2009

Task oriented simulation and control of a wheelchair mounted robotic arm

Fabian Farelo
University of South Florida

Task Oriented Simulation And Control Of A Wheelchair Mounted Robotic Arm

by

Fabian Farelo

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Biomedical Engineering
Department of Chemical and Biomedical Engineering
College of Engineering
University of South Florida

Major Professor: Rajiv Dubey, Ph.D.
Redwan Alqasemi, Ph.D.
Kyle Reed, Ph.D.

Date of Approval:
November 5, 2009

Keywords: rehabilitation robotics, dual-trajectory, mobile robot, manipulator, redundancy, adl

Acknowledgments

I would like to thank God for giving me the strength, patience and love for what I do. This allowed me to carry out this important part of my life. I would like to thank my parents Guzman and Ony for giving me the opportunity to come to the United States to pursue my undergraduate studies. Without their love and sacrifice, this would not have been possible. Thanks to my sister Natalia for making me laugh whenever we talked.

I am also very thankful to Dr. Rajiv Dubey for giving me the opportunity to pursue my graduate studies as one of his graduate students. I would like to thank him for his support and encouragement throughout these past 2 years.

I would like to thank Dr. Redwan Alqasemi and Dr. Kathryn De Laurentis for their invaluable support and guidance during my Master's program. I'm grateful to Dr. Kyle Reed for accepting to be in my thesis committee at such a short notice; thanks for your interest in my work and your eagerness to help.

I am very grateful to my lab partners. I would like to thank them for their help and companionship inside and outside the university. Thanks to Ana for believing in me.

Lastly, I would like to thank my uncles in Miami Rita and Jaime for their unconditional support throughout my stay in this country. Thanks to all of my friends in Tampa and Barranquilla, my hometown, for supporting me all the time!

Table of Contents

List of Tables

List of Figures

Task Oriented Simulation and Control of a Wheelchair Mounted Robotic Arm

Fabian Farelo

ABSTRACT

The main objective of my research is to improve the control structure for the new Wheelchair Mounted Robotic Arm (WMRA) to include new algorithms for optimized task execution; that is, making the WMRA a modular task oriented mobile manipulator. The main criterion to be optimized is the fashion in which the wheelchair approaches a final target as well as the starting and final orientation of the wheelchair. This is a novel approach in non-holonomic wheeled manipulators that will help in autonomously executing complex activities of daily living (ADL) tasks.

The WMRA is a 9 degree of freedom system, which provides 3 degrees of kinematic redundancy. A single control structure is used to control the WMRA system, which gives much more flexibility to the system. The combination of mobility and manipulation expands the workspace that a mobile base attains to a manipulator. This approach opens a broad field of applications: from maintenance and storage to rehabilitation robotics. This structure is based on optimization algorithms that can resolve redundancy based on several subtasks: maximizing the manipulability measure,

minimizing the joint velocities (hence minimizing the energy), and avoiding joint limits. This work utilizes redundancy to control 2 separate trajectories, a primary trajectory for the end-effector and an optimized secondary trajectory for the wheelchair. Even though this work presents results and implementation in the WMRA system, this approach offers expandability to many wheeled base mobile manipulators in different types of applications.

The WMRA usage was simulated in a virtual environment, by developing a test setting for sensors and task performance. The different trajectories and tasks can be shown in a virtual world created not only for illustration purposes, but to provide training to the users once the system is ready for use.

Chapter 1: Introduction

1.1 Motivation

According to the latest available data from the US Census Bureau [1], about 54.4 million Americans had some level of disability, 34.9 million of them had a severe disability. About 11 million Americans older than 6 years of age needed personal assistance with one or more activities of daily living (ADL). This work focuses on people with limited upper and lower extremity mobility due to spinal cord injury or dysfunction, or genetic predispositions. Robotic aides used in these applications vary from advanced limb orthosis to robotic arms [2].

A wheelchair mounted robotic arm provides mobility and manipulation capabilities enhancement to these persons. The workspace of the system allows the reach of objects that were otherwise impossible to reach. On the other hand, it also provides independence from external human aide, since the arm is mounted on the wheelchair and powered by its batteries. There are 2 commercially available WMRAs, the Manus from Exact dynamics in the Netherlands and the Raptor from Applied Resources in the US. Two prototypes of WMRAs have been designed and developed at the University of South Florida, and this work intends to build upon the performance of the system to overcome the limited commercial success that previous attempts have had, such as low payload and

difficulty to be maneuvered by the final user. This work intends to transform the WMRA system into a task oriented mobile manipulator with the objective of aiding persons with disability to successfully perform activities of daily living. Several methods for optimization can still be used as constrains for redundancy resolution in the WMRA. The absence of these constrains causes an undesired completion of the tasks, and requires a higher input level from the final user. The WMRA control algorithm combines mobility and manipulation for task oriented performance; however, each task is different and requires a different subroutine sequence in terms of trajectory generation for both the end- effector and the wheelchair.

1.2 Thesis Objectives

The main objective of my research is to improve the control structure and performance of the WMRA to include complete sequences of tasks that utilize redundancy to optimize the wheelchair and arm motion for autonomous task execution. The main criterion to be optimized is the fashion in which the wheelchair approaches a final target as well as the starting and final orientation of the wheelchair. This is a novel approach in non-holonomic wheeled manipulators that will increase the ease of the task execution for the final user since the system will orient itself depending on the desired task. The WMRA usage will be simulated in a virtual environment by developing a test setting for sensors and task performance. The different trajectories for each task can be shown in a virtual world created not only for illustration purposes, but to provide training

to the user once the system is ready to be used. The objectives are summarized as follows:

- ✓ Develop new criterion functions for the optimization of the translation and orientation of the wheelchair and the end effector in a specific task
- ✓ Develop a refined test bed for the WMRA system for experimentation in task performance, and trajectory generation.
- ✓ Develop program modules for several ADl tasks, creating subroutines for each task to autonomously execute.
- ✓ Create a virtual reality environment to test the algorithm and to train the user on the WMRA use while collecting data for further development and improvements.

1.3 Thesis Outline

This thesis will provide a literature review and present the state of the art on the important subjects regarding this work in chapter 2. Chapter 3 presents the trajectory generation problems. The reader is given an overview of the procedures implemented in this work. Chapter 4 presents the Virtual Reality environment developed for the simulation of the system. Chapter 5 presents the detailed procedure followed to achieve a dual-trajectory control in the WMRA for performing activities of daily living. Chapter 6 presents the results in Virtual Reality simulation and Matlab graphics following the main variables of the system. A discussion of these results is also addressed in this chapter. The final chapter briefly goes over the conclusions of this thesis and proposes future

work that can be completed later. A list of references and appendices are presented at the end with the code implemented in this thesis.

Chapter 2: Background

In this chapter, the state of the art in the research area is presented to the reader. This includes research in rehabilitation robotics, mobile manipulators, redundant robots, virtual reality environments and previous work developed in wheelchair mounted robotic arms.

2.1 Wheelchair Mounted Robotic Arms

Several designs of workstation-based robotic arm systems were developed over the years, such as Handy-1 [3], RAIL project [4], ProVAR [5] and the design conducted by Gunnar Bolmsjo, et al. [6]. WMRA combines the idea of a workstation and a mobile-base robot to mount a manipulator arm onto a power wheelchair. The most important design consideration of where to mount a robotic arm in a power wheelchair is the safety of the operator [7]. WMRA can be mounted in the front, side or rear of the wheelchair [8].

2.1.1 Commercially Available Prototypes

The two commercially-available commercial WMRAs utilize side mounting on a power wheelchair. These two commercial arms are the Manus, manufactured by Exact Dynamics; and the Raptor, manufactured by Applied Resources.

The Manus manipulator arm can be programmed in a manner comparable to industrial robotic manipulators. It has been under development since the mid 1980s and it entered into production in the early 1990s [9]. It is a 6 DOF arm, with servomotors all housed in a cylindrical base as shown in figure 1. Another production WMRA is the Raptor [10], which mounts to the right side of the wheelchair. This manipulator has four degrees of freedom plus a planar gripper as shown in figure 1. The user directly controls the arm joints with either a joystick or a 10-button controller. Typically, the joystick that controls the manipulator arm is located on the armrest opposite to the input device that controls the steering of the power wheelchair. Because the Raptor does not have encoders, the manipulator cannot be controlled in Cartesian coordinates. This compromise was done to minimize the overall system cost.

Figure 1 Raptor Arm

Figure 2 Manus Arm

2.1.2 USF WMRA First Prototype

Previous work has shown the analysis of commercial WMRAs and the development of a wheelchair-mounted robotic arm (WMRA-I) system with combined mobility and manipulation control [11,12,13]. The WMRA-I is comprised of a seven-degree-of-freedom robot arm, a gripper, and a power wheelchair as shown in figure 3. That system was designed to use Matlab to control the arm and the chair motion with a single graphical user interface (GUI) which can be used to control the end effector in Cartesian space. User interfaces include a touch screen, a spaceball with 3-D input capabilities and a Brain Computer Interface (BCI) that uses the stimulated P-300 signal.

The arm carries seven revolute joints and a gripper designed for ADLs. The gripper [14] is powered by a Faulhaber coreless DC servomotor that is compact and capable of producing 6N of grasping force at the gripper paddles. In order for the WMRA to have more commercial success, the weight must be reduced and the payload needs to be increased. Reducing the overall weight of the robot arm that is attached to the power wheelchair will reduce the power consumption, allowing longer system usage before the batteries need to be recharged. A lighter weight WMRA will also be less restrictive on the allowable user weight because the WMRA is an aftermarket modification and power wheelchairs are rated for a maximum weight capacity by the manufacturer.

Figure 3 WMRA-I

2.1.3 Composite Materials in Robotic Arms

Industrial robotic arm companies have begun to use composite materials, such as carbon fiber, as major structural components to reduce weight while keeping the necessary structural strength [15], but they have not been widely used in the field of rehabilitation robotics, specifically for WMRAs. Utilizing these composites in the construction of a WMRA can help reduce the weight of the overall design.

The DLR research group has been working on producing the lightweight robot (LWR) arm for industrial usage [16], specifically for packaging robots, but it also has attributes that allow it to be used for human interaction. They have developed two LWR

arms previous to the current arm, both of which have been improved upon in multiple areas [16]. The LWR-I is a seven-degree-of-freedom robot arm that used carbon fiber for its structure. It also utilized double-planetary gear heads and torque sensing for control, both of which proved to be issues for manufacturing or robustness. DLR then developed the LWR-II which used harmonic drive gear heads instead of the double-planetary gears as well as incorporating a feedback system for joint torque and motor and link position. All of the electronic systems were housed inside the arm, eliminating the external control box, which most industrial robots have.

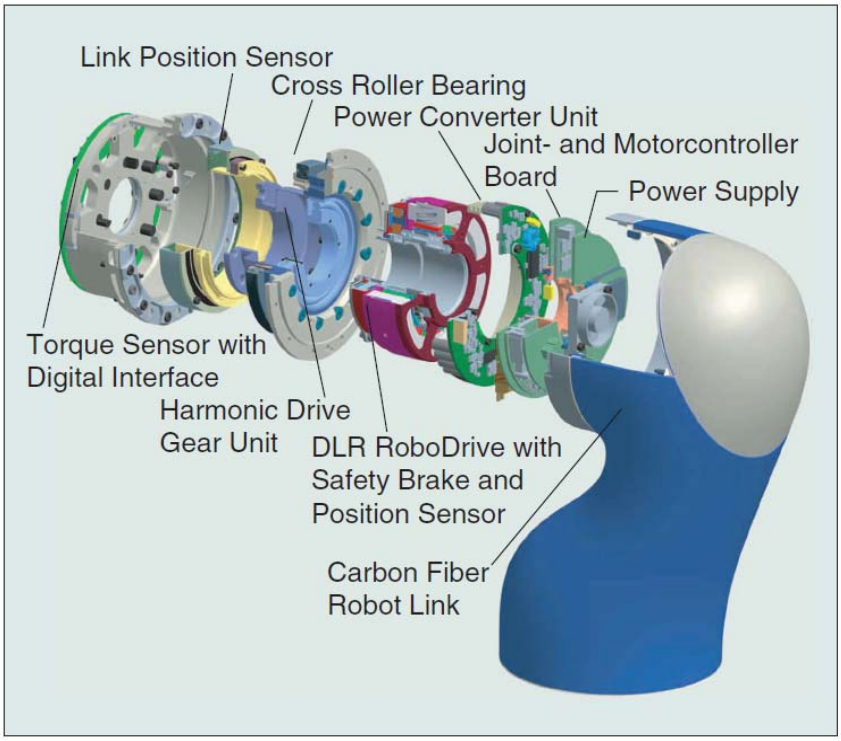

Figure 4 Mechatronic Joint Design of the DLR-LWR III [16]

2.1.4 USF WMRA Prototype Improvements

Based on this advances in the state of the art, an attempt was made to use a different material for a lighter arm using pultruded carbon fiber tubes as the structural member of each of the three main links of the arm [17]. However, due to the material failure a new design is already under development.

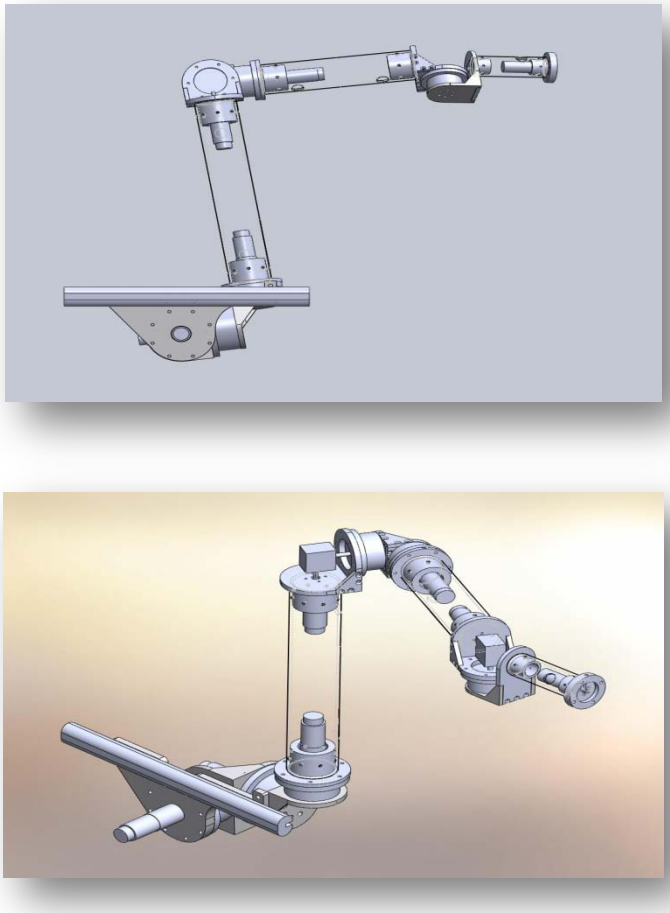

Figure 5 New WMRA Prototype SolidWorks Model

This design returns to the aluminum tubes used in the WMRA-I. However, the thickness of the tubes was reduced and the links and brackets are bolted instead of welded for better maintenance and aesthetics. The motors used for the second prototype

are also smaller in size, while keeping the same torque capabilities, accounting for a big percentage of weight reduction.

2.2 Redundant Mobile Manipulators

Redundant mobile manipulators as a research topic have gained interest with its potential for a wide range of applications. In [18], a 7 DoF mobile manipulator consisting of a 5 DoF arm mounted on a 2 DoF wheeled platform was controlled by coordinating the platform motion and the gripper motion. The platform was driven to a destination that put the target within the gripper's workspace.

The non-holonomic wheeled platform of manipulators was addressed in [19], where redundancy for a planar mobile manipulator was resolved using extended reduced gradient and projected gradient optimization-based methods. This approach was tested in simulation by having the end effector pointing at a pre-specified orientation while the wheelchair followed a circular trajectory, however they did not attempt to control two separate trajectories for the end effector and the non-holonomic base.

Path planning for non-holonomic mobile robots has been addressed by researchers for more than 2 decades. In [20], this was implemented for obstacle avoidance and implemented using the non-holonomic constrains of the platform. However, combining this constrains with a redundant manipulator was not considered at that time. In [21], an on-line planner for obstacle avoidance with moving targets was presented. Their model is

suitable for real time generation of trajectories and it was tested in crowded simulated environments.

Recent work in redundancy resolution of mobile robots has accomplished the task of sustaining separate trajectories for the end effector and the platform. In [22] they implemented redundancy resolution of a 2D mobile manipulator using independent controllers developed within each other's decoupled space, which facilitated the redundancy resolution at a dynamic level. The separate trajectories will be controlled by extending the weighted least norm solution method [23] to constrain or prioritize the motion of the platform to follow certain trajectories. This method was intended for resolving redundancy while minimizing unnecessary motion of the joints. This approach has also been used along with specific criterion functions to avoid joint limits [24].

Some applications with mobile platforms have implemented four wheel drives to account for a better accuracy of the platform motion. In [25] a patented omnidirectional mobile platform with 4 wheel drive was used for wheelchair applications.

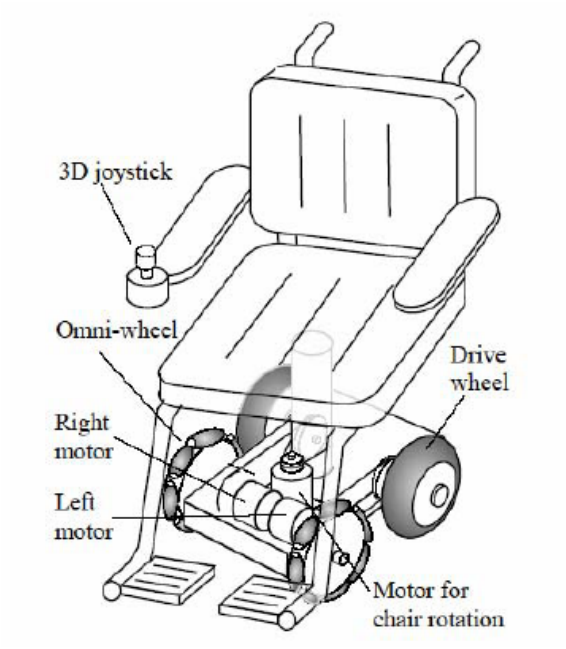

Figure 6 4WD Omni Directional Wheelchair [25]

In the paper the kinematics of the 4WD platform are analyzed and they developed a control method for omnidirectional motion. This includes the addition of a third motor for the rotation of the chair. This an application that can improve the nonholonomic constrains that are attached to WMRAs.

2.3 Virtual Reality Environments

Testing and simulation is an important step in every design process. In WMRAs the main design factor is the safety of the operator [7]. For this, 3D animations and simulations turn into a useful tool to test the behavior and robustness of the algorithms before coding it into the physical system. Previous work has shown the good use of

virtual environments for simple mobile robots [26,27] to prove the control concept in simulation.

In [25, 26] the authors describe a robotics architecture developed for their particular system called VIRbot. It is a robot designed for action planning using AI concepts. A virtual environment is implemented by the description of the working environment of the robot.

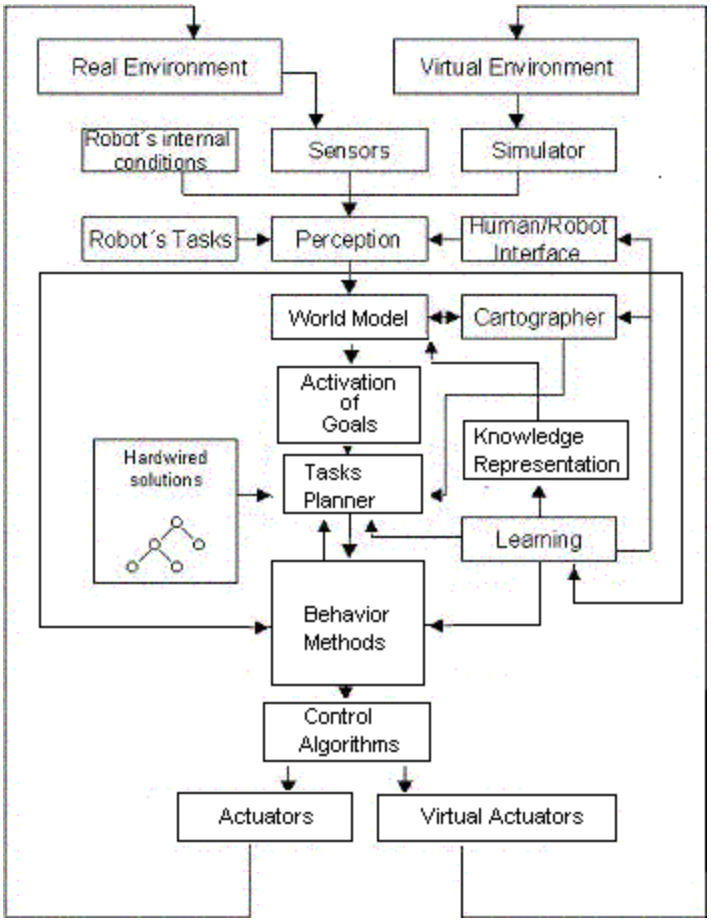

Figure 7 Virbot Subsystems Scheme [25]

The virtual environment is shown in 3D using a system called ROC2. This system uses C/C++ as its main platform, which also reads from the sensors and the main robot

program. This system uses the virtual environment as a tool to compare the actual environment with the one generated by the sensors. This is a very promising application but it does not provide the ability of running the simulation offline to test the control program as it is intended in this work.

Chapter 3: Virtual Reality Environment

Testing and simulation is an important step in every design process. In WMRAs the main design factor is the safety of the operator [7]. For this, 3D animations and simulations turn into a useful tool to test the behavior and robustness of the algorithms before coding it into the physical system. Previous work has shown the good use of virtual environments for simple mobile robots [25, 26] to prove the control concept in simulation. This work uses the virtual environment as a tool to modify and debug the control system and check for the user's safety before implementing it on the physical arm.

3.1 Virtual Reality Modeling Language

The Virtual Reality Modeling Language (VRML) is the language used to display three-dimensional objects in a browser. It is considered a 3-D web standard. Since 1994 VRML1 has been implemented in several browsers, but it allowed only the creation of static virtual worlds. This limitation reduced its widespread use. The VRML2 or VRMl97 standard was created to overcome this issue and add animation and interactivity to a

virtual world. VRML97 represents an open and flexible platform for creating interactive three-dimensional scenes (virtual worlds) [28].

Using Matalb® Virtual Reality Toolbox, a communication between the control program and the virtual worlds can be established. This adds to the main program the versatility of a 3-D animation to monitor the simulated WMRA performance.

3.2 Object Definition

The virtual environment designed for the WMRA is made of several dynamic and static objects. The objects included in this simulation were created in three different ways. For simple objects such as boxes or walls, the VRML code was typed to create the geometry, texture, location, scale and material properties. For moderately complicated geometries such as a table or a couch, the VR builder tool was used. This application allows the creation of simple primitives as a CAD program and then converts the part into VRML code. For highly complicated components such as the wheelchair and the robotic arm, SolidWorks® was used. Once the part is completed it can be easily exported into VRML files, keeping the same reference frame used in the CAD drawing. Some of the objects are included in the control loop, while others only move when there is a collision or the WMRA performs a specific task. Figure 8 shows a snapshot of the VR environment created for ADL applications.

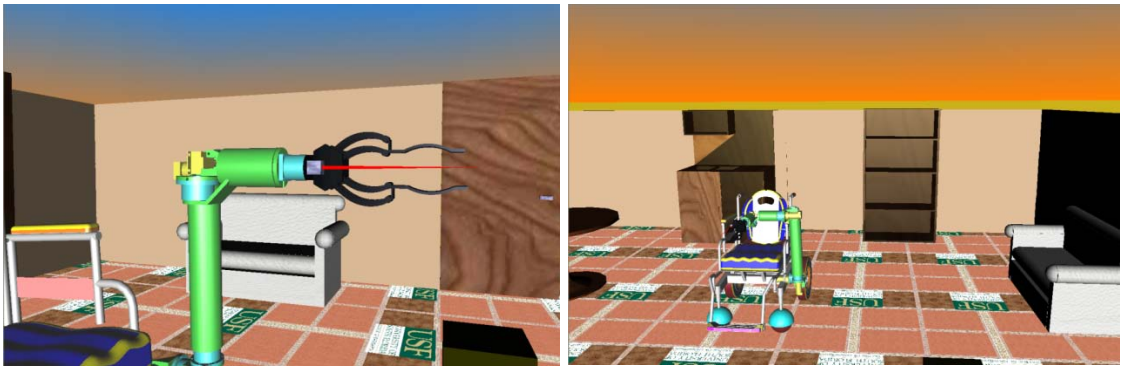

Figure 8 Virtual Reality Environment Snapshot

Table 1 shows a summarized description of the main elements developed for the virtual reality environment. The type of element shows if the element moves within the environment. The creation field specifies which type of procedure was used for its generation. And the control field presents the role of the object in the control of the simulation.

Table 1 Object Definition

Object	Type	Creation	Control
WMRA	Dynamic	SoldWorks	Control Loop
Walls	Static	Typed	N/A
Floor	Static	Typed	N/A
Laser	Dynamic	VRBuilder	Control Loop
Couch	Static	VRBuilder	N/A
Table	Static	VRBuilder	N/A
Shelves	Static	VRBuilder	N/A
Switches	Dynamic	VRBuilder	Collision
Doors	Dynamic	VRBuilder	Collision
Boxes	Dynamic	Typed	Collision
Sink	5èyl·Yì0	SoldWorks	N/A

3.2.1 Object Visualization

In this section, snapshots of the main components of the VR environment are presented. These components, and the procedure used for their creation are listed in table 1.

The first object to be presented is the couch. This object was made using VRBuilder. This software is part of Matlab Virtual Reality Toolbox and has simple geometry primitives as well as a commando prompt for VRML code input. The couch used for the simulation consists of a combination of cylinders cubes and spheres in most of the edges. VRBuilder also allows the implementation of textures. Figure 9 shows a snapshot of the couch presented in the virtual reality simulation.

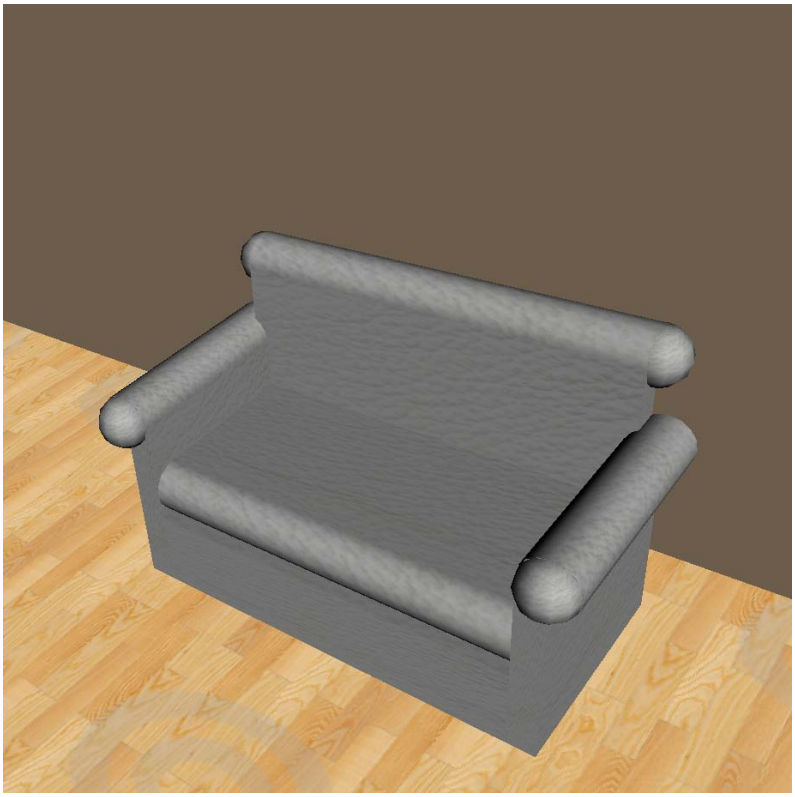

Figure 9 Snapshot of the Couch in VR Environment

The simplest objects in the simulation are the boxes. These were typed manually since it consists only of a box geometry primitive. Figure 10 shows a snapshot of the environment with the boxes in place. These objects will be used in future work for obstacle avoidance control.

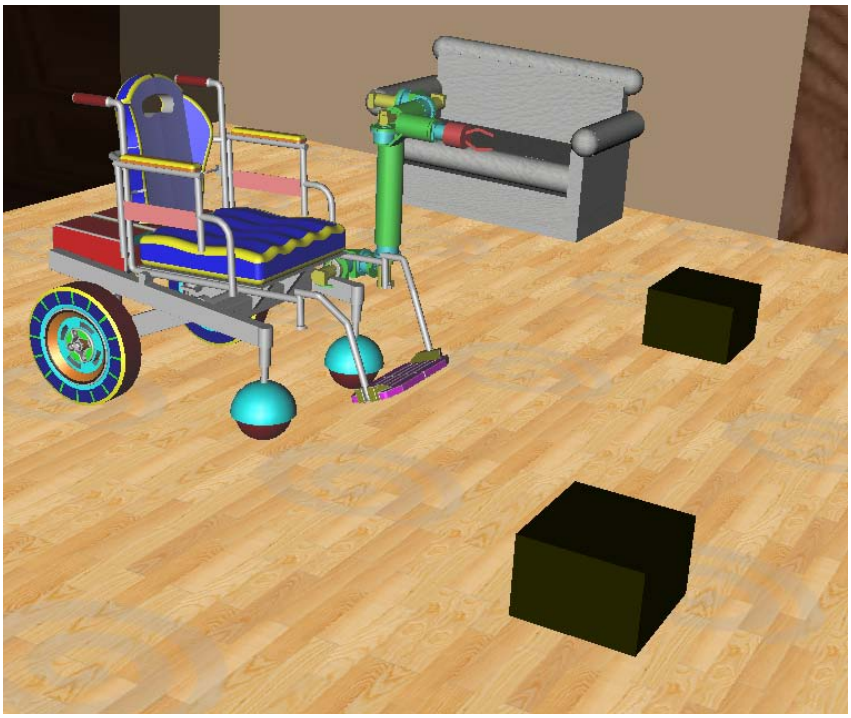

Figure 10 Environment Snapshot with Boxes in Place

A table was made to simulate “pick and place” tasks and also for obstacle avoidance. The table was also created with VRBuilder and it consisted of cylinder primitives mostly. Figure 11 shows an auxiliary view of the table in the environment.

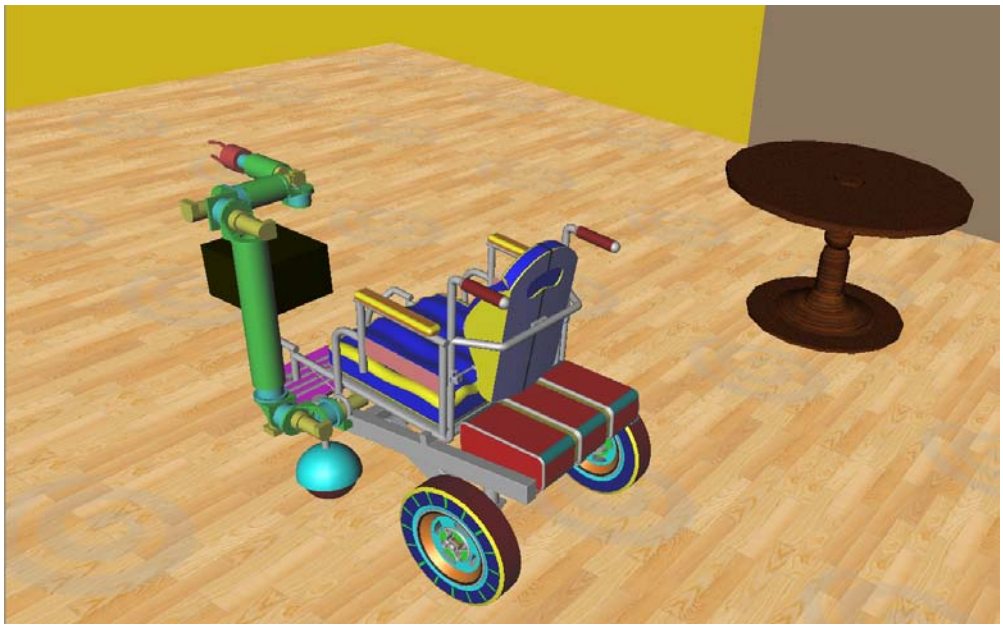

Figure 11 Snapshot of the Environment with the Table in Place

A sink was simulated to include several ADLs for future work. For instance it has doors to simulate an “opening door” task, but it also has a tab and a higher small shelf for object placement. Figure 12 shows a snapshot of the sink that is used in the simulation.

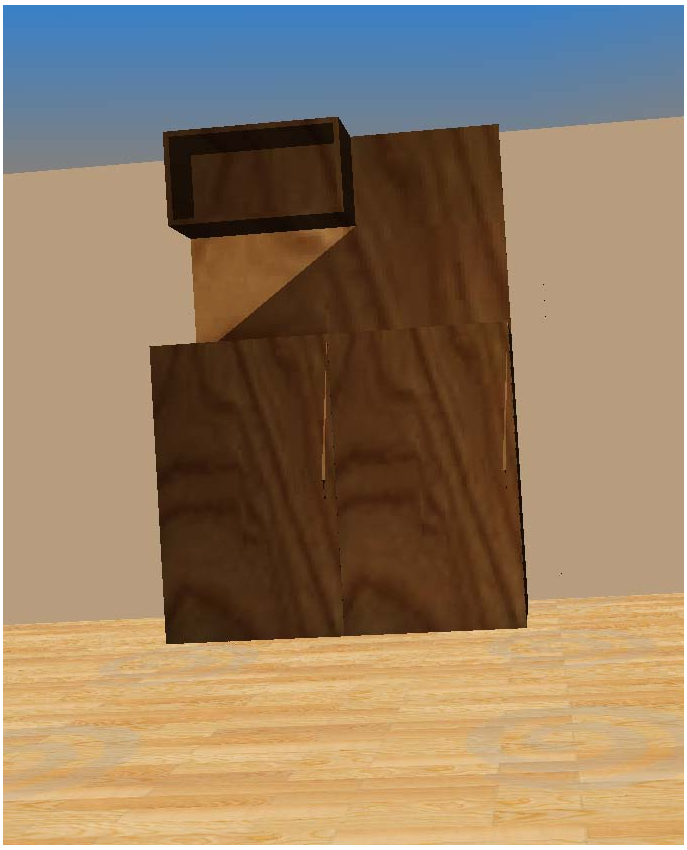

Figure 12 Snapshot of the Environment with the Sink in Place

A shelf and a book were also developed for the environment. The shelf and the book were both made using Solid Works. The parts can be saved as “wrl” files directly from the CAD program. The main objective for these objects is to simulate a pick and place task.

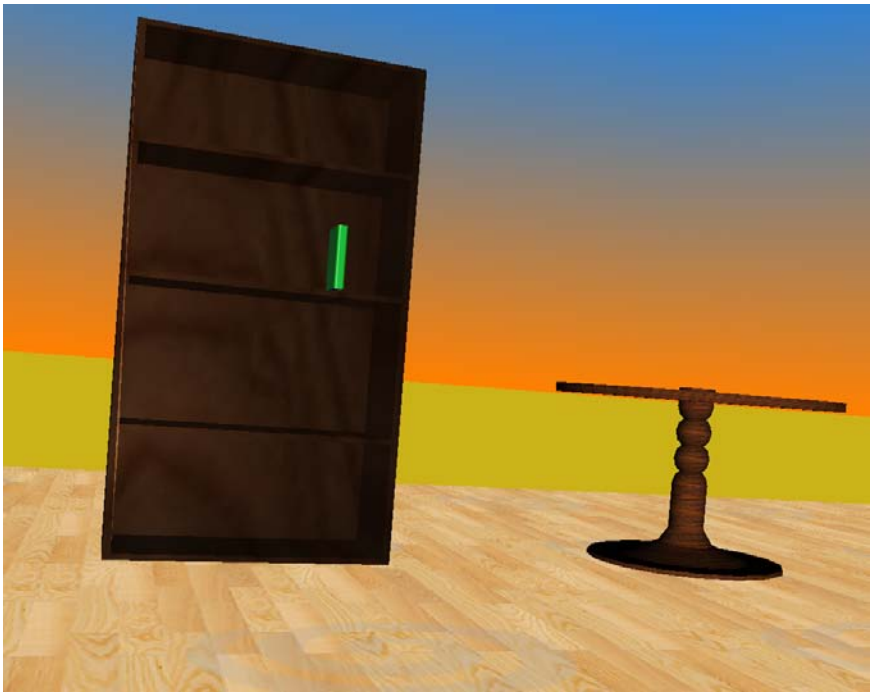

Figure 13 Snapshot of the Environment with the Shelf and the Book in Place

The objects developed for the simulation can be interfaced through Matlab and the Virtual Reality Toolbox. The configuration of the environment can be easily modified by entering the transformation matrix of the objects.

Chapter 4: Dual Trajectory Control

This is the core chapter of this thesis. The combination of the wheelchair mobility and the arm manipulation in an optimized redundancy resolution algorithm [24] allowed for the possibility of programming pre-set ADL tasks to be autonomously executed. Some ADL tasks require wheelchair orientation control to place the WMRA in a configuration that makes the task possible. In this context, having a secondary trajectory for the wheelchair to follow while the arm is following its main trajectory allows for an easier task execution.

This work utilizes redundancy to control 2 separate trajectories, a primary trajectory for the end-effector and an optimized secondary trajectory for the wheelchair. Even though this work presents results and implementation in the WMRA virtual reality simulation environment, this approach offers expandability to many wheeled base mobile manipulators in different types of applications. In this work, we develop and optimize a control system that combines the manipulation of the robotic arm and the mobility of the wheelchair in a single control algorithm. Redundancy resolution is to be optimally solved to avoid singularities and joint limits. While the end-effector follows a primary trajectory, we introduce a secondary trajectory to be followed by the wheelchair as part of the redundancy resolution and optimization algorithm.

4.1 Trajectory Generation

4.1.1 Activities of Daily Living

The main objective of the WMRA system is to maximize the manipulation capabilities of persons with disabilities who are confined to a wheelchair. This objective is reached by enhancing the performance of certain tasks that are regularly carried out every day. Activities such as turning a light switch, grabbing a cup or a book, picking up objects from the floor, opening a drawer, a cabinet or a door, are referred to as activities of daily living (ADL).

As expressed in the introduction of this chapter, the main control objective of the system is to carry out a desired end- effector trajectory. That is how activities of daily living gain importance in the control of the system; each ADL is comprised of one or several different trajectories that will need to be programmed and executed by the WMRA to complete a full task. A set of ADLs can be pre-specified towards turning the WMRA into a task oriented mobile manipulator in which the cognitive load of the user will be significantly reduced; that is by reducing the need to specify the different trajectories needed to carry out a certain task. For instance, a subset of linear and circular trajectories that are needed to open a door can be specified to be performed autonomously by the system; therefore, all that is left to be specified are the door variables (knob location, width, side to open). A simulation of the WMRA carrying out this ADL is presented in the chapter 5.

4.1.2 Trajectory Subtasks

As described in the previous subsection, every ADL can be divided into a set of trajectories that need to be executed by the system to complete a certain task. This subsection intends to provide a sense of how the trajectories are subdivided to be completed as a whole task by the WMRA

4.1.3 Trajectory Stages

End-effector trajectory and wheelchair trajectory are used as a primary and secondary trajectory respectively throughout this work. A “trajectory stage” is a term we use to pinpoint a certain portion of the trajectory to be followed in a certain period of time. Also there might be a stage in which no motion is required by the wheeled base while the end-effector is carrying out a specific trajectory. A trajectory stage is then defined as a portion of the task that follows a specific combination of motions by the WMRA to partially fulfill a desired task that is composed by several trajectories.

4.1.4 Trajectory Planning

For any given task to be programmed in a manipulator, a path needs to be followed. The orientation part of the trajectory can be represented in a single rotational angle that makes it possible to divide the path into angle steps along the trajectory. The single angle of rotation can be found from the homogeneous transform as [29]:

$$\theta = \tan^{-1}\left(\frac{\sqrt{(o_z - a_y)^2} + (a_x - n_z)^2 + (n_y - o_x)^2}{(n_x + o_y + a_z - 1)}\right) \quad (1)$$

where a typical homogeneous transformation matrix consists of three rotational vectors and one translational vector:

$$T = \begin{bmatrix} n_x & o_x & a_x & P_x \\ n_y & o_y & a_y & P_y \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Once we have the single angle of rotation and the axis of rotation, we can generate the trajectory using a linear equation. The approach used to generate the trajectory utilizes a constant transformation change along the trajectory, which means that the trajectory will be divided into “n” transformation matrices, with “δT” transformations between every two consecutive points in the trajectory.

This trajectory generation solves the problem of going from one point to another, however, going from rest to a full joint speed in no time and vice versa generates a discontinuous acceleration at the beginning and at the end of the trajectory, and therefore a linear trajectory is not the most adequate when the start and finish velocity is zero

To take care of the problems of linear trajectory, a polynomial trajectory is introduced so that when the arm starts from rest, the trajectory points are very close to each other, and then the arm will reach a maximum speed and ramp back down to zero velocity when it reaches the destination.

The governing equation for such a polynomial can be written for any variable that needs to be ramped as follows:

$$X(t) = X_0 + \frac{3}{t_f^{\,2}} \cdot (X_f - X_0) \cdot t^2 + \frac{2}{t_f^{\,3}} \cdot (X_f - X_0) \cdot t^3 \tag{2}$$

When using the polynomial function to generate a non-linear trajectory, efficiency of the trajectory-following task was not acceptable. The reason is that the desired velocity is reached at the mid-point of the trajectory, and ramping that velocity up and down takes a very long time. To overcome this problem, a polynomial blending procedure was adopted [30]. The blending factor accelerates the velocities at the beginning of the trajectory, and then set the acceleration to zero throughout the major part of the trajectory when the desired velocity is reached, and then decelerate the velocity back down to zero at the end of the trajectory-following task.

We first begin by defining the blending factor "b", a constant from 0 to 10. Then we define the acceleration during blending as:

$$\ddot{X}_b = 4 \cdot b \cdot \frac{X_f - X_i}{{t_f}^2} \tag{3}$$

Where "t_f" is the time at which the trajectory-following task is completed. Then we define the time when blending ends as:

$$t_b = \frac{t_f}{2} - \frac{\sqrt{\ddot{X}^2 \cdot t^2 - 4 \cdot \ddot{X} \cdot (X_f - X_i)}}{2 \cdot \ddot{X}} \tag{4}$$

For each ADL to be performed by the WMRA a set of trajectories have to be generated. Details on the optimization algorithm for the combined mobility and manipulation of the system can be found in previous publications [12, 24].

This procedure can be implemented with any type of trajectory. Any trajectory will be divided in steps according to the specified method. If a circular path is needed then the equation of a circle is programmed and so on.

4.2 WMRA Combined Kinematics

Two of the DoFs are provided by the non-holonomic motion of the wheelchair. This subsystem is controlled using 2 input variables: the linear position of the wheelchair along its x-axis, and the angular orientation of the wheelchair about its z-axis (see figure 14). The planar motion of the wheelchair includes three variables: the x and y positions, and the z-orientation of the wheelchair [31].

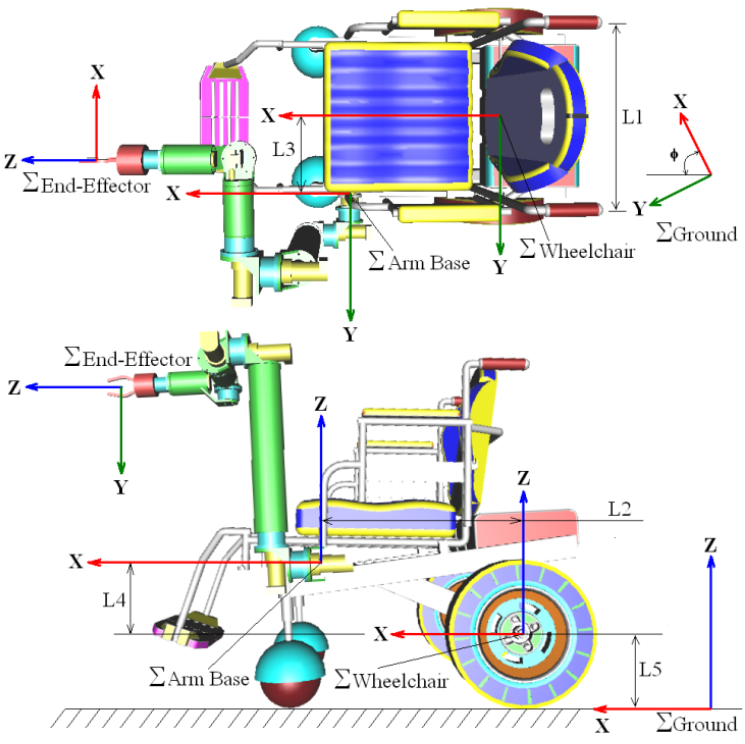

Figure 14 WMRA Coordinate Frames

Assuming that the manipulator is mounted on the wheelchair with L2 and L3 offset distances from the center of the differential drive across the x and y coordinates respectively (see Figure 14), the mapping of the wheels' velocities to the manipulator's end effector velocities along its coordinates is defined by:

$$\dot{r}_c = J_c \cdot J_W \cdot V_c, \tag{5}$$

where the end effector velocity and manipulator velocity are

$$\dot{r}_c = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} & \dot{\alpha} & \dot{\beta} & \dot{\phi} \end{bmatrix}^T, V_c = \begin{bmatrix} \dot{\theta}_l \\ \dot{\theta}_r \end{bmatrix} \text{ and } \quad J_c = \begin{bmatrix} [I]_{2x2} & \begin{bmatrix} -(P_{xg} \cdot S\phi + P_{yg} \cdot C\phi) \\ P_{xg} \cdot C\phi - P_{yg} \cdot S\phi \end{bmatrix} \\ [0]_{2x2} & [0]_{3x1} \\ [0]_{2x2} & 1 \end{bmatrix}_{6x3},$$

$$J_W = \frac{l_5}{2} \begin{bmatrix} c\phi_c + \frac{2}{l_1}(l_2 s\phi_c + l_3 c\phi_c) & c\phi_c - \frac{2}{l_1}(l_2 s\phi_c + l_3 c\phi_c) \\ s\phi_c - \frac{2}{l_1}(l_2 c\phi_c - l_3 s\phi_c) & s\phi_c + \frac{2}{l_1}(l_2 c\phi_c - l_3 s\phi_c) \\ \frac{-2}{l_1} & \frac{2}{l_1} \end{bmatrix}_{3x2}$$

where P_{xg} and P_{yg} are the x-y coordinates of the end-effector based on the arm base frame, ϕ is the angle of the arm base frame, which is the same as the angle of the wheelchair based on the ground frame and L5 is the wheels' radius. The above Jacobian can be used to control the wheelchair and the arm after combining their respective jacobians together.

The wheelchair will move forward when both wheels have the same speed and direction while rotational motion will be created when both wheels rotate at the same velocity but in opposite directions. Since the wheelchair's position and orientation are our control variables rather than the left and right wheels' velocities, V_c can be redefined as:

$$V_c = \begin{bmatrix} \dot{X} \\ \dot{\phi} \end{bmatrix}, \tag{6}$$

where: $\dot{\phi} = \frac{2l_5\dot{\theta}_r}{l_1}$, *and* $\dot{X} = l_5\dot{\theta}_r$

In the previous expression, $\dot{\phi}$ can be expressed in terms of both $\dot{\theta}_r$ or $\dot{\theta}_l$ since for pure rotation these speeds are equal but in opposite directions.

Seven degrees of freedom are provided by the robotic arm mounted on the wheelchair. From the DH parameters of the robotic arm specified in earlier publications [13], the 6x7 Jacobian that relates the joint rates to the Cartesian speeds of the end effector based on the base frame is generated according to Craig's notation [28]:

$$\dot{r}_A = J_A \cdot V_A \tag{7}$$

where: $\dot{r}_A = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} & \dot{\alpha} & \dot{\beta} & \dot{\gamma} \end{bmatrix}^T$ is the task vector,

and $V_A = \begin{bmatrix} \dot{\theta}_1 & \dot{\theta}_2 & \dot{\theta}_3 & \dot{\theta}_4 & \dot{\theta}_5 & \dot{\theta}_6 & \dot{\theta}_7 \end{bmatrix}^T$ is the joint rate vector, and J_A is the robotic arm's Jacobian.

Combining the wheelchair and arm kinematics yields the total system kinematics. In the case of combined control, let the task vector be:

$$r = f(q_c, q_A), \tag{8}$$

where q_c and q_A are the control variables of the wheelchair and arm respectively. Differentiating (8) with respect to time gives:

$$\dot{r} = \frac{\partial f}{\partial q_c} V_c + \frac{\partial f}{\partial q_A} V_A = J_c J_W V_c + J_A V_A = \begin{bmatrix} J_A & J_c J_W \end{bmatrix} \begin{bmatrix} V_A \\ V_c \end{bmatrix}, \tag{9}$$

or

$$\dot{r} = J \cdot V \tag{10}$$

where: $\dot{r} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} & \dot{\alpha} & \dot{\beta} & \dot{\phi} \end{bmatrix}^T$, $J = \begin{bmatrix} J_A & J_c J_W \end{bmatrix}$, and

$V = \begin{bmatrix} V_A & V_c \end{bmatrix}^T = \begin{bmatrix} \dot{\theta}_1 & \dot{\theta}_2 & \dot{\theta}_3 & \dot{\theta}_4 & \dot{\theta}_5 & \dot{\theta}_6 & \dot{\theta}_7 & \dot{X} & \dot{\phi} \end{bmatrix}^T$.

4.3 Redundancy Resolution and Optimization

Redundancy is resolved in the algorithm using S-R inverse of the Jacobian [32] to give a better approximation around singularities, and use the optimization for different subtasks. Manipulability measure [33] is used as a factor to measure how far is the current configuration from singularity. This measure is defined as

$$w = \sqrt{\det(J * J^T)} \tag{11}$$

The S-R Inverse of the Jacobian in this case is defined as:

$$\underline{\underline{J}}^* = \underline{\underline{J}}^T * (\underline{\underline{J}} * \underline{\underline{J}}^T + k * \underline{I_6})^{-1} \tag{12}$$

where I_6 is a 6x6 identity matrix, and k is a scale factor. It has been known that this method reduces the joint velocities near singularities, but compromises the accuracy of the solution by increasing the joint velocities error. Choosing the scale factor k is critical to minimize the error. Since the point in using this factor is to give approximate solution

near and at singularities, an adaptive scale factor is updated at every time step to put the proper factor as needed:

$$k = \begin{cases} k_0 * (1 - \frac{w}{w_0})^2 & for \quad w < w_0 \\ 0 & for \quad w \geq w_0 \end{cases} \tag{13}$$

where w_0 is the manipulability measure at the start of the boundary chosen when singularity is approached, and k_0 is the scale factor at singularity.

Weighted Least Norm solution proposed by [23] can be integrated to the control algorithm to optimize for secondary tasks. In order to put a motion preference of one joint rather than the other (such as the wheelchair wheels and the arm joints), a weighted norm of the joint velocity vector can be defined as:

$$|V|_W = \sqrt{V^T W V} \tag{14}$$

where W is a 9X9 symmetric and positive definite weighting matrix, and for simplicity, it can be a diagonal matrix that represent the motion preference of each joint of the system. For the purpose of analysis, the following transformations are introduced:

$$J_W = J W^{-1/2} \text{ , and } V_W = W^{-1/2} V \tag{15}$$

Combining these equations, it can be shown that the weighted least norm solution integrated to the S-R inverse is:

$$|V|_W = W^{-1} J^T \left(J W^{-1} J^T + k * I_6 \right)^{-1} \dot{r} \tag{16}$$

The above method has been used in simulation of the 9-DoF WMRA system with the nine control variables (V) that represent the seven joint velocities of the arm and the linear and angular wheelchair's velocities. An optimization of criteria functions can be accomplished when used in the weighting matrix W.

4.4 Secondary Trajectory Planning

For the completion of an activity of daily living, the main task will be given as a set of trajectories for the end-effector to follow. Although the main task is followed, wheelchair position and orientation can be important for the task to be successfully completed. A secondary subtask representing the best position and orientation of the wheelchair is represented as a secondary set of trajectories for the wheelchair to follow.

An optimal position/orientation combination of the non-holonomic motion of the wheelchair can be achieved if the secondary trajectory is divided into 3 stages. The first one is to orient the wheelchair facing its desired linear trajectory. The second stage is to proceed with a linear motion along the secondary trajectory to approach the final planar coordinates. Once the wheelchair reaches its final position, the third stage will be to orient the wheelchair to its final desired orientation. Figure 15 shows the three stages implemented for the secondary trajectory.

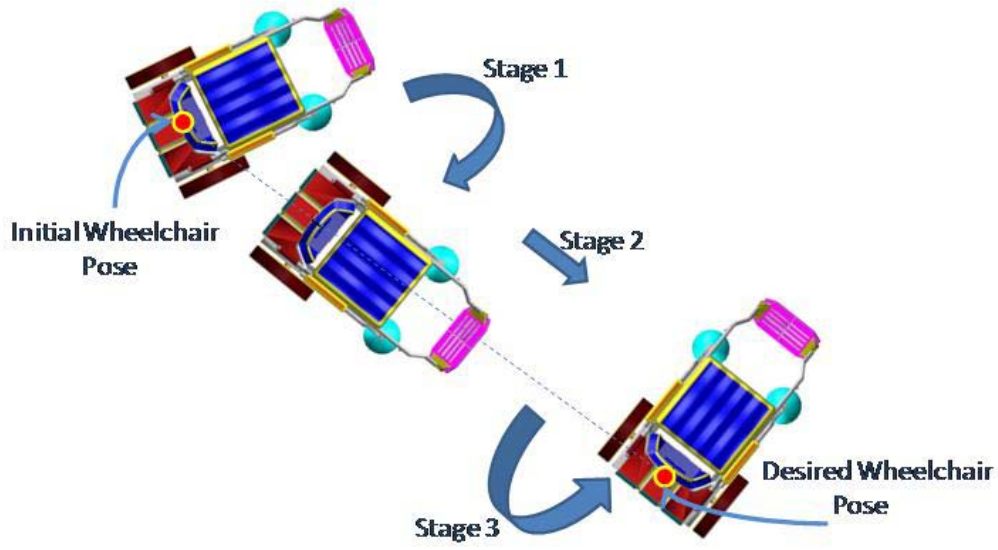

Figure 15 A General Case of the Three Stages for the Secondary Trajectory to be Followed by the Wheelchair

The three stages to be applied for the secondary trajectory will only involve the position "X" and orientation "Φ" variables of the wheelchair. The nned for the three stages comes from the non-holonomic constrains attached to the wheelchair. There are three variables to be controlled, which are the x and y position as well as the orientation of the wheelchair around the z axis, however there are only two control variables, which are the left and right wheel velocities. For an expanded derivation of these constrains in the WMRA the reader is advised to go through previous work [24]

As shown in Figure 16, knowing the initial and final transformations of the wheelchair base, the trajectory angle α can be defined as:

$$\alpha = a\tan 2\left[(y_f - y_i) \quad , \quad (x_f - x_i)\right] \tag{17}$$

That defines the amount of motion needed for the three stages to be followed in the following order:

- ✓ Rotation by the amount of $\beta_1 = \alpha - \phi_i$
- ✓ Translation by the amount of $tr = \sqrt{(x_f - x_i)^2 + (y_f - y_i)^2}$
- ✓ Rotation by the amount of $\beta_2 = \phi_f - \alpha$

The above three wheelchair motion values can be utilized in the weight matrix as criteria to enforce the wheelchair motion.

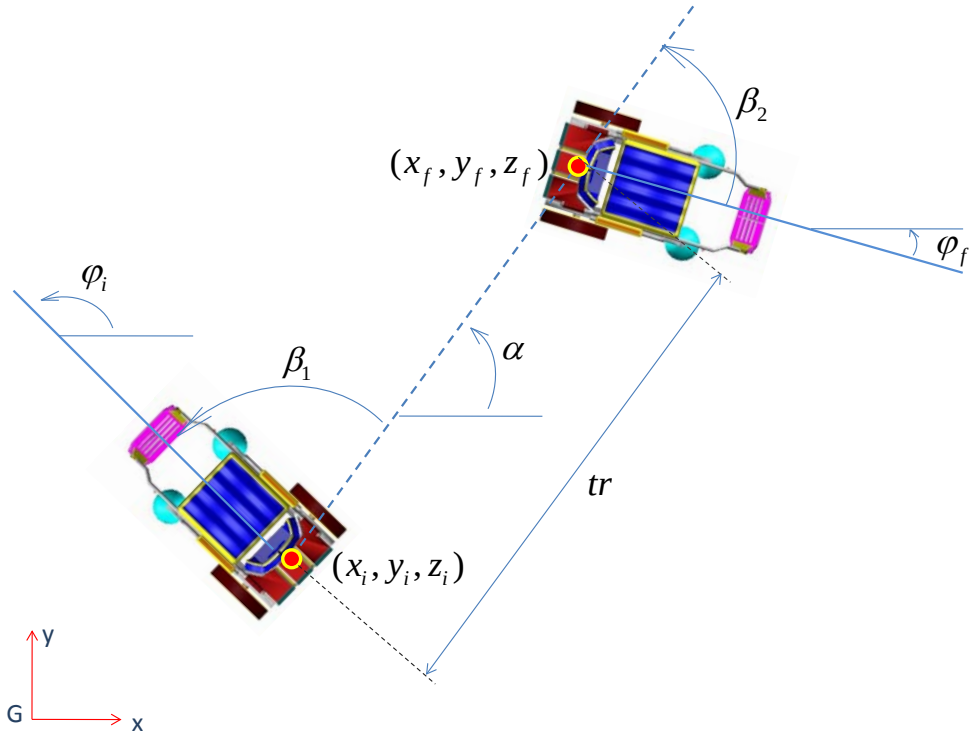

Figure 16 Definition of Optimization Variables

4.5 Criteria Functions

The criteria functions used in the weight matrix for optimization can be defined based on different requirements.

4.5.1 Joint Limit Avoidance

For the robotic arm, the physical joint limits can be avoided by minimizing an objective function that represents this criterion. One of these mathematical representations was proposed by [23] as follows:

$$H(q) = \sum_{i=1}^{7} \frac{1}{4} \cdot \frac{(q_{i,\max} - q_{i,\min})^2}{(q_{i,\max} - q_{i,current}) \cdot (q_{i,current} - q_{i,\min})} \tag{18}$$

where "q_i" is the angle of joint "i". This criterion function becomes "1" when the current joint angle is in the middle of its range, and it becomes "infinity" when the joint reaches either of its limits. The gradient projection of the criterion function can be defined as:

$$\frac{\partial H(q)}{\partial q_i} = \frac{(q_{i,\max} - q_{i,\min})^2 \cdot (2 \cdot q_{i,current} - q_{i,\max} - q_{i,\min})}{4 \cdot (q_{i,\max} - q_{i,current})^2 \cdot (q_{i,current} - q_{i,\min})^2} \tag{19}$$

When any particular joint is in the middle of the joint range, (19) becomes zero for that joint, and when it is at its limit, (19) becomes "infinity", which means that the joint will carry an infinite weight that makes it impossible to move any further.

4.5.2 Weighted Optimization

For the wheelchair, the criteria functions can be defined for each stage of its trajectory based on the desired motion of the wheelchair. Similar to the arm, mathematical representations can be obtained by treating the range of the desired wheelchair motion as a motion limit. The upper limit in this case is set to be the current initial orientation (or position for the second trajectory stage) of the wheelchair. The lower limit is set to be double the rotation angle β_1 or β_2 (or double the translation distance tr for the second trajectory stage). In this case, the middle of that range will be the desired orientation/position of the wheelchair, and either limit will be avoided. Figure 17 shows an example of the limits for the first wheelchair trajectory stage.

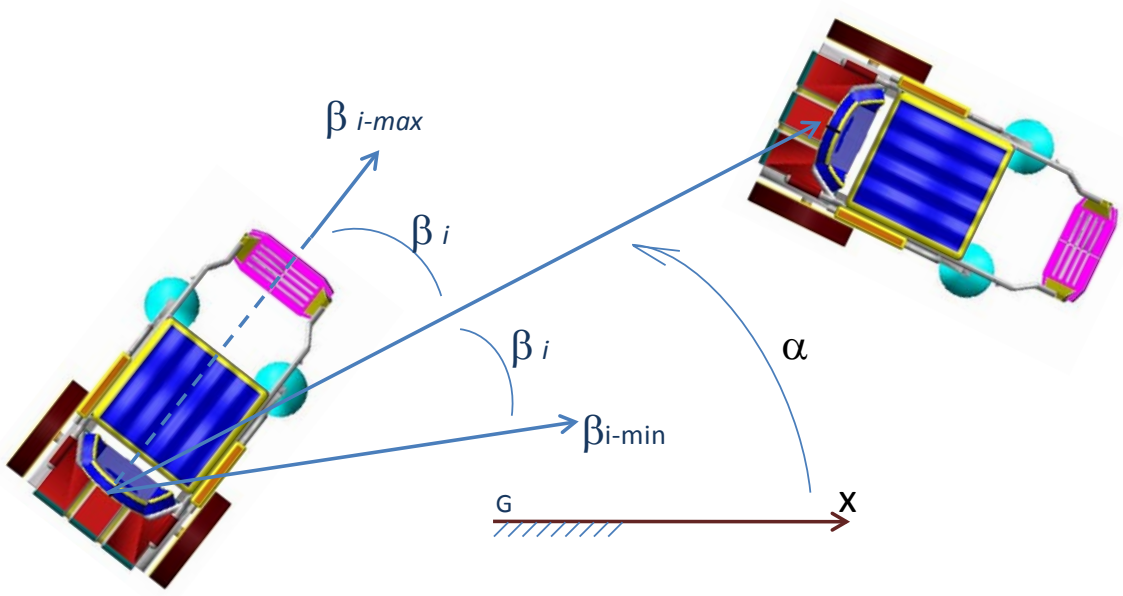

Figure 17 Gradient Variable Limits for the First Wheelchair Trajectory Stage

To generalize the representation of the objective function, let variable “P” be a representative for β1, β2 or tr. The objective function in this case is:

$$L(P) = \frac{1}{4} \cdot \frac{(P_{\max} - P_{\min})^2}{(P_{\max} - P_{current}) \cdot (P_{current} - P_{\min})} \tag{20}$$

and the gradient of the criterion function can be defined as:

$$\frac{\partial L(P)}{\partial P} = \frac{(P_{\max} - P_{\min})^2 \cdot (2 \cdot P_{current} - P_{\max} - P_{\min})}{4 \cdot (P_{\max} - P_{current})^2 \cdot (P_{current} - P_{\min})^2} \tag{21}$$

For the first stage, when the wheelchair‘s angle is in the middle of its allowable range, (21) becomes zero, and when it is at its limit, (21) becomes “infinity”, which means that the variable will carry an infinite weight that makes it impossible to move any further. This value of the gradient will be placed at the translational part of the weight matrix. The rotational part on the other hand will start with a very low value for (21). This way, rotational motion in the first stage will be active (with small weight), and the translational motion will be inactive (with high weight). The diagonal weight matrix W can then be constructed as:

$$W = \begin{bmatrix} w_1 + \left|\frac{\partial H(q)}{\partial q_1}\right| & 0 & \cdots & \cdots & 0 \\ 0 & w_2 + \left|\frac{\partial H(q)}{\partial q_2}\right| & 0 & \cdots & 0 \\ \vdots & 0 & \ddots & \cdots & \vdots \\ \vdots & \vdots & \vdots & w_X + \left|\frac{\partial L(X)}{\partial X}\right| & 0 \\ 0 & 0 & \cdots & 0 & w_\phi + \left|\frac{\partial L(\beta)}{\partial \beta}\right| \end{bmatrix} \tag{22}$$

where for stages 1 and 3:

$$\frac{\partial L(\beta)}{\partial \beta} = \frac{(\beta_{\max} - \beta_{\min})^2 \cdot (2 \cdot \beta_{current} - \beta_{\max} - \beta_{\min})}{4 \cdot (\beta_{\max} - \beta_{current})^2 \cdot (\beta_{current} - \beta_{\min})^2}, \beta_{\min} < \beta < \beta_{\max}$$

$$\frac{\partial L(X)}{\partial X} = \frac{(X_{\max} - X_{\min})^2 \cdot (2 \cdot X_{current} - X_{\max} - X_{\min})}{4 \cdot (X_{\max} - X_{current})^2 \cdot (X_{current} - X_{\min})^2}, |X| < tol$$

At the start of these stages $\left|\frac{\partial L(X)}{\partial X}\right| = \infty$ and $\left|\frac{\partial L(\beta)}{\partial \beta}\right| = 0$

And for stage 2:

$$\frac{\partial L(\beta)}{\partial \beta} = \frac{(\beta_{\max} - \beta_{\min})^2 \cdot (2 \cdot \beta_{current} - \beta_{\max} - \beta_{\min})}{4 \cdot (\beta_{\max} - \beta_{current})^2 \cdot (\beta_{current} - \beta_{\min})^2}, |\beta| < tol$$

$$\frac{\partial L(X)}{\partial X} = \frac{(X_{\max} - X_{\min})^2 \cdot (2 \cdot X_{current} - X_{\max} - X_{\min})}{4 \cdot (X_{\max} - X_{current})^2 \cdot (X_{current} - X_{\min})^2}, X_{\min} < X < X_{\max}$$

At the start of stage 2 $\left|\frac{\partial L(X)}{\partial X}\right| = 0$ and $\left|\frac{\partial L(\beta)}{\partial \beta}\right| = \infty$

w_i is a user-set preference value for each joint and the position/orientation of the wheelchair. These values can achieve the user preference if joint limits are not approached and wheelchair motion is at its desired position.

This procedure can achieve the desired trajectory combinations to successfully execute tasks that require separate end-effector and wheelchair trajectories. Examples of ADL tasks were tested in simulation and the results will be presented in the following chapter.

Chapter 5: Results and Discussion

This chapter presents the results in simulation that have being obtained for this thesis work. The computational work was done in a Core 2 Quad PC with Windows XP OS. Matlab and its virtual Reality Toolbox were used to program and run the simulation.

Trajectory planning is essential for the task-programming. Once the WMRA reaches the desired position, autonomous control will be used to follow different trajectories a task may need. The Matlab Graphics simulation allowed keeping track of the variables of concern for the control of the manipulator. The linear trajectory motion has already been tested for optimal control. More detailed results can be found in previous publications [24]. WMRA variables were monitored for different end-effector motion patterns to ensure the smooth completion of ADL tasks that include various trajectory combinations. Most of the trajectories involved in ADLs can be decomposed in several linear and circular trajectories. The next couple of figures present the result of circular trajectories. Figure 18 shows a snapshot of the Matlab graphics simulation with the WMRA model performing a circular trajectory and Figure 19 shows the joint angular displacement versus time for the completion of a circular trajectory of 1m of radius which is the radius of a typical door.

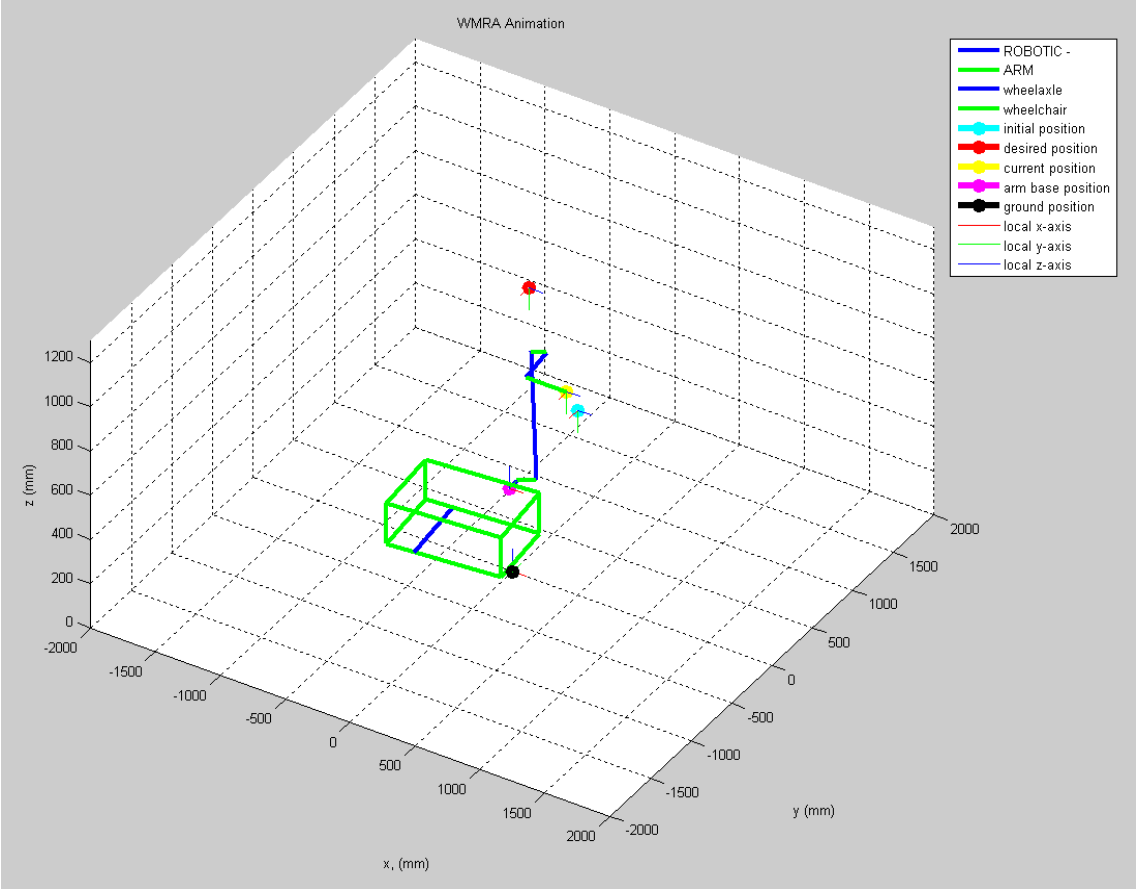

Figure 18 Circular Path Matlab Animation

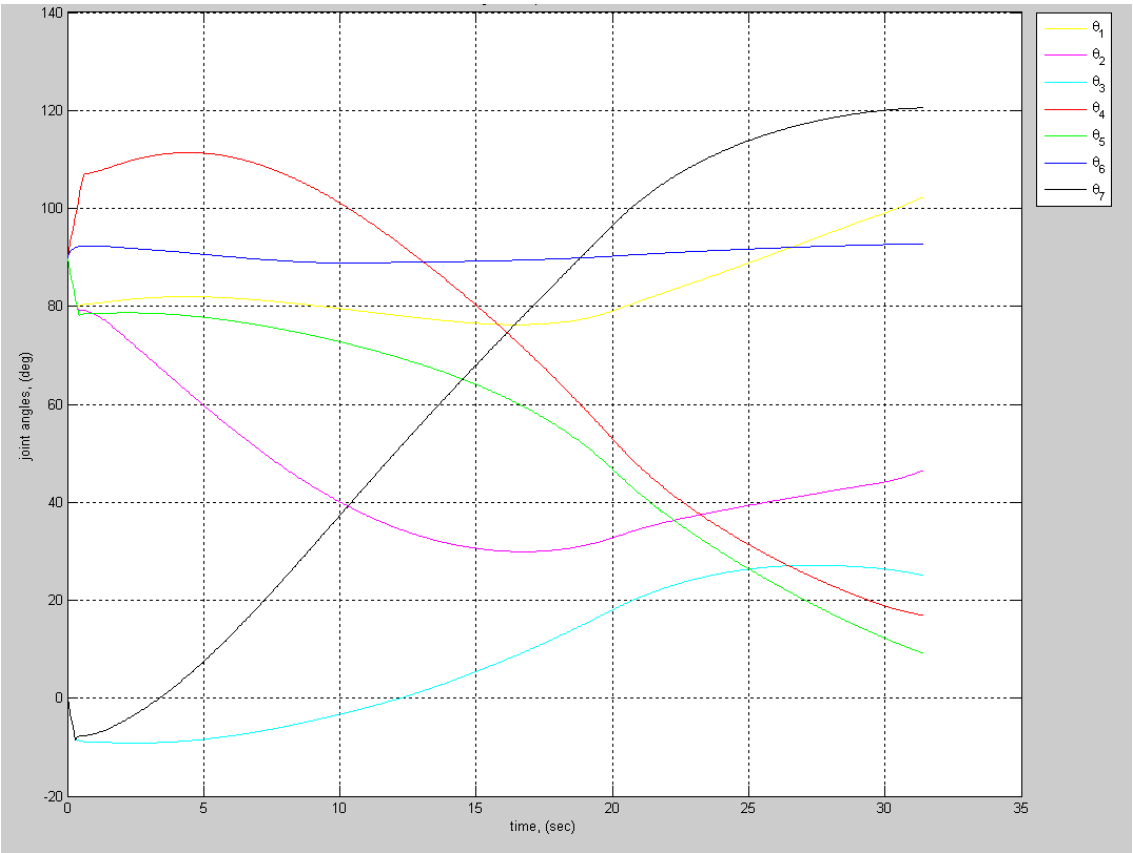

Figure 19 Joint Angular Displacement vs. Time for Circular Path

To illustrate the simulation, a virtual reality WMRA environment was developed for several activities of daily living. In this section, a "Go To and Open the Door" task was selected to prove the concept. For illustration purposes, the initial and final transformations of the end-effector trajectory are known. The initial and final transformations of the wheelchair approach trajectory are also given.

The task process is divided into two sub-tasks. The first task is to approach the door knob while both the end-effector and the wheelchair are following their respective trajectories. In this sub-task, the end-effector follows its straight-line trajectory from start

to end, and the wheelchair follows the three stages of its trajectory to approach the door approximately at the desired position and orientation. The second sub-task is to open the door inwards while the wheelchair is backing up away from the door. In this sub-task, the end-effector follows its circular trajectory to open the door, and the wheelchair follows a single-stage straight-line trajectory to back up away from the door. The wheelchair orientation during this sub-task is kept constant.

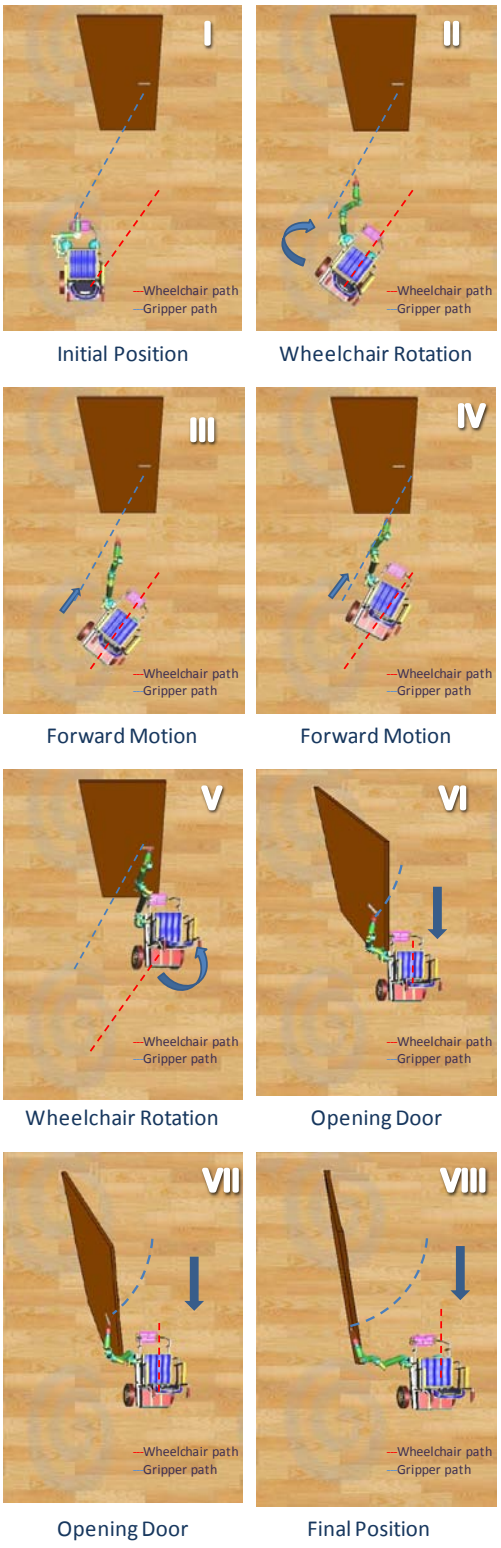

Figure 20 Virtual Reality Simulation Sequence for "Approach and Open Door" Task

Figure 20 shows the complete task execution; reaching the door and following a circular path to open it. In that sequence, the transition between figures 20-I and 20-II shows how the end-effector was following its 3D trajectory while the wheelchair was rotating without translation to reach its trajectory orientation. After reaching an approximate angle equivalent to that of the trajectory line, the wheelchair started moving forward without rotation, while the end-effector was still following its trajectory as shown in figures 20-III and 20-IV. The transition between figures 20-IV and 20-V shows how the end-effector kept following its 3D trajectory while the wheelchair was rotating back without translation to the desired orientation.

Figure 20-VI shows the end-effector following the circular trajectory to open the door, and the wheelchair was moving backwards to clear the space for the door to open. Figures 20-VII and 20-VIII show the completed task of opening the door and arriving at the final pose. Notice here that a third sub-task can be added to have the WMRA go through the door. In that case, the end-effector will need to stay stationary while the wheelchair moves forward to go through the door.

The first sub-task included a 3D trajectory for the end-effector as shown in Figure 2121, and a planar trajectory for the wheelchair. The second sub-task included two planar trajectories since the end-effector is opening the door in a planar motion.

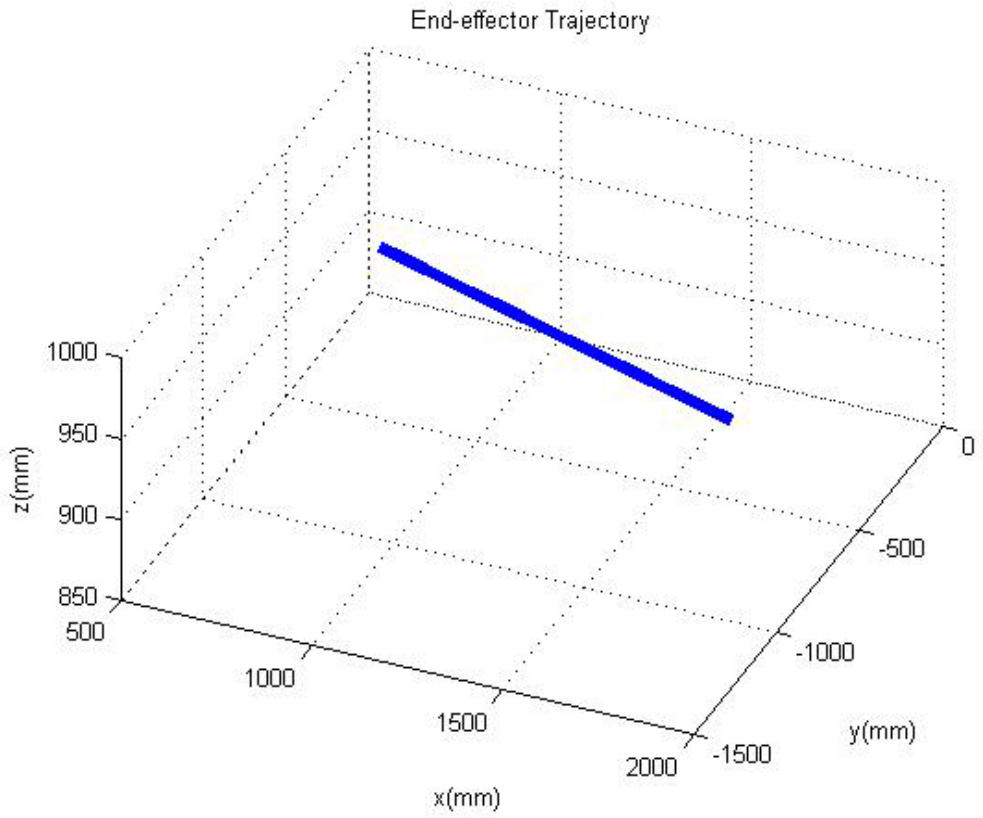

Figure 21 3D End-effector Trajectory for Approaching Task

Note that the secondary trajectory will not be followed in a precise motion. As the weights of the wheelchair position and orientation are updated at every iteration in the weight matrix, the relative motion is kept to minimum for the undesired variable motion, while the relative motion of the desired variable is kept to maximum. Figure 22 and 23 show the resulting wheelchair motion in its three position/orientation stages for the first sub-task of approaching the door. In Figure 22, it can be seen that the position stayed close to its desired trajectory throughout the wheelchair motion. Orientation, however, seems to have slightly higher error in following its desired orientation as seen in Figure 23.

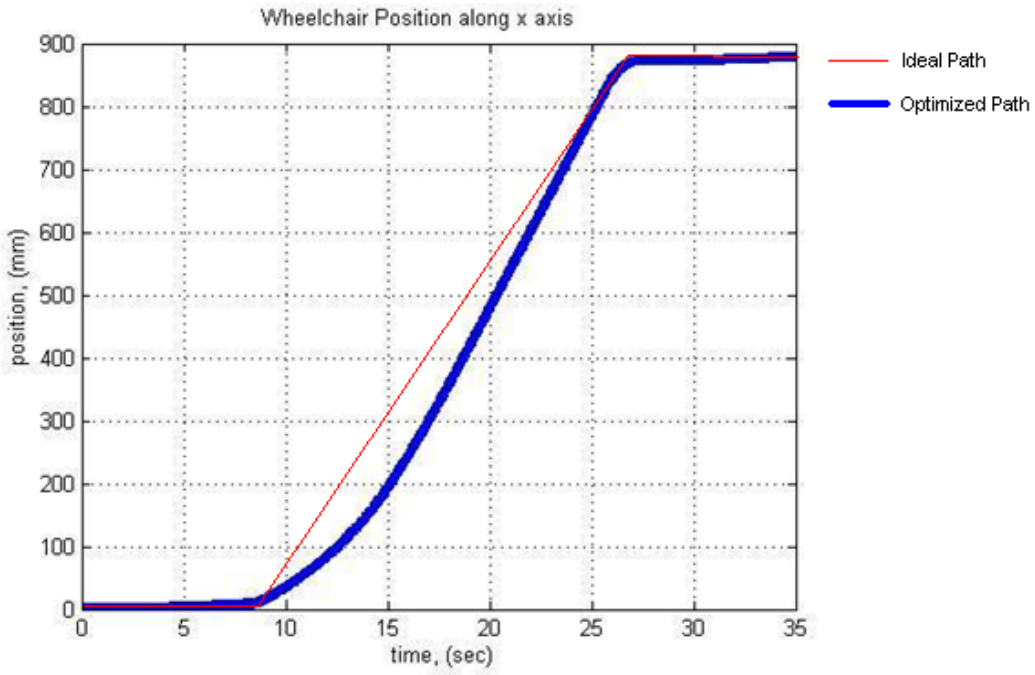

Figure 22 Wheelchair Position Vs. Time for Approaching Task

In some extreme cases, failure can occur when this algorithm is used due to the fact that it is impossible to achieve both trajectories at the same time. An example of that is when the end-effector is commanded to go in a direction that is deviating away from the desired wheelchair direction. It is shown, however, that even if the sub-task that is being performed by the wheelchair fails at certain instances, the trajectory-following of the end-effector stays unaffected.

There is a slight offset from the desired path and the actual path covered by the wheelchair. This is an optimizazion process and the results are always going to be approximates. The reason for this error is because the end-effector is trying to follow its commanded trajectory and the arm is stretching to its limit while the wheelchair is still

rotating. At this point, the wheelchair trajectory is compromised and the wheelchair starts to move forward so that the end-effector keeps up with its trajectory-following task. It was also noted that the wheelchair accelerates forward in its "stage 2" motion faster than the gripper so that it accounts for the time it will take to perform "stage 3" rotation while the end-effector is still moving along its final steps of its trajectory.

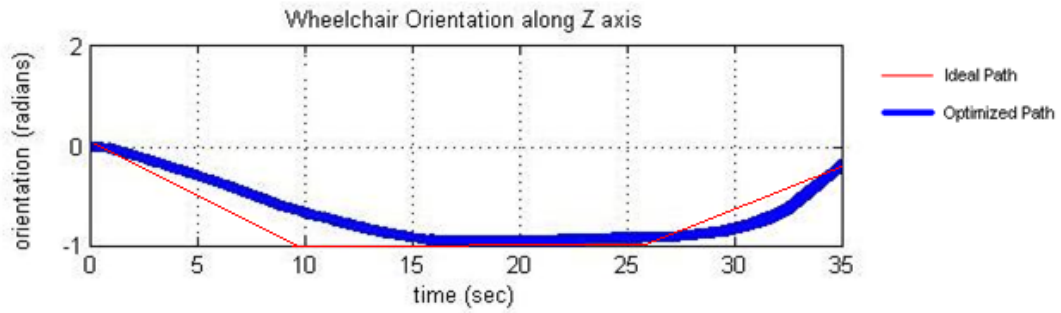

Figure 23 Wheelchair Orientation Vs. Time for Approaching Task

The same issue is presented for the wheelchair orientation while approaching the target. Even if the ideal path is set to rotate quickly and start moving forward, even when the wheelchair starts moving the desired rotation angle is not reached. Therefore, the wheelchair will keep rotating as long as the end-effector trajectory remains unaltered. This issue could also be present in the stage three. If the time for the wheelchair rotation is not accurately coordinated, the final orientation may be achieved once the end-effector has already reached its final destination

This example assumes that the door opens towards the wheelchair and towards the left side of the user. If the door opens to the right side of the user while the robotic arm is

mounted on the left side of the user, a more complicated trajectory is required to achieve acceptable results. Programming several sequences of trajectories for both the wheelchair and the end-effector can be utilized to form a complete set of tasks that can be autonomously preformed to make the WMRA system a task-oriented system.

A second orientation of the door is presented in an aerial view to illustrate the efficiency of the algorithm (see Figure 24)

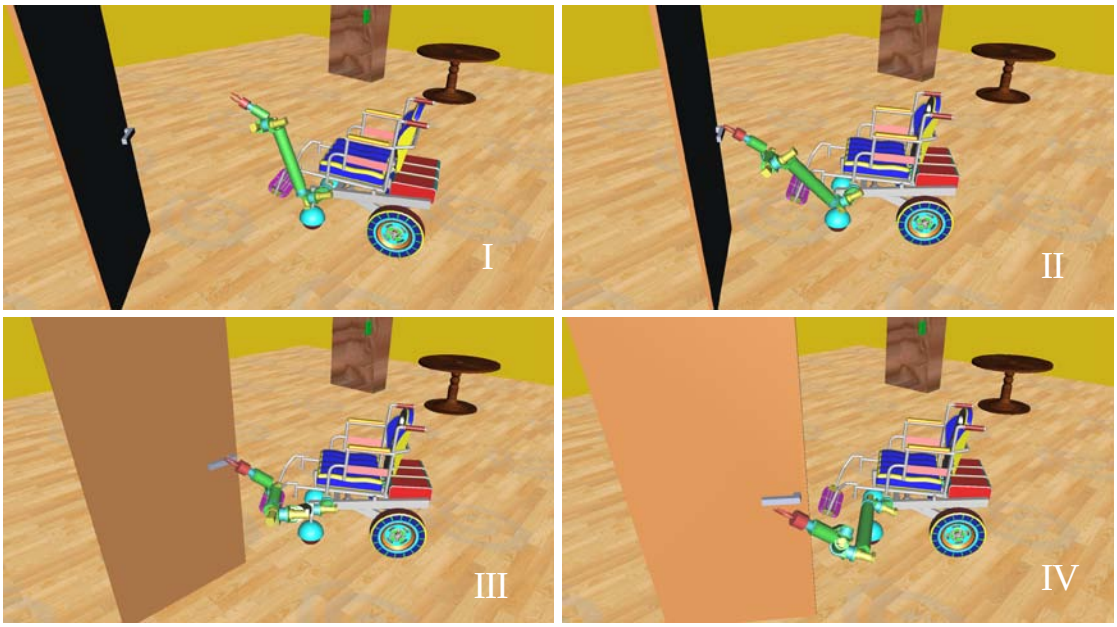

Figure 24 VR Sequence of a Second "Open Door Task"

Figures 24-I and 24-II show a sequence of the WMRA approaching the door knob, and 24-III and 24-IV show the opening of the door.

This task is the most complex tasks since it includes the dual-trajectory track to approach an object as well as different types of trajectories, for instance it includes a circular trajectory to be followed to open the door. Having successfully performed this

task allows the implementation of several trajectories in series to perform any other simpler ADL task. Figure 25 shows the sequence to approach and pick up a book.

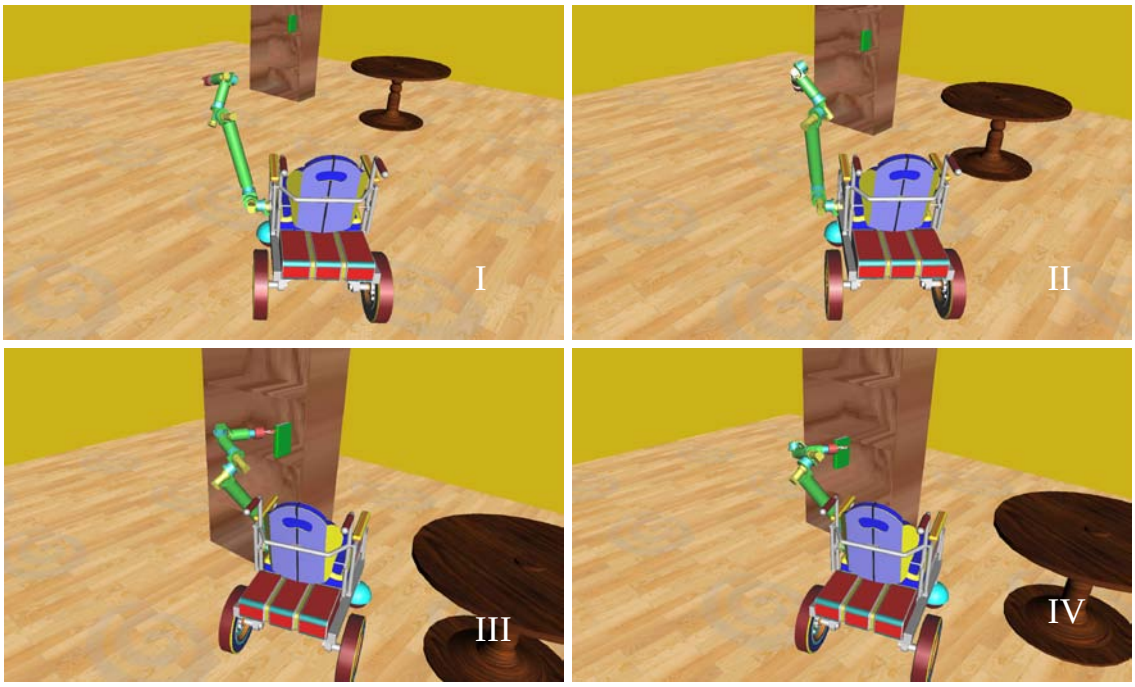

Figure 25 VR Sequence Book Pick Up

In Figures 25-I and 25-II, the WMRA is approaching the shelf where the book is. Once the book is reached in 25-III, the transformation of the book will be the same of that of the gripper. In a human-like motion the book is pulled back and the user can either take it or command the WMRA to place it somewhere else.

Chapter 6: Conclusions and Future Work

Optimized dual-trajectory following control system was presented for a 9-DoF redundant wheelchair-mounted robotic arm system to be used for people with disabilities to help them in their ADL tasks. S-R inverse was used with a weighting matrix to solve for the resolved rate solution to follow a primary trajectory. A secondary trajectory for the wheelchair to follow was mathematically represented and implemented for a "Go To and Open the Door" task. Joint limits for the manipulator joint variables and the position/orientation variables for the wheelchair were used in the weight matrix to prioritize or penalize the motion of the nine control variables. A simulation of the task in virtual reality simulation and the results were presented.

Future work includes the addition of a pool of ADL tasks in the program and the incorporation of a laser range finder to obtain position information of the target and the environment. Implementation of the control system will be done in the new prototype WMRA under development. Clinical human testing of actual ADL tasks will follow, and data will be collected and presented in future publications.

An Ongoing effort in the implementation of a Brain Computer Interface (BCI) as a user interface is also part of the future work to be accomplished for this work. This will allow the control of the system by persons with more severe disabilities such as locked-in

syndrome. The implementation of different programming languages for the control of the system is also being tested.

References

[1] US Census Bureau, “Americans with disabilities,” *Census Brief*, December 2008, http://www.census.gov/prod/2008pubs/p70-117.pdf

[2] Reswick J.B., “The moon over dubrovnik - a tale of worldwide impact on persons with disabilities,” *Advances in External Control of Human Extremities*, 1990.

[3] Topping, M.J., “The development of handy-1, a robotic system to assist the severely disabled,” Proc ICORR ’99, pp. 244-249, 1999.

[4] Topping, M., Heck, H., Bolmsjo, G., and Weightman, D., 1998, “The development of RAIL (Robotic Aid to Independent Living),” Proceedings of the third TIDE Congress.

[5] N. Katevas (Ed), “Mobile robotics in health care services,” IOS Press , pp. 227-251, Amsterdam, 2000.

[6] G. Bolmsjö, M. Olsson, P. Hedenborn, U. Lorentzon, F. Charnier, H. Nasri, “Modular robotics design - system integration of a robot for disabled people,” Proceedings of the EURISCON’98, Athens, 1998.

[7] Holly A. Yanco., “Integrating robotic research: a survey of robotic wheelchair development,” AAAI Spring Symposium on Integrating Robotic Research, Stanford, California, March 1998.

[8] Warner, P.R and Prior, S.D. , “Investigations into the design of a wheelchair-mounted rehabilitation robotic manipulator,” Proceedings of the 3rd Cambridge Workshop on Rehabilitation Robotics, Cambridge University, England, April 1994.

[9] H.Eftring, K.Boschian, “Technical results from manus user trials,” Proc. ICORR ’99, 136-141, 1999

[10] M. Hillman, A. Gammie, “The Bath institute of medical engineering assistive robot”, Proc. ICORR ’94, 211-212, 1994.

[11] R. Alqasemi, E. McCaffrey, K. Edwards and R. Dubey, “Analysis, Evaluation and Development of Wheelchair-Mounted Robotic Arms.” Proceedings of the 2005 IEEE 9th International Conference on Rehabilitation Robotics, Chicago, IL, June 2005.

[12] R. Alqasemi and R. Dubey. “Maximizing Manipulation Capabilities for People with Disabilities Using a 9-DoF Wheelchair-Mounted Robotic Arm System”. Proceedings of the 2007 IEEE 10th International Conference on Rehabilitation Robotics.

[13] K. Edwards, R. Alqasemi and R. Dubey, “Wheelchair-Mounted Robotic Arms: Design and Development,” The First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics 2006.

[14] R. Alqasemi, S. Mahler, R. Dubey, “Design and construction of a robotic gripper for activities of daily living for people with disabilities,” Proceedings of the 2007 ICORR, Noordwijk, the Netherlands, June 13–15, 2007.

[15] A. Albu-Schäffer et-al. “From Torque Feedback-Controlled Lightweight Robots to Intrinsically Compliant Systems”. *IEEE Robotics & Automation Magazine, September 2008. P.20-30*.

[16] Herzinger, G., Sporer, N., Albu-Schaffer, A., Hahnle, M., Krenn, R., Pascucci, A., & Schedl, M. (2002). *“DLR’s Torque Controlled Lightweight Robot III – Are We Reaching the Technological Limits Now?*” Proceedings of the IEEE International Conference on Robotics & Automation. Pgs 1710-1716.

[17] P. Shrock, F. Farelo, R. Alqasemi and R. Dubey. “Design, Simulation and Testing of a New Modular Wheelchair Mounted Robotic Arm to Perform Activities of Daily Living” Proceedings of the *2009 IEEE 11th International Conference on Rehabilitation Robotics*.

[18] C. Ding, P. Duan, M. Zhang and H. Liu. “The Kinematics of a Redundant Mobile Manipulator”, Proceedings of the *IEEE International Conference on Automation and Logistics*. 2009.

[19] A. Luca, G. Oriolo, and P. Giordano, “Kinematic Modeling and Redundancy Resolution for Nonholonomic Mobile Manipulators”, Proceedings of the *2006 IEEE International Conference on Robotics and Automation* (ICRA), pp. 1867-1873.

[20] R. Zapata, JB. Thevenon, M. Perrier, E. Pommier and E. Badi. “Path Planning abd trajectory Planning for Non-holonomic Mobile Robots”, *IEE/RJS International Workshop on Intelligent Robot and Systems*. 1989

[21] O. Gal, Z. Shiller and E. Rimon. “Efficient and Safe On-line Motion planning ib Dynamic Environments”, Proceedings of the *2009 IEEE International Conference on Robotics and Automation* (ICRA).

[22] G.D. white, R.M. Bhatt, C Pei Tang and V.N. Krovi. “Experimental Evaluation of Dynamic redundancy resolution in a Nonholonomic wheeled Mobile Manipulator”. *IEE/ASME Transactions on Mechatronics*, Vol. 14, No 3. 2009

[23] T. Chan, and R. Dubey, “A Weighted Least-Norm Solution Based Scheme for Avoiding Joint Limits for Redundant Joint Manipulators”, *IEEE Robotics and Automation Transactions* (R&A Transactions). V. 11, N. 2, pp. 286-292, 1995.

[24] R. Alqasemi. “Maximizing Manipulation Capabilities for People with Disabilities Using a 9-DoF Wheelchair-Mounted Robotic Arm System”. Doctoral Dissertation. University of South Florida. Tampa, 2007.

[25] M Wada. “ An omnidirectional 4WD Mobile Platform for Wheelchair Applications.” Proceedings of the 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics.

[26] J. Savage et al. “ViRbot: A System for the Operation of Mobile Robots.” Lecture Notes in Computer Science. Springer Berlin/ Heidelberg. Vol 5001. 512-519, 2008.

[27] J. Savage, M. Billinghurst, A. Holden, “The Virbot: A virtual reality mobile robot driven with multimodal commands.” Expert Systems with Applications 15. 413-419, 1998.

[28] The Mathworks. “Virtual Reality Toolbox User’s Guide.” The Mathworks Inc. Version 4.0.1, 2006

[29] R. Paul, 1981, “Robot Manipulators: Mathematics, Programming and Control”, The MIT Press, ISBN 026216082-X.

[30] Craig, J., 2003, “Introduction to robotics mechanics and control”, third edition, Addison- Wesley Publishing, ISBN 0201543613.

[31] E. Papadopoulos, J. Poulakakis, “Planning and model-based control for mobile manipulators,” *Proceedings of the 2001 IROS*, 2000

[32 Nakamura, Y., “Advanced robotics: redundancy and optimization,” Addison- Wesley Publishing, 1991, ISBN 0201151987.

[33] Yoshikawa, T., “Foundations of robotics: analysis and control,” MIT Press, 1990, ISBN 0262240289.

Appendices

Appendix A. Virtual Reality Modeling Language

9_WMRA

```
#VRML V2.0 utf8


NavigationInfo { type "EXAMINE" speed 30 avatarSize [ 1, 0, 0 ]
headlight TRUE }
DEF WMRAROBOT Group {
  children[
    Group {
      children [

DEF EXT_SETTINGS Group {
      children [
        WorldInfo { title "Wheelchair Mounted Robotic Arm, By: Redwan
Alqasemi, USF 2007"},
        NavigationInfo {
           type "EXAMINE"
           avatarSize 180
           visibilityLimit 200
           speed 1000
        },


        Background {
            groundColor [ 0.8 0.7 0.1 , 0.8 0.7 0.1]
            groundAngle [1.57]
            skyColor [ 0 0 1 , 0 0.5 1 , 0 0.5 1 , 0.5 0.5 0.5 , 1 0.5
0]

            skyAngle [ 1 1.15 1.35 1.57]
            #topUrl "cloud.jpg"
        },


        DEF DynamicView Transform {
        rotation 0 1 0 0
        translation 0 0 0
          children [
            Viewpoint {
              description "a_start"
              position 2500 500 1800
```

```
  orientation 0 1 0 0.8
  jump FALSE
},
Viewpoint {
  description "a_far"
  position 900 6000 -200
  orientation -0.601 -0.547 -0.582 2.172
  jump FALSE
},
Viewpoint {
  description "a_bk-lt-up"
  position -1300 1600 -1600
  orientation -0.1 -1 -0.25 2.4
  jump FALSE
},
Viewpoint {
  description "a_bk-lt-dn"
  position -1400 400 -1800
  orientation 0.025 -1 0.037 2.4
  jump FALSE
  },
Viewpoint {
  description "a_ft-lt-up"
  position 1600 1800 -1400
  orientation -0.1 0.9 0.25 2.4
  jump FALSE
},
Viewpoint {
  description "a_ft-lt-dn"
  position 1700 400 -1600
  orientation 0.031 1 -0.052 2.4
  jump FALSE
},
Viewpoint {
  description "a_ft-rt-up"
  position 1600 1900 1500
  orientation -0.4 0.5 0.14 0.85
  jump FALSE
},
Viewpoint {
  description "a_ft-rt-dn"
  position 1700 300 1900
  orientation 0.191 1 -0.075 0.615
  jump FALSE
},
Viewpoint {
  description "a_bk-rt-up"
  position -1700 1700 1700
  orientation -0.25 -0.5 -0.12 1
  jump FALSE
},
Viewpoint {
  description "a_bk-rt-dn"
```

Appendix A (Continued)

```
        position -1800 500 1900
        orientation 0.116 -1 0.021 0.818
        jump FALSE
      },
      Viewpoint {
        description "a_birdeye"
        position -1100 4900 -1900
        orientation -0.56 -0.72 -0.4 2.2
        jump FALSE
      },
      Viewpoint {
        description "a_top"
        position 200 3100 0
        orientation -0.577 -0.577 -0.577 2.1
        jump FALSE
      },
    ]}
  Viewpoint {
    description "top"
    position 200 3100 0
    orientation -0.577 -0.577 -0.577 2.1
    jump FALSE
  },
  Viewpoint {
    description "birdeye"
    position -1100 4900 -1900
    orientation -0.56 -0.72 -0.4 2.2
    jump FALSE
  },
  Viewpoint {
    description "bk-rt-dn"
    position -1800 500 1900
    orientation 0.116 -1 0.021 0.818
    jump FALSE
  },
  Viewpoint {
    description "bk-rt-up"
    position -1700 1700 1700
    orientation -0.25 -0.5 -0.12 1
    jump FALSE
  },
  Viewpoint {
    description "ft-rt-dn"
    position 1700 300 1900
    orientation 0.191 1 -0.075 0.615
    jump FALSE
  },
  Viewpoint {
    description "ft-rt-up"
    position 1600 1900 1500
    orientation -0.4 0.5 0.14 0.85
    jump FALSE
  },
```

Appendix A (Continued)

```
        Viewpoint {
          description "ft-lt-dn"
          position 1700 400 -1600
          orientation 0.031 1 -0.052 2.4
          jump FALSE
        },
        Viewpoint {
          description "ft-lt-up"
          position 1600 1800 -1400
          orientation -0.1 0.9 0.25 2.4
          jump FALSE
        },
        Viewpoint {
          description "bk-lt-dn"
          position -1400 400 -1800
          orientation 0.025 -1 0.037 2.4
          jump FALSE
          },
        Viewpoint {
          description "bk-lt-up"
          position -1300 1600 -1600
          orientation -0.1 -1 -0.25 2.4
          jump FALSE
        },
        Viewpoint {
          description "far"
          position 900 6000 -200
          orientation -0.601 -0.547 -0.582 2.172
          jump FALSE
        },
        Viewpoint {
          description "start"
          position 2500 500 1800
          orientation 0 1 0 0.8
          jump FALSE
        },

        DEF GROUND Transform {
           rotation 1 0 0 0
           translation 0 0 0
           children [
           Shape {
           geometry Box { size 10000 1 10000 }
           appearance Appearance {
           texture ImageTexture { url "woodfloor2.jpg" repeatS TRUE
repeatT TRUE }
           textureTransform TextureTransform {
           rotation 0
           center 0 0
           translation  0 0
           scale    3 3
        }}}]},
```

```
    ]}
    ]}

        # Transforming the wheelchair world coordinate system to the
VR's world coordinate system:
        DEF World Transform {
        rotation 1 0 0 -1.5707963
        translation 0 0 0
        children [

        DEF Chair Transform {
        rotation 0 0 1 0
        translation -440 -230 168
          children [
#          DEF WCR SphereSensor {}
#          DEF WCT PlaneSensor { minPosition -400 0 maxPosition 400 0 }
          Group {
          children [Inline { url "0_Chair.wrl" }

        DEF LWheel Transform {
        rotation 0 1 0 0
        translation 0 0 0
          children [
#          DEF LW CylinderSensor { diskAngle 0 minAngle 1.5707963
maxAngle 1.5707963 }
          Group {
          children [Inline { url "0_LWheel.wrl" }]}]}

        DEF RWheel Transform {
        rotation 0 1 0 0
        translation 0 0 0
          children [
#          DEF RW CylinderSensor { diskAngle 0 minAngle 1.5707963
maxAngle 1.5707963 }
          Group {
          children [Inline { url "0_RWheel.wrl" }]}]}

        DEF ARM1 Transform {
        rotation 1 0 0 1.5707963
        translation 440 220 139
          children [
#          DEF JOINT1 CylinderSensor { diskAngle 0 minAngle 1.5707963
maxAngle 1.5707963 }
          Group {
          children [Inline { url "1.wrl" }

        DEF ARM2 Transform {
        rotation 0 0 -1 1.5707963
        translation 0 42.69 -75.1
          children [
#          DEF JOINT2 CylinderSensor { diskAngle 0 minAngle -1.5708
maxAngle 1.5708 }
```

```
          Group {
          children [Inline { url "2.wrl" }

        DEF ARM3 Transform {
        rotation 0 1 0 1.5707963
        translation -1.73 75.08 -42.7
          children [
#          DEF JOINT3 CylinderSensor { diskAngle 0 minAngle -3.1416
maxAngle 3.1416 }
          Group {
          children [Inline { url "3.wrl" }

        DEF ARM4 Transform {
        rotation 0 0 -1 0
        translation -2.92 42.64 -75.08
          children [
#          DEF JOINT4 CylinderSensor { diskAngle 0 minAngle -3.1416
maxAngle 3.1416 }
          Group {
          children [Inline { url "4.wrl" }

        DEF ARM5 Transform {
        rotation 0 1 0 1.5707963
        translation -11.45 74.85 -423.58
          children [
#          DEF JOINT5 CylinderSensor { diskAngle 0 minAngle -3.1416
maxAngle 3.1416 }
          Group {
          children [Inline { url "5.wrl" }

        DEF ARM6 Transform {
        rotation 0 0 -1 1.5707963
        translation -2.17 45.99 -75.1
          children [
#          DEF JOINT6 CylinderSensor { diskAngle 0 minAngle -3.1416
maxAngle 3.1416 }
          Group {
          children [Inline { url "6.wrl" }

        DEF ARM7 Transform {
        rotation 0 1 0 1.5707963
        translation -2.92 -61.52 -161.49
          children [
#          DEF JOINT7 CylinderSensor { diskAngle 0 minAngle -1.5708
maxAngle 1.5708 }
          Group {
          children [Inline { url "7.wrl" }

        DEF ARM8 Transform {
        rotation 0 0 -1 0
        translation -1.78 61.39 -192.29
          children [
```

Appendix A (Continued)

```
#          DEF JOINT8 CylinderSensor { diskAngle 0 minAngle -3.1416
maxAngle 3.1614 }
          Group {
          children [Inline { url "8.wrl" }

        ]}]} ]}]} ]}]} ]}]} ]}]} ]}]} ]}]} ]}]} ]}]} ]}]} ]}

#         ROUTE WCT.translation_changed TO Chair.set_translation
#         ROUTE WCR.rotation_changed TO Chair.set_rotation
#         ROUTE LW.rotation_changed TO LWheel.set_rotation
#         ROUTE RW.rotation_changed TO RWheel.set_rotation
#         ROUTE JOINT1.rotation_changed TO ARM1.set_rotation
 #        ROUTE JOINT2.rotation_changed TO ARM2.set_rotation
  #       ROUTE JOINT3.rotation_changed TO ARM3.set_rotation
   #      ROUTE JOINT4.rotation_changed TO ARM4.set_rotation
    #     ROUTE JOINT5.rotation_changed TO ARM5.set_rotation
     #    ROUTE JOINT6.rotation_changed TO ARM6.set_rotation
      #   ROUTE JOINT7.rotation_changed TO ARM7.set_rotation
       # ROUTE JOINT8.rotation_changed TO ARM8.set_rotation

]}
```

Box

```
#VRML V2.0 utf8
Group {
children [
    Transform {
    translation 0 0 0
    children [
        DEF BOX Shape {
                    appearance  Appearance {
                        material    DEF _mat1 Material {
                            ambientIntensity    0.2
                            diffuseColor    0.2 0.2 0
                            emissiveColor   0 0 0
                            shininess   0.2
                            specularColor   0 0 0
                            transparency    0
                        }
}

        geometry Box { size 300 200 300}
}
]
}
]
}
```

Appendix A (Continued)

Couch

```
#VRML V2.0 utf8

#Created with V-Realm Builder v2.0
#Integrated Data Systems Inc.
#www.ids-net.com


Group {
    children [
        NavigationInfo {
        }

        Transform {
            scale   60 60 60
            children [
            Group {
                children Transform {
                    translation 8 2 4
                    children [
                        Shape {
                            appearance  Appearance {
                                material    DEF _mat1 Material {
                                    ambientIntensity    0.2
                                    diffuseColor    0.1 0.4 0.5
                                }

                                texture ImageTexture {
                                    url "texture/leather_white.jpg"
                                }

                            }

                            geometry    Sphere {
                                radius  1
                            }

                        }

                        Transform {
                            translation 0 0 -4
                            rotation    1 0 0  1.57
                            children Shape {
                                appearance  Appearance {
                                    material    USE _mat1
                                    texture ImageTexture {
                                        url "texture/leather_white.jpg"
```

Appendix A (Continued)

```
                                }

                            }

                            geometry    Cylinder {
                                height  8
                                radius  1
                            }

                        }
                    }
                ]
            }
        }

        Group {
            children Transform {
                translation -8 2 4
                children [
                    Shape {
                        appearance  Appearance {
                            material    USE _mat1
                            texture ImageTexture {
                                url "texture/leather_white.jpg"
                            }

                        }

                        geometry    Sphere {
                            radius  1
                        }

                    }

                    Transform {
                        translation 0 0 -4
                        rotation    1 0 0  1.57
                        children Shape {
                            appearance  Appearance {
                                material    USE _mat1
                                texture ImageTexture {
                                    url "texture/leather_white.jpg"
                                }

                            }

                            geometry    Cylinder {
                                height  8
                                radius  1
                            }
```

Appendix A (Continued)

```
                }
                    }
                ]
            }
        }

        Group {
            children Transform {
                translation 7.5 -1.5 0
                children [
                    Shape {
                        appearance  Appearance {
                            material    USE _mat1
                            texture ImageTexture {
                                url "texture/leather_white.jpg"
                            }

                        }

                        geometry    Box {
                            size    1 7 8
                        }

                    }

                    Transform {
                        translation -15 0 0
                        children Shape {
                            appearance  Appearance {
                                material    USE _mat1
                                texture ImageTexture {
                                    url "texture/leather_white.jpg"
                                }

                            }

                            geometry    Box {
                                size    1 7 8
                            }

                        }
                    }
                ]
            }
        }

        Group {
            children Transform {
                translation 0 -3.5 0
                children Shape {
                    appearance  Appearance {
```

Appendix A (Continued)

```
                    material    USE _mat1
                        texture ImageTexture {
                            url "texture/leather_white.jpg"
                        }

                    }

                    geometry    Box {
                        size    14 3 8
                    }

                }
            }
        }

        Group {
            children Transform {
                translation 0 1.5 -3.5
                children [
                    Shape {
                        appearance  Appearance {
                            material    USE _mat1
                            texture ImageTexture {
                                url "texture/leather_white.jpg"
                            }

                        }

                        geometry    Box {
                            size    15 7 1
                        }

                    }

                    Transform {
                        translation 7.5 3.5 -0.5
                        children [
                        Shape {
                            appearance  Appearance {
                                material    USE _mat1
                                texture ImageTexture {
                                    url "texture/leather_white.jpg"
                                }

                            }

                            geometry    Sphere {
                                radius  1
                            }

                        }
```

Appendix A (Continued)

```
                            Transform {
                                translation -15 0 0
                                children [
                                    Shape {
                                        appearance  Appearance {
                                            material    USE _mat1
                                            texture ImageTexture {
                                                url
"texture/leather_white.jpg"
                                            }

                                        }

                                        geometry    Sphere {
                                            radius  1
                                        }

                                    }

                                    Transform {
                                        translation 7.5 0 0
                                        rotation    0 0 1  1.57
                                        children Shape {
                                            appearance  Appearance {
                                                material    USE _mat1
                                                texture ImageTexture {
                                                    url
"texture/leather_white.jpg"
                                                }

                                            }

                                            geometry    Cylinder {
                                                height  15
                                                radius  1
                                            }

                                        }
                                    }
                                ]
                            }
                            ]
                        }
                    ]
                }
            }

            Group {
                children Transform {
                    translation 0 -1 0
                    children [
```

Appendix A (Continued)

```
                        Shape {
                            appearance  Appearance {
                                material    USE _mat1
                                texture ImageTexture {
                                    url "texture/leather_white.jpg"
                                }

                            }

                            geometry    Box {
                                size    14 2 6
                            }

                        }

                        Transform {
                            translation 0 0 3
                            rotation    0 0 1  1.57
                            children Shape {
                                appearance  Appearance {
                                    material    USE _mat1
                                    texture ImageTexture {
                                        url "texture/leather_white.jpg"
                                    }

                                }

                                geometry    Cylinder {
                                    height  15
                                    radius  1
                                }

                            }
                        }
                    ]
                }
            }
            ]
        }
    ]
}
```

Appendix A (Continued)

Door

```
#VRML V1.0 ascii
Separator {
MaterialBinding {
value OVERALL
}
Material {
ambientColor [
 0.796078 0.823529 0.937255
]
diffuseColor [
 0.796078 0.823529 0.937255
]
emissiveColor [
 0.063686 0.065882 0.074980
]
specularColor [
 0.756275 0.782353 0.890392
]
shininess [
 0.550000
]
transparency [
 0.000000
]
}
Coordinate3 {
point [
 0.000000 0.000000 0.000000, 0.000000 0.000000 2.200000, 0.000000
0.010000 0.000000, 0.000000 0.010000 2.200000, 0.900000 0.000000 -
0.000000,
 0.900000 0.000000 2.200000, 0.900000 0.010000 -0.000000, 0.900000
0.010000 2.200000
]
}
IndexedFaceSet {
coordIndex [
 2, 6, 0, -1, 0, 6, 4, -1, 3, 2, 1, -1,
 1, 2, 0, -1, 7, 3, 5, -1, 5, 3, 1, -1,
 6, 7, 4, -1, 4, 7, 5, -1, 3, 7, 2, -1,
 2, 7, 6, -1, 5, 1, 4, -1, 4, 1, 0, -1
]
}
}
```

Appendix A (Continued)

Laser

```
#VRML V2.0 utf8

#Created with V-Realm Builder v2.0
#Integrated Data Systems Inc.
#www.ids-net.com


Transform {
scale   60 60 60
    translation 0 0 0
    rotation    1 0 0 3.141592
        children Shape {
            appearance  Appearance {
                        material    DEF _mat1 Material {
                            ambientIntensity    0.2
                            diffuseColor    0 0 0
                            emissiveColor   1 0 0
                            shininess   0.2
                            specularColor   0 0 0
                            transparency    0
                        }
}
            geometry    Cone {
            bottomRadius    0.05
            height  3.8
        }

    }
}
```

Laser Mount

```
#VRML V2.0 utf8

#Created with V-Realm Builder v2.0
#Integrated Data Systems Inc.
#www.ids-net.com


Transform {
    translation -9.53674e-007 -3.21865e-006 0
    children Shape {
        appearance  Appearance {
            material    Material {
            }
```

```
            texture DEF Metal ImageTexture {
                url "texture/Metal.jpg"
            }

        }

        geometry    Box {
            size    20 20 20
        }

    }
}
```

Shelf

```
#VRML V2.0 utf8

#Created with V-Realm Builder v2.0
#Integrated Data Systems Inc.
#www.ids-net.com


Transform {
    translation -110.317 -144.058 -153.659
    rotation    -0.173679 0.966449 -0.189238  0.0127889
    scale   799.999 799.999 799.999
    children Shape {
        appearance  Appearance {
            material    Material {
                ambientIntensity    1
                diffuseColor    0.668235 0.564706 0.404706
                emissiveColor   0.167059 0.141176 0.101176
                shininess   0.31
                specularColor   0.668235 0.564706 0.404706
            }

            texture DEF Wood_Tan ImageTexture {
                url "texture/Wood_6.gif"
            }

        }

        geometry    IndexedFaceSet {
            color   Color {
                color   0.835294 0.705882 0.505882
            }

            coord   Coordinate {
                point   [ -0.044476 -0.045526 -0.02,
                          -0.044476 -0.045526 0.4,
```

```
                -0.044476 1.75447 -0.02,
                -0.044476 1.75447 0.4,
                -0.016654 -0.023677 0,
                -0.016654 -0.023677 0.4,
                -0.016654 0.393539 0,
                -0.016654 0.393539 0.4,
                -0.016654 0.413539 0,
                -0.016654 0.413539 0.4,
                -0.016654 0.908174 0,
                -0.016654 0.908174 0.4,
                -0.016654 0.928174 0,
                -0.016654 0.928174 0.4,
                -0.016654 1.35288 0,
                -0.016654 1.35288 0.4,
                -0.016654 1.37288 0,
                -0.016654 1.37288 0.4,
                -0.016654 1.73632 0,
                -0.016654 1.73632 0.4,
                0.927702 -0.023677 0,
                0.927702 -0.023677 0.4,
                0.927702 0.393539 0,
                0.927702 0.393539 0.4,
                0.927702 0.413539 0,
                0.927702 0.413539 0.4,
                0.927702 0.908174 0,
                0.927702 0.908174 0.4,
                0.927702 0.928174 0,
                0.927702 0.928174 0.4,
                0.927702 1.35288 0,
                0.927702 1.35288 0.4,
                0.927702 1.37288 0,
                0.927702 1.37288 0.4,
                0.927702 1.73632 0,
                0.927702 1.73632 0.4,
                0.955524 -0.045526 -0.02,
                0.955524 -0.045526 0.4,
                0.955524 1.75447 -0.02,
                0.955524 1.75447 0.4 ]
        }

        normal  Normal {
            vector  [ -1 0 0,
                      0 -1 0,
                      0 0 -1,
                      0 0 1,
                      0 1 0,
                      1 0 0 ]
        }

        colorPerVertex  FALSE
        normalPerVertex TRUE
        coordIndex  [ 7, 3, 5, -1, 5, 3, 1, -1,
```

```
                5, 1, 21, -1, 21, 1, 37, -1,
                21, 37, 23, -1, 13, 15, 3, -1,
                29, 13, 11, -1, 11, 13, 3, -1,
                11, 3, 9, -1, 9, 3, 7, -1,
                9, 7, 25, -1, 3, 35, 39, -1,
                39, 35, 33, -1, 7, 23, 25, -1,
                25, 23, 37, -1, 25, 37, 27, -1,
                27, 37, 39, -1, 11, 27, 29, -1,
                29, 27, 39, -1, 29, 39, 31, -1,
                31, 39, 33, -1, 31, 33, 15, -1,
                15, 33, 17, -1, 15, 17, 3, -1,
                3, 17, 19, -1, 3, 19, 35, -1,
                0, 1, 2, -1, 2, 1, 3, -1,
                36, 37, 0, -1, 0, 37, 1, -1,
                38, 39, 36, -1, 36, 39, 37, -1,
                2, 3, 38, -1, 38, 3, 39, -1,
                19, 17, 18, -1, 18, 17, 16, -1,
                17, 33, 16, -1, 16, 33, 32, -1,
                33, 35, 32, -1, 32, 35, 34, -1,
                35, 19, 34, -1, 34, 19, 18, -1,
                21, 23, 20, -1, 20, 23, 22, -1,
                23, 7, 22, -1, 22, 7, 6, -1,
                7, 5, 6, -1, 6, 5, 4, -1,
                5, 21, 4, -1, 4, 21, 20, -1,
                25, 27, 24, -1, 24, 27, 26, -1,
                27, 11, 26, -1, 26, 11, 10, -1,
                11, 9, 10, -1, 10, 9, 8, -1,
                9, 25, 8, -1, 8, 25, 24, -1,
                15, 13, 14, -1, 14, 13, 12, -1,
                13, 29, 12, -1, 12, 29, 28, -1,
                29, 31, 28, -1, 28, 31, 30, -1,
                31, 15, 30, -1, 30, 15, 14, -1,
                28, 30, 12, -1, 12, 30, 14, -1,
                10, 8, 26, -1, 26, 8, 24, -1,
                6, 4, 22, -1, 22, 4, 20, -1,
                32, 34, 16, -1, 16, 34, 18, -1,
                0, 2, 36, -1, 36, 2, 38, -1 ]
    normalIndex [ 3, 3, 3, -1, 3, 3, 3, -1,
                3, 3, 3, -1, 3, 3, 3, -1,
                3, 3, 3, -1, 3, 3, 3, -1,
                3, 3, 3, -1, 3, 3, 3, -1,
                3, 3, 3, -1, 3, 3, 3, -1,
                3, 3, 3, -1, 3, 3, 3, -1,
                3, 3, 3, -1, 3, 3, 3, -1,
                3, 3, 3, -1, 3, 3, 3, -1,
                3, 3, 3, -1, 3, 3, 3, -1,
                3, 3, 3, -1, 3, 3, 3, -1,
                3, 3, 3, -1, 3, 3, 3, -1,
                3, 3, 3, -1, 3, 3, 3, -1,
                3, 3, 3, -1, 3, 3, 3, -1,
                0, 0, 0, -1, 0, 0, 0, -1,
                1, 1, 1, -1, 1, 1, 1, -1,
```

Appendix A (Continued)

```
                        5, 5, 5, -1, 5, 5, 5, -1,
                        4, 4, 4, -1, 4, 4, 4, -1,
                        5, 5, 5, -1, 5, 5, 5, -1,
                        4, 4, 4, -1, 4, 4, 4, -1,
                        0, 0, 0, -1, 0, 0, 0, -1,
                        1, 1, 1, -1, 1, 1, 1, -1,
                        0, 0, 0, -1, 0, 0, 0, -1,
                        1, 1, 1, -1, 1, 1, 1, -1,
                        5, 5, 5, -1, 5, 5, 5, -1,
                        4, 4, 4, -1, 4, 4, 4, -1,
                        0, 0, 0, -1, 0, 0, 0, -1,
                        1, 1, 1, -1, 1, 1, 1, -1,
                        5, 5, 5, -1, 5, 5, 5, -1,
                        4, 4, 4, -1, 4, 4, 4, -1,
                        5, 5, 5, -1, 5, 5, 5, -1,
                        4, 4, 4, -1, 4, 4, 4, -1,
                        0, 0, 0, -1, 0, 0, 0, -1,
                        1, 1, 1, -1, 1, 1, 1, -1,
                        3, 3, 3, -1, 3, 3, 3, -1,
                        3, 3, 3, -1, 3, 3, 3, -1,
                        3, 3, 3, -1, 3, 3, 3, -1,
                        3, 3, 3, -1, 3, 3, 3, -1,
                        2, 2, 2, -1, 2, 2, 2, -1 ]
        }

    }
}
```

Sink Door

```
#VRML V2.0 utf8

#Created with V-Realm Builder v2.0
#Integrated Data Systems Inc.
#www.ids-net.com


Transform {
    scale   800 800 800
    children Shape {
        appearance  Appearance {
            material    Material {
                ambientIntensity    1
                diffuseColor    0.721569 0.646275 0.404706
                emissiveColor   0.180392 0.161569 0.101176
                shininess   0.31
                specularColor   0.721569 0.646275 0.404706
```

```
           }

            texture DEF Wood_Tan ImageTexture {
                url "texture/Wood_6.gif"
            }

        }

        geometry    IndexedFaceSet {
            color   Color {
                color   0.901961 0.807843 0.505882
            }

            coord   Coordinate {
                point   [ 0 0 0,
                          0 0 0.02,
                          0 1 0,
                          0 1 0.02,
                          0.553993 0.46 0.02,
                          0.553993 0.46 0.05,
                          0.553993 0.47 0.02,
                          0.553993 0.47 0.04,
                          0.553993 0.51 0.02,
                          0.553993 0.51 0.04,
                          0.553993 0.52 0.02,
                          0.553993 0.52 0.05,
                          0.563994 0.46 0.02,
                          0.563994 0.46 0.05,
                          0.563994 0.47 0.02,
                          0.563994 0.47 0.04,
                          0.563994 0.51 0.02,
                          0.563994 0.51 0.04,
                          0.563994 0.52 0.02,
                          0.563994 0.52 0.05,
                          0.6 0 0,
                          0.6 0 0.02,
                          0.6 1 0,
                          0.6 1 0.02 ]
            }

            normal  Normal {
                vector  [ -1 0 0,
                          0 -1 0,
                          0 0 -1,
                          0 0 1,
                          0 1 0,
                          1 0 0 ]
            }

            colorPerVertex  FALSE
            normalPerVertex TRUE
```

Appendix A (Continued)

```
            coordIndex  [ 18, 23, 10, -1, 10, 23, 3, -1,
                      10, 3, 8, -1, 6, 14, 16, -1,
                      16, 8, 6, -1, 6, 8, 3, -1,
                      6, 3, 4, -1, 4, 3, 1, -1,
                      4, 1, 12, -1, 12, 1, 21, -1,
                      12, 21, 14, -1, 14, 21, 23, -1,
                      14, 23, 16, -1, 16, 23, 18, -1,
                      1, 3, 0, -1, 0, 3, 2, -1,
                      21, 1, 20, -1, 20, 1, 0, -1,
                      23, 21, 22, -1, 22, 21, 20, -1,
                      3, 23, 2, -1, 2, 23, 22, -1,
                      22, 20, 2, -1, 2, 20, 0, -1,
                      6, 4, 7, -1, 7, 4, 5, -1,
                      7, 5, 9, -1, 9, 5, 11, -1,
                      9, 11, 8, -1, 8, 11, 10, -1,
                      13, 5, 12, -1, 12, 5, 4, -1,
                      16, 18, 17, -1, 17, 18, 19, -1,
                      17, 19, 15, -1, 15, 19, 13, -1,
                      15, 13, 14, -1, 14, 13, 12, -1,
                      7, 15, 6, -1, 6, 15, 14, -1,
                      11, 19, 10, -1, 10, 19, 18, -1,
                      17, 9, 16, -1, 16, 9, 8, -1,
                      7, 9, 15, -1, 15, 9, 17, -1,
                      13, 19, 5, -1, 5, 19, 11, -1 ]
             normalIndex [ 3, 3, 3, -1, 3, 3, 3, -1,
                      3, 3, 3, -1, 3, 3, 3, -1,
                      3, 3, 3, -1, 3, 3, 3, -1,
                      3, 3, 3, -1, 3, 3, 3, -1,
                      3, 3, 3, -1, 3, 3, 3, -1,
                      3, 3, 3, -1, 3, 3, 3, -1,
                      0, 0, 0, -1, 0, 0, 0, -1,
                      1, 1, 1, -1, 1, 1, 1, -1,
                      5, 5, 5, -1, 5, 5, 5, -1,
                      4, 4, 4, -1, 4, 4, 4, -1,
                      2, 2, 2, -1, 2, 2, 2, -1,
                      0, 0, 0, -1, 0, 0, 0, -1,
                      0, 0, 0, -1, 0, 0, 0, -1,
                      0, 0, 0, -1, 0, 0, 0, -1,
                      1, 1, 1, -1, 1, 1, 1, -1,
                      5, 5, 5, -1, 5, 5, 5, -1,
                      5, 5, 5, -1, 5, 5, 5, -1,
                      5, 5, 5, -1, 5, 5, 5, -1,
                      4, 4, 4, -1, 4, 4, 4, -1,
                      4, 4, 4, -1, 4, 4, 4, -1,
                      1, 1, 1, -1, 1, 1, 1, -1,
                      2, 2, 2, -1, 2, 2, 2, -1,
                      3, 3, 3, -1, 3, 3, 3, -1 ]
        }

    }
}
```

Appendix A (Continued)

Table

```
#VRML V2.0 utf8

#Created with V-Realm Builder v2.0
#Integrated Data Systems Inc.
#www.ids-net.com


Group {

    children [

        NavigationInfo {
        }
        Transform {
        scale   60 60 60

            children [
                        Shape {

            appearance  Appearance {
                material    DEF _mat1 Material {
                    ambientIntensity    0.2
                    diffuseColor    0.25 0.15 0.1
                    emissiveColor   0 0 0
                    shininess   0.2
                    specularColor   0 0 0
                    transparency    0
                }

                texture DEF Wood_Brown ImageTexture {
                    url "texture/Wood_5.jpg"
                }

            }


            geometry    Cylinder {

                height  0.5
                radius  8
            }

        }
]}
        Transform {
```

Appendix A (Continued)

```
translation 0 -1 0
scale   60 60 60
children [
Shape {
    appearance  Appearance {
        material    USE _mat1
        texture DEF Wood_Brown ImageTexture {
            url "texture/Wood_5.jpg"
        }

    }

    geometry    Cylinder {
        height  1
        radius  1
    }

}

Transform {
    translation 0 -0.5 0
    children [
        Shape {
            appearance  Appearance {
                material    USE _mat1
                texture DEF Wood_Brown ImageTexture {
                    url "texture/Wood_5.jpg"
                }

            }

            geometry                                Sphere {
            radius  1
            }

        }

        Transform {
            translation 0 -1.5 0
            children [
            Shape {
                appearance  Appearance {
                    material    USE _mat1
                    texture DEF Wood_Brown ImageTexture {
                        url "texture/Wood_5.jpg"
                    }

                }
```

Appendix A (Continued)

```
                            geometry
Sphere {
                                radius  1
                            }

                        }

                        Transform {
                            translation 0 -1.5 0
                            children [
                                Shape {
                                    appearance  Appearance {
                                        material    USE _mat1
                                        texture DEF Wood_Brown
ImageTexture {
                                            url "texture/Wood_5.jpg"
                                        }

                                    }

                                    geometry
Sphere {
                                    radius  1
                                    }

                                }

                                Transform {
                                    translation 0 -1.5 0
                                    children [
                                    Shape {
                                        appearance  Appearance {
                                            material    USE _mat1
                                            texture DEF Wood_Brown
ImageTexture {
                                                url
"texture/Wood_5.jpg"
                                            }

                                        }

                                        geometry
Sphere {
                                            radius  1
                                        }

                                    }

                                    Transform {
                                        translation 0 -1.5 0
```

Appendix A (Continued)

```
{
_mat1
Wood_Brown ImageTexture {
"texture/Wood_5.jpg"
Appearance {
_mat1
Wood_Brown ImageTexture {
"texture/Wood_5.jpg"
2
Appearance {
children [
      Shape {
          appearance  Appearance
              material    USE
              texture DEF
                  url
              }
          }
          geometry    Cylinder {
              height  3
              radius  1
          }
      }
      Transform {
          translation 0 -1 0
          children [
          Shape {
              appearance
                  material    USE
                  texture DEF
                      url
                  }
              }
              geometry    Cone {
                  bottomRadius
                  height  3
              }
          }
          Transform {
              translation 0 -1 0
              children [
                  Shape {
                      appearance
```

Appendix A (Continued)

```
material    USE _mat1
DEF Wood_Brown ImageTexture {
"texture/Wood_5.jpg"


Cone {
bottomRadius    4
1.5


0 -0.75 0
Shape {
appearance  Appearance {
material    USE _mat1
texture DEF Wood_Brown ImageTexture {
url "texture/Wood_5.jpg"


geometry    Cylinder {
height  0.25
radius  5.5
                                                    texture
                                                        url
                                                    }
                                                }
                                                geometry
                                                height
                                                }
                                            }
                                            Transform {
                                                translation
                                                children
                                                    }
                                                }
                                                }
                                            }
                                        ]
                                    }
                                    ]
```

Appendix A (Continued)

```
                                        }
                                    ]
                                }
                                ]
                            }
                        ]
                    }
                    ]
                }
            ]
        }
        ]
    }
}
```

Wall

```
#VRML V2.0 utf8

#Created with V-Realm Builder v2.0
#Integrated Data Systems Inc.
#www.ids-net.com


Collision {
    children []
}
Group {
    children [
        NavigationInfo {
        }

        Transform {
            translation 1.5 0 0
            children Transform {
                scale   201 201 201
                children Shape {
                    appearance  Appearance {
                        material    Material {
                            ambientIntensity    1
                            diffuseColor    0.9 0.767329 0.619635
                            shininess   1
                        }

                        texture NULL
                    }

                    geometry    Box {
```

Appendix A (Continued)

```
                        size    14.8 7.1 0.25
                    }

                }
            }
        }
    ]
}
```

Appendix B. Matlab Functions

WMRA_final_orientation

```
% This function simulates the wmra final orientation acording to the
needs
% of the task. the final angle to be rotated is an input

% Function Declaration:
function WMRA_final_orientation(ini, vr, ml, arm, Tiwc, qi,ang)

% Closing the Arm library and Matlab Graphics Animation and Virtual
Reality Animation and Plots windows:
if ini==3
    if arm==1
        try
            WMRA_ARM_Motion(ini, 0, 0, 0);
        end
    end
    if vr==1
        try
            WMRA_VR_Animation2(ini, 0, 0);
        end
    end
    if ml==1
        try
            WMRA_ML_Animation2(ini, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
        end
    end
    return;
end

% Defining the used conditions:
qd=qi(1:7);% Final joint angles
%qd=[pi/2;pi/2;0;pi/2;pi/2;pi/2;0];   % Final joint angles (Ready
Position).
ts=10;       % (5 or 10 or 20) Simulation time to move the arm from any
position to the ready position.
n=100;      % Number of time steps.
dt=ts/n;    % The time step to move the arm from any position to the
ready position.
dqw=(ang)/(0.5*n+5);
%dq=(qd-qi(1:7))/(0.5*n+5);
% Initializing the physical Arm:
if arm==1
    WMRA_ARM_Motion(ini, 2, [qi;0], dt);
    ddt=0;
end

% Initializing Virtual Reality Animation:
```

```
if vr==1
    WMRA_VR_Animation2(ini, Tiwc, qi);

end

% Initializing Robot Animation in Matlab Graphics:
if ml==1
    % Inputting the D-H Parameters in a Matrix form:
    DH=WMRA_DH(qi);

    % Calculating the transformation matrices of each link:

T01=WMRA_rotx(DH(1,1))*WMRA_transl(DH(1,2),0,0)*WMRA_rotz(DH(1,4))*WMRA
_transl(0,0,DH(1,3));

T12=WMRA_rotx(DH(2,1))*WMRA_transl(DH(2,2),0,0)*WMRA_rotz(DH(2,4))*WMRA
_transl(0,0,DH(2,3));

T23=WMRA_rotx(DH(3,1))*WMRA_transl(DH(3,2),0,0)*WMRA_rotz(DH(3,4))*WMRA
_transl(0,0,DH(3,3));

T34=WMRA_rotx(DH(4,1))*WMRA_transl(DH(4,2),0,0)*WMRA_rotz(DH(4,4))*WMRA
_transl(0,0,DH(4,3));

T45=WMRA_rotx(DH(5,1))*WMRA_transl(DH(5,2),0,0)*WMRA_rotz(DH(5,4))*WMRA
_transl(0,0,DH(5,3));

T56=WMRA_rotx(DH(6,1))*WMRA_transl(DH(6,2),0,0)*WMRA_rotz(DH(6,4))*WMRA
_transl(0,0,DH(6,3));

T67=WMRA_rotx(DH(7,1))*WMRA_transl(DH(7,2),0,0)*WMRA_rotz(DH(7,4))*WMRA
_transl(0,0,DH(7,3));
    % Calculating the Transformation Matrix of the initial and desired
arm positions:
    Ti=Tiwc*T01*T12*T23*T34*T45*T56*T67;
    Td=Tiwc*WMRA_q2T(qd);
    WMRA_ML_Animation2(ini, Ti, Td, Tiwc, T01, T12, T23, T34, T45, T56,
T67);
end

% Check for the shortest route:
diff=qd-qi(1:7);
for i=1:7
    if diff(i) > pi
        diff(i)=diff(i)-2*pi;
    elseif diff(i) < (-pi)
        diff(i)=diff(i)+2*pi;
    end
end

% Joint angle change at every time step.
dq=[diff/n;0;0];
```

```
% Initialization:
qo=qi;
tt=0;

while tt <= (ts-dt)
    % Starting a timer:
    tic;

    % Calculating the new Joint Angles:
    qn=qo+dq;

    % Updating the physical Arm:
    if arm==1
        ddt=ddt+dt;
        if ddt>=0.5 || tt>=(ts-dt)
            WMRA_ARM_Motion(2, 1, [qn;0], ddt);
            ddt=0;
        end
    end

    % Updating Virtual Reality Animation:
    if vr==1
        WMRA_VR_Animation2(2, Tiwc, qn);
    end

    % Updating Matlab Animation:
    if ml==1
        % Calculating the new Transformation Matrix:

T1a=WMRA_rotx(DH(1,1))*WMRA_transl(DH(1,2),0,0)*WMRA_rotz(qn(1))*WMRA_t
ransl(0,0,DH(1,3));

T2a=WMRA_rotx(DH(2,1))*WMRA_transl(DH(2,2),0,0)*WMRA_rotz(qn(2))*WMRA_t
ransl(0,0,DH(2,3));

T3a=WMRA_rotx(DH(3,1))*WMRA_transl(DH(3,2),0,0)*WMRA_rotz(qn(3))*WMRA_t
ransl(0,0,DH(3,3));

T4a=WMRA_rotx(DH(4,1))*WMRA_transl(DH(4,2),0,0)*WMRA_rotz(qn(4))*WMRA_t
ransl(0,0,DH(4,3));

T5a=WMRA_rotx(DH(5,1))*WMRA_transl(DH(5,2),0,0)*WMRA_rotz(qn(5))*WMRA_t
ransl(0,0,DH(5,3));

T6a=WMRA_rotx(DH(6,1))*WMRA_transl(DH(6,2),0,0)*WMRA_rotz(qn(6))*WMRA_t
ransl(0,0,DH(6,3));

T7a=WMRA_rotx(DH(7,1))*WMRA_transl(DH(7,2),0,0)*WMRA_rotz(qn(7))*WMRA_t
ransl(0,0,DH(7,3));
```

Appendix B (continued)

```
        WMRA_ML_Animation(2, Ti, Td, Tiwc, T1a, T2a, T3a, T4a, T5a,
T6a, T7a);
    end

    % Updating the old values with the new values for the next
iteration:
    Tiwc=Tiwc*WMRA_rotz(dqw);
    qn(9)=qn(9)+dqw;
    qo=qn;
    tt=tt+dt;

    % Pausing for the speed sync:
    pause(dt-toc);

end
```

WMRA_Main_Both

```
% This is a simplified version of the Main program to acomplish a
subtask
% with motion control. most of the options are prespecified and will
not be
% changed by the user

% Declaring the global variables to be used for the touch screen
control:
global VAR_DX
global VAR_SCREENOPN
global dHo

% Defining used parameters:
d2r=pi/180; % Conversions from Degrees to Radians.
r2d=180/pi; % Conversions from Radians to Degrees.

% Reading the Wheelchair's constant dimentions, all dimentions are
converted in millimeters:
L=WMRA_WCD;
radi=1000;
e=1;
% User input prompts:

%choice000 = input('\n Choose what to control: \n For combined
Wheelchair and Arm control, press "1", \n For Arm only control, press
"2", \n For Wheelchair only control, press "3". \n','s');
choice000 = 1;
    WCA=1;
%choice00000 = input('\n Choose whose frame to base the control on: \n
For Ground Frame, press "1", \n For Wheelchair Frame, press "2", \n For
Gripper Frame, press "3". \n','s');
choice00000=1;
```

Appendix B (continued)

```
    coord=1;
%choice0000 = input('\n Choose the cartesian coordinates to be
controlled: \n For Position and Orientation, press "1", \n For Position
only, press "2". \n','s');
choice0000 =1;
    cart=1;
%choice5 = input('\n Please enter the desired optimization method: (1=
SR-I & WLN, 2= P-I & WLN, 3= SR-I & ENE, 4= P-I & ENE) \n','s');
choice5 =1;
    optim=1;
%choice50 = input('\n Do you want to include Joint Limit Avoidance? (1=
Yes, 2= No) \n','s');
choice50 =1;
    JLA=1;
%choice500 = input('\n Do you want to include Joint Limit/Velocity and
Obstacle Safety Stop? (1= Yes, 2= No) \n','s');
choice500 = 1;
        JLO=1;
%choice0 = input('\n Choose the control mode: \n For circle control,
press "6", \n For position control, press "1", \n For velocity control,
press "2", \n For SpaceBall control, press "3", \n For Psychology Mask
control, press "4", \n For Touch Screen control, press "5",\n For
Switch, press "7". \n','s');
choice0 ='1';
if choice0=='1'
    cont=1;
    %Td = input('\n Please enter the transformation matrix of the
desired position and orientation from the control-based frame \n (e.g.
[0 0 1 1455;-1 0 0 369;0 -1 0 999; 0 0 0 1]) \n');
    %Td=[0 0 1 1955;-1 0 0 -3000;0 -1 0 800; 0 0 0
1]*WMRA_rotx(90)*WMRA_roty(70);
    Td=[0 0 1 1955;-1 0 0 -1169;0 -1 0 999; 0 0 0 1] *WMRA_rotx(-
pi/6)*WMRA_roty(pi/4);
    %v = input('\n Please enter the desired linear velocity of the
gripper in mm/s (e.g. 50) \n');
    v=50;
    %choice00 = input('\n Chose the Trajectory generation function: \n
Press "1" for a Polynomial function with Blending, or \n press "2" for
a Polynomial function without Blending, or \n press "3" for a Linear
function.\n','s');
    choice00=1;
        trajf=1;
   elseif choice0=='7'
    cont=1;
    Td = [0 0 1 1455;-1 0 0 369;0 -1 0 999; 0 0 0 1];
    v = input('\n Please enter the desired linear velocity of the
gripper in mm/s (e.g. 50) \n');
    choice00 = input('\n Chose the Trajectory generation function: \n
Press "1" for a Polynomial function with Blending, or \n press "2" for
a Polynomial function without Blending, or \n press "3" for a Linear
function.\n','s');
    if choice00=='2'
        trajf=2;
```

```
    elseif choice00=='3'
        trajf=3;
    else
        trajf=1;
    end
elseif choice0=='2'
    cont=2;
    ts = input('\n Please enter the desired simulation time in seconds
(e.g. 2) \n');
    if cart==2
        Vd = input('\n Please enter the desired 3x1 cartesian velocity
vector of the gripper (in mm/s) (e.g. [70;70;-70]) \n');
    else
        Vd = input('\n Please enter the desired 6x1 cartesian velocity
vector of the gripper (in mm/s, radians/s) (e.g. [70;70;-
70;0.001;0.001;0.001]) \n');
    end
elseif choice0=='3'
    cont=3;
    % Space Ball will be used for control.
    v = input('\n Please enter the desired linear velocity of the
gripper in mm/s (e.g. 50) \n');
elseif choice0=='4'
    cont=4;
    % BCI 2000 Psychology Mask will be used for control.
    v = input('\n Please enter the desired linear velocity of the
gripper in mm/s (e.g. 50) \n');
    port1 = input('\n Please enter the desired port number (e.g. 19711)
\n');
    choice00 = input('\n Chose the Trajectory generation function: \n
Press "1" for a Polynomial function with Blending, or \n press "2" for
a Polynomial function without Blending, or \n press "3" for a Linear
function, or \n press "4" for a Circular Polynomial function with
Blending. \n','s');
elseif choice0=='6'
    choice0=6;
    cont=6;
    %radi = input('\n Please enter the radius of the cirle in mm (e.g.
1000) \n');
    radi=1000;
    %e = input('\n If the door opens to the left press "1".\n If it
opens to the right press "2" \n');
    e=2;
    %v = input('\n Please enter the desired linear velocity of the
gripper in mm/s (e.g. 50) \n');
    v=50;
    trajf=4;
else
    cont=5;
    % Touch Screen will be used for control.
    v = input('\n Please enter the desired linear velocity of the
gripper in mm/s (e.g. 50) \n');
end
```

```
choice1 = input('\n Choose animation type or no animation: \n For
Virtual Reality Animation, press "1", \n For Matlab Graphics Animation,
press "2", \n For BOTH Animations, press "3", \n For NO Animation,
press "4". \n','s');
if choice1=='2'
    vr = 0; ml = 1;
elseif choice1=='3'
    vr = 1; ml = 1;
elseif choice1=='4'
    vr = 0; ml = 0;
else
    vr = 1; ml = 0;
end

%choice10 = input('\n Would you like to run the actual WMRA? \n For
yes, press "1", \n For no, press "2". \n','s');
choice10='2';
if choice10=='1'
    arm=1;
else
    arm=0;
end
%choice2 = input('\n Press "1" if you want to start at the "ready"
position, \n or press "2" if you want to enter the initial joint
angles. \n','s');
choice2='1';
if choice2=='2'
    qi = input('\n Please enter the arms initial angles vector in
radians (e.g. [pi/2;pi/2;0;pi/2;pi/2;pi/2;0]) \n');
    WCi = input('\n Please enter the initial x,y position and z
orientation of the WMRA base from the ground base in millimeters and
radians (e.g. [200;500;0.3]) \n');
    ini=0;
else
    qi=[90;90;0;90;90;90;0]*d2r;
    WCi=[0;0;0];
    ini=0;
    if vr==1 || ml==1 || arm==1
        %choice3 = input('\n Press "1" if you want to include "park" to
"ready" motion, \n or press "2" if not. \n','s');
        choice3= '1';
        if choice3=='2'
            ini=0;
        else
            ini=1;
        end
    end
end
%choice4 = input('\n Press "1" if you do NOT want to plot the
simulation results, \n or press "2" if do. \n','s');
choice4 ='2';
if choice4=='2'
```

```
    plt=2;
else
    plt=1;
end

% Calculating the Transformation Matrix of the initial position of the
WMRA's base:
Tiwc=WMRA_p2T(WCi(1),WCi(2),WCi(3));

% Calculating the initial Wheelchair Variables:
qiwc=[sqrt(WCi(1)^2+WCi(2)^2);WCi(3)];

% Calculating the initial transformation matrices:
[Ti, Tia, Tiwc, T01, T12, T23, T34, T45, T56, T67]=WMRA_Tall(1, qi,
[0;0], Tiwc);

if cont==1
    % Calculating the linear distance from the initial position to the
desired position and the linear velocity:
    if coord==2
        D=sqrt( (Td(1,4)-Tia(1,4))^2 + (Td(2,4)-Tia(2,4))^2 + (Td(3,4)-
Tia(3,4))^2);
    elseif coord==3
        D=sqrt( (Td(1,4))^2 + (Td(2,4))^2 + (Td(3,4))^2);
    else
        D=sqrt( (Td(1,4)-Ti(1,4))^2 + (Td(2,4)-Ti(2,4))^2 + (Td(3,4)-
Ti(3,4))^2);
    end
    % Calculating the number of iteration and the time increment (delta
t) if the linear step increment of the tip is 1 mm:
    dt=0.05;    % Time increment in seconds.
    total_time=D/v;     % Total time of animation.
    n=round(total_time/dt); % Number of iterations rounded up.
    dt=total_time/n;    % Adjusted time increment in seconds.
    % Calculating the Trajectory of the end effector, and once the
trajectory is calculated, we should redefine "Td" based on the ground
frame:
    if coord==2
        Tt=WMRA_traj(radi,trajf, Tia, Td, n+1);
        Td=Tiwc*Td;
    elseif coord==3
        Tt=WMRA_traj(radi,trajf, eye(4), Td, n+1);
        Td=Ti*Td;
    else
        Tt=WMRA_traj(radi,trajf, Ti, Td, n+1);
    end
elseif cont==2
    % Calculating the number of iterations and the time increment
(delta t) if the linear step increment of the gripper is 1 mm:
    dt=0.05;    % Time increment in seconds.
    total_time=ts;     % Total time of animation.
    n=round(total_time/dt); % Number of iterations rounded up.
```

```
    dt=total_time/n;    % Adjusted time increment in seconds.
    dx=Vd*dt;
    Td=Ti;
elseif cont==3
    WMRA_exit(); % This is to stop the simulation in SpaceBall control
when the user presses the exit key.
    dt=0.05;
    dx=v*dt*[spdata1(3)/20 ; -spdata1(1)/40 ; spdata1(2)/30 ;
spdata1(6)/1500 ; -spdata1(4)/900 ; spdata1(5)/1300];
    dg=spdata1(7);
    Td=Ti;
    n=1;
elseif cont==4
    WMRA_exit(); % This is to stop the simulation in Psychology Mask
control when the user presses the exit key.
    dt=0.05;
    dx=v*dt*WMRA_psy(port1);
    dg=dx(7);
    dx=dx(1:6);
    Td=Ti;
    n=1;
elseif cont==6
    % Calculating the desired transformation matrix based on the
radius:
    Tdoor=Ti;
    Tdoor(1,4)=Ti(1,4)+radi/2;
    Tdoor(2,4)=Ti(2,4)+radi/2;
    Td=Tdoor;
    Td(1,4)=Td(1,4)-radi;
    Td(2,4)=Td(2,4)-((-1)^e)*radi;
    % Calculating the circular distance from the initial position to
the desired position and the linear velocity:
    D=0.5*pi*radi;
    % Calculating the number of iteration and the time increment (delta
t) if the linear step increment of the tip is 1 mm:
    dt=0.05;    % Time increment in seconds.
    total_time=D/v;     % Total time of animation.
    n=round(total_time/dt); % Number of iterations rounded up.
    dt=total_time/n;    % Adjusted time increment in seconds.
    % Calculating the Trajectory of the end effector, and once the
trajectory is calculated, we should redefine "Td" based on the ground
frame:
    if coord==2
        Tt=WMRA_traj(radi,trajf, Tia, Td, n+1,e);
        Td=Tiwc*Td;
    elseif coord==3
        Tt=WMRA_traj(radi,trajf, eye(4), Td, n+1,e);
        Td=Ti*Td;
    else
        Tt=WMRA_traj(radi,trajf, Ti, Td, n+1,e);
    end
else
```

```
    WMRA_screen('0');   % This is to start the screen controls.
Argument: '0'=BACK button disabled, '1'=BACK button enabled.
    dt=0.05;
    dx=v*dt*VAR_DX(1:6);
    dg=VAR_DX(7);
    Td=Ti;
    n=1;
end

% Initializing the joint angles, the Transformation Matrix, and time:
dq=zeros(9,1);
dg=0;
qo=[qi;qiwc];
To=Ti;
Toa=Tia;
Towc=Tiwc;
tt=0;
i=1;
dHo=[0;0;0;0;0;0;0];

% Initializing the WMRA:
if ini==0   % When no "park" to "ready" motion required.
    % Initializing Virtual Reality Animation:
    if vr==1
        WMRA_VR_Animation(1, Towc, qo);
    end
    % Initializing Robot Animation in Matlab Graphics:
    if ml==1
        WMRA_ML_Animation(1, To, Td, Towc, T01, T12, T23, T34, T45,
T56, T67);
    end
    % Initializing the Physical Arm:
    if arm==1
        WMRA_ARM_Motion(1, 2, [qo;dg], 0);
        ddt=0;
    end
elseif ini==1 && (vr==1 || ml==1 || arm==1)  % When "park" to "ready"
motion is required.
    WMRA_park2ready(1, vr, ml, arm, Towc, qo(8:9));
    if arm==1
        ddt=0;
    end
end

% Re-Drawing the Animation:
if vr==1 || ml==1
    drawnow;
end

% Starting a timer:
tic
ANG=zeros(1,n+1);
```

Appendix B (continued)

```
while i<=(n+1)

%Calculating the angle between the trajectory and the wheelchair's x
axis
the=WMRA_opt_angle(Td,Ti,Towc,L,T01, T12, T23, T34, T45, T56, T67);
ANG(i)=the;
% Starting the Iteration Loop:

    % Calculating the 6X7 Jacobian of the arm in frame 0:
    [Joa,detJoa]=WMRA_J07(T01, T12, T23, T34, T45, T56, T67);

    % Calculating the 6X2 Jacobian based on the WMRA's base in the
ground frame:
    phi=atan2(Towc(2,1),Towc(1,1));
    Jowc=WMRA_Jga(1, phi, Toa(1:2,4));

    % Changing the Jacobian reference frame based on the choice of
which coordinates frame are referenced in the Cartesian control:
    % coord=1 for Ground Coordinates Control.
    % coord=2 for Wheelchair Coordinates Control.
    % coord=3 for Gripper Coordinates Control.
    if coord==2
        Joa=Joa;
        Jowc=[Towc(1:3,1:3)' zeros(3); zeros(3) Towc(1:3,1:3)']*Jowc;
    elseif coord==3
        Joa=[Toa(1:3,1:3)' zeros(3); zeros(3) Toa(1:3,1:3)']*Joa;
        Jowc=[To(1:3,1:3)' zeros(3); zeros(3) To(1:3,1:3)']*Jowc;
    elseif coord==1
        Joa=[Towc(1:3,1:3) zeros(3); zeros(3) Towc(1:3,1:3)]*Joa;
        Jowc=Jowc;
    end

    % Calculating the 3X9 or 6X9 augmented Jacobian of the WMRA system
based on the ground frame:
    if cart==2
        Joa=Joa(1:3,1:7);
        detJoa=sqrt(det(Joa*Joa'));
        Jowc=Jowc(1:3,1:2);
        Jo=[Joa Jowc];
        detJo=sqrt(det(Jo*Jo'));
    else
        Jo=[Joa Jowc];
        detJo=sqrt(det(Jo*Jo'));
    end

    % Finding the Cartesian errors of the end effector:
    if cont==1 || cont==6
        % Calculating the Position and Orientation errors of the end
effector, and the rates of motion of the end effector:
        if coord==2
            invTowc=[Towc(1:3,1:3)' , -Towc(1:3,1:3)'*Towc(1:3,4);0 0 0
1];
```

Appendix B (continued)

```
            Ttnew=invTowc*Tiwc*Tt(:,:,i);
            dx=WMRA_delta(Toa, Ttnew);
        elseif coord==3
            invTo=[To(1:3,1:3)' , -To(1:3,1:3)'*To(1:3,4);0 0 0 1];
            Ttnew=invTo*Ti*Tt(:,:,i);
            dx=WMRA_delta(eye(4), Ttnew);
        else
            dx=WMRA_delta(To, Tt(:,:,i));
        end
    elseif cont==2

    elseif cont==3
        dx=v*dt*[spdata1(3)/20 ; -spdata1(1)/40 ; spdata1(2)/30 ;
spdata1(6)/1500 ; -spdata1(4)/900 ; spdata1(5)/1300];
        dg=spdata1(7);
    elseif cont==4
        dx=v*dt*WMRA_psy(port1);
        dg=dx(7);
        dx=dx(1:6);
    else
        dx=v*dt*VAR_DX(1:6);
        dg=VAR_DX(7);
    end

    % Changing the order of Cartesian motion in the case when gripper
reference frame is selected for control with the screen or psy or
SpaceBall interfaces:
    if coord==3 && cont>=3
        dx=[-dx(2);-dx(3);dx(1);-dx(5);-dx(6);dx(4)];
    end

    if cart==2
        dx=dx(1:3);
    end

    % Calculating the resolved rate with optimization:
    % Index input values for "optim": 1= SR-I & WLN, 2= P-I & WLN, 3=
SR-I & ENE, 4= P-I & ENE:
    if WCA==2
        dq=WMRA_Opt(optim, JLA, JLO, Joa, detJoa, dq(1:7), dx, dt,
qo,the,n,choice0);
        dq=[dq;0;0];
    elseif WCA==3
        dq=WMRA_Opt(optim, JLA, JLO, Jowc, 1, dq(8:9), dx(1:2), dt,
1,the,n,choice0);
        dq=[0;0;0;0;0;0;0;dq];
    else
        dq=WMRA_Opt(optim, JLA, JLO, Jo, detJo, dq, dx, dt,
qo,the,n,choice0);
    end

    % Calculating the new Joint Angles:
```

```
    qn=qo+dq;

    % Calculating the new Transformation Matrices:
    [Tn, Tna, Tnwc, T01, T12, T23, T34, T45, T56, T67]=WMRA_Tall(2, qn,
dq(8:9), Towc);

    % A safety condition function to stop the joints that may cause
colision of the arm with itself, the wheelchair, or the human user:
    if JLO==1 && WCA~=3
        dq(1:7)=WMRA_collide(dq(1:7), T01, T12, T23, T34, T45, T56,
T67);
        % Re-calculating the new Joint Angles:
        qn=qo+dq;
        % Re-calculating the new Transformation Matrices:
        [Tn, Tna, Tnwc, T01, T12, T23, T34, T45, T56, T67]=WMRA_Tall(2,
qn, dq(8:9), Towc);
    end

    % Saving the plot data in case plots are required:
    if plt==2
        WMRA_Plots(1, L, r2d, dt, i, tt, qn, dq, Tn, Tnwc, detJoa,
detJo);
    end

    % Updating Physical Arm:
    if arm==1
        ddt=ddt+dt;
        if ddt>=0.5 || i>=(n+1)
            WMRA_ARM_Motion(2, 1, [qn;dg], ddt);
            ddt=0;
        end
    end

    % Updating Virtual Reality Animation:
    if vr==1
        WMRA_VR_Animation(2, Tnwc, qn);
    end

    % Updating Matlab Graphics Animation:
    if ml==1
        WMRA_ML_Animation(2, Ti, Td, Tnwc, T01, T12, T23, T34, T45,
T56, T67);
    end

    % Re-Drawing the Animation:
    if vr==1 || ml==1
        drawnow;
    end

    % Updating the old values with the new values for the next
iteration:
    qo=qn;
```

```
    To=Tn;
    Toa=Tna;
    Towc=Tnwc;
    tt=tt+dt;
    i=i+1;

    % Stopping the simulation when the exit button is pressed:
    if cont==3 || cont==4 || cont==5
        if (VAR_SCREENOPN == 1)
            n=n+1;
        else
            break
        end
    end

    % Delay to comply with the required speed:
    if toc < tt
        pause(tt-toc);
    end
end

% Reading the elapsed time and printing it with the simulation time:
if cont==1 || cont==2 || cont==6, fprintf('\nSimula. time is %6.6f
seconds.\n' , total_time); end
toc

% Plotting:


if vr==1 || ml==1 || arm==1
    %Orienting the Wheelchair to a desired final orientation
    choice9 = input('\n Do you want to rotate the wheelchair? \n Press
"1" for Yes, or press "2" for No. \n','s');
    if choice9=='1'
        ang = input('\n enter the desired angle in radians \n');
      WMRA_final_orientation(2, vr, ml, arm, Tnwc, qn,ang)
    else
        ang=0;
    end
    if plt==2
    WMRA_Plots(2, L, r2d, dt, i, tt, qn, dq, Tn, Tnwc, detJoa, detJo);
    end

    % Going back to the ready position:
    %choice6 = input('\n Do you want to go back to the "ready"
position? \n Press "1" for Yes, or press "2" for No. \n','s');
    choice6 ='2';
    if choice6=='1'
        WMRA_any2ready(2, vr, ml, arm, Tnwc, qn);
        % Going back to the parking position:
        choice7 = input('\n Do you want to go back to the "parking"
position? \n Press "1" for Yes, or press "2" for No. \n','s');
```

```
        if choice7=='1'
            WMRA_ready2park(2, vr, ml, arm, Tnwc, qn(8:9));
        end
    end

    % Closing the Arm library and Matlab Graphics Animation and Virtual
Reality Animation and Plots windows:
    %choice8 = input('\n Do you want to close all simulation windows
and arm controls? \n Press "1" for Yes, or press "2" for No. \n','s');
    choice8 ='2';
    if choice8=='1'
        if arm==1
            WMRA_ARM_Motion(3, 0, 0, 0);
        end
        if vr==1
            WMRA_VR_Animation(3, 0, 0);
        end
        if ml==1
            WMRA_ML_Animation(3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
        end
        if plt==2
            close
(figure(2),figure(3),figure(4),figure(5),figure(6),figure(7),figure(8),
figure(9),figure(10),figure(12));
        end
    end

end
qn=[qn(1:8);qn(9)+ang]
Tnwc

WMRA_Main_Open(qn,Tnwc)
```

WMRA_Main_Open

```
% This is a simplified version of the Main program to acomplish a
subtask
% with motion control. most of the options are prespecified and will
not be
% changed by the user

% Declaring the global variables to be used for the touch screen
control:
function WMRA_Main_Open(qn,Tnwc)
global VAR_DX
global VAR_SCREENOPN
global dHo

% Defining used parameters:
```

```
d2r=pi/180; % Conversions from Degrees to Radians.
r2d=180/pi; % Conversions from Radians to Degrees.

% Reading the Wheelchair's constant dimentions, all dimentions are
converted in millimeters:
L=WMRA_WCD;
radi=1000;
e=1;
% User input prompts:

%choice000 = input('\n Choose what to control: \n For combined
Wheelchair and Arm control, press "1", \n For Arm only control, press
"2", \n For Wheelchair only control, press "3". \n','s');
choice000 = 1;
    WCA=1;
%choice00000 = input('\n Choose whose frame to base the control on: \n
For Ground Frame, press "1", \n For Wheelchair Frame, press "2", \n For
Gripper Frame, press "3". \n','s');
choice00000=1;
    coord=1;
%choice0000 = input('\n Choose the cartesian coordinates to be
controlled: \n For Position and Orientation, press "1", \n For Position
only, press "2". \n','s');
choice0000 =1;
    cart=1;
%choice5 = input('\n Please enter the desired optimization method: (1=
SR-I & WLN, 2= P-I & WLN, 3= SR-I & ENE, 4= P-I & ENE) \n','s');
choice5 =1;
    optim=1;
%choice50 = input('\n Do you want to include Joint Limit Avoidance? (1=
Yes, 2= No) \n','s');
choice50 =1;
    JLA=1;
%choice500 = input('\n Do you want to include Joint Limit/Velocity and
Obstacle Safety Stop? (1= Yes, 2= No) \n','s');
choice500 = 1;
        JLO=1;
%choice0 = input('\n Choose the control mode: \n For circle control,
press "6", \n For position control, press "1", \n For velocity control,
press "2", \n For SpaceBall control, press "3", \n For Psychology Mask
control, press "4", \n For Touch Screen control, press "5",\n For
Switch, press "7". \n','s');
choice0 ='6';
if choice0=='1'
    cont=1;
    %Td = input('\n Please enter the transformation matrix of the
desired position and orientation from the control-based frame \n (e.g.
[0 0 1 1455;-1 0 0 369;0 -1 0 999; 0 0 0 1]) \n');
    Td=[0 0 1 1955;-1 0 0 -1169;0 -1 0 999; 0 0 0 1] *WMRA_rotx(-
pi/6)*WMRA_roty(pi/4);
    %v = input('\n Please enter the desired linear velocity of the
gripper in mm/s (e.g. 50) \n');
    v=50;
```

```
    %choice00 = input('\n Chose the Trajectory generation function: \n
Press "1" for a Polynomial function with Blending, or \n press "2" for
a Polynomial function without Blending, or \n press "3" for a Linear
function.\n','s');
    choice00=1;
        trajf=1;
   elseif choice0=='7'
    cont=1;
    Td = [0 0 1 1455;-1 0 0 369;0 -1 0 999; 0 0 0 1];
    v = input('\n Please enter the desired linear velocity of the
gripper in mm/s (e.g. 50) \n');
    choice00 = input('\n Chose the Trajectory generation function: \n
Press "1" for a Polynomial function with Blending, or \n press "2" for
a Polynomial function without Blending, or \n press "3" for a Linear
function.\n','s');
    if choice00=='2'
        trajf=2;
    elseif choice00=='3'
        trajf=3;
    else
        trajf=1;
    end
elseif choice0=='2'
    cont=2;
    ts = input('\n Please enter the desired simulation time in seconds
(e.g. 2) \n');
    if cart==2
        Vd = input('\n Please enter the desired 3x1 cartesian velocity
vector of the gripper (in mm/s) (e.g. [70;70;-70]) \n');
    else
        Vd = input('\n Please enter the desired 6x1 cartesian velocity
vector of the gripper (in mm/s, radians/s) (e.g. [70;70;-
70;0.001;0.001;0.001]) \n');
    end
elseif choice0=='3'
    cont=3;
    % Space Ball will be used for control.
    v = input('\n Please enter the desired linear velocity of the
gripper in mm/s (e.g. 50) \n');
elseif choice0=='4'
    cont=4;
    % BCI 2000 Psychology Mask will be used for control.
    v = input('\n Please enter the desired linear velocity of the
gripper in mm/s (e.g. 50) \n');
    port1 = input('\n Please enter the desired port number (e.g. 19711)
\n');
    choice00 = input('\n Chose the Trajectory generation function: \n
Press "1" for a Polynomial function with Blending, or \n press "2" for
a Polynomial function without Blending, or \n press "3" for a Linear
function, or \n press "4" for a Circular Polynomial function with
Blending. \n','s');
elseif choice0=='6'
    choice0=6;
```

```
    cont=6;
    %radi = input('\n Please enter the radius of the cirle in mm (e.g.
1000) \n');
    radi=1000;
    %e = input('\n If the door opens to the left press "1".\n If it
opens to the right press "2" \n');
    e=2;
    %v = input('\n Please enter the desired linear velocity of the
gripper in mm/s (e.g. 50) \n');
    v=50;
    trajf=4;
else
    cont=5;
    % Touch Screen will be used for control.
    v = input('\n Please enter the desired linear velocity of the
gripper in mm/s (e.g. 50) \n');
end

choice1 = input('\n Choose animation type or no animation: \n For
Virtual Reality Animation, press "1", \n For Matlab Graphics Animation,
press "2", \n For BOTH Animations, press "3", \n For NO Animation,
press "4". \n','s');
if choice1=='2'
    vr = 0; ml = 1;
elseif choice1=='3'
    vr = 1; ml = 1;
elseif choice1=='4'
    vr = 0; ml = 0;
else
    vr = 1; ml = 0;
end

%choice10 = input('\n Would you like to run the actual WMRA? \n For
yes, press "1", \n For no, press "2". \n','s');
choice10='2';
if choice10=='1'
    arm=1;
else
    arm=0;
end
%choice2 = input('\n Press "1" if you want to start at the "ready"
position, \n or press "2" if you want to enter the initial joint
angles. \n','s');
choice2='2';
if choice2=='2'
    %qi = input('\n Please enter the arms initial angles vector in
radians (e.g. [pi/2;pi/2;0;pi/2;pi/2;pi/2;0]) \n');
    %WCi = input('\n Please enter the initial x,y position and z
orientation of the WMRA base from the ground base in millimeters and
radians (e.g. [200;500;0.3]) \n');
    qi=qn(1:7);
    ini=0;
else
```

```
    qi=[90;90;0;90;90;90;0]*d2r;
    WCi=[0;0;0];
    ini=0;
    if vr==1 || ml==1 || arm==1
        %choice3 = input('\n Press "1" if you want to include "park" to
"ready" motion, \n or press "2" if not. \n','s');
        choice3= '1';
        if choice3=='2'
            ini=0;
        else
            ini=1;
        end
    end
end
%choice4 = input('\n Press "1" if you do NOT want to plot the
simulation results, \n or press "2" if do. \n','s');
choice4 ='2';
if choice4=='2'
    plt=2;
else
    plt=1;
end

% Calculating the Transformation Matrix of the initial position of the
WMRA's base:
Tiwc=Tnwc;

% Calculating the initial Wheelchair Variables:
qiwc=[qn(8);qn(9)];

% Calculating the initial transformation matrices:
[Ti, Tia, Tiwc, T01, T12, T23, T34, T45, T56, T67]=WMRA_Tall(1, qi,
[0;0], Tiwc);

if cont==1
    % Calculating the linear distance from the initial position to the
desired position and the linear velocity:
    if coord==2
        D=sqrt( (Td(1,4)-Tia(1,4))^2 + (Td(2,4)-Tia(2,4))^2 + (Td(3,4)-
Tia(3,4))^2);
    elseif coord==3
        D=sqrt( (Td(1,4))^2 + (Td(2,4))^2 + (Td(3,4))^2);
    else
        D=sqrt( (Td(1,4)-Ti(1,4))^2 + (Td(2,4)-Ti(2,4))^2 + (Td(3,4)-
Ti(3,4))^2);
    end
    % Calculating the number of iteration and the time increment (delta
t) if the linear step increment of the tip is 1 mm:
    dt=0.05;    % Time increment in seconds.
    total_time=D/v;     % Total time of animation.
    n=round(total_time/dt); % Number of iterations rounded up.
    dt=total_time/n;    % Adjusted time increment in seconds.
```

```
    % Calculating the Trajectory of the end effector, and once the
trajectory is calculated, we should redefine "Td" based on the ground
frame:
    if coord==2
        Tt=WMRA_traj(radi,trajf, Tia, Td, n+1);
        Td=Tiwc*Td;
    elseif coord==3
        Tt=WMRA_traj(radi,trajf, eye(4), Td, n+1);
        Td=Ti*Td;
    else
        Tt=WMRA_traj(radi,trajf, Ti, Td, n+1);
    end
elseif cont==2
    % Calculating the number of iterations and the time increment
(delta t) if the linear step increment of the gripper is 1 mm:
    dt=0.05;    % Time increment in seconds.
    total_time=ts;     % Total time of animation.
    n=round(total_time/dt); % Number of iterations rounded up.
    dt=total_time/n;    % Adjusted time increment in seconds.
    dx=Vd*dt;
    Td=Ti;
elseif cont==3
    WMRA_exit(); % This is to stop the simulation in SpaceBall control
when the user presses the exit key.
    dt=0.05;
    dx=v*dt*[spdata1(3)/20 ; -spdata1(1)/40 ; spdata1(2)/30 ;
spdata1(6)/1500 ; -spdata1(4)/900 ; spdata1(5)/1300];
    dg=spdata1(7);
    Td=Ti;
    n=1;
elseif cont==4
    WMRA_exit(); % This is to stop the simulation in Psychology Mask
control when the user presses the exit key.
    dt=0.05;
    dx=v*dt*WMRA_psy(port1);
    dg=dx(7);
    dx=dx(1:6);
    Td=Ti;
    n=1;
elseif cont==6
    % Calculating the desired transformation matrix based on the
radius:
    Tdoor=Ti;
    Tdoor(1,4)=Ti(1,4)+radi/2;
    Tdoor(2,4)=Ti(2,4)+radi/2;
    Td=Tdoor;
    Td(1,4)=Td(1,4)-radi;
    Td(2,4)=Td(2,4)-((-1)^e)*radi;
    % Calculating the circular distance from the initial position to
the desired position and the linear velocity:
    D=0.5*pi*radi;
    % Calculating the number of iteration and the time increment (delta
t) if the linear step increment of the tip is 1 mm:
```

```
    dt=0.05;    % Time increment in seconds.
    total_time=D/v;     % Total time of animation.
    n=round(total_time/dt); % Number of iterations rounded up.
    dt=total_time/n;    % Adjusted time increment in seconds.
    % Calculating the Trajectory of the end effector, and once the
trajectory is calculated, we should redefine "Td" based on the ground
frame:
    if coord==2
        Tt=WMRA_traj(radi,trajf, Tia, Td, n+1,e);
        Td=Tiwc*Td;
    elseif coord==3
        Tt=WMRA_traj(radi,trajf, eye(4), Td, n+1,e);
        Td=Ti*Td;
    else
        Tt=WMRA_traj(radi,trajf, Ti, Td, n+1,e);
    end
else
    WMRA_screen('0');   % This is to start the screen controls.
Argument: '0'=BACK button disabled, '1'=BACK button enabled.
    dt=0.05;
    dx=v*dt*VAR_DX(1:6);
    dg=VAR_DX(7);
    Td=Ti;
    n=1;
end

% Initializing the joint angles, the Transformation Matrix, and time:
dq=zeros(9,1);
dg=0;
qo=[qi;qiwc];
To=Ti;
Toa=Tia;
Towc=Tiwc;
tt=0;
i=1;
dHo=[0;0;0;0;0;0;0];



% Starting a timer:
tic

while i<=(n+1)

%Calculating the angle between the trajectory and the wheelchair's x
axis
the=WMRA_opt_angle(Td,Ti,Towc,L,T01, T12, T23, T34, T45, T56, T67);
% Starting the Iteration Loop:

    % Calculating the 6X7 Jacobian of the arm in frame 0:
    [Joa,detJoa]=WMRA_J07(T01, T12, T23, T34, T45, T56, T67);
```

Appendix B (continued)

```
    % Calculating the 6X2 Jacobian based on the WMRA's base in the
ground frame:
    phi=atan2(Towc(2,1),Towc(1,1));
    Jowc=WMRA_Jga(1, phi, Toa(1:2,4));

    % Changing the Jacobian reference frame based on the choice of
which coordinates frame are referenced in the Cartesian control:
    % coord=1 for Ground Coordinates Control.
    % coord=2 for Wheelchair Coordinates Control.
    % coord=3 for Gripper Coordinates Control.
    if coord==2
        Joa=Joa;
        Jowc=[Towc(1:3,1:3)' zeros(3); zeros(3) Towc(1:3,1:3)']*Jowc;
    elseif coord==3
        Joa=[Toa(1:3,1:3)' zeros(3); zeros(3) Toa(1:3,1:3)']*Joa;
        Jowc=[To(1:3,1:3)' zeros(3); zeros(3) To(1:3,1:3)']*Jowc;
    elseif coord==1
        Joa=[Towc(1:3,1:3) zeros(3); zeros(3) Towc(1:3,1:3)]*Joa;
        Jowc=Jowc;
    end

    % Calculating the 3X9 or 6X9 augmented Jacobian of the WMRA system
based on the ground frame:
    if cart==2
        Joa=Joa(1:3,1:7);
        detJoa=sqrt(det(Joa*Joa'));
        Jowc=Jowc(1:3,1:2);
        Jo=[Joa Jowc];
        detJo=sqrt(det(Jo*Jo'));
    else
        Jo=[Joa Jowc];
        detJo=sqrt(det(Jo*Jo'));
    end

    % Finding the Cartesian errors of the end effector:
    if cont==1 || cont==6
        % Calculating the Position and Orientation errors of the end
effector, and the rates of motion of the end effector:
        if coord==2
            invTowc=[Towc(1:3,1:3)' , -Towc(1:3,1:3)'*Towc(1:3,4);0 0 0
1];
            Ttnew=invTowc*Tiwc*Tt(:,:,i);
            dx=WMRA_delta(Toa, Ttnew);
        elseif coord==3
            invTo=[To(1:3,1:3)' , -To(1:3,1:3)'*To(1:3,4);0 0 0 1];
            Ttnew=invTo*Ti*Tt(:,:,i);
            dx=WMRA_delta(eye(4), Ttnew);
        else
            dx=WMRA_delta(To, Tt(:,:,i));
        end
    elseif cont==2
```

```
    elseif cont==3
        dx=v*dt*[spdata1(3)/20 ; -spdata1(1)/40 ; spdata1(2)/30 ;
spdata1(6)/1500 ; -spdata1(4)/900 ; spdata1(5)/1300];
        dg=spdata1(7);
    elseif cont==4
        dx=v*dt*WMRA_psy(port1);
        dg=dx(7);
        dx=dx(1:6);
    else
        dx=v*dt*VAR_DX(1:6);
        dg=VAR_DX(7);
    end

    % Changing the order of Cartesian motion in the case when gripper
reference frame is selected for control with the screen or psy or
SpaceBall interfaces:
    if coord==3 && cont>=3
        dx=[-dx(2);-dx(3);dx(1);-dx(5);-dx(6);dx(4)];
    end

    if cart==2
        dx=dx(1:3);
    end

    % Calculating the resolved rate with optimization:
    % Index input values for "optim": 1= SR-I & WLN, 2= P-I & WLN, 3=
SR-I & ENE, 4= P-I & ENE:
    if WCA==2
        dq=WMRA_Opt(optim, JLA, JLO, Joa, detJoa, dq(1:7), dx, dt,
qo,the,n,choice0);
        dq=[dq;0;0];
    elseif WCA==3
        dq=WMRA_Opt(optim, JLA, JLO, Jowc, 1, dq(8:9), dx(1:2), dt,
1,the,n,choice0);
        dq=[0;0;0;0;0;0;0;dq];
    else
        dq=WMRA_Opt(optim, JLA, JLO, Jo, detJo, dq, dx, dt,
qo,the,n,choice0);
    end

    % Calculating the new Joint Angles:
    qn=qo+dq;

    % Calculating the new Transformation Matrices:
    [Tn, Tna, Tnwc, T01, T12, T23, T34, T45, T56, T67]=WMRA_Tall(2, qn,
dq(8:9), Towc);

    % A safety condition function to stop the joints that may cause
colision of the arm with itself, the wheelchair, or the human user:
    if JLO==1 && WCA~=3
        dq(1:7)=WMRA_collide(dq(1:7), T01, T12, T23, T34, T45, T56,
T67);
```

```
        % Re-calculating the new Joint Angles:
        qn=qo+dq;
        % Re-calculating the new Transformation Matrices:
        [Tn, Tna, Tnwc, T01, T12, T23, T34, T45, T56, T67]=WMRA_Tall(2,
qn, dq(8:9), Towc);
    end

    % Saving the plot data in case plots are required:
    if plt==2
        WMRA_Plots(1, L, r2d, dt, i, tt, qn, dq, Tn, Tnwc, detJoa,
detJo);
    end

    % Updating Physical Arm:
    if arm==1
        ddt=ddt+dt;
        if ddt>=0.5 || i>=(n+1)
            WMRA_ARM_Motion(2, 1, [qn;dg], ddt);
            ddt=0;
        end
    end

    % Updating Virtual Reality Animation:
    if vr==1
        WMRA_VR_Animation(2, Tnwc, qn);
    end

    % Updating Matlab Graphics Animation:
    if ml==1
        WMRA_ML_Animation(2, Ti, Td, Tnwc, T01, T12, T23, T34, T45,
T56, T67);
    end

    % Re-Drawing the Animation:
    if vr==1 || ml==1
        drawnow;
    end

    % Updating the old values with the new values for the next
iteration:
    qo=qn;
    To=Tn;
    Toa=Tna;
    Towc=Tnwc;
    tt=tt+dt;
    i=i+1;

    % Stopping the simulation when the exit button is pressed:
    if cont==3 || cont==4 || cont==5
        if (VAR_SCREENOPN == 1)
            n=n+1;
        else
```

```
            break
        end
    end

    % Delay to comply with the required speed:
    if toc < tt
        pause(tt-toc);
    end
end

% Reading the elapsed time and printing it with the simulation time:
if cont==1 || cont==2 || cont==6, fprintf('\nSimula. time is %6.6f
seconds.\n' , total_time); end
toc

% Plotting:
if plt==2
    WMRA_Plots(2, L, r2d, dt, i, tt, qn, dq, Tn, Tnwc, detJoa, detJo);
end

if vr==1 || ml==1 || arm==1
    %Orienting the Wheelchair to a desired final orientation
    choice9 = input('\n Do you want to rotate the wheelchair? \n Press
"1" for Yes, or press "2" for No. \n','s');
    if choice9=='1'
        ang = input('\n enter the desired angle in radians \n');
      WMRA_final_orientation(2, vr, ml, arm, Tnwc, qn,ang)
    end
    % Going back to the ready position:
    %choice6 = input('\n Do you want to go back to the "ready"
position? \n Press "1" for Yes, or press "2" for No. \n','s');
    choice6 ='2';
    if choice6=='1'
        WMRA_any2ready(2, vr, ml, arm, Tnwc, qn);
        % Going back to the parking position:
        choice7 = input('\n Do you want to go back to the "parking"
position? \n Press "1" for Yes, or press "2" for No. \n','s');
        if choice7=='1'
            WMRA_ready2park(2, vr, ml, arm, Tnwc, qn(8:9));
        end
    end

    % Closing the Arm library and Matlab Graphics Animation and Virtual
Reality Animation and Plots windows:
    %choice8 = input('\n Do you want to close all simulation windows
and arm controls? \n Press "1" for Yes, or press "2" for No. \n','s');
    choice8 ='2';
    if choice8=='1'
        if arm==1
            WMRA_ARM_Motion(3, 0, 0, 0);
        end
        if vr==1
```

Appendix B (continued)

```
            WMRA_VR_Animation(3, 0, 0);
        end
        if ml==1
            WMRA_ML_Animation(3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
        end
        if plt==2
            close
(figure(2),figure(3),figure(4),figure(5),figure(6),figure(7),figure(8),
figure(9),figure(10),figure(12));
        end
    end

end
end
```

WMRA_Main_Reach

```
% This is a simplified version of the Main program to acomplish a
subtask
% with motion control. most of the options are prespecified and will
not be
% changed by the user

% Declaring the global variables to be used for the touch screen
control:
function WMRA_Main_Reach
global VAR_DX
global VAR_SCREENOPN
global dHo

% Defining used parameters:
d2r=pi/180; % Conversions from Degrees to Radians.
r2d=180/pi; % Conversions from Radians to Degrees.

% Reading the Wheelchair's constant dimentions, all dimentions are
converted in millimeters:
L=WMRA_WCD;
radi=1000;
e=1;
% User input prompts:

%choice000 = input('\n Choose what to control: \n For combined
Wheelchair and Arm control, press "1", \n For Arm only control, press
"2", \n For Wheelchair only control, press "3". \n','s');
choice000 = 1;
    WCA=1;
```

Appendix B (continued)

```
%choice00000 = input('\n Choose whose frame to base the control on: \n
For Ground Frame, press "1", \n For Wheelchair Frame, press "2", \n For
Gripper Frame, press "3". \n','s');
choice00000=1;
    coord=1;
%choice0000 = input('\n Choose the cartesian coordinates to be
controlled: \n For Position and Orientation, press "1", \n For Position
only, press "2". \n','s');
choice0000 =1;
    cart=1;
%choice5 = input('\n Please enter the desired optimization method: (1=
SR-I & WLN, 2= P-I & WLN, 3= SR-I & ENE, 4= P-I & ENE) \n','s');
choice5 =1;
    optim=1;
%choice50 = input('\n Do you want to include Joint Limit Avoidance? (1=
Yes, 2= No) \n','s');
choice50 =1;
    JLA=1;
%choice500 = input('\n Do you want to include Joint Limit/Velocity and
Obstacle Safety Stop? (1= Yes, 2= No) \n','s');
choice500 = 1;
        JLO=1;
%choice0 = input('\n Choose the control mode: \n For circle control,
press "6", \n For position control, press "1", \n For velocity control,
press "2", \n For SpaceBall control, press "3", \n For Psychology Mask
control, press "4", \n For Touch Screen control, press "5",\n For
Switch, press "7". \n','s');
choice0 ='1';
if choice0=='1'
    cont=1;
    %Td = input('\n Please enter the transformation matrix of the
desired position and orientation from the control-based frame \n (e.g.
[0 0 1 1455;-1 0 0 369;0 -1 0 999; 0 0 0 1]) \n');
    Td = [0 0 1 1455;-1 0 0 -1169;0 -1 0 999; 0 0 0 1];
    %v = input('\n Please enter the desired linear velocity of the
gripper in mm/s (e.g. 50) \n');
    v=50;
    %choice00 = input('\n Chose the Trajectory generation function: \n
Press "1" for a Polynomial function with Blending, or \n press "2" for
a Polynomial function without Blending, or \n press "3" for a Linear
function.\n','s');
    choice00=1;
        trajf=1;
   elseif choice0=='7'
    cont=1;
    Td = [0 0 1 1455;-1 0 0 369;0 -1 0 999; 0 0 0 1];
    v = input('\n Please enter the desired linear velocity of the
gripper in mm/s (e.g. 50) \n');
    choice00 = input('\n Chose the Trajectory generation function: \n
Press "1" for a Polynomial function with Blending, or \n press "2" for
a Polynomial function without Blending, or \n press "3" for a Linear
function.\n','s');
    if choice00=='2'
```

```
        trajf=2;
    elseif choice00=='3'
        trajf=3;
    else
        trajf=1;
    end
elseif choice0=='2'
    cont=2;
    ts = input('\n Please enter the desired simulation time in seconds
(e.g. 2) \n');
    if cart==2
        Vd = input('\n Please enter the desired 3x1 cartesian velocity
vector of the gripper (in mm/s) (e.g. [70;70;-70]) \n');
    else
        Vd = input('\n Please enter the desired 6x1 cartesian velocity
vector of the gripper (in mm/s, radians/s) (e.g. [70;70;-
70;0.001;0.001;0.001]) \n');
    end
elseif choice0=='3'
    cont=3;
    % Space Ball will be used for control.
    v = input('\n Please enter the desired linear velocity of the
gripper in mm/s (e.g. 50) \n');
elseif choice0=='4'
    cont=4;
    % BCI 2000 Psychology Mask will be used for control.
    v = input('\n Please enter the desired linear velocity of the
gripper in mm/s (e.g. 50) \n');
    port1 = input('\n Please enter the desired port number (e.g. 19711)
\n');
    choice00 = input('\n Chose the Trajectory generation function: \n
Press "1" for a Polynomial function with Blending, or \n press "2" for
a Polynomial function without Blending, or \n press "3" for a Linear
function, or \n press "4" for a Circular Polynomial function with
Blending. \n','s');
elseif choice0=='6'
    choice0=6;
    cont=6;
    %radi = input('\n Please enter the radius of the cirle in mm (e.g.
1000) \n');
    radi=1000;
    %e = input('\n If the door opens to the left press "1".\n If it
opens to the right press "2" \n');
    e=2;
    %v = input('\n Please enter the desired linear velocity of the
gripper in mm/s (e.g. 50) \n');
    v=50;
    trajf=4;
else
    cont=5;
    % Touch Screen will be used for control.
    v = input('\n Please enter the desired linear velocity of the
gripper in mm/s (e.g. 50) \n');
```

```
end

choice1 = input('\n Choose animation type or no animation: \n For
Virtual Reality Animation, press "1", \n For Matlab Graphics Animation,
press "2", \n For BOTH Animations, press "3", \n For NO Animation,
press "4". \n','s');
if choice1=='2'
    vr = 0; ml = 1;
elseif choice1=='3'
    vr = 1; ml = 1;
elseif choice1=='4'
    vr = 0; ml = 0;
else
    vr = 1; ml = 0;
end

%choice10 = input('\n Would you like to run the actual WMRA? \n For
yes, press "1", \n For no, press "2". \n','s');
choice10='2';
if choice10=='1'
    arm=1;
else
    arm=0;
end
%choice2 = input('\n Press "1" if you want to start at the "ready"
position, \n or press "2" if you want to enter the initial joint
angles. \n','s');
choice2='1';
if choice2=='2'
    qi = input('\n Please enter the arms initial angles vector in
radians (e.g. [pi/2;pi/2;0;pi/2;pi/2;pi/2;0]) \n');
    WCi = input('\n Please enter the initial x,y position and z
orientation of the WMRA base from the ground base in millimeters and
radians (e.g. [200;500;0.3]) \n');
    ini=0;
else
    qi=[90;90;0;90;90;90;0]*d2r;
    WCi=[0;0;0];
    ini=0;
    if vr==1 || ml==1 || arm==1
        %choice3 = input('\n Press "1" if you want to include "park" to
"ready" motion, \n or press "2" if not. \n','s');
        choice3= '1';
        if choice3=='2'
            ini=0;
        else
            ini=1;
        end
    end
end
%choice4 = input('\n Press "1" if you do NOT want to plot the
simulation results, \n or press "2" if do. \n','s');
choice4 ='2';
```

Appendix B (continued)

```
if choice4=='2'
    plt=2;
else
    plt=1;
end

% Calculating the Transformation Matrix of the initial position of the
WMRA's base:
Tiwc=WMRA_p2T(WCi(1),WCi(2),WCi(3));

% Calculating the initial Wheelchair Variables:
qiwc=[sqrt(WCi(1)^2+WCi(2)^2);WCi(3)];

% Calculating the initial transformation matrices:
[Ti, Tia, Tiwc, T01, T12, T23, T34, T45, T56, T67]=WMRA_Tall(1, qi,
[0;0], Tiwc);

if cont==1
    % Calculating the linear distance from the initial position to the
desired position and the linear velocity:
    if coord==2
        D=sqrt( (Td(1,4)-Tia(1,4))^2 + (Td(2,4)-Tia(2,4))^2 + (Td(3,4)-
Tia(3,4))^2);
    elseif coord==3
        D=sqrt( (Td(1,4))^2 + (Td(2,4))^2 + (Td(3,4))^2);
    else
        D=sqrt( (Td(1,4)-Ti(1,4))^2 + (Td(2,4)-Ti(2,4))^2 + (Td(3,4)-
Ti(3,4))^2);
    end
    % Calculating the number of iteration and the time increment (delta
t) if the linear step increment of the tip is 1 mm:
    dt=0.05;    % Time increment in seconds.
    total_time=D/v;     % Total time of animation.
    n=round(total_time/dt); % Number of iterations rounded up.
    dt=total_time/n;    % Adjusted time increment in seconds.
    % Calculating the Trajectory of the end effector, and once the
trajectory is calculated, we should redefine "Td" based on the ground
frame:
    if coord==2
        Tt=WMRA_traj(radi,trajf, Tia, Td, n+1);
        Td=Tiwc*Td;
    elseif coord==3
        Tt=WMRA_traj(radi,trajf, eye(4), Td, n+1);
        Td=Ti*Td;
    else
        Tt=WMRA_traj(radi,trajf, Ti, Td, n+1);
    end
elseif cont==2
    % Calculating the number of iterations and the time increment
(delta t) if the linear step increment of the gripper is 1 mm:
    dt=0.05;    % Time increment in seconds.
    total_time=ts;     % Total time of animation.
```

```
    n=round(total_time/dt); % Number of iterations rounded up.
    dt=total_time/n;    % Adjusted time increment in seconds.
    dx=Vd*dt;
    Td=Ti;
elseif cont==3
    WMRA_exit(); % This is to stop the simulation in SpaceBall control
when the user presses the exit key.
    dt=0.05;
    dx=v*dt*[spdata1(3)/20 ; -spdata1(1)/40 ; spdata1(2)/30 ;
spdata1(6)/1500 ; -spdata1(4)/900 ; spdata1(5)/1300];
    dg=spdata1(7);
    Td=Ti;
    n=1;
elseif cont==4
    WMRA_exit(); % This is to stop the simulation in Psychology Mask
control when the user presses the exit key.
    dt=0.05;
    dx=v*dt*WMRA_psy(port1);
    dg=dx(7);
    dx=dx(1:6);
    Td=Ti;
    n=1;
elseif cont==6
    % Calculating the desired transformation matrix based on the
radius:
    Tdoor=Ti;
    Tdoor(1,4)=Ti(1,4)+radi/2;
    Tdoor(2,4)=Ti(2,4)+radi/2;
    Td=Tdoor;
    Td(1,4)=Td(1,4)-radi;
    Td(2,4)=Td(2,4)-((-1)^e)*radi;
    % Calculating the circular distance from the initial position to
the desired position and the linear velocity:
    D=0.5*pi*radi;
    % Calculating the number of iteration and the time increment (delta
t) if the linear step increment of the tip is 1 mm:
    dt=0.05;    % Time increment in seconds.
    total_time=D/v;     % Total time of animation.
    n=round(total_time/dt); % Number of iterations rounded up.
    dt=total_time/n;    % Adjusted time increment in seconds.
    % Calculating the Trajectory of the end effector, and once the
trajectory is calculated, we should redefine "Td" based on the ground
frame:
    if coord==2
        Tt=WMRA_traj(radi,trajf, Tia, Td, n+1,e);
        Td=Tiwc*Td;
    elseif coord==3
        Tt=WMRA_traj(radi,trajf, eye(4), Td, n+1,e);
        Td=Ti*Td;
    else
        Tt=WMRA_traj(radi,trajf, Ti, Td, n+1,e);
    end
else
```

```
    WMRA_screen('0');   % This is to start the screen controls.
Argument: '0'=BACK button disabled, '1'=BACK button enabled.
    dt=0.05;
    dx=v*dt*VAR_DX(1:6);
    dg=VAR_DX(7);
    Td=Ti;
    n=1;
end

% Initializing the joint angles, the Transformation Matrix, and time:
dq=zeros(9,1);
dg=0;
qo=[qi;qiwc];
To=Ti;
Toa=Tia;
Towc=Tiwc;
tt=0;
i=1;
dHo=[0;0;0;0;0;0;0];

% Initializing the WMRA:
if ini==0   % When no "park" to "ready" motion required.
    % Initializing Virtual Reality Animation:
    if vr==1
        WMRA_VR_Animation(1, Towc, qo);
    end
    % Initializing Robot Animation in Matlab Graphics:
    if ml==1
        WMRA_ML_Animation(1, To, Td, Towc, T01, T12, T23, T34, T45,
T56, T67);
    end
    % Initializing the Physical Arm:
    if arm==1
        WMRA_ARM_Motion(1, 2, [qo;dg], 0);
        ddt=0;
    end
elseif ini==1 && (vr==1 || ml==1 || arm==1)  % When "park" to "ready"
motion is required.
    WMRA_park2ready(1, vr, ml, arm, Towc, qo(8:9));
    if arm==1
        ddt=0;
    end
end

% Re-Drawing the Animation:
if vr==1 || ml==1
    drawnow;
end

% Starting a timer:
tic
ANG=zeros(1,n+1);
```

```
bmax=WMRA_opt_angle(Td,Ti,Towc,L,T01, T12, T23, T34, T45, T56, T67);
while i<=(n+1)

%Calculating the angle between the trajectory and the wheelchair's x
axis
the=WMRA_opt_angle(Td,Ti,Towc,L,T01, T12, T23, T34, T45, T56, T67);
ANG(i)=the;
% Starting the Iteration Loop:

    % Calculating the 6X7 Jacobian of the arm in frame 0:
    [Joa,detJoa]=WMRA_J07(T01, T12, T23, T34, T45, T56, T67);

    % Calculating the 6X2 Jacobian based on the WMRA's base in the
ground frame:
    phi=atan2(Towc(2,1),Towc(1,1));
    Jowc=WMRA_Jga(1, phi, Toa(1:2,4));

    % Changing the Jacobian reference frame based on the choice of
which coordinates frame are referenced in the Cartesian control:
    % coord=1 for Ground Coordinates Control.
    % coord=2 for Wheelchair Coordinates Control.
    % coord=3 for Gripper Coordinates Control.
    if coord==2
        Joa=Joa;
        Jowc=[Towc(1:3,1:3)' zeros(3); zeros(3) Towc(1:3,1:3)']*Jowc;
    elseif coord==3
        Joa=[Toa(1:3,1:3)' zeros(3); zeros(3) Toa(1:3,1:3)']*Joa;
        Jowc=[To(1:3,1:3)' zeros(3); zeros(3) To(1:3,1:3)']*Jowc;
    elseif coord==1
        Joa=[Towc(1:3,1:3) zeros(3); zeros(3) Towc(1:3,1:3)]*Joa;
        Jowc=Jowc;
    end

    % Calculating the 3X9 or 6X9 augmented Jacobian of the WMRA system
based on the ground frame:
    if cart==2
        Joa=Joa(1:3,1:7);
        detJoa=sqrt(det(Joa*Joa'));
        Jowc=Jowc(1:3,1:2);
        Jo=[Joa Jowc];
        detJo=sqrt(det(Jo*Jo'));
    else
        Jo=[Joa Jowc];
        detJo=sqrt(det(Jo*Jo'));
    end

    % Finding the Cartesian errors of the end effector:
    if cont==1 || cont==6
        % Calculating the Position and Orientation errors of the end
effector, and the rates of motion of the end effector:
        if coord==2
```

```
            invTowc=[Towc(1:3,1:3)' , -Towc(1:3,1:3)'*Towc(1:3,4);0 0 0
1];
            Ttnew=invTowc*Tiwc*Tt(:,:,i);
            dx=WMRA_delta(Toa, Ttnew);
        elseif coord==3
            invTo=[To(1:3,1:3)' , -To(1:3,1:3)'*To(1:3,4);0 0 0 1];
            Ttnew=invTo*Ti*Tt(:,:,i);
            dx=WMRA_delta(eye(4), Ttnew);
        else
            dx=WMRA_delta(To, Tt(:,:,i));
        end
    elseif cont==2

    elseif cont==3
        dx=v*dt*[spdata1(3)/20 ; -spdata1(1)/40 ; spdata1(2)/30 ;
spdata1(6)/1500 ; -spdata1(4)/900 ; spdata1(5)/1300];
        dg=spdata1(7);
    elseif cont==4
        dx=v*dt*WMRA_psy(port1);
        dg=dx(7);
        dx=dx(1:6);
    else
        dx=v*dt*VAR_DX(1:6);
        dg=VAR_DX(7);
    end

    % Changing the order of Cartesian motion in the case when gripper
reference frame is selected for control with the screen or psy or
SpaceBall interfaces:
    if coord==3 && cont>=3
        dx=[-dx(2);-dx(3);dx(1);-dx(5);-dx(6);dx(4)];
    end

    if cart==2
        dx=dx(1:3);
    end

    % Calculating the resolved rate with optimization:
    % Index input values for "optim": 1= SR-I & WLN, 2= P-I & WLN, 3=
SR-I & ENE, 4= P-I & ENE:
    if WCA==2
        dq=WMRA_Opt(optim, JLA, JLO, Joa, detJoa, dq(1:7), dx, dt,
qo,the,n,choice0,bmax);
        dq=[dq;0;0];
    elseif WCA==3
        dq=WMRA_Opt(optim, JLA, JLO, Jowc, 1, dq(8:9), dx(1:2), dt,
1,the,n,choice0,bmax);
        dq=[0;0;0;0;0;0;0;dq];
    else
        dq=WMRA_Opt(optim, JLA, JLO, Jo, detJo, dq, dx, dt,
qo,the,n,choice0,bmax);
    end
```

```
    % Calculating the new Joint Angles:
    qn=qo+dq;

    % Calculating the new Transformation Matrices:
    [Tn, Tna, Tnwc, T01, T12, T23, T34, T45, T56, T67]=WMRA_Tall(2, qn,
dq(8:9), Towc);

    % A safety condition function to stop the joints that may cause
colision of the arm with itself, the wheelchair, or the human user:
    if JLO==1 && WCA~=3
        dq(1:7)=WMRA_collide(dq(1:7), T01, T12, T23, T34, T45, T56,
T67);
        % Re-calculating the new Joint Angles:
        qn=qo+dq;
        % Re-calculating the new Transformation Matrices:
        [Tn, Tna, Tnwc, T01, T12, T23, T34, T45, T56, T67]=WMRA_Tall(2,
qn, dq(8:9), Towc);
    end

    % Saving the plot data in case plots are required:
    if plt==2
        WMRA_Plots(1, L, r2d, dt, i, tt, qn, dq, Tn, Tnwc, detJoa,
detJo);
    end

    % Updating Physical Arm:
    if arm==1
        ddt=ddt+dt;
        if ddt>=0.5 || i>=(n+1)
            WMRA_ARM_Motion(2, 1, [qn;dg], ddt);
            ddt=0;
        end
    end

    % Updating Virtual Reality Animation:
    if vr==1
        WMRA_VR_Animation(2, Tnwc, qn);
    end

    % Updating Matlab Graphics Animation:
    if ml==1
        WMRA_ML_Animation(2, Ti, Td, Tnwc, T01, T12, T23, T34, T45,
T56, T67);
    end

    % Re-Drawing the Animation:
    if vr==1 || ml==1
        drawnow;
    end
```

```
    % Updating the old values with the new values for the next
iteration:
    qo=qn;
    To=Tn;
    Toa=Tna;
    Towc=Tnwc;
    tt=tt+dt;
    i=i+1;

    % Stopping the simulation when the exit button is pressed:
    if cont==3 || cont==4 || cont==5
        if (VAR_SCREENOPN == 1)
            n=n+1;
        else
            break
        end
    end

    % Delay to comply with the required speed:
    if toc < tt
        pause(tt-toc);
    end
end

% Reading the elapsed time and printing it with the simulation time:
if cont==1 || cont==2 || cont==6, fprintf('\nSimula. time is %6.6f
seconds.\n' , total_time); end
toc

% Plotting:


if vr==1 || ml==1 || arm==1
    %Orienting the Wheelchair to a desired final orientation
    choice9 = input('\n Do you want to rotate the wheelchair? \n Press
"1" for Yes, or press "2" for No. \n','s');
    if choice9=='1'
        ang = input('\n enter the desired angle in radians \n');
      WMRA_final_orientation(2, vr, ml, arm, Tnwc, qn,ang)
    end
    if plt==2
    WMRA_Plots(2, L, r2d, dt, i, tt, qn, dq, Tn, Tnwc, detJoa, detJo);
    end

    % Going back to the ready position:
    %choice6 = input('\n Do you want to go back to the "ready"
position? \n Press "1" for Yes, or press "2" for No. \n','s');
    choice6 ='2';
    if choice6=='1'
        WMRA_any2ready(2, vr, ml, arm, Tnwc, qn);
        % Going back to the parking position:
```

```
        choice7 = input('\n Do you want to go back to the "parking"
position? \n Press "1" for Yes, or press "2" for No. \n','s');
        if choice7=='1'
            WMRA_ready2park(2, vr, ml, arm, Tnwc, qn(8:9));
        end
    end

    % Closing the Arm library and Matlab Graphics Animation and Virtual
Reality Animation and Plots windows:
    %choice8 = input('\n Do you want to close all simulation windows
and arm controls? \n Press "1" for Yes, or press "2" for No. \n','s');
    choice8 ='2';
    if choice8=='1'
        if arm==1
            WMRA_ARM_Motion(3, 0, 0, 0);
        end
        if vr==1
            WMRA_VR_Animation(3, 0, 0);
        end
        if ml==1
            WMRA_ML_Animation(3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
        end
        if plt==2
            close
(figure(2),figure(3),figure(4),figure(5),figure(6),figure(7),figure(8),
figure(9),figure(10),figure(12));
        end
    end

end
TIME=[1:n+1];
Or=ANG+0.75;
figure(17)
plot(TIME,ANG,'LineWidth',5)
axis([0 700 0 5])
grid on; title('Angle \Theta Vs Iteration number');xlabel('Iteration
number');ylabel('\Theta');
figure(15)
plot(TIME,Or,'LineWidth',5)
axis([0 700 0 5])
grid on; title('Wheechair orientation Vs Iteration
number');xlabel('Iteration number');ylabel('\Orientation');
end
```

Appendix B (continued)

WMRA_OP

```
%##########################################################################
####
% This M file creates optimized parameters for the weighted matrix
values
% of the wheels to account for positition during the
% execution of a given trajectory.


% Function Declaration
function W1=WMRA_OP(the,i,n,bmax)
if atan2(sin(the),cos(the))<0.3

    W1=1;
elseif atan2(sin(the),cos(the))~=0
   W1=10;
end
```

WMRA_Opt

```
% This function is for the resolved rate and optimization solution of
the USF WMRA with 9 DOF.
% Function Declaration:
function [dq]=WMRA_Opt(i, JLA, JLO, Jo, detJo, dq, dx, dt,
q,the,n,choice0,bmax)

% Declaring a global variable:
global dHo
% Reading the Wheelchair's constant dimentions, all dimentions are
converted in millimeters:
L=WMRA_WCD;
% The case when wheelchair-only control is required with no arm motion:
if i==0
    WCA=3;
    % claculating the Inverse of the Jacobian, which is always non-
singular:
    pinvJo=inv(Jo(1:2,1:2));
    % calculating the joint angle change:
    % Here, dq of the wheels are translated from radians to distances
travelled after using the Jacobian.
    dq=pinvJo*dx;
    dq(1)=dq(1)*L(5);
else
    % Reading the physical joint limits of the arm:
    [qmin,qmax]=WMRA_Jlimit;
```

Appendix B (continued)

```
    % Creating the gradient of the optimization function to avoid joint
limits:
    dH=[0;0;0;0;0;0;0];
    if JLA==1
        for j=1:7
            dH(j)=-0.25*(qmax(j)-qmin(j))^2*(2*q(j)-qmax(j)-
qmin(j))/((qmax(j)-q(j))^2*(q(j)-qmin(j))^2);
            % Re-defining the weight in case the joint is moving away
from it's limit or the joint limit was exceeded:
            if abs(dH(j)) < abs(dHo(j)) && q(j) < qmax(j) && q(j) >
qmin(j)
                dH(j)=0;
            elseif abs(dH(j)) < abs(dHo(j)) && (q(j) >= qmax(j) || q(j)
<= qmin(j))
                dH(j)=inf;
            elseif abs(dH(j)) > abs(dHo(j)) && (q(j) >= qmax(j) || q(j)
<= qmin(j))
                dH(j)=0;
            end
        end
    end
    dHo=dH;
% The case when arm-only control is required with no wheelchair motion:
    if max(size(dq))==7
        WCA=2;
        wo=20000000;
        ko=350000;
        % The weight matrix to be used for the Weighted Least Norm
Solution with Joint Limit Avoidance:
        W=diag(1*[1;1;1;1;1;1;1]+1*abs(dH));
        % The inverse of the diagonal weight matrix:
        dia=diag(W);
        Winv=diag([1/dia(1); 1/dia(2); 1/dia(3); 1/dia(4); 1/dia(5);
1/dia(6); 1/dia(7)]);
% The case when wheelchair-and-arm control is required:
    else
        WCA=1;
        wo=34000000;
        ko=13;
        % The weight matrix to be used for the Weighted Least Norm
Solution:
        if choice0==6;
        W=diag([1*[1;1;1;1;1;1;1]+1*abs(dH);0.5;1000]);
        else

W=diag([1*[1;1;1;1;1;1;1]+1*abs(dH);WMRA_OP(the,i,n,bmax);WMRA_OR(the,i
,n,bmax)]);
        end
        % The inverse of the diagonal weight matrix:
        dia=diag(W);
        Winv=diag([1/dia(1); 1/dia(2); 1/dia(3); 1/dia(4); 1/dia(5);
1/dia(6); 1/dia(7); 1/dia(8); 1/dia(9)]);
    end
```

```
    % Redefining the determinant based on the weight:
    if i==1 || i==2
        detJo=sqrt(det(Jo*Winv*Jo'));
    end
    dof=max(size(dx));
end

% SR-Inverse and Weighted Least Norm Optimization:
if i==1
    % Calculating the variable scale factor, sf:
    if detJo<wo
        sf=ko*(1-detJo/wo)^2;        % from eq. 9.79 page 268 of
Nakamura's book.
    else
        sf=0;
    end
    % claculating the SR-Inverse of the Jacobian:
    pinvJo=Winv*Jo'*inv(Jo*Winv*Jo'+sf*eye(dof));
    % calculating the joint angle change optimized based on the
Weighted Least Norm Solution:
    % Here, dq of the wheels are translated from radians to distances
travelled after using the Jacobian.
    if WCA==2
        dq=pinvJo*dx;
    else
        dq=pinvJo*dx;
        dq(8)=dq(8)*L(5);
    end

% Pseudo Inverse and Weighted Least Norm Optimization:
elseif i==2
    % claculating the Pseudo Inverse of the Jacobian:
    pinvJo=Winv*Jo'*inv(Jo*Winv*Jo');
    % calculating the joint angle change optimized based on the
Weighted Least Norm Solution:
    % Here, dq of the wheels are translated from radians to distances
travelled after using the Jacobian.
    if WCA==2
        dq=pinvJo*dx;
    else
        dq=pinvJo*dx;
        dq(8)=dq(8)*L(5);
    end

% SR-Inverse and Projection Gradient Optimization based on Euclidean
norm of errors:
elseif i==3
    % Calculating the variable scale factor, sf:
    if detJo<wo
        sf=ko*(1-detJo/wo)^2;        % from eq. 9.79 page 268 of
Nakamura's book.
    else
        sf=0;
```

```
    end
    % claculating the SR-Inverse of the Jacobian:
    pinvJo=Jo'*inv(Jo*Jo'+sf*eye(dof));
    % calculating the joint angle change optimized based on minimizing
the Euclidean norm of errors:
    % Here, dq of the wheels are translated from distances travelled to
radians, and back after using the Jacobian.
    if WCA==2
        %dq=pinvJo*dx+(eye(7)-pinvJo*Jo)*dq;
        dq=pinvJo*dx+0.001*(eye(7)-pinvJo*Jo)*dH;
    else
        %dq(8)=dq(8)/L(5);
        %dq=pinvJo*dx+(eye(9)-pinvJo*Jo)*dq;
        dq=pinvJo*dx+0.001*(eye(9)-pinvJo*Jo)*[dH;0;0];
        dq(8)=dq(8)*L(5);
    end

% Pseudo Inverse and Projection Gradient Optimization based on
Euclidean norm of errors:
elseif i==4
    % claculating the Pseudo Inverse of the Jacobian:
    pinvJo=Jo'*inv(Jo*Jo');
    % calculating the joint angle change optimized based on minimizing
the Euclidean norm of errors:
    % Here, dq of the wheels are translated from distances travelled to
radians, and back after using the Jacobian.
    if WCA==2
        %dq=pinvJo*dx+(eye(7)-pinvJo*Jo)*dq;
        dq=pinvJo*dx+0.001*(eye(7)-pinvJo*Jo)*dH;
    else
        %dq(8)=dq(8)/L(5);
        %dq=pinvJo*dx+(eye(9)-pinvJo*Jo)*dq;
        dq=pinvJo*dx+0.001*(eye(9)-pinvJo*Jo)*[dH;0;0];
        dq(8)=dq(8)*L(5);
    end
end

if JLO==1
    % A safety condition to stop the joint that reaches the joint
limits in the arm:
    if WCA~=3
        for k=1:7
            if q(k) >= qmax(k) || q(k) <= qmin(k)
                dq(k)=0;
            end
        end
    end
    % A safety condition to slow the joint that exceeds the velocity
limits in the WMRA:
    if WCA==3
        dqmax=dt*[100;0.15]; % Joiny velocity limits when the time
increment is dt second.
    else
```

Appendix B (continued)

```
        dqmax=dt*[0.5;0.5;0.5;0.5;0.5;0.5;0.5;100;0.15]; % Joiny
velocity limits when the time increment is dt second.
    end
    for k=1:max(size(dq))
        if abs(dq(k)) >= dqmax(k)
            dq(k)=sign(dq(k))*dqmax(k);
        end
    end
end
```

WMRA_opt_angle

```
%#########################################################################
####
%This M-File keeps track of the angel created between the projection of
the
%trajectory in the XY plane and the center axis of the whhechair.


function the=WMRA_opt_angle(Td,Ti,Towc,L,T01, T12, T23, T34, T45, T56,
T67)

 % Arm:
    T1=Towc*T01;
    T2=T1*T12;
    T3=T2*T23;
    T4=T3*T34;
    T5=T4*T45;
    T6=T5*T56;
    T7=T6*T67;

Lt=sqrt((T7(1,4)-Td(1,4))^2+(T7(2,4)-Td(2,4))^2+(T7(3,4)-Td(3,4))^2);

%Updating the transformation matrices
 T8=Towc;                                        % Arm Base Position.
 T9=T8*WMRA_transl(-L(2),-L(3),-L(4));           % Wheelbase Center.
 T10=T9*WMRA_transl(0,-L(1)/2,0);                % Right Wheel Center.
 T11=T9*WMRA_transl(0,L(1)/2,0);                 % Left Wheel Center.

V3D=[Td(1,4)-Ti(1,4);Td(2,4)-Ti(2,4);Td(3,4)-Ti(3,4)];
%Let n be the normal vector to the plane Z = 0
n=[0;0;1];
%Computing the cross product between these two vectors will give us a
%pivotal vector in the X Y plane:
Vpiv=cross(V3D,n);
```

Appendix B (continued)

```
%Then computing the cross product between this vector and the plane
normal
%again will result in the third vector of an orthogonal triad which is
the
%projection of the trajectory in the X Y plane:
Vp=cross(n,Vpiv);
%find the vector defined by the wheels' axis of rotation, we subtract
the X
%Y Z coordinates from their transformation Matrices
Vw1=[T11(1,4)-T10(1,4);T11(2,4)-T10(2,4);T11(3,4)-T10(3,4)];
Vw=cross(Vw1,n);
%Finding the angle between the two vectors
the=acos(dot(Vp,Vw)/(norm(Vp)*norm(Vw)));
```

WMRA_OR

```
%##########################################################################
####
% This M file creates optimized parameters for the weighted matrix
values
% of the wheels to account for rotation during the
% execution of a given trajectory.


% Function Declaration
function W1=WMRA_OR(the,i,n,bmax)
%if atan2(sin(the),cos(the))~=0
%W1=1;
%elseif atan2(sin(the),cos(the))<0.3
%W1=10;
bmin=-bmax;
W1=1/(((bmax-bmin)^2*(2*the-bmax-bmin))/(4*(bmax-the)^2*(the-bmin)^2));

end
```

WMRA_VR_Animation2

```
% This function does the animation of USF WMRA with 9 DOF using Virtual
Reality Toolbox.
% Function Declaration:
function WMRA_VR_Animation2(i, Twc, q)

% Declaring the global variables:
```

Appendix B (continued)

```
global L WMRA

% The initialization of the animation plot:
if i==1
    % Reading the Wheelchair's constant dimentions, all dimentions are
converted in millimeters:
    L=WMRA_WCD;
    % Opening the WMRA file:
    WMRA = vrworld('\9_wmra.wrl');
    open(WMRA);
    % Changing the View Point of the simulation:
    WMRA.DynamicView.translation=[Twc(1,4) 0 -Twc(2,4)];
    % Calculating the wheelaxle transform instead of the arm base
transform:
    Twc=Twc*[eye(3) -L(2:4) ; 0 0 0 1];
    % The orientation about Z of the wheelchair:
    phi=q(9);
    % Calculating wheelchair's wheels' angles:
    ql=q(8)/L(5)-L(1)*q(9)/(2*L(5));
    qr=q(8)/L(5)+L(1)*q(9)/(2*L(5));
    % Updating the VRML file for the new angles and distances:
    WMRA.Chair.translation=[Twc(1,4) Twc(2,4) L(5)];
    WMRA.Chair.rotation=[0 0 1 phi];
    WMRA.LWheel.rotation=[0 1 0 ql];
    WMRA.RWheel.rotation=[0 1 0 qr];
    WMRA.ARM2.rotation=[0 0 -1 q(1)];
    WMRA.ARM3.rotation=[0 1 0 q(2)];
    WMRA.ARM4.rotation=[0 0 -1 q(3)];
    WMRA.ARM5.rotation=[0 1 0 q(4)];
    WMRA.ARM6.rotation=[0 0 -1 q(5)];
    WMRA.ARM7.rotation=[0 1 0 q(6)];
    WMRA.ARM8.rotation=[0 0 -1 q(7)];
    % Viewing the simulation:
    view(WMRA);

% Closing the animation plot:
elseif i==3
    close(WMRA);
    delete(WMRA);

% Updating the animation plot:
else
    WMRA.DynamicView.translation=[Twc(1,4) 0 -Twc(2,4)];
    Twc=Twc*[eye(3) -L(2:4) ; 0 0 0 1];
    phi=q(9);
    ql=q(8)/L(5)-L(1)*q(9)/(2*L(5));
    qr=q(8)/L(5)+L(1)*q(9)/(2*L(5));
    WMRA.Chair.translation=[Twc(1,4) Twc(2,4) L(5)];
    WMRA.Chair.rotation=[0 0 1 phi];
    WMRA.LWheel.rotation=[0 1 0 ql];
    WMRA.RWheel.rotation=[0 1 0 qr];
    WMRA.ARM2.rotation=[0 0 -1 q(1)];
    WMRA.ARM3.rotation=[0 1 0 q(2)];
```

Appendix B (continued)

```
    WMRA.ARM4.rotation=[0 0 -1 q(3)];
    WMRA.ARM5.rotation=[0 1 0 q(4)];
    WMRA.ARM6.rotation=[0 0 -1 q(5)];
    WMRA.ARM7.rotation=[0 1 0 q(6)];
    WMRA.ARM8.rotation=[0 0 -1 q(7)];

end
```