ROBOTIC MOBILITY REHABILITATION SYSTEM USING VIRTUAL REALITY

BY RARES FLORIN BOIAN

A Dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Professor Grigore C. Burdea

and approved by

_____

_____

_____

_____

_____

New Brunswick, New Jersey

January, 2005

ABSTRACT OF THE DISSERTATION

Robotic Mobility Rehabilitation System Using Virtual Reality

by RARES FLORIN BOIAN

Dissertation Director: Professor Grigore C. Burdea

The application of robotic technology and virtual reality in rehabilitation is an emerging research field. The motivation behind the research efforts is the large patient populations in need for better rehabilitation therapy at lower costs. Walking impairments pose serious problems to the disabled when clearing a curb, walking on an uneven sidewalk surface, or crossing a street.

A common feature of the existing gait rehabilitation systems is the treadmill, which provides the patient with means to exercise walking on a smooth and continuous surface. However, in real life, the patient needs to negotiate uneven surfaces with various properties. The main subject of this thesis is the development of a walking simulator that can render a larger variety of walking surfaces. The simulator uses two compact Stewart platform robots attached to the patient's feet. The servo loop software controls simultaneously the positions and forces of each robot. The two robots simulate the functioning of a treadmill by moving one foot backward while the patient moves the other one forward. Using the 6DOF provided by each Stewart platform, the system can render uneven walking surfaces, and various surface conditions such as mud, ice, or gravel. A VR simulation of crossing a street has been developed and integrated with the simulator. The virtual environment supports several configurations to adjust the exercise difficulty to the patient's abilities, as well as provide him with a realistic exercising environment. The simulation configurable variables include shape and height

of the sidewalk edge, street surface condition, scene visibility, and vehicle behavior. The sensor readings output by the controller during the therapy are stored on the PC running the VR simulation and transmitted to an Oracle database for later remote consultation and evaluation.

The simulator is part of a distributed virtual reality rehabilitation framework. Several prototype rehabilitation systems using robotics and VR have been integrated in a common framework that supports telerehabilitation and telemonitoring. The framework consists of a network of servers. It provides the remote therapist with a simplified version of the VR exercise executed by the patient. The VR applications displayed to the patient and therapist are synchronized in real–time over the network. The therapist controls the parameters of the patient's exercise, through the interface provided by the simplified VR simulation. The rehabilitation framework also provides the integrated systems with a unified approach to data storage in an Oracle database. A web portal for the mobility simulator is currently under development to allow the therapist to evaluate the patient's progress using the clinical data collected during therapy.

Acknowledgements

I would like to express my thanks to Dr. Grigore Burdea for his support and guidance during my research years in the Human-Machine Interface Lab. He was an ever present driving force that offered advice and encouragement toward the completion of the thesis project.

I am grateful to Dr. Judith Deutsch for her valuable guidance in the design of the rehabilitation system. Her extensive experience helped create a much better system design.

Many thanks go to Dr. Mourad Bouzit for his invaluable insights into the mechanical and robotic control details of the mobility simulator. His experience and intuition in the field of robotics help speed up the development process by solving and preventing several problems.

I would like to thank Dr. Zoran Gajic for answering my questions regarding the stability of the robot controller. My thanks are also extended to Dr. Manish Parashar and Dr. Deborah Silver for taking time off their busy schedule to read and review my thesis.

I would also like to express my gratitude to Professors Alma Merians, Sergei Adamovich, Howard Poizner, Marilyn Tremaine, and Michael Recce. My collaboration with them on other research projects expanded my knowledge and understanding of physical therapy, neuroscience, and human-machine interfaces.

I would like to acknowledge the contributions made to this work by my colleagues Karen Erickson and Jeffrey Lewis. The collaboration with them was a great experience for which I am grateful. I would also like to thank all the colleagues I have worked with during my years at Rutgers. Among them, special thanks go to Dr. George Popescu, Chan Su Lee, Scott Winter, Hristiyan Kourtev, and Manjuladevi Kuttuva.

The research reported here was supported by grants from the National Science Foundation (BES 0201687 and BES 9708020) and from the New Jersey Commission for Science and Technology.

Dedication

To M, T, and C. You guys are the best.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The application of robotic technology and virtual reality in rehabilitation is an emerging research field. The motivation behind the research efforts is the large patient populations in need for better therapy. One such population is that of individuals post-stroke. According to the American Hearth Association, there are approximately 3 million stroke survivors in the United States who live with disabilities [4]. The ability to walk is one of several functions affected by stroke. Immediately after the stroke only 37% of the survivors are able to walk [63]. Of the patients with initial paralysis only 21% regain walking function [115]. Walking impairments pose serious problems to the recovering stroke survivors. Common daily activities such as clearing a curb, walking on an uneven sidewalk surface, or crossing a street in time become difficult due to lack of control over the feet and the auditory and visual interference of the surrounding environment [13].

Wade et al. [114] showed that training could reduce disabilities provoked by stroke even if the rehabilitation takes place in the chronic phase (9 months or more after the date of the stroke). In a literature review, Kwakkel showed that a small, but statistically significant, treatment effect was due to the intensity of the training procedures [67]. Longer and more intense training regimen proved to be more effective toward the recovery.

Although the research results say that prolonged and intensive rehabilitation is needed to treat patients post–stroke, the real life situation is different for most patients. Immediately after stroke, a patient goes through an intensive rehabilitation program that lasts approximately 40 days [59]. After that, the patient is discharged from the hospital and the rehab continues in outpatient sessions of half an hour once or twice a day. Eventually this is decreased to once or twice a week. In our view, the number of people still suffering of the post–stroke effects shows that the amount of conventional

rehabilitation provided by hospitals and clinics is not sufficient. Possible reasons for this are the high cost of the traditional therapy and limits placed in reimbursements.

Following this line of thought, researchers have focused on different approaches to provide rehabilitation for those in need at lower costs. Studies have shown that virtual environments (VE) are a technology suitable for rehabilitation therapy due to their inherent ability of simulating real–life tasks [49, 48] while minimizing the hazardous aspects of such activities. Besides helping to engage the patient in life–like activities at low cost and in safe conditions, virtual environments provide the means to better measure and evaluate the patient's performance. Data can be collected from the sensors attached to the patient's body and evaluated in real–time, providing performance feedback. In addition, the data can be stored transparently in web accessible databases for future evaluation [100, 98].

VEs can provide a patient safe and more controlled rehabilitation means. They can also solve a major drawback of the repetitive training required in post–stroke rehab: boredom. In conventional therapy, the patients usually get bored of doing the same exercises repeatedly, which leads to lower motivational levels. A lack of motivation toward the training program decreases the efficiency of the therapy. In virtual reality, every exercise can be wrapped in one or more game–like simulations that motivate the patient by making less obvious the monotonous aspect of the training [100, 58, 59]. Virtual worlds are characterized by a large degree of flexibility that can be used to adjust the exercises to the patient's needs.

VR–based rehabilitation can also be done remotely. The current advances in technology produced smaller and cheaper devices that can be deployed in a home. Not all the existing rehabilitation devices can be deployed at the patient home, in which case the rehabilitation can be done at a rural clinic while being monitored by a city expert.

1.2 Approach

Most of the gait rehabilitation systems currently used for therapy rely on treadmills. Such systems are the Biodex Gait Trainer, the Robomedica system, and the Lokomat®

system. Walking on a treadmill is similar to walking on a smooth floor indoors. In real life, a stroke survivor has to walk on more complex surfaces indoors and outdoors. While walking, a patient may have to negotiate stairs, uneven terrain, slippery surfaces, or mud. Treadmills are unsuitable for exercising walking in such conditions, because they lack the flexibility to render complex surfaces. The focus of this thesis is the design and development of a robotic system for gait rehabilitation (see Figure 1.1) that can render such surfaces.

A second goal is the creation of a telerehabilitation framework to integrate several existing VR–based rehabilitation systems built in the Human–Machine Interface Lab at Rutgers. The framework provides a remote therapist using one of the systems integrated with tools to monitor the activity of the patient and adjust the rehabilitation parameters on–line. It also provides unified data collection and storage routines, an Oracle database for the clinical data, and a web-based portal for data access.

1.3 Mobility Simulator

The setup of the mobility simulator is shown in Figure 1.1. It consists of two Rutgers Mega–Ankle (RMA) 6DOF robots, an unweighing frame, and a large screen display. The two RMA robots are bolted to a wood platform on the floor so they do not slip during exercises. On the same platform, the Biodex unweighing frame is positioned above the two RMA robots, the vertical beams of the frame being approximately aligned with the platforms. The patient, suspended in the unweighing frame, has each foot secured to the corresponding RMA. At approximately 1.5 meters in front of the patient, the virtual reality exercise is rendered on the large projection screen display. The two robots simulate walking in a way similar to the functioning of a treadmill: the robot connected to the supporting foot slides backward while the robot attached to the swinging foot follows the motion compensating for its own weight. Using the 6DOF provided by the Stewart platform architecture of the RMA robots, the system can render uneven walking surfaces, and various surface conditions such as mud, ice, or gravel.

Figure 1.1: Mobility simulator [13].

1.4 VR Rehabilitation Framework

The framework design is inspired from the system developed by Popescu [100] hand rehabilitation. The architecture can be divided in four logical sites that map over multiple different physical locations (Figure 1.2): rehabilitation site, monitoring server site, data storage site, and monitoring site.

1.4.1 Rehabilitation Site

The rehabilitation site is the location where a patient goes through the therapy program. The system components deployed here are sensing and haptic devices (i.e. the mobility simulator, the ankle-in-sitting system, etc.), virtual reality exercises, graphics workstation and audio/video capture instruments for remote communication. Using the sensing and haptic hardware, the patient interacts with virtual reality exercises running on the graphics workstation. The VR exercises are game–like simulations that engage

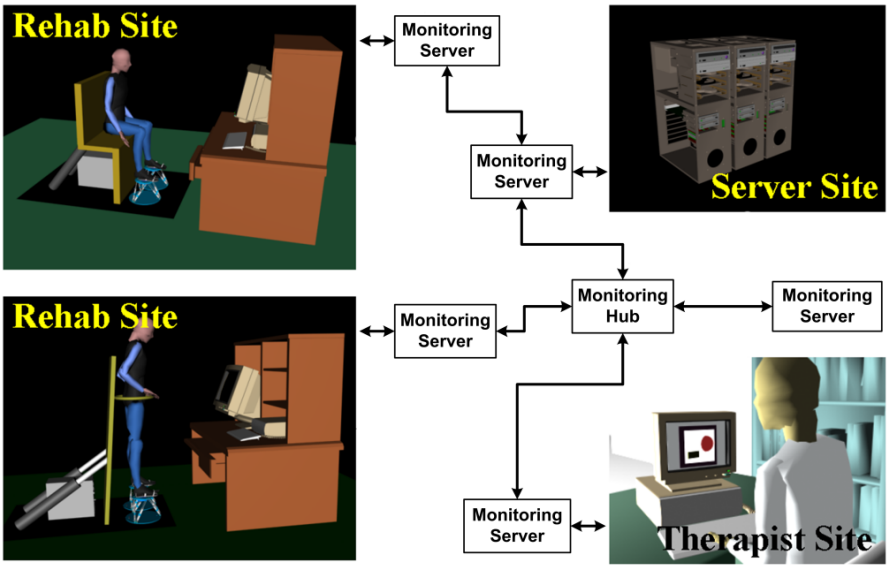


Figure 1.2: Telerehabilitation framework sites.

the patient into winning the games' challenges. In order to win the game, the patient has to execute a variety of motions used in physical rehabilitation. In our opinion, by mapping these procedures on games, the therapy becomes more engaging and the patient is kept motivated throughout the entire course of the rehabilitation.

The system collects and evaluates the data output by the sensing devices used during therapy. These data are evaluated in real–time and displayed on the computer screen as real–time performance feedback to the patient. The raw and evaluated data are also stored on the hard–drive for future evaluation by the therapist or physician.

The therapist overseeing the therapy can do that from a remote location. The patient and the therapist are connected over the Internet through teleconferencing using the microphone and video camera deployed at the rehabilitation and monitoring sites. This separation between the patient and therapist has not been achieved for the mobility simulator presented in this thesis. However, successful tests have been conducted for the hand and ankle rehabilitation systems developed at Rutgers in collaboration with the University of Medicine and Dentistry of New Jersey and the New Jersey Institute for Science and Technology.

1.4.2 Monitoring Server Site

Although teleconferencing could provide the therapist and patient with a way of interacting audibly and visually, it cannot provide the therapist with real–time information on the VR exercises progress. Such information would have to include everything that the patient sees on the workstation screen. To make this information available to the therapist, a distributed real–time monitoring service has been developed. The service transfers the therapy exercise data from the rehabilitation site to the monitoring site.

To avoid overloading the graphical workstation, the therapist site does not connect directly to the service running on the rehabilitation computer to retrieve the exercise data. Instead, the rehab workstation sends the data in real–time to a server, which then makes it available to the therapist site. The location of the monitoring server proved to be critical in an environment consisting of heterogeneous network connections. To avoid situations where the patient and the therapist are connected on the same network but the server is on another (hence reducing the quality of service), multiple server sites have been deployed. The data generated by the rehabilitation site is multicast to all the server nodes involved. Thus, the remote monitoring application can choose an optimal connection from the several possible options.

1.4.3 Data Storage Site

The data storage site is the location of central servers coordinating the distributed rehabilitation system. This site consists of an Oracle database server, a web server running the data access portal and the monitoring service main hub.

The Oracle database stores the clinical data collected at the rehabilitation site during the therapy sessions. This database and the rehabilitation sites are synchronized daily.

The web data access portal provides the therapist and physician with the means of analyzing the progress of the patient during therapy and the possibility of consulting data at several levels of granularity.

The monitoring service hub is responsible for coordinating the network of server sites multicasting the real–time patient data.

1.4.4 Monitoring Site

The monitoring site is the location where the therapist supervises the patient's activity. Physically, it can be any network–connected computer. Using the monitoring application presents the therapist with a simplified version of the exercise executed by the patient. The application window displays all the parameters involved in the therapy session along with the animated 3D simulation of the patient's activities. Work done by Lewis and Boian has added the ability to change the exercise parameters remotely. Thus, the therapist can preconfigured or adjust the simulation variables during the rehabilitation session without being present at the patient's location [70].

1.5 Contributions

The work presented in this thesis can be grouped in four parts: design and development of a robotic locomotion simulator, development of a virtual reality simulation integrated with the simulator, design and development of a distributed monitoring service for telerehabilitation support, and the development of an Oracle database and web portal for storage and access of clinical data. This thesis makes contributions in the following fields: haptic devices, virtual environments, haptic modeling, virtual rehabilitation, and telerehabilitation. The list below summarizes these contributions by field.

1. *Haptic devices*
    - Dual Stewart platform mobility simulator;
    - Design and control of the robotic devices to sustain the weight of a person during walking.
2. *Robotic control*
    - Simultaneous dual robot control in time–sharing control loops.
3. *Virtual environments*:
    - Street crossing simulation integrated with the mobility simulator for rehabilitation of patients post–stroke. The simulation integrates visual, haptic

and auditory feedback to provide the patient with a realistic experience.

4. *Haptic modeling*:

   - Simulation of a person walking using two Rutgers Mega–Ankle robots. Definition and implementation of novel haptic materials and surface patches used for modeling the properties of the virtual walking surface.

5. *Telerehabilitation*:

   - Distributed telerehabilitation framework, including web–based data access. The framework has been designed to support the development of PC–based telerehabilitation exercises by providing the infrastructure necessary for data collection, storage, evaluation and streaming.

1.6 Thesis Outline

The material presented in this thesis is structured in eleven chapters. Chapters 3, 4, 5, 6, 7, and 8 discuss the mobility simulator. Chapters 9 and 10 present the design and development of the VR rehabilitation framework.

Chapter 2 gives an overview of the research state of the art in applying robotics and virtual reality in rehabilitation.

Chapter 3 presents the hardware components of the rehabilitation system developed as part of the thesis research. It provides an overview of the hardware as well as in depth descriptions of the custom made parts and the design decisions behind them.

Chapter 4 studies the kinematics and dynamics of the Rutgers Mega–Ankle (RMA) Stewart platform. The workspace, maximum output force and torque as well as mechanical bandwidth and dual platform interference issues are discussed.

Chapter 5 presents the RMA robots' servo control. The approaches and problems raised by the pressure, position and force control are also discussed. An overview of the software is also given with details about the implementation of the time–sharing between the tasks controlling each platform.

Chapter 6 studies the task level control of the mobility simulator. The implementation of walking and the haptic effects designed to support the simulation of various walking surface properties are described.

Chapter 7 presents the virtual reality exercise developed as part of the system. The exercise consists of a street crossing simulation integrated with the mobility simulator.

Chapter 8 presents preliminary results from tests of the system on healthy individuals.

Chapter 9 describes the real–time remote monitoring service developed to provide the therapist with access to and control over the therapy session without being at the same geographical location with the patient.

Chapter 10 presents the data storage and access part of the system. The database design, along with performance and security issues are also discussed. The web–based portal is also described, with details about the graphical user interface (GUI) and the graph generation from the data stored during the therapy.

Conclusions and future work directions are given in the Chapter 11.

Chapter 2

Robotics and Virtual Reality in Rehabilitation

The system presented in this thesis system is multidisciplinary integrating virtual reality, rehabilitation, telerehabilitation, robotic devices, and web–accessible databases. Some of the significant work and results in these fields are presented in the next sections.

2.1 VR–based Rehabilitation

Virtual reality can be used in rehabilitation process as a complement to the conventional therapy or as a stand–alone intervention approach that replaces the conventional therapy. The former case is known as *VR–augmented rehabilitation* and the latter as *VR-based rehabilitation* [25]. The mobility simulator system presented here belongs to the VR-based rehabilitation category. While our system targets rehabilitation of gait primarily in patients post–stroke, other research projects address rehabilitation of function in hand, arm, ankle, etc. Several such state of the art systems are presented below.

The system presented in this thesis is an extension of the work done by Girone et al. in 1999 at Rutgers University [42]. Girone and Bouzit built the first prototype of the air actuated Stewart platform called the "Rutgers Ankle". Using one platform, Girone developed a VR exercise for ankle diagnosis of orthopedic patients. The simulation displayed a virtual leg mapped to the position and orientation of the patient's leg. The motions of the virtual model were augmented so that they provided a clear visual feedback when the patient. The ankle orientation/torque data was collected in real–time and stored in a local database. Proof of concept patient trials were run on this system with encouraging results [43, 44]. Using the same setup Latonio and Deutsch developed an exercise that was more engaging and better addressed a larger number of motions targeted by the therapy. The patient used the Stewart platform as a joystick to fly a

virtual airplane through hoops using his ankle. The hoops were scattered in the sky following a pattern chosen by the therapist (Figure B.5). The system was applied to both orthopedic and post–stroke patients. The results of the trials conducted by Deutsch in Summer 2000 in a clinic showed improvement in both types of patients [35, 36]. The systems used by Girone and Latonio had the rehabilitation functionality reduced to the patient exercising in sitting one foot at a time and required the presence of a therapist at the rehabilitation site. The system did not support multiple simultaneous rehabilitation sessions. In addition, no remote data access or remote patient monitoring capabilities were provided.

Popescu developed a similar system that addressed the rehabilitation of patients with hand injury [100]. The system used the Rutgers Master II haptic glove [99, 45] developed in the Human–Machine Interface Lab at Rutgers University. VR exercises were developed to simulate the use of a DigiKey device, squeezing of a rubber ball, or sticking pegs in a board with holes (PegBoard) [98]. The architecture designed by Popescu served as a base for Girone's and Latonio's systems. All these three systems allowed one to one patient/therapist interaction and classical client–server remote data access. The rehabilitation site was in California (Stanford University) or in New Jersey with data being stored locally and accessed remotely from Rutgers.

In 2000, Jack and Boian adapted Popescu's idea and implemented a system to be used by patients post–stroke [78, 3]. The system involved the use of two sensing gloves: the CyberGlove by Immersion technologies Inc. and the Rutgers Master II glove [45, 26, 25, 23, 22] developed at Rutgers. The initial version of the system was a proof of concept implementation that targeted mainly high quality rehabilitation exercises and on–line data access. The system did not provide real–time remote monitoring of the patient, concurrent patient rehabilitation sessions, or web–based data access. In 2001, Boian redesigned the system for concurrent access and implemented real–time web–based monitoring of the patients activities. The second–generation hand exercises are presented in Appendix B. The system has gone through one month of patient trials and the results were published at the MMVR2002 conference [12]. A significant part of this system's architecture will be used in the system proposed here.

Hogan and Krebs, at MIT, developed a robotic arm used for post–stroke rehabilitation [65]. The patient was interacting with a simplified video game by using the robotic arm. The goal was to have the patient move the robot end–effector according to the game. If the patient could not execute a task, the robotic arm would help by guiding the patient's hand.

In 2000, Riva integrated a commercially available gait–inducing exoskeleton in a virtual environment. The system was used for rehabilitating patients with spinal chord injuries. The patient wearing an HMD had to walk in a virtual environment simulating a stroll in the mountains [103].

More recently, Jaffe developed a system that uses virtual reality to improve walking in stroke patients. Although the goal of this project is identical with ours, the two systems are fundamentally different. Jaffe used [60] a treadmill on which the patients were walking wearing a head mounted display. The patient had to walk and avoid the virtual obstacles generated by the computer on their path. The treadmill limits the simulation to flat terrain and no special surfaces (ice, mud, etc)

Brown et al. [20] developed virtual environments for travel training. Their system simulates a city featuring a cafe, a house, a supermarket, and traffic on the street. The user input is done through the keyboard, the goal of the system being only the visual navigation.

2.2 VR Telerehabilitation

According to the Centers for Medicare and Medicaid Services [39], telemedicine is defined as

> ... the use of communication equipment to link health care practitioners and patients in different locations. This technology is used by health care providers for many reasons, including increased cost efficiency, reduced transportation expenses, improved patient access to specialists and mental health providers, improved quality of care, and better communication among providers.

Within the broad definition of telemedicine, telerehabilitation is a branch concerned with providing rehabilitation therapy without having the patients and therapists at same geographical location. Several state of the art telerehabilitation research projects are discussed below.

Among the systems presented in the previous section, those developed by Girone, Latonio and Jack implemented no telerehabilitation utilities. The presence of a therapist in the same room with the patient was necessary.

The system developed by Popescu and Boian presented tools that allowed the therapist to interact remotely with the patient through teleconferencing, remote web–monitoring and remote haptic interaction [97].

In 2000, Grimes et al. developed a web–based hand evaluation system. The diagnostic device is a dynamometer connected to a PC host. The data collected from the dynamometer is displayed at the clinic site to the therapist. A NetMeeting teleconference session is open between the therapist and the patient during the evaluation period. The patient interacts with the therapist using a microphone and a camera [46].

In 1998, Pfeifer et al. developed a client/server architecture for remote patient monitoring [95]. The system was used to measure patient's vital signs once a day. The data collected was stored in a database located at the clinic. The main idea was that active involvement of the patient in the therapy would translate into better outcomes of the therapy. Hence, the patient was in charge of manipulating the monitoring system by interacting with the computer through a touch sensitive screen. Pfeifer's system featured no VR components.

In 2003, Holden et al. [50] developed a rehabilitation system for patients post–stroke involving a one to one remote interaction metaphor between a patient at home and a therapist at the clinic. While communicating with the therapist through teleconferencing, the patient had to move their arms following the paths presented by the VR simulation. Magnetic sensors were used to capture the patient's motions and the data was stored transparently in a database. Their system was deployed and tested by Piron et al. in Italy [96].

2.3 Locomotion Simulators

Locomotion simulators, which attempt to simulate the sensations of walking, have been the focus of many researchers due to their applicability in simulating real–life tasks. The existing gait simulators have been classified by Hollerbach [107] into three categories based on the design: walk–in–place devices, foot platforms and treadmills.

2.3.1 Walk–in–Place Devices

The general idea behind these devices is to have the user walking in place without advancing while the motions are tracked by sensors. From the user's motion, the driving workstation computes the direction and speed of the virtual avatar and changes the view in the virtual environment. The haptic feedback of these systems is reduced to the floor contact, which in most cases is a flat surface.

One such is system the Gaiter developed by Templeman et al [112]. Magnetic trackers are attached to the user's thighs, waist, head and hand. The direction and speed of motion are computed from the user's knee movements. The orientation of the body is measured by the waist sensors. Force sensors are placed on the user's footpads to help detect the steps more accurately.

A similar approach by Parsons et al. [89] used magnetic trackers attached to the ankles. A step was generated if the ankle was raised above a threshold.

Iwata tried to simulate walking by using low–friction shoes [57, 55]. The user was attached to a frame by a belt around the waist to counteract the forward motion and to improve safety. A toe pad was used to break the sliding.

In 2002, Bouguilla et al. developed a turntable for walking simulation in large–scale virtual environments [17]. The user stepped in place on a turntable platform while facing a large screen displaying the virtual environment. Pressure sensors were mounted on the platform to detect the user's position during the simulation. The turntable was actuated and rotated to cancel out the user's change of direction keeping him/her facing the screen.

2.3.2 Treadmills

A special treadmill with controllable tilt called the Sarcos Treadport (Figure 2.1(a)) was developed at the University of Utah. The novelty of this treadmill versus the conventional ones is the possibility to simulate steeper up–hill walking by increasing the tilt beyond the approximately 5 degrees in conventional treadmills. Using a stick attached to the user's back, the Treadport can also be used to simulate inertial forces [27, 52].

Iwata also approached the idea of using a treadmill for walking simulation. His research focused on creating a treadmill that allowed the user to walk in any direction [54]. The Torus Treadmill (Figure 2.1(b)) is an assembly of ten conveyer belts that can move in any direction. The speed of motion is limited to about 0.5 m/sec, which corresponds to slow walking.

Miyasato developed another treadmill–like locomotion simulator called ATR–GSS (Figure 2.1(c)). A regular treadmill was modified with controllable panels that can be raised or lowered to simulate uneven surfaces. After the user steps on the raised/lowered panel, as the user's foot is moved backward by the belt, the panel is brought back to its original position [81, 85]. Stair–like surfaces are very appropriate for rendering on this system.

Wang et al. at Arizona State University [116] developed a simple but interesting omni–directional treadmill using a low friction cloth on top of a rigid board. Casters with high friction wheels are placed on top of the cloth pressing it against the board. By rotating the casters, the system controls the motion of the cloth moving it in the opposite direction of the user's motion, essentially pulling the carpet under the user's feet.

2.3.3 Foot Platforms

At the University of Utah, Hollerbach and Christensen designed the Sarcos Biport [107]. The device is made of two hydraulic 3DOF platforms attached to the user's feet. When the user's foot is swinging forward taking the next step, the platforms are controlled to

(a) Sarcos Treadport

(b) Torus Treadmill

(c) ATR GSS

Figure 2.1: Treadmill–based walking simulators [51].

compensate for their own weight. The virtual environment is displayed on a wall screen in front of the user (Figure 2.2(a)).

In Japan, Iwata developed two generations of foot platforms named GaitMaster and GaitMaster2 [56, 87]. As can be seen in Figure 2.2(b) the GaitMaster is an assembly of two 3DOF platforms mounted on a turntable. The two bases follow the motion of the user's feet using the input provided by trackers mounted on the user's legs, hence simulating walking on an infinite floor. The second–generation device, GaitMaster2 (Figure 2.2(c)), is made of two 2DOF motion platforms (one for each foot). The platforms are actuated by motors and chains [87]. As in the initial GaitMaster, the

platforms follow the users foot using magnetic trackers.

(a) Sarcos Biport

(b) GaitMaster

(c) GaitMaster2

Figure 2.2: Foot platforms [51].

The foot–platforms systems presented here are superior to treadmills in that uneven terrain can be simulated. However, they are extremely bulky and complex, which makes them difficult to install outside the lab.

2.3.4 Other Walking Simulators

A very realistic simulation of uneven terrain was created by Noma through the development of the Terrain Surface Simulator ALF [85]. The simulator is a rectangular surface made of many small tiltable plates that can be controlled in real–time. By tilting these small plates, the walking surface can be set in a large variety of shapes. The device is not a treadmill so the user can only walk around in the actuated area (see Figure 2.3(a)).

A novel approach that combines walking simulation with immersive VR is the entertainment sphere manufactured by VR Systems UK (North Baddesley, Southampton, UK), shown in Figure 2.3(b). The concept shows the user inside a movable sphere that rotates under his/her footsteps. Although infinite in size, the walking surface is always smooth, and depending on the diameter of the sphere is more or less curved. Larger spheres will yield results that are more realistic by reducing the unnatural curvature of the surface. The virtual environment is displayed on the same sphere using outside

projectors.

E–motek Inc. (Amsterdam, Netherlands) has developed the CAREN system, a hydraulically actuated Stewart platform robot for simulating surfaces with any tilt angle (see Figure 2.3(c)). The user stands on a 2–meter diameter board placed on top of such a platform and can exercise balance while the platform changes its orientation to match the virtual world scenario.

In 2003, Comeau et al. developed Laval University a gait rehabilitation system using CAREN [30]. They placed a treadmill on top of the platform to make the walking surface infinite. The user's steps are measured by magnetic trackers mounted on the feet. To prevent undesired interference with the sensors, the treadmill is made exclusively of non–magnetic materials. This device cannot simulate uneven terrain or stairs but it is very appropriate for indoor walking simulations.

2.4 Rendering for Walking Simulations

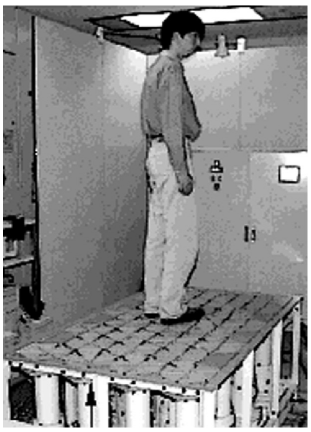
The primary concern of a simulation of a user walking is to update of the visual, audio and haptic devices according to the user's motions. A second aspect that makes a simulation more realistic is the walking terrain rendering. In the following sections, we present some of the existing solutions to these issues.

2.4.1 Detecting User Motions

When the user is walking in the virtual environment using the simulator, the rendering engine has to track his/her motions in order to update the viewpoint and the hardware components. For example, if the user turns around, the scene has to move in the opposite direction. In addition, the haptic devices have to adapt so that the user has the turning feeling while still facing the display.

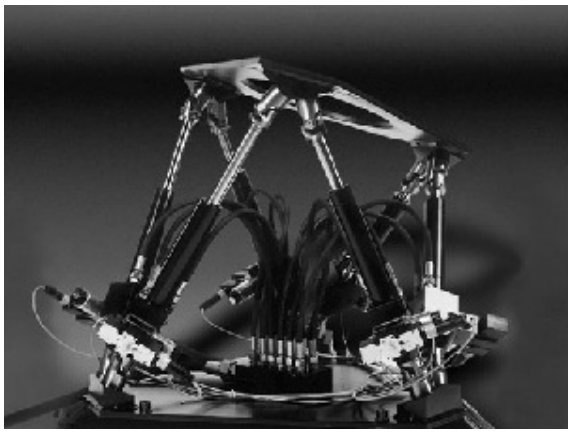
The systems presented in the previous sections use tracking approaches such as:

- Magnetic trackers attached to the user's body [112];
- Optical trackers pointed at the user's feet;



(a) Terrain Surface Simulator ALF [84]

(b) Immersive sphere by VR Systems UK



(c) CAREN by E–Motek Inc.

Figure 2.3: Other walking simulators.

- Platforms attached to the user's feet [107, 56, 87];
- Force/pressure sensors on the floor to detect the feet position [17];
- Rigid stick attached to the user's back [27, 52].

Based on the sensor readings, the criteria for deciding whether the user changes the direction was based on:

- Knee lateral movement;
- Foot orientation;

- Body rotation/movement to a side.

2.4.2 Haptic Rendering

Change of direction: Haptic support for the user's change of direction can be implemented in a number of ways. In the case of the Gaiter [112], the user can turn around on the floor and the only thing that needs to be updated is the view in the virtual world displayed in the HMD. On the other hand, Bouguila's system displays the environment on a wall screen, which requires a way to keep the user facing it regardless of the change of direction. This was solved by rotating the turntable to cancel out the user's change in orientation. In both these cases, the user needs to signal the turn through a special foot gesture.

Treadmills, such as the Sarcos Treadport, can implement turning support by having a turntable under the treadmill that rotates as the user changes direction. This approach was explored by Hollerbach et al. in the design of the Sarcos Uniport, and by Comeau et al. using the CAREN system. The Torus Treadmill developed by Iwata solves this problem at the cost of motion speed, user workspace, and system complexity.

Similar to the treadmills, the foot platforms cannot usually support turning by themselves. A solution for this problem was given by Iwata with the GaitMaster. The two 3DOF platforms of the GaitMaster are mounted on a turntable, which rotates following the user's motion.

Slope: The Sarcos Treadport renders a slope in a very natural way by tilting the treadmill to a certain angle. In addition, the tether stick attached to the user's back can apply inertial forces making the simulation even more realistic. The CAREN system solves this issue just as naturally as the Treadport by using the 6DOF of the Stewart platform.

The ATR–GSS developed by Miyasato solves the slope rendering problem with a stair climbing approach. The plates under the treadmill's belt are raised in front of the user and, after contact with the foot, they start to lower as they are moved back.

Terrain shape and stiffness: Uneven terrain can be simulated with the ATR–GSS just as easily as simulating stair climbing. The ATR–ALF [85] makes a more realistic

rendering of uneven terrain by changing the shape of the walking surface.

Foot platforms are a suitable option for various terrains rendering because of their inherent mobility. The Sarcos Biport can simulate bumps in the terrain by changing the platforms' elevation. Besides terrain shape, foot platforms allow rendering of terrain stiffness (corresponding to mud or gravel) by changing the platform impedance.

Chapter 3

Hardware Architecture

The mobility simulator employs two Rutgers Mega–Ankle robots, a electro–pneumatic controller, an unweighing system (Biodex Co.), a graphics workstation, and a large screen display to allow the patients exercise in standing 1.1.

The two RMA robots are connected electronically and pneumatically to the haptic control interface (HCI) (not shown in Figure 1.1). The HCI runs the servo controller software, which regulates the air pressure and the position of each double–acting pneumatic cylinder actuating the RMAs. The HCI is connected to the graphics workstation over the serial port, and receives from the VR simulation commands regarding the haptic feedback to the user's feet. The HCI also returns the sensors readings of each RMA platform to the VR simulation.

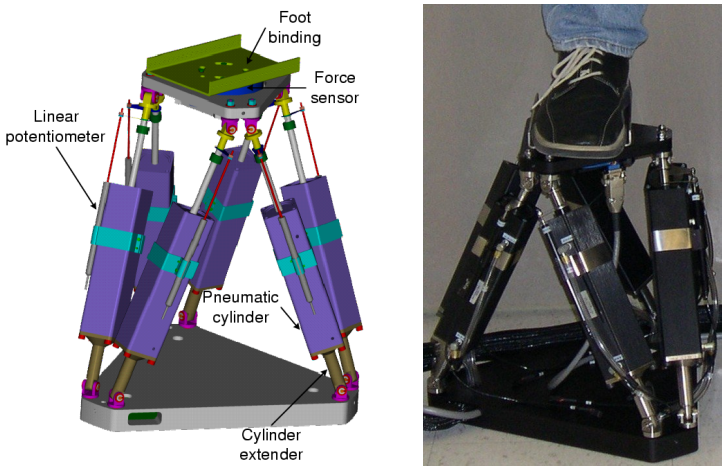
The graphics workstation is currently a dual processor computer running the VR therapy exercises. Based on the input received from the HCI, it updates the virtual environment and displays the real–time performance evaluation of the patient's activity.

3.1 Rutgers Mega–Ankle Stewart Platform

The Rutgers Mega–Ankle is a compact hexapod custom–made robotic device (Figure 3.1(b)) commonly called "Stewart platform" after its inventor [110]. The robot consists of two plates (bases) interconnected by six pneumatic cylinders positioned in a zigzag sequence. The large base is fixed (bolted to the floor) while the top plate is mobile and can be moved and rotated in all directions by changing the individual position of each of the six actuating cylinders. The main advantages of the Stewart platform architecture are the good size–to–power ratio provided by these robots and their six degrees of freedom. The platform was designed to have the user's foot strapped into the binding mounted on the mobile base. By controlling the air pressure in the pneumatic cylinders, the platform can apply 6DOF forces and torques to the user's foot and

control the user's foot position. The forces and torques at the mobile platform base are directly measured by a 6DOF force sensor (JR3 CO.) mounted under the foot binding. The position of the platform is computed from the displacements of the six cylinders measured by linear potentiometers attached to each of them. The air is brought to the upper and lower chambers of the cylinders by pairs of flexible tubing that are connected to the HCI.

The pneumatic actuators were chosen against the more powerful hydraulic ones because they were smaller, cleaner and safer. Both approaches have fluid escapes but a hydraulic oil leakage can be physically dangerous to the user and also very messy. Another possible solution would have been electrical actuators, but they have a lower size–to–power ratio and cannot maintain continuous forces without overheating. In the case of the mobility simulator, continuous high forces are necessary to sustain the user's weight.



(a) RMA CAD design (b) RMA Prototype

Figure 3.1: The Rutgers Mega–Ankle Stewart Platform [15].

3.1.1 Hardware Design

The Rutgers Mega–Ankle is similar in structure to the initial, Rutgers Ankle version developed by Girone and Bouzit [42] (Figure 3.2). The initial platform could not generate enough force and torque to be used in standing. Although the standing system

uses two platforms, each of them has to be able to handle the weight of a person by itself because walking involves shifting one's weight from one foot to the other. The smaller version of the platform could lift at most 75 kg of static weight. The torque output of the platforms was much less, so the maximum load the initial platform could balance was around 20 kg. Making a Stewart platform powerful enough to lift and balance a patient is easily done by using larger actuators. However, the RMA robot has to be small enough to be easily deployed, and have little friction in order to achieve a usable mechanical bandwidth. These requirements bring along several limitations.

Figure 3.2: Rutgers Ankle (RA) robot [14].

The lateral distance between the user's feet while using the simulator imposes a constraint on the RMA robot's footprint dimensions. The height of the robots has to be limited as well so that taller users can fit on the system without interference with the top cross bar of the unweighing frame. In addition, an increased robot height can potentially make the user uncomfortable inducing stress.

The low friction requirement limits even more the choices for actuators, because low friction pneumatic cylinders are usually available only in small sizes that are not powerful enough for the mobility simulator.

Pneumatic Cylinders

The solution to increase the lifting capability of the RMA robot is to use larger, more powerful pneumatic cylinders. Besides having a larger force output, the new cylinders

have to be frictionless and as small as possible. The model that best fit these requirements was the M32 cylinder produced by Airpot Inc. (Norwalk, CT) Figure 3.3 shows side by side the new M32 model and the E16 model used by the smaller RA version of the platform. The new cylinder is visibly larger and can output 572 N of force, which is four times more than that of the E16 model. This means that, roughly, the new platform is able to lift four times more weight than the old one. Section 4.6 presents in detail the new platform's load–handling capabilities. In spite of being longer, the M32 model still has the same 100 mm stroke as the E16 model. Both models support air pressures up to 100 PSI.

(a) Model M32 (b) Model E16

Figure 3.3: Pneumatic cylinders from Airpot Inc.

The new M32 cylinders are not only longer but also have a larger diameter than model E16. To avoid interference between neighboring cylinders at their base mountings they had to be distanced farther apart. For a Stewart platform to have a maximal lateral workspace, the distance between neighboring mountings has to be as small as possible. The initial version of the platform had a distance of 25.4 mm between two neighboring cylinders due to their small diameter. The M32 model has a square cross–section of 50 mm by 50 mm. To avoid interference, the M32 cylinders should be mounted at a distance of at least the diameter of the circle circumscribing the cross–section. The diameter of that circle is approximately 70 mm. To reduce this distance, and by that to increase the lateral workspace, the cylinders have to be attached to thinner extenders that lift the bulky part of the cylinder (Figure 3.1(b)). Because the cylinders are always at an angle, the extenders move the large diameter parts in positions where they are far enough from each other. The trade–off is that the platform becomes higher and

possibly less comfortable to use. The new platform's 458 mm height is almost double that of the RA design.

Cylinder/Potentiometer Assembly

Each of the six cylinders has a linear potentiometer attached to it to measure the shaft displacement (Figure 3.1(b)). The potentiometer has to be very well aligned with the cylinder so that the dual–shaft assembly can move without friction. Because both cylinder and potentiometer shafts have a tight fit in their respective bodies, the smallest misalignment can cause the assembly to get stuck and eventually break. The fixed parts of the cylinder and the potentiometer are connected with wide and rigid brackets that keep the two bodies parallel. The connection of the two mobile shafts is more complex because it has to overcome the inherent slight misalignments. The solution is a flexible mobile link (Figure 3.4) that can bend along the motion axis and swivels a little around it. The free motion allowed by this link still has to be restricted to prevent the potentiometer shaft from being forced in extreme positions.

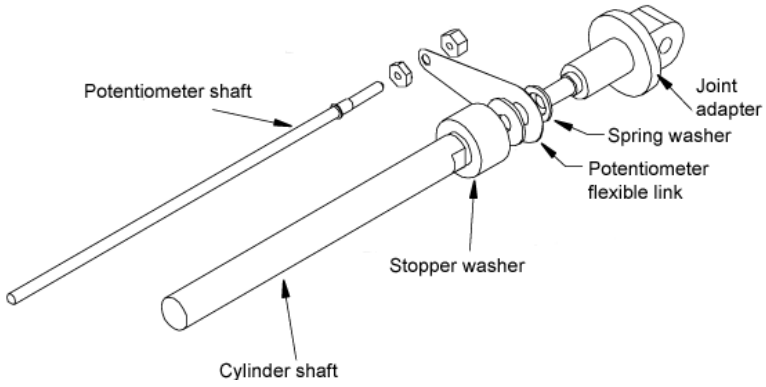


Figure 3.4: Cylinder/potentiometer mobile assembly.

As mentioned before, the cylinder extender has the role of making the assembly thinner at the joint region so that the mountings on the fixed base can be closer. At the end of the cylinder shaft, the joint adapter also has an extension that prevents the interference between the mobile links of two neighboring assemblies. The role of the spring washer above the mobile link is to prevent it from getting stuck due to machining residue around the inner edges. The stopper washer is designed to prevent the cylinder

from being broken by not allowing the shaft reach the bottom when being slammed close.

After the Rutgers Mega–Ankle has been built and entered the controller development phase, it became apparent that the mobile links of neighboring cylinders were interfering, causing damage to the linear potentiometers. To avoid such problems, the design of the mobile link has been changed so that the linear potentiometer connection was lowered relative to the cylinder body, hence increasing the space between neighboring assemblies (Figure 3.5).

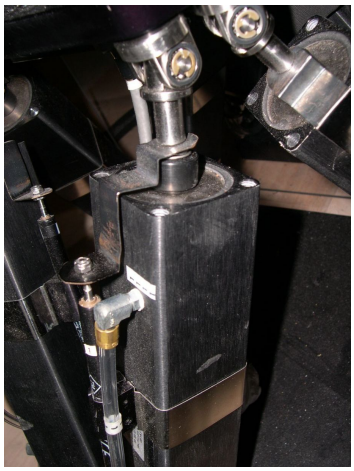


Figure 3.5: New mobile link detail.

Joint Swiveling Range

The joints of the Rutgers Mega–Ankle robot are 2DOF universal joints custom designed to be small and frictionless. An important aspect of the joint design is the swiveling blocker that restricts the joint's motion (Figure 3.6). Without it, the joint range is more than 180 deg, which allows the cylinders or the mobile base to get in singular positions from where the controller cannot bring it back.

For the joints attached to the mobile base, the swiveling blocking mechanism is designed as two discs, one around the joint adapter and one around the joint fork. The position of the disc along the adapter's axis and the radii of the two discs are computed so that when the joint is rotated to the extreme, they hit each other. The blocker reduces each side of the joint range of motion by 7 degrees leaving a maximum rotation

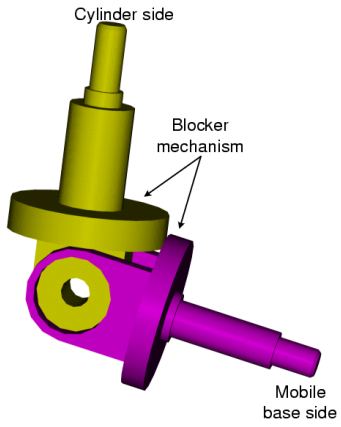


Figure 3.6: Joint range stopper.

range of 166 deg.

The fixed base joint blockers were not part of the initial design. However, due to the slight differences between the CAD design and the prototype, the unconstrained range of the bottom joints allowed the cylinders to reach vertical orientations. A force applied in a direction parallel to the joint's horizontal axes, while the cylinder is in a vertical position, will not cause any torque to rotate the cylinder around its vertical axis, hence the assembly will not move and the joint will be forced in a direction it was not designed to support. The forces applied by a human foot to the RMA robot during walking are considerable and could damage it if ever applied in the situation described above. To prevent such events from happening, joint blockers have been built (Figure 3.7) and attached to the cylinder extender. The joint range has been reduced by 10 deg, which is enough to cause the cylinder to rotate under a horizontal force parallel to the joint's axle.

Fixed Base

The fixed base is the largest part of the platform (Figure 3.8). It sits on the floor on rubber strips and it is bolted down to prevent it from sliding. Inside, the base has a cavity where the tubing and wires coming from the cylinders and potentiometers are connected together. These connections are protected from outside interference by a lid. The six joint mounting holes are placed on a circle of 203.2 mm radius and neighboring

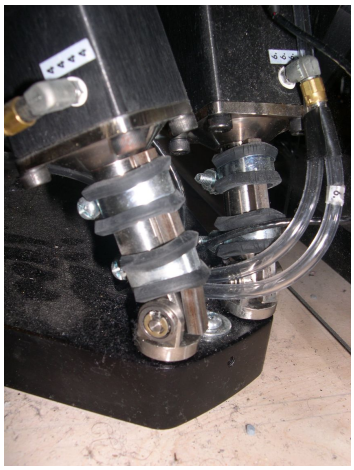
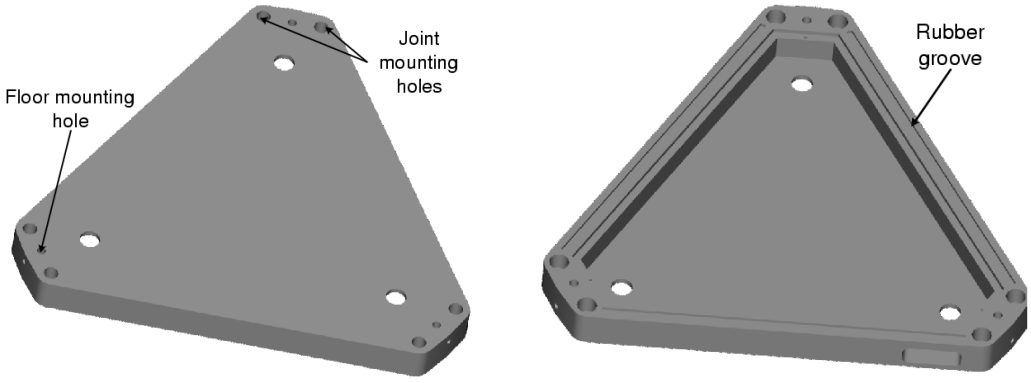


Figure 3.7: Lower joint range blockers detail.

holes are positioned 50.8 mm apart. This is less than the 70 mm necessary without using the extenders but it is still twice as far compared to the distance in the RA robot.



(a) Top view (b) Bottom view

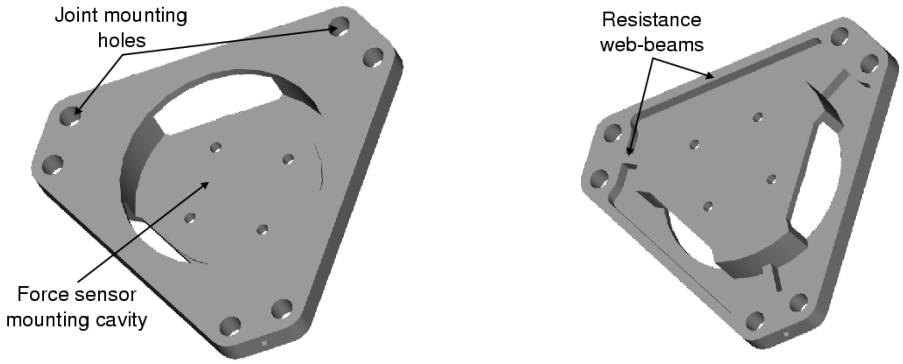
Figure 3.8: Fixed base.

Although larger, the distance between the new platform's joint mountings has little effect on the workspace because in comparison to the distant joints this distance is still small enough. However, it does have an effect on the kinematic model. The RA platform used a simplified, 3–3 model (Figure 4.1(c)) that considered neighboring joint positions coincidental. Although this model yielded low positional errors, it is not fit for the RMA design where the distance is considerably larger. The shape of the base was changed from the RA's circular version to a triangular one to allow the platforms

to be positioned closer to each other. See Section 3.1.2 for details.

Mobile Base

The mobile base closes the loops of the Stewart platform robot connecting together the mobile ends of the actuators. In general, it has to be lightweight to avoid additional load and strong enough to support the forces applied at the end–effector. In the Rutgers Mega–Ankle case, the mobile base has to provide mounting means for the JR3 sensor and has to be as small and light as possible.



(a) Top view (b) Bottom view

Figure 3.9: Mobile base.

As shown in Figure 3.9, the mobile base has a bucket like structure for mounting the force sensor. Lowering the force sensor in the mobile base saves about 20 mm from the overall platform height. The entire base has been cut to a triangular shape to reduce its size. To reinforce the peculiar shape resulting after the cut, web–beams have been added underneath (Figure 3.9(b)).

The joint mountings are positioned on a circle of 101.6 mm radius. The distance between neighboring mountings is 30.48 mm. Although larger by 5 mm than in the case of the RA, the ratio between the close and distant joint distances is small enough not to affect the size of the workspace.

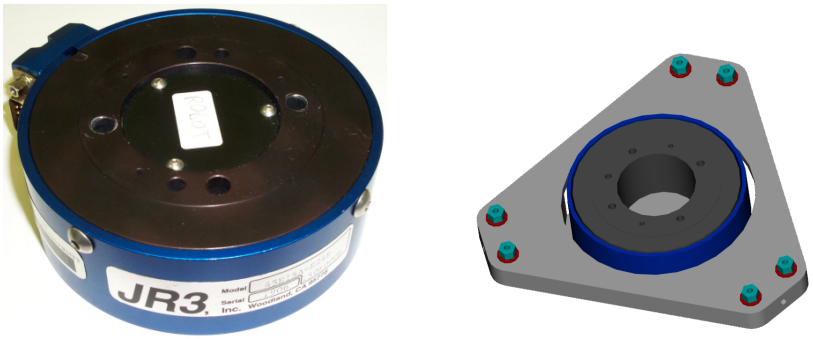
Force Sensor

The force sensor used by the Rutgers Mega–Ankle platform is a JR3 6DOF sensor model 45EE15A (Figure 3.10(a)). The new platform design brings a difference in the force sensor also. Although the model is the same as the one used by the smaller RA platform, the input ranges have to be larger to accommodate the weight of a person (Table 3.1).

Table 3.1: Force sensor input range.

	X		Y		Z	
	Min	Max	Min	Max	Min	Max
Force	-667 N	667 N	-667 N	667 N	-2224 N	2224 N
Torque	-127 Nm	127 Nm	-127 Nm	127 Nm	-127 Nm	127 Nm

The larger ranges raise a reading resolution issue. The force sensor outputs voltages between -10 V and 10 V. The voltage is read through a 12–bit A/D converter. This means that the Z force interval of $(-2224, 2224)$ is mapped on the digital interval of $(-2048, 2047)$. This gives a theoretical Z–force resolution of 1.085 N. The maximum forces measurable along the X and Y axes are half as those measured along the Z–axis. Thus, the resolution of the sensor on the X and Y axes (horizontal plane) is approximately 0.5 N. The measured noise of the sensor is below 0.12 N.



(a) Model 45E15A (b) Force sensor Assembly

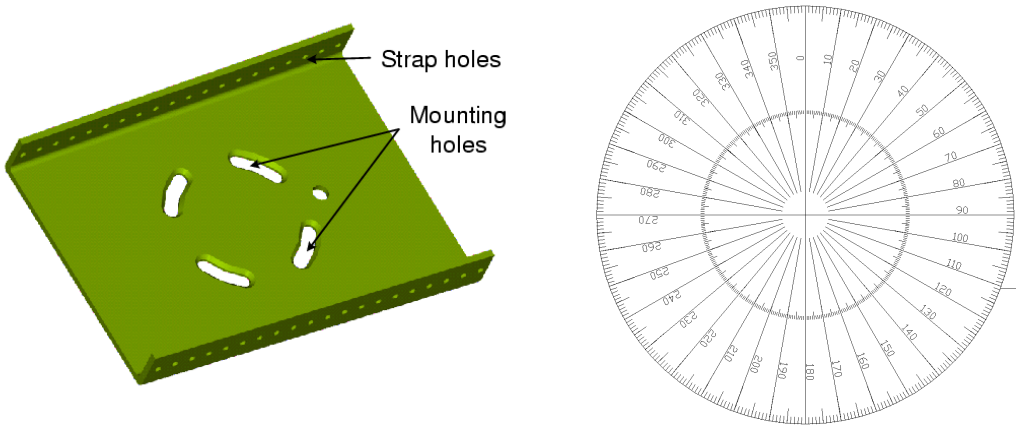
Figure 3.10: JR3 force sensor and assembly.

Figure 3.10(b) shows the assembly of the mobile base and the force sensor. As we have mentioned before, the "bucket" in the mobile base is designed to reduce the height of the platform. However, the sensor does not completely sink into the bucket, rather

it sticks out by half of its height. This elevation actually prevents the foot binding installed on top to interfere with the nuts holding the joints to the base.

Foot Binding

The foot binding is a part connected to the force sensor on which the user's foot is secured prior to the exercise (Figure 3.11(a)). It is designed as a thin but rigid stainless steel plate with multiple side holes for mounting the Velcro straps.



(a) Foot Binding

(b) Protractor mounted under the binding

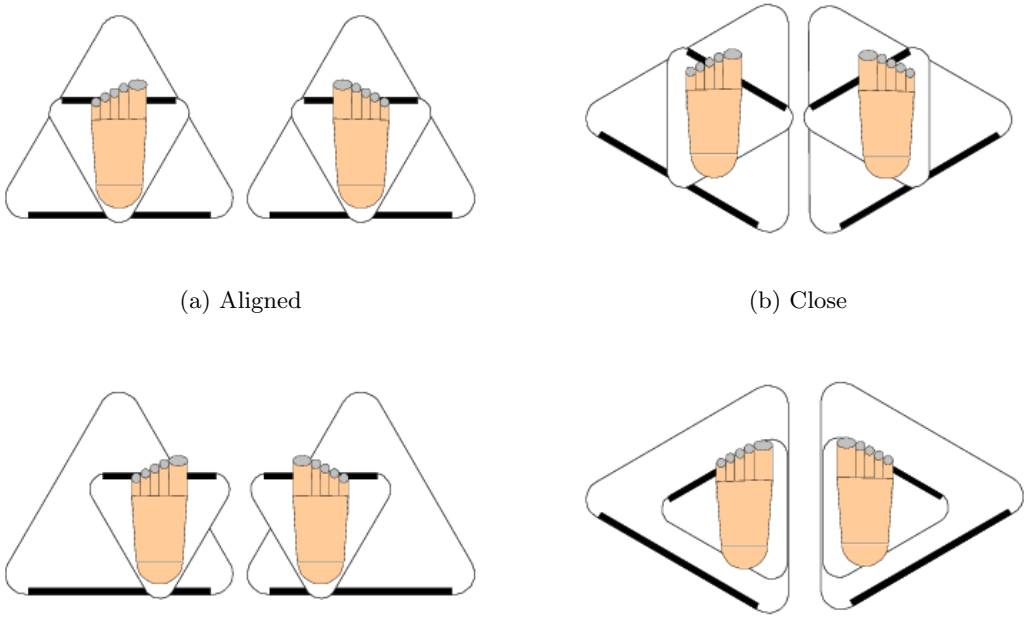
Figure 3.11: Rutgers Mega–Ankle End–Effector.

The connection to the force sensor is made using four bolts. The holes for the screws are extended as arches to allow easy orientation adjustments. Such adjustments are necessary depending on the relative position of the two platforms. The orientation of the foot binding is a parameter that must be given to the controller in order to adjust the reference coordinate frames. The orientation angle must be accurate and easy to find so that adjusting the bindings does not become time consuming. A paper protractor has been designed and attached to the force sensor underneath the foot binding. The angle values on the protractor show through a specially designed opening in the foot binding.

3.1.2 Dual Platform Configurations

The relative position of the two platforms is important mainly because of the distance it creates between the user's feet. During walking, while the swinging foot passes the support foot, the ankles get very close to touching each other. Hence, the distance between the two platforms must be as small as possible in order to simulate realistic gait. Figure 3.12 shows four possible configurations of the dual platform systems. The system is defined by five reference frames: the user's reference frame, the fixed bases' reference frames and the mobile bases' reference frames.

In aligned configuration, (Figure 3.12(a)) all the five frames are parallel to each other. Because this situation does not take advantage of the triangular cut of the platforms, the minimum distance between the user's ankles is 558 mm. This configuration is suitable for sitting exercises or special lateral step training. A simple way to make this configuration work for walking is to move the mobile bases' home positions inward (Figure 3.12(c)). The distance between the ankles is reduced to 355 mm but the workspace of platform is also drastically reduced.



(a) Aligned

(b) Close

(c) Aligned–displaced

(d) Close–rotated

Figure 3.12: Top views of dual platform configurations [11].

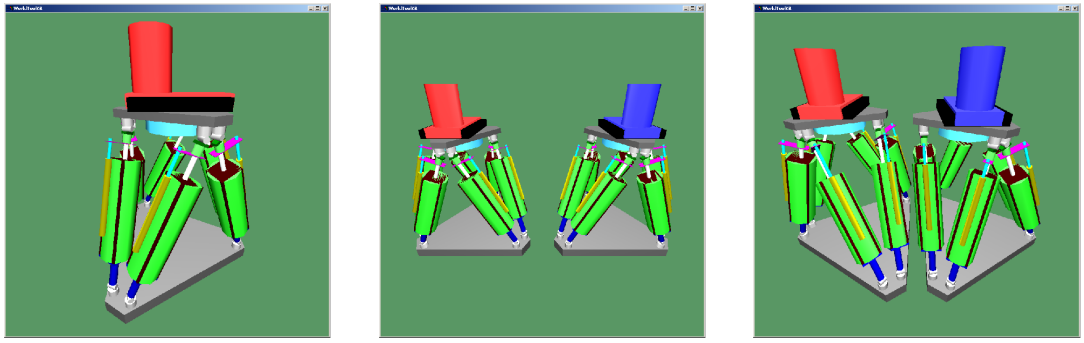
A configuration suitable for walking is shown in Figure 3.12(b). The distance between the user's ankles is reduced to 254 mm. This configuration requires software transformations of the readings since the platforms are not aligned with the reference frame. These transformations are handled transparently by the controller once it is given the foot binding orientation. An ideal situation would be the one shown in Figure 3.12(d) where the mobile bases are rotated and displaced inward reducing the distance between ankles to only 127 mm. Unfortunately, this configuration cannot be reached by the current design of the platform, and even if it were reachable the workspace would be practically null.

Platform Interference

Setting up the two platforms in such close proximity, makes it possible for their cylinders to hit each other causing uncontrollable disturbances, improper functioning and damage. The evaluation of the platforms interference was done using a VR simulation of the RMA robots.

The VR simulation was designed to be configurable for testing various situations. A parameter file provided the dimensions for the designed parts. The dimensions were taken from the CAD files and input into the application. The simulation provided support for one or two platforms to be rendered and analyzed simultaneously. In a two–platform setup, the relative position and orientation of the two platforms was also configurable. In order to test the dual platform configurations presented in Section 3.1.2, the orientation of the mobile bases and the foot bindings were also linked to configurable parameters. The simulation was run in three main configurations: single platform (Figure 3.13(a)), two aligned platforms (Figure 3.13(b)), and two close platforms (Figure 3.13(c)).

The platforms were positioned sequentially in configurations obtained by sampling an estimated bounding space at equidistant points. At every point, the inverse kinematic equations were solved and the length of the cylinders computed. If the cylinder lengths were within the physical ranges of the actual actuators, the simulation proceeded to check for interference between parts. The interference was checked by doing



(a) One platform (b) Two aligned platforms (c) Two close platforms

Figure 3.13: Rutgers Mega–Ankle platform simulation developed with WorldToolKit (Sense8 Co.).

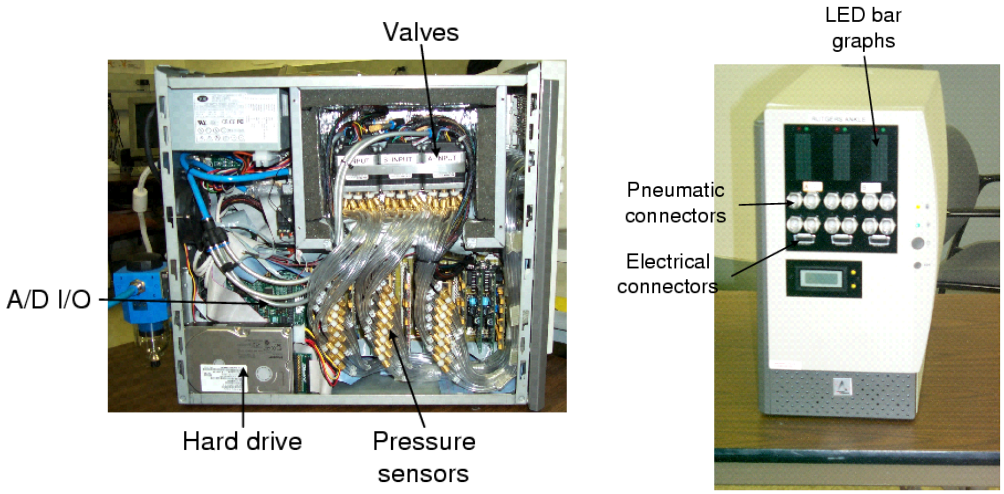
bounding box collision detection between the following parts: actuator envelopes, potentiometers, force sensor mounting and mobile joints. The actuator envelopes are the cylinders that enclose the actual actuator dimensions (Figure 3.13). The actuators are modeled as boxes and their corners are sticking out of the cylindrical envelopes. This is because the actual actuators have rounded corners and the envelopes are made to fit the real dimensions.

The results of the analysis showed that the optimal inter–foot distance versus platform workspace ratio was provided by the configuration shown in Figure 3.12(b). The minimum distance between the platforms in this situation was calculated to be 180 mm. This distance added to this configuration's minimum inter–ankle distance of 254 mm was still too large for simulating gait (during walking the feet can be as close as to nearly touch each other). To be able to reduce this distance and avoid interactions between the platforms, a Plexiglas sheet was installed vertically between the platforms shielding them from each other. With the shield separating them, the platforms were installed at 100 mm distance.

3.2 Electro–Pneumatic RMA Controller Interface

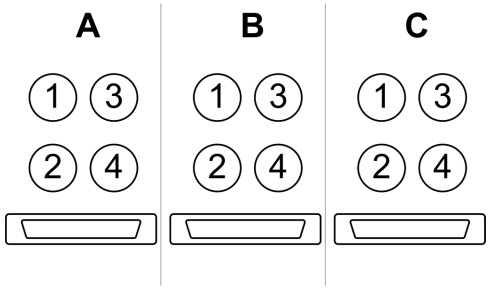
The RMA control interface is shown in Figure 3.14. It consists of a regular PC case containing a compact size PC board, a hard–drive, custom electronics, pneumatic valves

and tubing. The embedded PC board (Ampro Inc., San Jose, CA) has a Pentium III 933 MHz processor and 128 MB of RAM. The board provides a PC–104 interface used to connect the A/D I/O boards. Each Stewart platform is connected to the front of the interface (Figure 3.14(b) and 3.14(c)) by six pairs of pneumatic tubing (one for each cylinder) and one electrical connector. The connectors on the front of the interface are divided in three groups, each group having one electrical connector and four double pneumatic connectors. This division in three separate modules reflects the internal design of the interface. The green rectangles above each connector groups are LED bar graphs that show in real time the pressure in the corresponding air channels.



(a) Internal view

(b) Front view



(c) Connector groups

Figure 3.14: The Haptic Control Interface (HCI).

The interface shown in Figure 3.14 is the second–generation design from Girone's

initial system, the author of both designs being Mourad Bouzit [18]. This new architecture has a modular approach that allows the interface to drive simultaneously two Rutgers Ankle or Rutgers Mega–Ankle platforms. In addition, the new design is compatible with the Rutgers Master Haptic Glove, being able to run up to three gloves in parallel or one platform and one glove. There are four possible device connection configurations. All these configurations are maximal in the sense that nothing else can be connected to the control interface. Incomplete connections are also supported. The detailed connections corresponding to these configurations are shown in Table 3.2. "A", "B", and "C" are the three connector groups, "P" means platform, "G" means glove and "-" means empty. A configuration is described by a group of three characters showing the devices in order from left to right.

Table 3.2: Device connection configurations.

Electrical connection (ABC)	Pneumatic connection		
	Device 1	Device 2	Device 3
P–P	A1,A2,A3,A4,B1,B2		B3,B4,C1,C2,C3,C4
P–G	A1,A2,A3,A4,B1,B2		C1,C2,C3,C4
G–P	A1,A2,A3,A4		B3,B4,C1,C2,C3,C4
GGG	A1,A2,A3,A4	B1,B2,B3,B4	C1,C2,C3,C4

3.2.1 Electric and Pneumatic Connections

Each of the three groups of connectors on the front of the control box corresponds to a hardware module inside the box. A module consists of eight air–channels (pneumatic tubing with controlled air pressure), twelve analog channels for reading sensors, eight digital input channels for reading the ID of the connected device, and sixteen DIO channels for opening and closing the valves.

The air pressure is controlled using solenoid ON/OFF matrix valves manufactured by ITA Matrix Inc. Each such matrix valve contains eight independent micro–valves. There are two types of valves used in the system: intake and exhaust. The matrix valves are visible in the top–right corner of Figure 3.14(a).

An air channel consists of one air tubing having an intake and an exhaust valve connected at one end, a pressure sensor somewhere in the middle and the outside

connector at the other end. Each platform requires twelve air channels to actuate its six double–acting cylinders. The control interface contains twenty–four air–channels, which are sufficient to control two platforms simultaneously.

Figure 3.15 shows how the hardware is grouped. The eight air–channels of each module use a pair of intake and exhaust matrix valves. The middle module is split between the platforms. Each module is read and controlled through a Micro/Sys A/D–I/O board model MPC550. Each A/D–I/O board provides sixteen A/D channels on top of eight converters, and twenty–four DIO channels. Each matrix valve is controlled through an 8–bit DIO port (eight channels). The remaining eight DIO channels are wired to the front connector and are used to read the device ID. The A/D channels of one Micro/Sys board are not enough to cover all the sensors needed to control one platform, so the middle module's board is shared between the side outputs. While splitting the air–channels can be done in the software controller, the A/D channels need to be multiplexed in hardware between the sensors. A switching mechanism was designed by Bouzit to change the wiring configuration so that the interface could be used with the Rutgers Master Haptic Glove also. The next section describes in detail the workings of the switching mechanism.

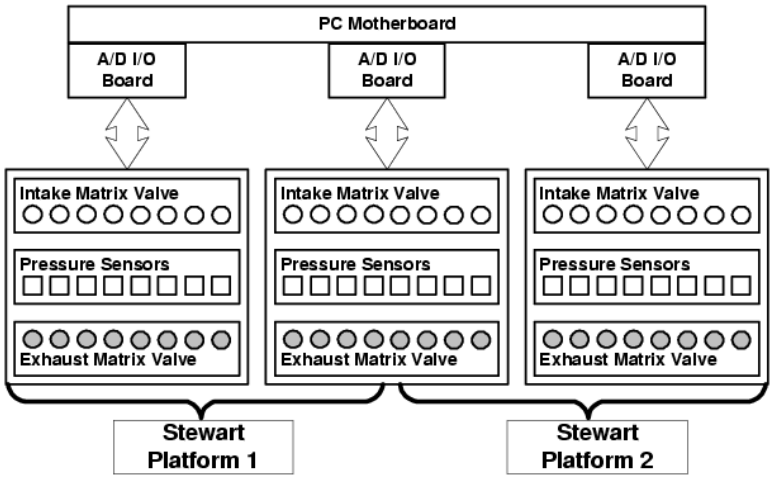


Figure 3.15: Electrical and pneumatic connection diagram of the HCI.

3.2.2 Hardware Switching

One platform requires a total of twenty–four A/D channels to read its sensors: twelve to read the pressure sensors, six to read the linear potentiometers and another six A/D channels to read the 6DOF force sensor. In other words, there are twelve signals read directly from the RMA robot and twelve signals coming from inside the controller box from its pressure sensors.

One haptic glove requires only sixteen A/D channels: four to read the pressure sensors, four to read the glove cylinder displacements, and eight to read the Hall–Effect sensors on the glove. Thus, a glove can be handled by only one module.

The switching mechanism allows the hardware to be reconfigured so that the middle module could control a glove or can help controlling a platform connected to one of the side connectors.

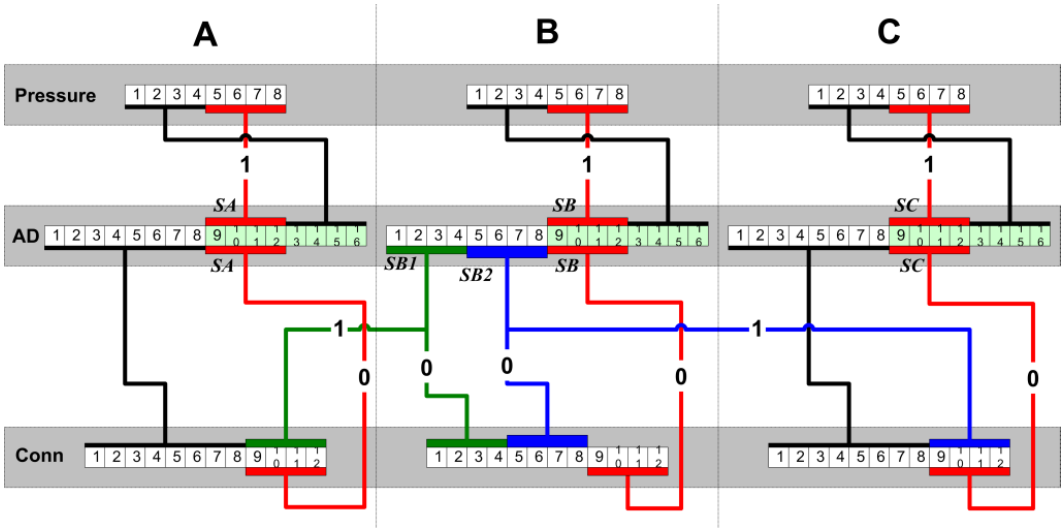


Figure 3.16: Hardware switching.

The switching mechanism changes the connection of the A/D channels between the pressure sensors and the front connector pins. The switches are controlled through the DIO channels provided by the PC board's parallel port. In Figure 3.16 the three modules are placed vertically, each having a group of eight pressure sensors corresponding to the eight air–channels, one A/D I/O board and a front connector. The switches in the diagram are the lines with "0" or "1" written on them. Each line shows what sensors

are connected to the A/D channels and what bit value activates that connection. The lines that are not numbered represent fixed connections.

The first eight A/D channels of each A/D board read the first eight pins of their corresponding front connector. The last four channels read the first four pressure sensors in their module. Channels 9 through 12 can be changed between the last four pressure sensors or the last four pins of the front connector (switches SA, SB and SC). The middle A/D board has two additional switches, SB1 and SB2 that split the channels to the side connectors A and C. So SB1 can switch the first four A/D channels of the middle board between the last four pins of connector A and the first four pins of connector B. Switch SB2 changes the connections between pins five to eight of connector B and the last for pins of connector C.

When module B is not split, all the modules have the front connector fully read by the A/D channels while four of the pressure sensors are ignored. This is the case when only gloves are connected to the system because they need only four air–channels. When a platform is connected to module A or C, the four unread pressure sensors are covered by the A/D channels reading the last four connector pins. Since those connector pins are necessary to read the twelve signals coming from the platform, the middle module reroutes four of its A/D channels to read them. Once that is done, the middle connector cannot be used with any device. Table 3.3 shows the four main device configurations and the switch values necessary to accommodate them.

Table 3.3: Device connection configurations and the corresponding switching values.

Device configuration	SA	SB1	SB2	SB	SC
P–P	1	1	1	1	1
P–G	1	1	0	1	1
G–P	0	0	0	1	1
GGG	0	0	0	0	0

3.3 Unweighing Frame

Another hardware component of the mobility simulator is the Unweighing System (Biodex Co.). The Rutgers Mega–Ankle robots are strong enough to lift a 300 kg

load each. In order to simulate a walking surface, the robots must not only sustain the user but also keep the feet in fixed orientations. The capacity to maintain the orientation of the load is determined by the RMA device's torque output capability. Based on the size of the foot binding mounted on top of the RMA, it was determined that one platform can balance a maximum dynamic load of 50 kg. During walking, a person's weight is fully shifted from one foot to another; hence, each of the two RMA robots must be able to handle by itself the full weight of the user. The role of the Unweighing System is to overcome the 50 kg maximum load limitation and make the simulator usable by the majority of patients. Due to the frame's capacity to unload up to 60% of a user's weight, patients weighing up to 120 kg (250 lbs) may use the system successfully. It has been determined empirically that supporting more than 40% of the user's body weight alters gait significantly. This finding is confirmed by the maximum unweighing percentage chosen by Visintin in his study on training gait in stroke patients using body weight support and a treadmill [113]. Thus, the maximum user weight supported by the simulator is 83 kg (184 lbs)

Besides reducing the patient's weight, the unweighing frame also improves safety and comfort for the patient. A side effect of having the patient suspended in the frame is a tendency to spin sideways while using the simulator. This is mitigated by the frame handles, which the patient can hold during the exercise, thus reducing body lateral spin.

3.4 Large Screen Display

The Rutgers Ankle System used a 21–inch monitor to display the virtual environment that was part of the exercise. Although not large, the monitor was sufficient to produce immersion, as demonstrated during studies in which the therapy took place in a busy clinic environment [16]. The role of vision in scanning the environment and preparing for movement is a well–demonstrated requirement for gait [90, 91, 92]. It was considered essential to produce an immersive environment whose complexity could be systematically manipulated based on task hierarchies [40]. Important aspects of gait related to ambient conditions, attentional demands and traffic level can be delivered with the proper visual surround. Thus, it was deemed necessary to scale the visual scene up for

the mobility simulator system. A large back–projection screen (74" diagonal, viewable area) was constructed by Scott Winter for displaying the virtual environment, in order to increase the patient's immersion. The display presents monoscopic images using a high–intensity, high–resolution projector. Unlike other large displays (e.g. Barco), which require dimming the ambient light, our display can be comfortably viewed with the clinic lights on.

Chapter 4

Rutgers Mega–Ankle Kinematics and Dynamics

4.1 Stewart Platform Kinematic Model

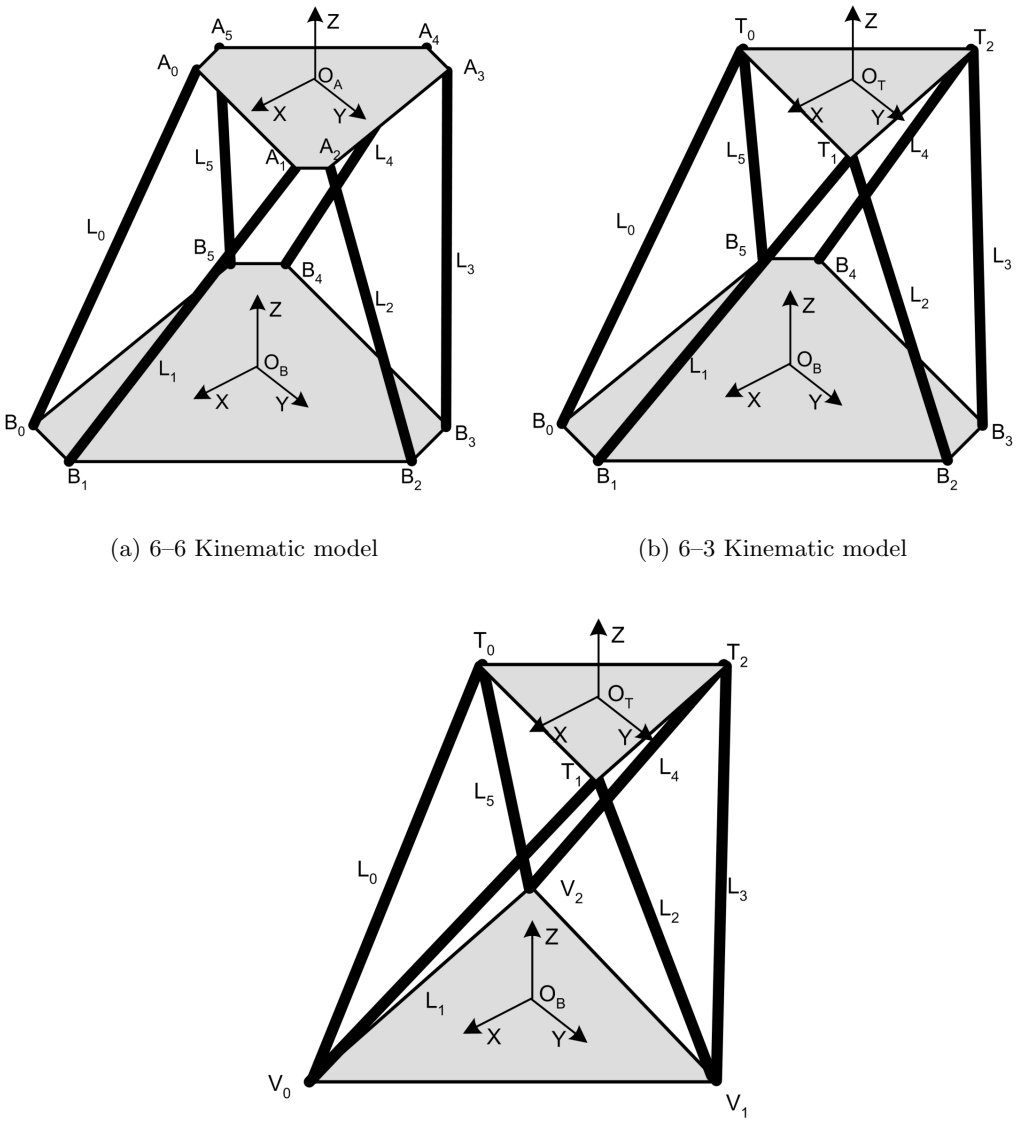
The general kinematic model of a Stewart platform does not make any assumptions about shape or size of the two bases or about the position of the joints on each base. Theoretically, the joints on each base need not be coplanar and do not need to be positioned in symmetric fashion. The choice of architecture for a robot must consider the solvability of the kinematic chains. The inverse kinematic calculations for a Stewart platform robot are generally straight forward, but the forward kinematics are very complex due to insufficient input data. However, certain "regular" aspects of a design can drastically simplify the calculations [111, 118, 77].

In the case of the Rutgers Mega–Ankle (Figure 4.1(a)), all the six joints of each base are coplanar and positioned at equal distances as follows: neighboring joints and distant joints are placed on circles at equal distances within their respective bases. Neighboring joints are pairs of joints placed close to each other on each base (e.g. B_0 and B_1). The remaining pairs of joints are the distant ones (e.g. B_1 and B_2). The fixed base joint circle has a radius twice the size of that of the mobile base. This model is generally known as the 6–6 model because it has distinct joints on each base.

Although the model shown in Figure 4.1(a) simplifies the kinematic equations, they are still quite complex. The simpler 6–3 kinematic model is presented in Figure 4.1(b). The model is identical to the one presented before except that the neighboring mobile joints are considered coincidental. This model does not reflect the actual structure of the Rutgers Mega–Ankle robot. However, due to the small ratio between the neighboring and distant joint distances, the accuracy of the results will not be affected significantly. Collapsing the neighboring joints to one point can be used to reduce the number of kinematic equations.

A further model simplification step can be taken by considering the neighboring

joints on the fixed base to be coincidental. This is the 3–3 Stewart platform model in which all the neighboring joints are sharing the same position (Figure 4.1(c)). This can reduce the calculations even further but the errors created by the differences from the real robot architecture are larger.



(a) 6–6 Kinematic model

(b) 6–3 Kinematic model

(c) 3–3 Kinematic model

Figure 4.1: Rutgers Mega–Ankle kinematic models.

Before proceeding to the discussion of the kinematic equations, it is necessary to make a few notation conventions (Table 4.1). The vectors representing the actuators are marked as $\overrightarrow{L_i}$, and the vectors representing the joint position relative to their respective

base are marked as $\overrightarrow{A_i}, \overrightarrow{B_i}$. The X, Y and Z components of a vector $\overrightarrow{P}$ will be written as X_P, Y_P, Z_P.

Table 4.1: Kinematic model notations.

Model 6–6	Model 6–3	Model 3–3
$\overrightarrow{L_i} = \overrightarrow{B_i A_i}, \quad i = 0, \ldots, 5$	$\overrightarrow{L_i} = \overrightarrow{B_i A_{\lfloor i/2 \rfloor}}, \quad i = 0, \ldots, 5$	$\overrightarrow{L_i} = \overrightarrow{B_i A_i}, \quad i = 0, \ldots, 2$
$\overrightarrow{A_i} = \overrightarrow{O_A A_i}, \quad i = 0, \ldots, 5$	$\overrightarrow{A_i} = \overrightarrow{O_T T_i}, \quad i = 0, \ldots, 2$	$\overrightarrow{A_i} = \overrightarrow{O_T T_i}, \quad i = 0, \ldots, 2$
$\overrightarrow{B_i} = \overrightarrow{O_B B_i}, \quad i = 0, \ldots, 5$	$\overrightarrow{B_i} = \overrightarrow{O_B B_i}, \quad i = 0, \ldots, 5$	$\overrightarrow{B_i} = \overrightarrow{O_B B_i}, \quad i = 0, \ldots, 2$

4.2 Inverse Kinematics

The problem of inverse kinematics for a manipulator in general is to compute the position of the actuators based on the position and orientation of the end–effector (foot binding) [32, 80]. In the Rutgers Mega–Ankle case, the inverse kinematics determines the actuator vectors based on the position of the foot binding. This is necessary when one controls the platform position through the position of the cylinders or for computing the force applied by the platform by summing the force vectors of each actuator.

Because the position (translation and orientation) of the foot binding uniquely defines the position of the mobile base and through it the position of each mobile joint, the solution of the inverse kinematics problem is quite simple. The input data to the model are the position coordinates (x, y, z) and Euler angles (α, β, γ) of the mobile base with respect to the fixed base. From the Euler angles, the rotation matrix R of the mobile base is shown in Equation 4.1 [32]. To simplify the writing $c\alpha$ is used for $\cos\alpha$ and $s\alpha$ for $\sin\alpha$.

$$R = \begin{pmatrix} c\alpha \cdot c\beta & c\alpha \cdot s\beta \cdot s\gamma - s\alpha \cdot c\gamma & c\alpha \cdot s\beta \cdot c\gamma + s\alpha \cdot s\gamma \\ s\alpha \cdot c\beta & s\alpha \cdot s\beta \cdot s\gamma + c\alpha \cdot c\gamma & s\alpha \cdot s\beta \cdot c\gamma - c\alpha \cdot s\gamma \\ -s\beta & c\beta \cdot s\gamma & c\beta \cdot c\gamma \end{pmatrix} \tag{4.1}$$

For the 6–6 kinematic model, considering $\vec{P} = (X_P, Y_P, Z_P)$ the position vector of the fixed base, the actuator vectors $\overrightarrow{L_i}$ can be expressed as

$$\overrightarrow{L_i} = R \cdot \overrightarrow{A_i} + \overrightarrow{P} - \overrightarrow{B_i}, \quad i = 0, \ldots, 5 \tag{4.2}$$

Substituting the values of R and P in Equation 4.2, the actuator vector values are

$$\begin{aligned} \overrightarrow{L_i} &= \begin{pmatrix} X_{A_i}\cdot c\alpha\cdot c\beta + Y_{A_i}\cdot(c\alpha\cdot s\beta\cdot s\gamma - s\alpha\cdot c\gamma) + Z_{A_i}\cdot(c\alpha\cdot s\beta\cdot c\gamma + s\alpha\cdot s\gamma) + X_P - X_{B_i} \\ X_{A_i}\cdot s\alpha\cdot c\beta + Y_{A_i}\cdot(s\alpha\cdot s\beta\cdot s\gamma + c\alpha\cdot c\gamma) + Z_{A_i}\cdot(s\alpha\cdot s\beta\cdot c\gamma - c\alpha\cdot s\gamma) + Y_P - Y_{B_i} \\ -X_{A_i}\cdot s\beta + Y_{A_i}\cdot c\beta\cdot s\gamma + Z_{A_i}\cdot c\beta\cdot c\gamma + Z_P - Z_{B_i} \end{pmatrix} \\ i &= 0, \ldots, 5 \end{aligned} \tag{4.3}$$

The inverse kinematic solutions for the 6–3 model can be obtained from Equation 4.3 by substituting A_i by $T_{\lfloor i/2 \rfloor}$. Similarly, The 3–3 model inverse kinematics can be obtained from the 6–3 model by replacing B_i by $V_{\lfloor i/2 \rfloor}$ and removing the three identical equations resulting after the substitution.

4.3 Forward Kinematics

The forward kinematic problem for a Stewart platform manipulator is to compute the Cartesian position of the end–effector (foot binding) from the lengths of the six actuators [32, 80]. The problem is significantly more complex than that of inverse kinematics due to the lack of orientation information about the actuators. The problem requires finding X_P, Y_P, Z_P, α, β and γ from the six non–linear equations shown in 4.3.

The most common approach to solve this problem is by using the Newton–Raphson iterative method [118]. The functions f_i used in the iteration are derived from Equation 4.2 by applying the norm to both sides. The algorithm starts from a solution estimate, which is usually in the middle of the workspace. The next solution is calculated using Equation 4.5, where $f(q)$ is replaced by 4.4.

$$\begin{aligned} f_i(q) &= \left(R(q)\overrightarrow{A_i} + \overrightarrow{P(q)} - \overrightarrow{B_i}\right)^T \left(R(q)\overrightarrow{A_i} + \overrightarrow{P(q)} - \overrightarrow{B_i}\right) - \overrightarrow{L_i}\cdot\overrightarrow{L_i} \\ q &= (X_P, Y_P, Z_P, \alpha, \beta, \gamma), \quad \text{solution estimate} \end{aligned} \tag{4.4}$$

The next solution estimate is defined using the inverse Jacobian.

$$q_{n+1} = q_n - \left(\frac{\partial f(q_n)}{\partial q_n}\right)^{-1} f(q_n) \tag{4.5}$$

The procedure is repeated until the error between the computed actuator length and the real measured actuator length is lower than a given error bound or until a maximum number of iterations have been completed [41, 83].

The advantage of the Newton–Raphson solution is that it can be applied to any type of hexapod robot without changing the equations. It is also easy to implement. The disadvantage is that there is no guarantee that it will converge [66]. Considering that the iterative forward kinematics must yield results in real time, the number of iterations must be limited due to the heavy floating point calculations required by the procedure. If the algorithm does not converge in the given number of iterations, the result has to be discarded and usually the previously calculated solution is kept as the current position [118].

Extensive research has been done toward finding better solutions to the Stewart platform forward kinematics. The most frequent ideas in the literature are focused on finding a reasonable set of constraints for the kinematic model so that the resulting equations would be either simpler or, even better, have closed–form solutions.

Zhang and Song [121] reduce the forward kinematic equations of a *nearly general* Stewart platform to a 20^{th} order polynomial. The *nearly general* constraint requires the joint to be coplanar within their bases. The resulting polynomial is solved then by numerical methods using Maple (Maplesoft Co.). Due to the duality of the system, there are forty resulting solutions out half of which are eliminated by the requirement of having the Z-coordinate positive.

Lee and Roth published a closed–form solution for a Stewart platform with the joints placed on regular hexagons on both bases [69]. Although this simplifies the kinematic equations, the workspace of such a robot is very small. For the same bases and actuators, the robot's workspace is larger if the joints are positioned alternatively close and far from the previous joint on each base.

Liu et al. [73] derive the forward kinematic equations of a 6–3 Stewart platform. The equations are simpler than the general 6–6 case because of the identities between the mobile base joints. The equations are solved by the Newton–Raphson method.

Nanua et al. [82] reduce the forward kinematic equations of a simplified 3–3 model

to an 8^{th} order polynomial. The resulting solutions are all valid and the correct one has to be chosen using the physical orientation ranges of the platform actuators.

Other solutions to the forward kinematics problem using approaches similar to the ones above can be found in [118, 79, 73, 53, 61, 68, 69, 104].

A closed form solution for the 6–6 model can be found if the joints are positioned on similar (linearly related) hexagons. Solutions to this configuration have been given by Yang and Geng [118], Sreenivasan et al. in [106] and Ji and Wu in [61]. All solutions use the similarity of the fixed and mobile base to express the mobile joint vectors $\overrightarrow{A_i}$ in terms of fixed base joint vectors $\overrightarrow{B_i}$ (Equation 4.6). After substituting Equation 4.6 into Equation 4.2, the terms of the six equations are arranged in a linear system of six variables. After the linear system is solved using Gaussian elimination [101], the solution of the forward kinematics is computed by solving a few second order equations. The solution proposed in [61] by Ji and Wu has been implemented and tested. Although the implementation could replicate the results given in the article when applied to the Rutgers Mega–Ankle robots, the characteristic matrix of the initial six by six linear system was very frequently near–singular yielding numbers beyond the computer's floating point representation. For validation, the same algorithm was implemented in Matlab and the results were identical. The same results were obtained by Bruyninckx in [21].

$$\overrightarrow{A_i} = \mu \overrightarrow{B_i} \quad i = 0, \ldots, 5 \tag{4.6}$$

4.3.1 Rutgers Mega–Ankle Forward Kinematics Solution

The forward kinematics solution implemented for the Rutgers Mega–Ankle is the one proposed by Nguyen and Pooran in [83]. The initial implementation has been done by Girone for the smaller Rutgers Ankle robot prototype. The algorithm has been adapted by Boian to the RMA controller and it has been improved in accuracy.

The approach proposed in [83] solves the direct position calculations of a 3–3 Stewart platform model using Newton–Raphson iterations presented above. The solution is stable and converges on average in three to five iterations. The results returned by the implementation are not accurate though because they solve a 3–3 model rather than a

6–6 model. The algorithm's errors were small enough to make it suitable for the sitting exercises, but they were too large for the mobility simulator and created problems when performing collision detection between the foot and the virtual surface. The main cause of the problems came from the discrepancy between the forward kinematics solved on a 3–3 model and the accurate inverse kinematics solved on the real 6–6 model. The forward kinematics applied on the actuator positions calculated by the inverse kinematics was not yielding the same initial Cartesian position. For instance, when the platform was positioned at an elevation Z_{real} that mapped in the virtual world to a position above the walking surface, the forward kinematics were returning a smaller measured $Z_{3\_\_3}$ that mapped to a virtual position below the walking surface.

The simplest way to solve this problem is to switch the inverse kinematics to use the 3–3 model. While this approach would have yielded matching FK and IK results, the actual Cartesian positioned calculated by the controller would have been inaccurate and could have lead to position control problems.

A more computationally expensive solution would have been the implementation of the 6–6 iterative forward kinematics. Since the design of the RMA platforms is close to having the joints placed on similar hexagons and given the near singular matrices of the closed–form solution implementation, there were good chances that the 6–6 model could have been either unstable or require many iterations to converge.

The solution implemented relied on iterations applied outside the forward kinematics algorithm. The inverse kinematics algorithm was known to be accurate. The results of the 3–3 forward kinematics were input to the inverse kinematics procedure and the resulting actuator positions compared to the measured actuator positions. The average of the values were input back into the forward kinematics algorithm. The procedure was repeated until the difference between the calculated and measured actuator positions was less the 0.1 mm. This procedure requires an average three iterations to converge, one iteration taking approximately 0.39 ms.

4.4 Inverse Dynamics

The dynamic equations of a Stewart platform robotic system make the connection between the position, velocity, acceleration and forces of the end–effector and those of the cylinders [80]. As in the case of the kinematics, the equations can be solved in two directions: forward dynamics and inverse dynamics. The forward dynamics problem starts knowing the forces of the platform's cylinders and outputs the end–effector position, velocity and acceleration. This is useful for simulating the behavior of the Stewart platform. For the mobility simulator application, the controller needs to resist the external disturbances created by the user's foot on the end–effector. For this purpose, only the inverse dynamics of the RMA robot are calculated.

The problem of the inverse dynamics has been extensively researched and various solutions have been offered. These solutions differ either by the formulation of the equations of motion that are used, or by certain simplifications of the kinematic model. In [73] Liu et al. derived the dynamic equations of a 6–3 Stewart platform using the Lagrange equations of motion. Do and Yang [38] offer a fully derived solution for the inverse dynamics based on the Newton–Euler equations of motion. Their platform model uses spherical joints at the points where the cylinders are connected to the bases. This brings in an extra degree of freedom that is idle and hence uncontrollable [120]. Zhang and Song in [120] address this problem using a model with universal joints. They also use the Newton–Euler equations of motion but they are linked using the virtual work principle. Other significant approaches can be found in [74, 72, 102, 64, 62, 88].

The inverse dynamics solution proposed by Zhang and Song [120] was adapted and implemented for Rutgers Mega–Ankle controller. The model used by this method is shown in Figures 4.2 and 4.3. The platform model presented in Figure 4.2 is the same as that in Figure 4.1(a) except that point C_A, which represents the center of mass of the mobile assembly, is coincidental with the origin of the frame of reference O_A.

Figure 4.3 presents the frames of references and variables attached to one actuator. Frame $[X_jY_jZ_j]$ is parallel to the fixed base frame of reference O_B in Figure 4.2. i_j, j_j and k_j are the unit vectors of frame $[X_jY_jZ_j]$. The frame $[X_aY_aZ_a]$ has the X_a axis

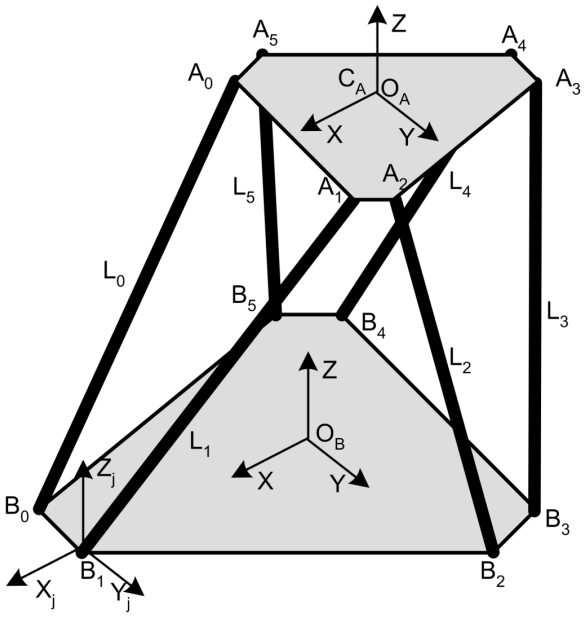


Figure 4.2: 6–6 dynamic model; frames of reference.

aligned with the actuator pointing toward A_i (X_A has the same direction as L_i). i_a, j_a and k_a are the unit vectors of frame $[X_aY_aZ_a]$. $[X_mY_mZ_m]$ is an intermediate frame between $[X_jY_jZ_j]$ and $[X_aY_aZ_a]$ having axis Z_m identical to Z_j and axis Y_m identical to Y_a. The center of gravity of the cylinder's fixed part is marked as $C_{i,1}$ and the cylinder shaft's center of mass as $C_{i,2}$. s_i is the distance from the center of mass of the fixed part of the actuator to the mounting point B_i. t_i is the distance between the center of mass of the mobile part of the actuator (shaft) to the mounting point B_i. l_i is the position of the actuator (length of vector $\overrightarrow{L_i}$). r_i is the projection of L_i on X_m. ϕ_i and β_i are the angles of the universal joint connecting the actuator to the fixed base.

4.4.1 Dynamic Model Considerations

The model assumes that the top base's center of mass C_A (Fig. 4.2) and the end–effector position coincide with the center of the hexagon formed by the mobile base joints. The Rutgers Mega–Ankle top base does not have a simple disc shape (Figure 3.9) and the mobile base joints are not in the same plane with the end–effector. In addition, the end–effector's orientation can be changed. The inverse dynamics algorithm can still be applied if all positions, velocities, accelerations and external forces of the mobile

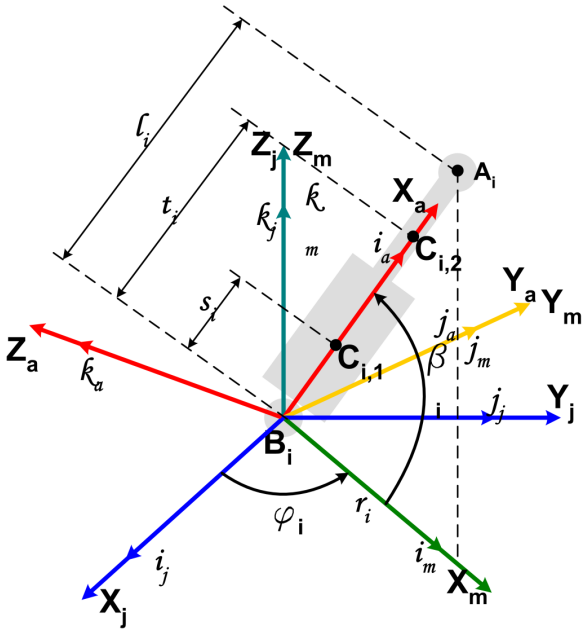


Figure 4.3: 6–6 Dynamic model; actuator i frames of reference and parameters.

assembly are translated on the Z–axis into the mobile base joint's frame of reference. This frame of reference is parallel to the original one, but is lowered so that its (X, Y) plane contains all the joints. The mobile base center of mass can be considered to be in the same position but the inertia tensor of the mobile assembly has to be calculated with respect to lower frame's origin. The joints attached to the fixed base require further transformations of the input data and results because they are elevated from the ground and hence are not in the same plane as the platform's frame of reference.

The parameters required by the transformations mentioned above are specified in a configuration file for each RMA robot. These parameters are:

- Measured elevation of the end–effector with respect to the frame of the mobile base joints;
- The inertial tensor of the mobile assembly, calculated using AutoCAD Mechanical Desktop with respect the point O_A;
- Mass of the mobile base;
- Mass of the fixed part of the actuator including the mass of the fixed part of the potentiometer;

- Mass of the mobile part of the actuator, including the mass of the potentiometer shaft;
- Position of the centroid of mass of the fixed part of the actuator measured from point B_i;
- Position of the centroid of mass of the mobile part of the actuator measured from point A_i;
- Inertia tensor of the fixed part of the actuator calculated with respect to the centroid of mass;
- Inertia tensor of the mobile part of the actuator calculated with respect to the centroid of mass;

4.4.2 Inverse Dynamics Implementation

The implementation of the inverse dynamics is adapted from [120] by adding the initial data input transformations. The algorithm is structured in the following phases:

1. Input data transformation: transform the end–effector position, velocity, acceleration and forces in the O_A frame of reference;
2. Actuator position analysis: compute r_i, t_i, l_i, s_i, ϕ_i and β_i;
3. Actuator velocity analysis: compute the linear and angular velocity of the mobile end of each actuator;
4. Actuator acceleration analysis: compute the linear and angular accelerations of the mobile end of the actuator and the linear accelerations of the two centroids of gravity;
5. Actuator inertial forces and torques analysis: compute the inertia forces and torques of each actuator;
6. Mobile base inertial forces and torques analysis: compute the inertia forces and torques of the mobile base;

7. Resulting forces and moments calculation: compute the resulting forces and moments of the actuators and mobile assembly;

8. Jacobian derivation;

9. Partial angular velocity matrix derivation;

10. Partial linear velocity matrix derivation;

11. Equations of motion derivation using the virtual work principle;

12. Actuator forces calculation.

The notations are used throughout the algorithm are presented in Table 4.2.

Variable	**Type**	**Description**
$\overrightarrow{{}^{B}p_E}, \overrightarrow{{}^{B}\gamma_E}$	(3,1) vector	Input data. Measured position (linear and angular) of the end–effector represented in frame O_B.
$\overrightarrow{{}^{B}v_E}, \overrightarrow{{}^{B}\omega_E}$	(3,1) vector	Input data. Measured velocity (linear and angular) of the end–effector represented in frame O_B.
$\overrightarrow{{}^{B}a_E}, \overrightarrow{{}^{B}\alpha_E}$	(3,1) vector	Input data. Measured acceleration (linear and angular) of the end–effector represented in frame O_B.
$\overrightarrow{{}^{B}Fx_E}, \overrightarrow{{}^{B}\tau x_E}$	(3,1) vector	Input data. Measured forces (linear and angular) of the end–effector represented in frame O_B.
$\overrightarrow{{}^{A}p_E}$	(6,1) vector	Input data. End–effector linear position represented in frame O_A.
$m_{i,1}$	number	Input data. Mass of the fixed part of actuator i
$m_{i,2}$	number	Input data. Mass of the mobile part of actuator i
m_A	number	Input data. Mass of the mobile base
$\overrightarrow{{}^{B}p_A}, \overrightarrow{{}^{B}\gamma_A}$	(6,1) vector	Position of the end–effector (linear and angular) transformed to frame O_A.
$\overrightarrow{{}^{B}v_A}, \overrightarrow{{}^{B}\omega_A}$	(6,1) vector	Velocity of the end–effector (linear and angular) transformed to frame O_A.

$\overrightarrow{{}^{B}a_A}, \overrightarrow{{}^{B}\alpha_A}$	(6,1) vector	Acceleration of the end–effector (linear and angular) transformed to frame O_A.
$\overrightarrow{{}^{B}Fx_A}, \overrightarrow{{}^{B}\tau x_A}$	(3,1) vector	Measured forces (linear and angular) transformed to frame O_A.
$\overrightarrow{{}^{B}F_A}, \overrightarrow{{}^{B}\tau_A}$	(6,1) vector	Inertial forces of the end–effector (linear and angular)
${}^{B}R_A$	(3,3) matrix	Rotation matrix of the end–effector O_A with respect to frame O_B
${}^{B}R_i$	(3,3) matrix	Rotation matrix of actuator i with respect to frame O_B
$\overrightarrow{{}^{B}v_i}$	(3,1) vector	Linear velocity of joint i in frame O_B
$\overrightarrow{{}^{e}\omega_i}$	(3,1) vector	Angular velocity of joint i in frame $[X_eY_eZ_e]$
$\overrightarrow{{}^{B}a_{Ai}}$	(3,1) vector	Linear acceleration of joint i in frame O_B
$\overrightarrow{{}^{e}\alpha_i}$	(3,1) vector	Angular acceleration of joint i in frame $[X_eY_eZ_e]$
$\overrightarrow{{}^{e}a_{c1,i}}$	(3,1) vector	Linear acceleration of fixed center of mass of actuator i in frame $[X_eY_eY_e]$
$\overrightarrow{{}^{e}a_{c2,i}}$	(3,1) vector	Linear acceleration of mobile center of mass of actuator i in frame $[X_eY_eZ_e]$
$\overrightarrow{{}^{e}F_{c1,i}}$	(3,1) vector	Inertial force of fixed center of mass of actuator i in frame $[X_eY_eZ_e]$
$\overrightarrow{{}^{e}F_{c2,i}}$	(3,1) vector	Inertial force of mobile center of mass of actuator i in frame $[X_eY_eZ_e]$
$\overrightarrow{{}^{e}M_i}$	(3,1) vector	Inertial torque of actuator i in frame $[X_eY_eZ_e]$
$\overrightarrow{{}^{e}I_{c1,i}}$	(3,1) vector	Inertial matrix of the fixed part of actuator i in frame $[X_eY_eZ_e]$
$\overrightarrow{{}^{e}I_{c2,i}}$	(3,1) vector	Inertial matrix of the mobile part of actuator i in frame $[X_eY_eZ_e]$
$\overrightarrow{{}^{A}I_A}$	(3,1) vector	Inertial matrix of the mobile base in frame O_A
$\overrightarrow{{}^{B}F_A}$	(3,1) vector	Inertial force of the mobile base O_B
$\overrightarrow{{}^{B}M_A}$	(3,1) vector	Inertial torque of the mobile base in frame O_B

$\overrightarrow{{}^{e}R_{c1,i}}$	(3,1) vector	Resultant force of the center of mass of the fixed part of actuator i in frame $[X_eY_eZ_e]$
$\overrightarrow{{}^{e}R_{c2,i}}$	(3,1) vector	Resultant force of the center of mass of the mobile part of actuator i in frame $[X_eY_eZ_e]$
$\overrightarrow{{}^{e}T_i}$	(3,1) vector	Resultant torque of actuator i in frame $[X_eY_eZ_e]$
$\overrightarrow{{}^{B}R_A}$	(3,1) vector	Resultant force of the mobile base in frame O_B
$\overrightarrow{{}^{B}T_A}$	(3,1) vector	Resultant torque of the mobile base in frame O_B
J	(6,6) matrix	Jacobian matrix
${}^{e}G_i$	(6,3) matrix	Partial angular velocity matrix of the fixed center of mass of actuator i in frame $[X_eY_eZ_e]$
${}^{e}H_{c1,i}$	(6,3) matrix	Partial linear velocity matrix of the fixed center of mass of actuator i in frame $[X_eY_eZ_e]$
${}^{B}G_A$	(6,3) matrix	Partial angular velocity matrix of the mobile base in frame O_B
${}^{B}H_A$	(6,3) matrix	Partial Linear velocity matrix of the mobile base in frame O_B
λ	(6,1) vector	Actuator forces
$u_{i,n}$	number	Auxiliary variables
$X_{\overrightarrow{P}}Y_{\overrightarrow{P}}Z_{\overrightarrow{P}}$	number	The X, Y or Z components of the subscript variable

Table 4.2: Inverse dynamics notations.

Input Data Transformation

${}^{B}R_A$ is calculated from Equation 4.1 for the angles of $\overrightarrow{{}^{B}Pa_E}$.

$$\begin{aligned} \overrightarrow{{}^{B}p_A} &= \overrightarrow{{}^{B}p_E} - {}^{B}R_A\overrightarrow{{}^{A}p_E} \\ \overrightarrow{{}^{B}\gamma_A} &= \overrightarrow{{}^{B}\gamma_E} \end{aligned} \tag{4.7}$$

$$\begin{aligned} \overrightarrow{{}^{B}v_A} &= \overrightarrow{{}^{B}v_E} + \overrightarrow{{}^{B}\omega_E} \times \overrightarrow{{}^{A}p_E} \\ \overrightarrow{{}^{B}\omega_A} &= \overrightarrow{{}^{B}\omega_E} \end{aligned} \tag{4.8}$$

$$\begin{aligned} \overrightarrow{{}^{B}v_{A}} &= \overrightarrow{{}^{B}a_{E}} + \overrightarrow{{}^{B}\alpha_{E}} \times \overrightarrow{{}^{A}p_{E}} + \overrightarrow{{}^{B}\omega_{E}} \times \overrightarrow{{}^{B}\omega_{E}} \times \overrightarrow{{}^{A}p_{E}} \\ \overrightarrow{{}^{B}\alpha_{A}} &= \overrightarrow{{}^{B}\alpha_{E}} \end{aligned} \tag{4.9}$$

$$\begin{aligned} \overrightarrow{{}^{B}Fx_{A}} &= \overrightarrow{{}^{B}Fx_{E}} \\ \overrightarrow{{}^{B}\tau x_{A}} &= \overrightarrow{{}^{B}\tau x_{E}} + \overrightarrow{{}^{A}p_{E}} \times \overrightarrow{{}^{B}Fx_{E}} \end{aligned} \tag{4.10}$$

$\overrightarrow{L_i}$ is calculated by applying the inverse kinematics to $\overrightarrow{{}^{B}p_{A}}$ and $\overrightarrow{{}^{B}\gamma_{A}}$.

$$\overrightarrow{A_i} = \overrightarrow{L_i} + \overrightarrow{B_i} \tag{4.11}$$

Actuator Position Analysis

Considering the auxiliary variables $u_{i,0}$ and $u_{i,1}$ to be

$$\begin{aligned} u_{i,0} &= X_{\overrightarrow{A_i}} - X_{\overrightarrow{B_i}} \\ u_{i,1} &= Y_{\overrightarrow{A_i}} - Y_{\overrightarrow{B_i}} \end{aligned} \tag{4.12}$$

the values of r_i and l_i can be written as

$$\begin{aligned} r_i &= \sqrt{u_{i,0}^2 + u_{i,1}^2} \\ l_i &= ||\overrightarrow{L_i}|| \end{aligned} \tag{4.13}$$

s_i and t_i are known quantities given as parameters of the dynamic model. The ϕ_i and β_i angles are calculated as

$$\begin{aligned} sin(\phi_i) &= u_{i,1}/r_i \\ cos(\phi_i) &= u_{i,0}/r_i \\ \phi_i &= atan2(sin(\phi_i), cos(\phi_i)) \\ sin(\beta) &= \frac{X_{\overrightarrow{A_i}}}{l_i} \\ cos(\beta) &= \frac{r_i}{l_i} \\ \beta_i &= atan2(sin(\beta_i), cos(\beta_i)) \end{aligned} \tag{4.14}$$

${}^{B}R_i$ is calculated from equation 4.1 for the angles $(0, -\beta_i, \phi_i)$;

Actuator Velocity Analysis

From the kinematic model in Figure 4.1(a), the linear and angular velocities of the mobile assembly and cylinders can be written as

$$\begin{aligned}
\dot{\overrightarrow{A_i}} &= \overrightarrow{{}^{B}v_A} + \overrightarrow{{}^{B}\omega_A} \times (\overrightarrow{{}^{B}p_A} - \overrightarrow{A_i}) \\
\dot{r_i} &= cos(\phi_i)X_{\dot{\overrightarrow{A_i}}} + sin(\phi_i)Y_{\dot{\overrightarrow{A_i}}} \\
\dot{l_i} &= cos(\beta_i)\dot{r_i} + sin(\beta_i)Z_{\dot{\overrightarrow{A_i}}} \\
\dot{\phi_i} &= \frac{1}{r_i}(cos(\phi_i)Y_{\dot{\overrightarrow{A_i}}} - sin(\phi_i)X_{\dot{\overrightarrow{A_i}}}) \\
\dot{\beta_i} &= \frac{1}{r_i}(Z_{\dot{\overrightarrow{A_i}}} - sin(\beta_i)\dot{l_i}) \\
\overrightarrow{{}^{e}\dot{\omega}_i} &= \dot{\phi_i}\overrightarrow{{}^{e}k_m} - \dot{\beta}\overrightarrow{{}^{e}j_e} = \begin{bmatrix} \dot{\phi_i}sin(\beta_i) \\ -\dot{\beta_i} \\ \dot{\phi_i}cos(\beta_i) \end{bmatrix}
\end{aligned} \tag{4.15}$$

Actuator Acceleration Analysis

Using the kinematic model, the mobile assembly acceleration can be calculated as follows.

$$\ddot{\overrightarrow{A_i}} = \overrightarrow{{}^{B}a_A} + \overrightarrow{{}^{B}\alpha_A} \times (\overrightarrow{{}^{B}p_A} - \overrightarrow{A_i}) + \overrightarrow{{}^{B}\omega_A} \times \overrightarrow{{}^{B}\omega_A} \times (\overrightarrow{{}^{B}p_A} - \overrightarrow{A_i}) \tag{4.16}$$

$\ddot{r_i}$, $\ddot{l_i}$, $\ddot{\phi_i}$ and $\ddot{\beta_i}$ can be calculated by deriving the expressions of $\dot{r_i}$, $\dot{l_i}$, $\dot{\phi_i}$ and $\dot{\beta_i}$. $\overrightarrow{{}^{e}\ddot{\omega}_i}$ is calculated through the derivation of $\overrightarrow{{}^{e}\dot{\omega}_i}$

$$\begin{aligned}
\ddot{r_i} &= cos(\phi_i)X_{\ddot{\overrightarrow{A_i}}} + sin(\phi_i)\dot{\phi}Y_{\dot{\overrightarrow{A_i}}} \\
\ddot{l_i} &= \frac{1}{l_i}(X^2_{\dot{\overrightarrow{A_i}}} + Y^2_{\dot{\overrightarrow{A_i}}} + Z^2_{\dot{\overrightarrow{A_i}}} - \dot{l_i}^2 + u_{i,0}X_{\ddot{\overrightarrow{A_i}}} + u_{i,1}Y_{\ddot{\overrightarrow{A_i}}} + Z_{A_i}Z_{\ddot{\overrightarrow{A_i}}}) \\
\ddot{\phi_i} &= \frac{1}{r_i}(cos(\phi_i)Y_{\ddot{\overrightarrow{A_i}}} - sin(\phi_i)X_{\ddot{\overrightarrow{A_i}}} - 2\dot{\phi_i}\dot{r_i}) \\
\ddot{\beta_i} &= \frac{1}{r_i}(Z_{\ddot{\overrightarrow{A_i}}} - sin(\beta_i)\ddot{l_i} - \dot{\beta_i}cos(\beta_i)l_i - \dot{\beta_i}\dot{r_i})
\end{aligned} \tag{4.17}$$

$$\overrightarrow{{}^{e}\ddot{\omega}_i} = \begin{bmatrix} \ddot{\phi_i}sin(\beta_i) + cos(\beta_i)\dot{\phi_i}\dot{\beta_i} \\ -\ddot{\beta_i} \\ \ddot{\phi_i}cos(\beta_i) - sin(\beta_i)\dot{\phi_i}\dot{\beta_i} \end{bmatrix}$$

Referring to Figure 4.1(a) the linear acceleration of the two centers of mass of actuator i can be expressed as:

$$\overrightarrow{{}^{e}a_{c1,i}} = \begin{bmatrix} -s_i\left(Y^2_{\overrightarrow{{}^{e}\omega_i}} + Z^2_{\overrightarrow{{}^{e}\omega_i}}\right) \\ s_i\left(Z_{\overrightarrow{{}^{e}\alpha_i}} + X_{\overrightarrow{{}^{e}\omega_i}} Y_{\overrightarrow{{}^{e}\omega_i}}\right) \\ s_i\left(-Y_{\overrightarrow{{}^{e}\alpha_i}} + X_{\overrightarrow{{}^{e}\omega_i}} Z_{\overrightarrow{{}^{e}\omega_i}}\right) \end{bmatrix} \tag{4.18}$$

$$\overrightarrow{{}^{e}a_{c2,i}} = \begin{bmatrix} \frac{s_i}{t_i} X_{\overrightarrow{{}^{e}a_{c1,i}}} + \ddot{l}_i \\ \frac{s_i}{t_i} Y_{\overrightarrow{{}^{e}a_{c1,i}}} + 2\dot{l}_i Z_{\overrightarrow{{}^{e}\omega_i}} \\ \frac{s_i}{t_i} Z_{\overrightarrow{{}^{e}a_{c1,i}}} + 2\dot{l}_i Y_{\overrightarrow{{}^{e}\omega_i}} \end{bmatrix} \tag{4.19}$$

Actuator Inertial Forces and Torques Analysis

$$\begin{aligned} \overrightarrow{{}^{B}g} &= [0.0 \quad 0.0 \quad 9.81]^T \\ \overrightarrow{{}^{e}F_{c1,i}} &= -m_{i,1}\left(\overrightarrow{{}^{B}a_{c1,i}} + {}^{B}A_i^T \overrightarrow{{}^{B}g}\right) \\ \overrightarrow{{}^{e}F_{c2,i}} &= -m_{i,2}\left(\overrightarrow{{}^{B}a_{c2,i}} + {}^{B}A_i^T \overrightarrow{{}^{B}g}\right) \\ \overrightarrow{{}^{e}M_i} &= -\left(\overrightarrow{{}^{e}I_{c1,i}} + \overrightarrow{{}^{e}I_{c2,i}}\right)\overrightarrow{{}^{e}\alpha_i} - \overrightarrow{{}^{e}\omega_i} \times \left(\overrightarrow{{}^{e}I_{c1,i}} + \overrightarrow{{}^{e}I_{c2,i}}\right)\overrightarrow{{}^{e}\omega_i} \end{aligned} \tag{4.20}$$

Mobile Base Inertial Forces and Torques Analysis

$$\begin{aligned} \overrightarrow{{}^{B}F_A} &= -m_A\left(\overrightarrow{{}^{B}a_A} + \overrightarrow{{}^{B}g}\right) \\ \overrightarrow{{}^{B}M_A} &= {}^{B}R_A\left(-{}^{A}I_A{}^{B}R_A^T \overrightarrow{{}^{B}\alpha_A} - {}^{B}R_A^T \overrightarrow{{}^{B}\omega_A} \times {}^{A}I_A{}^{B}R_A^T \overrightarrow{{}^{B}\omega_A}\right) \end{aligned} \tag{4.21}$$

Resulting Forces and Moments Calculation

$$\begin{aligned} \overrightarrow{{}^{e}R_{i,1}} &= \overrightarrow{{}^{e}F_{c1,i}} \\ \overrightarrow{{}^{e}R_{i,2}} &= \overrightarrow{{}^{e}F_{c2,i}} \\ \overrightarrow{{}^{e}T_i} &= \overrightarrow{{}^{e}M_i} + \overrightarrow{{}^{e}i_i} \times \left(s_i \overrightarrow{{}^{e}F_{c1,i}} + t_i \overrightarrow{{}^{e}F_{c2,i}}\right) \\ \overrightarrow{{}^{B}R_A} &= \overrightarrow{{}^{B}F_A} + \overrightarrow{{}^{B}Fx_A} \\ \overrightarrow{{}^{B}T_A} &= \overrightarrow{{}^{B}M_A} + \overrightarrow{{}^{B}\tau x_A} \end{aligned} \tag{4.22}$$

Jacobian Derivation

Considering $\dot{q} = [\dot{q_0}\ \dot{q_1}\ \dot{q_2}\ \dot{q_3}\ \dot{q_4}\ \dot{q_5}]^t$ and using on the expressions of $\overrightarrow{A_i}$, $\overrightarrow{{}^{e}\omega_i}$, $\dot{r_i}$, and $\dot{l_i}$ calculated above we can write

$$\dot{l_i} = J_{i0}\dot{q_0} + J_{i1}\dot{q_1} + J_{i2}\dot{q_2} + J_{i3}\dot{q_3} + J_{i4}\dot{q_4} + J_{i5}\dot{q_5} = [J_{i0}\ J_{i1}\ J_{i2}\ J_{i3}\ J_{i4}\ J_{i5}]\dot{q} \quad (4.23)$$

Where, $\dot{q} = [\dot{q_0}\ \dot{q_1}\ \dot{q_2}\ \dot{q_3}\ \dot{q_4}\ \dot{q_5}]^t = [\dot{\overrightarrow{A_i}}\ {}^e\dot{\overrightarrow{\omega_i}}]^t$. The values of J_{ij} can be derived from the equation above as functions of already calculated entities.

Partial Angular Velocity Matrix Derivation

The partial angular velocity matrix is used to calculate angular velocities of the actuators from the motion of the end–effector. To derive that matrix it is necessary to express ${}^e\dot{\overrightarrow{\omega}}_i$ in terms of $\dot{q_0}$, $\dot{q_1}$, $\dot{q_2}$, $\dot{q_3}$, $\dot{q_4}$, $\dot{q_5}$.

For ease of writing, we will use auxiliary variable $u_{i,2}$ through $u_{i,11}$ to store intermediary expressions.

$$\begin{aligned}
u_{i,2} &= X_{\overrightarrow{A_i}} - X_{\overrightarrow{O_A}} \\
u_{i,3} &= Y_{\overrightarrow{A_i}} - Y_{\overrightarrow{O_A}} \\
u_{i,4} &= Z_{\overrightarrow{A_i}} - Z_{\overrightarrow{O_A}} \\
u_{i,5} &= -sin(\phi_i)/r_i \\
u_{i,6} &= cos(\phi_i)/r_i \\
u_{i,7} &= -u_{i,4}cos(\phi_i)/r_i \\
u_{i,8} &= -u_{i,4}sin(\phi_i)/r_i \\
u_{i,9} &= u_{i,2}cos(\phi_i)/r_i + u_{i,3}sin(\phi_i)/r_i \\
u_{i,10} &= l/r_i \\
u_{i,11} &= -sin(\beta_i)/r_i
\end{aligned} \quad (4.24)$$

Using the variables above and the expression of ${}^e\dot{\overrightarrow{\omega}}_i$ calculated above, ${}^e\dot{\overrightarrow{\omega}}_i$ can be expressed as

$$
\begin{aligned}
\overrightarrow{{}^{e}\dot{\omega}_i} &= \begin{bmatrix} sin(\beta_i) & 0 \\ 0 & -u_{i,10} \\ cos(\beta_i) & 0 \end{bmatrix} \begin{bmatrix} u_{i,5} & u_{i,6} & 0 & u_{i,7} & u_{i,8} & u_{i,9} \\ 0 & 0 & 1 & u_{i,3} & -u_{i,2} & 0 \end{bmatrix} \dot{q} + \\
&\quad \begin{bmatrix} 0 \\ -u_{i,11} \\ 0 \end{bmatrix} [J_{i0}\ J_{i1}\ J_{i2}\ J_{i3}\ J_{i4}\ J_{i5}]\dot{q}
\end{aligned} \tag{4.25}
$$

Hence, matrix ${}^{e}G_i$, for which $\overrightarrow{{}^{e}\dot{\omega}_i} = {}^{e}G_i\dot{q}$, can be expressed as

$$
\begin{aligned}
{}^{e}G_i &= \begin{bmatrix} sin(\beta_i) & 0 \\ 0 & -u_{i,10} \\ cos(\beta_i) & 0 \end{bmatrix} \begin{bmatrix} u_{i,5} & u_{i,6} & 0 & u_{i,7} & u_{i,8} & u_{i,9} \\ 0 & 0 & 1 & u_{i,3} & -u_{i,2} & 0 \end{bmatrix} + \\
&\quad \begin{bmatrix} 0 \\ -u_{i,11} \\ 0 \end{bmatrix} [J_{i0}\ J_{i1}\ J_{i2}\ J_{i3}\ J_{i4}\ J_{i5}]
\end{aligned} \tag{4.26}
$$

Partial Linear Velocity Matrix Derivation

The partial linear velocity matrix makes the connection between the actuators' linear velocities and the end effector motion. Because the fixed part of each cylinder does not have any linear velocity, its ${}^{e}H_{i,1}$ matrix will be zero. The velocity of the mobile actuator shaft is the X component of $\dot{l}_i$, hence

$$
{}^{e}H_{i,2} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} [J_{i0}\ J_{i1}\ J_{i2}\ J_{i3}\ J_{i4}\ J_{i5}] \tag{4.27}
$$

Equations of Motion Derivation Using the Virtual Work Principle

According to the virtual work principle, the sum of virtual work done by all forces and torques should be zero during the time interval δt [8]. In the case of the Stewart platform, these forces are: the forces and torques applied by the mobile base, the forces applied on each of the six cylinders, the torques of the center of mass of the cylinders, and the forces applied by the user on the top base. Considering that the platform undergoes a virtual motion $\dot{q}^*$ that is within the boundaries of the workspace, the following equations hold true:

$$\dot{\phi}^* = J\dot{q}^* \tag{4.28}$$

$$\omega_i^* = {}^eG_i\dot{q}^* \tag{4.29}$$

$$v_{i,1}^* = {}^eH_{i,1}\dot{q}^* \tag{4.30}$$

$$v_{i,2}^* = {}^eH_{i,2}\dot{q}^* \tag{4.31}$$

ω_i^* is the angular velocity of cylinder i. $v_{i,1}^*$ and $v_{i,2}^*$ are the velocities of the two centers of mass of cylinder i. ${}^eG_i$ is the matrix of partial angular velocities of cylinder i and ${}^eH_{i,1}$ and ${}^eH_{i,2}$ are partial velocity matrices of the fixed and mobile centers of mass of cylinder i.

Considering ${}^eR_{i,1}$, ${}^eR_{i,2}$ and T_i the resultants of all the forces and torques applied on cylinder i, ${}^BR_A$ and ${}^BT_A$ the force and torque applied to the mobile base by the user, and finally considering τ the vector of forces and torques applied by the end–effector, the virtual work equation can be written as:

$$\dot{\phi}^{*T}\tau\delta t + \left(\sum_{i=0}^{5} v_{i,1}^{*Te}R_{i,1} + \sum_{i=0}^{5} v_{i,2}^{*Te}R_{i,2} + \sum_{i=0}^{6} \omega_i^{*Te}T_i\right)\delta t + \begin{pmatrix} {}^BR_A \\ {}^BT_A \end{pmatrix}\dot{q}^*\delta t = 0 \tag{4.32}$$

Substituting equations 4.28, 4.29, 4.30 and 4.31 in equation 4.32 and eliminating δt and $\dot{q}^*$

$$J^T + \sum_{i=1}^{6} {}^{e}H_{i,1}\,{}^{e}R_{i,1} + \sum_{i=1}^{6} {}^{e}H_{i,2}\,{}^{e}R_{i,2} + \sum_{i=1}^{6} {}^{e}G_i\,{}^{e}T_i + \begin{pmatrix} {}^{B}R_A \\ {}^{B}T_A \end{pmatrix} = 0 \tag{4.33}$$

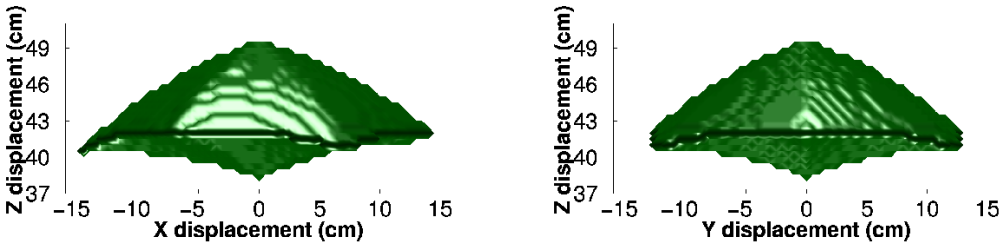
All the terms appearing in equation 4.33 can be derived using inverse and forward kinematics. The Jacobian calculation was presented in section 4.4.2. From equation 4.33, τ can be calculated using the Gaussian elimination method. From τ, the cylinder forces can be computed using the inverse Jacobian.

4.5 Platform Workspace

A "C" program was developed to compute the workspace of the RMA robot. A bounding box of the workspace was calculated by physically measuring the minimum and maximum positions that the platform could reach. The resulting space was then sampled at constant intervals in all directions. For each point the inverse kinematics were performed to calculate the actuators' positions if the end–effector was positioned there. If the calculated actuator lengths were within the physical limits, then the point was part of the workspace. If the calculated actuator lengths were exceeding the physical limits the point was discarded.

Following an approach similar to Badescu's solution for calculating the angular workspace [5], several mobile base orientations are tried for each sampled point. The orientations are chosen by equidistantly sampling the space defined by the minimum and maximum angles the robot can reach in each of three directions. For each orientation that is achieved, a counter attached to the sampled point is increased. Thus, the resulting data will offer a better understanding of the orientational workspace of the platform at various workspace points.

In the dual platform configuration, the workspace of the system is the union of each platform's workspace (Figure 4.4) considering the distance between the mobile bases. Considering that the workspace of the platform does not extend horizontally



(a) View Along the X–axis

(b) View along the Y–axis

Figure 4.4: Platform workspace [11].

beyond the limits of a circle circumscribing its fixed base, the interference between two neighboring platforms is determined by the amount of overlapping between the two circles. The largest overlap occurs in the close configuration (Figure 3.12(b)). The fixed base circumscribed circles overlap by 63 mm so the combined workspace of two platforms will have a butterfly shape.

4.6 Platform Force and Torque Output

The force and torque output of the platform were computed by adding the cylinder vectors, scaled by the minimum or maximum forces that they could apply. The two cylinder forces (i.e. minimum and maximum) are not equal in absolute value because the piston area is not equal in both the upper and lower chambers. The maximum force (outward) output of one cylinder is 581 N. The minimum force (inward) is 525.7 N. The maximum lifting force (Z–axis) that the platform can apply in various points with a flat orientation is presented in Figure 4.5.

The maximum and minimum torques that the platform could generate were computed by maximizing the cylinder vector sums on the X and Y directions separately. First, the unit torques were calculated (i.e. summing the normalized vectors without scaling them). Then each torque vector was scaled back using the maximum cylinder force if the torque sign was positive and the minimum cylinder torque if the torque

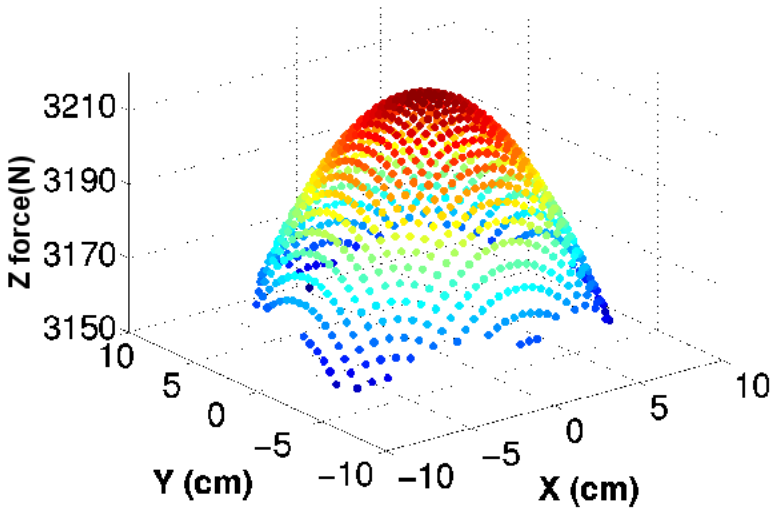
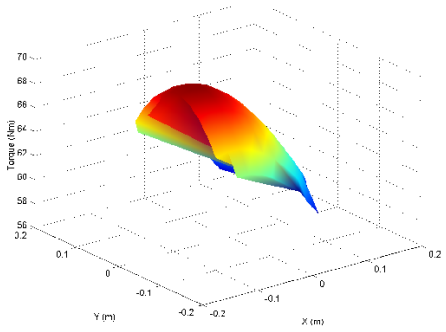
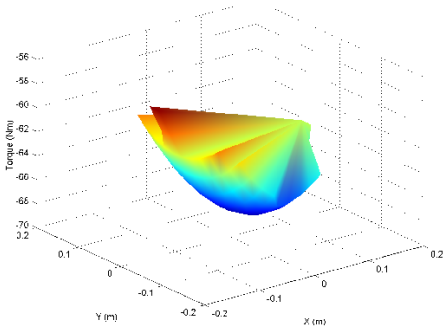


Figure 4.5: Maximum upward force (Z–axis) [11].

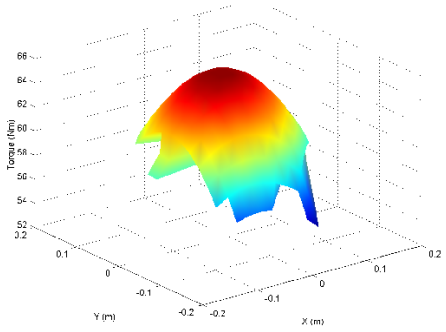
sign was negative. The resulted torques in each point of the workspace are presented in Figure 4.6.



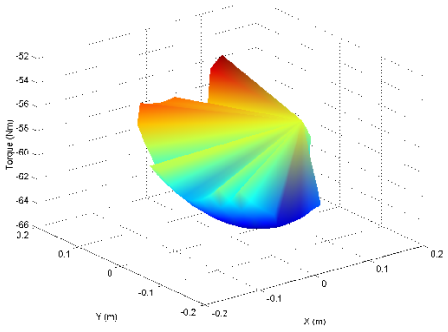
(a) Maximum pitch torque (Y–axis)



(b) Minimum pitch torque (Y–axis)



(c) Maximum roll torque (X–axis)



(d) Minimum roll torque (X–axis)

Figure 4.6: Platform torque output.

Chapter 5

Servo Controller

The design of the Mobility Simulator requires each of the Rutgers Mega–Ankle robots to perform several tasks such as: sustain the user's weight, follow the motion of the user's foot, move the front foot backward once contact with the virtual surface has been established, or shake, jolt, or resist the motion of the foot based on the virtual surface properties. To accomplish these tasks the RMA platforms must be able to function in either position control mode or force control mode. Furthermore, they must be able to compensate for external disturbances and must easily switch between these functioning modes based on the commands from the virtual reality simulation.

The RMA robots have Stewart platform architecture and use pneumatic actuators. The control of both platforms is executed by the control interface presented in chapter 3. The servo–controller architecture is divided in three loops (Figure 5.1). The inner most loop implements the pressure control for each actuator air chamber. The middle loop implements the cylinder level position and force control. The outer most loop implements the platform level control including position, force and external disturbance compensation.

5.1 Pressure Control

Each cylinder chamber is connected to the controller interface through a pneumatic tubing. Inside the interface, this tubing is split in two branches, one branch being connected to the main air intake and the other one to the main exhaust. Each of the two branches is controlled by a separate solenoid ON/OFF valve. The air pressure inside the tubing and the cylinder chamber is regulated by opening or closing the solenoid valves. Based on a desired change in pressure, the pressure control loop calculates the duration each of the two valves should be kept open. This approach is known as Pulse–Width Modulation (PWM).

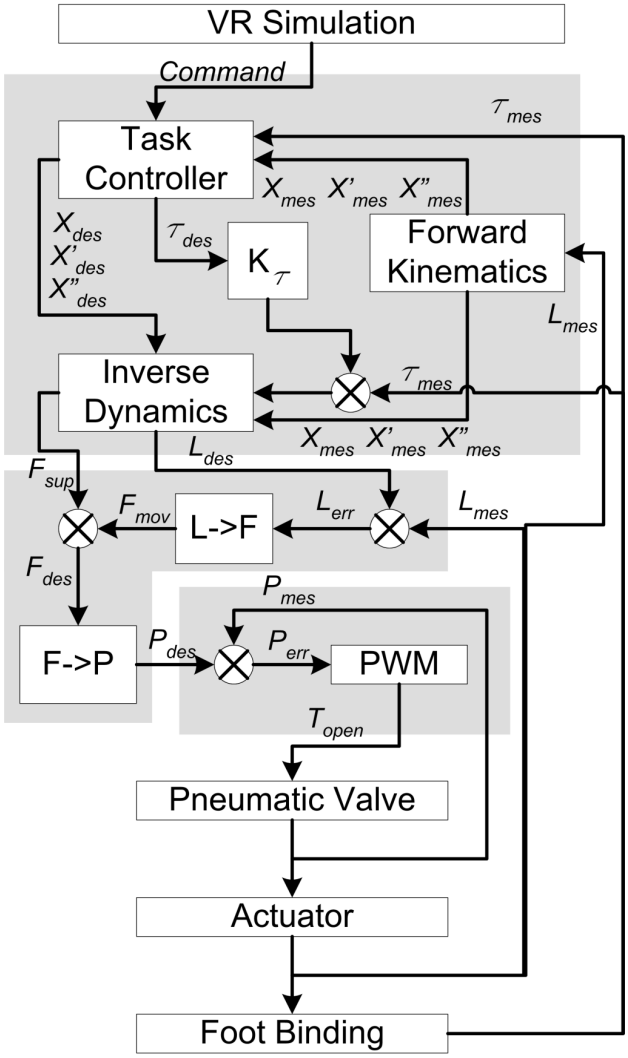


Figure 5.1: Controller diagram.

The transition function from desired pressure change to valve opening time is implemented using a proportional gain and a derivative gain. The PD controller performed well so that an integral term was not necessary. During testing, the integral component has been tested but it has been determined that the improvement it brought was minimal, while it created the risk of integrator windup. Such a situation occurred frequently when applying external disturbances with the same frequency as the controller bandwidth. Such disturbances caused the controller to lag behind, hence creating a continuous error that accumulated in the integrator term. When the disturbance was discontinued, the integral factor caused the pressure values to skyrocket and then very slowly to come back to the desired values. The initial form of the pressure control

transition function is shown in Equation 5.1.

$$T_{valve} = k_p P_{error} + k_d \dot{P}_{error} \tag{5.1}$$

Solenoid Valve Dead–Band

In its initial form, the PD controller displayed oscillations of the controlled pressure. Reducing the k_p gain eliminated the oscillations but the steady state error increased significantly. Studies on the solenoid valves' reaction time [119] determined that after the moment are open/close command was issued, there was a time delay of about 4 ms until any change in pressure was measured. The delay, called *dead–band*, varied from valve to valve depending mostly on the age of the hardware. To compensate for this delay, each T_{valve} was added the valve's dead–band (Equation 5.2).

$$T_{valve} = k_p P_{error} + k_d \dot{P}_{error} + T_{DeadBand} \tag{5.2}$$

Cylinder Chamber Volume Compensation

The control scheme presented above functioned satisfactorily when applied to the smaller Rutgers Ankle robots. In the case of the Rutgers Mega–Ankle platforms, the same implementation was characterized by larger steady state errors for large changes in pressure. The only difference between the two robots that could affect the pressure control was the cross section of the pneumatic actuators. It has been determined that for large changes in pressure it was not enough to consider only the pressure in calculating the output time, but also the volume of uncompressed air that had to be pushed into the cylinders. The volume of air compensation is presented in Equation 5.3. The response of the pressure controller to step and sinusoidal inputs are presented in Figures 5.2, 5.3, and 5.4.

$$T_{valve} = k_p P_{error} + k_d \dot{P}_{error} + T_{dead--band} + k_v V_{uncompressed} \tag{5.3}$$

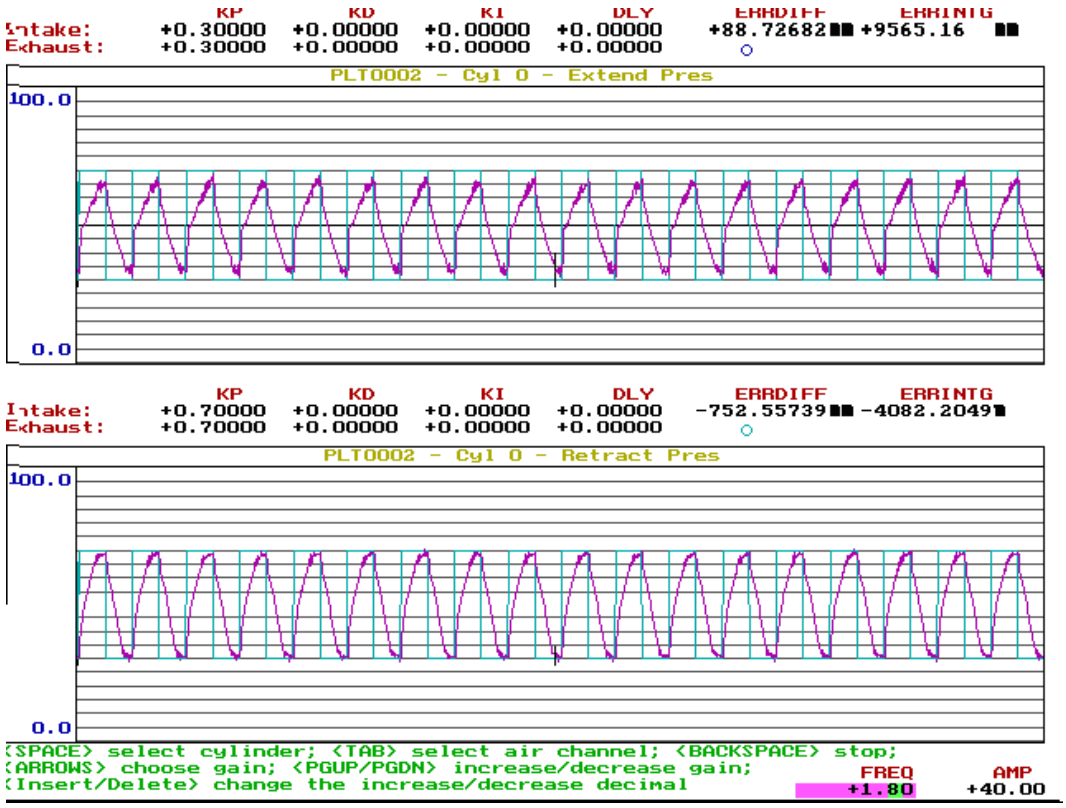


Figure 5.2: Pressure control response to a step input of 40 PSI amplitude and 1.8 Hz frequency.

5.2 Cylinder Control

By controlling the air pressure in the lower and upper chambers of a double acting cylinder it is possible to control the force applied by the piston along the cylinder axis as well as its displacement.

5.2.1 Cylinder Force Control

The force applied by the cylinder shaft is a direct consequence of the pressures in the two chambers of the cylinder. This direct dependency allows the controller to use an open loop strategy. The task of the force control is to calculate the new desired pressure of each cylinder chamber based on the desired force of the actuator. There are two important aspects of this transformation:

1. The change in pressure in each cylinder chamber should be minimal. Thus, the

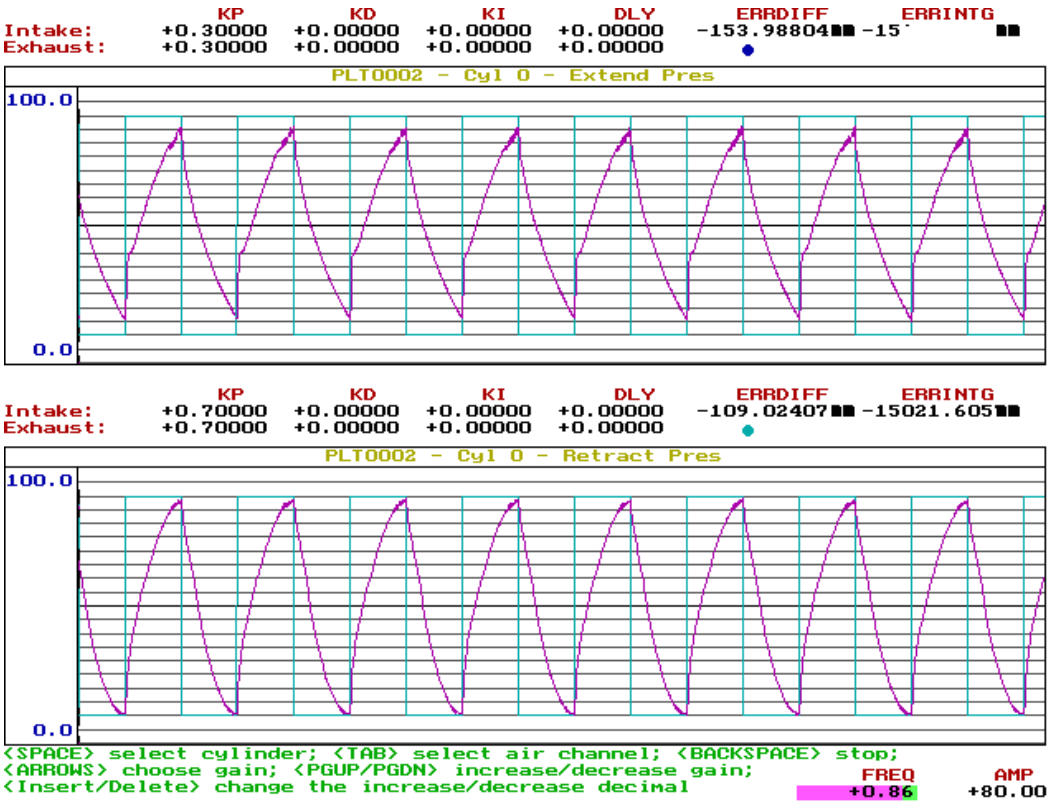


Figure 5.3: Pressure control response to a step input of 80 PSI amplitude and 0.86 Hz frequency.

transformation of the cylinder force into two pressures has to take into account the current pressures in the cylinder chambers.

2. The pressures in the two chambers should be balanced. If the previous requirement cannot be achieved, choose pressures by subtracting equal amounts from the equilibrium pressures.

For instance, a force of $20N$ can be obtained by having the two cylinder chambers apply any of the following $(lower, upper)$ pairs of forces: $(100N, 80N)$, $(20N, 0N)$, or $(400N, 380N)$. The controller has to pick the one closest to the current pressures in the cylinder chambers. Considering P_L and P_U as the measured pressures in the lower and upper chamber of the cylinder, s_U and s_L as the area of the two cylinder chambers, and F_d as the new desired pressures for the two chambers P_{dU} and p_{dL} are calculated using the formula in 5.4.

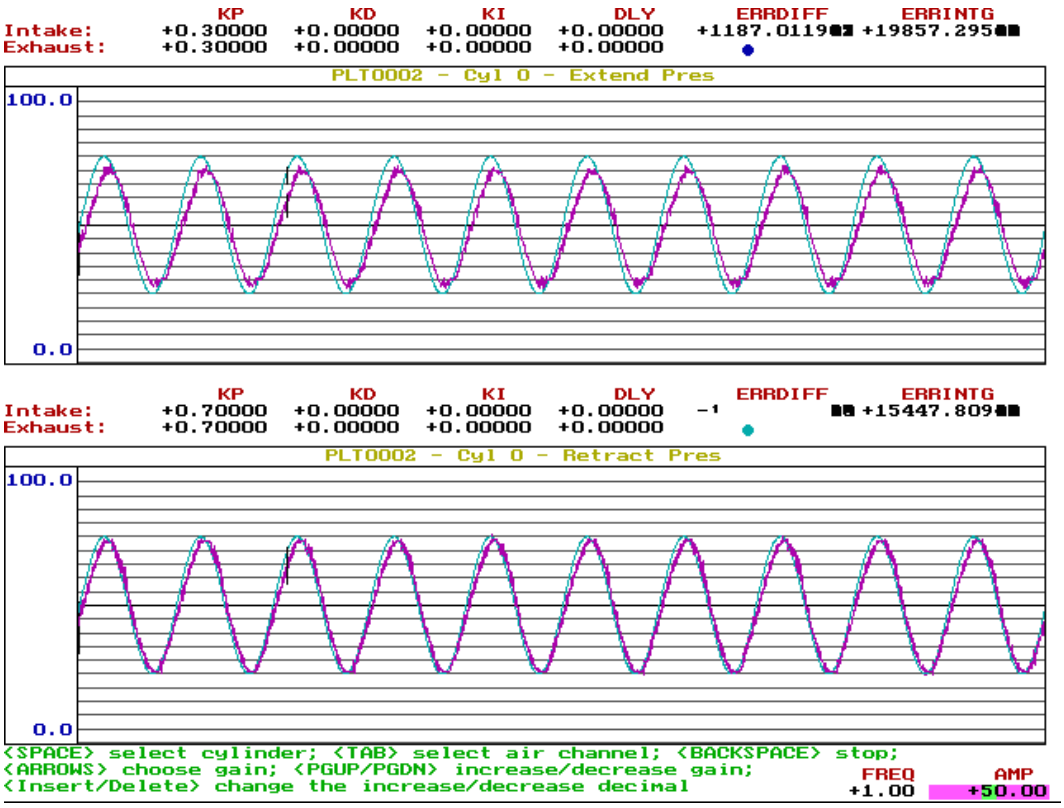


Figure 5.4: Pressure control response to a sinusoidal input of 50 PSI amplitude and 1.0 Hz frequency.

$$\begin{aligned} P_{diff} &= \frac{P_L * s_L - P_U * s_U - F_d}{s_U + s_L} \\ P_{dL} &= P_L - P_{diff} \\ P_{dU} &= P_U - P_{diff} \end{aligned} \tag{5.4}$$

The resulting solution (P_{dL}, P_{dU}) can be mathematically correct and yet invalid. For instance, $20N$ of force can be obtained by applying the following pair of pressures $(580N, -600N)$. In such cases, the force difference is calculated around half of the maximum controllable pressure (Equation 5.5).

$$\begin{aligned} P_{diff} &= \frac{0.5 F_d}{s_L} \\ P_{dL} &= \frac{0.5 P_{CtlMax}}{s_L} \\ P_{dU} &= \frac{s_L}{s_U}(0.5 P_{CtlMax} - P_{diff}) \end{aligned} \tag{5.5}$$

Because the two chambers of the pneumatic cylinder do not have the same area, it is possible for Equation 5.5 to yield negative values forP_{dL} or P_{dU}. In such case, the minimal change and the balanced pressure goals are not enforced anymore, and the forces are calculated by applying the minimum controllable pressure in one chamber and the difference in the other one (if not greater than the maximum controllable pressure).

5.2.2 Cylinder Position Control

While force control can be implicitly done through pressure control, the position control cannot follow the same approach, because there is not direct relationship between the air pressure in the chambers and the position of the piston. The controller in this case is a closed–loop PD. The loop is closed by the linear potentiometers that measure the displacement of each piston.

In order to position the cylinder shaft, the controller has to calculate a cylinder force to be applied to the shaft based on the change in displacement. In addition, the controller has to compensate for the external disturbances constantly applied to the platform, and hence to the pneumatic cylinders, by the user's foot. The cylinder level disturbances are calculated using the Jacobian from the platform level disturbances measured by the force sensor. Hence, the input to the position controller consists of a new desired position X_d and a new desired force F_d calculated to compensate for disturbances. Based on these two inputs, the control function computes the new cylinder desired force, and from this force computes the desired pressures in the cylinder chambers. The control function in its initial form is

$$F_d = k_p X_e + k_d \dot{X}_e + F_d \tag{5.6}$$

The model above functioned properly only for light external disturbances. A heavier load on the cylinder would cause it to either have a large steady state error or, if the gains were increased, to become unstable by entering oscillations with increasing amplitude. It has been initially expected that adding the cylinder measured disturbance F_d will be sufficient for reaching the target under external forces. However, the pressure control

bandwidth being as low as 2 Hz, the delay between the moment the external disturbance was measured and the response to it was large, and hence the tracking error was not reduced.

To solve this behavior it was necessary to add adaptive factors to the proportional and derivative gains in the classical PD model above.

$$\begin{aligned} k_{p_{adaptive}} &= k_p(1 + k_{pf}F_d) \\ k_{d_{adaptive}} &= k_d(1 + k_{df}F_d) \end{aligned} \tag{5.7}$$

The k_{pf} and k_{df} gains were given small positive values. The effect of the adaptive component was to increase the proportional and the derivative gain with a fraction of the external disturbance. This permitted lower k_p gains for better stability, but also caused the gains to increase to overcome the effects of the external forces. The k_{df} gain was also proportionally increased with the disturbance in order to reduce the oscillations caused by the increased k_p values.

The model described above, handles the control of the robot's position while resisting external forces. During the operation of the simulator, the control mode has to be switched from force to position quite often. This is accomplished through a switch variable that cancels out the positional term from the transition function. The complete implementation of the controller is presented in Equation 5.8. The responses of the position controller to step inputs of various amplitudes and frequencies are presented in Figures 5.5 and 5.6.

$$F_{d_{adaptive}} = k_{switch}\left(k_p(1 + k_{pf}F_d)X_e + k_d(1 + k_{df}F_d)\dot{X}_e\right) + F_d \tag{5.8}$$

5.3 Platform Control

The outer–most control loop handles the robot's behavior as a whole. The position and force control at platform level implement an open–loop model. This is possible because of the direct relation between the position and force of the actuators and the position and force of the platform. Using the forward and inverse kinematics and the inverse

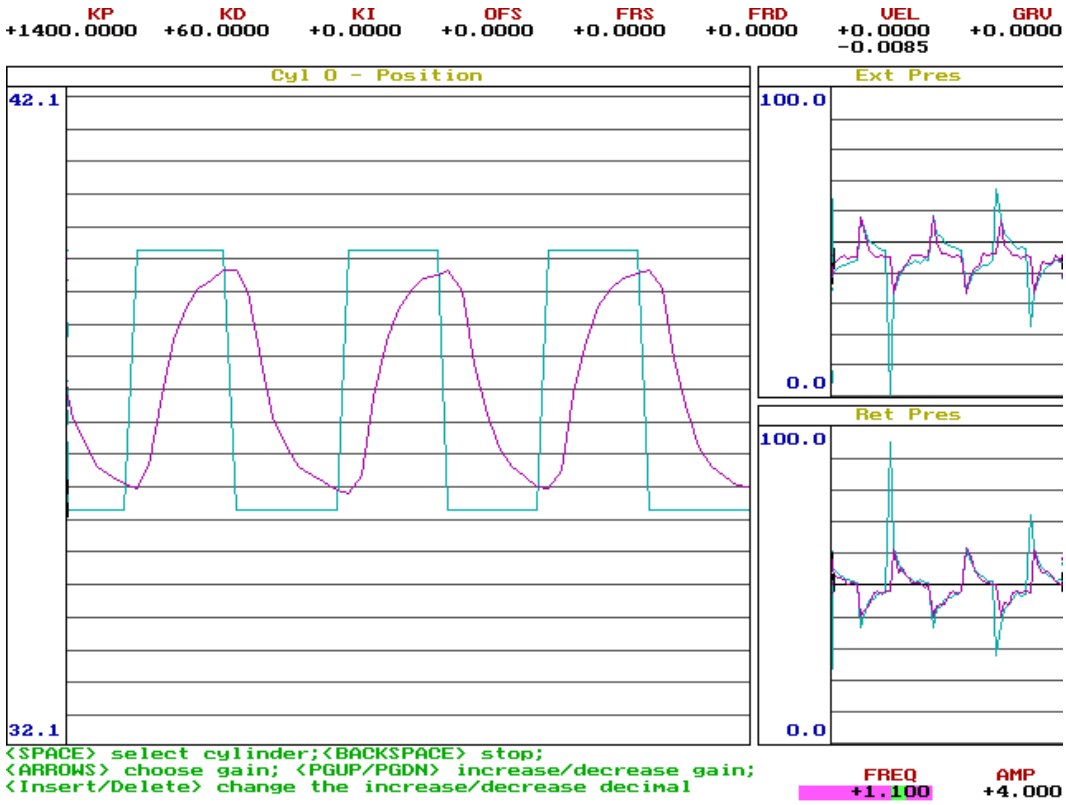


Figure 5.5: Cylinder position control response to a step input of 4 cm amplitude and 1.1 Hz frequency.

dynamics, the desired motion and force of the platform is transformed in cylinder space, where the control scheme is simpler and faster.

The choice to execute the control solely in cylinder space relies mainly on the need for real–time response of the robot. The cylinder control implementation is compact and very fast because it does not involve complex kinematic and dynamic calculations. On the other hand, the kinematics and dynamics of the platform are much slower, and cannot be run in a timely manner. Achieving a constant execution rate for the platform level loop would require the introduction of delays meant to keep the iterations' duration constant.

The duration of one iteration can vary significantly depending on the position of the platform. This is due to the iterative process necessary to solve the forward kinematics. The convergence condition is not satisfied at the same point for all platform positions. Because the platform–level operations are too complex and slow to be run inside an

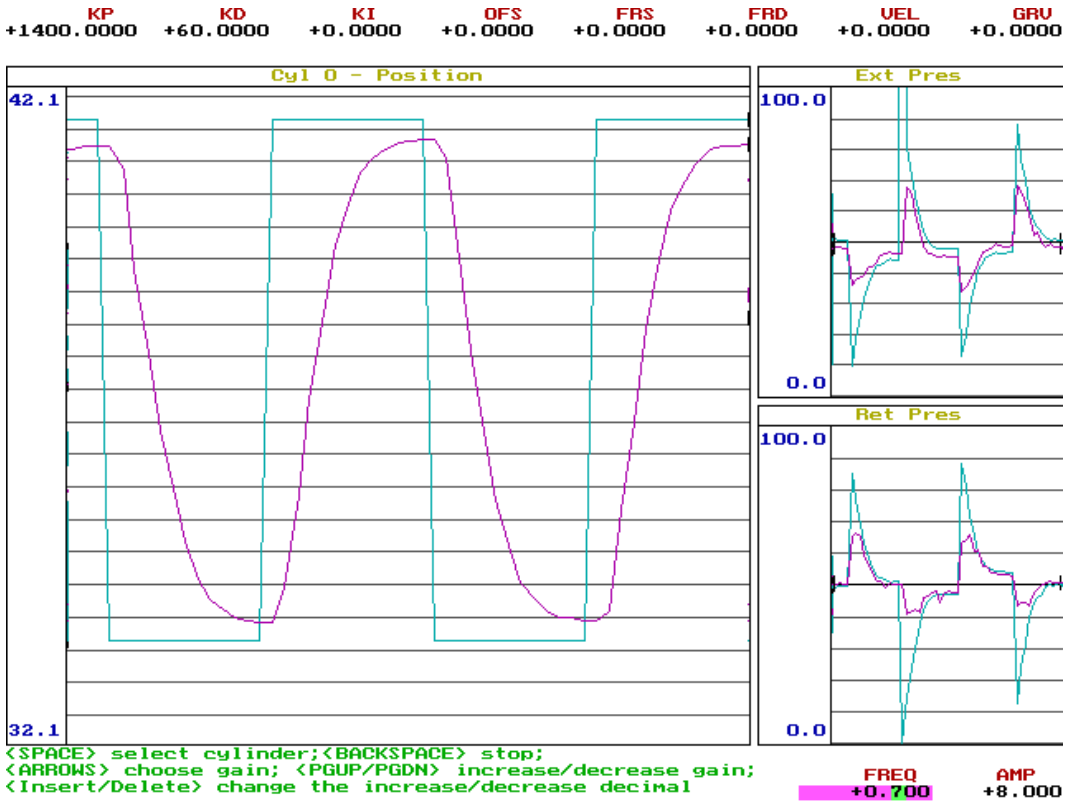


Figure 5.6: Cylinder position control response to a step input of 8 cm amplitude and 0.7 Hz frequency.

interrupt handler, it is necessary to execute them as part of the main program. Another reason for the varying loop rate is the serial communication that the main program has to handle at the same priority as the platform level control.

Hence, a choice was made not to try to stabilize the rate of the outer loop to avoid the delays such an approach would create in the communication with the host PC. The kinematics and dynamics are executed as fast and as often as possible, but the actual robot behavior is controlled at actuator level by a timely accurate loop run inside an interrupt handler.

5.3.1 External Force Compensation

During a simulation, the RMA robots function in two main modes: position control or force control. The position control poses the problem of resisting external disturbances while keeping the desired position. This model requires a combination of force and

position control, which is usually difficult due to the inherent incompatibility of the two control modes. Applying a force in a certain direction implies an acceleration that will change the position of the robot. The combination of these two modes is implemented at cylinder level. The role of the platform level loop is to provide the cylinder control loop with the forces each cylinder has to resist.

An earlier version of the system used impedance control to resist external forces [119]. The controller used the measured forces to calculate a new desired position for the platform. The position was calculated so that the platform would move against the direction of the forces.

$$P_{d_{impedance}} = P_d - k_{impedance} F_{measured} \tag{5.9}$$

The impedance control yielded good results for the system in sitting. For the mobility simulator, the impedance control failed due to lack of accuracy in the calculated feedback. To sustain the weight of a person the platforms had to function very close to their maximum capacity, outputting large forces and torques in order to balance the user. The transformation of force into position change modeled by the impedance is an inaccurate approximation of the feedback, which failed to correlate the actuators properly to resist such large forces and torques. The final version of the controller uses the inverse dynamics algorithm presented in chapter 4.

The inverse dynamics calculate accurately the external forces at cylinder level given the forces measured by the force sensor and the measured position, velocity and acceleration of the platform's end–effector. For more usage flexibility, the measured forces are scaled by a responsiveness gain before being input to the inverse dynamics. This permits to easily adjust the behavior of the robot with regard to disturbances. The default value of the responsiveness gain is 1.0.

$$F_{input} = k_{resp} F_{measured} \tag{5.10}$$

5.3.2 Back–drive Mode

The RMA robot functions in back–drive mode whenever it has to follow the user's swinging foot. In these situations, the platform has to compensate for its own weight and apply the same forces the user applies to the end–effector. Thus, the back–drive mode is essentially force control mode with a negative responsiveness gain. While the inverse dynamics take into account the mass and inertial tensors of the cylinders and mobile base of the platform, the back–drive mode is obtained by just setting the responsiveness gain to -1.0.

With a gain of -1.0 the platform follows the user's foot, without resistance, but only for very small forces. While a gain of 1.0 is sufficient to resist very large forces, the -1.0 gain is not sufficient to follow them. This can be explained by the fact that when following the foot's motion, the forces recorded by the force sensor change very fast and the cylinder level force and the pressure control loops are not fast enough to follow the change in forces. Essentially, the Rutgers Mega–Ankle robot itself has its own elasticity and damping created by the controller and by the limited air input to the cylinder. In order to overcome the resistance when following high forces, it was necessary to increase the absolute value of the responsiveness gain. The gain also had to be given different values based on the direction of the force it was applied to. This is caused by the number of cylinders affected by the force. For instance, an upward vertical force, requires an identical change in force in all six cylinders. This becomes a problem given the limited size of the main air input. A horizontal force causes an equal number of cylinders to intake and exhaust air, hence creating less resistance from the main air intake size. The table below presents the responsiveness gain values used during back–drive mode.

Table 5.1: Back–drive gains.

	X	Y	Z
Linear	-10.0	-10.0	-40.0
Angular	-16.0	-16.0	-16.0

5.3.3 Mechanical Bandwidth

The mechanical bandwidth of the Rutgers Mega–Ankle platform has been measured using the step response for several amplitudes of the input signal and along all directions separately. Figure 5.7 shows the platform response to a step input signal of amplitude 10 cm and frequency 0.8 Hz. The bandwidth measurement results are summarized in Table 5.2.

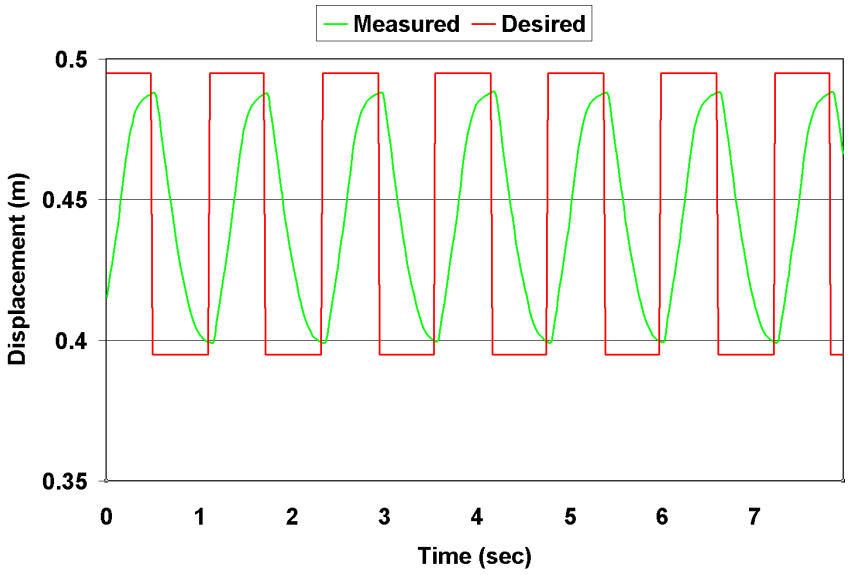


Figure 5.7: Platform response to a step input function of amplitude 10 cm and frequency 0.8 Hz along the Z–axis.

The bandwidth is not identical along the three directions. On the Z–axis, the bandwidth is 60% lower than on the Y–axis and 64% lower than on the X–axis. When oscillating along the Z–axis, the platform has to make larger position changes in all cylinders than when moving along the X or Y axes. The difference in bandwidth along the Y and X axes is explained by the asymmetry of the platforms in the XY plane.

Table 5.2: Rutgers Mega–Ankle Mechanical bandwidth.

	Axis X	**Axis Y**	**Axis Z**
Amplitude 5 cm	2.5 Hz	2.3 Hz	1.0 Hz
Amplitude 10 cm	2.2 Hz	2.0 Hz	0.8 Hz

5.3.4 Robot Stability in Foot Support Mode

One of the first problems encountered during the development of the system was the stability of the RMA platforms under load. The robots were stable when subjected to external forces if there was no load attached to them. However, the working regime for which they were developed, involves resisting forces while supporting the weight of the user. Figure 5.8 shows the response of the robot to a 0.5 Hz sinusoidal input with amplitude of 0.18 m. The load on the robot was 50 lbs strapped rigidly to the foot binding. Under load, the motion was distorted and the amplitude of the robot increased slowly eventually becoming unstable.

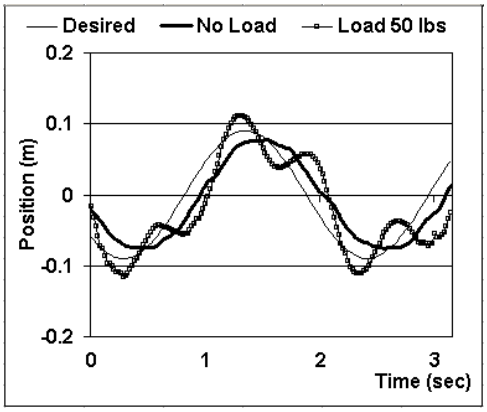


Figure 5.8: RMA platform response to a sinusoid input along the Y–axis (back–front) with 0.5 Hz frequency and 0.18 m amplitude.

The cause of this problem was the addition of the desired cylinder force F_{des} with the cylinder motion force F_{mov}. Under load, the resulting forces were too high and caused the robot to become unstable. Increasing the derivative gain slowed down the instability but it didn't solve it. The solution was to reduce the proportional gains by a minimum of 42%. With lower gains, the platform was stable under load, but had very little power to move the user's foot backward during the foot support phase, hence making the system unusable. In addition, when the system was unloaded, the steady state error was significantly larger. The use of an integrator term was avoided because the usage of the system caused it to windup consistently.

Two adaptive gains were used to bring the robot to respond properly under load. The gains added a fraction of the measured cylinder load to the proportional and

derivative gains respectively. The proportional adaptive component helped increase the moving force of the platform when under load, while the derivative adaptive component was increased to compensate for the high proportional gains and insure the stability of the system.

Figure 5.9 presents the response of the platform to the same sinusoidal input signal, under a 50 lbs load, with and without the adaptive component added to the lowered proportional and derivative gains. While both the constant and the adaptive response were stable, the adaptive strategy provided the necessary power to move the load closer to the desired position, and reducing the error by approximately half. The adaptive gains did bring a side effect slightly visible in Figure 5.9; when the robot moves with higher velocity, the adaptive derivative gains increases the damping of the system slowing it down and the releasing it as the load on the robot shifts and the force is reduced. This can be seen as a change in the slope of the adaptive curve in Figure 5.9.

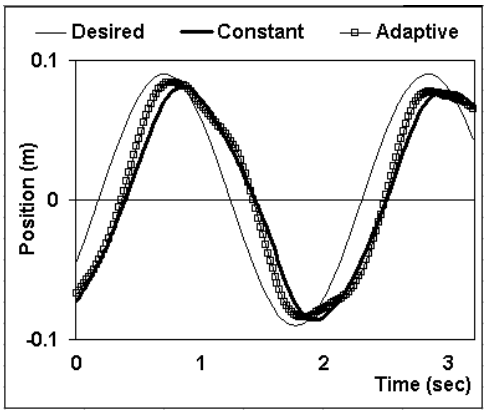


Figure 5.9: Response comparison lower constant gains with and without the adaptive component.

A more evident effect of the adaptive strategy is illustrated in Figure 5.10. Having the robot standing still in the center of the workspace, a horizontal disturbance (jolt) is applied along the Y–axis. Without the adaptive gains, the robot becomes unstable and spins out of control. With the adaptive gains, the disturbance is counteracted and the robot comes back to the desired position. The 4 seconds long recovery time of this test is shorten during usage, because the user's foot provides sufficient damping to make the stabilization much quicker.

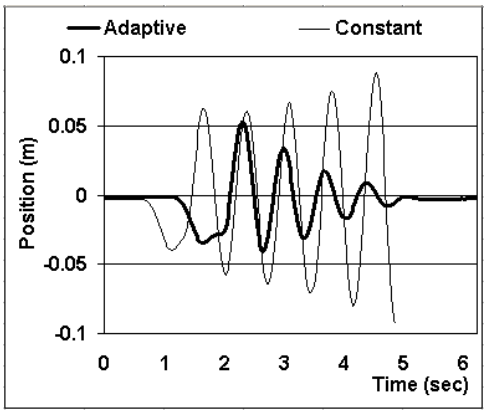


Figure 5.10: Robot response to a jolt along the Y–axis.

5.3.5 Force Minimization in Free Motion Mode

The second major functioning mode of the RMA robot is following the foot during the swing phase of the gait. In this mode, the platform has to compensate for its own weight and for the forces applied by the user to the end–effector, so that the user can swing the foot without effort. To achieve this, the servo controller disables the cylinder position control by canceling out the moving force F_{mov}, and by switching the measured forces signs by changing the value of K_t from 1 to -1. While these changes cause the RMA robot to follow the user's foot, the motion is very slow and large forces are felt at the foot. Figure 5.11 shows the forces measured at the foot during one swing phase.

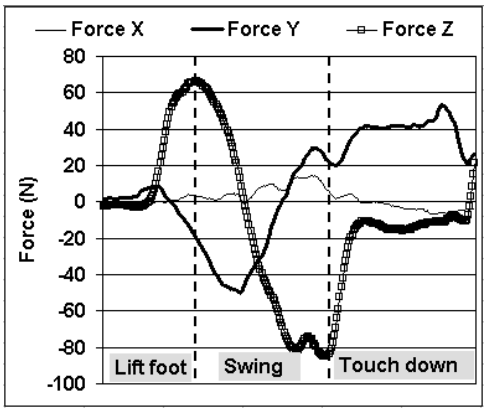


Figure 5.11: Free mode forces during swing for K_t = -1.

The source of these large forces has been determined to be the damping created by the pneumatic actuators. While a K_t of 1 is sufficient to resist forces in foot support

mode, in free mode, besides responding with a force to the user's force, the robot also has to move, which involves a much large air intake/exhaust activity. The damping is mostly coming from airflow limitations imposed by the small intake and exhaust sections of the cylinder chambers. To overcome this problem, K_t was increased in absolute value. The increase was done for each of the robot's 6DOF. The Z–axis (up–down) translation gain was approximately four times larger than the rest of the gains, because the motion on that direction required all the cylinders to either intake or exhaust, hence putting more airflow in a single direction. The measured forces for the increased K_t gains are shown in Figure 5.12. The forces are now reduced approximately eight times to a maximum of 11 N, which is comparable to the weight of a snow boot.

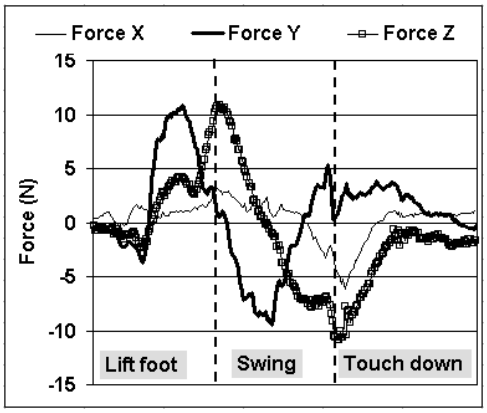


Figure 5.12: Free mode forces during swing for larger K_t.

5.4 Controller Task Scheduling

To insure proper timing accuracy, the software was designed to use hardware interrupt handlers to execute the controller tasks. Such tasks included sensor readings, pressure control, position control, etc. The hardware interrupts were generated by one of the three A/D I/O boards used inside the haptic control interface.

During development, the controller brought up run–time problems such as wrong time measurement, and slow or unstable serial port communication. Both these aspects are crucial to the system. Inaccurate time measurement results in control instabilities, while a lack of reliable serial communication makes the system unusable for integration

with a VR simulation. The source of the problems was traced to be the amount of work done by the control tasks in the hardware interrupt handler. The controller time was measured by dividing the number of interrupts started by the hardware timer by the timer frequency. The result was approximately 40% less than the real duration measured by an independent stopwatch. This was the effect of overloaded interrupts. If an interrupt runs for longer than its period, the next interrupt will be activated one period later than normal. That means that the interrupt counter will be less than it should be by one interrupt, hence the time measurement error. The cause of the unstable serial port communication proved also to be caused by these long interrupt handlers, since it relied on a separate interrupt triggered at 200 KHz. Being delayed too much by the control interrupt handlers, the serial communication routine was missing bits transmitted by the PC host.

While purely an implementation issue, this behavior was posing a serious problem to the proper functioning of the system. In order to solve it, the controller task durations were measured and then broken down into pieces that were small enough to fit inside the period of the control interrupts. Once the tasks were divided into smaller pieces, it was necessary to optimize the order of their execution. Without this step, the controller would execute too many sub–tasks in the same interrupt handler, hence getting back to the initial behavior.

5.4.1 Controller Tasks

During the simulation, the low level controller has to execute the following tasks:

1. *Platform level control: kinematics and dynamics.* As discussed above these operations are too long to be executed inside an interrupt handler.

2. *Cylinder level control: position and force.* Executed for 12 cylinders (6 for each RMA platform)

3. *Pressure control.* Executed for 24 air chambers. It is executed with the frequency of the PWM controller.

4. *Read sensors.* Each platform is associated with 24 sensors: 12 pressure sensors, 6 linear potentiometers measuring the actuator displacement, and 6 force readings from the force sensor.

5. *Close valves at the end of their PWM cycle slice.* This is a very short task that has to run very fast, its frequency being the product between the PWM control frequency and the PWM resolution

Task Frequencies

The frequencies of these tasks are defined by the mechanical ON/OFF frequency of the solenoid valves. While the vendor data specify that the valves can work at a maximum 300 Hz open/close frequency, during testing it was determined that the prolonged usage required by the simulator caused the electronic components to over–heat at this maximum frequency. To protect the hardware, the valves are currently used at a 250 Hz frequency. This frequency is used for the pressure PWM controller.

The resolution of the PWM controller is the number of intervals into which the PWM period is divided. This value defines the time accuracy of the controller. Empirically it was determined that the optimal resolution is 30. Thus, the frequency of the valve–closing task is $7500Hz$.

The frequencies of the rest of the tasks were chosen empirically using as criteria the fact that outer loop frequencies should be less than the inner loop frequencies. Because of the different control task frequencies, the sensor readings were divided so that the pressure sensors were read at a higher frequency as needed by the pressure control while the others were read at a lower rate. The tasks and their frequencies are presented in Table 5.3. The timer interrupt frequency was set to the largest of the task frequencies, in this case 7.5 KHz.

Task Durations

The duration of each task was measured by executing it separately for a large number of repetitions and dividing the total time by the number of repetitions. As a result,

Table 5.3: Controller tasks and frequencies.

Task	*Frequency*
Cylinder level control	100 Hz
Pressure control	250 Hz
Close valves	7500 Hz
Pressure sensor readings	500 Hz
Position and force sensor readings	100 Hz

it became apparent that the longest task was reading the sensors. The sensor reading task was then divided into 4, 8 and 12 subtasks and each of those were measured as well. The separation between pressure sensor readings and the rest of the tasks was done by having a part of the sub–tasks handle only pressure sensors. The measured durations are shown in microseconds in Table 5.4.

The overhead of starting the interrupt handler was measured to be 0.000005 microseconds.

5.4.2 Static Scheduling Algorithm

An interrupt's activity should not last longer than the period with which it is generated. This requirement is particularly difficult for the system presented here because the amount of work to be done is close to the limit of what the processor can do and because the controller is designed to support not only a dual–platform configuration but also, single platform or dual–haptic glove configurations too. To automatically accommodate the changes in control work caused by the devices connected, a static load balancing routine was implemented to distribute the work uniformly across interrupt handlers. The load–balancing algorithm considered all the subtasks with their durations and frequencies, and assigned each of them a starting interrupt so that the maximum number of subtasks executed in each handler was minimized.

The starting moment of each subtask is calculated starting from the subtask with the highest frequency to the one with the lowest frequency. For each subtask, all the possible starting moments are tried and the one for which the maximum load of an interrupt is minimal is assigned to the subtasks. It is important to mention that the starting moments are tried over a number K of possibilities where K is the least common

Table 5.4: Controller task durations.

Task	**Duration** (μsec)
Close Valves	0.9475
Cylinder level control	1.3325
Pressure control	5.0800
Read sensors (all at once)	494.4000
Read sensors. Subtask 1 of 4	124.2000
Read sensors. Subtask 2 of 4	123.0000
Read sensors. Subtask 3 of 4	124.2000
Read sensors. Subtask 4 of 4	124.2000
Read sensors. Subtask 1 of 8	61.6000
Read sensors. Subtask 2 of 8	61.4000
Read sensors. Subtask 3 of 8	61.6000
Read sensors. Subtask 4 of 8	61.4000
Read sensors. Subtask 5 of 8	61.6000
Read sensors. Subtask 6 of 8	62.6000
Read sensors. Subtask 7 of 8	61.6000
Read sensors. Subtask 8 of 8	63.6000
Read sensors. Subtask 1 of 12	41.8000
Read sensors. Subtask 2 of 12	41.8000
Read sensors. Subtask 3 of 12	40.6000
Read sensors. Subtask 4 of 12	41.8000
Read sensors. Subtask 5 of 12	41.6000
Read sensors. Subtask 6 of 12	41.8000
Read sensors. Subtask 7 of 12	41.8000
Read sensors. Subtask 8 of 12	40.6000
Read sensors. Subtask 9 of 12	40.6000
Read sensors. Subtask 10 of 12	39.4000
Read sensors. Subtask 11 of 12	43.0000
Read sensors. Subtask 12 of 12	42.8000

multiple of the task frequencies. The subtasks will have the same start time relative to each other within every group of K interrupts. The load of an interrupt is calculated by summing up the execution times of the subtasks it has to run.

An example of scheduled and un–scheduled task execution is presented in Figure 5.13. The example presents the case of controlling one platform only, because the number of tasks for two platforms is too large to be included in the image.

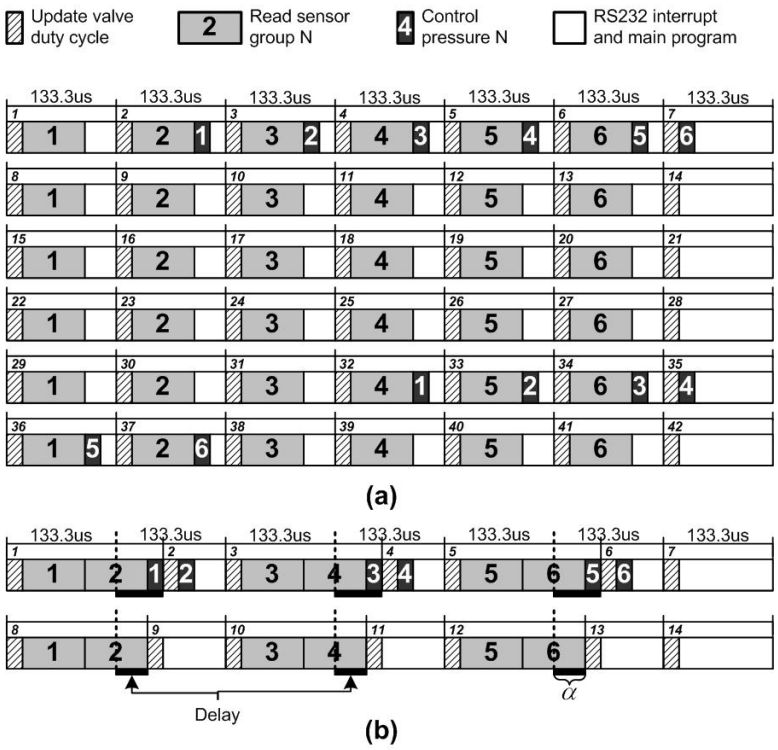


Figure 5.13: Controller task execution. (a) Scheduled; (b) Unscheduled.

5.5 Controller Software

Prior to the haptic control interface presented in chapter 3 the controller boxes handling the haptic glove or the original Rutgers Ankle robot were simpler and designed to serve only one type of device. The new controller box was designed to function with all the haptic devices developed by the lab. To accommodate this extension, the software controller had to be re–written. Although core control models such as the pressure or cylinder control were kept and later extended, the framework of the software had to be designed to satisfy the following requirements.

- *Flexibility.* The software had to support from the start three devices: the RMII–ND haptic gloves, the Rutgers Ankle platforms, and the Rutgers Mega–Ankle platforms. The implementation had to make the switch between devices transparent to the user.

- *Configurability.* The controller had to accept new devices of the types described

above without rebuilding it from the source code. In addition, changes to the device settings had to be also done without recompiling.

- *Stability.* The controller had to be virtually error free to avoid accidents. Given the large force output by the Rutgers Mega–Ankle robots, software failures were not admissible.

- *Comprehensive testing.* The testing of previous systems proved to be complex procedures involving several steps. To improve the reliability of the testing procedures, clear and consistent interfaces were necessary to move focus from the testing configuration to the actual hardware functioning.

- *Easy calibration.* Similar to the testing process, the calibration of the controller interface and the haptic devices is complex and requires full focus on the behavior of the hardware. Since the calibration is also performed as a periodical maintenance procedure, it was necessary to streamline it and make it as intuitive and flexible as possible.

- *Adaptation to hardware failures.* The Rutgers Mobility Simulator is a prototype system and hardware failures are inherent. The controller software had to provide the possibility of quick fixes to solve the problems such as a broken sensor or a faulty solenoid valve.

- *RS232 communication protocol.* A structured RS232 communication protocol is necessary for the support of multiple devices connected to the controller interface.

The controller software was implemented as a command line shell supporting commands for manipulating or controlling the hardware. An API was designed and implemented to model all the hardware entities with software structures for a consistent approach to hardware manipulation. A comprehensive set of controller commands was implemented to provide support for easy hardware testing, controller gain tuning, and various simulation modes for the mobility simulator as well as for the system in sitting. Details on the controller hardware API and the command set are presented in Appendix A

Chapter 6

Task level control

The system simulates walking by moving the platforms back and forth similar to stepping on a treadmill. Each robot either follows the user's swinging foot or slides the supporting foot backward. The user starts from a position with both feet touching the virtual ground, where both RMA robots support the weight of the user. Upon lifting one foot, the corresponding platform switches the control mode from position to force control. It then follows the user's foot compensating for its own weight. When the swinging foot touches the simulated ground (in front of the supporting foot), the corresponding platform switches back to position control. When the back foot is lifted to take the next step, the front foot robot slides backwards toward the starting position. The simulation tasks are split between the servo control interface and the PC. The control interface handles the changing of the control mode, sliding the front foot backwards, and coordinating the simultaneous motions of both platforms. The PC performs collision detection and notifies the RMA control interface when the foot touches the ground, so that the RMA robots switch the functioning mode from free–motion too support. The PC sends along information about the surface properties and the haptic effects to be applied.

6.1 Controller Interface Algorithm

The algorithm controlling the platforms is defined by a state transition diagram. The nine possible states (Table 6.1) describe the operation modes of each platform (see Figure 6.1). The transition between two states is either implicit when the current mode ends, or is triggered by a command from the PC or by the actions of the other platform.

At system startup, both platforms are in NULL (N) state. From this state, the platforms switch to PARKING (G) mode, which moves the platforms to the "home" position located in the middle on the workspace. Upon reaching the parking position,

Table 6.1: Walking states.

Acronym	State name
N	NULL
G	Parking
P	Parked
S	Standby
R	Release
F	Free
L	Locking
T	Translation
H	Hold

the platform's state is changed to PARKED (P). This state is used while the VR simulation on the host PC is not running. Any of the remaining states switch to PARKING at the end of the simulation. These transitions are not displayed in Figure 6.1 to improve its readability.

The transition from PARKED to STANDBY (S) is triggered by the STANDBY command sent by the simulation at startup. This state corresponds to the supporting foot during walking. The controller interface reads the forces applied by the user. If the user pulls up with more than 10 N, it releases the foot by switching to the RELEASE (R) state. The 10 N force is the threshold force that the foot has to apply to signal lifting. It has been determined empirically based on the following two criteria:

1. The foot–lift threshold force should be small enough not to cause discomfort;

2. The foot–lift threshold force should be sufficiently large so that accidental foot motions during the support phase will not cause a functioning mode change.

The RELEASE state is responsible for gradually changing the control mode over a 200–millisecond interval from holding position to free motion. The 200–millisecond delay is necessary to avoid control instabilities resulting from sudden gain changes. This applies to changes in gains from positive to negative and from negative to positive. Several transition functions have been tested for the gain change over the 200 ms interval. Linear and sinusoidal transformations caused very quick state changes between LOCKING and RELEASING due to the foot being pushed to forcefully upward by

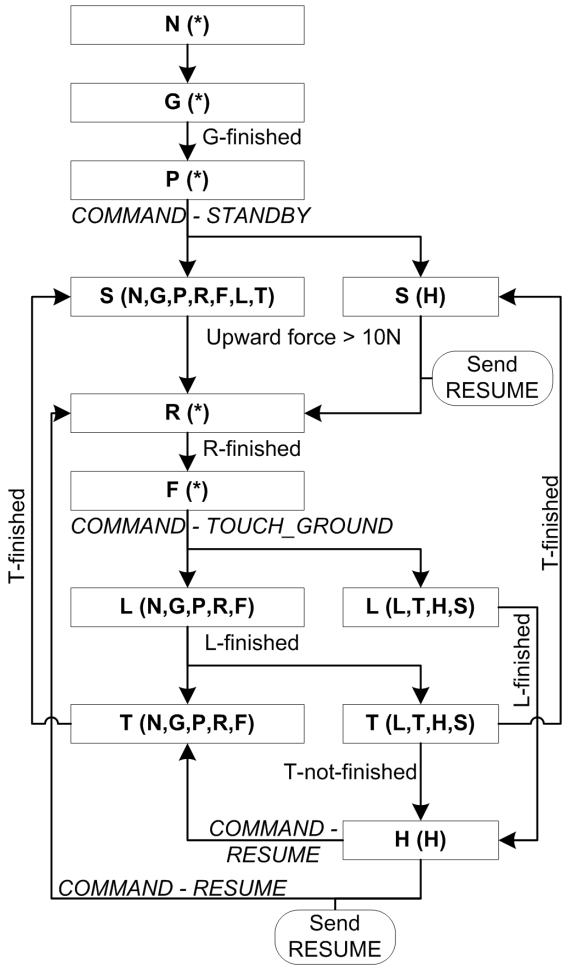


Figure 6.1: The walking state transition diagram [11].

the sudden gain change. Once the foot was too far up, the platforms would pull it down causing forces larger the 10 N necessary to trigger the RELEASE state. A double nested sine transformation was used to solve the problem. This curve is very smooth at the ends of the interval where the change is critical.

When the control mode switching is finished, the platform enters the FREE (F) state and follows the patient's foot motion compensating for its own weight. When the simulation sends the TOUCH_GROUND command the platform changes its state to LOCK (L). As a safety feature, if the user's foot is lowered in the bottom half of the workspace and the weight is about to be supported by the cylinders' armature instead of pressured air, the HCI controller automatically switches to LOCK state and notifies the VR simulation about it.

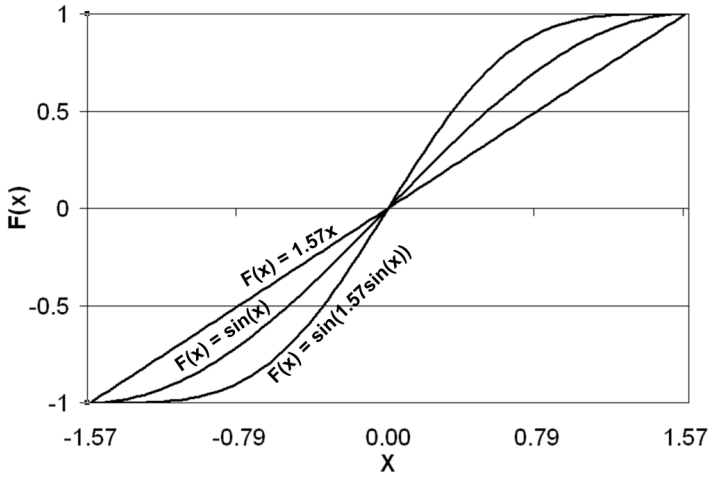


Figure 6.2: Gain transition functions.

At the end of the LOCK state, the platform can either start sliding backwards in the TRANSLATION (T) state, or hold position in HOLD (H) state. The TRANSLATION state moves the front foot backwards to imitate the functioning of a treadmill. However, if the other foot is touching the virtual ground, the platform must hold the position without generating motion. This is where the HOLD state is necessary. It is used to correlate the motions of the two platforms. In general, when a foot touches the ground while the other foot is still sliding backwards, both platforms switch to HOLD. HOLD is identical to STANDBY, except that it reacts to the actions of the other platform. At the end of the RELEASE state, if the other platform is on HOLD, the current platform sends it a RESUME command to continue with its previous state.

During evaluation, the algorithm presented above raised two problems. The first problem occurred when the user tried to lift the foot before the TRANSLATE state was finished. In the initial implementation, the TRANSLATE state was ignoring the upward pull of the user's foot, and was taking the support foot all the way to the back, hence insuring that the entire workspace length would be available for the next step. This behavior had to be disabled, so that an upward force larger than 10 N would cause the state to change from TRANSLATE to RELEASE. Although this change eliminated the usage confusion, it allowed the user to lift the supporting foot before it reached the back of the workspace hence limiting the space available for the next step.

The second problem was related to the HOLD state. If the user accidentally touched the ground, both platforms would stop moving. This can happen due to the small workspace of the robots. Once both platforms are stopped, the users generally tend to lift the other foot, not the one that accidentally touched the ground. Since both feet are somewhere in the middle of the workspace, the length of the step to be taken is limited. To solve this problem we had to allow the platforms to slide backward regardless of the fact that the other one touched the ground, hence canceling the effect of the HOLD state as far as simulating real life behavior goes. However, the HOLD state is still necessary to detect the situation described above and to ignore the motion generated by the backward motion of the platform.

6.2 Workstation Simulation Algorithm

The platform controller sends to the graphical workstation the position, orientation and state of the two RMA robots. The simulation has to transform this information into walking through the virtual environment.

6.2.1 Modeling Virtual Steps

The position of the RMA robots changes continuously during an exercise. However, the position changes can be considered for generating movement in the virtual world only for a subset of the functioning states presented above. For instance, the inevitable change in position of a platform in STANDBY mode due to the patient shifting his weight from one foot to the other should be ignored by the VR simulation.

At every iteration (simulation loop), the PC has to compute (based on the current and previous data) the motion vector of each foot. There are three possible situations: do not move either foot, move both feet independent of each other, or move one foot by the sum of that foot's change in position and the absolute value of the backward translation of the other. Figure 6.3 presents the action taken by the simulation based on the states of the two platforms.

The first case applies when both feet are on the ground, which is equivalent with

<table>
<tr><th></th><th>Right:N</th><th>Right:G</th><th>Right:P</th><th>Right:S</th><th>Right:H</th><th>Right:L</th><th>Right:T</th><th>Right:R</th><th>Right:F</th></tr>
<tr><td>Left:N</td><td colspan="7" rowspan="6">Left: no motion; Right: no motion</td><td colspan="2" rowspan="6">Move Right platform adding the absolute value of the Left platform's backward motion</td></tr>
<tr><td>Left:G</td></tr>
<tr><td>Left:P</td></tr>
<tr><td>Left:S</td></tr>
<tr><td>Left:H</td></tr>
<tr><td>Left:L</td></tr>
<tr><td>Left:T</td><td colspan="7" rowspan="2">Move Left platform adding the absolute value of the Right platform's backward motion</td><td colspan="2" rowspan="2">Move feet independently</td></tr>
<tr><td>Left:R</td></tr>
<tr><td>Left:F</td><td colspan="7">Move Left platform adding the absolute value of the Right platform's backward motion</td><td colspan="2">Move feet independently</td></tr>
</table>

Figure 6.3: Transforming platform motion into virtual walking.

having both feet in one of the G, P, S, L, T or H states. Although the control interface switches the states to HOLD when both feet touch the ground, it is possible for the simulation to receive mismatched pairs of states from the control interface because the communication is not synchronous with the control algorithm.

The independent movement of each foot corresponds to the situation when both feet are in the air. Since it is not possible to jump using the simulator, this case can happen if the patient lifts both feet and remains hanging in the unweighing system's harness. Since both feet are in the air, their motions will be applied to the virtual avatar, but the displacement gain will be reduced to the length of one step.

The last case is the most frequent one and occurs when one foot is in the swing phase (FREE state) and the other is on the ground. In such case, the supporting foot on the ground is sliding back during the TRANSLATE state. Its backward motion is added with changed sign to the forward motion of the swinging foot doubling the length of the virtual step.

To overcome the slow motion allowed by the simulator and increase the realism of the simulation, in the final phase of the algorithm the calculated foot motion vector is scaled by a gain factor that makes the virtual step distance closer to real life values. The scale factors are different for the three Cartesian directions to counteract the dimensions of the RMA platform workspace. The values of these gains were determined by comparing the RMA workspace with a normal footstep, and then empirically adjusted based on the user experience. The values of the scaling gains are shown in Table 6.2.

Table 6.2: Virtual step scale values.

	X	Y	Z
Linear	5.0	7.0	5.0
Angular	1.0	1.0	1.0

6.2.2 Modeling Direction Changes

By design, the Mobility Simulator does not allow the user to rotate about his body longitudinal axis (due to the un–weighting harness the user wears). This makes it impossible to directly measure a user's intention to turn. The same problem was faced by Iwata [56] in the development of the GaitMaster. Their solution relied on the orientation of the user's feet. The new direction of motion was calculated as the bisector of the angle formed by the two feet. This solution yields realistic results in general but it may be unstable when applied to patients post–stroke, who do not exhibit normal gait. Stroke causes partial paralysis of one of the patient's legs. Thus, the control a patient can exert over the affected foot is limited. This can cause the foot to be used with inconsistent yaw angles, which would throw off the bisector and hence the direction calculation.

A more stable alternative is to always orient the viewpoint along the desired path. This approach requires the value of the path tangent. Since the path is a polygonal mesh, the calculation of the tangent is complex. Another possibility is orienting the viewpoint in the direction of motion. The walking direction is initially set to a known value that points the user along the path. The new walking direction is calculated as the line uniting the viewpoint position last recorded when both feet were on the ground and the current position of the viewpoint (Figure 6.4).

The viewpoint position is updated in real time while the swinging foot moves to take a new step. A filter is applied to the new values to remove tremors in the rendered scene. This solution works satisfactorily in most cases, but fails when the user takes a side step.

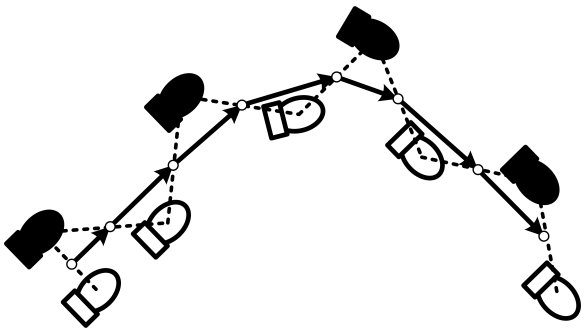


Figure 6.4: Walking direction calculation.

6.3 Virtual Ground Haptic Modeling

The starting point of this work has been the necessity to develop challenging and controlled simulations of ground surfaces similar to what patients with gait disorders would have to negotiate on a daily basis. Such surfaces require support for modeling a patch of ice or snow on the sidewalk, a gravel–covered or muddy park path, or a puddle of water on the road. These properties could affect either the whole surface or just some limited areas.

6.3.1 Haptic Surface Materials

The ground surface is specified as a polygonal mesh that matches the shape of the visual 3D geometries in the virtual environment. The physical properties of the surface are specified using haptic materials, which are applied in layers that can be either distinct (like ice above water) or mixed (like mud on the bottom of a lake). The polygonal surface is unbreakable and the haptic materials can be placed on top of it in layers. This approach insures that the foot stepping down will always be supported if it penetrates through all the materials stacked on top of the bottom surface.

A haptic material is defined as a collection of numerical parameters describing the physical properties of the surface. Given the requirements of an interactive virtual reality simulation, the haptic modeling computations need to be efficient. Thus, the model is not intended to be physically accurate but rather a good approximation of reality. Hence, the set of variables used is limited to *stiffness*, *damping*, *friction*, *haptic texture* and *breaking coefficient*.

The stiffness and damping coefficients are used for defining the material based on Hooke's law. Stepping on elastic materials is not something very common, however, the stiffness coefficient can be used to simulate Archimedes' law if considering the foot section constant when stepping into a liquid. The damping coefficient can be used to simulate the slow sinking sensation of walking on a thick carpet. A low friction coefficient can be used for simulating ice. If the foot applies horizontal forces to the material larger than the friction coefficient, the contact will break.

The haptic texture is defined as a vibration with a given amplitude and frequency. The breaking coefficient multiplied with the thickness of the material specifies the maximum force that the material can support. A haptic material is not rendered if the applied force is larger than its breaking force. The breaking coefficient makes it easy to simulate stepping on a thin layer of ice on top of a puddle of water.

6.3.2 Haptic Material Blending

It is a normal phenomenon for water in a puddle to mix with the earth around it and yield mud. This type of behavior can be simulated using the haptic material with a negligible cost to performance. Haptic materials can be mixed using an approach similar to the blending of graphical textures. Given a set of materials and their respective normalized ratios, the resulting new material is calculated by assigning to each of its parameters the weighted average of the mixed materials' corresponding parameters.

This simple blending procedure can be successfully used to address the case of two overlapping materials such as water and earth. Although physically inaccurate, the resulting material (mud) will have a lower damping and friction coefficients, as well as a smoother texture than earth (lower amplitude and frequency).

Besides mixing overlapping materials, blending can be used to create new materials in a more intuitive way. Starting with a predefined set of basic materials that can conceivably be stepped onto when walking on a real road, one can reduce or augment their properties by mixing them. For instance, the viscosity of mud (defined through its damping coefficient) can be reduced by mixing it with water. The mixing ratios of a blended material can also be negative, which will have the effect of extracting material

from the mix. In other words, taking the earth out of the mud material will leave water.

6.3.3 Dynamic Haptic Materials

The haptic material concept can be further extended by allowing the properties to be changed based on an external variable such as force, time, or penetration distance. A dynamic material is defined by an initial material, a final material, an interval of possible values for the external variable and a transform function that converts the value of the external variable into a fraction of final material to be mixed with the initial material. The parameters of the dynamic material for a certain value x of the external variable is calculated as

$$param^{A}_{dyn} = F(x)param^{A}_{fin} + (1 - F(x))param^{A}_{ini} \tag{6.1}$$

The dynamic materials can be used to simulate nonlinear aspects of the surface properties. For instance, when stepping on snow the forces applied to the foot are initially very small. As the foot goes deeper and the snow is compressed, the forces increase linearly until they reach saturation and are strong enough to support the foot weight. With dynamic materials, snow can be simulated from an initial material with low stiffness and damping coefficients and a final material with very high values for the same parameters using the ratio of the penetration depth versus the material depth as the external variable. The transform function would have a profile similar to the one in Figure 6.5.

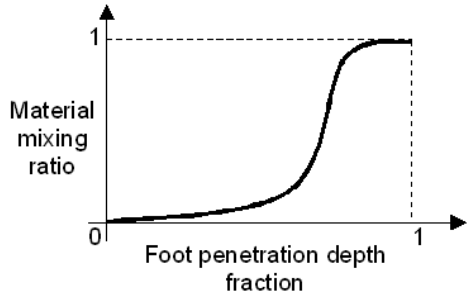
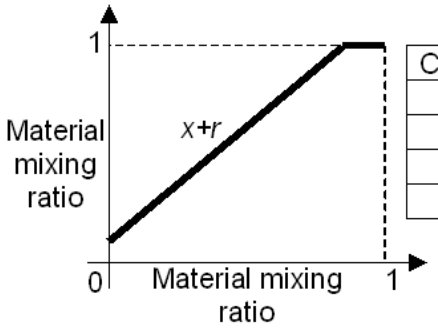


Figure 6.5: Dynamic material transform function for modeling snow.

Dynamic materials can also be used to simulate processes such as water evaporation

or the hardening of concrete. The evaporation of water from a surface can be simulated with water as the initial material and a dry surface as the final material. The external variable can be the time and the transform function will be defined over the desired period of time over which the evaporation should take place. More realistically, the transform can specify a rate of change rather than the ratio at a certain moment, by using as external variable the material–mixing ratio itself and as the transform the function in Figure 6.6. Using this approach, the drying of a wet surface will take correspondingly longer if water is spilled on it.



Cycle	Ext Var	Trans
0	0	r
1	r	2*r
…		
N	(n-1)*r	n*r

Figure 6.6: Using the transform function to specify the material rate of change.

6.3.4 Applying Haptic Materials to the Virtual Ground

The haptic materials are applied on the virtual ground surface with layered polygonal patches parallel with the horizontal plane. A haptic patch is defined by a polygon, a haptic material, bottom limit and a top limit. The material of a patch is applied on all the points on or above the surface whose projections on the horizontal plane fall inside the polygon and their vertical position is within the bottom and top limits of the patch.

Two types of patches have been defined to support modeling of ice, mud or water puddles. Lining patches (Figure 6.7) are used to model surface properties that are uniformly laid on the entire surface. Examples of such materials are ice and snow. The bottom and top limits of a lining patch are defined along the normal to each surface polygon in the positive direction. The filling patches (Figure 6.7) are necessary to simulate liquid materials contained in a pocket of the surface. For this type of patches, the bottom and top limits are measured along the vertical axis of the world, and the

zero limit matches the lowest surface point that projects onto the patch.

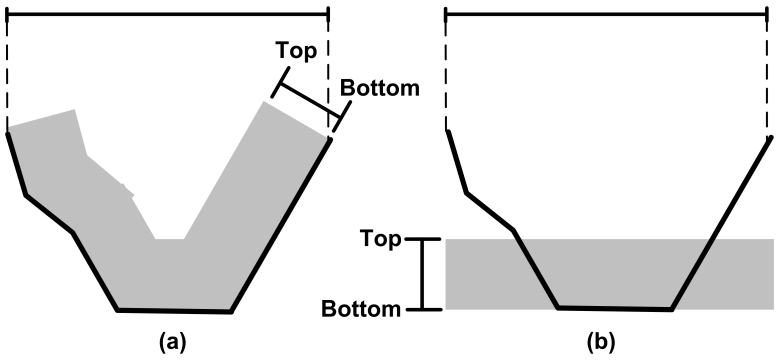


Figure 6.7: Section view of haptic patch rendering: (a) lining patch; (b) filling patch.

A third type of patch is the matrix patch (Figure 6.8). It has been designed to specify fine–grain materials that would be too costly (in cycles) to specify with the contiguous patches defined above. The matrix patch is very similar to a graphical texture in the fact that it is given as a grid of points within the bounding polygonal patch. Each point of the grid is characterized by a separate material and separate depth limits. Although more difficult to specify, this patch type allows for simulation of circumstances such as walking through a field of rocks covered in snow or on a very uneven mountain trail covered in leaves, without having to model the rocks as part of the virtual ground surface.

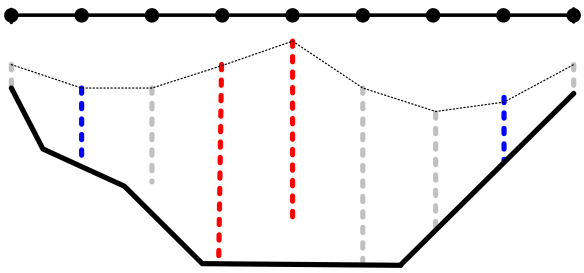


Figure 6.8: Section view of a matrix haptic patch. Each grid point is assigned separate material and depth limits.

The material properties for a point within a matrix patch are calculated as a gradient of the neighboring values. Because the matrix points do not have the same vertical boundaries, it is possible that when calculating the gradient, some neighboring points will not have a material specification for the desired depth, in which case a default value

will have to be used.

6.4 Haptic Rendering for Walking Over Virtual Terrain

The Mobility Simulator transforms the input position, force and functioning mode of each RMA platform into haptic feedback to the feet and visual update of the virtual scene using the virtual surface specifications. The input to the simulator consists of two sets of thirteen values, and the virtual ground data. Each set of values contains a flag indicating the platform functioning mode, three position coordinates, three angles, three forces and three torques, all represented in the frame of reference of the corresponding RMA platform's fixed base. The virtual surface specifications are pre–stored on the graphics workstation. The haptic output data consists of two sets of values specifying the functioning mode to be used by each platform, the 6DOF forces to be applied and the vibrations to be applied to the user's foot.

The processing necessary to calculate the graphics and haptic feedback can be divided into several stages that are executed at every simulation cycle (Figure 6.9). Only the swinging foot (free motion) is considered for the entire rendering process. The fixed foot (load compensation mode) is addressed only in the last stage of the process.

The process starts by reading the feet positions and functioning modes from the control interface. The functioning mode value is used to decide whether a foot should be moved or not. A foot in load compensation mode is kept fixed although the platform slides it backward.

The next stage requires the calculation of the change in real foot position to be added to the virtual feet. Because the simulator's workspace cannot cover the entire range of motion of the legs, it was necessary to scale the change in each foot's position to increase the virtual walking velocity so that the simulation felt real. The scale is also applied to the vertical displacement making it possible to negotiate realistic virtual obstacles that are visually larger than the platform work envelope.

The next phase updates the positions of the virtual feet with the calculated change. The changes are applied in a frame of reference aligned with the virtual avatar's walking

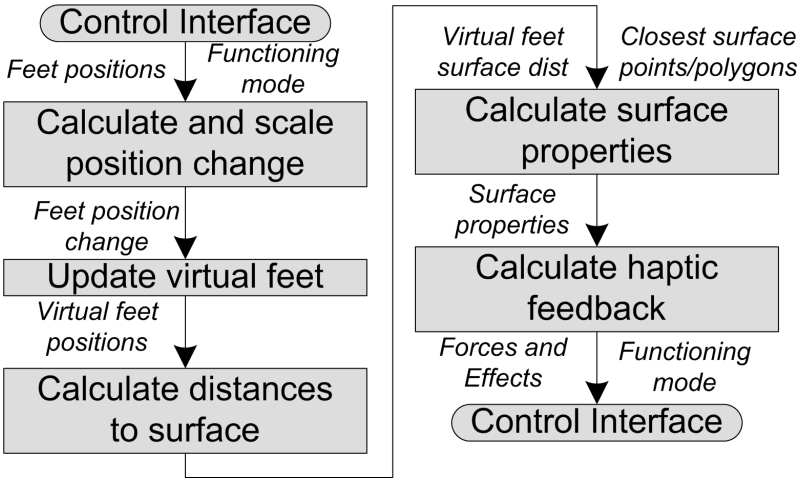


Figure 6.9: Haptic rendering stages.

direction calculated in the previous cycle.

After the feet have been mapped into the virtual world, the viewpoint has to be moved accordingly. Positioning the virtual camera above the center of the segment defined by the two feet yields good results although it is not what really happens with the center of gravity of a walking person.

The next stage is calculating the distance between the foot and the surface. This stage will also find the closest surface points to the foot and their corresponding polygons. Similar to the real case, certain surface properties are manifested above the surface (e.g. mud or snow) hence the distance to the surface is relevant to the haptic feedback even if there is no collision with the underlying ground. Based on the results of the previous stage the simulator can find the surface properties around the closest points on the surface.

The last stage of the process is the calculation of forces and haptic effects to be applied to the user's foot, based on the surface properties and the depth of the collision.

6.4.1 Virtual Foot Modeling

The interaction between the virtual foot and the virtual ground surface is based on the haptic mesh concept developed by Popescu [99] as an extension to Ho's simpler haptic point concept [47]. The virtual foot implemented for the Mobility Simulator is

modeled as a mesh of points positioned on the shoe sole. From a haptic point of view, the RMA platform can only render forces in one point. The use of a mesh of points to calculate the interaction of the foot with the surface is necessary for realistic surface contact calculation, and to determine the resultant force.

The number of points in the mesh should be minimized because it is directly proportional to the amount of collision detection calculations, and it increases factorially the number of contact stability calculations. The minimum number of mesh points has been empirically chosen to be five. One point is positioned in the center of the mesh, while the rest are positioned on a rectangle around it 6.10. The dimensions of the rectangle match the shape of the end–effector foot attachment plate to which the user's foot is secured.

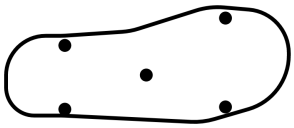


Figure 6.10: Foot haptic mesh.

6.4.2 Distance to Ground Surface Calculation

The virtual ground surface is the second input to the simulator besides the RMA position/force sensor readings. Since the haptic interaction occurs at foot level, the walking surface has to include only the ground and the obstacles. The visual geometries with higher elevations than the maximum vertical range of the system do not need to be added to the haptic surface.

The distance between each foot mesh point (surface distance - SD) and the ground surface is calculated as the minimum distance between that point and each surface polygon. The vertical distance to the surface (VSD) is also computed at this stage by finding the surface polygon that intersects a vertical ray through the haptic interaction point. The signs of SD or VSD are calculated based on the polygon normal. It is important that the surface polygons be specified consistently so that their normals point toward the walking side of the surface. For convex surfaces, the normals can be substituted with the vertical direction. Positive distances indicate that the haptic

point is above the surface while negative ones indicate collision with the surface. The calculations of SD and VSD also find the closest surface point (CSP) and closest vertical point (CVP).

The haptic patches covering each haptic point in the foot mesh are searched for in this stage using their corresponding CSP and CVP. The SD and VSD are used to sort out the patches that cover the haptic point but do not enclose it. If the haptic point is under the surface, the negative distances are replaced with zero so that the search will find the haptic material applied directly on the surface if it exists.

The SD is used with lining patches since their depth limits are aligned with the surface normals. The VSD is used with filling and matrix patches because their depth limits are aligned with the worlds vertical direction. For each of resulting patches, the haptic point penetration depth is calculated along the haptic point's direction of motion using the SD or the VSD in accordance with the patch type.

6.4.3 Haptic Feedback

The last stage of the haptic rendering process has to translate the individual interactions of the haptic mesh points with the virtual ground into a resultant force feedback to be rendered on the haptic platforms. Each haptic point is characterized by the following parameters:

- Distance from the haptic surface (SD);
- Collision status;
- List of touching haptic material patches;
- Penetration vector for each haptic patch touched.

Based on the above data, this stage calculates the force at each mesh point based on its contact with the surface and the material patches, decides whether the foot is making a firm contact with the surface and combines all the point forces into a resultant 6DOF force/torque to be rendered to the user's foot.

Haptic Contact Point Force

The forces at the haptic contact points are calculated in two steps. First, for each point, the force applied by the interaction with the material or the surface is calculated using the formula

$$F(k) = - \sum_{minAffectedMaterials} \begin{pmatrix} Stiffness(m)PntrVec(k,m)PntrDepth(k,m) \\ + \\ Damping(m)Velocity(k) \end{pmatrix} \tag{6.2}$$

where PntrVector(k,m) is the unit penetration vector of point k inside material m. PntrDepth(k,m) is the penetration depth of point k in material m. Velocity(k) is the velocity of point k.

FN(k), the component of F(k) normal to the material is then compared with the breaking force FBRK(m) of each material m calculated as the product between the breaking coefficient and the patch depth. The depth is calculated from the haptic patch vertical limits. If FN(k) is larger than FBrk(k), then the haptic patch is considered broken and hence removed from the list of patches affecting the mesh point k. Finally, F(k) is recalculated from the new reduced list of materials.

The friction coefficient FrcCoef(k) affecting each point k is calculated by averaging the friction parameters of materials in the list. Similarly, Tex(k) the haptic texture of point k is calculated by averaging the textures defined by the materials in the list attached to point k. The texture of the surface and its friction can be felt also by the supporting foot although it is not moving; hence, last operations are executed for both feet.

Ground Contact Evaluation

When a foot touches the haptic surface on a solid region that can support its weight, the swinging phase of the foot is over and the support phase is about to begin. The switch between these two phases is tightly connected to the functioning of the servo–controller, which has to be notified to start sliding the support foot backward.

The foot/surface interaction is evaluated by comparing vertical components of the forces applied by the materials to the haptic points with the vertical components of the forces applied by the user's foot to the simulator. The former are the F(k) values calculated in the previous step, while the latter (Fext(k)) are calculated by transforming the forces measured directly by the force sensor mounted on top of the RMA platforms. A point k of the foot haptic mesh is "supported" if the vertical component of F(k) + Fext(k) is pointing upward.

The three contact possibilities that can be differentiated based on the supported status of the haptic mesh points are presented in Table 6.3 and Figure 6.11.

Table 6.3: Contact status based on the haptic mesh point support.

Contact status	Description
No contact	None of the mesh points are supported
Stable	Minimum three non–collinear points are supported
Unstable	Remaining cases

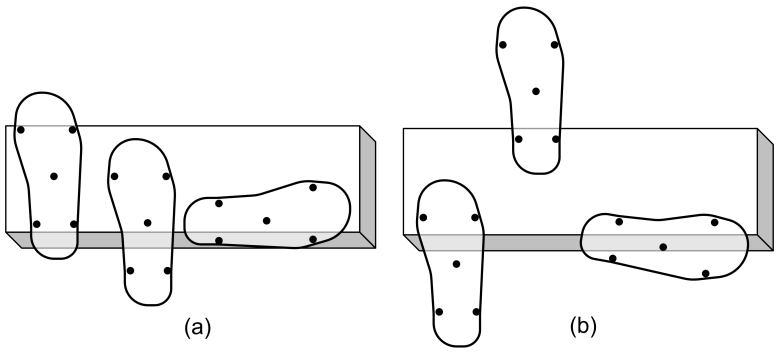


Figure 6.11: Foot/surface contact types: (a) stable, (b) unstable.

Resultant Feedback Calculation

In the last step, the forces and torque at the foot are calculated from the haptic point forces by translating them to the center of the mesh. The resultant force is transmitted to the control interface to be applied to the swinging foot. If the swinging foot made a stable contact with the surface, the controller is notified to switch the functioning mode. The friction and surface textures are sent to the controller for both feet, regardless of their state.

6.4.4 Low–level Haptic Effects

The mobility simulator system is designed to execute most of the haptic calculation on the workstation, and render the results on the Stewart platforms using a reduced set of basic level effects:

- Apply a 6DOF force/torque;
- Change in position (positional jolt);
- Vibrations.

These low level haptic effects are extensions of the haptic effects developed for a similar system using smaller Stewart platform robots, and designed for stroke rehabilitation in sitting [37].

The simulation can request the controller to apply a certain force during both free or foot support functioning modes. If the request is made while in free motion mode, the given forces are scaled using positive values of the free mode gains and then applied. The scaling is necessary to counteract the same damping behavior mentioned above.

The changes in position are used as an alternative force requests during foot support mode. For instance, to simulate slipping on the ice, a lateral displacement is used instead of applying a lateral force. This is preferred because a known displacement is more controllable and can be adjusted to a comfortable level easier than applying a force, which will have a different effect from one user to another, mainly due to differences in their weight.

The vibrations are used to simulate surface textures. The frequency and amplitude of the vibrations are calculated on the graphical workstation and sent to the controller. The vibrations are rendered only as changes in orientation around the Y–axis (back–front direction) because it interferes the least with the rest of the foot measurements necessary to calculate the direction of motion, or intersection with the virtual surface.

Chapter 7

VR Applications for Rehabilitation

The Rutgers Mobility Simulator has been designed as a rehabilitation device for patients post–stroke. Virtual reality has been used in numerous studies to increase the motivation of the patient by making the rehabilitation therapy less tedious [103, 24]. VR can be used to simulate real life situations and hence train the patient for specific tasks that one might encounter in activities of daily living (ADLs). The virtual exercises can be specifically designed to guide the patient through the practice by setting goals achievable through the motions necessary to recover the injured limb function. In order to be effective, the VR rehabilitation exercises have to satisfy the following requirements:

1. Require the patient to do the therapeutic exercises intensively and for long durations [67];

2. Be engaging and distract the patient's attention from the monotonous and sometimes frustrating rehabilitation procedures [98];

3. Keep a high win/lose proportion to keep the patient's attitude and motivation positive [12];

4. Provide the patients with performance feedback to keep them informed about their progress and hence getting them more involved in the process [12].

The most challenging part of designing a VR rehabilitation game is satisfying the therapy requirements and making the game interesting and engaging. The rehabilitation therapy, and especially the post–stroke therapy, relies on repeatedly executing the same motions a large number of times. According to the National Institute and Neurological Disorders and Stroke [86], the repetitive use of the impaired limbs helps reduce disabilities by encouraging the creation of new neurological pathways ("brain

plasticity") that can handle the exercised motions. Kwakkel [67] showed that there exists a small but statistically significant effect of the therapy intensity on the efficacy of the rehabilitation.

Two virtual reality exercises were created for the Mobility Simulator. The exercises were designed by Burdea, Deutsch, Boian, and Kourtev. The implementation of the exercises was done by Boian and Kourtev [15].

One of the exercises was the simulation of a walk on a park alley (see Figure 7.1). The shape of the alley can be customized to train turning. In addition, obstacles can be placed on the path and the ground surface conditions can be changed.

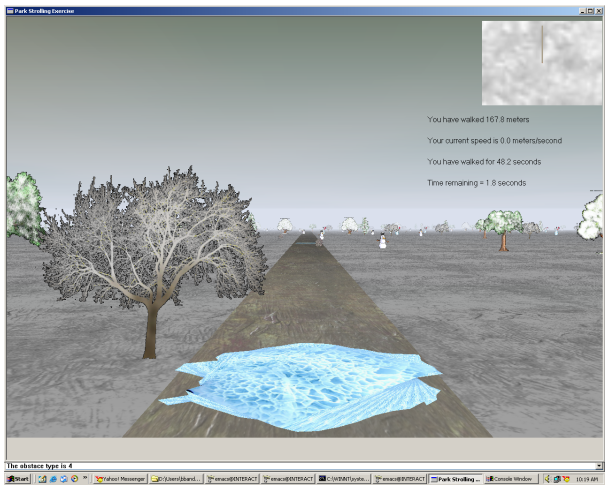


Figure 7.1: Park walk exercise for gait training.

While the park walk simulation targets directly gait training, the second simulation is designed to train the patient to negotiate with a very common ADL, crossing the street in time, before the pedestrian light turns red.

7.1 Street Crossing Exercise

Crossing a virtual street is the first VE exercise developed for the mobility simulator described here. The patient's task is to cross the street while the pedestrian light is green. This VE was designed to combine important requirements of community ambulation, which include gait speed and the negotiation of uneven surfaces (curbs).

The parameters required for walking in complex virtual environments were modeled

on the framework presented by Patla [93, 94] and the task taxonomy presented by Gentile [40]. For example, perturbations in this environment can be added to increase the attentional demands of the patient, such as introducing vehicles that encroach on the crosswalk, blow horns, etc [13]. Ambient conditions can be altered by changing the time of day and the seasons. Traffic level is increased by adding vehicles or pedestrians. Task complexity is adjusted to patient's abilities to maintain engagement and reduce frustration.

The application window (Figure 7.2) is divided into three areas: the 3D virtual environment, the control panel and the 2D therapist feedback panel. The virtual environment occupies the large central portion of the screen and consists of a dynamic street model containing two sidewalks, driving lanes, stoplights, pedestrian lights and intelligent vehicles. The difficulty of the exercise can be adjusted through a number of configuration parameters that define it. These parameters have been chosen to allow setting the distance and speed at which the patient has to walk as well as changing the virtual world's appearance by season, time of day, or type of community. All the parameters are set by the therapist prior to the exercise session and a subset of them can be modified while the patient is exercising as well.

7.1.1 Configuration Parameters of the Simulation

The main parameter of the simulation, based on which the success of each tasks is decided, is the patient's average speed required to cross the street. The speed parameter is a percentage between the normal and maximum walking speed the patient can achieve on the simulator. A baseline of these values is obtained daily at the beginning of the therapy session.

The width of the street is defined prior to the exercise start by specifying the number of lanes the street should have. While this affects the distance the patient has to walk to reach the other side of the street, it also triggers a change in the street's appearance. One or two lane streets cause the building textures populating the back of the scene to show houses resembling a suburban setting (Figure 7.3). Streets with more than two lanes make the environment have an urban look by displaying city buildings (Figure 7.2). This

Figure 7.2: Street crossing exercise. Normal road surface in an urban environment.

parameter cannot be changed at run–time. From the speed parameter and the width of the street, the simulation automatically calculates the duration of the pedestrian green light. A flashing red light signals to patient that the time to cross the street is running out. In case pedestrians are caught in the street by the red pedestrian light, the cars in the same lane will wait for them to cross while the vehicles in the other lanes will continue driving until pedestrians step into their lane.

The behavior of the vehicles can be set as a percentage between "well behaved and aggressive". Aggressive vehicles will accelerate faster, drive at higher speeds and brake later. They will also behave impatiently when forced to stop; when stopped at the red light, they will slowly inch on the pedestrian crossing. Vehicles will sounds their horns when pedestrians are caught in the middle of the street, during a pedestrian red light.

The sidewalk curb height and shape are also configurable. In real settings, the shape of the sidewalk edge can be either a curb or it can be sloped to accommodate wheelchairs. The same situations are supported by the simulation both visually and

Figure 7.3: Street crossing exercise. Icy patch in suburban road setting.

with haptic feedback. By adjusting these two parameters, the therapist controls the difficulty of the transition from the sidewalk to the road surface. A higher sidewalk can mean either a higher curb or a steeper slope. The curb height is defined as a percentage of the maximum height that can be rendered by the Rutgers Mega–Ankle robots. This feature can be changed at run–time from its current value if the patient is unable to climb the opposite sidewalk, unlike real environments.

Changing the lighting of the virtual world makes elements of the street more or less obvious, thus adjusting the difficulty of the task. The parameter controlling this aspect is a percentage between a minimum and maximum lighting level. Changes in the lighting level also require changes in the building and sky textures. Because the images used as textures were created during a certain time of the day, they may not match the world's lighting level, by showing a sun lit house in a dark environment. To solve this problem, multiple copies of each texture have been created by manipulating the luminosity. The value of the lighting level causes the building and texture images to be switched to lighter or darker versions.

The surface condition parameter is used to define the haptic feedback applied to the patient's feet. Possible settings are ice, mud, gravel, or normal. Since the surface conditions are mostly determined by the weather, this parameter also triggers changes in the visual aspect of the scene. An icy surface will cause the scene to switch to a

winter look by applying different house and walking surface texture images (Figure 7.3). The gravel and normal surfaces will trigger a summer scene setup. The muddy surface will darken the scene as to simulate a rainy day (Figure 7.4).

The difficulty of surface conditions can be set on a scale from 0 to 100. The difficulty of a surface property is defined separately for each type of surface:

- Ice: higher difficulty level equivalent with lower friction (the ice is more slippery);
- Mud: higher difficulty level equivalent with a larger damping coefficient (the mud is more sticky);
- Gravel: higher difficulty level equivalent with larger amplitude vibrations when stepping on the surface.

Figure 7.4: Street crossing exercise. Muddy road in a nighttime urban environment.

Finally, the pedestrian crossing can be configured to be a zebra–type crossing or a pelican–type crossing (two lines bordering the cross walk strip). The zebra crossing provides a greater degree of visual contrast, which may make crossing easier.

7.1.2 Off–line Session Configuration

The number of parameters defining an exercise is large, and the duration of one exercise (one street crossing) is less than a minute. Anticipating that the amount of practice and repetition on this task may need to be considerably longer, the simulation can

run a series of street configurations set by the therapist prior to the rehabilitation session. Each configuration can be named and the therapist can specify the number of trials (i.e. street crossings) to be used. At the end of each trial, the patient is provided with knowledge of results about the success of crossing, and if necessary, the street is reconfigured to match the next settings. After that, the patient's viewpoint is repositioned on the starting side of the street and the next trial is ready to begin. Using this mechanism, the therapist can plan a complete intervention session with a sequence of simulations and then monitor the patient's performance and adjust the parameters as required.

7.1.3 Run–time Control

Certain configuration parameters can be adjusted while the patient is exercising through the GUI control panel at the bottom of the application window (shown in Figure 7.5). Percentage parameters such as patient speed level, ambient lighting level, vehicle behavior and curb height are changed using sliders. Buttons are provided for changing on/off parameters such as ignoring the patient's change in walking direction, or disabling the ground surface haptic effects. Such settings are useful in the early phases of training on the simulator, when the patient and the therapist may need to adjust several parameters to achieve the best frequency, duration and intensity of training.

Figure 7.5: Street crossing simulation control panel [13].

Four more buttons are provided to allow the therapist to change the flow of the session. The EXIT button ends the session, while the PAUSE/RESUME button can be used for giving the patient a resting break. The NEXT button is used to skip to the next preconfigured trial. This can be used to shorten the current therapy session if the patient shows signs of fatigue. The placement and appearance of the controls presented above has been optimized based on the study done by Whitworth and Deutsch [117, 34] on the visual interface of the ankle in sitting system (see Appendix B).

7.1.4 Performance Real–Time Feedback

The third area of the application window is a GUI 2D panel displaying session and patient information intended for the therapist (see Figure 7.6). The patient's instantaneous and average speeds are displayed in relation to the trial's target street crossing speed and the baselined normal and maximum speeds. The instantaneous speed is measured over the short time intervals between two sensor readings. The average speed is calculated dividing the distance walked by the patient by the time elapsed since the light turned green. The patient's measured speed is displayed as a scaled bar graph on which the three reference values (trial's target street crossing speed, baseline normal speed, and baseline maximum speed) are marked with thin vertical bars. The bar graphs switch color from red to green as the their value exceeds the corresponding set target. Adjacent to the speed bar graphs the number of steps taken by the patient is displayed during the current street crossing simulation.

The session progress versus the preconfigured setup is displayed by showing a list of configuration names followed by the number of trials completed and the number of trials that have to be executed for that particular configuration.

Figure 7.6: Street crossing simulation 2D feedback panel [13].

Chapter 8

Mobility Simulator Validation

The mobility simulator has been evaluated over three criteria: comparison with existing art, robot behavior during gait, and comparison between normal gait and gait on the simulator. To assess the system's value as a research project it was compared to the state of the art technology in gait rehabilitation. The behavior of the RMA robots was evaluated for three specific gait aspects in comparison with the real life situations. These aspects are: the behavior of the RMA robots when the virtual foot touches the virtual ground, the stability of the support foot during the backward translation phase, and the trajectory of the swinging foot. Finally, a full gait comparison was performed between walking on the simulator and normal walking.

8.1 Comparison with the Lokomat Orthosis

One of the most active research projects for developing a mobility training robotic device is the Lokomat® system developed at the AI Lab of the University of Zurich [28, 29]. The system targets the rehabilitation of paralyzed persons. Figure 8.1 presents the Lokomat® system and how it is used by the patient. The main components are a treadmill, an unweighing frame and an active orthosis. The patient is suspended in the unweighing frame and the orthosis is attached to the legs. The orthosis aids the patient to move the feet on the treadmill.

This system is very appropriate for paraplegic rehabilitation but its application to post stroke rehabilitation because it constrains the motion of the patient. The orthosis fully restrains the lower body of the patient reducing or canceling some gait degrees of freedom.

The Rutgers Mobility simulator is smaller than the Lokomat® system mainly because it does not use a treadmill. The unweighing frame makes the size of the two systems vertically comparable. The Rutgers system brings a series of advantages when

Figure 8.1: The Lokomat® gait orthosis. From the Automatic Control Laboratory, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland.

compared with the features of the Lokomat® System.

1. It provides individual foot 6DOF feedback that can be customized to simulate walking on various surfaces the patient is likely to encounter in ADLs.

2. It is less encumbering. The Rutgers simulator does not constrain the motions of the legs except at ankle level. While the flexing joint range is reduced by the workspace of the platforms, the rotational range of the joints is not limited. The torso is suspended in an unweighing vest similar to Lokomat®.

3. The Rutgers system is integrated with VR simulations that engage and motivate the patient, while training him or her for real life tasks.

4. The Rutgers system is integrated with a full framework for data collection and analysis and remote monitoring that allows the therapist control and evaluate the therapy without the need to be present in the same location.

8.2 Comparison with the Robomedica BWS System

The Robomedica Body Weight Support (BWS) system (Robomedica Inc., Irvine, CA) is a commercially available installation designed for gait rehabilitation of various patient populations (see Figure 8.2). The system consists of a treadmill installed under a pneumatic unweighing arm. The patient, secured in the unweighing vest walks on the treadmill while using the sidebars for balance. Two chairs are positioned by the treadmill, one on each side, lowered at a level with the patient's foot. Two therapists, sitting on these chairs assist the patient into moving the feet. A computer connected to the system monitors the patient's body weight unloading and adjusts it to reduce or increase the ground reaction force when stepping down.

Figure 8.2: The Robomedica BWS System. From Robomedica Inc.

The BWS system provides an ergonomic setup that makes the gait rehabilitation procedures more comfortable for both patient and therapist. However, it does not bring significant innovations to the classical procedures. The use of the BWS System is

comparable in time and personnel costs with the existing gait rehabilitation procedures.

In comparison, the Rutgers Mobility Simulator applies robotics and virtual environments to enriching the patient experience while saving the therapist the physical effort necessary to move the patient's feet. The independent 6DOF feedback to each foot provided by the Rutgers system supports a wide variety of walking surfaces. The Rutgers system also reduces the cost of the therapy by reducing the number of therapists in the room.

8.3 Behavior at Foot Contact with the Ground

When the virtual foot is touching the virtual surface, the RMA platforms are commanded to hold the position. It is important that the platforms are able to hold that position rigidly simulating the resistance of the ground. A less stable support would be perceived by the user as unnatural. The stabilization of the platforms under the user's load was the most difficult problem in the development of the system. The fact that the robots are used closed to the limit of the load range made the problem even more difficult. The change in platform position was measured during the user's step down. The recorded data are presented in Figures 8.3 and 8.4.

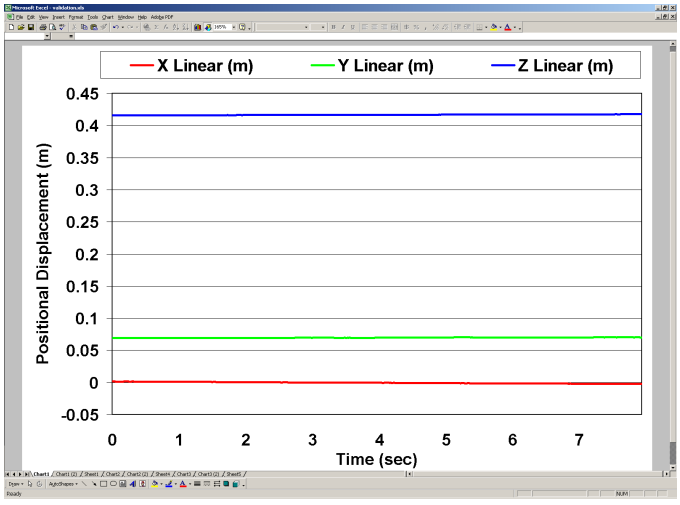


Figure 8.3: Changes in the position of the RMA robot when the user's steps down.

The vertical position (Z–axis) increases during the switch from free mode to support mode by approximately 2 mm. The change is small enough not to be sensed by the

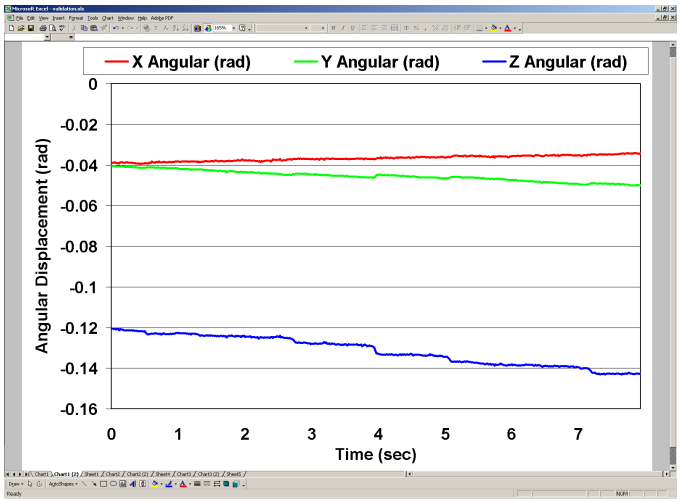


Figure 8.4: Changes in the orientation of the RMA robot when the user's steps down.

user. The cause of the change is the tuning of the controller to react promptly to disturbances. A change of approximately $0.03 radians = 1.7 deg$ is also visible around the Y–axis and Z–axis.

One disadvantage of the Mobility simulator is that the foot cannot be flexed as in real walking. During the touch down phase, the foot normally steps down with the heel and then the toes reach the ground. The RMA robots limit this motion because it usually takes place at the edge of the workspace where the orientational flexibility is minimal. Hence, the touch down phase of the gait is done with a flat foot.

8.4 Support Foot Stability during the Backward Translation Phase

Another interesting aspect of the simulator is the stability of the backward motion of the support foot. The change in position of one RMA robot has been measured and the results are presented in Figures 8.5 and 8.6. The major changes in position occur along the Y–axis and around the X–axis. The Y–axis position change is the actual translation of the foot and it is normal. It should be noted that the translation is smooth and without irregularities. The X–axis orientation corresponds to the pitch motion. The orientation of the foot is slightly upward at the beginning of the support phase and slightly downward towards the end, right before it gets ready to take another step.

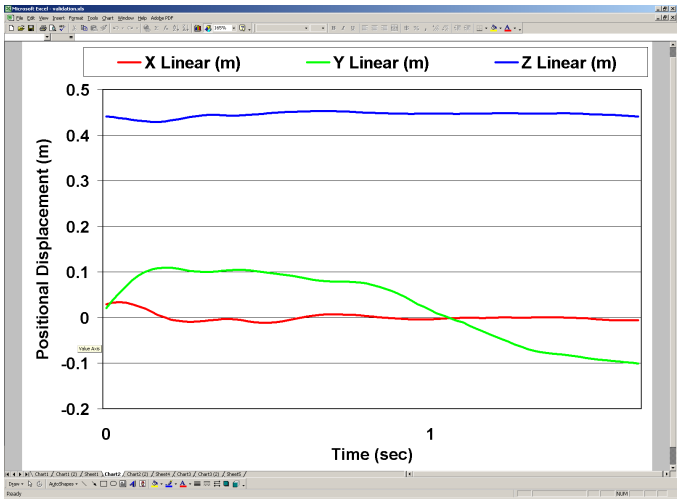


Figure 8.5: Variation in the position of the RMA robot while sliding the support foot backward.

The same flat foot problem applies to this situation too. When lifting the foot off the ground, the toes are normally flexed. The Mobility Simulator does not support the motion in the current version, although mechanical changes could be implemented to allow a less constraining foot attachment.

8.5 Swing Foot Forces and Trajectory

During the swing phase of the gait, the RMA platform is required to follow the motion of the foot while compensating for its own weight and for the forces applied by the user. This task required careful calibration to overcome the impedance created by the air pressure control and the delay caused by the force change bandwidth. The forces recorded during this phase have been presented in section 5.3.5. An important aspect of the swinging phase is the trajectory of the foot. Figure 8.7 presents the recorded position and Figure 8.8 the orientation values of the RMA robot.

The motion is characterized primarily by the reduced range, which is a consequence of the RMA's limited workspace. The motion is however smooth and follows a curve similar to that of a regular foot during walking. The changes in orientation are larger in this case than in the two situations discussed above, but that is normal given the "free" aspect of the motion.

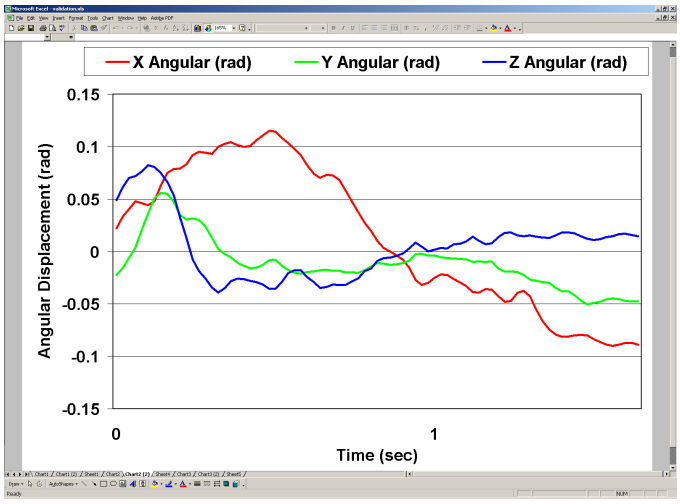


Figure 8.6: Variation in the orientation of the RMA robot while sliding the support foot backward.

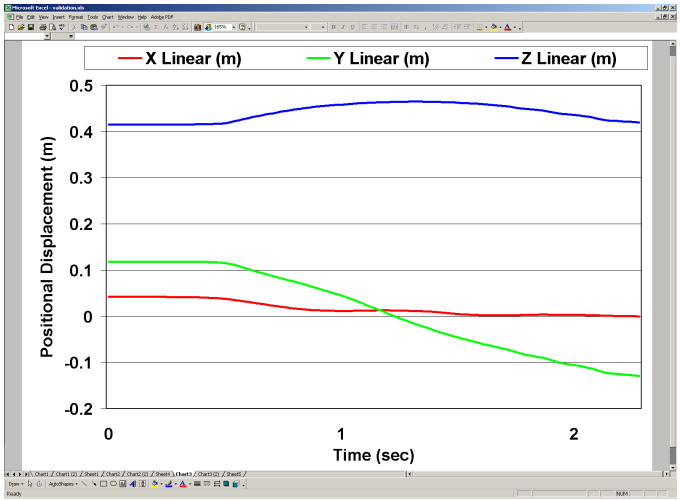


Figure 8.7: Trajectory of the swinging foot when using the Rutgers Mobility Simulator.

8.6 Comparison with Over–ground Walking

The final evaluation of the system was a comparison between the user's gait while on the simulator and the normal gait of the person. The study was designed by the Lewis and Deutsch (RiVERS Lab, UMDNJ) and was implemented in collaboration with The Human Machine Interface Lab at Rutgers. These results have been received from Lewis through personal communication, during his work on a special problem project.

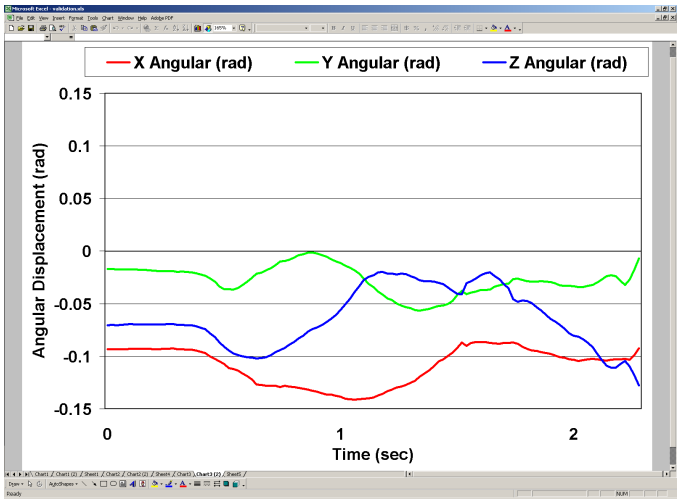


Figure 8.8: Angular trajectory of the swinging foot when using the Rutgers Mobility Simulator.

8.6.1 Experimental Setup

The validation experiment used a GAITRite Sensing mat to record the walking parameters of normal walking in three healthy subjects. Each subject then used the Mobility Simulator to cross a virtual street over a distance of twenty steps. The simulator walking was done with the street crossing simulation initially off, then turned on. A third case was configuration was with the simulation on but with the viewpoint decoupled from the virtual avatar. In the last case, the user would see the virtual world from a fix point and watch the avatar cross the street.

Optical markers were attached to the user's body while the simulator was used and the entire procedure was recorded on a video camera. The video recording was synchronized with the simulator's data collection routine in order to match the digital readings with the gait phases calculated from the video recording.

8.6.2 Measured Parameters

The measured parameters are divided into two groups: temporal parameters and kinematic parameters. The temporal parameters describe the duration of various gait

phases, while the kinematic parameters describe the angles lower body and lower extremities during gait. The temporal parameters and the positional kinematic parameters were measured for each patient during normal gait using the GAITRite mat and while using the Mobility Simulator. The angular kinematic parameters were measured only for the Mobility Simulator using the video recording. Tables 8.1 and 8.2 list the measured parameters and their descriptions.

Table 8.1: Gait temporal parameters.

Parameter	**Description**
Swing time	The duration of the gait swing phase. During the swing phases the foot moves freely above the ground [105]
Stride time	The duration of a stride. A stride consists of two consecutive steps. During one stride each foot goes through the support and swing phases [105, 33].
Cadence	The cadence is number of steps the person walk in a minute

Table 8.2: Gait kinematic parameters.

Parameter	**Description**
Step length	The distance between the two feet during the double limb support phase of gait. The length of the steps can differ depending on which foot is supporting in which is swinging.
Stride length	The length of one stride
Velocity	The walking velocity
Hip angle	The angle of the hip joint measured between three sensors: one placed on the body above the hip, one placed on the hip joint and the third placed on the knee joint.
Knee angle	Knee joint angle
Ankle angle	Ankle joint angle

All the angular parameters measurements use as reference the angles of the joints in neutral position (standing). The normal values of the joint angles in standing are presented in table 8.3. These values were subtracted from the angles measured using the video camera recording. The resulting angles show the differences between normal gait angles and the angles achieved when using the Mobility Simulator.

Table 8.3: Normal neutral hip, knee and ankle angles in neutral position.

Hip	**Knee**	**Ankle**
119 deg	159 deg	107 deg

8.6.3 Results

Subject 1

The measurements performed for Subject 1 are shown in Tables 8.4, 8.5, and 8.6. Each table presents side by side, the measurements in four different conditions: during normal walking as measured by GAITRite (Table 8.4) or as given by normative measurements (Tables 8.5 and 8.6), while using the simulator without the VR simulation displayed in front of the subject, with the VR simulation displayed, or with the VR simulation displayed but with the viewpoint not mapped to the user motions.

As expected, when compared to normal walking the gait on the simulator is much slower. From Table 8.4 it is visible that the swing duration on the simulator is between 59% and 77% longer than for normal walking. The stride duration is 90% to 102% longer than normal walking. The ratio difference between swing time and stride time point to a longer double limb support on the simulator. The reduced RMA robot workspace causes the steps and strides taken on the simulator to be between 78% and 80% shorter than normal steps. The walking velocity is between 88% and 90% slower on the simulator. These results were expected from the beginning because of the small RMA workspace. The walking on the simulator is intended to be a slow short–step walking, which is naturally slower than normal walking.

Table 8.4: Subject 1: Temporal and linear kinematic measurements.

	GAITRite	**Simulator without VR**	**Simulator with VR**	**Simulator with VR (static viewpoint)**
Swing time	0.406 sec	0.674 sec	0.722 sec	0.705 sec
Stride time	1.066 sec	2.161 sec	2.026 sec	2.044 sec
Cadence	112.2 steps/min	55.6 steps/min	59.4 steps/min	58.54 steps/min
Step length	0.821 m	0.162 m	0.176 m	0.173 m
Stride length	1.645 m	0.321 m	0.352 m	0.342 m
Velocity	1.538 m/sec	0.150 m/sec	0.174 m/sec	0.168 m/sec

An interesting aspect is the comparison between the simulator walking with and without a VR simulation displayed on the screen. The situation where the simulation is displayed but the viewpoint is not updated lies in between the values of the previous two cases. When the VR world is not visible, the subject tended to watch her feet during walking, or to watch her reflection in the large screen display in front of her. Anecdotally, the first time the simulator was used for walking, it became apparent that watching the feet while walking caused awkward motions. The presence of the virtual world removes the need to watch the feet, leaving the walking to be control by the other senses. As a result, the swing time is 6% shorter when the virtual environment is displayed. The stride time however is 6% shorter, which means the subject spent less time standing on both feet. With the simulation on, the step length increased by 8% and the stride length by 9%. The increase in velocity was 13%.

The angles of the hip, knee and ankle joints when the foot is lifted off the ground are shown in Table 8.5. The hip angles differ on the simulator by less than 5 deg from the normal gait, while the knee angles differ by less than 10 deg. The differences are caused by the short step walking, which does not allow the foot to extend backward properly. The ankle angle is significantly different between normal and simulator walking. This is caused mainly by the limitations of the RMA rotational workspace. Although in the center of the workspace, the platform can reach the necessary orientation, at the periphery of the workspace the angular workspaces is drastically reduced.

Table 8.5: Subject 1: Joint angle measurements at the beginning of the swing phase is about to break contact with the ground.

	Normal	**Simulator without VR**	**Simulator with VR**	**Simulator with VR (static viewpoint)**
Hip angle	0 deg	4.5 deg	4.33 deg	5 deg
Knee angle	40 deg	30.66 deg	32.5 deg	31.66 deg
Ankle angle	20 deg	-14.16 deg	-16 deg	-15.33 deg

Table 8.6 presents the lower limb angles when the foot touches the ground. In this case, the knee and ankle values are close to the normal gait. The hip angle is significantly smaller than expected. During normal walking, the swing foot lands far ahead of the body causing the 30 degree angle shown in Table 8.6. The short workspace

of the RMA robots does not allow for such long steps, hence the difference in hip angle.

Table 8.6: Subject 1: Joint angle measurements at the end of the swing phase when the foot touches the ground.

	Normal	**Simulator without VR**	**Simulator with VR**	**Simulator with VR (static viewpoint)**
Hip angle	30 deg	2.5 deg	3.33 deg	4 deg
Knee angle	0 deg	3.16 deg	3.83 deg	3 deg
Ankle angle	0 deg	-3.33 deg	-4.83 deg	-3.33 deg

Subject 2

The measurements performed for Subject 2 are shown in Tables 8.7, 8.8, and 8.9. The patterns present in the first subject's data are visible with certain differences in the second subject's data as well. When using the simulator the swing time is between 62% and 104% longer than during normal walking. The stride duration was also longer by 37% to 47%. The larger difference in swing time means that the double leg support time when using the simulator is shorter than when walking normally. The changes in the step length are between 79% and 80%, while the differences in stride length are between 73% and 75%. The walking velocity on the simulator is slower by 83% to 86%. These results are consistent with the parameters of a short–step and slow walking. The interesting walking pattern in this subject's data is the very short ground contact with both feet. This could be an effect of the unweighing system. The percentage of supported weight it is a parameter still to be tuned. If too much weight is supported, then the subject can essentially keep both feet in the air for the short amount of time it takes the suspending chord to extend and lower the user.

The performance improvements while immersed in the VR simulation noticed for the first subject are found in the second subject's data as well. The step length and the stride length increased by 9% while the virtual environment was displayed in front of the subject, while the velocity increased by 15%. Subject 2 also presents performance differences between the VR simulation with and without the viewpoint coupled to walking. Interestingly without the viewpoint coupled to the motion, the performance values are lower than those recorded when the VR scene was not displayed at all.

Table 8.7: Subject 2: Temporal and linear kinematic measurements.

	GAITRite	**Simulator without VR**	**Simulator with VR**	**Simulator with VR (static viewpoint)**
Swing time	0.486 sec	0.786 sec	0.995 sec	0.818 sec
Stride time	1.280 sec	1.845 sec	1.748 sec	1.889 sec
Cadence	93.43 steps/min	65.16 steps/min	68.89 steps/min	63.31 steps/min
Step length	0.863 m	0.183 m	0.199 m	0.177 m
Stride length	1.456 m	0.369 m	0.401 m	0.358 m
Velocity	1.347 m/sec	0.199 m/sec	0.229 m/sec	0.187 m/sec

As discussed for Subject 1, the differences in hip and ankle joint angles are mainly caused by the reduced robot workspace. The data shown in Tables 8.8 and 8.9 shows significant differences from the normal angles, in the ankle lift–up angle and the hip touchdown angle. The knee angles are with 14 degrees difference from the normal angles

Table 8.8: Subject 2: Joint angle measurements at the beginning of the swing phase is about to break contact with the ground.

	Normal	**Simulator without VR**	**Simulator with VR**	**Simulator with VR (static viewpoint)**
Hip angle	0 deg	-3.06 deg	0.6 deg	2.6 deg
Knee angle	40 deg	26.8 deg	31.13 deg	31.8 deg
Ankle angle	20 deg	-15.46 deg	-16.63 deg	-14.13 deg

Table 8.9: Subject 2: Joint angle measurements at the end of the swing phase when the foot touches the ground.

	Normal	**Simulator without VR**	**Simulator with VR**	**Simulator with VR (static viewpoint)**
Hip angle	30 deg	-4.9 deg	-4.4 deg	-6.73 deg
Knee angle	0 deg	3.8 deg	6.96 deg	2.8 deg
Ankle angle	0 deg	-0.3 deg	2.36 deg	-0.46 deg

Subject 3

The measurements performed for Subject 3 are shown in Tables 8.10, 8.11, and 8.12. The gait patterns displayed by the third subject when using the simulator are consistent with the results of the previous two subjects. The measurements show an increase in the swing duration of 72% to 134%. The increase in stride duration is between 94% and

159%. The step and stride lengths are shorter than normal walking by 74% to 78%, and the velocity is lower by 88% to 90%. The measurements of Subject 3 are similar with the results of the other two subjects, but there is a noticeable percentage increase in the swing and stride duration and a larger percentage decrease in the walking velocity when using the simulator. Among the reasons for this behavior is the height of Subject 3, who was shorter than the other subjects were. Being shorter, the subject was affected more by the lateral distance between the two platforms of the simulator. Therefore, the difference gait while on the simulator was larger for this subject than for the others.

The effect of the VR simulation on performance is evident for this subject too. The swing and stride durations were reduced 35%, respectively 33% when using the VR simulation. The velocity while immersed in the virtual environment increased by 14%.

Table 8.10: Subject 3: Temporal and linear kinematic measurements.

	GAITRite	**Simulator without VR**	**Simulator with VR**	**Simulator with VR (static viewpoint)**
Swing time	0.399 sec	0.935 sec	0.689 sec	0.767 sec
Stride time	1.02 sec	2.642 sec	1.987 sec	2.185 sec
Cadence	117.73 steps/min	45.51 steps/min	60.81 steps/min	55.06 steps/min
Step length	0.752 m	0.194 m	0.165 m	0.186 m
Stride length	1.509 m	0.389 m	0.331 m	0.374 m
Velocity	1.479 m/sec	0.147 m/sec	0.168 m/sec	0.171 m/sec

The lift–up and touchdown angles of knee are very close to the normal values for Subject 3. The touchdown ankle angle and the hip angles for both situations are larger and respectively lower than the normal values, which is similar to the behavior of subjects 1 and 2. A notable difference is in the touchdown ankle angle, which is larger than normal, where as the other subjects presented values close to the normal neutral position.

8.6.4 Gait Comparison Conclusions

For all subjects, the walking velocity on the simulator was approximately a fifth of the normal walking velocity. The step lengths were shorter being constrained by the

Table 8.11: Subject 3: Joint angle measurements at the beginning of the swing phase is about to break contact with the ground.

	Normal	**Simulator without VR**	**Simulator with VR**	**Simulator with VR (static viewpoint)**
Hip angle	0 deg	16.16 deg	6.83 deg	6.16 deg
Knee angle	40 deg	40 deg	37.16 deg	40 deg
Ankle angle	20 deg	-11.33 deg	-13.66 deg	-13.33 deg

Table 8.12: Subject 3: Joint angle measurements at the end of the swing phase when the foot touches the ground.

	Normal	**Simulator without VR**	**Simulator with VR**	**Simulator with VR (static viewpoint)**
Hip angle	30 deg	12 deg	8.83 deg	6.5 deg
Knee angle	0 deg	0.16 deg	-0.66 deg	-3.33 deg
Ankle angle	0 deg	17.33 deg	17 deg	16.66 deg

workspace of the RMA robots. The joint angles measured during walking differ between the normal and simulator walking at hip level during touchdown due to the short forward displacement of the leg imposed by the robots, and at ankle level during lift–up due to the reduce orientational workspace of the RMA robots at the edge of the positional workspace.

The results presented in the previous sections show that the presence of the virtual environment in front of the user can change the gait of the user bringing it closer to the natural motions. This is mainly caused by the "vision capture" phenomenon that removes the user's focus from their own body, allowing it to be controlled by reflexes rather than by a conscientiously controlled feedback loop. The improvements related with the presence of the virtual reality simulation were visible in the length of the steps and the walking velocity, which was increased by 8% to 14% compared with the velocity achieved without VR.

The validation tests also pointed a few system parameters that need to be tuned for each user. The amount of body weight supported by the frame proved to be an important factor that can cause unnatural gait by making the body too light. The height of the user is also an aspect that has to be taken into account. The shorter the user the more he or she will be affected by the lateral distance between the platforms. In the current stage of the system, this aspect can be improved only by giving the user

more time to get accustomed to the system.

Chapter 9

Remote Real–Time Monitoring and Therapy Control

The real–time remote monitoring feature was added to the telerehabilitation framework in the third version of the software. The need for such a tool arose during full day patient trials for the hand and ankle system that required a number of researchers and therapists to be present in the same room with the patients to watch and coordinate the flow of the therapy. After the first weeks of trials, it was apparent that their involvement could be just as effective from a remote location if they could see and interact with the patients and their exercises in real–time. Such an approach would have saved time for the therapists and researchers involved in the study.

Existing approaches for solving this problem involved the use of video conferencing. Although mature at the time, the teleconferencing tools did not allow the therapist to watch directly the patient's activity but also the exercises displayed in front of them by the graphical workstation. Pointing the camera to the screen yielded faded and noisy images unusable to watch the VR exercises in action. In addition, the camera had to be continuously moving from focusing on the screen to focusing on the patient.

The remote monitor solved this problem by bringing the activity of the virtual patient in front of the remotely located therapist. Simplified mock–ups of the VR exercises are displayed on the therapist's computer and synchronized with the rehabilitation side activity.

The initial monitoring implementation was aimed only for the post–stroke rehabilitation exercises for the hand (see Appendix B). The monitoring applet shown in Figure 9.1 displayed a simplified hand model optimized for fast rendering, and information about the progression of the rehabilitation session. The positions of the virtual fingers were updated based on datasets sent by the rehabilitation application.

The VR exercises collect and evaluate data in real–time. The rate of the data thread has to be as constant as possible for filtering purposes. To avoid interfering with this

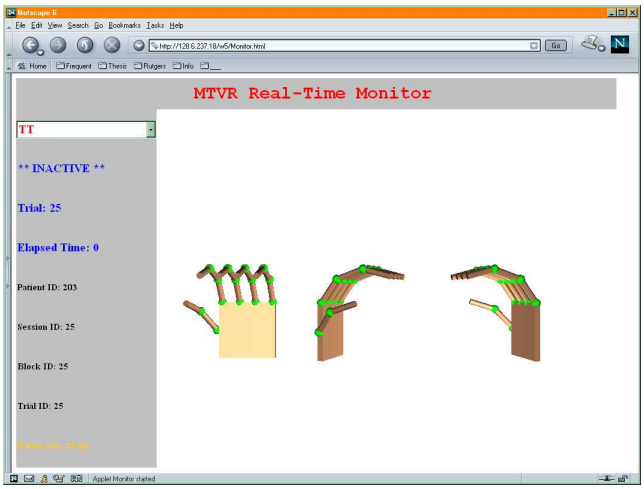


Figure 9.1: Initial hand rehabilitation monitoring applet [97].

balance, which was tuned to work with the intense graphics part of the exercises, the VR simulation stream out the data packets to a server running on the same computer as a different process or on a remote computer. Any number of monitoring clients could connect to the server without affecting the performance of the VR simulation. This implementation was tested and the conclusions pointed to the need for improvements in the data transmission approach and the visual aspect of the monitoring applet.

The graphical improvements for the monitoring applet required the implementation of simplified mockups of each exercise in the rehabilitation system, so that the therapist could get a closer feel of the user's experience and better understand his or her reactions during the therapy session. In addition to this feedback, a video conferencing tool was also added to the system to bring to the remote therapist direct visual feedback of the patient's motions, facial expression and verbal comments. Figure 9.2 shows the mockup developed by Lewis for the hand strength exercise as part of his thesis research work [70, 71].

The main issue with the data transmission was the location of the server. While the trials were run in the RiVERS laboratory of the University of Medicine and Dentistry in Newark, NJ, the server was running on a machine 35 miles away, at Rutgers in Piscataway, NJ. This was a good setup for somebody monitoring the therapy from

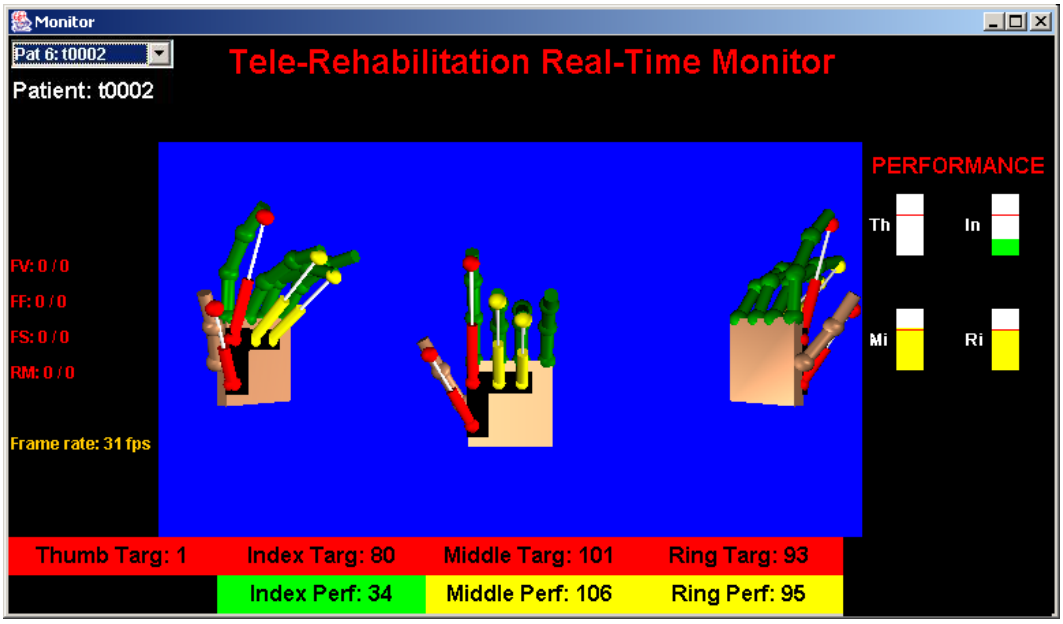


Figure 9.2: Monitoring applet mockup of the post–stroke finger strength exercise [2].

Piscataway, but for the researchers testing the system in Newark, the data was unnecessarily delayed by first being sent to the server in Piscataway and then back to Newark to the monitoring applet. Moving the server on a computer in Newark would have solved the problem for the UMDNJ staff, but worsen the condition for those in Piscataway. The responsiveness of the system was affected by the surprisingly slow and unstable Internet connection available between the two universities. The current chapter presents the new architecture of the data transfer service and the additional tasks that it has to accomplish.

9.1 Distributed Virtual Environments

As a result of advancements in the communication hardware and software performance, network–based collaboration is a very active area of research. The possibility of meeting and working together with other persons without the necessity of being at the same physical location brings the major benefit of making such events convenient and less costly without losing efficiency. Virtual environments are an ideal candidate for such collaborations, either in a pure 3D graphic form or augmented with inserts of real–life video streaming, or instrumented with force and touch feedback devices.

The most popular distributed virtual environments today come from the on–line gaming companies. A more practical application is the collaboration between international experts on the design of a building [75] or any other CAD project. A more challenging application is remote force feedback cooperation. This is a very sensitive area due to the haptic device stability concerns due network delay and jitter.

These examples require collaborative or cooperative type of interaction between the users involved. One major issue in such environments is keeping the virtual environment consistent on all parties' display. This requires implementing a distributed synchronization scheme [76, 108], whose correctness usually has to be balanced with the performance of the system. In some cases, the synchronization algorithm is greatly simplified by allowing only two users to interact at one time. An example of such application is Popescu's shared virtual rehabilitation room with force feedback [98]. In the case of the distributed games, the synchronization is usually done by a dedicated server that manages an instance of the game, accepting up to a certain number of simultaneous users.

9.2 The Remote Monitoring Application as a Distributed VE

The remote monitoring system presented here is a simplified version of a distributed virtual environment. It does not require direct interaction between the two users in the virtual environment. The patient at one end has full control of the VR game, while the therapist at the other end only watches the VR scene and possibly changes some simulation parameters. Although both parties modify aspects of the simulation, they work in isolated spaces. The patient is concerned only with the virtual side of the exercise while the therapist has control only over the underlying configuration. This approach reduces the problems raised by conventional shared distributed VEs in that it doesn't require a complex synchronization mechanism.

9.3 Monitoring Service Architecture

The new design of the data transfer service uses multiple server nodes (forwarding nodes) running in various geographical locations. These nodes are organized into a multicast overlay network where each node becomes a source of data if a VR application connects to it. The data streamed by the VR exercise is multicast by the source node to the rest of the server nodes in the network. A remote monitoring client then connects to one of the server nodes (selected by the response delay and age of the data) and requests data streaming for one or more patients. Besides monitoring, the remote therapist also needs to be able to control and direct the rehabilitation session. To facilitate such a feature, each rehabilitation session is required to have a local server running in the background. This server would be responsible for taking the data from the simulation and multicasting it to the rest of the nodes and transmit to the simulation or change configuration files on the local hard drive upon request from the remote therapist. Figure 9.3 shows a schematic view of the monitoring network structure.

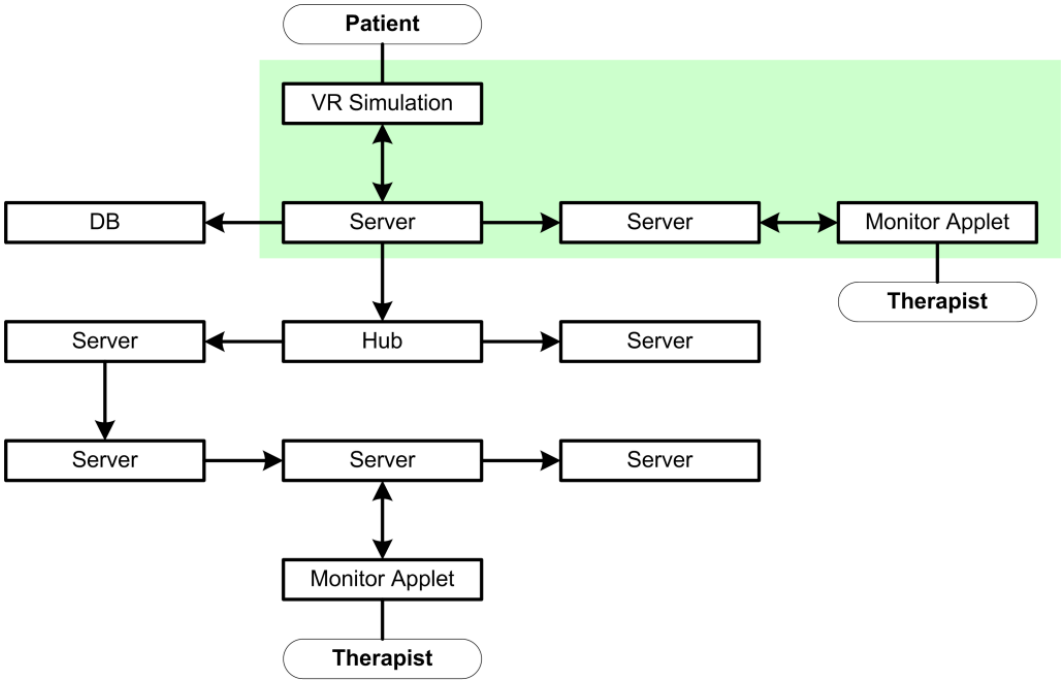


Figure 9.3: Monitoring network structure. The shaded area encloses LAN connections, while the rest of the connections are WAN.

The monitoring network is composed of three types of nodes: server, monitor, and

VR simulation. The server node connects the entire overlay network together. It is responsible for handling the requests of all the other types of nodes. The monitor node is used by the therapist to connect to the overlay network and watch the activity of one or more patients or adjust their rehabilitation sessions based on their performance. The VR simulation node is practically any of the VR rehabilitation exercises. Such a node only streams out the data collected by the sensors during therapy and receives run–time parameter change commands forwarded through the local server node from a remote monitor node.

One of the server nodes is designated as hub and serves as rendezvous point for all the new servers or monitor nodes that enter the network. A VR simulation node is not registered with the hub directly, but only with the local server node.

Another special server node is the DB node. This is the server running on the database computer. It is responsible for receiving all the bulk patient data and writing it into the database.

9.4 Server Node

During the redesign of the service, the server's role was expanded to be used for not only monitoring and remote control, but also to support bulk patient data transfers from the rehabilitation site to the database site, and serve as an on–line directory transparently mapping patient and therapist usernames to their network addresses. The full list of tasks executed by a server is

1. Multicast patient data generated by the VR exercises to all the nodes in the network;

2. Continuously measure the round trip delay between the nodes in the network; this will be used in QoS adoptions to poor or slow Internet connections;

3. Periodically rebuild the multicast tree based on the round trip delay updates and the new nodes arrived into the system;

4. Store directory of connected users (patients or therapists) and perform transparent mapping between their names and their network addresses;
5. Schedule bulk patient data transfers to the database site;
6. Respond to remote commands sent by the therapist reading local configuration file changes;
7. Communicate with the local VR simulation, sending configuration parameters to be changed at run–time;
8. Automatically start the teleconferencing utility upon request from one of the parties;
9. Stream patient data to a monitoring client upon request.

The monitoring service has to run in network environments belonging to hospitals, universities, research centers, and patient's home. In such a heterogeneous network the system had to be designed for small data transfers to address the usually low bandwidth of the home connections, and also be easy to integrate in the tight firewall security provided by hospitals and some universities. The firewall issue was a serious concern in the design of the system, considering that network administrators are rather reluctant to grant access to a new unknown application. To make the application acceptable to them, the severs were designed to use asynchronous communication over two sockets: one for UDP transfers and the other for TCP transfers [9, 10]. The TCP data transfers are necessary for critical information such as multicast tree messages, bulk data transfers or monitoring command and requests. The UDP protocol was selected for transferring small and fast packages of VR simulation data. Granting UDP permission through firewalls or across virtual private networks (VPN) has proven problematic, so a quick switching option was implemented to allow the traffic designed for UDP to be sent over TCP sockets.

The server was designed as an inter–node communication framework with the tasks presented above implemented as services running simultaneously in their own threads. This approach fitted best the requirements of an expanding system that has to easily

support extra features. Each service is instrumented with a message queue that stores the messages arrived on the common UDP socket or the TCP sockets. Details about the services implemented in the current version of the server are presented in Appendix C.

9.5 Message Format

The message format used by the remote monitoring system starts with a common identification and addressing header followed by message specific data. Table 9.1 presents the message fields.

Table 9.1: Message format.

Field name	**Byte Length**	**Description**
Length	4	The length of the message buffer as it is sent over the network
Type	1	ID of the message type
Flag	1	Bitwise flag settings regarding the way the message should be handled. Supported flags are MULTICAST and REALTIME and STORE
Source ID	2	The ID of the node the generated the message. This is relevant for multicast messages where the source does not coincide with the node that the message was received from
Destination service type	1	The type ID of the destination service.
Destination service ID	1	ID of the service on the remote node that has to process this message
Source service type	1	Type ID of the service that sent the message
Source service ID	1	ID of the service that sent the message
Multicast tree ID	2	The ID of the multicast tree used to forward this message
Data length	4	Length of the message specific data

The messages are exchanged between C/C++ programs on various platforms and Java applications. Before it is sent out in the network, the message is encoded in a byte buffer and then decoded at the destination. To insure type compatibility between the various platforms, the values of the message fields are translated into the Java number format before they are stored into the buffer.

Each message has been implemented as a separate class. To reduce the size of the

code, messages with very similar data have been collapsed into a single class implementation with a subtype filed identifying it. This permits a comfortable development and more error safety. However, encoding and decoding of messages requires memory handling and data format translations, which cause certain delays. These operations are not performed on messages being multicast in real–time. Instead of fully extracting the message information into the class members, only a reduced set of values necessary to identify the message destination and source are decoded. Messages marked by the STORE flag are copied in the nodes data structures in their byte buffer format and streamed to the client directly, thus saving the cost of and additional encoding/decoding for each message.

9.6 Monitoring Client

The monitoring client has been implemented using the Java language and the Java3D graphics library. This choice was made to take advantage of the platform independence and in–browser access to Java applets. Offering the client as an applet is the most convenient and accessible option, since the therapist is not required to perform any software installation on the local machine. The monitoring client is the project developed by Lewis for his Master degree research. Lewis extended the initial simple client and integrated it with the new monitoring service presented above [70]. This caused problems with the original web applet approach, since the client has to create network connection to several other computers besides the one it has be loaded from. This conflicts with the default Java security settings. This issue was solved by switching the implementation from Java applet to a stand–alone application. The access to the application is still done through a web link, but the delivery is done by the Java Web Start technology packaged with the newer versions of the Java SDK.

The monitoring client is designed to connect to the server nodes in the overlay network and measure the round trip delay to each of them, then connect to and require patient data streaming from the node with the fastest connection. This is necessary to remove the client's sluggish interactivity caused by a slow network.

9.6.1 Rehabilitation Simulation Mock–ups

The remote monitoring client uses simplified mock–ups of the VR rehabilitation exercises to give the therapist insight into patient's activity. These simplified simulations were designed to be as light on resources as possible because of the following reasons:

- The computer used by the therapist to access the client is rarely a high end workstation;
- The client may at some point be run in a browser, which usually slows down the execution;
- The therapist does not need a complete simulation since he or she only needs to watch without interaction;
- Developing and maintaining smaller mock–ups is much easier than keeping in sink two versions of the same software.

A mock–up of an exercise does not contain any of the underlying logic of the real simulation. It only creates and displays the essential 3D objects and features of the real exercise and updates their positions based on the data sent from the rehabilitation site where the patient undergoes the therapy. For instance, the mockup of the airplane exercise for ankle rehabilitation in sitting, displays a boxy airplane model and three hoops head of the airplane (see Figure 9.4). Thus, the client simulation only has to update the position and orientation of the airplane and of the next three hoops as sent by the real simulation. The side effect of this approach the doubling the size of the messages. However, even with this doubling, the maximum message size in the current implementation does not go above 150 bytes.

9.6.2 Initial Street Crossing Simulation Mockup

The Java simplified version of the street crossing simulation is under development. A preliminary version is shown in Figure 9.5. The street exercise simulation contains more visual entities than the rest of the exercises, but not all of them need to appear in the

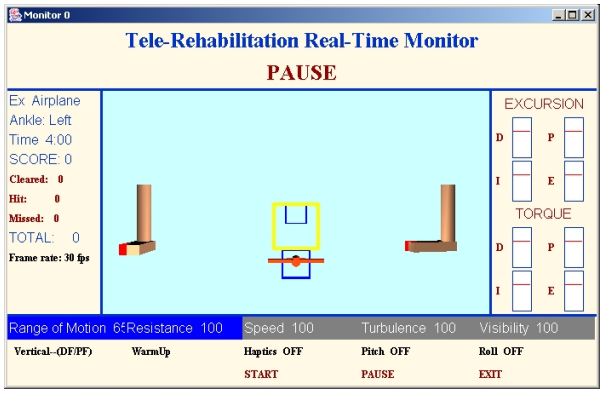


Figure 9.4: Airplane exercise monitoring mockup [71].

mockup as a graphical entity, nor do they have to be represented ass realistically as they are in the real simulation. The street crossing simulation parameters and their presence in the monitor is presented in Table 9.2. All the 2D feedback displayed on the screen will appear in the monitoring client as 2D feedback too.

Figure 9.5: Initial street crossing exercise monitoring mockup screen.

Table 9.2: Message content.

Parameter	**Appears in mockup**	**Implementation Details**
Street width	Yes	On horizontal rectangle will be displayed for each street lane and sidewalk
Curb height	Yes	The elevation of the sidewalk will be changed in accordance with the curb height
Sidewalk edge shape	Yes	If the sidewalk edge is sloped the portion of the sidewalk corresponding to the crossing will be tilted
Crossing type	Yes	The crossing will be displayed either as a zebra or as two parallel lines
Surface patches	No	The patches will not be displayed since they contain numerous vertices and will slow down the client and the communication. Instead, the type of the patches will be transmitted and the walking surface will be displayed in an appropriate color. When the user steps on a patch of special surface, the foot in the monitoring client will be highlighted.
Vehicles	Yes	Only two four vehicles will be displayed using simplified boxy 3D models. They will be the vehicles closest to the crossing on the current lane and the next lane.
Stop–light color	Yes	
Sounds	No	

Chapter 10

Data Storage and Access

The design of the Multiplexed Telerehabilitation System relies on a central high–end server for data storage. Besides collecting the data, the server site also provides a web application for data access to be used by the physicians and therapist involved. The application was designed to support four different rehabilitation projects involving exercises for the rehabilitation of hand, ankle and gait. The similarities of the therapeutic procedures have been identified and the implementation easily supports extensions for new rehabilitation projects. The design is split over three tiers.

1. The back tier consists of an Oracle database server that stores the data collected by the VR exercises during the rehabilitation sessions;

2. The middle tier consists of Java servlets that mainly interpret the requests of the user and return his or her data in the desired format;

3. The front tier is designed to simplify the construction of the data requests to be sent to the server. It currently has two implementations: a simple HTML implementation that provides access to predefined requests, and a Java applet that brings full flexibility for data selection.

10.1 Data Entities

The data collected by the rehabilitation exercises are stored in files in a hierarchical structure reflecting the flow of the therapy. Each patient can participate in one or more rehabilitation projects, over the period of one or more studies. The projects are necessary to separate the data based on the type of therapy, exercises and hardware employed. The studies provide data separation with respect to time, grouping patients over time–periods.

For a certain project, a patient goes through a sequence of therapy sessions, scheduled usually either daily or three times a week. For certain situations, a patient may have to go through multiple sessions in a day.

Each rehabilitation session consists of a series of exercises provided by the system, and executed in the order decided by the therapist or physician. An instance of such an exercises is called trial. Depending on the exercises, the length of such a trial can vary significantly between 30 seconds and 15–20 minutes. Regardless of the length of a trial, the patient executes a sufficiently large number of trials to last for an hour or more of therapy.

In the case of the short trials, a single instance did not give much information so it was necessary to view such trials in contiguous groups called blocks. The structure of a session is presented in Figure 10.1.

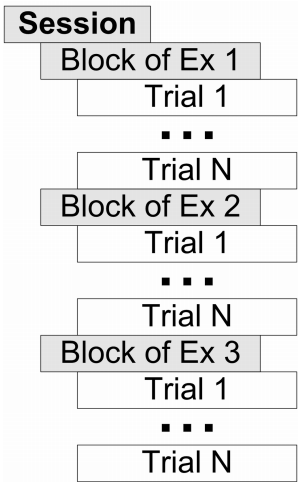


Figure 10.1: The structure of one session.

For each trial, the system collects data in real time and stores them in files. These data are called “raw data” and they are the great majority of the data stored by the system. Besides raw data, the system also stores the configuration of the exercises and the performances recorded during the session. The performances are calculated in relation with numerical targets set by the therapist, which are also stored.

10.1.1 Data Size

The system collects raw data with a frequency of 50 Hz. Since raw data are the vast majority of all the data stored, the rest of the data will be ignored for estimating the size of the data.

The size of a raw data record varies between projects. For instance, the hand rehabilitation projects store the angles of the finger joints and the position and forces applied by the fingertip. The ankle exercises store the 6DOF position and forces of the ankle. The gait exercise stores the 6DOF position and forces of each foot, in both real and the virtual environments. On average, the size of a record is of 20 floating point numbers.

Considering the length of a session to be on average one hour, and the percentage of time during which the patient exercises to be 60%, the size of the data collected for one session is approximately 8.2 MB. With a session count between 12 and 20, the size of the data stored for a patient during a therapy study is between 98 MB and 164 MB.

10.2 The Back Tier: Oracle Database

Although modern tiered application usually solve the authentication and permissions in the middle tier, in this case it was proven more efficient to use the features offered by the Oracle server instead of reinventing the wheel and implementing a whole new security scheme. The data access is controlled based on the roles assigned to each user and the user's association with each study. While for access based on the user privileges, Oracle provided the roles implementation, the later type of access control required row–based data access separation, which was implemented manually using database views and synonyms.

The database was design to reflect the structure of a therapy session. The tables are organized on a four hierarchical levels: session tables, block tables, trial tables, and raw data tables. Optimizations for fast access were implemented by storing raw data evaluations in separate tables and propagating foreign keys down through the table hierarchy. The architecture also provides tables for data filter storage designed support

comprehensive data selection for filtering purposes as well as fast access to filtered and unfiltered data. Details on the database design are presented in Appendix D.

10.3 The Middle Tier: Java Servlets

The middle tier makes the connection between the database and the end–user interface. The main activity of the middle tier is to upload to the user the data requested as charts (images), or as plain numbers in spreadsheet, text, or HTML format.

A data request is usually defined by a large number of parameters specifying the patient, the exercise, the body part, the time interval, and specific exercises parameters. During the prototype phase, the portal had to provide the researchers full accessibility to all the possible options in a concise manner.

The data requests coming from the user are encoded in a hierarchical text format that is decoded by the middle tier and transformed into graphs or other data formats. Although XML is the preferred choice when dealing with hierarchical data, it is too verbose for the purposes of this project and slow to parse. The developed format is inspired from XML, using as delimiters curly brackets instead of tags. The format does not implement the equivalent of the XML attributes.

A data request is organized in multiple hierarchical entities, each of which consists of a set of general parameters, a set of module specific parameters (i.e. exercise dependent), and a list of child entities. Figure 10.2 shows the structure of a list of requests that is sent to the middle tier by the user.

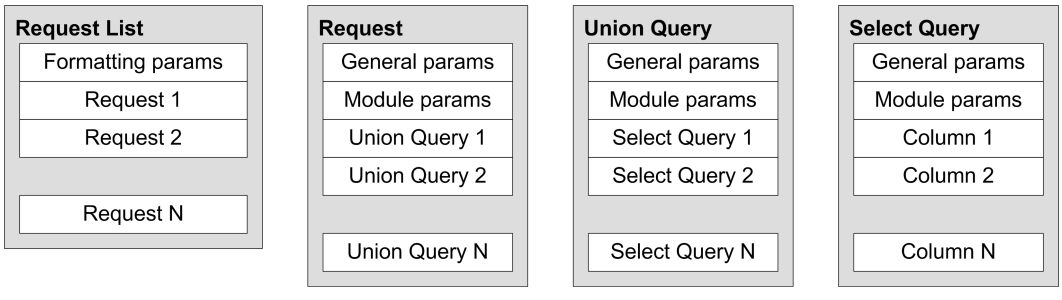


Figure 10.2: Data request list.

10.3.1 Data Request Structure

There are two major types of data requests: performance history request, or raw data requests. The processing required for each of these two types differs significantly. The raw data graphs need to show the data and mark on the graph the position of the events that occurred during the trial (see Figure 10.3). Besides the type of the request, the general parameters include information about the size and density of the graphs to be generated. This information is ignored if the return type is a plain data one. The density of the graphs specifies the maximum number of data points to be shown per pixel of image width. In case there is too much data, more than one graph will be generated for one request, by splitting the data and hence making the charts easy to read.

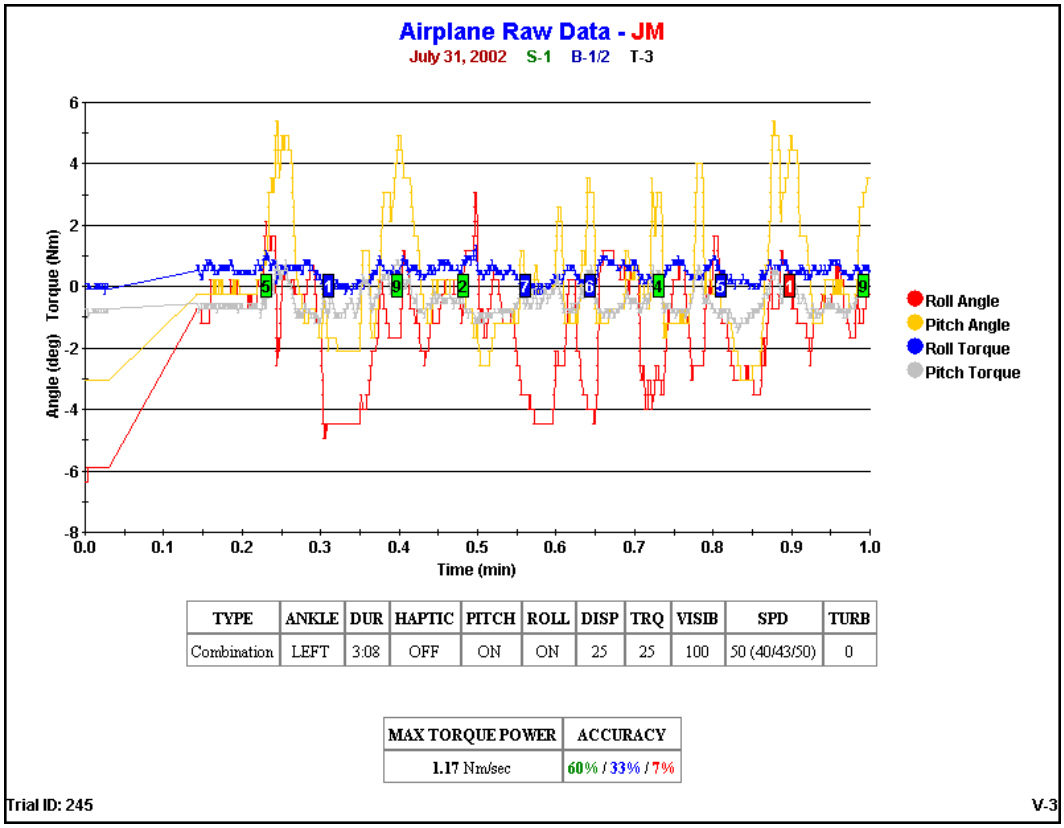


Figure 10.3: Airplane exercise raw data.

There is only one module specific parameter in the structure of the data request. It is used for performance history graphs to specify whether the X–axis should be a scaled

timeline or just show the data points equidistantly ignoring the actual time values. The former format gives a better view of the therapy sequence over time, while the latter allows easier comparison between data points.

The last part of a request is a list of union query specifications. From these specifications, the servlets will generate the database queries necessary to retrieve the data and create the charts.

10.3.2 Union Query Structure

The union query part of a request was introduced only in newer versions of the portal. An SQL union clause simply concatenates the results of two or more regular select queries. The use of such queries became necessary when the ankle project was added a second exercise very similar with the initial one. The two exercises had enough differences to justify different table storage but it also made sense to look at both exercise data together. Hence, the patient history values of the two exercises were retrieved with regular select queries and concatenated using the union clause. In the situations where the data requested comes from only one tables, the union query will not have an effect on the processing speed.

The only general parameter of the union queries is the username of the patient whose data are requested. In the current stage of the system, the raw data union queries do not need any module specific parameters. The performance history union queries use six parameters:

1. *Data grouping level* - Allows grouping the data across days or keeping it at trial level;

2. *Data grouping function* - If the data are to be grouped, this parameter specifies how it is grouped. The supported options are: summation, average, minimum and maximum;

3. *Starting date* - Extract data only newer than the given date;

4. *Ending date* - Extract only data older than the given date;

5. *Visible data aspects* - For each data column extracted, the portal can display in a graph the data itself and the unfiltered version of the data. In addition, the data points can be added the values of the standard deviation resulted from grouping. The last feature that can be displayed is the a fitting line calculated with the least squares method (linear regression);

6. *Chart type* - Selects whether the chart should be a line chart, scattered plot, or bar graph.

The last parameter of the union query is the list of select queries. The select queries will be used to generate single table queries that will be concatenated under the UNION clause. All the select queries specified for a union query must have the same number of columns selected and the columns must match types across the select queries. Figures 10.4, 10.5, 10.6, 10.7, 10.8 show a few examples of data graphs generated by the portal.

10.3.3 Select Query Structure

The select queries take as general parameter the exercises name. This is used to identify the table from which to extract the data. For convenience, multiple exercises can be specified, and the portal will split the given select query in multiple sub–queries that will eventually be concatenated by the union clause.

For the raw data queries, the module specific parameter is the trial ID. The performance history queries several parameters necessary to provide the therapist with the flexibility needed to see all the aspects of the data. The first group of parameters is used to describe the data to be selected. These parameters include information about the limb side to be graphed, and value ranges of exercise specific settings.

The last of the parameters is the name of the filter to be used when retrieving the data. The filter name will be correlated with the requested data, and the data selection parameters to find the ID of the filter in the FLTS table.

The general and common parameters of the select query are followed by a list of column names to be retrieved from the database table. These names are not necessarily

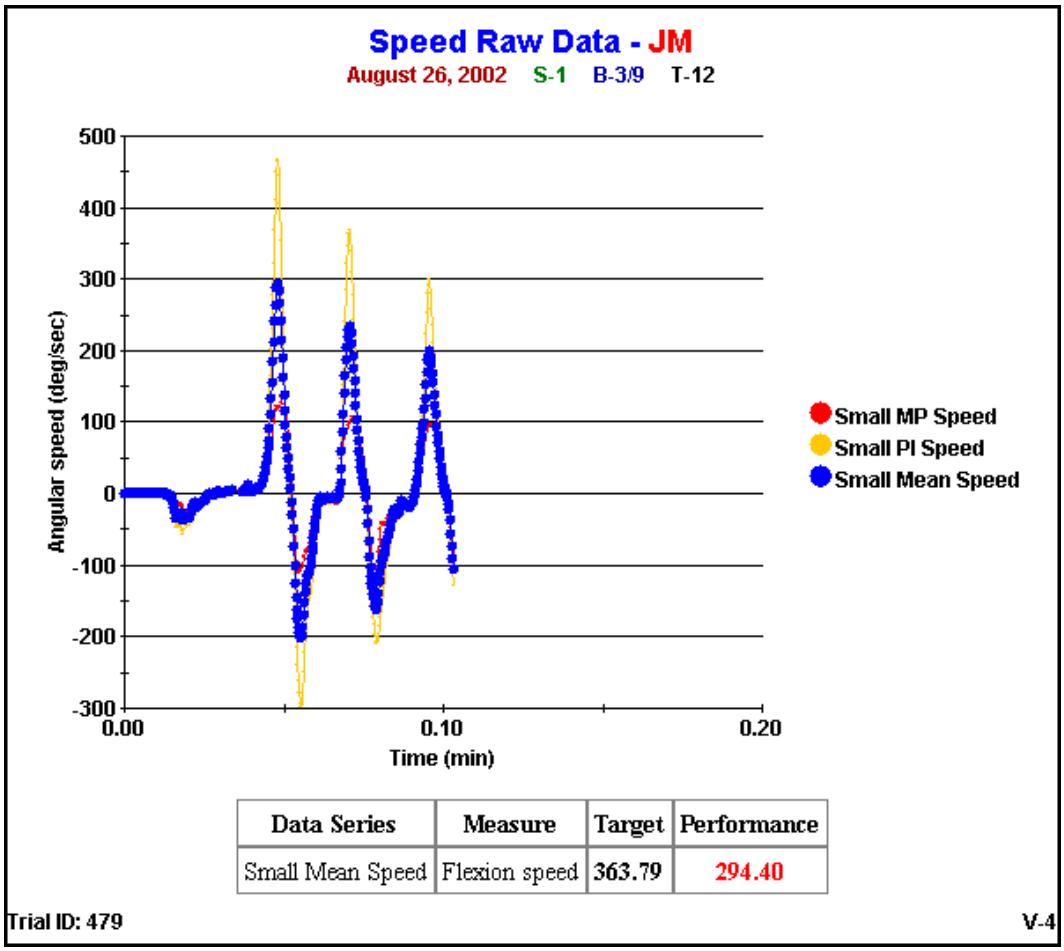
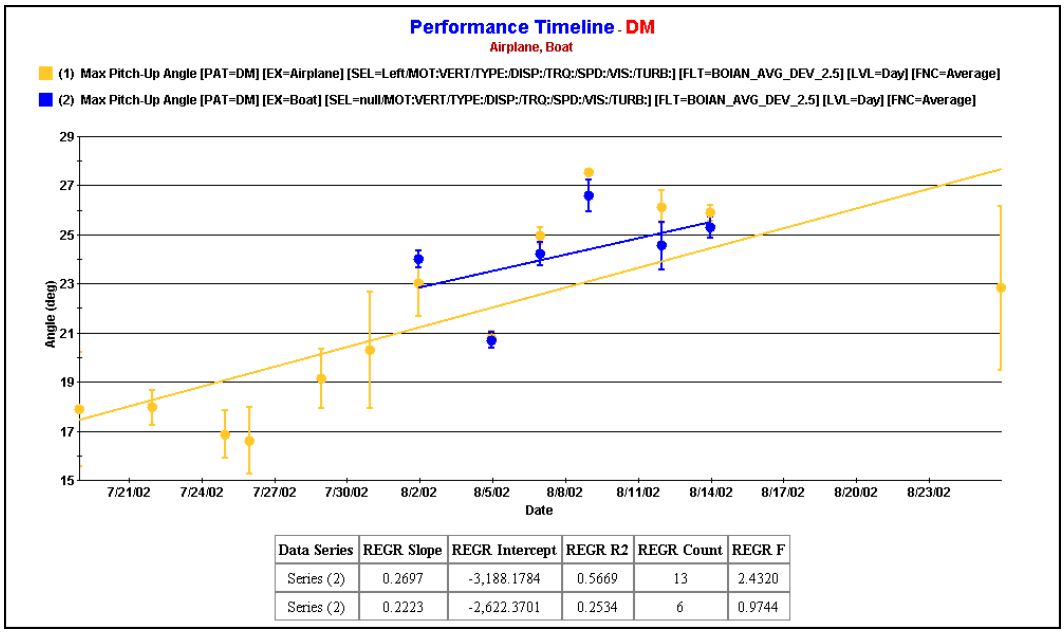


Figure 10.4: Finger velocity exercise raw data.

matching the exact database names. They are implemented as aliases that are matched in an external dictionary with an SQL expression. Using this feature, the data stored in ISO units can be transparently converted into units more familiar to the therapist (e.g. degrees instead of radians).

10.4 The Front Tier: User Interface

The front tier of the web portal has been designed to provide the therapist with an easy and intuitive way of browsing the data collected during the rehabilitation sessions. There are two versions of the web portal: a non–interactive HTML–based implementation and a flexible more interactive version implemented as a Java applet.



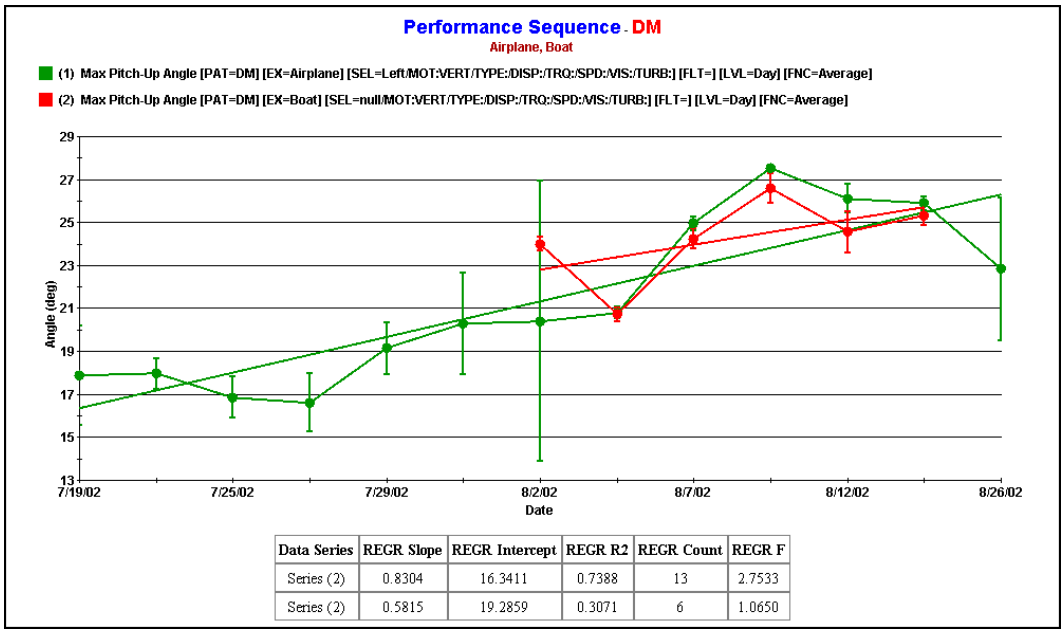
Data Series	REGR Slope	REGR Intercept	REGR R2	REGR Count	REGR F
Series (2)	0.2697	-3,188.1784	0.5669	13	2.4320
Series (2)	0.2223	-2,622.3701	0.2534	6	0.9744

Figure 10.5: Ankle exercises patient history data. The graphed parameter is the PITCH–UP angle. The X–axis is formatted as a time line, and the data are grouped by days.

10.4.1 HTML–based Interface

The HTML interface has been developed as an easy to modify interface that required less development time and provides immediate access to the data stored by new systems added to the telerehabilitation framework. The web page is split in two frames (see Figure 10.9). The left hand side frame displays a list of studies and the subject that was part of them. Upon clicking one of the patients, the right hand side frame displays a history of the patient's activity. The history is grouped by days, session and blocks. Each block displays a list of its trials with details about the trial configuration and links that generate raw data graphs (see Figure 10.10).

The performance history access is done through links to predefined graph configurations, letting the user choose the body side, data grouping level and the filter coefficient to be used when selecting the data from the database tables (see Figure 10.11).



Data Series	REGR Slope	REGR Intercept	REGR R2	REGR Count	REGR F
Series (2)	0.8304	16.3411	0.7388	13	2.7533
Series (2)	0.5815	19.2859	0.3071	6	1.0650

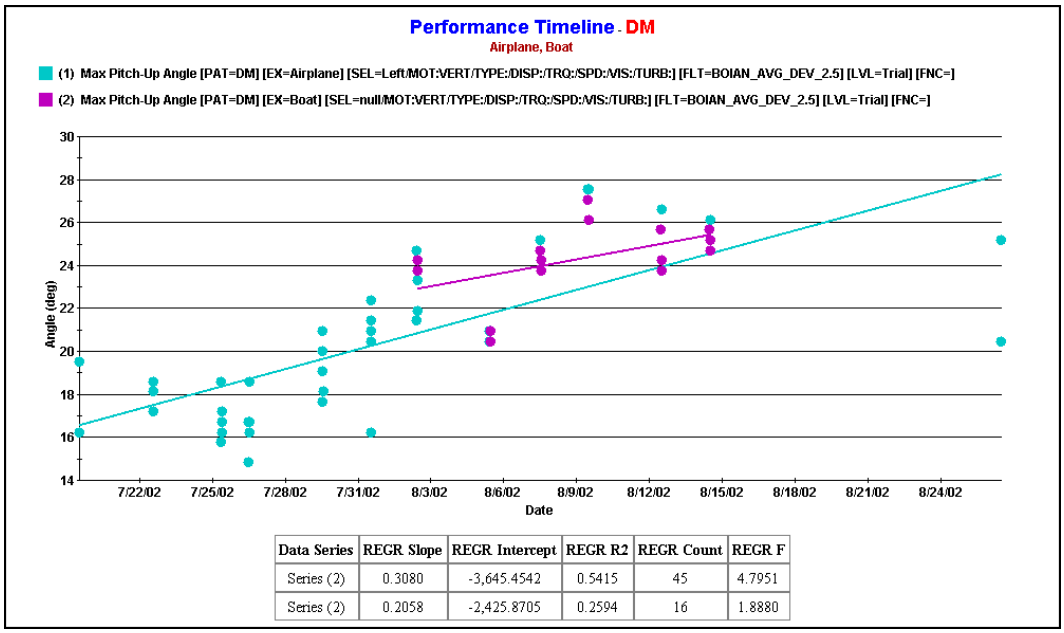
Figure 10.6: Ankle exercises patient history data. The graphed parameter is the PITCH–UP angle. The X–axis is formatted as an equidistant sequence of points. The data are grouped by days.

10.4.2 Java–based Interface

While the HTML interface provides the therapist with predefined graph choices, the Java implementation allows the therapist to build customized graphs through an interface that translates the user selections in data request format.

For raw data graphs, the interface uses drop down lists to select the study, patient, session block and trial to be graphed. A checkbox list is displayed for selecting the body part performance to be displayed. This list is customized for each exercise separately (see Figure 10.12).

The performance history interface provides the user with a larger number of options. Once the study, the patient, and the body side are selected, the therapist has to decide which exercises to look at, what variables of the exercises to consider, over what length of time to group the data and what function to use for that. This is a large number of choices that could become confusing and lead to errors. Furthermore, all these selection are part of the "Basic" configuration tab (Figure 10.13). The advanced configuration tab (Figure 10.14) brings choices like time interval, filtering, and exercise dependent

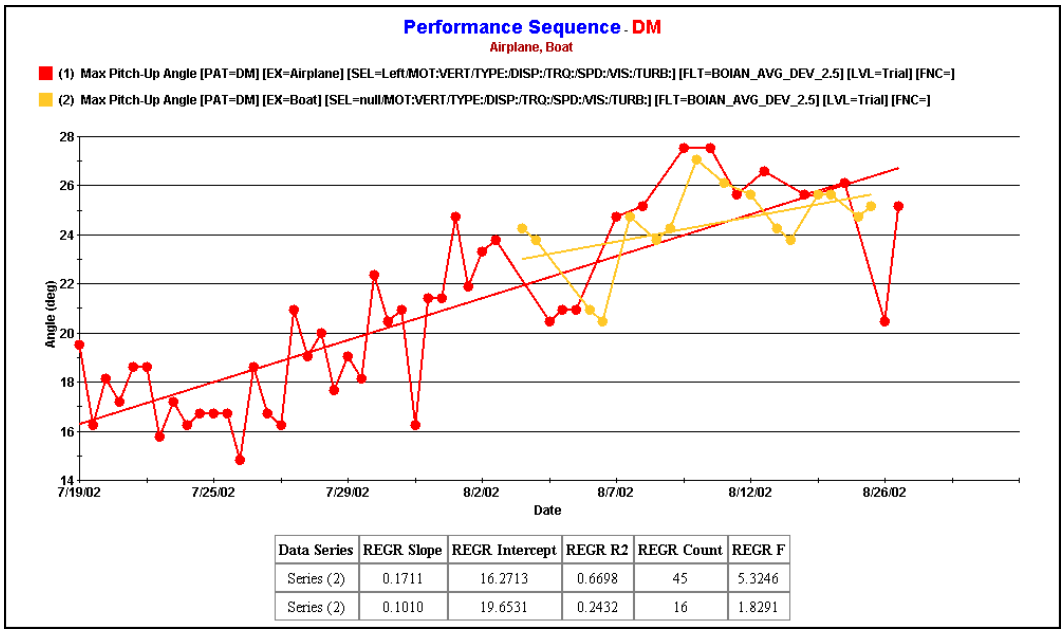


Data Series	REGR Slope	REGR Intercept	REGR R2	REGR Count	REGR F
Series (2)	0.3080	-3,645.4542	0.5415	45	4.7951
Series (2)	0.2058	-2,425.8705	0.2594	16	1.8880

Figure 10.7: Ankle exercises patient history data. The graphed parameter is the PITCH–UP angle. The X–axis is formatted as a time line. Data graphs individual trials.

parameters.

To simplify this process, it was necessary to define default values for the majority of the parameters so that the selection of three or four parameters generated a most likely needed configuration. This was done using a text dictionary format that stored configuration possibilities. Searching this dictionary by a few of the parameter values yields a predefined configuration that is guaranteed to offer proper results. Of course, these predefined selections can be modified, but it is necessary to have a valid configuration in place to avoid empty data responses.



Data Series	REGR Slope	REGR Intercept	REGR R2	REGR Count	REGR F
Series (2)	0.1711	16.2713	0.6698	45	5.3246
Series (2)	0.1010	19.6531	0.2432	16	1.8291

Figure 10.8: Ankle exercises patient history data. The graphed parameter is the PITCH–UP angle. The X–axis is formatted as a sequence of equidistant points. Data graphs individual trials.

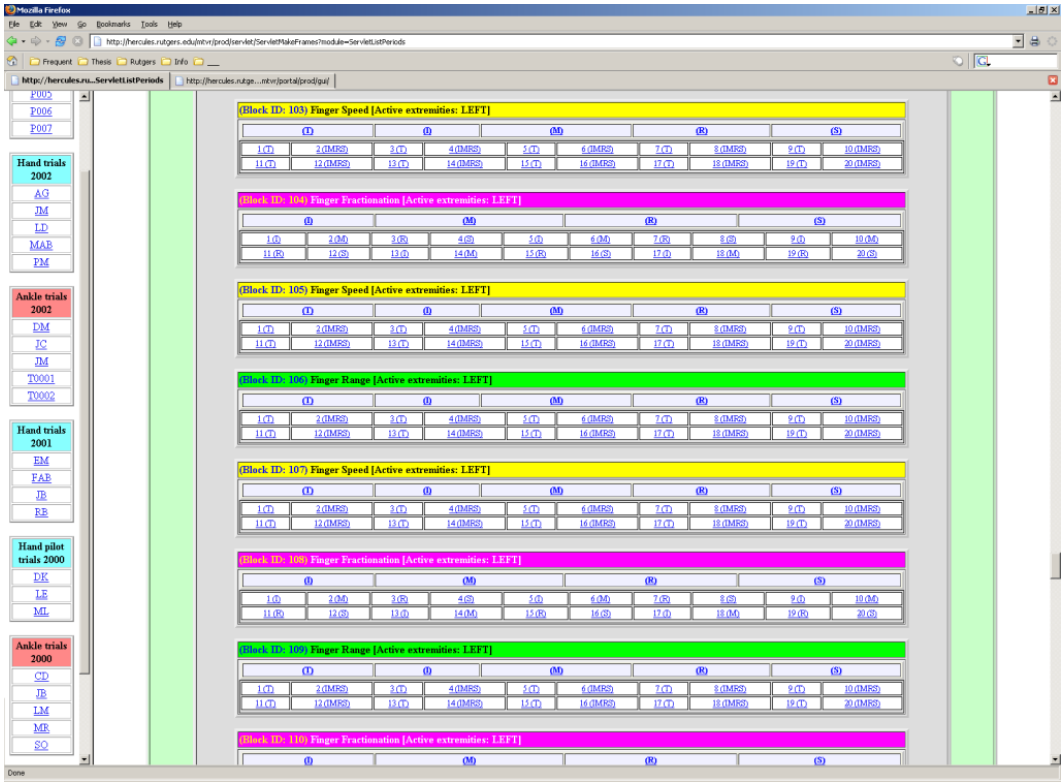


Figure 10.9: HTML portal. Hand exercises.

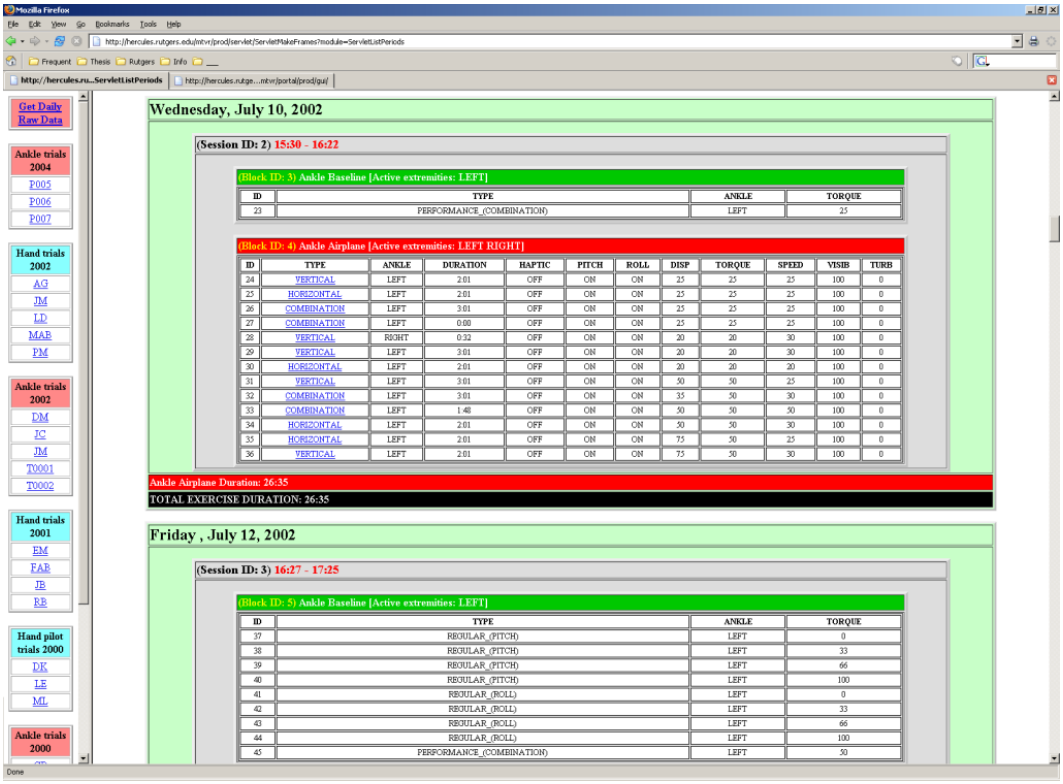


Figure 10.10: HTML portal. Ankle exercises.

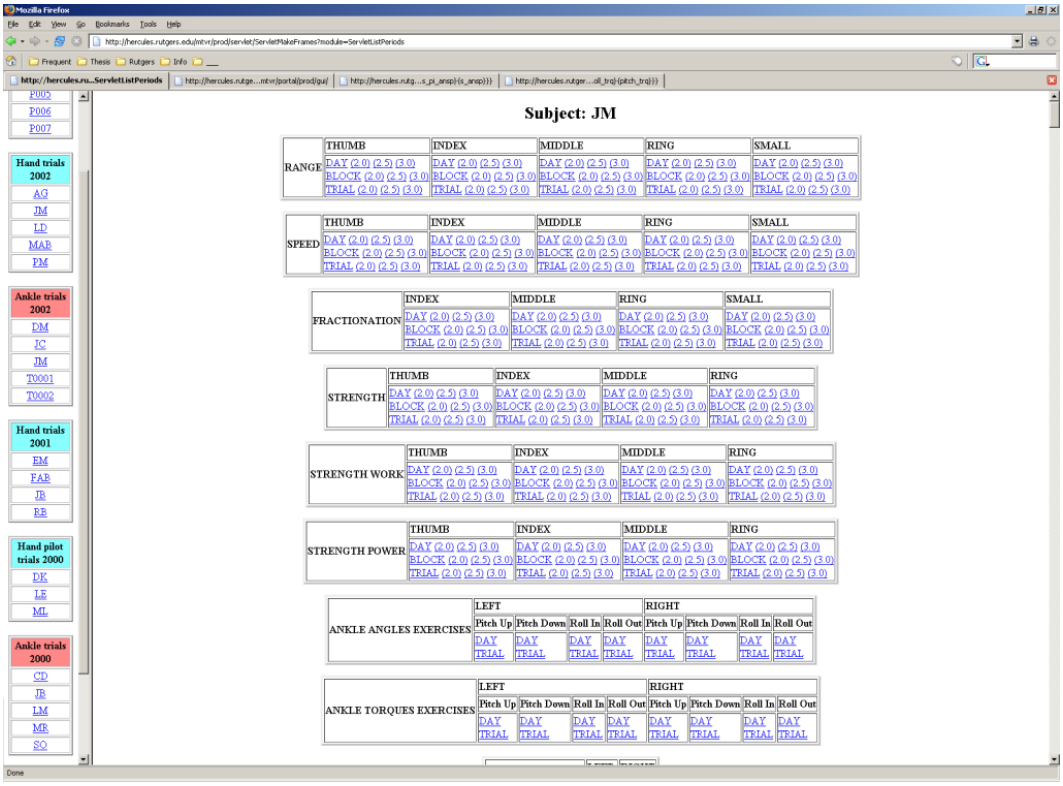


Figure 10.11: HTML portal. Performance history graphs links.

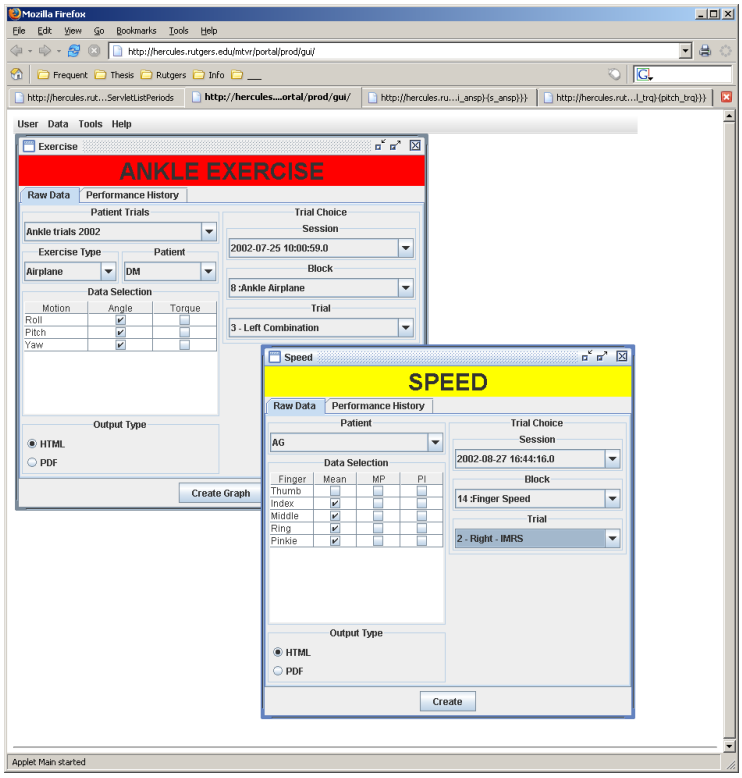


Figure 10.12: Java portal. Raw data selection interface.

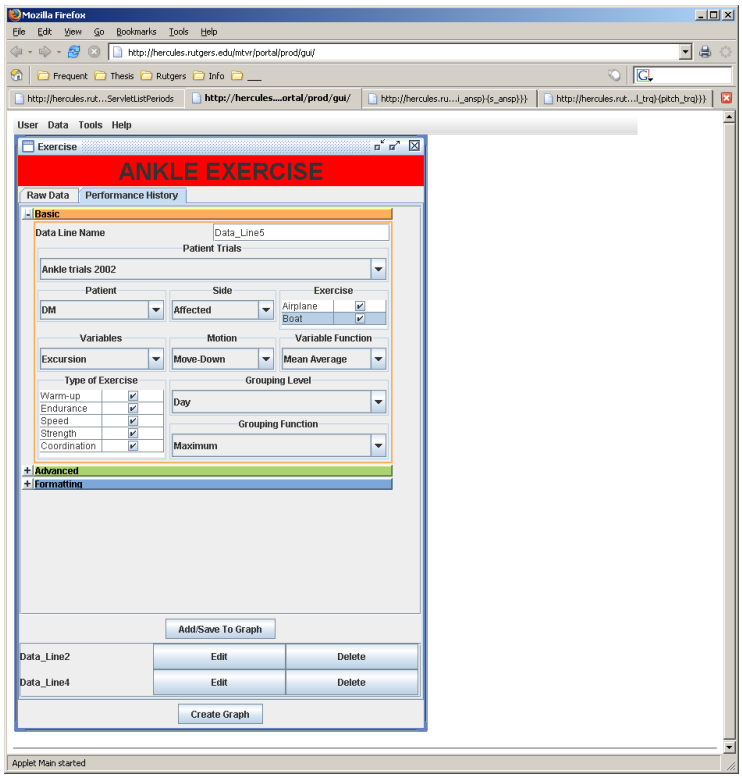


Figure 10.13: Java portal. Performance history basic data selection interface.

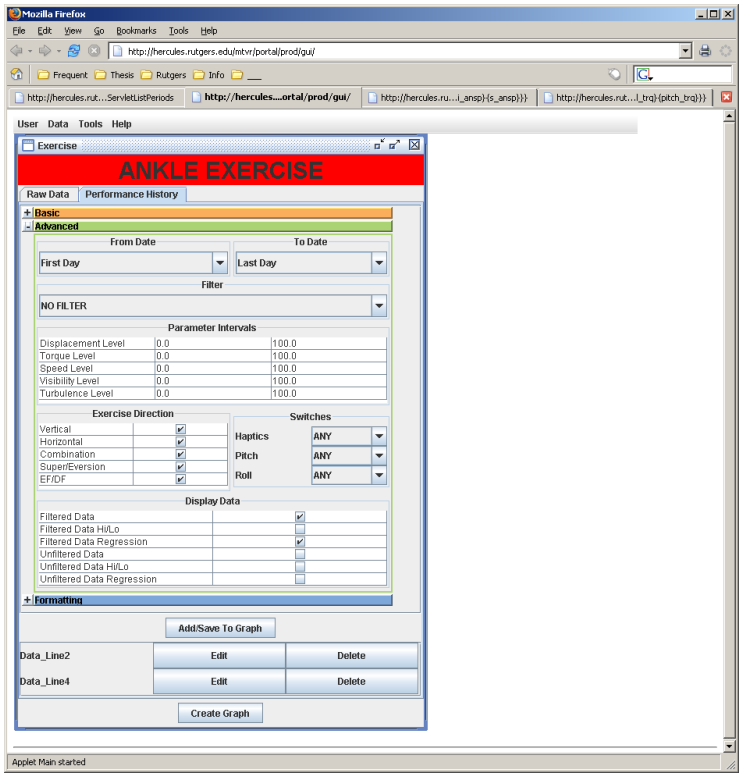


Figure 10.14: Java portal. Performance history advanced data selection interface.

Chapter 11

Conclusions and Future Work

11.1 Conclusions

The work presented in this thesis is a mobility simulator using two compact size Stewart platform robots. A virtual reality simulation of a street crossing has been developed and integrated with the simulator to be used as novel means for gait rehabilitation in patients post–stroke. A telerehabilitation framework has been designed and developed to integrate the Mobility Simulator along with other existing projects for hand [100, 12] and ankle rehabilitation [16]. The framework integrates the projects, with database storage of the collected clinical data, web portal to the database for post therapy data analysis. A monitoring service that allows monitoring and control of the rehabilitation session to be done by a remote therapist is under development.

11.1.1 Mobility Simulator

The proposed Mobility Simulator was designed to use two pneumatic Stewart platform robotic devices developed at Rutgers and attached to the user's feet. This design fits into the "foot platform walking devices" category defined by Hollerbach in [51]. The two Rutgers Mega–Ankle (RMA) robots are connected to and controlled by a prototype haptic control interface (HCI) containing an embedded Pentium board and customized electronics and pneumatics for air pressure control.

Servo control software has been developed to control the position and forces applied by two RMA robots that are simultaneously connected to the interface and active. The controller uses an adaptive PD strategy for position control and a PD strategy in the PWM air pressure control loop. The controller has been tuned to insure stability and position and force accuracy tests have been conducted.

A nine state walking simulation algorithm has been developed on top of the controller. The algorithm is designed to handle the switch between two major functioning

modes of the RMA robots. The fist mode requires one of the platforms to resist external disturbances during the stance phase of gait (i.e. when one foot supports the weight of the body). The second mode is the free motion mode used for the swinging foot. In this mode, the RMA robot is compensating for its own weight and for the forces applied by the user's foot, allowing it to move freely.

11.1.2 VR Rehabilitation Exercises

A virtual reality simulation of a street crossing has been developed and integrated with the Mobility Simulator (the street crossing exercise has been designed to train patients post–stroke in a frequent and challenging activities of daily living). Reaching the other side of the street in time is often a difficult task for a stroke survivor who has to focus on walking while coping with the distractions of a busy street. The simulation features intelligent vehicles that can be configured to act more or less aggressively by blowing the horns and encroaching on the crossing while stopped at the red light. Furthermore, the street surface can have patches of special haptic materials simulating ice, mud or gravel. All these parameters can be reconfigured by the therapist, and some can be changed during the run–time to adjust the exercise difficulty to the patient's performance level.

A second exercise has been developed by Kourtev [15] for the mobility simulator for training walking in general. The exercise displays a park scene in front of the patient. The patient is required to walk on the path and cope with various obstacles placed in front of him by the simulation. The shape of the path can be configured as well as the frequency and difficulty of the obstacles.

11.1.3 Haptic Rendering

A novel approach to rendering of ground haptic surfaces has been developed and implemented for the Mobility Simulator. The concepts of *haptic patch* and *haptic material* have been developed using as inspiration the already existing graphical materials and textures. The haptic materials describe the haptic properties of a surface in terms of stiffness, damping, breaking point, and friction. New materials can be obtained by blending several basic haptic materials using either the default linear transformation or

a different function of choice.

The haptic materials are applied on the ground surface in patches defined by a polygonal shape projected on the virtual ground surface. Two types of patches have been designed. Lining patches apply the material to the surface itself similar to how snow covers the ground. Filling patches are used to fill in cavities and create puddles of material.

11.1.4 Data Storage and Web Portal

A data storage and web–access framework has been developed to unify the analysis of the clinical data collected by four VR–based rehabilitation projects for hand, ankle and knee. The framework uses the Oracle RDBMS server for data storage and Java servlets and applets for web–based consultation and analysis of patients' data.

11.1.5 Remote Monitoring Service

A monitoring service has been designed, implemented and partially integrated with the existing VR rehabilitation projects. The service provides a geographically remote therapist to monitor and control a therapy session. The monitoring service features an overlay network of server nodes running on various sites. A VR exercise connects to such a server running on the local machine and streams the collected data to it. The server, multicasts the received data through the overlay network to the rest of the nodes. A remote therapist will connect to the network of nodes using a Java client, request the list of on–line patients, and visualize the activity of one or more chosen patients through mockups of the VR exercise run on the rehabilitation site. The mockup is kept in sync with the on–going VR simulation through a stream of data packets multicast from the rehabilitation site through the network of nodes.

11.1.6 Thesis Contributions

This thesis makes contributions in the following fields: haptic devices, virtual environments, haptic modeling, virtual rehabilitation, and telerehabilitation. The list below summarizes these contributions by field.

1. *Haptic devices*: Dual Stewart platform mobility simulator;
2. *Robotic control*: Simultaneous dual robot control in time–sharing control loops;
3. *Virtual environments*: Street crossing simulation integrated with the mobility simulator for rehabilitation of patients post–stroke;
4. *Haptic modeling*: Simulation of a person walking using two Rutgers Mega–Ankle robots;
5. *Telerehabilitation support*: Distributed telerehabilitation communication framework, including web–based data access.

11.2 Future Work

11.2.1 Mobility Simulator

The validation tests of the Mobility Simulator confirmed the initial expectation that the system can be used to simulate walking with small steps. It would be beneficial if the simulator supported the full stepping range of a person. Such a feature combined with the flexibility offered by the 6DOF RMA robots has the potential to greatly improve the rehabilitation therapy.

It is known that the workspace size of a Stewart platform robot is in direct relationship with the size of the fixed and mobile bases, as well as the stroke of the actuators. In order to achieve a workspace that could accommodate a full human step, the RMA robots would have to have a lower base about five time larger in diameter than they currently have and actuators with the same proportion larger stroke. Such an approach would make the system unusable given the lateral distance between the feet, as discussed in section 3.1.2. A simpler and less costly way of increasing the back–front size of the simulator's workspace is to place each RMA robot on a rail. This could be done by mounting low friction casters under the fixed base of the robot and attaching to the base a linear actuator with the fixed end attached to the wall (see Figure 11.1). The new actuator would move the RMA robot towards the front of the rail during the swing phase of the gait and backward during the stance. In this approach, the RMA robot

would only be used for rendering haptic effects and forces without having to slide the foot during stance. This would allow the foot to stay in the center of the workspace and hence have sufficient rotational freedom of motion at the beginning and of the swing.

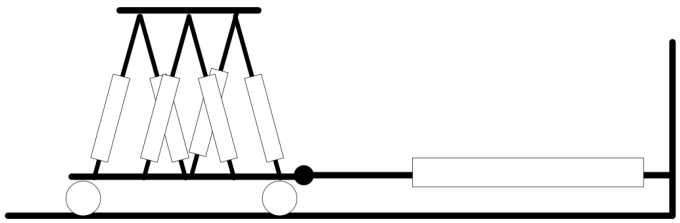


Figure 11.1: Extending the front–back workspace of the RMA platform by placing it on casters.

Another improvement that could be implemented is the use of larger diameter actuators and valves. Larger diameter actuators would increase the power output of the platform while the larger diameter valves would increase the air pressure control bandwidth as well as the mechanical bandwidth. At the time of this writing, the author is not aware of any commercially available frictionless pneumatic actuators larger than the actuators currently used by the RMA.

The same applies to the solenoid valves. Although there are large diameter pneumatic valves, they do not have a high enough ON/OFF frequency, which makes them unsuitable for our controller. A workaround this situation would be to double the number of valves in the controller interface from twenty–four pairs of intake/exhaust micro–valves to forty–eight. This would double the number of valves per air channel to two intake and two exhaust valves. Such a change would also require the increase of the main air input tubing so that the system is provided sufficient airflow.

11.2.2 VR Simulations and Haptic Rendering

The VR simulations developed for the Mobility Simulator could be improved by adding intelligent human avatars to the scene. In real life, crossing a street is often done in the company of other pedestrians. This raises new accommodation problems to the patient post–stroke who now has to also watch for the dynamic obstacles that are pedestrians.

Besides just adding avatars that cross the street along with the patient, these entities

could be added intelligent behavior and allow them to interact with the patient, assisting him or distracting him. For instance, one avatar could be programmed to cross the street at the same time with the patient but stay a few steps ahead of him. This avatar would walk at a speed that insures reaching the other sidewalk in time, thus pacing the patient and helping him walk in the necessary rhythm. Avatars can also be distracting on purpose by cutting off the patient's line of sight. On a higher level that requires the use of speech synthesizing software, the avatars could talk with each other and to the patient based on an AI routine embedded in the simulation. For instance, calling the patient name while the patient is in the middle of the street would constitute a serious distraction that would cause the patient to stop and look for the caller. Such a situation could prove dangerous in real life, while training for it in VR system would pose no hazards.

The ability of the Mobility Simulator for rendering stairs and small step walking could be used to integrate it with walking inside a house. The simulation would require the patient to go around the furniture and up and down the stairs to the second floor or basement.

11.2.3 Remote Monitoring Service

The monitoring service has been developed to solve one specific problem: provide the therapist with a reasonable interactivity that is not affected by the slow or unstable network between him and the patient. This work has been done to serve as foundation for further improvements some of which are already under development by Lewis.

One important improvement is integrating audio and video feeds between the therapist and patient with our current remote access implementation. This extension is currently under development and the system has been instrumented with a Java based conferencing tool that uses the overlay network to identify the conferencing parties and connect them.

Further work can also be done in improving the adaptability of the service to slow and unstable network connections. The current implementation provides this features at a reduced scale that relies on the small scale of the system. However, given the fast

developments in mobile computing, the monitoring service can be adapted to function on handheld devices, the technology for such a transfer already existing. Bringing mobile PDAs into the framework will require a more scalable algorithm for adapting to the network environment to provide satisfying quality of service (QoS) to the therapist. Bandwidth evaluation can be added to the service and used along with the round trip delay and jitter measurements to adjust the data streamed by the node local to the rehabilitation session. The data can be organized on hierarchical levels of real–time requirements. If the network conditions worsen, the less stringent data such as instantaneous patient performance can be sent with lower frequencies. For this purpose, the message format is being redesigned to support partial data transmission with minimal decoding overhead.

The entire remote monitoring console for the mobility simulator has to be developed and integrated with the existing hand and ankle in sitting application.

11.2.4 Patient Trials

Currently, the Mobility Simulator has been tested only on healthy individuals. Further studies on post–stroke and other patient populations are necessary. These tests will yield valuable data on the validity, ease of use, and medical efficacy of the system that will lead to further hardware and software improvements.

Appendix A

Servo Controller Software

A.1 Hardware API

The main goal of the hardware API was to hide from the control and simulation algorithms the usually cumbersome details necessary to write or read a signal from a port. The hardware entities modeled in software are presented below.

A.1.1 Port

A port is a pair of a memory addresses and a byte value. Although the operations for writing or reading ports are simple, in our case, multiple devices are controlled or read through the same port. This made it necessary to always store the value of the port in a variable, so that when sending a command through only a few bits of the port byte, the rest of the devices will receive the signal they already have, hence not changing their state.

A.1.2 Channel

A channel is the equivalent of an electrical wire. It is used to send commands to a hardware entity or to read the value of a hardware entity. A channel has attached a number of ports and a mask defining which bits of those ports are to be used by the channel. The value of the channel, its sign and bit resolution are also stored.

The channel structure is very useful for modeling a more complex aspect of reading the sensors through the A/D converter. The A/D I/O boards used in the controller interface can convert 16 channels but allow only two to be read at a time. This is accomplished with two selectors. Each selector must first be given the ID of the channel to be converted, then the software has to wait for the conversion to be completed reading a confirmation bit, and only after that read the value resulted from the conversion. To implement this procedure, the selector and confirmation bits have been

also implemented as channels, and the channel structure was added references to such corresponding entities. If a channel does not need to use a selector and a confirmation channel, then these references are set to NULL. In addition to the selector and confirmation channel references, the regular channel also stores the selection and expected confirmation values.

This data structure along with the read and write functions simplified significantly the A/D I/O operations by reducing them to single function calls (implemented as C macros for speed) that hide all the details presented above. Even more, using the C language definition upon declaration syntax, the hardware specifications were implemented in a tabular form that is very easy to follow and adjust.

A.1.3 Sensors and Valves

Using the channel structure, the modeling of sensors was done by attaching a channel, a value and a transform polynomial to each sensor entity. The value of a sensor would be calculated by applying the transform polynomial to the value of the channel attached to the sensor.

The solenoid valves were also modeled using the channel structure. A valve is defined by a reference to the channel that controls it and by a state variable and a counter variable marking the time that the valve has to be kept open.

A.1.4 Air Channel

An air channel is the software model of the air hose/cylinder chamber assembly. It is an entity used for pressure control, hence its implementation is more complex. An air channel is defined by an intake and exhaust valve, a pressure sensor, and the necessary pressure control coefficients. Each air channel had its own independent control variable to provide full configuration flexibility. Such a detailed implementation was necessary to address the differences in hardware caused by extended use.

A.1.5 Double–Acting Cylinders

Using the structures defined above, the double acting cylinder structure was defined by putting together two air channels, one for the lower chamber and one for the upper, along with a linear potentiometer sensor and the position and force control coefficients. Again, each cylinder was assigned independent coefficients for maximum flexibility.

The double acting cylinder was applicable for both the regular and the Mega ankle robot. For the control of a haptic glove, a similar but simpler structure would have to be defined to model a single acting cylinder.

A.1.6 Stewart Platform

The Stewart platform structure puts together all the structures defined above. It consists of six double acting cylinder structure references, one force sensor reference, and all the kinematic parameters of the robot. Both models of the Rutgers Ankle robot use the same structure since they differ only in size.

A.1.7 Configuration Files

The core of the hardware API is the channel structure. Using it, rearranging the software to follow the hardware switch configurations can be done by just reassigning channels to sensor and valve entities as necessary. Due to initial hardware failures, it was necessary to be able to easily rewire an unused pressure sensor to an air channel whose sensor was faulty, and change the software to handle it.

To avoid having to change the software so often and lose track of what was working and what was not, a configuration file format was implemented to store as many of the settings as possible outside the source code. A configuration file has a simple format of $parameter = value$.

The configuration files make it very easy to add new devices to the system by just adding a new file storing that device's parameters. For instance, each Stewart platform comes with such a configuration file storing the kinematic parameters, inertia

tensors, cylinder control gains, force sensor transformation matrix for converting electrical signals to forces, and torques, and signal filter coefficients. Similarly, each control box has its own configuration file specifying the hardware addresses and connections (in case they have been modified), pressure sensor filter coefficients, and control loop frequencies.

A.2 Controller Commands

To increase the interactivity of the software and the modularity of the implementation, it was necessary to design it as a simple command line interpreter. While not having all the special features a regular operating system interpreter comes with, the controller allowed a strict but easily extensible set of commands. Each command was used to implement testing or simulation procedures as needed without affecting the rest of the code. The complexity of the commands ranged from simple ON/OFF signals to the valves to more complex visual interfaces for full hardware testing or on–screen oscilloscopes for watching the control signals. A few of the most used commands are presented below.

A.2.1 Hardware Test

The hardware testing command was implemented for low level hardware debugging by the company that manufactured the prototype following the Rutgers design (Delaware Technologies, Mount Laurel, NJ). The command presented a full view of the hardware status and allows the manipulation of valves and hardware switches, while showing the voltages read from the sensors (Figure A.1).

A.2.2 Pressure Control Tuning

The pressure control tuning command allows the change of control coefficients at run–time while showing the desired and measured pressure in cylinder chambers using a software oscilloscope. The command supports either manual signal manipulation or

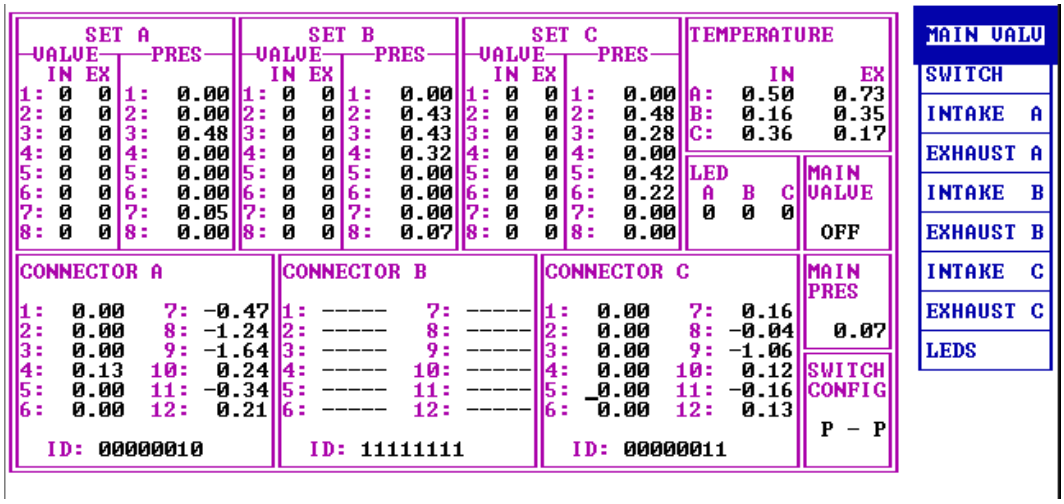


Figure A.1: Hardware test interface.

it can output sinusoid, square or triangular waves of a specified frequency and amplitude (Figure A.2).

A.2.3 Cylinder Position Control Tuning

Similar to the pressure control command the position control command, allows run–time control coefficient changes while showing the desired and measured displacement. For better insight into the functioning of the controller, the desired and measured pressures in the cylinders chambers are also graphed (Figure A.3).

A.2.4 Full Platform Testing

Although the two commands presented above were sufficient to achieve a reasonable tune–up, for more complex platform behaviors, it was necessary to watch simultaneously any number of system values, and allow changes to each parameter individually. This was achieved by displaying a list of the graph–able values, and allow the adjustment of scaling and offset coefficients necessary to make the values visible in comparison with the other lines being graphed (Figure A.4). Given the very large number of parameters involved, a set of global parameters was defined whose change would cause their respective individual parameters to take the same value. For instance, instead of changing all the pressure proportional gains separately, it was sufficient to properly

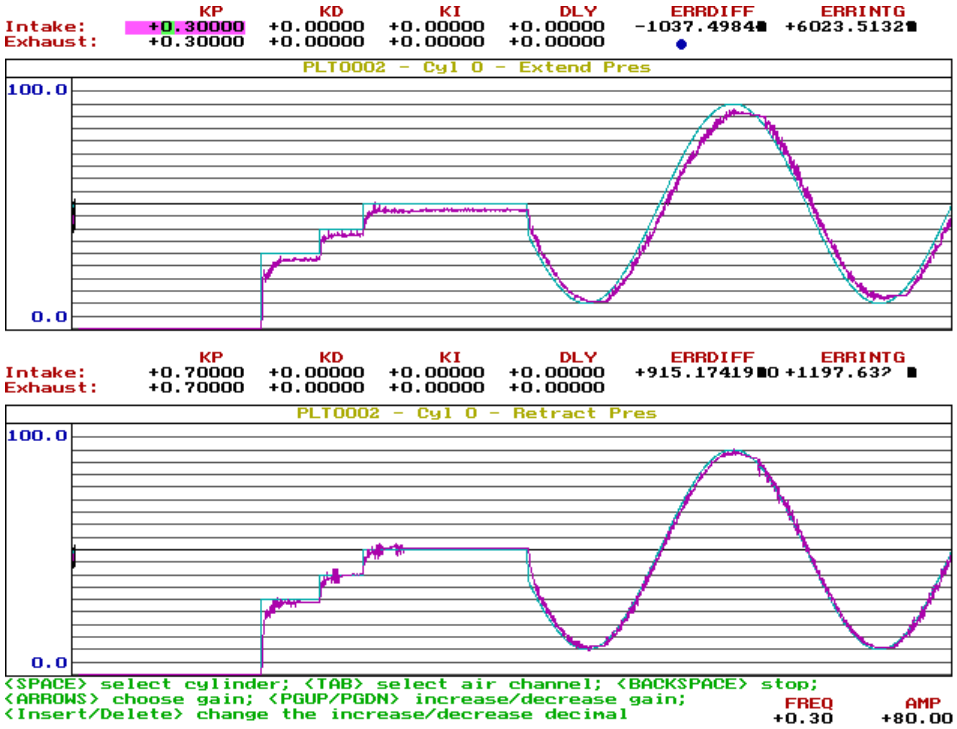


Figure A.2: Pressure control tuning interface.

configure the global pressure gain. After the global gain was set, fine–tuning was also possible by changing each gain individually.

A.2.5 Simulation Commands

The gait simulation or the sitting system platform behaviors were implemented as separate commands. Hence, to use the system with one or another it was sufficient to start the controller and issue the appropriate command. In a future version, this will be done transparently by the VR simulation, without being necessary to actually type in the command at the controller terminal.

A.2.6 Utility Commands

Sixty–seven commands have been implemented during the development process. Among these, approximately 60 are used for testing and configuring the hardware. The most used commands are: valve individual manipulation, forward kinematics testing, pressure sensors calibration, hardware switch state changes, serial communication testing,

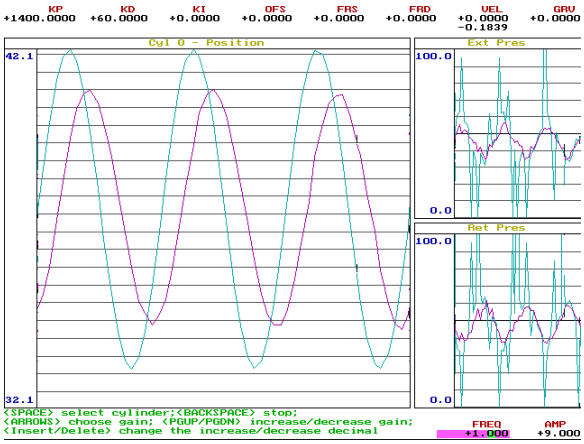


Figure A.3: Cylinder position control tuning interface.

Stewart platform position specification, timer accuracy testing, and a usage manual command. The Stewart platform orientation command is used to specify the position of the platforms (see 3.12), so that the reference axes were accordingly modified.

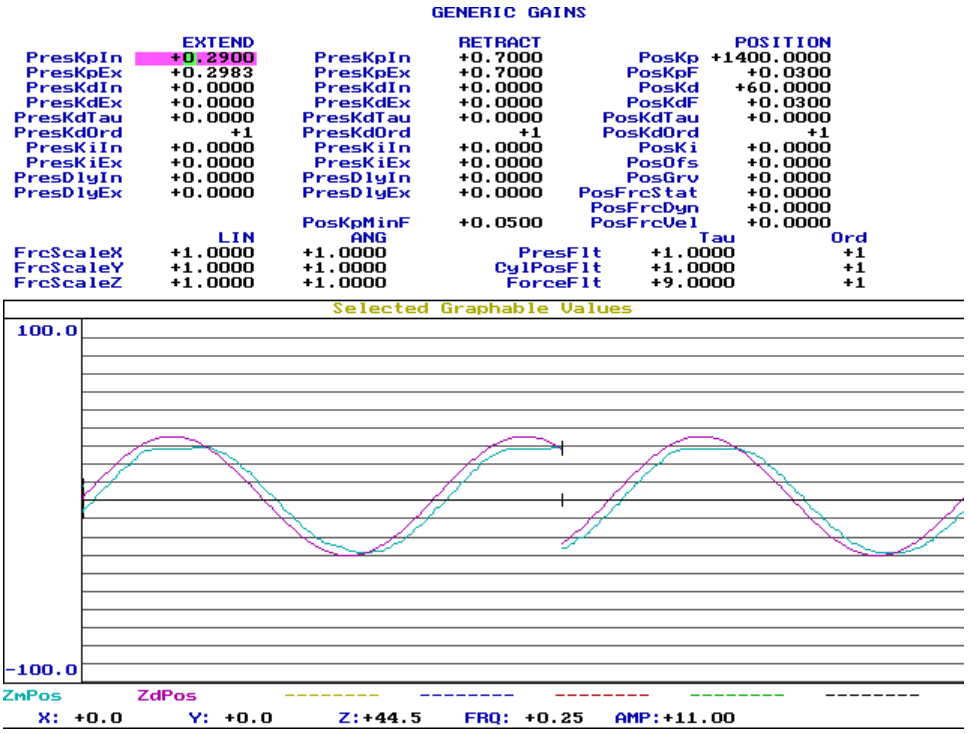


Figure A.4: Full platform testing interface.

Appendix B

Hand and Ankle VR Exercises

B.1 Post–Stroke Hand Rehabilitation Exercises

The post–stroke hand rehabilitation set of exercises has been design to improve the control over the fingers by repeating two basic movements under various conditions. The exercises were done using a CyberGlove (Immersion Co.) or a Rutgers Master II Haptic Glove. All the exercises were to be executed in a discrete manner, each motion constituting a trial. This was necessary for easier post–therapy analysis of the data. The discrete aspect of the exercises required a game–like situation that engaged the patient and lasted no longer than approximately 30 seconds at a time.

The first exercise targeted the range of motion of the fingers. The patient was asked to open the hand fully and then close it completely. The simulation presented the user with a virtual hand whose finger motion was mapped to the patient's hand motion. As the fingers covered a larger range of motion, a picture covered by vertical strips of "dirty pixels" was cleared. The vertical screen zones were moved away proportionally with the ratio of the targeted range that the patient achieved (Figure B.1) [1, 2]. The pictures were chosen from a large selection of interesting images.

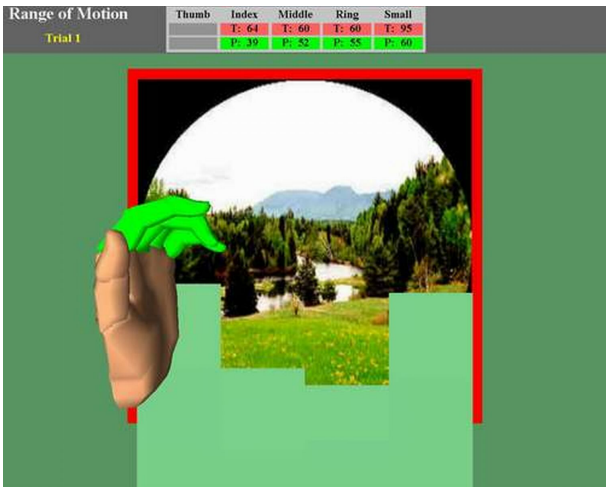


Figure B.1: Finger range exercise [1, 2].

The second exercise asked the patient to close the fingers as fast as possible. The game showed the virtual hand with a virtual butterfly flying in circles above it (Figure B.2). If the patient moved the fingers fast enough to achieve the desired velocity target, the butterfly was programmed to fly away, as if scared by the fingers. Otherwise, it would ignore the user's motion and continue its circular flying pattern. This exercise proved to be very engaging, the subjects getting quite involved into hushing away the virtual butterfly. A subtle aspect of this exercise is the connection between the performance and the goal of the game. The idea of actually catching the butterfly is actually the first that comes to mind when seeing the exercise. However, implementing a realistic catch requires the detection of the patient's fingers end of motion. Such detection was tried in the initial phase of the system, but proven unreliable due to the atypical motions of the affected fingers.

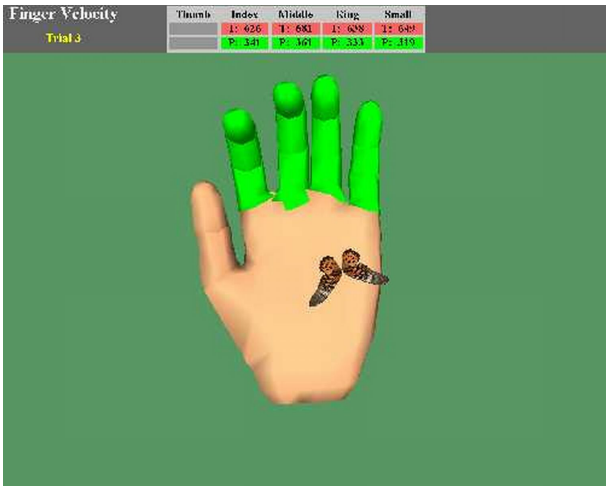


Figure B.2: Finger velocity exercise [1, 2].

The third exercise required to patient to move each finger independent of the others. This motion resembled somewhat playing a piano, so the simulation was designed to present the hand above a miniature piano keyboard. A good motion was causing the pressed virtual key to play the appropriate sound and turn green. An incorrect movement, involving several fingers at a time, caused the piano keys corresponding to the non–active fingers to turn red. Figure B.3 presents a screen capture of the exercise.

The last exercise trained the patient's finger endurance (mechanical force output).

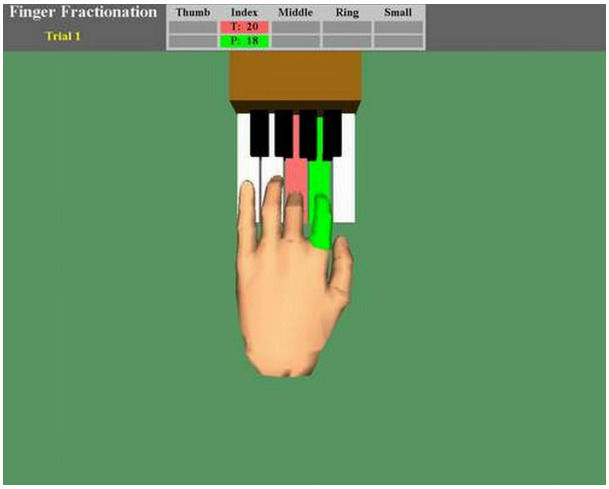


Figure B.3: Finger fractionation exercise [1, 2].

The exercise was similar with the range of motion exercise, except it was done against forces. The force to the fingers (only four of them: thumb, index, middle and ring) was provided by the RMII–ND haptic Glove developed at Rutgers. The reduced range of motion offered by the glove and the difficulty of the exercise, did not allow for too much flexibility in the design of the game. The chosen game shoed a virtual replica of the Rutgers Master glove on the virtual hand, with the pistons filling with color as the real pistons were being closed against a constant force (Figure B.4). Since force was constant, it was easy to compute the mechanical work performed by each finger, something not available in traditional rehabilitation devices.

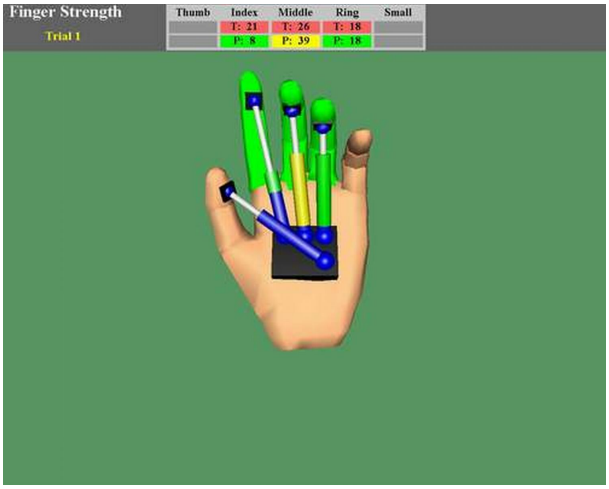


Figure B.4: Finger strength exercise [1, 2].

All hand exercises displayed a numerical performance feedback updated in real time at the top of the screen. The numbers showed the desired target in red and the patient's performance in green.

B.2 Post–Stroke Ankle Rehabilitation Exercises

The ankle exercises were designed to improve the ankle function through repeated pitch and roll motions against controlled resistance. The haptic device involved was the Rutgers Ankle robot [14], a smaller version of the robots used by the mobility simulator. Latonio, Burdea and Deutsch developed an initial airplane piloting VR simulation exercise controlled by the orientation of the user's ankle, while the patient was in sitting. The Rutgers Ankle robot was used as a foot joystick, and its orientation was mapped to a virtual airplane flying at constant speed through the virtual scene. The patients were engaged into flexing the ankle, by asking them to fly the airplane through hoops placed in the path of the airplane (Figure B.5). The displacement of the hoop from the straight path required the patient to orient the airplane toward it by flexing the ankle. The speed of the airplane, the placement, of the hoops and the duration of the flight were controlled to vary the exercise difficulty. Later, Boian and Deutsch developed the second version of the exercise adding haptic effects and more flexibility in positioning the hoops in space. The haptic effects were designed to increase the realism of the scene and make the tasks more challenging [14]. The new hoop placement implementation was used to better customize the motions the patient had to execute. For example, a patient that has difficulty flexing the ankle upward would be exercising on a configuration where most of the hoops are positioned at higher elevations than the previous ones, hence requiring the patient to execute the dorsiflexion motion more often.

Lee [16] later developed a boat sailing exercise on a wavy sea sailing through a sequence a buoys (Figure B.6). Although similar to the airplane exercise, the boat simulation imposed a constraint on the path between two targets. Where the airplane could fly between two hoops on any path, the boat had to be kept on water. This was particularly difficult when the patient had to switch the orientation of the boat

Figure B.5: Ankle airplane exercise [16].

on the top of a wave. This is a perfect example of how designing an exercise for a specific task is affected by the need to keep it realistic. The difficulty of the boat exercise is increased exclusively by the need to keep the boat behavior realistic. The need for realism also limits configurability of the target positioning (corresponding to the ankle motions exercised by the patient). The airplane targets can be placed in a stair–like arrangement for ankle pitch–up training. The boat exercise does not provide this feature because it would cause the virtual sea to look unnatural.

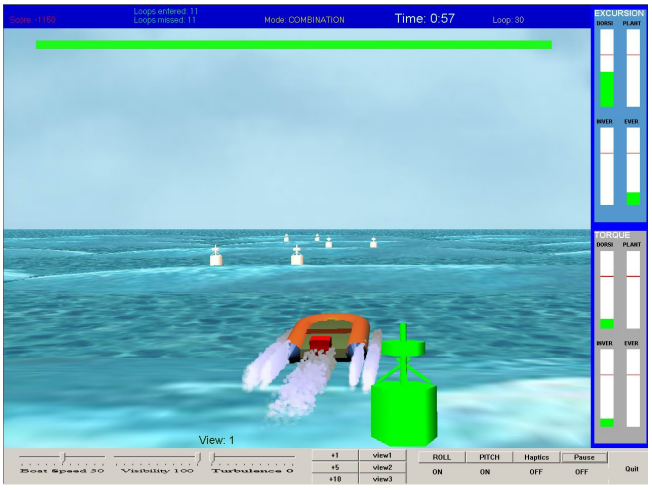


Figure B.6: Ankle boat exercise [16].

Appendix C

Monitoring Server Services

The following services are implemented in the current version of the monitoring server: dispatcher service, ping service, streaming service, registration service, command service, database service registration service, initialization service, multicast service, and multicast tree service.

C.1 Initialization Service

The initialization service is the first active service when the server starts. It is responsible for reading the configuration parameters and starting the registering procedure. If the server node is a hub node (its IP address and ports match the given hub IP and port), then the initialization service only has to start the rest of the services and then wait for the termination signal. Upon termination, the initialization stops all the other services and exits.

C.2 Registration Service

The registration procedure allows a new node that comes alive to make itself known to all the other active nodes in the system and create connections to all of them. Upon activation, a non–hub node opens a socket [109] to the hub node and sends a registration message containing information about its TCP welcome port, and UDP send and receive ports. The hub node receives the registration message and connects back to the node sending its own registration message. The returned registration message has appended a list the information of all the other known nodes in the network. The connecting node parses this list, opens sockets to all the other nodes, and sends the registration message to them. Then it waits for the other nodes to connect back to it, hence becoming a fully connected network.

The registration service implements the above protocol and manages the list of

known nodes of each server. It is also responsible for clearing out the "dead" nodes from the list. A dead node is marked by either a failure in communication or by prolonged silence detected by the ping service.

C.3 Dispatcher Service

The dispatcher service reads the messages from the sockets, sends and posts them in the queue of the appropriate service. If the destination service ID is not given, the message is posted in the queue of the least–busy service that matches the specified destination service type. The server spawns one dispatcher service for each five nodes in the network.

The dispatcher service is sensitive to the flags of each message. If the message is flagged by the MULTICAST flag, besides posting it in the destination's service queue, it also adds it to the multicast service queue to be forwarded. An exception to this rule is the REALTIME flag, which force the dispatcher service to run the multicasting routine itself, thus avoiding the delay of queuing the message. If the message is flagged as STORE, it means that the data must be stored on the local node in the structures belonging to the source node of the message. This is a generic feature of the server that currently applies only to the patient data streamed by the VR exercises.

C.4 Ping Service

The ping service measures the round trip delay and jitter between the local node and all the other nodes. The operation is done with configurable frequency. If a node fails to respond to the ping message with an interval of 60 seconds, the node is marked as "dead" and the registration service will remove it upon the next parse of the node list.

C.5 Streaming Service

The streaming service answers data requests from a monitor node. Given a patient name, the streaming service adds it along with the requester node to the list of streaming requests. Each node stores a structure with the newest data collected for each patient

active in the system. For each streaming request, the service sends this with a frequency of 30 Hz.

C.6 Command Service

The command service is responsible for servicing special requests from the monitor client. Such requests include configuration parameter changes to the local VR simulation, either off–line in the configuration files or while the simulation is running. This service is also used for synchronizing the access to the exercise configuration files. These files need to be protected from remote access while the exercise is running. Hence, the service supports requests for locking and unlocking of files. The locks are only internal to the server. The accessing clients or exercises are required to first acquire the lock and only then proceed to reading the files.

C.7 Database Service

The database service plays a dual role depending on the node that is running on. If the service is running on the database node, then it is responsible for receiving data archives and storing them in the local repository, then start the database writer that moves the information form files to the Oracle database.

If the service is running on a regular node, it schedules daily parses of the patient data (if it exists). If there is any new data, the service stores it in an archive and sends it to the database node.

C.8 Multicast Tree Service

The multicast tree service is activated periodically to rebuild the multicast tree based on the latest changes in the network of servers. The multicast tree is built taking into account the round trip delay between each pair of nodes and the bandwidth of each node. This information is sent periodically by each node in the overlay network. Straight forward shortest graph path algorithms do not take into account the overloading of each node, hence causing bottlenecks in the overlay network. If the bandwidth is taken

into consideration, then we face a known NP–hard problem that has not been solved yet. Current approaches relay on various heuristics to approximate a result. Research in this direction has been done among others by Brosh in [19], Bauer in [7], and Banerjee in [6].

In our case, the system has a small footprint on the network requiring a maximum of 56 Kbs. In the current university and clinic setups, the available network connections offer a much higher bandwidth than that. In addition, our system is not intended to host hundreds of nodes, but rather one or two dozen at most. Under these conditions, the bandwidth requirement has to be addressed only to the cases for those nodes (usually local to the rehabilitation sites) that are connected over a low bandwidth connection such as a modem. For these cases, the system sets the low bandwidth node as a leaf node to the closest node in terms of data transmission delay. The same node will also server as the single child of the low bandwidth node when it acts as a data source. Once these nodes have been assigned, the rest of the multicast tree is built using the classical Dijkstra shortest path algorithm [31].

C.8.1 Attaching New Nodes to the Multicast Tree

Since the multicast tree service is activated only periodically, the nodes that connect to the overlay network in between will not be included in an optimal tree. Initially these nodes are attached children of their closest high bandwidth node. The multicast tree of the new node is built by the node immediately after startup.

C.8.2 Pushing a New Multicast Tree

The approach taken by this implementation provides that periodically, given changes in the overlay network, each node will push a new multicast tree whose root it is, through to the rest of the nodes. This can potentially raise a problem if a message multicast based on the previous tree arrives to a destination node after that tree has been replaced. In such a case, it is possible that the message will not reach all nodes. Given the size and dynamics of our application, such situations are rare. Furthermore, all the multicast messages currently implemented are patient data messages. Since they

are sent with high frequency, losing some of these messages will not be a problem. As a partial solution to this situation, each node could cache a few versions of the multicast tree and use it when such a message occurs.

C.9 Multicast Service

The multicast service is responsible is responsible for forwarding the messages in its queue to the child nodes of the local node as defined by the tree rooted by the source node of each message. As described in the previous section each node stores the multicast trees of the other nodes.

Appendix D

Database Design

D.1 Database Users

The database supports four types of users, defined in Oracle terms by four different roles: “root”, “staff”, “therapist” and “patient”. These roles provide table level access control and are used to restrict the type of access each has over the patient data.

D.1.1 Root

The root role is applied to only one user that is the equivalent of the UNIX operating system root user. The root is the administrator of the telerehabilitation database and has full access to all the data in the system. The root user account also stores common access data readable (with restrictions) by all users.

D.1.2 Staff

The staff user is the next more privileged role in terms of data access and administration. In the stage of the system, staff users are usually researchers or data managers involved in the project that might need more than read–access to the patients’ data they are studying. A staff user has write access on the root tables concerned with the creation of new users and their assignment studies. They also have write access on all the patient data belonging to studies common to the accessing staff member. This access is generally needed to fix data anomalies (e.g. mark trials invalidated by a malfunction), filter the performance data in order to obtain more meaningful graphs, and adjust the patient’s clinical information based on their progress. A staff user also has the ability of creating new “therapist” and “patient” accounts.

D.1.3 Therapist

The therapist role is designed for physicians and therapists directly involved with the patients and the rehabilitation sessions. A therapist user is given write access to the patients' data limited to adding notes attached to a therapy session. They have full read access to the patient data belonging to a common study. The therapist is given the ability to create a new user, but that is done through a root security level stored procedure, hence not giving them full access to a patient's data account.

D.1.4 Patient

The patient role is the most restrictive role in the database. It is given read access only to the common tables stored in the root schema. The patient schemas store all the data generated during the rehabilitation sessions. In an initial version of the system, when the data were reduced in size, the separation of the data among patient accounts was not necessary, which made the overall database design significantly simpler. However, given the gigabytes of data currently stored, this separation makes not only sense logically in terms of data entities, but most importantly, it provides a performance boost.

D.2 Database Tables

In the current version of the system, only the root and patient schemas contain tables, the staff and therapist roles being necessary exclusively for access control purposes.

D.2.1 Root Tables

The root schema contains only administrative tables storing information about the clinical data rather then data itself. These tables contain information about the data entities supported by the database. The entities described by the root tables are: users, projects (modules), studies, exercises, trial run–time event types, hardware devices and patient questionnaires.

The user information is stored in the *USRS* table. This table keeps track of all the users in the telerehabilitation systems. It stores the user's ID (not the system or

database ID but an application specific ID), the user name and an alias. For security purposes, no user identification information is stored in the database. Being a prototype system, the focus is on functionality, which may leave security holes in the implementation. Storing the patient identification on an outside medium prevents any undesired privacy breach. In addition, the user name is chosen to reveal as little as possible about the identity of the patient. The user name and the alias of a user are generally identical except when the desired username overlaps with an existing database username outside the scope of out application. In this case the user name is given a unique value, while the alias is kept as chosen for convenience. The aliases are unique within the web application.

The modules supported by the database are stored in the table *MODS*. A module consists of a collection of data structures required to store the information generated by a rehabilitation project. For instance, there are currently four modules in the database each addressing a separate battery of exercises. The four modules are:

1. Post–stroke hand rehabilitation;
2. Post–surgery hand rehabilitation;
3. Post–stroke ankle rehabilitation;
4. Post–stroke gait rehabilitation.

The table stores a module ID along with a name and a description. The ID is used as foreign key for implementing the relations between the patient session data and the module is belongs to.

The information about the studies is stored in table *STUS*. A study refers to a period of time during which a patient undergoes therapy in one or more of the modules. A study is defined by an ID, a name, an alias, a description, a start date and an end date. Each patient data record has a foreign key column referencing the study ID. Using this ID the access to the data is implemented at row level. A therapist user can see only data belonging to patients in the studies to which the therapist is associated. The alias is a short version of the study's name and it is provided only for convenience.

The trial run–time event types are stored in the table *EVTS*. Events are used to mark the time when the patient or the therapist executed an action, whose occurrence is necessary for interpreting the data correctly. Examples of such events are a change in the exercise parameters at runtime such as modifying the speed of the airplane or boat in the ankle exercises, or switching from the hand opening phase to the actual exercise in the case of the hand post–stroke rehabilitation exercises.

D.2.2 Patient Tables

The data stored in the patient tables can be classified into three categories: data about the patient's condition, data recorded during the therapy sessions on the system, and filtering data necessary to better describe statistically the therapy data. Figure D.1 presents the patient account schema.

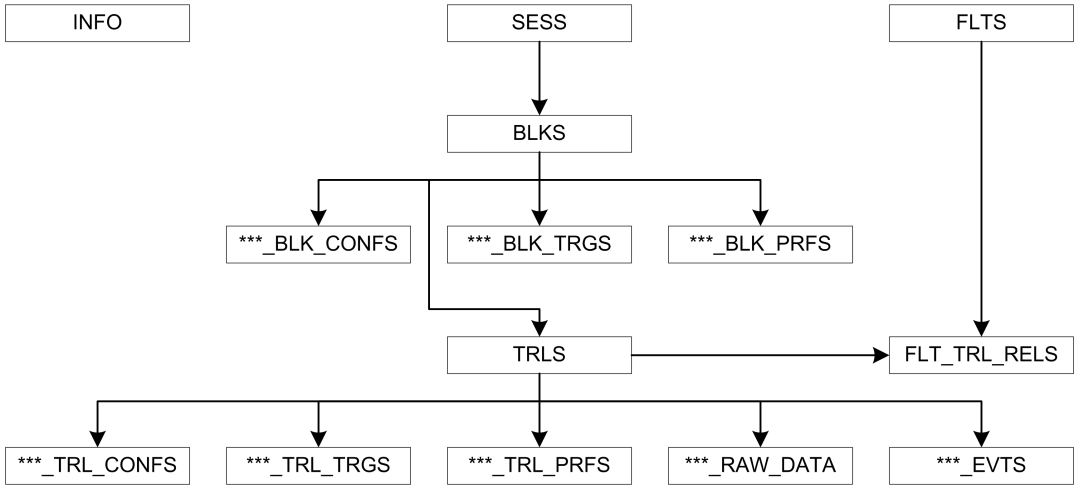


Figure D.1: Patient account schema.

The first data category listed above, consists of static information with regard to the patient's illness, the side affected, age, and other relevant information for the therapist and physician consulting the records. These data are kept to a minimum necessary because of privacy considerations. Because it is not too structured, this information is stored in a single table, with a single record.

The therapy data are the largest data category in the system. The storage is organized to reflect the protocol used during the therapy. The tables are grouped in *trunk*

tables and *exercise specific* tables. The trunk tables are common to all the data entries and store the common data that characterize a session, block or trial, regardless of the exercises. The rest of the tables contain entries specific to every exercise in the system.

The trunk tables are named *SESS*, *BLKS*, *TRLS*, for sessions, blocks and trials respectively. Every table contains and ID field, which is used as foreign key by the lower level or exercises specific tables. Other common attributes are:

- START_DT, END_DT - the start and ending date of the entity (i.e. session block or trial);
- CLOSED - marks whether the entity has been successfully stored in the database;
- VALID - necessary to cancel out entities that were unexpectedly terminated during therapy;
- Foreign keys of the tables higher in the hierarchy (i.e. the TRLS table contains references to the corresponding block, and the block contains references to the corresponding session;
- Foreign keys relating the record to the appropriate study. This is an out of schema relationship since the *STUS* table is stored in the ROOT's schema. The reference to the study is necessary to provide row level access control.

In addition to the entries above, the blocks table stores the ID of the exercise of its child trials, and information about the body parts active during the block. This is stored in bit buffer, with each bit characterizing a body part. The bit encoding is shown in Table D.1. The body parts covered are limited to the necessities of the current phase system. The trial entries also have this attribute describing the parts active during the trial. The block level attribute is a logical OR operation over the attributes of the child trials.

The trial is the lowest level entity in the trunk tables. Trials are the only entities that have raw data tables and event tables attached to them. Given the size of these tables sometimes it is more convenient to skip storing it until a later time. To mark

Table D.1: Body parts bit encoding.

Bit	Hand System	Ankle System	Gait System
0	Left hand	Left ankle	Left leg
1	Right hand	Right ankle	Right leg
2	Thumb	–	–
3	Index	–	–
4	Middle	–	–
5	Ring	–	–
6	Pinky	–	–

the status of this, the TRLS table provides the RAW_DATA_SYNC and EVTS_SYNC attributes.

The exercise specific data are stored in five types of tables: configuration tables, target tables, performance tables, raw data tables and event tables. With the exception of the last two, all the other categories are applied to both block and trial level. For each exercise in the system, the database is added a set of such tables. The configuration tables store exercise parameters not directly related to performance measurements, such as planned trial duration, or trial difficulty level. The target tables store numerical values directly related to the performance evaluation. For instance, the patient may be required to achieve a certain finger flexion speed in order to succeed in a trial. The performance tables contain the same columns as the target tables making it easy to compare the patient performance in relation with the targets set by the therapist manually or by the system automatically.

The raw data table stores the high frequency sampled data necessary to evaluate the patient's performance. All the measurements can be reproduced for these data, and hence the rest of the tables are redundant, yet necessary for performance optimization.

The event tables store the time, type and data about the events that occur during the trial. Such events are pauses in exercise (marked by putting the simulation in a pause state), changes in exercise parameters, or exercise phases such as the moment a target was achieved or failed.

The last category of patient data is the filtering data. The performance of stroke patients can vary significantly from day to day causing either very low or very high measurements. In order to evaluate these data, it is necessary to remove the outliers. To

improve the data access time, the filtering is done once for all the data in a study. In the process several filters are applied to the data and stored in the database. When accessing the data, the user has to choose whether to retrieve filtered data and what filter to use in that case. The filters are stored in two tables. The *FLTS* table stores the ID of the filter along with a name and filter parameters. The second table, *FLT_TRL_RELS* stores the M:N relationship between filters and trials. If a trial is eliminated by a filter, its ID will not be paired with the filter's ID in the *FLT_TRL_RELS* table.

D.2.3 Data Filtering

Filtering the data is a complex problem given the many configuration parameters of a trial. When the therapist or physician analyze the data they will select trials, based on the type of exercise, the motion done during the exercise and body part used during the exercise. Since during a trial there can be multiple parts involved, it is possible for a trial to be an outlier with respect to one body part performance while being normal with respect to the rest. Hence, a filter is not only defined by a table, column and filter numerical parameters, but also by exercise parameters that define a subset of the rows in that table.

A filter with the same numerical parameters is stored multiple times and with different IDs in the *FLTS* table, once for each column and rows subset. Since the number of such combinations is very large, the filters are applied only for the most common situations, the filtering task being left for the user in the rest of the cases.

An apparently simple way to avoid this would be to filter the data dynamically upon the user's selection, without storing the filters statically. This approach causes inconsistencies when the therapist chooses to analyze the same data over various periods, hence reducing the set to be filtered. This will cause trial common to two such time periods to appear in one as an accepted point while in the other as an outlier.

There are two filtering algorithms used to eliminate data outliers. One keeps as normal points, all the data lying away from the average within a certain fraction of the standard deviation. The second algorithm uses the same standard deviation based interval but measures it around the data median.

D.3 Database Views

The rehabilitation database stores patient data, which is subject to very strict privacy rules. To enforce these rules the following conventions have to be respected:

1. A patient can only access his or her own data, and the public data stored in the ROOT schema. A patient cannot access the data of other patients;
2. A STAFF member can access only the data of the patients assigned to the studies the STAFF member is part of;
3. A STAFF member can create a new patient user only for one of the studies that staffer is part of;
4. A STAFF member can edit the filters on a patient's data and can mark sessions, blocks, or trials as valid or invalid. No other data access is allowed;
5. A member of the ADMIN group or the ROOT user has complete access to the entire database.

The above rules require four levels of data restriction. The first level is schema access permissions. This is implemented using the role and privilege security model offered by the Oracle RDBMS. The next level is table access restriction. This can also be implemented using Oracle's privilege system.

To restrict the user access to data belonging to different studies, it is necessary to apply row level control on the trunk tables (i.e. SESS, BLKS, TRLS). This is implemented using views that restrict the access through a query that matches the row's study ID with the studies to which the accessing user belongs. Table D.2 presents the view associated to the *SESS* table.

D.4 Database Synonyms

The synonyms are an Oracle RDBMS feature that allows the developer to associate a name with a database entity. In our case, they are used to unify the naming discrepancies resulted from the usage of table and views. The use of synonyms also makes

Table D.2: SESS table view.

```
create or replace view v_sess as
    select *
    from sess
    where
        stu_id in
            (select stu_id
             from s_stu_usr_rels
             where usr_id =
                 (select id
                  from s_usrs
                  where upper(user_name) = upper(user)));
```

it easier to make changes to the database without having to change the other tiers by simply changing the mapping of the synonyms to the new structures.

D.5 Data Access Speed Optimizations

When analyzing a patient, the therapists looks most of the time at a history of their performance, and making the decisions for the next session based on this information. A second type of data access is to the high frequency data collected during a specific trial. Although less frequent, it is sometimes necessary to view this data in order to better understand the manifestation of the patient's condition. For instance, raw data graphs can show a certain tremor in the patient's motion, which would not be visible otherwise.

D.5.1 Performance History Optimization

In order to speed up the retrieval of performance data, the database uses the *TRL_PRFS* and *BLK_PRFS* tables. The trial level performance tables store the values of various measurements relevant to that specific exercise. This evidently shortens the access time considerably by skipping the calculation of these values for every request. The block level performance tables store the average and standard deviation of the performance of the trial belonging to each block.

D.5.2 Raw Data Access Optimization

A raw data table can have a few hundred thousand records. Each query for the raw data of a trial has to find all the rows containing the key of the trial and retrieve them. In the context of generating ten or more such graphs at a time the duration of the operation is of about 5–7 minutes, depending on the size of the raw data table. To reduce this duration, the raw data tables were created using Oracle's hash partitioning feature. Each table was split in 20 partitions, hashed by the trial ID field of each row, which is the attribute used for searching data in the raw data tables. This shortens the access time by approximately 70%.

D.5.3 Entity Relationship Sub–queries

Almost all of the queries sent to the database contain sub–queries matching sessions with block and trials by searching the tables after the appropriate attribute. For instance, the query necessary to find all the trials IDs belonging to one session is shown in Table D.3. The query has to parse both the TRLS table and the BLKS table. To reduce this overhead, the session ID fields were added to the TRLS table too. Although redundant, the space cost of this decision is negligible while the performance was improved by about 10%. Table D.4 shows the faster query using the redundant session ID field

Table D.3: Query for retrieving all the trials belonging to session 100.

```
select id
from trls
where blk_id in
    (select id
     from blks
     where ses_id = 100)
```

Table D.4: Query for retrieving all the trials belonging to session 100, using the redundant session ID field.

```
select id
from trls
where ses_id = 100
```

References

[1] S.V. Adamovich, A. Merians, R. Boian, M. Tremaine, G. Burdea, M. Recce, and H. Poizner. A virtual reality based exercise system for hand rehabilitation post-stroke. In *Proc. Second Int. Workshop on Virtual Rehabilitation*, pages 74–81, September 2003.

[2] S.V. Adamovich, A. Merians, R. Boian, M. Tremaine, G. Burdea, M. Recce, and H. Poizner. A virtual reality based exercise system for hand rehabilitation post-stroke. *Presence, Special Issue on Virtual Rehabilitation*, 2005.

[3] S.V. Adamovich, A.S. Merians, R. Boian, M. Tremaine, G.C. Burdea, M. Recce, and H. Poizner. A virtual reality based exercise system for hand rehabilitation post-stroke: Transfer to function. In *Proceedings of IEEE EMBS*, pages 4936–4939, September 2004.

[4] American Heart Association. Stroke statistics. http://www.americanheart.org/presenter.jhtml?identifier=4725.

[5] M. Badescu. *Reconfigurable Robots Using Parallel Platforms as Modules*. PhD thesis, Rutgers University, New Jersey, USA, May 2003.

[6] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller. Construction of an efficient overlay multicast infrastructure for ral-time applications. In *Proceedings of IEEE INFOCOM*, 2003.

[7] F. Bauer and A. Varma. Degree constrained multicasting in point-to-point networks. In *Proceedings of IEEE INFOCOM*, pages 369–376, 1995.

[8] F.P. Beer and E.R. Johnston Jr. *Vector Mechanics for Engineers*. McGraw-Hill, 4 edition, 1984.

[9] F.M. Boian. *Programarea Distribuita in Internet. Metode si Aplicatii*, chapter 3. Grupul Microinformatica, 3 edition, 2000.

[10] F.M. Boian, C. Ferdean, R. Boian, and R. Dragos. *Programare concurenta pe platforme Unix, Windows, Java*, chapter 3. Grupul Microinformatica, 2002.

[11] R. Boian, M. Bouzit, G. Burdea, and J.E. Deutsch. Dual stewart platform mobility simulator. In *Proceedings of IEEE EMBS*, pages 4848–4851, September 2004.

[12] R. Boian, A. Sharma, C. Han, A. Merians, G. Burdea, S. Adamovich, M. Recce, M. Tremaine, and H. Poizner. Virtual reality-based post stroke rehabilitation. In *Proceedings of Medicine Meets Virtual Reality*, pages 64–70, Newport Beach, CA, January 2002.

[13] R.F. Boian, G.C. Burdea, J.E. Deutsch, and S.H. Winter. Street crossing using a virtual environment mobility simulator. In *Proceedings of Third Int. Workshop on Virtual Rehabilitation*, pages 27–33, Lausanne, Switzerland, September 2004.

[14] R.F. Boian, J.E.Deutsch, C.S. Lee, G.C. Burdea, and J. Lewis. Haptic effects for virtual reality-based post-stroke rehabilitation. In *Proceedings of the Eleventh Symposium on Haptic Interfaces For Virtual Environment And Teleoperator Systems*, pages 247–253, Los Angeles, CA, March 2003.

[15] R.F. Boian, H. Kourtev, K.M. Erickson, J.E. Deutsch, J.A. Lewis, and G.C. Burdea. Dual stewart-platform gait rehabilitation system for individuals post-stroke. In *Proc. Second Int. Workshop on Virtual Rehabilitation*, page 92, September 2003.

[16] R.F. Boian, C.S. Lee, J.E. Deutsch, G.C. Burdea, and J.A. Lewis. Virtual reality-based system for ankle rehabilitation post stroke. In *1st International Workshop on Virtual Reality Rehabilitation (Mental Health, Neurological, Physical, Vocational) VRMHR*, pages 77–86, Lausanne, Switzerland, November 2002.

[17] L. Bouguilla and M. Sato. Virtual locomotion system for large-scale virtual environment. In *Proceedings of the IEEE Virtual Reality Conference*, pages 291–292, March 2002.

[18] M. Bouzit, G. Burdea, G. Popescu, and R. Boian. The rutgers master ii-new design force-feedback glove. *IEEE/ASME Transactions on Mechatronics*, 7(2):256–263, June 2002.

[19] E. Brosh and Y. Shavitt. Approximation and heuristics algorithms for minimum delay application-layer multicast trees. In *Proceedings of IEEE Infocom*, Hong Kong, March 2004.

[20] D.J. Brown, N. Shopland, and J. Lewis. Flexible and virtual travel training environments. In *Proceedings of the 4th International Conference on Disability, Virtual Reality and Associated Technologies*, pages 181–188, Veszprem, Hungary, September 2003.

[21] H. Bruyninckx and J. De Schutter. Comments on 'closed form forward kinematics solution to a class of hexapod robots'. *IEEE Transactions on Robotics and Automation*, 15(4):326–328, April 1999.

[22] G. Burdea. *Force and Touch Feedback for Virtual Reality.* John Wiley & Sons, New York, NY, USA, 1996.

[23] G. Burdea. Haptic feedback for virtual reality. *International Journal of Design and Innovation Research*, 2(1):17–29, July 2000.

[24] G. Burdea. Keynote address: Virtual rehabilitation: Benefits and challenges. In *Proceedings of the First International Workshop on Virtual Reality Rehabilitation (Mental Health, Neurological, Physical, Vocational)*, pages 1–11, Lausanne, Switzerland, November 7-8 2002.

[25] G. Burdea and P. Coiffet. *Virtual Reality Technology.* John Willey & Sons, 4 edition, 1994.

[26] G. Burdea, S. Deshpande, N. Langrana, D. Gomez, and B. Liu. A virtual reality-based system for hand diagnosis and rehabilitation. *Presence Teleoperators and Virtual Environments*, 6(2):229–240, April 1997.

[27] R.R. Christensen, J.M. Hollerbach, Y. Xu, and S.G. Meek. Inertial-force feedback for the treadport locomotion interface. *Presence*, 9(1):1–14, February 2000.

[28] G. Colombo, M. Joerg, R. Schreier, and V. Dietz. Treadmill training of paraplegic patients using a robotic orthosis. *Journal of Rehabilitation Research and Development*, 37(6):693–700, 2000.

[29] G. Colombo, M. Wirz, and V. Dietz. Driven gait orthosis for improvement of locomotor training in paraplegic patients. *Spinal Cord*, 39:252–255, 2001.

[30] F. Comeau, S. Chapdelaine, B.J. McFadyen, F. Malouin, A. Lamontagne, L.Galiana, D. Laurendeau, C.L. Richards, and J. Fung. Development of increasingly complex virtual environments for locomotor training after stroke. In *Proceedings of Second International Workshop on Virtual rehabilitation*, page 90, Piscataway, NJ, September 2003.

[31] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to algorithms*, chapter 25. MIT Press, 2 edition, 2001.

[32] J.J. Craig. *Introduction to Robotics Mechanics & Control*. Addison-Wesley, 1986.

[33] R.L. Craik and C.A. Oalis, editors. *Gait Analysys. Theory and Application*, chapter 6. Mosby, 1995.

[34] J. Deutsch, E. Whitworth, J. Lewis, R. Boian, G. Burdea, and M. Tremaine. Formative evaluation and preliminary findings of a virtual reality telerehabilitation system for the lower extremity. *Presence, Special Issue on Virtual Rehabilitation*, April 2005.

[35] J.E. Deutsch, J. Latonio, G. Burdea, and R. Boian. Post-stroke rehabilitation with the rutgers ankle system - a case study. In *Presence*, volume 10, pages 416–430. MIT Press, August 2001.

[36] J.E. Deutsch, J. Latonio, G. Burdea, and R. Boian. Rehabilitation of musculoskeletal injuries using the rutgers ankle haptic interface: Three case reports. In *Proceedings of Eurohaptics Conference*, pages 11–16, Birmingham, UK, 2001.

[37] J.E. Deutsch, A.S. Merians, G. Burdea, R. Boian, S. Adamovich, and Poinzer H. Haptics and virtual reality used to increase strength and improve function in chronic patients post-stroke: Two case reports. *Neurology Report*, 26(2):78–85, 2002.

[38] W.Q.D. Do and D.C.H. Yang. Inverse dynamic analysis and simulation of a platform type of robot. *Journal of Robotic Systems*, 5(3):209–227, 1988.

[39] Centers for Medicare & Medicaid Services (CMS). Medicaid and telemedicine. http://www.cms.hhs.gov/states/telemed.asp, January 2005.

[40] A.M. Gentile. *Movement Science Foundations for Physical Therapy in Rehabilitation in Carr J and Shepherd (Eds) Movement Science Foundations for Physical Therapy*, pages 93–154. 1987.

[41] M. Girone. The 'rutgers ankle' orthopedic rehabilitation interface. Technical report, Rutgers University, 1999.

[42] M. Girone, G. Burdea, and M. Bouzit. The 'rutgers ankle' orthopedic rehabilitation interface. In *Proceedings of the ASME Haptics Symposium*, volume 67, pages 305–312, November 1999.

[43] M. Girone, G. Burdea, M. Bouzit, V.G. Popescu, and J.E. Deutsch. Orthopedic rehabilitation using the "rutgers ankle" interface. In *Proceedings of Medicine Meets Virtual Reality*, pages 89–95, 2000.

[44] M. Girone, G. Burdea, M. Bouzit, V.G. Popescu, and J.E. Deutsch. A stewart platform-based system for ankle telerehabilitation. *Autonomous Robots*, 10:203–212, 2001.

[45] D. Gomez. *Dextrous Hand Master with Force Feedback for Virtual Reality.* PhD thesis, Rutgers University, Graduate School-New Brunswick, 1997.

[46] G.J. Grimes, H. Dubois, S.J. Grimes, W.J.Greenleaf, S. Rothenburg, and D. Cunningham. Telerehabilitation services using web-based telecommunication. In *Proceedings of Medicine Meets Virtual Reality*, pages 113–118, 2000.

[47] C. Ho, C. Basdogan, and M. Srinivasan. Haptic rendering: Point- and ray-based interactions. In *Proceedings of the Second PHANToM Users Group Workshop*, Dedham, MA, Ostober 1997.

[48] M. Holden, T. Dyar, J. Callahan, L. Schwamm, and E. Bizzi. Motor learning and generalization following virtual environment training in a patient with stroke. *Neurology report*, 25(5), 2000.

[49] M. Holden, E. Todorow, J. Callaban, and E. Bizzi. Virtual environment trainig improves motor performance in two patients with stroke: Case report. *Neurology report*, 23(2):57–67, 1999.

[50] M.K. Holden, T.A. Dyar, L. Schwamm, and E. Bizzi. Home-based telerehabilitation using a virtual environment system. In *Proceedings of Second International Workshop on Virtual rehabilitation*, pages 4–12, Piscataway, NJ, September 2003.

[51] J. Hollerbach. Locomotion interfaces. Institute for Mathematics and Applications Workshop, January 2001.

[52] J.M. Hollerbach, Y. Xu, R. Christensen, and S.C. Jacobsen. Design specifications for the second generation sarcos treadport locomotion interface. In *Haptics Symposium, Proc. ASME Dynamic Systems and Control Division*, volume 69, pages 1293–1298, November 2000.

[53] C. Innocenti and V. Parenti-Castelli. A novel numerical approach to the closure of the 6–6 stewart platform mechanism. In *Proceedings of IEEEE Int. Conf on Advanced Robotics*, volume 1, pages 851–855, June 1991.

[54] H. Iwata. Walking about virtual environments on infinite floor. In *Proceedings of IEEE Virtual Reality Annual International Symposium*, 1999.

[55] H. Iwata. Locomotion interface for virtual environments. In *Proceedings of Robotics Research: The Ninth International Symposium*, pages 275–282, London, 2000.

[56] H. Iwata, H. Yano, and F. Nakaizumi. Gaitmaster: A versatile locomotion interface for uneven virtual terrain. In *Proceedings of the IEEE VR2001 Conference*, 2001.

[57] H. Iwata and Y. Yoshida. Path reproduction tests using a torus treadmill. *Presence: Teleoperators and Virtual Environments*, 8:587–597, 1999.

[58] D. Jack, R. Boian, A. Merians, S. Adamovich, M. Tremaine, M. Recce, G. Burdea, and H. Poizner. A virtual reality-based exercise program for stroke rehabilitation. In *Proceedings of ASSETS 2000: Fourth ACM SIGCAPH Conference on Assistive Technologies*, pages 56–63, November 2000.

[59] D. Jack, R. Boian, A. Merians, S. Adamovich, M. Tremaine, M. Recce, G. Burdea, and H. Poizner. Virtual reality-enhanced stroke rehabilitation. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 9(3):308–318, September 2001.

[60] D.L. Jaffe. Using virtual reality and telerehabilitation to improve walking in individuals following stroke. In *Proceedings of the State of the Science Conference on Telerehabilitation*, pages 57–59, Washington, DC, October 2001.

[61] P. Ji and H. Wu. A closed-form forward kinematics solution for the 6-6p stewart platform. *IEEE Transactions on Robotics and Automation*, 17(4), August 2001.

[62] Z. Ji. Dynamics decomposition for stewart platforms. *Journal of Mechanical Design*, 116:67–69, March 1992.

[63] H.S. Jorgensen, H. Nakayama, H.o. Raaschou, and T.S. Olsen. Recovery in walking function in stroke patients: The copenhagen stroke study. *Archives of Physical Medicine and rehabilitation*, 76(1):27–32, 1995.

[64] K. Kosuge, K. Takep, T. Fukuda, K. Kai, T. Mizuno, and H. Tomimatsu. Computation of parallel link manipulator dynamics. In *Proceeedings of IEEE Int. Conf on Industrial Electronics, Control, and Instrumentation*, volume 3, pages 1672–1677, November 1993.

[65] H. Krebs, N. Hogan, M. Aisen, and B. Volpe. Application of robotics and automation technlogy in neuro-rehabilitation. In *Proceedings of ASME Japan-USA Symposium on Flexible Automation*, volume 1, pages 269–275, 1996.

[66] E. Kreyszig. *Advanced Engineering Mathematics*. Jown Wiley & Sons Inc., 8 edition, 1999.

[67] K.G. Kwakkel, R.C. Wagenaar, and T.W. Koelman. Effects of intensity of rehabilitation after stroke: a research synthesis. *Stroke*, 28(8):1550–1556, 1997.

[68] Z. Lazarevic. Feasability of a stewart platform with fixed actuators as a platform for cabg surgery device. Master's thesis, Columbia University, Department of Bioengineering.

[69] H.Y. Lee and B. Roth. A closed-form solution of the forward displacement analysis of a class of in-parallel mechanisms. In *Proceedings of IEEE Int. Conf. on Robotics and Automation*, pages 720–724, Atlanta, GA, 1993.

[70] J. Lewis, R.F. Boian, G.C. Burdea, and J.E. Deutsch. Real-time web-based telerehabilitation monitoring. In *Proceeding of Medicine Meets Virtual Reality 11*, pages 190–192, Newport Beach, CA, January 2003.

[71] J.A. Lewis, R.F. Boian, G. Burdea, and J.E. Deutsch. Remote console for virtual telerehabilitation. In *Proceedings of MMVR 2005*, Long Beach, California, January 2005.

[72] D. Li and S.E. Salcudean. Modeling, simulation and control of a hydraulic stewart platform. In *Proceeedings of IEEE Int. Conf on Robotics and Automation*, pages 3360–3366, Albuquerque, New Mexico, April 1997.

[73] K. Liu, F. Lewis, G. Lebret, and D. Taylor. The singularities and dynamics of a stewart platform manipulator. *Journal of Intelligent and Robotic Systems*, 8:287–308, 1993.

[74] M. J. Liu, C.X. Li, and C.N. Li. Dynamics analysis of the gough-stewart platform manipulator. *IEEE Transactions on Robotics and Automation*, 16(1):94–97, February 2000.

[75] J.C.S. Lui. Constructing communication subgraphs and deriving an optimal synchronization interval for distributed virtual environment systems. *IEEE transactions on Knowledge and Data Engineering*, 13(5):778–792, September 2001.

[76] N.A. Lynch. *Distributed Algorithms*, chapter 10. Morgan Kaufmann Publishers Inc., 1996.

[77] O. Ma and J. Angeles. Optimum architecture design of platform manipulators. In *Proceedings of the Fifth International Conference on Advanced Robotics*, volume 2, pages 1130–1135, June 1991.

[78] A.S. Merians, D. Jack, R.F. Boian, M. Tremaine, G.C. Burdea, S. Adamovich, M. Recce, and H. Poizner. Virtual reality-augmented rehabilitation for patients following stroke. *Physical Therapy*, 82(9):898–915, September 2002.

[79] J.P. Merlet. An algorithm for the forward kinematics of general 6 d.o.f. parallel manipulators. Technical report.

[80] J.P. Merlet. *Parallel Robots*. Kluwer Academic Publishers, 2000.

[81] T. Miyasato. Tele-nursing system with realistic sensations using virtual locomotion interface. In *Proceedings of the 6th ERCIM Workshop "User Interfaces for All"*, Florence, Italy, October 2000.

[82] P. Nanua, k.J. Waldron, and V. Murthy. Direct kinematic solution of a stewart platform. *IEEE Transactions on Robotics and Automation*, 6(4):438–444, August 1990.

[83] C.C. Nguyen and F.J. Pooran. Kinematic analysis and workspace determination of a 6dof ckcm robot end-effector. *Journal of Mechanical Working Technology*, pages 283–294, 1989.

[84] H. Noma and T. Miyasato. Design for locomotion interface in a large scale virtual environment, atlas: Atr locomotion interface for active self motion. In *Proceedings of ASME-DSC*, volume 64, pages 111–118, 1998.

[85] H. Noma, T. Sugihara, and T. Miyasato. Development of ground surface simulator for tel-e-merge system. In *Preoccedings of the IEEE Virtual Reality Conference*, pages 217–224, New Brunswick, NJ, March 2000.

[86] Office of Communications, National Institute of Neurological Disorders Public Liaison, and Stroke. Post-stroke rehabilitation fact sheet. http://www.ninds.nih.gov/disorders/stroke/poststrokerehab.htm, July 2004.

[87] University of Tsukuba VRlab. Gaitmaster (omni-directional uneven surface display). http://intron.kz.tsukuba.ac.jp/GaitMaster/GM.htm.

[88] H. Pang and M. Shahinpoor. Inverse dynamics of a parallel manipulator. *Journal of Robotic Systems*, 11(8):693–702, 1994.

[89] J. Parsons, D.R. Lampton, K.A. Parsons, B.W. Knerr, D. Russell, G. Martin, J. Daly, B. Kline, and M. Weaver. Fully immersive team training: A networked testbed for ground based training missions. In *Proceedings of Interservice/Industry Training, Simulation and Education Conference*, Orlando, FL, 1998.

[90] A.E. Patla. Understanding the role of vision in the control of human locomotion (invited review paper). *Gait and Posture*, 5:54–69, 1997.

[91] A.E. Patla. How human gait is controlled by vision? *Ecological Psychology*, 10:287–302, 1998.

[92] A.E. Patla. *Adaptive Locomotion the eyes have it In Duyens J et al., (Eds.) Control of Posture and Gait*, pages 589–593. 2001.

[93] A.E. Patla. Mobility in complex environments: Implications for clinical assessment and rehabilitation. *Neurology Report*, 25(3):82–90, 2001.

[94] A.E. Patla and A. Shumway Cook. Dimensions of mobility: Defining the complexity and difficulty. *Journal Aging Physical Activity*, 7:7–10, 1999.

[95] J. Pfeifer, A. Hopper, and B. Sudduth. A patient-centric approach to telemedicine database development. In *Proceedings of Medicine Meets Virtual Reality*, pages 67–73, 1998.

[96] L. Piron, . Tonin, A. Atzori, E. Trivello, and M. Dam. A virtual-reality based motor tele-rehabilitation system. In *Proceedings of Second International Workshop on Virtual rehabilitation*, pages 21–26, Piscataway, NJ, September 2003.

[97] G.V. Popescu, G. Burdea, and R. Boian. Shared virtual environments for telerehabilitation. In *Proceedings of Medicine Meets Virtual Reality*, pages 362–368, Newport Beach CA, January 2002.

[98] V.G. Popescu. *Design and Performance Analysis of a Virtual Reality-Based Telerehabilitation System*. PhD thesis, Rutgers University, January 2001.

[99] V.G. Popescu, G. Burdea, and M. Bouzit. Virtual reality simulation modeling for a haptic glove. In *Proceedings of Computer Animation Conference*, pages 195–200, Geneva, Switzerland, May 1999.

[100] V.G. Popescu, G. Burdea, M. Bouzit, M. Girone, and V. Hentz. A virtual reality-based telerehabilitation system with force feedback. *IEEE Transactions on Information Technology in Biomedicine*, 4(1):45–51, March 2000.

[101] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C*, chapter 2. Cambridge University Press, 1999.

[102] C. Reboulet and T. Berthomieu. Dynamic models of a six degree of freedom parallel manipulators. In *Proceeedings of IEEE Int. Conf on Advanced Robotics*, volume 2, pages 1153–1157, June 1991.

[103] G. Riva. Virtual reality in rehabilitation of spinal cord injuries: A case report. *Rehabilitation Psychology*, 45(1):1–8, 2000.

[104] X. Shi and R.G. Fenton. A complete and general solution to the forward kinematics problem of platform-type robotic manipulators. In *Proceeedings of IEEE Int. Conf on Robotics and Automation*, volume 4, pages 3055–3062, May 1994.

[105] G.L. Smidt, editor. *Gait in Rehabilitation*, chapter 1. Churchill Livingstone, 1990.

[106] S.V. Sreenivasan, K.J. Waldron, and P. Nanua. Closed-form direct displacement analysis of a 6–6 stewart platform. *Mechanism and Machine Theory*, 29(6):855–864, 1994.

[107] K.M. Stanney, editor. *Handbook of Virtual Environments : Design, Implementation, and Applications (Human Factors and Ergonomics)*, chapter 11. Lawrence Erlbaum Assoc, February 2002.

[108] J.S. Steinman, C.A. Lee, L.F. Wilson, and D.M. Nicol. Global virtual time and distributed synchronization. In *Proceedings of the ninth workshop on Parallel and distributed simulation*, pages 139–148, 1995.

[109] W.R. Stevens, B. Fenner, and A.M. Rudoff. *UNIX Network Programming*, volume 1, chapter 5. Addison-Wesley, 3 edition, 2004.

[110] D. Stewart. A platform with 6 degrees of freedom. In *Proceedings of the Institution of Mechanical Engineers*, 1965.

[111] F. Tamasebi and L.W. Tsai. Closed-form direct kinematics solution of a new parallel minimanipulator. *Journal of Mechanical Design*, 116:1141–1147, December 1994.

[112] J.N. Templeman, P.S. Denbrook, and L.E. Sibert. Maintaining spatial orientation during travel in an immersive virtual environment. *Presence: Teleoperators and Virtual Environments*, 8:598–617, 1999.

[113] M. Visintin, H. Barbeau, N. Korner-Bitensky, and N.E. Mayo. A new approach to retrain gait in stroke patients through body weight support and treadmill stimulation. *Stroke*, 29:1122–1128, 1998.

[114] D.T. Wade, F.M. Collen, G.F. Robb, and C.P. Warlow. Physiotherapy intervention late after stroke and mobility. *British Medical Journal*, 304:609–613, 1992.

[115] A. Wandel, H.S. Jorgensen, H. Nakayama, H.o. Raaschou, and T.S. Olsen. Prediction of walking function in stroke patients with initial lower extremity paralysis: the copenhagen stroke. *Archives of Physical Medicine and rehabilitation*, 81(6):736–738, 2000.

[116] Z. Wang, K. Bauernfeind, and T. Sugar. Omni-direactional treadmill system. In *Proceedings of the Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 367–373, Los Angeles, CA, March 2003.

[117] E. Whitworth, J.A. Lewis, R. Boian, M. Tremaine, G. Burdea, and J. Deutsch. Formative evaluation of a virtual reality telerehabilitation system for the lower extremity. In *Proc. Second Int. Workshop on Virtual Rehabilitation*, pages 13–20, September 2003.

[118] J. Yang and Z.J. Geng. Closed form forward kinematics solution to a class of hexapod robots. *IEEE Transactions on Robotics and Automation*, 14(3):503–508, June 1998.

[119] J. Yoon, J. Ryu, G. Burdea, and R. Boian. Control of the rutgers ankle rehabilitation device. In *ASME Winter Annual Congress and Exhibition, Proc. of Advances in Robot Dynamics and Control*, New Orleans, USA, Nov. 17-22 2002.

[120] C.D. Zhang and S.M. Song. An efficient method for inverse dynamics of manipulators based on the virtual work principle. *Journal of Robotic Systems*, 10(5):605–627, 1993.

[121] C.D. Zhang and S.M. Song. Forward position analysis of nearly general stewart platforms. *Transaction of the ASME Journal of Mechanical Design*, 116:54–60, March 1994.

Curriculum Vitae

Rares F. Boian

Ph.D. in Computer Engineering; Rutgers University, NJ, USA.

MS Computer Science, concentration in Computer Graphics; GPA: 9.80/10.00, "Babes–Bolyai" University, Cluj–Napoca, Romania

BS Computer Science, concentration in Distributed Computing and Database Systems GPA 9.89/10.00; 2nd GPA in the year; "Babes–Bolyai" University, Cluj–Napoca, Romania

Publications

S. V. Adamovich, A. S. Merians, R. Boian, M. Tremaine, G. S. Burdea, M. Recce, and H. Poizner. A virtual reality based exercise system for hand rehabilitation post–stroke: Transfer to function. In *Proceedings of IEEE EMBS*, pages 4936–4939, September 2004.

S.V. Adamovich, A. Merians, R Boian, M. Tremaine, G. Burdea, M. Recce, and H Poizner. A virtual reality based exercise system for hand rehabilitation post–stroke. In *Proc. Second Int. Workshop on Virtual Rehabilitation*, pages 74–81, September 2003.

R Boian, M. Bouzit, G. Burdea, and J.E. Deutsch. Dual stewart platform mobility simulator. In *Proceedings of IEEE EMBS*, pages 4848–4851, September 2004.

R. Boian, A. Sharma, C. Han, A. Merians, G. Burdea, S. Adamovich, M. Recce, M. Tremaine, , and H. Poizner. Virtual reality–based post stroke rehabilitation. In *Proceedings of Medicine Meets Virtual Reality*, pages 64–70, Newport Beach, CA, January 2002.

R. F. Boian, H. Kourtev, K. M. Erickson, J.E. Deutsch, J.A. Lewis, and G.C. Burdea. Dual stewart–platform gait rehabilitation system for individuals post–stroke. In *Proc. Second Int. Workshop on Virtual Rehabilitation*, page 92, September 2003.

R.F. Boian, G.C. Burdea, J.E. Deutsch, and S. H. Winter. Street crossing using a virtual environment mobility simulator. In *Proceedings of Third Int. Workshop on Virtual Rehabilitation*, pages 27–33, Lausanne, Switzerland, September 2004.

R.F. Boian, J.E.Deutsch, C.S. Lee, G.C. Burdea, and J. Lewis. Haptic effects for virtual reality–based post–stroke rehabilitation. In *Proceedings of the Eleventh Symposium on Haptic Interfaces For Virtual Environment And Teleoperator Systems*, pages 247–253, Los Angeles, CA, March 2003.

R.F. Boian, C.S. Lee, J.E. Deutsch, G. Burdea, and J.A. Lewis. Virtual reality–based system for ankle rehabilitation post stroke. In *1st International Workshop on Virtual Reality Rehabilitation (Mental Health, Neurological, Physical, Vocational) VRMHR*, pages 77–86, Lausanne, Switzerland, November 2002.

M. Bouzit, G. Burdea, G. Popescu, and R. Boian. The rutgers master ii–new design force–feedback glove. *IEEE/ASME Transactions on Mechatronics*, 7(2):256–263, June 2002.

J. E. Deutsch, J. Latonio, G. Burdea, and R. Boian. Post–stroke rehabilitation with the rutgers ankle system - a case study. In *Presence*, volume 10, pages 416–430. MIT Press, August 2001.

J. E. Deutsch, J. Latonio, G. Burdea, and R. Boian. Rehabilitation of musculoskeletal injuries using the rutgers ankle haptic interface: Three case reports. In *Proceedings of Eurohaptics Conference*, pages 11–16, Birmingham, UK, 2001.

Whitworth E., J. Deutsch, J. Lewis, R. Boian, Marilyn Tremaine, and G. Burdea. Formative evaluation and preliminary findings of a virtual reality telerehabilitation system for the lower extremity. *Presence, Special Issue on Virtual Rehabilitation*, April 2005.

Whitworth E., J. A. Lewis, R. Boian, M. Tremaine, G. Burdea, and J. Deutsch. Formative evaluation of a virtual reality telerehabilitation system for the lower extremity. In *Proc. Second Int. Workshop on Virtual Rehabilitation*, pages 13–20, September 2003.

Lewis JA, RF Boian, G Burdea, and JE Deutsch. Remote console for virtual telerehabilitation. In *Proceedings of MMVR 2005*, Long Beach, California, January 2005.

D. Jack, R. Boian, A. Merians, S. Adamovich, M. Tremaine, M. Recce, G. Burdea, and H. Poizner. A virtual reality–based exercise program for stroke rehabilitation. In *Proceedings of ASSETS 2000: Fourth ACM SIGCAPH Conference on Assistive Technologies*, pages 56–63, November 2000.

D. Jack, R. Boian, A. Merians, S. Adamovich, M. Tremaine, M. Recce, G. Burdea, and H. Poizner. Virtual reality–enhanced stroke rehabilitation. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 9(3):308–318, September 2001.

Deutsch JE, Merians AS, Burdea G, Boian R, and Adamovich S. Poinzer H. Haptics and virtual reality used to increase strength and improve function in chronic patients post–stroke: Two case reports. *Neurology Report*, 26(2):78–85, 2002.

J. Lewis, R.F. Boian, G.C. Burdea, and J.E.Deutsch. Real–time web–based telerehabilitation monitoring. In *Proceeding of Medicine Meets Virtual Reality 11*, pages 190–192, Newport Beach, CA, January 2003.

Alma S. Merians, David Jack, Rares Boian, Marilyn Tremaine, Grigore C. Burdea, Sergei Adamovich, Michael Recce, and Howard Poizner. Virtual reality–augmented rehabilitation for patients following stroke. *Physical Therapy*, 82(9):898–915, September 2002.

G.V. Popescu, G. Burdea, and R. Boian. Shared virtual environments for telerehabilitation. In *Proceedings of Medicine Meets Virtual Reality*, pages 362–368, Newport Beach CA, January 2002.

Adamovich S., A. Merians, R. Boian, M. Tremaine, G. Burdea, M. Recce, and H. Poizner. A virtual reality based exercise system for hand rehabilitation post–stroke. *Presence, Special Issue on Virtual Rehabilitation*, 2005.

J. Yoon, J. Ryu, G. Burdea, and R. Boian. Control of the rutgers ankle rehabilitation device. In *ASME Winter Annual Congress and Exhibition, Proc. of Advances in Robot Dynamics and Control*, New Orleans, USA, Nov. 17–22 2002.