 2005 and the projects involve the application of virtual reality gaming systems in combination with robot-assisted therapy for rehabilitation purposes.

2.2 Robotics Theory

2.2.1 Overview

According to Khalil and Dombre, a robot is an automatic system that can be programmed and controlled automatically, and characterized for having several degrees of freedom[3] or position variables (this term will be defined more accurately later). Hence, one can think of a robot as a mechanical device that can be programmed and which configuration facilitates human tasks, either by emulating human movements or developing actions that are difficult for a person to do by him/herself.

Examples of types of robots are[4]:

a. Manipulators: Imitating movements of human arms.
b. Walking robots: Imitating translation of humans and animals.
c. Mobile robots: used mainly for transportation of specific loads.

2.2.2 Definitions

Some definitions that characterize a mechanically robot are required. Although some of them are general, these definitions apply specifically to manipulators and walking robots. Mobile robots have different configurations and specifications and they are not part of the scope of this work:

Link: Each one of the physical components of a manipulator, adjacent one to the other, rigid or semi-rigid.

Joints: Connections between two adjacent links, which allow movement of one of them, relative to the other. They can be either prismatic or revolute, depending on the way that the links conform the joint displace one relative to the adjacent one. Revolute joints describe angular displacements respect to the other, while prismatic joints characterize by linear displacement along the same axis, one referenced to the other.

Degrees of freedom: Independent displacement variables that shall be intended to define exactly the position in space of every part of the robot/manipulator.

End-effector: Element located at the end of the chain of links, where the tool to develop the required task shall be located (i.e., a grip, a drill, etc). The final description of a given position for a robot is given according to the location and the reference frame of the end-effector. This is obvious if it is understood that it is the tool what has to be located in a defined position.

Reference Frame: Coordinate system attached to each robotic object (i.e., joints for manipulators), which allows describing its location and orientation in space.

Workspace: In the most general terms, one can define the workspace as the points in space reachable by the end-effector of the robotic manipulator.

2.2.3 Spatial description of a robotic manipulator

In the developing of its regular tasks, a robot requires continuous movement in space, which is associated with changing position and orientation of the different parts of the manipulator. These changes must be possible to be described and represented in a conventional way, allowing for mathematical manipulation.

A frame is the entity used to describe an object in space; it has four components or vectors grouped in two independent objects:

Position vector: A 3x1 entity used to describe the position of an object according to a determined reference frame or coordinate system.

Rotation matrix: A 3x3 object that defines the orientation of a coordinate system attached to the object described, according to the reference coordinate system.

The entity that can totally describe or map a quantity in terms of various reference frames is called a homogeneous transform[5]. It combines the position vector and the rotation matrix to create the homogeneous coordinates in Cartesian space. The result is a 4x4 matrix containing both in a way that if

$$^{i}\vec{P} = \begin{bmatrix} ^{i}P_x \\ ^{i}P_y \\ ^{i}P_z \end{bmatrix}$$

is the position vector of the point P in the reference frame i, and,

$$^{i}R_j = \begin{bmatrix} ^{i}s_j & ^{i}n_j & ^{i}a_j \end{bmatrix} = \begin{bmatrix} s_x & n_x & a_x \\ s_y & n_y & a_y \\ s_z & n_z & a_z \end{bmatrix}$$

is the rotation matrix of frame j relative to frame i, the homogeneous transform would be given by,

$$^{i}T_{j} = \begin{bmatrix} & ^{i}R_{j} & & ^{i}P_{j} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where, the last row (i.e., [0 0 0 1]) has been added to cast rotation and translation into this one single matrix, named the homogeneous transform[6]. $^{i}s_{j}$, $^{i}n_{j}$, $^{i}a_{j}$ represent the unit vectors of reference frame F_j along x_j, y_j, z_j axes respectively, according to the reference frame F_i. $^{i}P_{j}$ represents the coordinates of the reference frame F_j in terms of the frame F_i.

The entities described above may be interpreted as operators that are used to represent the displacement of any element of a robot either in position, orientation or both.

In terms of robotic manipulators, two or more rigid-bodies are connected one to the other by one-degree-of-freedom joints; these rigid bodies are called links. A link, thus, is the rigid-body that defines the relationship between two adjacent joint axes of a manipulator. A joint can be either revolute or prismatic and its axis is the imaginary line about which a link rotates according to its predecessor[7]. Figure 2.1 shows a model of manipulator designed in a 3D Virtual Reality software application based on the programming language VRML (Virtual Reality Modeling Language, which will be described later).

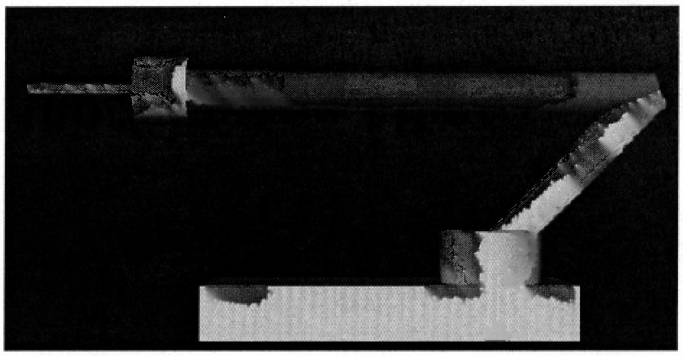


Figure 2.1. Model of a manipulator designed in VRML

Each link has associated a set of variables and parameters that describes it according to its location in space and connection to the other links in the manipulator. Denavit and Hartenber developed a method to describe mechanisms using four basic types of quantities[8] or parameters. This convention, known as the D-H model, defines an independent coordinate system or reference frame attached to each link. The parameters used by the D-H model are:

Link length (a_i): The mutually perpendicular line between two adjacent joint axes z_i.

Link twist (α_i): The angle between these two adjacent joint axes about the link length a_i already defined.

Link offset (d_i): The distance along a joint axis z_i between the end of a link length and the beginning of the next one.

Joint angle (θ_i): The angle between two link lengths about the common joint axis z_i.

Figure 2.2 shows a schematic of two links of a robot with its different associated parameters.

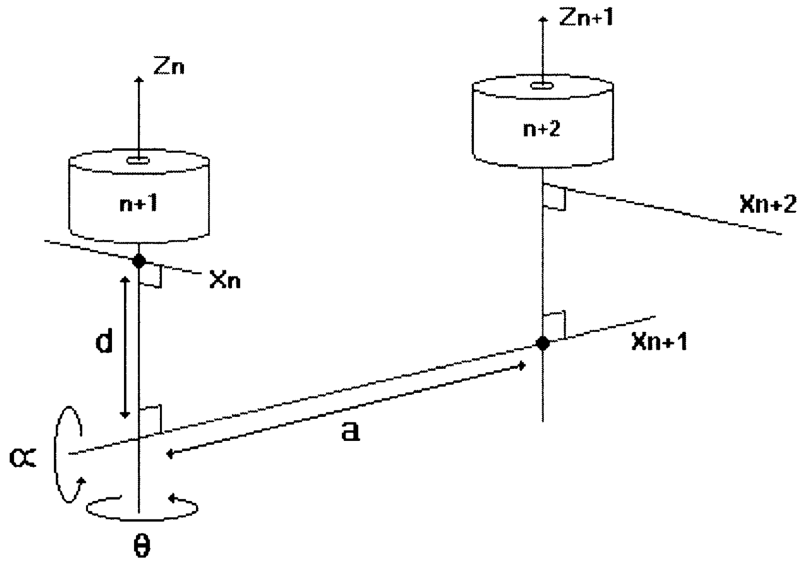


Figure 2.2. Schematic of two attached joints in a robot.
Source http://www.ee.unb.ca/tervo/ee4353/dhxform.htm. Oct. 2006.

When studying movement in robotics it is important to differentiate whether the forces causing these movements are of interest or not. In such way, two different fields arise: Kinematics which relates to movements without involving any applied force to cause or affect it, and dynamics which is the science that takes into account all the forces required to generate such changes.

Reference frames can be assigned in two different ways:

1. Frame i has its origin along the axis of joint i+1. This is called the "Standard D-H form".
2. Frame i has its origin along the axis of joint i. This is referred as the "Modified D-H form".

Figure 2.3 illustrates the two types of notation defining two adjacent links.

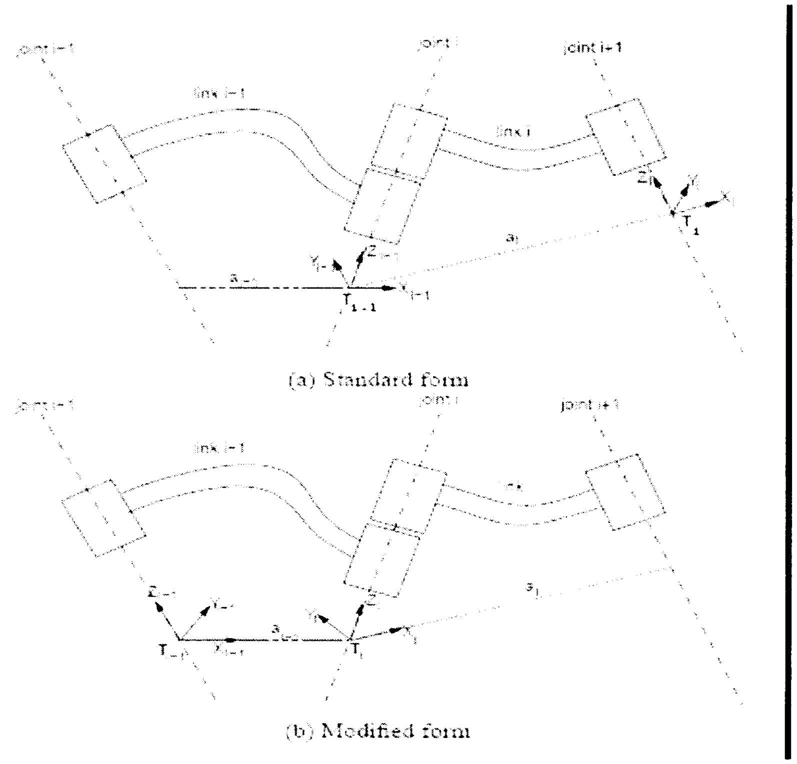


Figure 2.3. Difference form of Denavit-Hartenberg notation. Source: Corke, Peter I. Robotics Toolbox for Matlab. Release 6. http://www.cat.csiro.au/cmst/staff/pic/robot. Australia 2001.

2.3 Kinematics

Kinematics defines the description of properties of motion in time and space. For this purpose, the position of an object, as well as the multiple variables derived from this one (e.g. velocity, accelerations, gradients, etc.) must be analyzed and solved for an specific reference frame; then it has to be possible to do a mathematical representation of the model referred to another object's reference frame. Kinematics may be translational, rotational or both or may apply to one or many object or rigid-bodies.

The motion of one body A with respect to another body B is defined as the movement of B with respect to a common reference frame O minus the movement of A with respect to O. If these motions are considered being vectors (i.e., with magnitude and direction), we can derive this description from the definition of vector's addition:

$$\vec{O} = \vec{A} + \vec{B}$$

or,

$$\vec{A} = \vec{O} - \vec{B}$$

These relative motion equations, when interpreted as velocity vectors, allow analysis of motion according to any specified body in a system:

$$\vec{V}_{A/B} = \vec{V}_A - \vec{V}_B$$

To solve the final position of the link n of a manipulator (with reference frame located in the end-effector) relative to link 0 (which is associated to the origin of the manipulator; this origin is usually the base of the robot and is bound to the reference frame 0), a series of subsequent transformations must be derived and later on, concatenated. The general form of a transformation matrix for a robotic manipulator, for subsequent reference frames, is expressed as:

$$ {}^{n}T_{n+1} = \begin{bmatrix} c\theta_{i+1} & -s\theta_{i+1} & 0 & a_i \\ s\theta_{i+1}c\alpha_i & c\theta_{i+1}c\alpha_i & -s\alpha_i & -s\alpha_i d_{i+1} \\ s\theta_{i+1}s\alpha_i & c\theta_{i+1}s\alpha_{i+1} & c\alpha_i & c\alpha_i d_{i+1} \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

After solving each one of these individual sub-problems, the multiplication of all the transformations matrices will concatenate the results to obtain the final relation between the {n} reference frame corresponding to the n link or end-effector, and the {0} reference frame, located on base of the manipulator and corresponding to link 0:

$$ {}^{0}T_{n} = {}^{0}T_{1}\,{}^{1}T_{2}\,{}^{2}T_{3} \ldots {}^{n-1}T_{n} $$

Depending on the necessities and solution required for the enunciated problem, kinematics comprises two major branches or divisions: forward and inverse kinematics.

2.3.1 Forward kinematics

Also called direct kinematics, analyzes the problem by extracting the position and orientation of the tool (i.e., end-effector) from the known values of the joint angles associated to each one of the links of the manipulator. Being K the kinematics function for a given manipulator, forward kinematics is given by,

$$ {}^{0}T_{n} = K(q) $$

This equation uses K to relate the transformation matrix of a manipulator from its end-effector to its origin (i.e., reference frame n to reference frame 0), and the set of joint angles q that defines the positions and orientations for every link of the manipulator.

In other words, the independent variable or input is the set of joint angles, while the dependent variable or output will be the description of the position of the end-effector of the manipulator. The solution is given by the definition of the components of the robot in terms of the four parameters defined previously by the D-H model, and the transformations carried out according to the mathematical computation and concatenation of them. The final result is a pool of twelve equations, nine of them representing the rotation matrix ${}^{0}R_{j}$ and the remaining three indicating the position vector ${}^{0}P$. These equations are cast in the transformation matrix ${}^{0}T_{j}$, being j the reference frame of the tool. Craig gives two complete examples for obtaining forward kinematics solutions for industrial robots[9]. It is to be mentioned and can be concluded that forward kinematics solutions can be found for any manipulator in any configuration independent of the number of joints associated to it.

2.3.1 Inverse Kinematics

A more complex problem requires finding the joint angles associated to the links of the manipulator when a given position of the end effector is set.

$$q = K^{-1}\left({}^{0}T_{n}\right)$$

In this case the independent variable or input is the transformation matrix that gives the position and orientation of the tool while the dependent variable or output is the set of joint angles. Analyzing this definition, it can be concluded that not every single point selected in the space may evolve in a solution for a manipulator, since it has obvious restrictions; moreover, some points may bring infinite solutions, such points are called singularities. Therefore, the methods to solve inverse kinematics and determine whether a solution is possible or not, vary depending whether derivatives for positions and orientation angles (i.e., linear and angular velocities and accelerations) are known and considered as part of the statement of the problem.

The solution for an inverse kinematics problem is neither trivial nor linear and it can be solved either analytically (also called closed-form solutions) or numerically. According to Craig "all systems with revolute and prismatic joints having a total of six degrees of freedom in a single series chain is solvable"[10]. If a solution can be computed for robotic manipulators for all its joints, the manipulator is said to be solvable[11]. Currently, manipulators are design in such way that analytical solution can be reached; these are preferred to numerical solutions since the later require more time to compute and find a converging solution, in such way, analytical solution save calculation time. This work focuses in a 6-dof manipulator. In such way, and unless a generalization is established, calculation will be based on this assumption.

Models involving differentials of location (i.e., position and orientation) or, in other words linear and angular velocities, and differentials of joint angles and, how they relate to each other beyond static positioning, whether it is thru forward or inverse computations, require further analysis, every time that the elements of the manipulator are changing their position in time. To find a possible solution a whole computational theory has been developed based on another mathematical entity that must be studied deeply: the Jacobian matrix or Jacobian of the manipulator.

"The Jacobian matrix is the matrix of all first-order partial derivatives of a vector-valued function. Its importance lies in the fact that it represents the best linear approximation to a differentiable function near a given point. In this sense, the Jacobian is akin to a derivative of a multivariate function."[12]

The matrix that represents the jacobian is defined as

$$\begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}.$$

The solutions although applicable for manipulators with a simple design, still present some mathematical difficulties that remain unsolved[13].

Depending on the type of displacement of a link relative to its adjacent, whether the joint conformed by the union of them is revolute or prismatic; the parameter used as variable of the joint is either θ_i or d_i respectively. Thus, a general definition for a joint angle in a manipulator is given by[14]:

$$q_i = \overline{\sigma}_i \theta_i + \sigma_i d_i$$

with,

$\sigma_i = 0$ if i is revolute;

$\sigma_i = 1$ if i is prismatic;

$\bar{\sigma}_i = 1 - \sigma_i$

In the robotics theory, the jacobian is the mxn entity that relates the velocity of the end-effector to the joint velocities according to the equation:

$$[\dot{x}] = [J(q)] * [\dot{q}]$$

The jacobian matrix can be obtained by the partial derivatives of the location functions. If $[\dot{x}] = f(q)$, the jacobian is defined by,

$$J_{ij} = \frac{\partial}{\partial q_j} f_i(q),$$

for i=1,...m and j=1,...n, and J_{ij} is the (i,j) element of the jacobian matrix [J].

The velocity of the joint k, defined by $\dot{q}_k$ produces the velocity of the end effector $\dot{x}$, which is a combination of the linear velocity $v_{k,n}$ and angular velocity $\omega_{k,n}$, where n represents the reference frame at the tip of the tool. The general form for both velocities of the joint k and the reference frame n, is given by the equations

$$\vec{v}_{k,n} = \left[\sigma_k \vec{a}_k + \bar{\sigma}_k \left(\vec{a}_k * \vec{L}_{k,n} \right) \right] * \dot{q}_k$$

$$\vec{\omega}_{k,n} = \bar{\sigma}_k \vec{a}_k \dot{q}_k$$

,where a_k is the unit vector along z_k axis and $L_{k,n}$ is the vector that connects to frames (i.e., joints).

Two different cases must be considered according to the configuration of the manipulator:

k is a prismatic joint. In such case, $\sigma = 1$

$$\vec{v}_{k,n} = \sigma_k \vec{a}_k$$

$$\vec{\omega}_{k,n} = 0$$

k is a revolute joint. Thus, $\sigma = 0$

$$\vec{v}_{k,n} = \vec{a}_k \dot{q}_k * L_{k,n} = \left(\vec{a}_k * L_{k,n}\right) \cdot \dot{q}_k$$

$$\vec{\omega}_{k,n} = \vec{a}_k \dot{q}_k$$

Mathematical computation[15] takes to the expression of the kth column of the jacobian matrix as

$${}^{i}j_{n;k} = \begin{bmatrix} \sigma_k {}^{i}\vec{a}_k + \bar{\sigma}_k \left(-{}^{k}P_{n_y} {}^{i}\vec{s}_k + {}^{k}P_{n_x} {}^{i}\vec{n}_k\right) \\ \bar{\sigma}_k {}^{i}\vec{a}_k \end{bmatrix}$$

being ${}^{k}P_{nx}$ and ${}^{k}P_{ny}$ the x and y components of vector ${}^{k}\vec{P}_n$.

CHAPTER 3

ELEMENTS OF THE CONTROL SYSTEM

3.1 Model Schematics

A series of successive steps summarizes the implementation of the model of manipulator; this is the control block diagram:

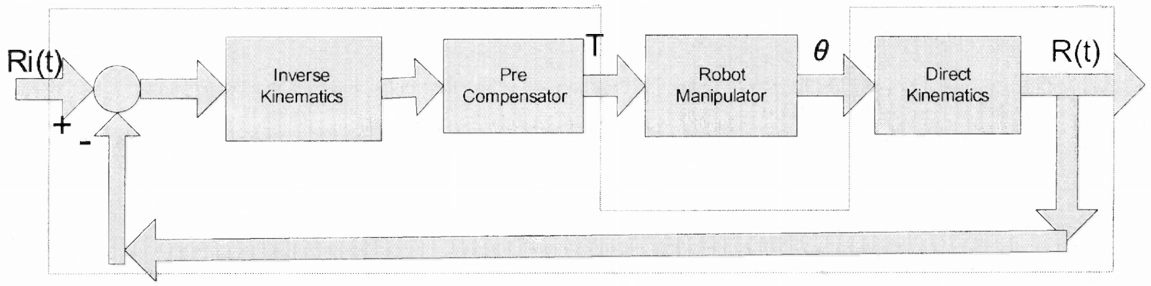


Figure 3.1. Conceptual Sketch of a robot manipulator control system. Source: Nakamura, Y., Hanafusa, Y. Inverse kinematics solutions with singularity robustness for robot manipulator control. Trans. of ASME, J. of Dynamic Systems, Measurements, and Control. Vol. 108. 1986, pp: 163-171.

The input is a position and orientation in space of the end-effector; the inverse kinematics block adjusts the angles from the previous block to the reference frame of the manipulator. The output of the manipulator is a set of angles that will be the input for the forward kinematics block, which will be delivered the actual position and orientation in space; this value returns to the input as negative feedback.

Being this a general sketch of a control system for any type of manipulator, some differences might be found with the control system being implemented in this work; first, it does not require a numerical determination of the position and orientation of the end-effector; therefore, the Forward Kinematics block is not required. Instead, the feedback is visual and delivered by the user, who is the one that determines if the final position corresponds to the one determined by the operation of the controllers.

3.2 Controllers

3.2.1 Immersion Probe and Personal Digitizer

The Immersion Probe^{TM} or Stylus Unit is a six-degree-of-freedom stylus-like tool with six revolute joints and three of them located in its end-effector; this is the device used to control the manipulator using position and orientation of the tip as input variables. This tool has one optical encoder located in each one of its joints; the information captured by the encoders is used as the input for the Personal Digitizer^{TM}. The Personal Digitizer^{TM} or Electronics Module has built-in software with a set of configuration commands that allow the programmer to set and/or get information from the probe such as the length of the links and the response to external stimuli.

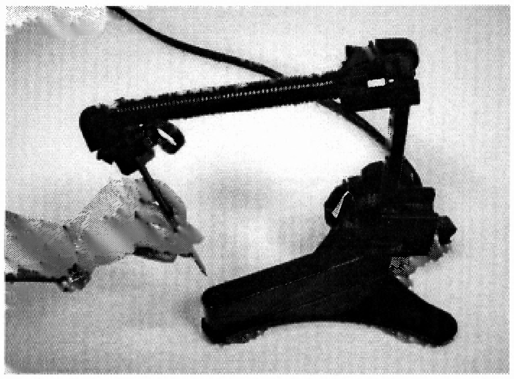


Figure 3.2. Immersion Probe ^{TM}. Developed by Immersion Corporation

A MC68HC11 processor with an 8-bit data bus commands the electronic module. The module uses built-in permanent and volatile memories to execute the code that contains the configuration commands. An internal A/D converter modifies analog signals to obtain the digital streams that are the final output of the module. Response to motion stimuli is obtain using the built-in real-time interrupt circuit. An internal clock synchronizes all the operations with a frequency of 1.8 MHz.

The encoders have different resolutions depending on the joint. The first two encoders (i.e., joints 0 and 1) have a resolution of 2048 pulses per revolution (PPR). The rest of the encoders have a resolution of 1024 PPR.

3.2.2 Spaceball® 5000

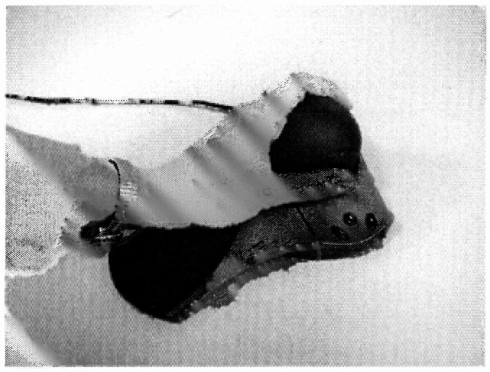


Figure 3.3. Spaceball® 5000. Developed by 3D Connexion.

The Spaceball® 5000 is recognized as a 6-dof motion controller that receives forces and torques as inputs (i.e., pressure applied directly to de device in six different directions) from optical sensors, to be interpreted either as positions or velocities, depending on the configuration. The sensitivity of the device can be adjusted, which is translated in the accuracy and speed of the movements of the model to be controlled, in this case, the manipulator.

3.3 Software Applications

3.3.1 Virtual Reality Viewer

A virtual reality viewer is a tool that allows the construction and visualization of 3D animated spaces. This tool requires a programming language, the Virtual Reality Modeling Language or VRML. More than a language

"VRML is simply a 3D interchange format. It defines most of the commonly used semantics found in today's 3D applications such as hierarchical transformations, light sources, viewpoints, geometry, animation, fog, material properties, and texture mapping".[16]

VRML uses hierarchical scene graphs to construct its virtual worlds containing entities called nodes. Nodes define the physical structure and location of objects, always related to other objects in the hierarchical organization (i.e., parents / children structure, a node can't contain itself). The information that characterizes each node is stored in fields; each node contains its particular fields of information. Nodes and fields that define objects in a scene are organized in files, which VRML code structure, containing functional parts, allows the definition of individual entities; these functional parts are: the header, the scene graph, the prototypes, and event routing[17]

V-Realm Builder is the graphic application used to build the 3D virtual model of the ARM Manipulator. It takes the hierarchical philosophy of VRML and brings a graphic-oriented environment to facilitate the design of 3D models.

3D models are built within a virtual world defined by geometric structures with different characteristics represented by its nodes. A virtual model is built and oriented in VRML using a Cartesian, right-handed, three-dimensional coordinate system, as seen in Figure 3.4.

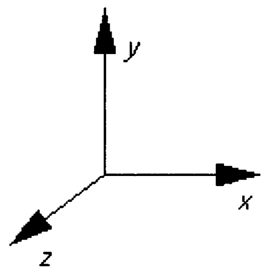


Figure 3.4. Cartesian, right-handed, three-dimensional coordinate system.

Each new element added to the virtual model is an actual node called a transform; this node groups its children, defines their coordinated system and contains a set of generic nodes that specify the parameters that characterizes new transform (i.e., each new transform added as a children of a precedent one will have a coordinated system in space relative to its parent).

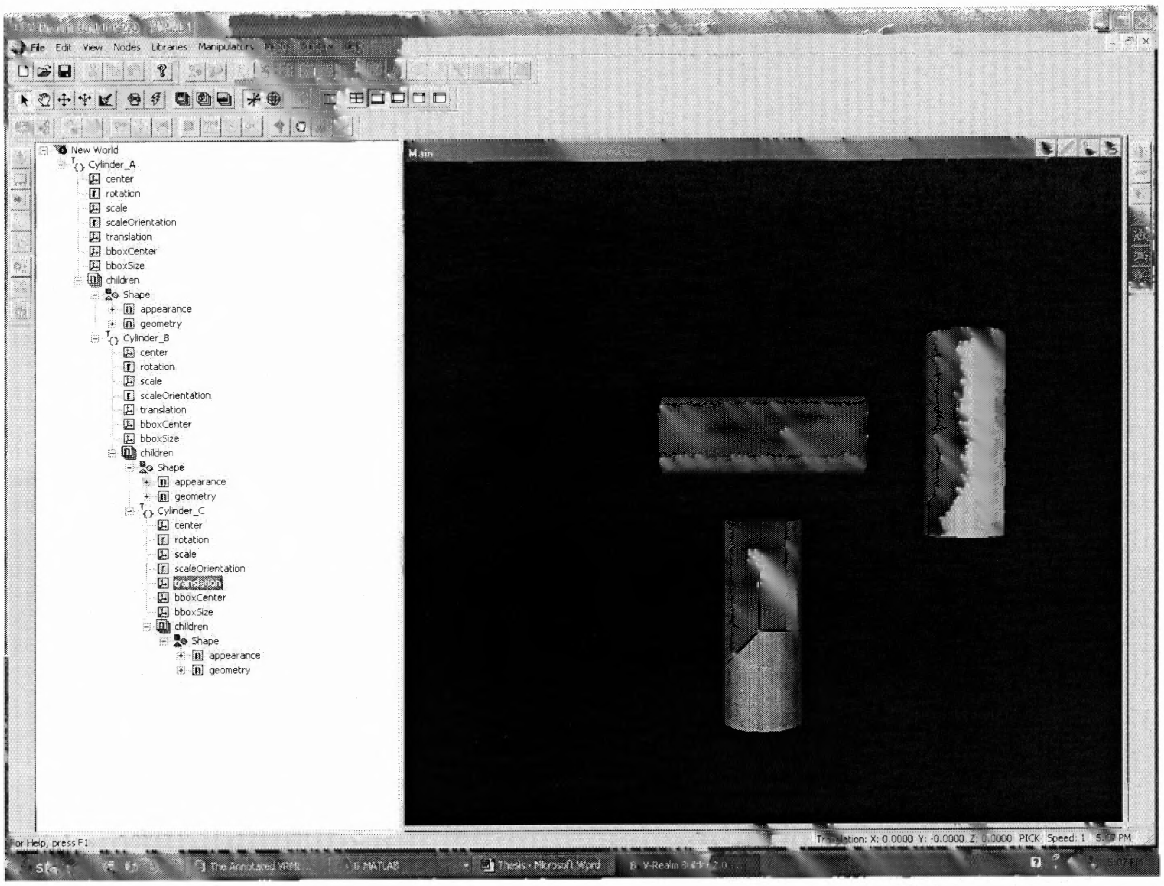


Figure 3.5. Typical screen of VRealm Builder^{TM}, the VRML editor used to make the model of the manipulator

In figure 3.5, cylinder A and cylinder B have coordinated frames rotated π/2 radians (90 degrees) one to the other. In the same way, cylinder A and cylinder B have coordinate frames rotated π radians one to the other.

3.3.2 Robotics Toolbox – Matlab

The robotics toolbox for Matlab is a development of Peter Corke who is the research director of the Autonomous Systems laboratory within the CSIRO ICT Centre in Australia. This toolbox is available to be downloaded from the World Wide Web for free with access to the source code written in Matlab[18]. It provides all the implementations for the developments of robotics-oriented calculations such as kinematics and dynamics for serial link manipulators.

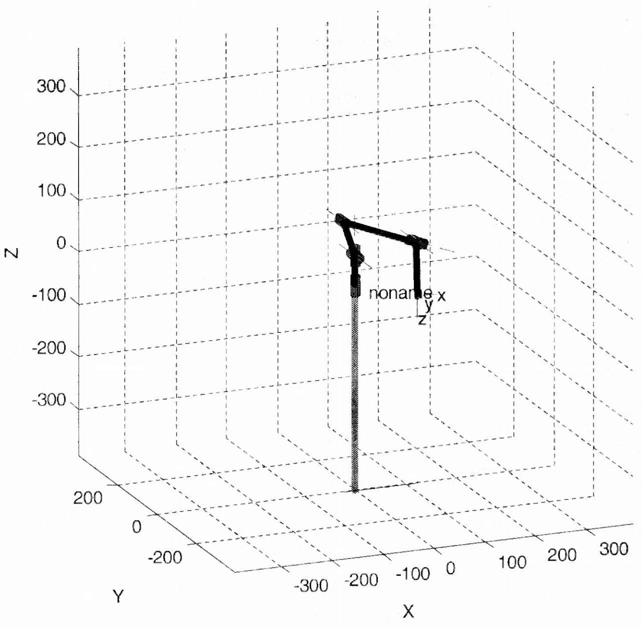


Figure 3.6. Schematic model of a robotic manipulator built in Matlab's Robotics Toolbox

The toolbox is based in the definition of a robot object that contains a series of links sequentially located. Links are defined according to DH model and depending on the requirements of the design, it can be implemented using either standard or modified model configuration. From the DH model, four basic parameters define a joint-link pair: link twist (α), link length (l), link offset (d), joint angle (θ).

3.3.3 Virtual Reality Toolbox – Matlab

The computation toolbox (i.e Robotics Toolbox) and the virtual reality viewer need an interface to be able to communicate. This is the Virtual Reality Toolbox contained in Matlab. This is the platform for the 3D scenes and the Simulink and Matlab animations. The toolbox gets the data from the input devices (i.e., the two controllers implemented in the system) and interprets the final result to deliver it to the output (i.e., the model of the ARM Manipulator).

3.3.4 Simulink

Simulink allows real-time simulations and, according to embedded mathematical manipulations developed in Matlab, Simulink interacts with both V-Realm Builder and with the Virtual Reality Toolbox and creates the platform for the execution of Matlab code. Functional blocks design allows modular configuration and the interaction with the Virtual Reality Toolbox makes the system fast enough to permit a real-time process. Simulink is used as a control tool only for the Spaceball, since the existent driver in the Virtual Reality Toolbox for the device was made in this platform.

CHAPTER 4

IMPLEMENTATION

4.1 Communication Protocol

4.1.1 Immersion Probe and Personal Digitizer

There are two steps in the communication process. The first one is from the controller to the electronic module of the controller (i.e., Personal Digitizer) and the second one from the electronic module to the CPU. The data acquisition port that communicates the probe unit with the electronic module uses a DB-37 connector to get the six two-word sets of signals from the six encoders of the stylus unit.[19] The number of pulses of encoder is translated into a binary number proportionally corresponding to the number of revolutions of each joint. The number of pulses per revolution varies depending on the joint of the controller.

The communication between the electronic module and the CPU is made via serial port, RS-232 protocol; there is bidirectional data transmission via a standard serial communications cable with DB9 connector.[20]

The communication with the host computer starts with an initialization sequence that starts when the electronic module is turned on; the stylus is assumed to be in its initial position (i.e., the stylus resting vertically on the base), in this position the angle registers are preloaded with default values for this position.

The second step is the identification of the host computer's baud rate; the PC sends the string "IMMC" (this can be done either by the ASCII characters or its numerical representation), and the module will try to receive and echo the string using different baud rates, where echoing the string means successful communication. After this, the communication has established and the Personal Digitizer is ready to receive instructions. This process is accomplished in Matlab using the algorithm below. The code instructs the CPU to carry out as many communication attempts as required until the controller echoes a string of characters. If the string is not the one required a warning requesting rebooting the controller is displayed.

```
d=0;
while d==0
    fprintf(C,'IMMC');
    d=C.BytesAvailable;
end
echo=fscanf(C,'%c',d);
a=strcmp(echo,'IMMC');
if a==0
    disp('Stylus needs to be rebooted')
    disp('Reboot Personal Digitizer and run the program again')
    return;
end
```

The first instruction to be sent is to signify the start of normal operation. The host sends the string "BEGIN", which will be echoed by the module with "PROBE", a null-terminated string, this is called the identifier string. Commands can be sent to the Personal Digitizer according to the instructions code and data collected from it. Attention must be paid to the format of the information received from the module; the length of the data packets varies depending on the instruction given. The implementation code is shown below.

```
fprintf(C,'BEGIN');
serialbreak(C,500);
d=C.BytesAvailable;
echo=fscanf(C,'%c',d);
```

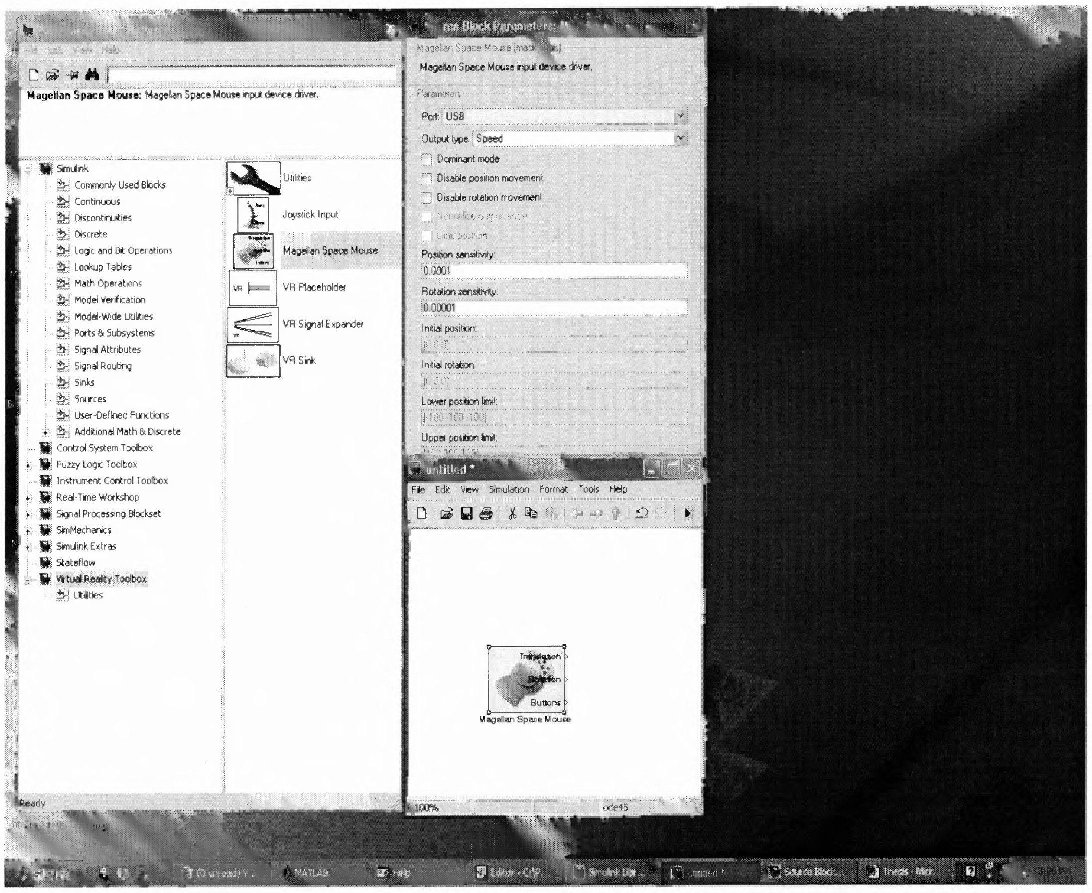
Each command is sent from the host in a single-byte format, where each bit represents a field that may or may not be activated, depending on its value.

When the session is to be finished, the "END" command (or the string "E") must be sent to stop communication and put the module in and baud rate identification state phase again. The angles must be initialized to their "home" or resting position.

This is the basic communication routine and operation process of the system.

4.1.2 Spaceball 5000

Universal Serial Bus USB communicates the spaceball with the host computer. The driver for the device is already installed in the Virtual Reality Toolbox and is the one in charge of processing the data obtained from the optical sensors of the device. The communication with the PC is established in Simulink and defined as part of the block parameters. The only parameter required for communications with Simulink is the port; data is always delivered in a unidirectional serial transmission protocol, there is no initialization protocol.



Magellan Space Mouse (mask) (link)

Magellan Space Mouse input device driver.

Parameters

Port: USB

Output type: Speed

Dominant mode

Disable position movement

Disable rotation movement

Normalize output angle

Limit position

Position sensitivity:
0.0001

Rotation sensitivity:
0.00001

Initial position:
[0 0 0]

Initial rotation:
[0 0 0]

Lower position limit:
[-100 -100 -100]

Upper position limit:

Figure 4.1. Control block of the Spaceball in the Virtual Reality Toolbox.

4.2 Design of the Virtual Model

V-Realm Builder is the editing tool used on the Matlab platform to create the model of the manipulator. The application is based on VRML node configuration and allows a graphical representation of its hierarchical distribution.

The manipulator's model starts from its base and keeps adding geometrical shapes to the structure in a parents / children relationship; these geometrical structures give shape and movement to the model according to the number of links and offsets and the rotation requirements for each joint.

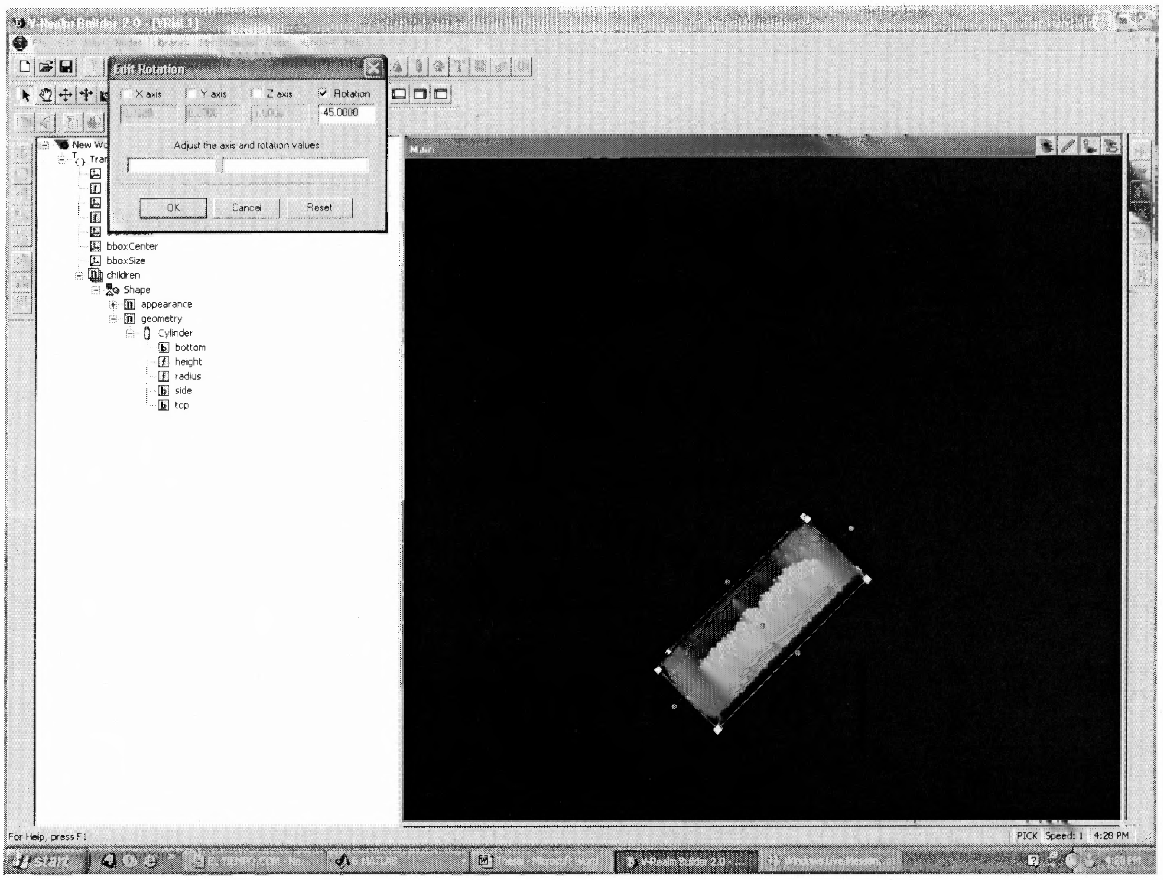


Figure 4.2. Example of rotation for a geometric shape in VRealm Builder.

A geometrical shape does not allow successive rotations or displacements; it means that if an object requires two rotations (e.g. one for location in space, the other for joint rotation), an additional geometric figure to perform the physical rotation in space must be added as child of the first object and parent of the second one.

The model was designed following the specifications of the actual manipulator ARM (Assistive Robot Manipulator), developed by Exact Dynamics from Netherlands.[21] The ARM is a 6 dof manipulator installed in wheelchairs, designed with the purpose of emulating human arm movements, to give more independence to people with conditions that have affected their upper limbs, such as muscular dystrophy, or brain stroke.

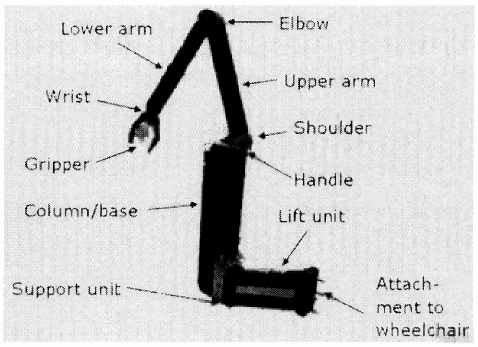


Figure 4.3. Drawing of the ARM Manipulator. Source: Exact Dynamics. http://www.exactdynamics.nl/

The dimensions and rotation of the links and joints remain, to create a model with the same characteristics as the actual manipulator.

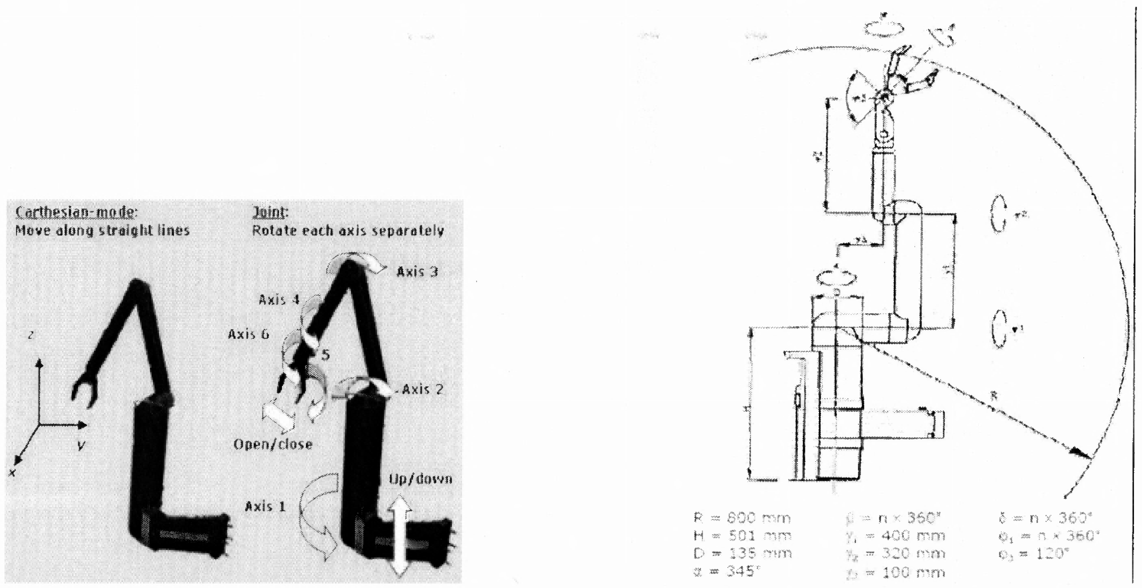


Figure 4.4. Drawing of the ARM Manipulator. Source: Exact Dynamics. http://www.exactdynamics.nl/

The model has 6 degrees of freedom and one button is active in each controller to control the grip.

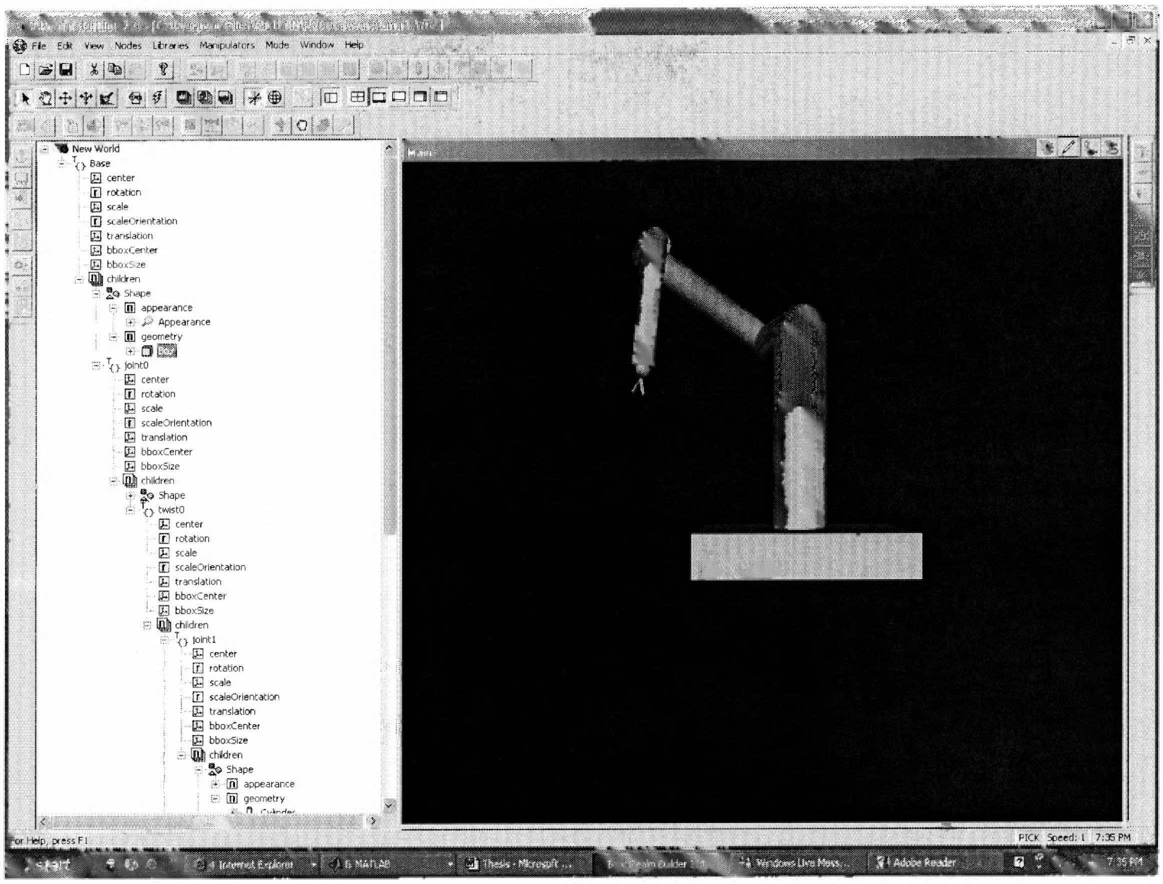


Figure 4.5. Model of the ARM Manipulator built in VRealm Builder, the VRML editor used for this work.

The design of the model required a hierarchical structure conformed by:

6 geometric shapes, one for each link (cylinders)

2 geometric shapes for the offsets (cylinders)

3 geometric shapes to change the orientation of the links (twists, spheres)

2 geometric shapes for the grip (cylinders)

1 geometric shape for the base (cube)

4.3 Control Models

4.3.1 Immersion Probe – Position control

The control system for the stylus or Immersion Probe is based on the interpretation of the encoders' data and its conversion in actual joint angles.

This is the simplest case; the optic encoders sense the position of each one of the joints and send this data to the electronic module, which converts the analog data in binary information subsequently transmitted to the host computer. In here the binary information is converted to angles and delivered to the model, which interprets the angles in its own coordinate system to represent the movements of the joints.

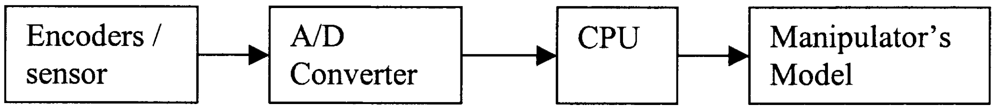


Figure 4.6. Control Model for the stylus controller. Scalable position approach.

4.3.2 Spaceball – Force Control

The Spaceball takes the forces and torques applied to it and converts them into speed; the final result is a vector with linear and angular velocities. This 6x1 vector is later associated to the end-effector, thus the velocities obtained are the velocities of the end-effector in the direction and/or orientation specified by the applied force/torque. The velocities from the tip of the manipulator are the input to the block of mathematical computation, which brings, as a final result, a vector containing the velocities of the joints of the manipulator (i.e., the rate of change in the joint angles). The integral of these velocities in time will result in a vector containing the joint angles in that particular moment that will feed the model of the manipulator.

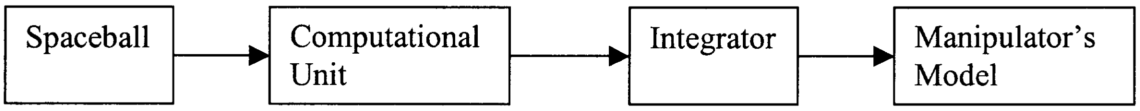


Figure 4.7. Control model for the Spaceball controller. Isometric approach.

This control system was developed in Simulink; it brought the advantage that a driver for the Spaceball already exists in the Virtual Reality Toolbox to be used with Simulink. Since the computation of the joint angles must be done using Matlab's Robotics Toolbox, Simulink had to call the functions at some point. To make this possible in real-time, the required Matlab functions are called from an embedded Matlab Function block; this block guarantees that the computation of the angles will be run alongside with each cycle of the simulation.

4.4 Computational Application

4.4.1 Immersion Probe –Scalable Control of Position

Several steps were considered when designing the application for the control of position at a scalable level. Once asynchronous communication is established with the serial port, the information from the encoders is read and transformed into correspondent joint angles, they are delivered to the model of the ARM Manipulator thru the Virtual Reality Toolbox. The control for the grip is also implemented following the same structure and forms part of the algorithm.

4.4.1.1 Computation of the angles from the stylus

A command that continuously monitors all the time-dependent variables is sent to the personal digitizer; when a changed beyond a defined threshold is detected in any one of these variables, a set of packets is sent to update this information.

```
fwrite(C,[79 0 15 163 0 0 0 0 0 0 0 0 0 0 63 0 63 0 31 0 31 0 31
0 15],'async');
%Bytes 14 - 25: (0.04369453125 (bytes 14 - 17)|0.087890625 (bytes
18 - 23)|0.17578125 (Bytes 24 - 25)) rad/pulse;
%Time in ms: bytes 2 and 3
```

The command is a 24-byte word; byte 1 is reserved for the instruction to be executed; bytes 2 and 3 are a 2-byte word designed as the time delay between packets in ms; byte 4 is the command to be executed when the threshold is surpassed; the last 12 bytes correspond to the individual thresholds assigned to each one of the encoders. By design, the number of pulses per revolution varies depending on the encoder, consequently, the number of radians per pulse.

```
echo=fread(C,2);
```

To empty the buffer, the echoed instructions are read; then, the stylus is ready to send information according to the parameters already set. To read the binary set and convert it into angles a switch command is used, which takes the information from the buffer in the port and assigns the values of each angle to a position in a vector.

```
angle=fread(C,15);
for i=1:6
    switch i
        case 1

            ang(i)=(angle(2*(i+1))*128+angle(2*(i+1)+1))*(2*pi/81
            92);
        case 2

            ang(i)=((angle(2*(i+1))*128+angle(2*(i+1)+1))*(2*pi/8
            192))-pi;
        case 3

            ang(i)=((angle(2*(i+1))*128+angle(2*(i+1)+1))*(2*pi/4
            096))-pi/2;
        case 4

            ang(i)=(angle(2*(i+1))*128+angle(2*(i+1)+1))*(2*pi/40
            96);
        case 5

            ang(i)=((angle(2*(i+1))*128+angle(2*(i+1)+1))*(2*pi/4
            096));
        otherwise

            ang(i)=(angle(2*(i+1))*128+angle(2*(i+1)+1))*(2*pi/20
            48);
    end
end
```

When the port is active and ready to read data, there is a timeout period after which a warning is generated in Matlab and the program keeps running. This behavior makes the program to break its continuity and eventually, it is aborted. To avoid this, the cycle of reading the data from the port was conditioned to the parameter *bytesavailable*. This is a property of the port and indicates the number of bytes currently available to be read from the input buffer. Its value is continuously updated as the input buffer is filled. Another property, this one bound to the variable containing the final joint angles is used to execute an action when the port times out. *Numel (var)* returns the number of elements in the variable *var*. In this case, when *numel*=0, the cycle of creating the angles vector is broken and the execution returns to the previous step of reading the port. A warning announcing the timeout period expiration is displayed but the program continues running cyclically.

4.4.1.2 Implementation of grip activation

A digital input of the electronic module is used to implement the activation of the grip. A pedal, which is supplied with the Personal Digitizer package, is the control that delivers the signal to open and close the grip. The algorithm forms part of the angles computation code by taking information from the same command. Once activation is detected, a function is called that updates the virtual model. The grip is conformed by two independent structures each one having a rotation assigned, opposite one to the other (i.e., they move between +/- pi/4 (open) and 0 (close) radians).

```
if numel(angle)==0
      break;
elseif angle(2)~=0
```

```
        gripang=0;
        vr_stylus(imstylus,q,gripang);
end
```

4.4.1.3 Call the VR model from Matlab

The model of the ARM manipulator is called using instructions from the Virtual Reality Toolbox. The *open* and *view* commands bring to model to execution. Then, the function *vr_stylus* associates the angles calculated from the stylus to the ARM.

```
imstylus = vrworld('arm1.wrl');
open(imstylus);
view(imstylus);
mynodes=get(imstylus,'Nodes');
vr_stylus(imstylus,q,gripang);


function vr_stylus (vrobj,ang,gripang)

vrobj.joint0.rotation=[0 1 0 ang(1)];
vrobj.joint1.rotation=[0 1 0 (ang(2)+pi/2)];
% 0 0 mean no rotation about x and y, 1 means rotation about z,
pi/4 is angle in rad.
vrobj.joint2.rotation=[0 1 0 (-(ang(3)+pi/2))];
vrobj.joint3.rotation=[0 1 0 (ang(4)+pi/2)];
vrobj.joint4.rotation=[0 0 1 (ang(5))];
vrobj.joint5.rotation=[0 1 0 ang(6)];

vrobj.grip0.rotation=[0 0 1 gripang];
vrobj.grip1.rotation=[0 0 1 -gripang];
vrdrawnow;

%Note, if joint has 2 dof, only last one called is moved.
%We need to create new joints so that all have 1 dof

vrdrawnow; %updates rendered image
```

The instruction that maps the angles requires specifying the direction of rotation, as well as adjusting the angles from the reference frame of the stylus to the reference frame of the VRealm Builder.

4.4.2 Spaceball - Control of Isometric Force

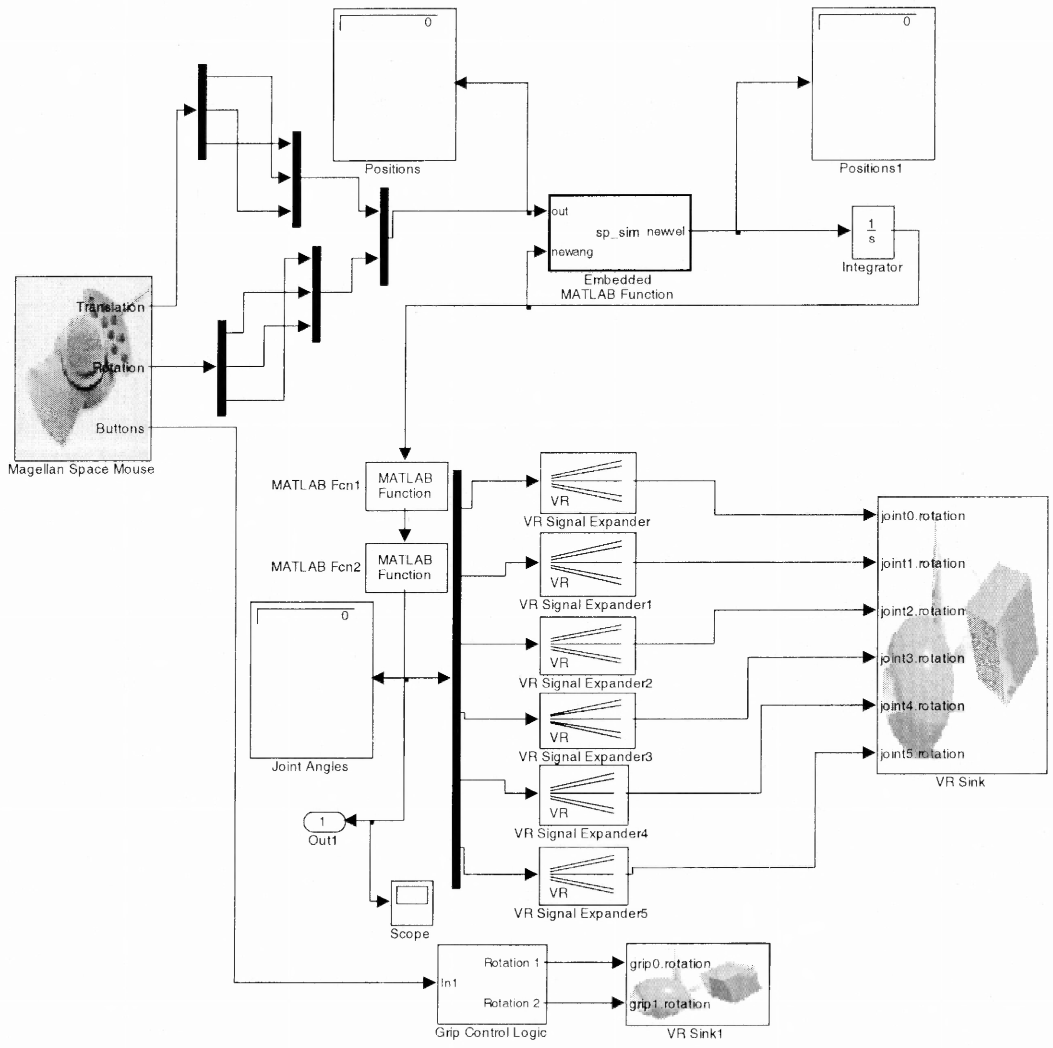


Figure 3.8. Schematic control system for the Spaceball controller, made in Simulink.

4.4.2.1 Inputs

The input is collected from the Spaceball in the form of linear and angular velocities. Spaceball. Depending on the necessities, Spaceball can isolate its types of outputs (i.e., inputs for the control system); dominant mode, disable of position movement or disable of rotation movement are parametric options that can be selected. An additional input taken from the buttons is used to control the grip of the manipulator.

4.4.2.2 Matlab embedded function

The driver for the spaceball was developed originally for Simulink. Thus, the control system was done in this application; since the robotics toolbox is developed in Matlab, an integration of both applications was necessary. Calling Simulink from Matlab would result in no-real time data acquisition; therefore, an embedded Matlab block was used in the Simulink control block. In this way the computation functions in Matlab are called every cycle of the simulation and the data are available on real time. This embedded Matlab function contains all the inverse kinematics necessary to give joint-angle velocities as an output.

Matlab's Robotics Toolbox contains predefined functions that calculate the mathematical operations required to obtain the required joint-angle velocities. To make the computations a model of the manipulator must be developed using the tools of the toolbox.

4.4.2.3 Design of the ARM-like manipulator in Robotics toolbox

The model of the ARM manipulator is based on the modified DH configuration[22]. It is scaled to match the dimensions of the ARM model built in VRealm Builder.

```
L{1}=link([0 0 0 50.1*4],'modified');
L{2}=link([-pi/2 0 0 -27*4],'modified');
L{3}=link([0 41.7*4 0 12.4],'modified');
```

```
L{4}=link([-pi/2 0 0 32*4],'modified');
L{5}=link([pi/2 0 0 0],'modified');
L{6}=link([-pi/2 0 0 0],'modified');
```

The variable L defines the object that contains the six links of the manipulator. The values in the vector correspond to twist angle, link length, joint angle and offset, respectively.

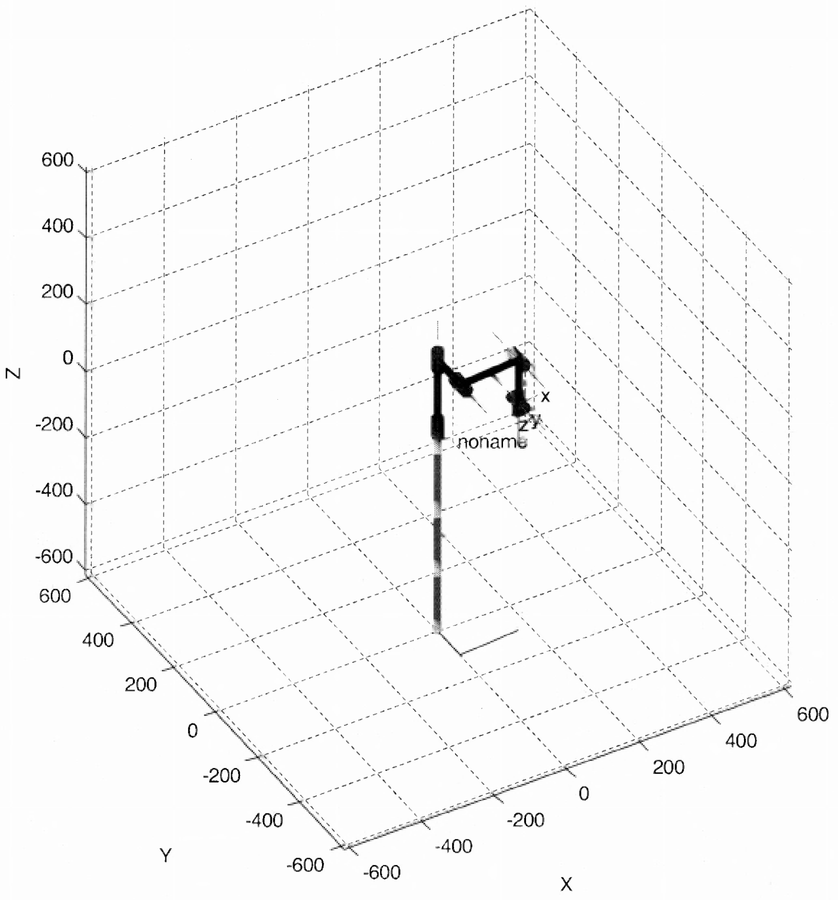


Figure 3.9. Model of the ARM Manipulator built in the Matlab's Robotics Toolbox.

This is the schematic representation of the manipulator. The object Link associated to the robot created allows the computation of the inverse kinematics.

4.4.2.4 Jacobian for robotics toolbox model

Inverse kinematics uses both the linear and angular velocities of the end-effector to calculate the joint angles. A mathematical entity, the Jacobian (see Chapter 2, numeral 2.3.2), is used for this purpose according to the equation

$$[\dot{x}] = [J(q)] * [\dot{q}]$$

and from here,

$$[\dot{q}] = [J(q)]^{-1} * [\dot{x}]$$

This equation can only be computed when *J(q)* is know to be square and non-singular; otherwise there is no solution for this equation because the inverse of the jacobian does not exist mathematically[23]. Therefore, this computation is developed using, instead of the regular inverse, a different type of matrix called pseudoinverse (also, Moore - Penrose pseudoinverse or generalized pseudoinverse), which is a generalization of inverse matrices[24]. The pseudoinverse matrix A^+ is the unique matrix which satisfies the following criteria:

1. $AA^+A = A;$
2. $A^+AA^+ = A$
3. $(AA^+)^* = AA^+$
4. $(A^+A)^* = A^+A$

This pseudoinverse is applied to the jacobian matrix, which guarantees a solution for the equation. However, this solution might not be the desired solution for the system, as it will be explained later on. The Robotics Toolbox has a built-in jacobian calculation function based on the formula explained before,

$$ {}^{i}j_{n;k} = \begin{bmatrix} \sigma_k\,{}^{i}\vec{a}_k + \bar{\sigma}_k\left(-{}^{k}P_{n_y}\,{}^{i}\vec{s}_k + {}^{k}P_{n_x}\,{}^{i}\vec{n}_k\right) \\ \bar{\sigma}_k\,{}^{i}\vec{a}_k \end{bmatrix} $$

From here, the jacobian ${}^{3}J_6$, gives the information necessary to obtain the angles of the joints for the wrist of the manipulator; hence, the first step to calculate the jacobian is to obtain the transformation matrices to complete the data for each column.

For example ${}^{0}T_1$ for the frame 1 related to frame 0 is,

```
{{cos[1],    -sin[1],     0, 0},
 {sin[1],    cos[1],      0, 0},
 {0,         0,           1, d1},
 {0,         0,           0, 1} }
```

Since the calculations of transformation matrices require matrices multiplications, the terms become extremely long and complex. For example, the first row of ${}^{1}T_6$ is (i.e., ${}^{1}T_6(1,1:4)$),

```
{cos[2]*(-(cos[6]*sin[3]*sin[5]) + cos[3]*(cos[4]*cos[5]*cos[6] -
sin[4]*sin[6])) -
sin[2]*(cos[3]*cos[6]*sin[5] + sin[3]*(cos[4]*cos[5]*cos[6] -
sin[4]*sin[6])),

cos[2]*(sin[3]*sin[5]*sin[6] + cos[3]*(-(cos[6]*sin[4]) -
cos[4]*cos[5]*sin[6])) -
sin[2]*(-(cos[3]*sin[5]*sin[6]) + sin[3]*(-(cos[6]*sin[4]) -
cos[4]*cos[5]*sin[6])),

cos[2]*(-(cos[5]*sin[3]) - cos[3]*cos[4]*sin[5]) -
sin[2]*(cos[3]*cos[5] - cos[4]*sin[3]*sin[5]), -

(d4*cos[3]*sin[2]) + cos[2]*(a2 - d4*sin[3])}
```

The Robotics Toolbox takes the values of the initial transformation matrix and calculates the jacobian by recursively concatenating the values of each term obtained[25].

```
U = L{j}( q(j) ) * U;
if L{j}.RP == 'R',
% revolute axis
      d = [ -U(1,1)*U(2,4)+U(2,1)*U(1,4)
            -U(1,2)*U(2,4)+U(2,2)*U(1,4)
            -U(1,3)*U(2,4)+U(2,3)*U(1,4)];
      delta = U(3,1:3)';  % nz oz az
end
```

4.4.2.5 Singularities

The inverse of the Jacobian presents innumerable problems when approaching singularities. According to Chang and Khatib, a singular configuration is a configuration at which the end-effector mobility, defined as the rank of the Jacobian matrix[26], locally decreases[27]. At a singular configuration, the end-effector locally loses the ability to move along or rotate about some direction in the Cartesian space. In this case the inverse of the jacobian does not exist and, therefore, there is no solution for the joint velocities. The singularities of a square jacobian matrix are defined as the zeros of the determinant of the jacobian, det(J)=0 or, in case of a redundant robot, the zeros of det($J*J^T$).

The generalized inverse or pseudoinverse is a matrix that can bring a solution for $\dot{\theta}$. The solution for the pseudoinverse provides a minimum norm solution[28]. In the same way, energy consumption can be related to the norm of joint velocities, therefore, solutions for the pseudoinverse are related to minimum instantaneous power consumption; in this sense an since getting near to a singularity means an increase in velocities, therefore, the pseudoinverse solutions, by providing minimum norm solutions, avoid these singularities[29]. Pseudoinverse, nevertheless, not always comes up with the desired results for the joint velocities due to stability problems. When in a singularity, the jacobian is no full row rank anymore (i.e., a direction of movement of the end-effector is not achievable). Curiously, when the manipulator is in an exact singularity, it will not try to move in an impossible direction and it will be "well-behaved". On the other side, if near to a singularity, small movements will bring large and unpredictable changes in joint angles. This phenomenon will signify uncontrolled and accelerated movements (e.g., tremors and 'jerk' movements), compromising the safety of the user.

For manipulators with a configuration as the ARM manipulator, called non-redundant manipulator (i.e., the number of degrees of freedom correspond to the number of joints), mobility and singularities are determined by the determinant of the Jacobian matrix.

4.4.2.6 Damping.

The problem of the type of discontinuities of the pseudoinverse solution at a singular configuration was approached using the *damped least-squares method.* In this case, instead of finding the minimum vector $\Delta\theta$ to solve the system (i.e., minimum norm solution), this method finds a $\Delta\theta$ that minimizes the expression

$$\left\|\dot{x} - \vec{J}\dot{q}\right\|^2 + \alpha^2 \left\|\dot{q}\right\|^2,$$

where α is a constant and is called the damping factor or scale factor.

The normal (i.e., least-squares solution) equivalent is represented by the system[30]

$$\begin{bmatrix} \vec{J} \\ \alpha I_n \end{bmatrix} \dot{q} = \begin{bmatrix} \dot{x} \\ 0_{nx1} \end{bmatrix}$$

This is equivalently rewritten as[31,32,33],

$$\dot{q}_a = J^T \left[J * J^T + \alpha^2 I_m \right]^{-1} \dot{x}$$

The damping factor α limits the norm of the solution. The objective of this damping factor, which is obtaining a feasible solution in the vicinities of singular points, is contrary to the requirement of reducing the error of inverse kinematics. A trade-off between an exact and a feasible solution must be found. An automatic adjustment of the damping factor tries to deal with both requisites and not to substantially affect the behavior of the manipulator.

Yoshikawa defined manipulability in a robot as its capacity to generate velocity and it is expressed in terms of the determinant of the jacobian[34,35]:

$$w(\dot{q}) = \sqrt{\det\left(J(\dot{q}) * J^T(\dot{q})\right)}$$

Manipulability can be interpreted as distance from singular points. To find singularities the determinants of both the jacobian and its transpose are tested in real-time using a function created in the robotics toolbox of Matlab and modified in-house.

```
w=[w;sqrt(det(jac*jac'))];
```

To adjust the damping factor, Yoshikawa proposes to adjust it as a function of manipulability as follows

$$\alpha = \begin{cases} \alpha_0 \left(1 - \frac{w}{w_0}\right)^2 & if\, \omega \prec w_0 \\ 0 & if\, \omega \geq w_0 \end{cases}$$

This function was used in the implementation of the inverse kinematics and the values were determined by trial an error using visual feedback:

```
im=eye(6);
r_arm=robot_def(init_ang);
dampf0=1;
m0=10;
jac=jacob0(r_arm,init_ang);
m=maniplty(r_arm,init_ang,jac);
if m<m0
    dampf=dampf0*(1-m/m0)^2;
else
    dampf=0.001;
end
inv_jac=inv(jac'*jac+dampf.*im)*jac';
```

4.4.2.7 Joint angles from joint velocities

The output value from the previous block is an angle velocity vector, according to the equation for the inverse kinematics solution. An integrator block outputs the joint angles, all the time that

$$\dot{\theta} = \frac{d\theta}{dt}$$

Since an initial value is required, this is taken as a 0 value vector (i.e., no change in position). The result of the integration is used as feedback for the computation block; this is the initial angle for every calculation of the velocities vector for the new iteration

4.4.2.8 Adjust of the angles

The angles obtained correspond to the computation in the Robotics toolbox, which has a different coordinate system than the Virtual Reality toolbox (i.e., although they both obey to the right-hand rule, the coordinate systems are rotated one with respect to the other). To compensate, two blocks were added, which make the angles from both toolboxes equivalent.

4.4.2.9 Outputs

Finally, the angles vector must be parameterized according to the VR Builder variables configuration. Angle rotations require four fields: the first three defined the axis of the rotation, the last one, the value of the rotation. Expander blocks create four-term vectors for each one of the positions of the angle vector and put this value in the last position of them. Each vector is assigned to the rotation transform corresponding to every joint in the manipulator model.

CHAPTER 5

RESULTS

5.1 Scalable Position – Immersion Probe

The control system was implemented in the Neuromuscular Engineering Laboratory of the Biomedical Department at NJIT.

The first observation to be made, when running the system, is that to establish a successful serial communication between the controller and the computer, the buffer of the port of the electronic module must be empty, otherwise, it will not echo the string that confirms the establishment of communication. The communication routine developed provides a security by the implementation of a conditional routine that requests the rebooting of the electronic module when a wrong echo is received. In the same order of ideas, another routine was implemented to guarantee that there is a port available and Matlab has priority to use it. These initialization routines avoid generation of errors and abortions of the program once it is run.

The serial port generates and interruption if data is not received within a pre-established period of time. This data is basically angles read from the electronic module of the controller. In other words, if the stylus does not execute a movement before the timeout period, an error is generated and the program aborts. To avoid this error and the abortion of the program, the timeout interruption is redirected to the beginning of the reading cycle. A warning is generated noticing the user that there is no movement detected but the program keeps running continuously.

The model precisely replicates the movements of the stylus controller. This required the necessary adjustments since each one of the modules (i.e., the controller and the modeling software) has its own coordinate system.

Although the configurations are slightly different, the visual feedback obtained from the model in the monitor of the computer, and the fact that the joints distribution is equivalent for both, the controller and the model, give the sensation of being moving directly the model in the screen and diminishes these differences.

The grip activation was implemented as part of the routine to read the angles. The command used in the Personal Digitizer allows detection of activation of analog inputs. This input is read from the buffer and lets it empty for more data to come allowing, therefore, the action of the grip and the movement of the manipulator simultaneously.

5.2 Isometric Forces – Spaceball 5000

The driver for the Spaceball 5000 was developed in Simulink, and a driver for Matlab will not be available until the second semester of 2007. The computation toolbox for robotics implementations is entirely developed in Matlab. Therefore, the control system requires implementation using both applications running together.. Thus, two different approaches were studied:

1. Use Matlab as platform and from there call Simulink to carry out the interface with the model. When this option was examined, it was unlikely that Simulink could obtain data from the Spaceball controller and delivered in real-time to Matlab. The way that Simulink runs its simulations and interacts with Matlab only allows for data to be delivered as a variable to the workspace in Matlab after the simulation has been stopped. This option was not considered since did not comply with the real-time requirements of the project.
2. The second option available was using Simulink as a platform and calling the computation Matlab routines from there. To avoid the same situation of no real-time data, exposed before, it was required to *"insert"* Matlab in Simulink using and Embedded Matlab Function Block, allowing Matlab to run as a part of Simulink. This block has operational limitations restricted to some Matlab functions. For this reason, the block was used to get the data from the Spaceball controller and call the computation routines in Matlab, then delivering its output to the next block of the simulation. The simulation runs uninterruptedly and uses both applications successfully.

The communication of the Spaceball controller is made via the USB port, which features facilitates the communication and does not have the restrictions due to timing out presented in the serial connection for the Personal Digitizer controller.

Unlike the previous control approaching, this controller does not use displacements as inputs. The operation, thus, at first sight must seem slightly more complex (i.e., there is not one-to-one match between the model and the controller) and might require some training for the user to be able to use it comfortably. The most important feature to be considered is that the user must understand that, by moving the controller in any of its three directions or three rotations, what the user is commanding is the end-effector or the grip of the manipulator; the links and joints will accommodate to take the tip to the desired position.

Rotation and translation are possible to be isolated one from the other as control features, as is the sensitivity of the controller. These are recommendable characteristics to be used when training the user to use the system. Same comment applies for a specific feature that isolates the dominant direction of movement from the others; this option makes the movements softer and smoother.

One particular case was noticed and is an interesting case point for future works: When the manipulator reaches its maximum extension and the user maintains a steady force applied to the controller, the model does not stop but continues with a rotation of the base joint sweeping the workspace, with the other links fully extended. This might be due to fact that the inverse kinematics functions keep finding minimum solutions for the system.

As mentioned before, a trade-off between an exact and a feasible solution must be found. Near a singularity the system reduces its speed perceptibly due to the damping effects; otherwise no solution would be possible and the system would be totally unstable. Despite the fact that the damping factor was implemented by trial and error the reduction of the speed does not compromise the total operation of the system and is likely that the user might be easily got used to these changes.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

Stylus (scalable position) and Spaceball (isometric method) were successfully adapted to be used as controllers for a 6-dof-manipulator model. Their outputs were used as an accuracy input for the control systems implemented.

The same configuration of manipulators with different workspace can be matched in a one-to-one match development. Since visual feedback is delivered by the manipulator's model, a learning process makes no necessary to reach a precise point in the control space to be modeled in the virtual world. Inverse kinematics eventually would allow the goal of reaching the same point in both workspaces (i.e., the control and the manipulator); it can be applied by defining the restrictions in the task space (i.e., the points in space defined by the control to be reached by the model in its own workspace). In this moment, with the available tools, this process is too costly in time and accuracy; a function to calculate inverse kinematics using the Robotics Toolbox was developed, but upon implementation, it was observed the for most of the points reached by the Immersion Probe and replicated to the ARM model, the solution did not converge and the resulting movements were never accurate neither reliable.

Maciejewski and Klein[36] use a property of pseudoinverse and least square methods, which projects the resultant matrix into the workspace of the Jacobian, what they called the *null workspace*. This nullspace method can be used as a tool to avoid obstacles. Nevertheless, the workspace has to be defined and the singularities identified; Botturi et al[37] developed a method to determine geometrically the workspace of a manipulator

Kinematics theory was applied to the selected configurations and real-time control was achieved with the expected results.

Future evaluation of controllers and training for potential users to determine their reliability is a necessary step prior to the implementation in the actual manipulator (ARM).

This implementation opens the possibility of using the designed model as an interactive tool, which could be used in learning and rehabilitation processes for kinds we disabilities.

The experience in the implementation brought knowledge and tools necessary to avoid possible errors in the implementation of the controllers in the ARM manipulator. Before implementing, the obstacle avoidance technique must be clearly defined. Securities must be provided before putting the ARM into movement.

The definition of the workspace and the determination of the singularities for this configuration may lead to an improvement in the manipulator's behavior, by selecting more appropriate damping factors. Right now this was determined by trial and error.

REFERENCES

[1] Simpson, D., Lamb, D. A System of Powered Prostheses for Sever Bilateral Upper Limb Deficiency. The Journal of Bone and Joint Surgery. Vol. 47B. No. 3. Aug. 1965. pp.: 442 – 447.

[2] Rosier, J.C., Van Woerden, J.A.,et al.Rehabilitation Robotics: the MANUS Concept. Advanced Robotics, 1991. Robots in Unstructured Environments, 91 ICAR., Fifth International Conference on. 19-22 Jun 1991. Pp: 893-898 vol.1

[3] Khalil, W & Dombre, E. Modeling, Identification & Control of Robots. Kogan Page Science. London, 2004.

[4] Ibid.

[5] Craig, J. Introduction to Robotics. Mechanics and Control. 3rd Ed. Pearson Prentice Hall. New Jersey, 2005.

[6] Ibid.

[7] Ibid.

[8] Denavit, J. and Hartenberg, R.S. A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices. Journal of Applied Mechanics, pp. 215-221, June 1955.

[9] Craig, J. Introduction to Robotics. Mechanics and Control. 3rd Ed. Pearson Prentice Hall. New Jersey, 2005.

[10] Ibid.

[11] Khalil, W & Dombre, E. Modeling, Identification & Control of Robots. Kogan Page Science. London, 2004.

[12] Wikipedia, the Free Encyclopedia. http://en.wikipedia.org/wiki/Jacobian. Dec. 2006.

[13] Gorla, B. ,and Renaud, M. , *Models of Robot Manipulators*, in Geometry and Robotics, pp. 306-335, Lecture Notes in Computer Science No. 391, eds. Boissonnat, J. -D., and Laumond, J. -P. , Proceedings of Workshop, Toulouse, May 1988.

[14] Khalil, W & Dombre, E. Modeling, Identification & Control of Robots. Kogan Page Science. London, 2004.

---

[15] Ibid.

[16] Carey, R., Bell, G. The Annotated VRML97 Reference Manual. http://www.cs.vu.nl/~eliens/documents/vrml/reference/BOOK.HTM Nov. 2006. Copyright © 1997 by Rikk Carey and Gavin Bell

[17] Ibid.

[18] Corke, P. I. A Robotics Toolbox for Matlab (Release 7). IEEE Robotics and Automation Magazine. Vol. 3. Number 1. Pages 24-32. Mar. 1996.

[19] Immersion Probe™ and Personal Digitizer™. Programmer's Technical Reference Manual: Immersion Probe and Personal Digitizer. Revision 2.0a. Immersion Corporation, Santa Clara, CA.

[20] Ibid.

[21] ARM: Assistive Robot Manipulator. Exact Dynamics. http://www.exactdynamics.nl/. Netherlands. Nov 2006.

[22] Denavit, J. and Hartenberg, R.S. A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices. Journal of Applied Mechanics, pp. 215-221, June 1955.

[23] Klein, C., Huang. CH. Review of Pseudoinverse Control for Use with Kinematically Redundant Manipulators. IEEE: Transactions on Systems, Man and Cybernetics. Vol. SMC-13, No. 3, Mar-Apr, 1983. pp. 245-250.

[24] Ben-Israel, A., Greville, T. Generalized Inverses. ISBN 0-387-00293-6, Springer-Verlag (2003)

[25] For the purposes of this document this function is partially shown and edited for better understanding.

[26] The rank of a matrix is the maximum number of independent rows (or, the maximum number of independent columns). A square matrix $A_{n\times n}$ is non-singular only if its rank is equal to n.

[27] Chang, KS., Khatib, O. Manipulator Control at Kinematic Singularities: A Dynamically Consistent Strategy. Robotics Laboratory Computer Science Department Stanford University Stanford, CA IEEE, 1995.

[28] Klein, C., Huang. CH. Review of Pseudoinverse Control for Use with Kinematically Redundant Manipulators. IEEE: Transactions on Systems, Man and Cybernetics. Vol. SMC-13, No. 3, Mar-Apr, 1983. pp. 245-250.

[29] Whitney, DE. The Mathematics of Coordinated Control of Prosthetic Arms and Manipulators. American Society of Mechanical Engineers, Winter Annual Meeting, New York, N.Y ; United States; 26-30 Nov. 1972. 7 pp. 1972

[30] Buss, S. Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares Methods. Department of Mathematics. University of California, San Diego, La Jolla, CA. April, 2004. Unpublished survey.

[31] Khalil, W & Dombre, E. Modeling, Identification & Control of Robots. Kogan Page Science. London, 2004

[32] Buss, S. Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares Methods. Department of Mathematics. University of California, San Diego, La Jolla, CA. April, 2004. Unpublished survey.

[33] Nakamura, Y., Hanafusa, Y. Inverse Kinematics Solutions with Singularity Robustness for Robot Manipulator Control. Trans. of ASME, J. of Dynamic Systems, Measurements, and Control. Vol. 108. 1986, pp: 163-171.

[34] Khalil, W & Dombre, E. Modeling, Identification & Control of Robots. Kogan Page Science. London, 2004

[35] Yoshikawa, T., Manipulability of Robotic Mechanisms. Prepr. 2nd International Symposium of Robotics Research. Tokio, Japan. 1984. pp: 91-98.

[36] Maciejewski, A., Klein, C. Obstacle Avoidance for Kinematically Redundant Manipulators in Dynamically Varying Environments. International Journal of Robotic Research, 4 (1985), pp. 109-117.

[37] Botturi, D., Fiorini, P., Martelli, S. A Geometric Method for Robot Workspace Computation. Proceedings of ICAR03 - The 11th International Conference on Advanced Robotics (2003).