University of South Florida
Scholar Commons

Graduate Theses and Dissertations

Graduate School

11-21-2008

Design and Implementation of a Hard Real-Time Telerobotic Control System Using Sensor-Based Assist Functions

Eduardo J. Veras-Jorge
University of South Florida

Follow this and additional works at: https://scholarcommons.usf.edu/etd

Part of the American Studies Commons

Scholar Commons Citation

Veras-Jorge, Eduardo J., "Design and Implementation of a Hard Real-Time Telerobotic Control System Using Sensor-Based Assist Functions" (2008). *Graduate Theses and Dissertations.*
https://scholarcommons.usf.edu/etd/546

This Dissertation is brought to you for free and open access by the Graduate School at Scholar Commons. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Scholar Commons. For more information, please contact scholarcommons@usf.edu.

www.manaraa.com

Design and Implementation of a Hard Real-Time Telerobotic Control System Using Sensor-Based Assist Functions

by

Eduardo J. Veras-Jorge

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Mechanical Engineering
College of Engineering
University of South Florida

Major Professor: Rajiv Dubey, Ph.D.
Kathryn J. De Laurentis, Ph.D.
Susana K. Lai-Yuen, Ph.D.
Craig Lusk, Ph.D.
Wilfrido Moreno, Ph.D.
Kandethody Ramachandran, Ph.D.

Date of Approval:
November 21, 2008

Keywords: computer vision, haptics, robotic control, scaled teleoperation, virtual fixtures

© Copyright 2008, Eduardo J. Veras

www.manaraa.com

Acknowledgments

The author graciously acknowledges the invaluable assistance and support provided by Dr Rajiv Dubey for trusting me in the first place and for helping me to identify a research problem in rehabilitation robotics. His guidance and sharing of his knowledge were instrumental during the course of my research. I would also like to thank the rest of the members of my committee, Prof. Craig Lusk, Prof. Kandethody Ramachandran, Dr. Kathryn De Laurentis, Prof. Susana Lai-Yuen, and Prof. Wilfrido Moreno. They have all contributed by providing their feedback and by sharing their experience in the form of invaluable comments for improving this document. Thanks to lab mates and good friends Chris Colbert, Kathryn De Laurentis, Karan Khokar, Peter Schrock, Ramya Swaminathan, Redwan Alqasemi, Stephanie Carey, and Stephanie Stiber. I have had the opportunity to work with them and share memorable moments during this project. Special thanks to Karan Khokar and Redwan Alqasemi for assisting with experiments and testing of software. I also would like to express my gratitude to the staff of the Center for Rehabilitation Engineering and Technology: Bill Calvin, Linda Colon, Rosannah Parma, Stephen Sundarrao, and Vilma Fitzhenry and the ME department staff: Shirley Tervort, Susan Britten, and Wes Frusher. Thanks to Johanna Cedeno for her help with the statistical analysis. In a more personal level, I would like to express my deepest gratitude to my family. Thanks to my mother, Lilliam, and brothers, Hugo, Jose Alfonso, Jose Eduardo, and Jacque, who always have been so supportive of my efforts. Last, and for most, thanks to God, for considering me one of his sons and for showing me a way to become a better person every day.

المنارة للاستشارات

www.manaraa.com

Dedication

To Auristela, my lovely wife, for her unconditional love, and for just finding the appropriate boost to give when I really needed it. Also, I would like to dedicate this work to my children, Adriana and Alfonso; I can only hope they will do much better than I had ever dreamed in life.

www.manaraa.com

Table Of Contents

المنارة للاستشارات
www.manaraa.com

المنارة للاستشارات
www.manaraa.com

المنارة للاستشارات
www.manaraa.com

List Of Tables

المنارة للاستشارات

www.manaraa.com

List Of Figures

المنارة للاستشارات

www.manaraa.com

المنارة للاستشارات
www.manaraa.com

المنارة للاستشارات

www.manaraa.com

المنارة للاستشارات

www.manaraa.com

المنارة للاستشارات
www.manaraa.com

المنارة للاستشارات

www.manaraa.com

Design and Implementation of a Hard Real-Time Telerobotic Control System Using Sensor-Based Assist Functions

Eduardo J. Veras

Abstract

This dissertation presents a novel concept of a hard real-time telerobotic control system using sensory-based assistive functions combining autonomous control mode, force and motion-based virtual fixtures, and scaled teleoperation. The system has been implemented as a PC-based multithreaded, real-time controller with a haptic user interface and a 6-DoF slave manipulator. A telerobotic system is a system that allows a human to control a manipulator remotely and the human control is combined with computer control. A telerobotic control system with sensor-based assistance capabilities enables the user to make high-level decisions, such as target object selection, and it enables the system to generate trajectories and virtual constraints to be used for autonomous motion or scaled teleoperation. The design and realization of a telerobotic system with the capabilities of sensing and manipulating objects with haptic feedback, either real or virtual, require utilization of sensor-based assist functions through an efficient real-time control scheme. This dissertation addresses the problem of integrating sensory information and the calculation of sensor-based assist functions (SAF's) in hard real-time using PC-based resources. The SAF's calculations are based on information from a laser range finder, with additional visual feedback from a camera, and haptic measurements for motion assistance and scaling during the approach to a target and while

www.manaraa.com

following a desired path. This research compares the performance of the autonomous control mode, force and motion-based virtual fixtures, and scaled teleoperation. The results show that a versatile PC-based real-time telerobotic platform adaptable to a wide range of users and tasks is achievable. A key aspect is the real-time operation and performance with multithreaded software architecture. This platform can be used for several applications in areas such as rehabilitation engineering and clinical research, surgery, defense, and assistive technology solutions.

المنارة للاستشارات

www.manaraa.com

Chapter 1

Introduction

1.1. Motivation

The practicalities of creating a telerobotic control system to provide assistance for a wide community of users impose computational constraints in the realization of such system. On one hand, the external assistance (scaling, virtual fixture or haptic force feedback) is integrated with optical sensory information for computing the kind of assistance to be provided. On the other hand, the use of supervisory control i.e. human-in-the-loop for physical control of the robot arm presents the possibility of introducing instability during task execution if the proper control action is delayed or the update rates are not consistent. It is desired to integrate a supervisory control (human-in-the-loop), in which the human is in control, and at times, might switch to autonomous control mode, scaling or virtual fixture teleoperation modes, in an accurate and deterministic fashion, for enabling stable control of the teleoperation while allowing sensor-based motion guidance.

The development of a hard real-time telerobotic controller with haptic and sensory integration requires that the generated assist functions are fully integrated in the control system. The implementation of hard real-time control algorithms is a fundamental step for the development of sensor-based assistive technology in such areas as rehabilitation and related training, surgery, defense, and assistive technology applications. During the user's interaction with real and virtual objects the haptic

المنارة للاستشارات
www.manaraa.com

response needs to be in real-time, allowing operation in a complex environment and providing user motion assistance during task execution. In this context, hard real-time means that all the timing constraints of the system are met every time. Besides the autonomous operation mode, others operations are implemented in position and velocity control modes by the implementation of regular, scaled, and virtual fixture teleoperation modes. In any of those control modes, the stability and predictability of the telerobotic system response depends on strict timing requirements. In order to satisfy the response time constraints for telerobotic system with sensor-based assistance, a flexible real-time and a multithreading approach are needed. The PC-based multithreaded architecture allows designing and implementing telerobotic tasks with additional capabilities for assistance and haptic manipulation of target objects.

1.2 Visual and Haptic Feedback

The integration of visual and haptic information is particularly difficult because of the different nature of the sensory signals. On one hand, the human brain can easily interpret continuous motion from visual signals being updated from 24-30 frames per second. On the other hand, the human sense of touch is much more demanding in terms of consistent timing and update rates. It is known that in order to generate a realistic sensation of touch the update rate must be at least 1000 Hz consistently to have rigid body sensations in the user's hands [1, 2]. A haptic interface such as the Phantom Omni requires a servo loop running between 1000-2000 Hz to transmit the sensation of a hard surface to the user's hands through its actuators. So, an additional constraint is the definition of the limits of the achievable stiffness for stable control of the haptic interface

المنارة للاستشارات
www.manaraa.com

[3]. The restrictions discussed above are very significant in telerobotic applications which require continuous control of the robot arm configurations (position and orientation) in autonomous or teleoperation modes. The design and implementation of a PC-based platform for sensor-assisted telerobotic system would provide a platform for the realization of a hard real-time teleoperation with a haptic interface by combining the desirable properties of autonomous and teleoperation control systems. Since PCs are ubiquitous, this platform can be more widely available and not exclusive to researchers or those who have access to major computer power.

1.3 Rehabilitation Robotics Applications

This platform can be used for the implementation and execution of different teleoperation tasks. The research environment in which it is realized is primarily concerned about the development of new technology or modifications to existing technology. This implementation would assist persons with disabilities to enhance their mobility and manipulation using robotic systems. This field is known as Rehabilitation Robotics. Rehabilitation robotics is a term associated with the use of robotic technology to assist persons with disabilities to perform tasks they are unable to accomplish, or have great difficulty accomplishing, without external assist methods to guide the user's interactions. Within this context, the experiments conducted to validate the system are related to task completion of Activities of Daily Living (ADL) such as pick-up-a-cup. Other ADL’s like opening-a-door, flipping-a-switch, and opening-a-faucet can be performed using the system. The testing of the system is conducted on healthy people performing a “pick-and-place” task, which is a common activity of daily living (ADL)

المنارة للاستشارات

www.manaraa.com

task. Three people are trained to use the Phantom Omni interface and to teleoperate the PUMA manipulator. The actual hardware used for performing the experiments include a 6-DoF PUMA 560 manipulator, a Phantom Omni haptic interface and the sensory suite consisting of a CCD camera, a Sick DT60 laser range finder and the PUMA encoders. The performance indicators are defined in terms of the "Absolute Position Error" (APE), the "Absolute Orientation Error" (AOE) indicators, and the task-completion time which are calculated using the recorded data sets for each experiment.

1.4 Dissertation Objectives

The major objectives of this dissertation are:

1. To begin the development of a PC-based hard real-time controller for a sensor-assisted telerobotic system with a haptic user interface and a 6-DoF slave manipulator.
2. To design a framework that can be useful for rehabilitation engineering, surgery, defense, and assistive technology applications.
3. The integration of visual and haptic feedback to assist the user’s motion for autonomous, and teleoperated manipulation of target objects.
4. To implement real-time sensor-based assist functions for user’s motion scaling.
5. To provide visual feedback combined with scaled teleoperation and virtual fixtures or constraints definitions to guide the user interactions while manipulating virtual and real objects.

المنارة للاستشارات
www.manaraa.com

6. To implement data structures and communication protocols that allows handling interactive simulations, haptic interactions, optical sensors, and robotic manipulations in real-time using a PC-based platform.
7. To develop a virtual reality model to simulate the telerobotic system in purely robotic mode and a haptic integrated mode for conceptual testing of the control algorithms.
8. To develop a control strategy based on a "closed form" solution for Puma-like manipulators and a "Jacobian-based" control strategy that is expandable to control redundant robot arms for which exact solutions are not available.

1.5 Dissertation Outline

This dissertation comprises eleven (11) chapters; each one deals with a major topic related to the development of the PC-based hard real-time telerobotic control system using sensory-based assist functions and the combination of autonomous control mode, force-based and motion-based virtual fixtures, and scaled teleoperation. Chapter 1 discusses the motivation for development of the system as well as the need for hard real-time telerobotics control combining autonomous and teleoperation control. Chapter 2 gives a background on previous work in the field of robotic teleoperation and assistance. The concept of real-time control and multithreading architecture of the teleoperation tasks is outlined in Chapter 3. Chapter 4 contains the basis of sensor-based telerobotic control implementation using position-based and velocity-based control modes. Chapter 5 describes the mapping of the sensors reference frames and the robot arm reference frame required for driving the robot arm using teleoperation with human-in-the-loop and

المنارة للاستشارات
www.manaraa.com

autonomous mode. Chapter 6 describes the sensor-based assistance functions for motion-dependent feedback. Chapter 7 explains the experimental methodology for performing the experiments and a definition of the performance measures utilized. Chapter 8 describes the virtual reality simulations developed for testing and debugging of the some of the algorithms implemented for the telerobotic and haptic system interfacing. Chapter 9 outlines the experiments conducted to show the control of the physical system and discussion of the results. Chapter 10 concludes the dissertation work with recommendations, and suggestions for future work are outlined in Chapter 11.

المنارة للاستشارات

www.manaraa.com

Chapter 2

Background

2.1 Introduction

Teleoperation tasks executed with the assistance of a haptic interface controller require controlling the position and orientation of a multiple degrees of freedom manipulator. Multiple joints of the manipulator are moved in a continuous way in order to obtain a particular configuration of its end-effector. The required tasks for the haptic interface, in general, are to follow a prescribed path, to provide force reflection through the device actuators, impedance simulation using simple mathematical models such as spring-type forces, and obstacle avoidance [4] [5]. These tasks are implemented with a human-machine interface which requires the user to be always-in-the-loop (supervisory control). In this work, a combination of supervisory control and autonomous control modes are implemented which requires the integration of haptic interfacing techniques with sensor-based assist functions (SAF's) and stable transitioning between control modes. The purpose is to reduce the burden of the user by eliminating the requirement of the user being "always-in-the-loop" and to provide assistance to guide the user using scaling and virtual fixtures. The concept of human-machine interactions combined with the concept of extending user's manipulation capabilities has been the topic of intensive research [6] [7] [8] [9]. The integration of sensory information to assist the user's motion by the generation of scaling and virtual constraints demands a consistent and stable timing response. The need for predictable performance is a key factor in the ability of a

المنارة للاستشارات

www.manaraa.com

hard real-time system to meet the application's response-time requirements for such applications. This chapter describes previous work done in the teleoperation and assistance areas. Also a summary containing the differential features of the system described in this dissertation is presented at the end of the chapter.

2.2 Teleoperation Robotics

Teleoperation refers to the concept of extending a person’s sensing and manipulation capability to a remote location [10]. It was first described by Ray Goertz who designed mechanisms such as mechanical pantograph devices to allow radioactive materials to be handled from a safe distance. Even though it was not a robotic application, it introduced a way for expanding research work in this direction. As teleoperation technology developed, the mechanical linkages were replaced by electrical servos and cameras replaced direct viewing, allowing the operator to be located arbitrarily far away. A more detailed description of several teleoperation types of systems and concepts are defined in the area of remote manipulation technology in [10].

The basics of computer-aided teleoperation technology were established around 1965-70 when robotics applications were implemented with the aim of increasing dexterity and manipulation [11]. In the early stages of the development of teleoperation technology, the primary applications appear in the area of nuclear waste handling and decommissioning, handling toxic chemicals and radioactive materials. The human operators were provided with visual aid through video displays, and operate remotely located slave robot via a hand controller, but not assistance was provided to them to effectively complete the task. The idea of supervisory control (which combines human

www.manaraa.com

and computer control) became apparent when researchers started to question how to teleoperate vehicles on the moon through the unavoidable time delay of three tenths of a second for the radio signal round trip to the Moon [10, 12, 13]. Early applications of teleoperation in space were basically implementing time delays in the control system where a human was remotely controlling a vehicle without force feedback or motion assistance. The time delays still continue to be a problem in space teleoperation for exploration.

In 1985, another area of research was developed to find ways to remotely operate underwater vehicles (RUV's). At that time, a RUV named Jason was used for exploring the sunken Titanic cruise. The control system of the Jason was designed by Yoerger [14] and it was tele-operated from the ARGO towed imaging platform from the surface. This system integrated a vision system to assist the researchers from surface during the underwater exploratory task. Nowadays, the underwater exploration system is commonly known as the ARGO/JASON system [15].

The term teleoperation typically refers to systems in which the human operator directly and continuously controls the remote manipulator or telerobot. In these systems, the kinematic chain manipulated by the operator is referred to as the "master", while the remote manipulator is referred to as the "slave". However, it is also used to define different levels of "autonomy". From this point of view, a "telerobot" is classified into two types [16]:

1. Tele-autonomy: refers to the combination of teleoperation and autonomous robotic control. In some cases, a unilateral controller is used where there is no feedback information from slave to master or from master to human.

المنارة للاستشارات
www.manaraa.com

2. Tele-collaboration: means that all operations are controlled by the human-machine interface, usually in the form of force reflection.

A teleoperation control system can be unilateral or bilateral depending on the data flow. In the case of a unilateral controller, the robot arm is operated as an open-loop system. If the master and the slave are physically separated, there may be a video feedback of the slave executing a task or even no video if the master and slave are in operator's viewing area. On the other hand, bilateral control provides force feedback to the teleoperator, thus forming a "kinesthetic" or "tele-presence" system [17, 18, 19]. In this case, human decisions are merged with the computer generated assistance to allow for complex forms of automatic control. The control system adds velocity/force inputs to those from the master in the impedance-controlled formulation to assist the motion of the manipulator. Bilateral impedance control allows force reflection to be provided to the operator during task execution [10, 20, 21]. In [18] Dubey et al proposed the variable impedance method where the impedance parameters are adapted to variable circumstances thus overcoming the conflict problem of choosing desired dynamics parameters. This controller is primarily used in tasks requiring contact, such as needle inserting into tissue or surface exploration. Teleoperation system design usually takes operation accuracy into account, not the convenience and simplification of the operation. With the improvement of the controller architecture and assistance attempt [22], the task performance of telerobotic system in rehabilitation engineering is still not satisfactory [23, 24, 25]. As explained in [26], for a simple "go get a cup and put it on a pad" task, it takes the operator an average of 50 seconds, mostly due to indexing the master once it reaches its workspace limit and tuning the gripper to grasp the target. Furthermore, the

المنارة للاستشارات
www.manaraa.com

performance largely depends on the operator's familiarity with the system. In most cases, using a robot as a teleoperated device to complete a task is much harder than using human arm and hand. It can soon become very exhausting, especially if it has to perform repeated tasks such as feeding, even with some assistance. Many researchers tried to improve the operation accuracy, reduce execution time and relieve the operator's mental labor through adding artificial intelligence (AI). Kawamura et al [27] looked at how far rehabilitation robots had come in possessing abilities that relieve the user from the mental burden of controlling the robot. This AI-based system contains modules for a voice-activated user interface which is capable to interpret fuzzy commands such as "move closer", "go slower" or "move a little bit faster". These "fuzzy terms" can be recorded through a macro action builder (similar to a script) which enables the user to specify a set of commands to perform a task. The macros can be replayed later as a high-level action commanded by the user. As described in [27], the system has the capability to plan the actions to take in order to achieve a goal by learning the preconditions and effects of those actions obtained through the macro builder interface. The utilization of sensors in intelligent telerobotic systems, such as vision-based assistance, has improved the operation of aligning the end effector with the target [28, 29] where the visual information is used as part of the user interface in the form of visual cues for guiding tool in order to reach a goal. This dissertation extends the utilization of sensors to the calculation of the assist functions to guide the user while following a trajectory as well as to align the tool (a Barrett hand) with the target.

المنارة للاستشارات
www.manaraa.com

2.3 Teleoperation Assistance

In a telerobotic system, a human operator controls the movements by sending commands or signals to the robot. In the last decade, developments in computer and communication technology have enabled the integration of the teleoperation robotics (telerobotics), sensory information, and haptic interfaces in such areas as rehabilitation, training, surgery, research, device testing, and assistive technologies development. These developments have allowed further development of the assistance algorithms to map the master commands to the slave in a way that scales up or down depending on the task and environment information (the scaling factors vary accordingly).

The assistance function concept consists of the generalization of position and velocity mappings between master and slave manipulators of a teleoperation system. It can be classified as regulation of position, velocity and contact forces. All of these assistance strategies are accomplished by modification of the control law parameters of simple mathematical models of spring-type and damping-type forces. A simple form of position assistance is scaling, in which the slave workspace is enlarged or reduced as compared to the master workspace. The velocity assistance is commonly used in approaching target and obstacle avoidance. In both cases, the velocity scaling varies according to whether motion in that particular direction is serving to further accomplishing the desired effect of the motion.

2.3.1 Position-Based Assistance Functions

In these functions, the motion of the manipulator is constrained to lie along a given line or in a 2D plane. Figures (2.1a) and (2.1b) illustrate the situation of the linear

المنارة للاستشارات
www.manaraa.com

and planar constraint definitions, respectively. A detailed explanation of the position-based assist functions can be found in [30]. In these particular functions, the force feedback is transferred to the user through the haptic device itself. This way the haptic is used as the actuation device to generate the force reflection as well as a positional sensor to measure the relative position between a trajectory point and the "tip" of the haptic device. This information is then compared with the external sensory information to correct for possible deviations from the intended trajectory.

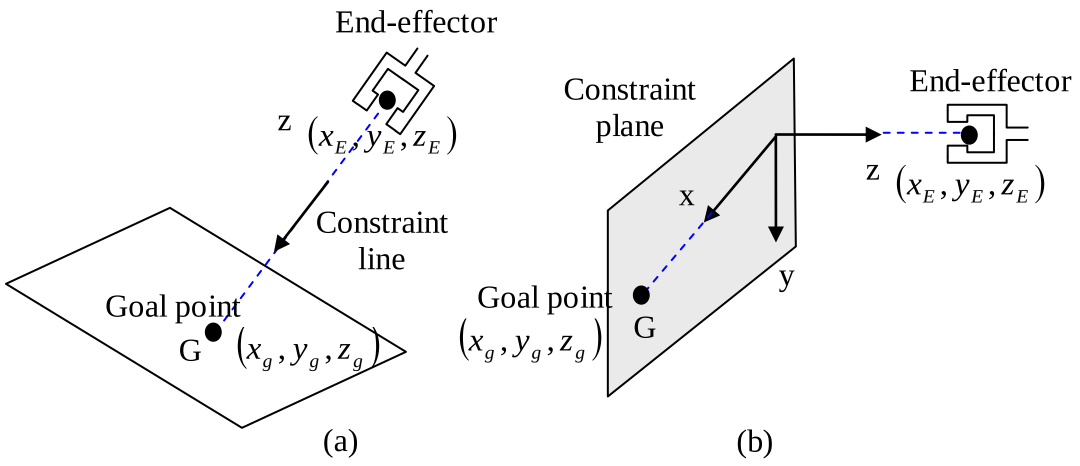


Figure 2.1 (a) End-effector Constrained to Motion on a Linear Path (b) End-effector Constrained to Motion on a Plane

2.3.2 Velocity Scaling Assistance Functions

In these functions, the level of assistance is based on velocity scaling according to whether the motion improves in the direction intended. In the approaching assistance mode, the velocity is scaled up (in free space) if the motion reduces the distance between the current and goal positions of the robot arm. Otherwise, the velocity is scaled down.

www.manaraa.com

Figure 2.2 shows scaling factors used for velocities scaling from previous work done in the Rehabilitation Robotic Lab [30].

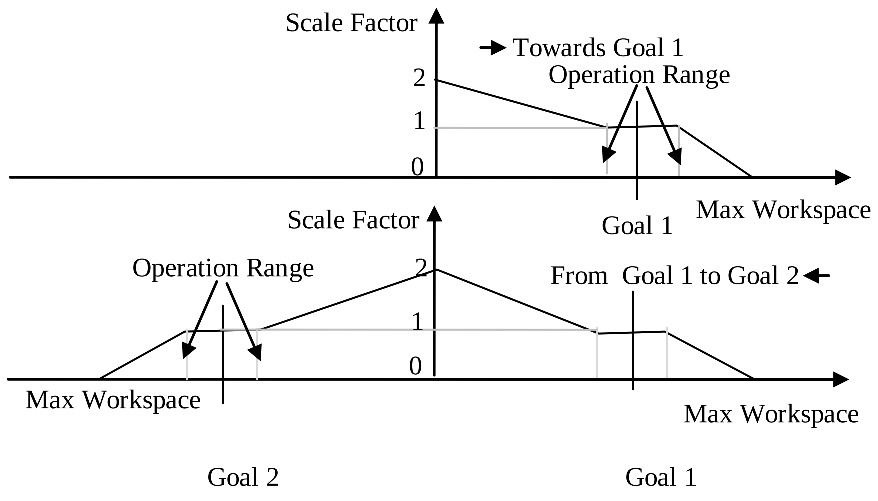


Figure 2.2 Scaling Factor Functions [26]

From this figure it can be observed that the change of the scaling factor depends on the proximity to the goal and the direction of motion. This same approach was used by Everett, who designed a vision-based mapping to align the end effector of the slave manipulator with a cross object [28, 31].

This is similar to what occurs using a Laser Range Finder readings and a vision system. Figure 2.3 shows how a velocity scaling factor varies based on the distance reading when the end-effector is approaching a wall. Using a vision system, the velocities that reduce the alignment error are scaled up and the ones that increase the alignment error are scaled down (Figure 2.4).

www.manaraa.com

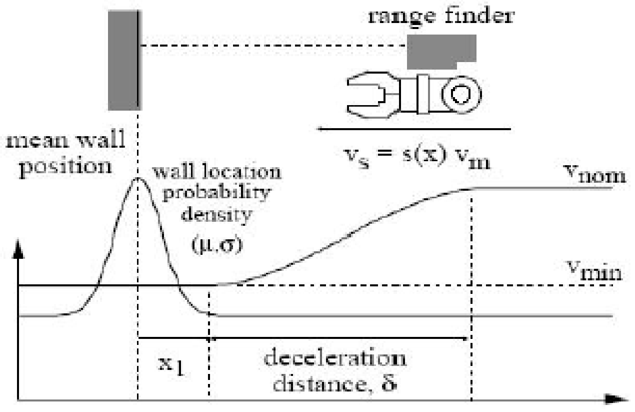


Figure 2.3 Scaling Factor Based on Laser Range Finder Reading [31]

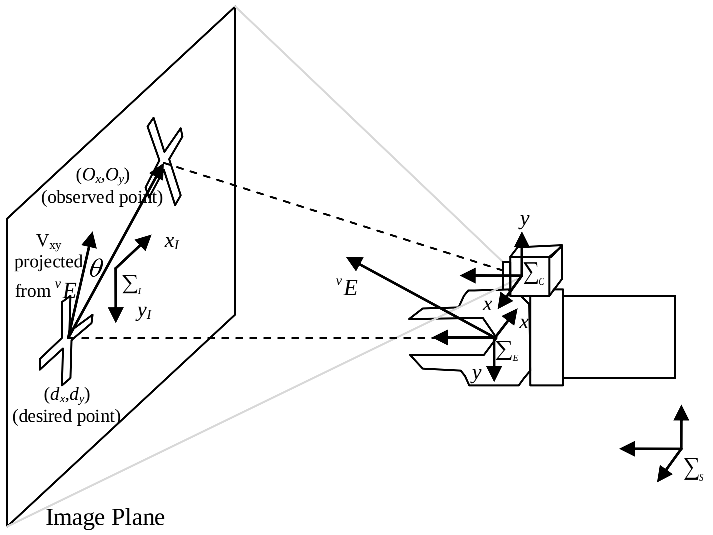


Figure 2.4 Cross Alignment Task Adapted from [31]

2.3.3 Virtual Fixture Assistance Functions

Another form of assistance used in tele-collaborative system is called “virtual fixtures” where the function parameters are time invariant and only vary according to spatial parameters. A canonical definition of virtual fixtures can be found in [32], as “abstract precepts overlaid on top of the reflected sensory feedback from a remote environment such that a natural and predictable relation exists between an operator’s

www.manaraa.com

kinesthetic activities (efference) and the subsequent changes in the sensations presented (afference)". As an example, a virtual 3D wall can be defined as a "fixture" to assist in linear trajectory following by creating a stop constraint to prevent a collision with a desktop. In teleoperation, a virtual fixture can be defined as a computed-generated spatial constraint that imposes positional or force limitations to a robot arm or operator movements. In practice, virtual fixtures are used to constrain a haptically controlled manipulator's motion along a desired path or to align the manipulator's end effector with a task [19, 33, 34, 35]. Usually, the stiffness coefficient along the desired path and stiffness orthogonal to the path are different. The stiffness ratio indicates the softness or hardness of the fixture. If the stiffness ratio is close to zero, it is the hardest fixture, which means that the end-effector can only move along the path without deviation. If the ratio is close to 1, it is the softest fixture, where the end-effector can move freely and it is usually used for trajectory following.

Virtual fixture can also be in the form of potential force fields [32, 36]. Potential fields are used to produce velocity commands, which, when added to those generated by the input device, maneuver the manipulator toward the target or away from obstacles [36]. Figure 2.5 shows that extract and insert fixtures restrict the motion of the end-effector when it is close to the tool grasping position. This behavior is implemented in order to avoid a collision of the manipulator with the tool, while allowing the operator to quickly reach the grasping position [36].

المنارة للاستشارات

www.manaraa.com

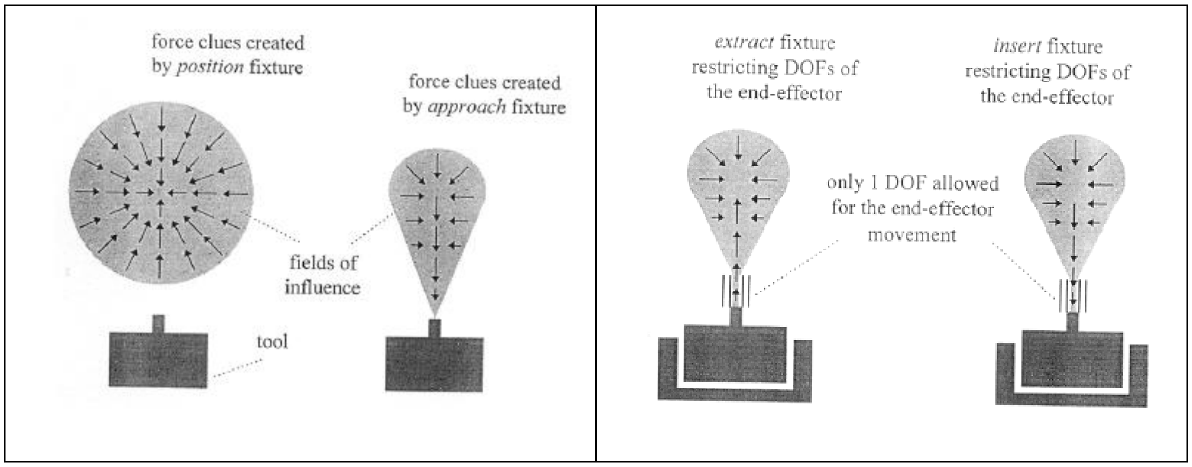

force clues created by *position* fixture
force clues created by *approach* fixture
fields of influence
tool
extract fixture restricting DOFs of the end-effector
insert fixture restricting DOFs of the end-effector
only 1 DOF allowed for the end-effector movement

Figure 2.5 Force Clues Generated by Position and Approach Fixtures (Left). Fixtures Restricting Degrees of Freedom (Right) [36]

The guiding force in this field is calculated using a potential function. This force can be attractive or repulsive, between the computer-controlled path following and the deviation from this path caused by the user input. To further explain this, the Lenard-Jones potential function is used here as an example.

The Lenard-Jones potential function is used in physics simulation of attraction or repulsion of atoms in Solid Mechanics. The acting regions of the force field are shown in Figure 2.6. The Lenard-Jones equation represents the inter-atomic potential energy, U, and is given by:

$$U = -\frac{A}{r^n} + \frac{B}{r^m} \tag{2.1}$$

In Eq. (2.1), r is the distance between atoms, and n, m, A, and B are constants. The first term in Eq. (2.1) represents the attraction force component, while the second term represents the repulsive force component. In order to compute the inter-atomic force between two atoms, the derivative of the potential energy is required as follows:

المنارة للاستشارات
www.manaraa.com

$$F = -\frac{d}{dr}U = \frac{-nA}{r^{n+1}} + \frac{mB}{r^{m+1}} \tag{2.2}$$

As can be observed from Eq. (2.2), the Lenard-Jones potential function can be used to avoid obstacles if the *A* parameter is made equal to zero (i.e., zeroing the attraction component) and keeping repulsion component only. On the other hand, if the parameter B is zeroed, then the potential function can be used to create a "stick" effect. In practice, boundaries defined around the desired path are created to act like virtual walls for guidance as explained above.

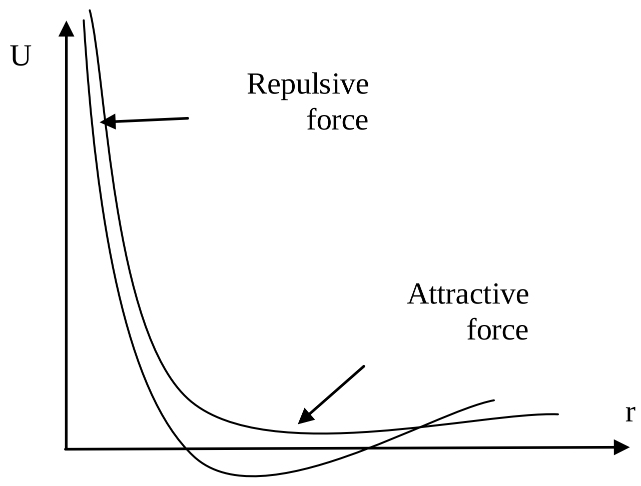


Figure 2.6 Lenard-Jones Potential Functions

2.4 Teleoperation in Real-time

There are several PC-based robotic control systems. Among these are QMotor 3.0 and QMotor RTK software packages developed by Costescu et al [37]. These packages use Object Oriented (OO) methods such as inheritance and polymorphism and a

المنارة للاستشارات
www.manaraa.com

Client/Server approach for asynchronous communication between different classes of services at the hardware and software control levels. The Operational Software Components for Advanced Robotics (OSCAR) framework is another program that uses OO framework for the development of control programs for robotic manipulators [38]. This particular software was developed as a set of GNU C++ classes for the Sun Solaris OS for graphical simulation and for VxWorks real-time OS for graphical and physical robot controllers. These two frameworks are useful for the control of the robotics manipulator as traditionally performed either through a GUI or manual input from the user using a keyboard. The QMotor RTK, for example, works exclusively at the joint level of the robotics arm and does not support a haptic application interface or sensor-based control.

The Open Robot Control Software (OROCOS) project is an open-source framework which runs on Linux OS named Linux RTAI (Real-Time Application Interface for Linux). This platform is a multi-purpose and modular framework for robot and machine control [39]. Being designed to work under Linux OS, the framework is not fully POSIX compliant limiting software portability and interoperability. At the time of this writing, the OROCOS platform does not support haptically controlled teleoperation.

A more recent system, Microsoft Robotics Studio (MSRS) [40, 41] by Microsoft, is based on services-oriented runtime architecture designed to run on Microsoft operating systems. MSRS allows asynchronous applications to communicate through Web-based or Windows-based interfaces developed in C#. A limitation of the services-based approach is that it does not allow for robotic framework integration and the human-machine interactions (HMI) through the sense of touch (haptic response) in hard real-time. In

المنارة للاستشارات
www.manaraa.com

addition, the integration of the sensor-based feedback when it is embedded in the control software would be difficult to achieve even in soft real-time.

A different platform using haptic control is described by Turro et al [42]. Turro's system was implemented as a client-server system for haptically augmented teleoperation using a master/slave scheme. The haptic feedback was achieved by using a slave controller consisting of a multi-processor Linux PC with 4 CPU's to control slave and one CPU to control the master device (for a total of five CPU's).

Some existing PC-based haptic systems are used for rehabilitation, but they do not integrate sensors and the assistance provided to the user is pre-recorded and, therefore, is not calculated in real-time. In [43], Hogan et al described the MIT-Manus, a robot-assisted therapy implementation aimed at the recovery of arm movement after stroke. The system uses a performance-based impedance control algorithm for controlling execution of tasks in a 2D plane. The patient receives assistance triggered by speed, time, or EMG thresholds. Charles et al [44] developed the Robot-Assisted Microsurgery (RAMS) telerobotic workstation in collaboration with JPL/NASA to augment micro-surgical dexterity. The system includes a 6-DoF robotic manipulator (slave) that holds surgical instruments. Motions of the instruments are commanded by moving the handle on a master device in the desired trajectories. The system was designed to assist skilled and able-bodied surgeons and is not suitable to assist people with disabilities to execute activities of daily living (ADL).

A bilateral teleoperation approach was implemented by Everett et al [45], where a slave manipulator (7 DOF K-2107 Robotics Research Corporation (RRC) robot manipulator) is controlled by tracking the motion of a master manipulator (Phantom

المنارة للاستشارات
www.manaraa.com

device). When the master touches an object, the slave reflects the forces back to the master device held by the operator [46]. It was developed using an SGI workstation and ControlShell graphical programming module running in the VxWorks OS. A Hidden Mark Model (HMM) based skill learning was developed by W. Yu et al, [47], to provide motion therapy using a haptic interface. This system can be used as a physical therapy for upper limb coordination, tremor reduction and motion control capabilities for persons with disabilities of the upper limb in a virtual environment. It was tested in simulation using a virtual reality representation of the RRC robotic arm. Chan et al [17] describes a telerobotic system, which includes variable stiffness and damping control schemes to control the master and the redundant slave dynamics to suit a given task. The functionality of the control scheme depends on sensed and commanded values of force and velocity, with no previous knowledge of the environment required. This prior research was not PC-based and not versatile for a wide range of applications. In 1999 researchers at the Budapest University of Technology and Economics in Hungary started the REHAROB project using standard, full-scale industrial robots for human therapy. This project is accounted to be the first in the world to target the use of standard, commercially available industrial robot (ABB manipulator) for the physiotherapy of spastic hemi-paretic stroke patients [48].

In contrast to these systems, the design described in this dissertation allowed us to create a simplified PC-based framework, which can be implemented widely. A key problem addressed is the integration of human-machine interactions combining the sense of touch and visual feedback as integral components of the robotic controller incorporating the advantages of real-time architecture in a PC-based framework. This

www.manaraa.com

platform provides for the benefits of a research laboratory setup to the user's desktop without demanding high-end computer resources. The autonomous and teleoperation control with capabilities for scaling and virtual constraint definitions are implemented with the intention of assisting the user’s motion by removing the restriction of the user of always being in the control loop, but keeping the high level decision making capabilities. This would result in fatigue reduction for task execution over long periods of time.

The combined work of Chan et al [17] and Everett et al [28] provided an approach for using uncertain sensor data based on the confidence of the measurements defined in terms of the mean and the standard deviation. The application of the assistance strategy concentrated on tasks related to radioactive waste tank cleanup. The nature of the associated tasks did not allow for autonomous command execution. In their work, the variable damping algorithm was implemented on a 7 DOF K-2107 Robotics Research Corporation, RRC, robot arm with position input from a 6 DOF Kraft master hand controller. The RS232 communication protocol was used to transfer the master controller signals to a SGI host workstation. A conversion from RS422 to RS232 was required because the Kraft‘s communication protocol is RS422. The system control software was implemented on a Silicon Graphics GTX 340 Workstation with 2-CPUs. One CPU is used for the master controller (6-DoF Kraft hand) and for the graphical user interface. The second CPU was used for the slave controller (RRC K-2107) and a low level programming approach in “Assembler” language for fast low level communication. The SGI host computer was connected to the RRC servo controller through a Bit3 VME-Multi-bus adapter.

المنارة للاستشارات
www.manaraa.com

In the present research work, the implementation of autonomous control and teleoperation control aims to facilitate the use of the assistive platform for any user making high-level decisions, such as target object selection. The system is capable of generating trajectories and virtual constraints to be used for autonomous motion or scaled teleoperation. This development involves the fusion of the optical sensor datasets and handling the transition states between the supervisory control system (human-in-the-loop) and the autonomous, sensory-driven control, and vice versa, in real-time. A summary of the demanded requirements is listed below:

1. The platform for development is a PC-based software controller which responds in real-time in robotic and haptic modes. The implementation runs under QNX Real-time Operating System (RTOS). QNX is a fully POSIX-compliant OS. This is a key feature because by following the POSIX (Portable Operating System Interface) standard, the application is portable to conformal POSIX standard OS. The following POSIX services were used in the current development:
    - i. Priority scheduling
    - ii. Real-time signals
    - iii. Real-time Timers
    - iv. Message passing
    - v. Thread creation and control
    - vi. Scheduling and synchronization of multiple threads
2. The telerobotic system uses two forms of robotic control: a closed-form solution of the inverse kinematics of the 6-DoF robot arm and a resolved-rate based algorithm. Both control strategies include gravity compensation.

المنارة للاستشارات
www.manaraa.com

3. The integration of the sensory data from the camera and laser is handled through an optimization solution to minimize the error using the Levenberg-Marquart methodology. The error function is defined by the distance between a given point in the world coordinate system and the same point given by the inverse perspective projection.
4. Sensor-based assist functions (SAF's) are implemented on a 6 DoF Puma560 robot arm with position input from a 3-DoF (force-based DoF) Phantom-Omni haptic device. The SAF helps the user to follow a trajectory path described in terms of the sensory input using motion scaling and virtual fixtures.
5. A low-level network protocol based on UDP (User Datagram Packets) packets provides the necessary flexibility, reduced latency, and resources for integrating data from diverse sensors. A single packet contains the vision information as well as the laser range finder information.
6. Rather than using conversion methods between different communication protocols, the UDP communication protocol is also used to transfer the master controller signals to the PC-based host computer. Support for TCP/IP streams is also provided.
7. The communication platform implements features to ensure the order of arrival of the data and mechanisms to handle data loses, if necessary.
8. The design takes into account that sensory datasets will be sent to multiple machines at once (for physical and virtual reality simulations) by using the multicast and broadcast transmission properties of the UDP protocol.

المنارة للاستشارات
www.manaraa.com

Chapter 3

Hard Real-Time Robotic Controller

3.1 Introduction

In the particular domain of telerobotics, the human is always in the control loop (supervisory control) while the robot arm is used to manipulate objects in a virtual or real environment. However, the users of telerobotic systems tend to fatigue over time and their performance is greatly reduced [49]. In these situations, it is useful to provide assistance to the user's motion and also to provide an autonomous mode of operation to reduce fatigue when the system is used over long periods of time. In this dissertation the assistance is provided to the users by the definition of sensor-based assisting or resisting forces as the users deviate from a trajectory as well as motion-based scaling and virtual fixture teleoperation. The calculated forces are delivered to the users through the haptic device (Phantom Omni) which provides the sensation of touch to the user's hands.

The integration of haptic feedback and the generation of the assisting or resisting forces based on sensory information is a challenge due to the uncertainty in the sensory information datasets, the deterministic timing and high frequency update rates for a realistic sensation of touch. In addition to this, the visual information extraction and data fusion requires computationally intensive pre-processing for obtaining the digital features from the images. This type of scenario imposes additional constraints in terms of the timing response of the system. This chapter discusses the approach followed in this dissertation to deal with the timing constraints and high update rates imposed by

المنارة للاستشارات
www.manaraa.com

separating the computational tasks into different running threads or “multithreading” the application with synchronization mechanisms for inter-processing communication to achieve real-time performance.

3.2 The Need for Real-Time Haptically Controlled Robotics

Real-time (RT) systems are defined as those systems in which the correctness of the system depends not only on the logical result of computations, but also on the time at which the results are produced [7]. Following this canonical definition, a real-time operating system (RTOS) is a specially designed operating system that supports real-time applications.

A distinctive characteristic of a RT application is that it must satisfy real-world timing boundaries without delays. In general, the main characteristics of RTOS are:

1. Respond predictably to unpredictable outside events
2. Meet timing deadlines
3. Ability to process multiple threads concurrently

In actual applications, RTOS specifications do not necessarily mean the response must be "fast". However, the timing requirements to complete the required tasks must be consistently accurate and predictable. If a computer process is designed and expected to update its data structure at a specified frequency of 1000Hz for example, the RTOS must not delay this process by allowing a low priority process to run first. In the literature, this property of RTOS is called determinism. When a RT application is running multiple threads or tasks concurrently, a running thread will be in control of certain resources of the CPU. The running thread must yield to another thread with higher priority, allowing

www.manaraa.com

the higher priority thread to run. The RTOS provides different mechanisms to handle this type of situations in real-time. Depending on the degree of failure if the system does not meet a specified deadline, a RTOS can be defined as "soft" or "hard" real-time operating system. In hard real-time systems, if the timing requirements are not met or the application response action is delayed for any reason, (e.g., elevators or aircrafts control systems) a catastrophic failure might occur. In control systems, for example, most applications must strictly meet real-world timing requirements in order to avoid catastrophic results. On the other hand, "soft" real-time systems will accept some level of lateness (e.g. a graphical user interface response for online authentication). Failure is not classified as catastrophic or incorrect in this case, but as an inconvenient response with a possible increased cost over time.

In the telerobotic application described in this work where sensor-based assist functions and haptic feedback are used to guide the user's motion, if the response-time requirements are not met, the robot controller will not be able to provide a stable control action, or it might be impossible to reach the prescribed destination with assistance. In this case, if the response-time constraint is violated, the result is an unrealistic effect or loss of the "sense of touch" in the user's hands. As shown by Salisbury et al [1], the haptic force feedback must be updated at a frequency of at least 1000 Hz consistently without delays in order to have a realistic sensation of touch. Even though the results in the haptic case might not be catastrophic, the system is described as a failure because the end results are not correct. Obstacle avoidance might be also an issue when negotiating obstacles resulting in a collision. The need for a predictable performance is, therefore, a

المنارة للاستشارات
www.manaraa.com

key factor in the ability of a real-time system to meet an application's response-time requirements.

The PC-based framework provided by this work allows implementing telerobotic applications with deterministic response times. The platform developed for real-time telerobotic, haptic feedback, and sensory data fusion systems is implemented as multithreaded application. The robotic system runs on QNX RTOS, which provides hard real-time timing, priority scheduling, and multithreading synchronization [50]. The haptic and sensory systems run on Windows XP OS, which is an event-driven and not a real-time operating system. The problem of predictability is alleviated by using a modified scheduler class developed to handle the high frequency update rates of the haptic thread under Windows. The platform sensory subsystem consists of a graphical user interface (GUI) which allows for image acquisition and post-processing. The laser ranger finder datasets are also displayed.

In this application, when the post-processing phase is completed, a different thread is assigned the task to act as a broadcasting server. This way, the user interface continues to be responsive and the display is immediately updated based on the most recently available data. If the data fusion is not programmed as a multithreaded application, the sensory subsystem will stop responding properly due to the event-driven nature of the Windows OS. The haptic and the simulation threads run concurrently, but they have different update rates, and therefore, the user will have a delayed response or an event-mismatching between the visual and the haptic feedback. In practice, the graphical simulation and display requires about 24 to 30 Hz to create a continuous motion sensation.

www.manaraa.com

3.3 Telerobotic Computational Tasks

In general, the computational tasks in telerobotic applications include the solution of forward and inverse kinematic problems, trajectory generation, and the calculation of the associated torques for commanding the motors to reach their destinations. The forward kinematics deals with the computation of the position and orientation of the tool frame relative to the base frame [51]. On other hand, the inverse kinematics deals with the problem of finding all possible sets of joint angles required to attain the given position and orientation of the end-effector of the robot arm [51]. The trajectory generation is related to the way a robot arm is moved from one location to another in a controlled manner. Generally, a trajectory planning module is implemented to create controlled movements in joint or Cartesian space. Finally, the torque calculations require the use of the kinematics and dynamics of the robot arm to achieve the desired joint angles. However, in practice, a form of linearized controller (Proportional-Integral-Derivative) is used as an approximation in order to reduce the computational intensive calculations required if the kinematics and the dynamics are used.

These computational tasks lead to the simultaneous motion in 3D space. In telerobotics this is achieved by controlling the position and orientation of the tool frame necessary to follow a desired trajectory or for reaching a specified point in space [51]. When the motion of the end-effector of the robot arm is controlled by a haptic interface (Phantom Omni, for example), the position and orientation of the end-effector of the haptic device ("haptic tip") must be mapped to that of the robot arm. The global position of the end-effector can be determined from the encoders feedback information located at each joint of the robot arm.

المنارة للاستشارات
www.manaraa.com

In the case of joint space control, the direct measurements from the haptic device encoders can be used to determine the joint angles which are then mapped to the corresponding joint angle of the manipulator. Given the numerical values of the haptic joint angles is relatively easy to map to the manipulator's reference frames. However, a more convenient way to map the different kinematics of the haptic and the robot arm is to use a Cartesian space solution, specially when the 3D motion of the robot arm is intended to be use for the execution of structured tasks.

3.4 Overview of the Robot Arm Controller and Forward Kinematics Equations

For modeling and controlling the robot arm, the kinematic equations of the links of the manipulator are necessary. These equations are obtained by systematically assigning coordinate frames to each link following the Denavit-Hartenberg (DH) convention [51]. The procedure described in [51] starts by assigning reference coordinate frames to each link starting at the base $\{L_0\}$, which is considered as a fixed link, and ending with frame $\{L_n\}$, attached to the robot end-effector of the Puma 560 for which n = 6 DoF. The following set of rules (0-13) and definitions are considered to assign coordinate frames to the links and therefore to determine the DH parameters based on Craig's notation [51]:

0. Number the joints from 1 to n starting with the base and ending with the tool yaw, pitch, and roll, in the specified order.
1. Assign a right-handed orthonormal coordinate frame $\{L_0\}$ to the robot base, making sure that z^0 aligns with the rotational axis of joint 1. Set $i = 1$.

www.manaraa.com

2. Align z^k with the rotational axis of joint $i+1$.
3. Locate the origin of $\{L_i\}$ at the intersection of z^i and z^{i-1} axes. If they do not intersect, use the intersection of z^i with a common normal between z^i and z^{i-1}.
4. Select x^i to be orthogonal to both z^i and z^{i-1}. If z^i and z^{i-1} are parallel, point x^i away from z^{i-1}.
5. Select y^i to form a right-handed orthonormal coordinate frame $\{L_i\}$.
6. Set $i = i+1$. If $i \prec n$, go to step 2; else continue.
7. Set the origin of $\{L_i\}$ at the tool tip. Align z^i with the approach vector, y^i with the sliding vector, and x^i with the normal vector to the tool. Set $i = 1$.
8. Locate point b^i at the intersection of x^i and z^{i-1} axes. If they do not intersect, use the intersection of x^i with a common normal between x^i and z^{i-1}.
9. Compute θ_i as the angle of rotation from x^{i-1} to x^i measure about z^{i-1}.
10. Compute d_i as the distance from the origin of frame $\{L_{i-1}\}$ to point b^i measured along z^{i-1}.
11. Compute a_i as the distance from point b^i to the origin of frame $\{L_i\}$ measured along x^{i-1}.
12. Compute α_i as the angle of rotation from z^{i-1} to z^i measure about x^i.
13. Set $i = i+1$. If $i \le n$, go to step 8; else stop.

Figure 3.1 shows the frame assignments and the zero pose configuration of the Puma 560 manipulator following the previous rules and definitions. Once the coordinate

www.manaraa.com

frames are assigned to every link on the chain, the transformations between adjacent coordinate frames can then be represented by the standard (4 x 4) homogenous coordinate transformation matrix, T. Therefore, the transformation matrix T is a mathematical description of the robot manipulator in terms of the DH parameters. Generally, the DH parameters are presented as a table containing one row of four parameters for each joint-link set with an attached coordinate frame. The DH parameters allow one reference frame to be located exactly with respect to the preceding link frame. The geometrical variables described by the modified DH parameters convention are presented in Table 3.1.

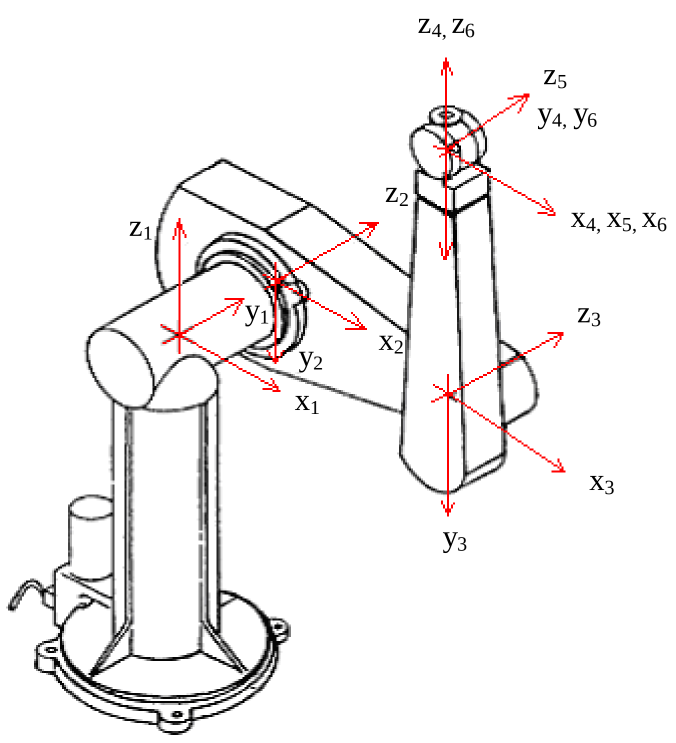


Figure 3.1 Coordinate Frame Assignments to Links of Puma 560 [51]

المنارة للاستشارات
www.manaraa.com

Table 3.1 DH Parameters of the Puma 560 Robot Arm [51]

Joint i	α_{i-1} (rad)	a_{i-1} (m)	d_i (m)	θ_i (rad)
1	0.0	0.0	0.0	θ_1
2	$-\frac{\pi}{2}$	0.0	0.2435	θ_2
3	0.0	0.4318	-0.0934	θ_3
4	$\frac{\pi}{2}$	-0.0203	0.4331	θ_4
5	$-\frac{\pi}{2}$	0.0	0.0	θ_5
6	$\frac{\pi}{2}$	0.0	0.0	θ_6

Figure 3.2 illustrates two adjacent link coordinate frames, $\{L_{i-1}\}$ and $\{L_i\}$, on a robot manipulator. The frame $\{L_i\}$ will be uniquely determined from frame $\{L_{i-1}\}$ by the definition of the DH parameters a_i, d_i, α_i and θ_i. The transformation matrix ${}^{i-1}_{i}T$ describing the position and orientation of the frame $\{L_i\}$ with respect to frame $\{L_{i-1}\}$ is determined (starting from frame $\{L_{i-1}\}$), as follows:

1. Translate a distance d_i from the origin of frame $\{L_{i-1}\}$ in the direction of z_{i-1} axis.
2. Determine the direction of x_i by rotating vector x_{i-1} by an angle θ_i around z_{i-1}.
3. Translate a distance a_{i-1} along the vector x_i. The position reached defines the origin of coordinate frame $\{L_i\}$, and the vector x_i is also determined.
4. Rotate the vector z_{i-1} about x_i by an angle α_{i-1} to determine the axis vector z_i.

www.manaraa.com

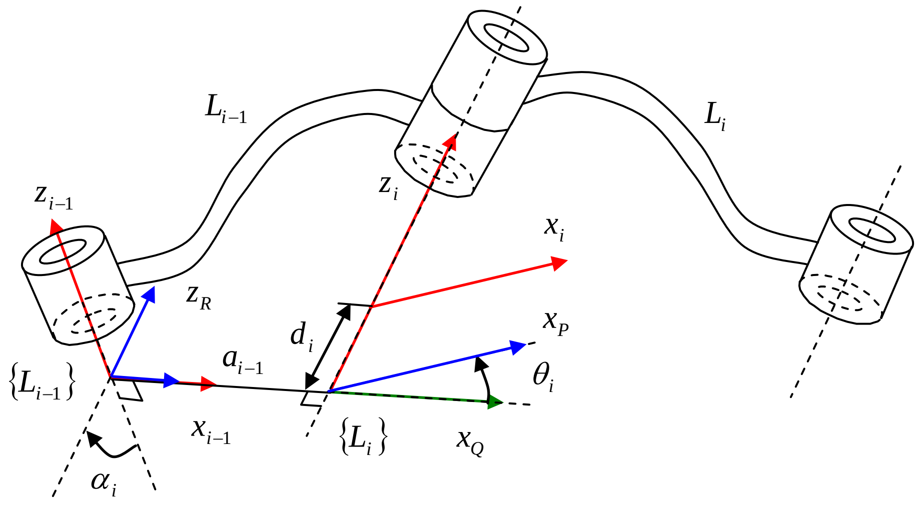


Figure 3.2 DH-Based Intermediate Transformations [51]

Symbolically, these four steps can be expressed as [51]:

$$ {}^{i-1}_{i}T = R_X(\alpha_{i-1}) D_X(a_{i-1}) R_Z(\theta_i) D_Z(d_i) \tag{3.1}$$

In this equation, the rotation matrix $R_X(\alpha_{i-1})$ defines a rotation about the x_i through an angle α_{i-1} and it is obtained as:

$$R_X(\alpha_{i-1}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & 0 \\ 0 & \sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

The translation transformation matrix along the x_i axis for a distance a_{i-1} is:

$$D_X(a_{i-1}) = \begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

المنارة للاستشارات
www.manaraa.com

The rotation matrix $R_Z(\theta_i)$ defines a rotation around z_{i-1} by an angle θ_i and is given by:

$$R_Z(\theta_i)=\begin{bmatrix}\cos(\theta_i) & -\sin(\theta_i) & 0 & 0\\ \sin(\theta_i) & \cos(\theta_i) & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{3.4}$$

The translation transformation matrix along the z_{i-1} axis for a distance d_i is:

$$D_Z(d_i)=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & d_i\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{3.5}$$

By substituting Equations (3.2) through (3.5) into Eq. (3.1) and performing the symbolic multiplications yield to the homogenous transformation matrix based on the modified DH parameters:

$$^{i-1}_{\ \ i}T=\begin{bmatrix}\cos(\theta_i) & -\sin(\theta_i) & 0 & a_{i-1}\\ \sin(\theta_i)\cos(\alpha_{i-1}) & \cos(\theta_i)\cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & -\sin(\alpha_{i-1})\cdot d_i\\ \sin(\theta_i)\sin(\alpha_{i-1}) & \cos(\theta_i)\sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & -\cos(\alpha_{i-1})\cdot d_i\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{3.6}$$

Table 3.1 shows the DH parameters at the home position. The objective now is to obtain the corresponding transformation matrices that relate the spatial position and orientation of the links connecting all the joints of the Puma 560 manipulator (See Appendix A). The transformation of the end-effector of the robot arm is found as:

$$^0_6T={}^0_1T\cdot{}^1_2T\cdot{}^2_3T\cdot{}^3_4T\cdot{}^4_5T\cdot{}^5_6T \tag{3.7a}$$

The final transformation obtained after the symbolic evaluation of Eq. (3.7a) can be written as:

المنارة للاستشارات
www.manaraa.com

$$ {}^{0}_{6}T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.7b} $$

where,

$$ r_{11} = c_1[c_{23}(c_4c_5c_6 - s_4s_5) - s_{23}s_5c_5] + s_1(s_4c_5c_6 + c_4s_6) \tag{3.7c} $$

$$ r_{21} = s_1[c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_6] - c_1(s_4c_5c_6 + c_4s_6) $$

$$ r_{31} = -s_{23}(c_4c_5c_6 - s_4s_6) - c_{23}s_5c_6 $$

$$ r_{12} = c_1[c_{23}(-c_4c_5s_6 - s_4c_6) + s_{23}s_5s_6] + s_1(c_4c_6 - s_4c_5s_6) $$

$$ r_{22} = s_1[c_{23}(-c_4c_5s_6 - s_4c_6) + s_{23}s_5s_6] - c_1(c_4c_6 - s_4c_5s_6) $$

$$ r_{32} = -s_{23}(-c_4c_5s_6 - s_4c_6) + c_{23}s_5s_6 $$

$$ r_{13} = -c_1(c_{23}c_4s_5 + s_{23}c_5) - s_1s_4s_5 $$

$$ r_{23} = -s_1(c_{23}c_4s_5 + s_{23}c_5) - c_1s_4s_5 $$

$$ r_{33} = s_{23}c_4s_5 - c_{23}c_5 $$

$$ p_x = c_1(a_2c_2 + a_3c_{23} - d_4s_{23}) - d_3s_1 $$

$$ p_y = s_1(a_2c_2 + a_3c_{23} - d_4s_{23}) + d_3c_1 $$

$$ p_z = -a_3s_{23} - a_2s_2 - d_4c_{23} $$

Eq. (3.7c) represents the forward kinematic equations of the Puma 560 manipulator. This is the set of equations used to determine the end-effector position in the Cartesian space. A similar procedure is followed to assign coordinate frames to the

المنارة للاستشارات

www.manaraa.com

sensors (laser and camera) as well as to the object of interest and the workstation. A detailed discussion of the techniques used is presented later.

3.5 General Nonlinear Robotic Model

In most practical applications of 6-DoF robot arms, the joint velocities required to achieve a predefined configuration (position and orientation) of the end-effector of the robot arm at a desired speed are obtained by linearization of the dynamic governing equation [52]. The explicit dynamic model solution of the manipulator for controlling the robot arm is avoided. However, as shown by Armstrong et al [52], an abbreviated explicit model of the Puma 560 is less computationally expensive which allows for a simplified realization. The equation of motion for the robot arm can be written in terms of the 6-dimensional vector of joint positions $q(t)$, as follows:

$$\tau = M(q)\ddot{q} + V\dot{q} + F(\dot{q},q) + G(q) \tag{3.8}$$

where,

$\tau = 6\times1$ vector of generalized input forces,

$M(q) = 6\times6$ inertia matrix,

$V = 6\times6$ viscous friction diagonal matrix,

$F(\dot{q},q) = 6\times1$ vector of Coriolis and centrifugal terms,

$G(q) = 6\times1$ vector of gravitational terms

For tracking the desired trajectories in joint space where the joint position $q(t)$ is specified, the required generalized input torques τ to control the robot arm are calculated so that all joints are able to reach the prescribed position and orientation at the desired

المنارة للاستشارات
www.manaraa.com

velocities and accelerations (if specified). Several solution schemes have been suggested to reduce the complexity of the solution to Eq. (3.8). The most commonly used technique for the linearization of (3.8) was devised by Whitney [53, 54]. This technique resolves the desired end-effector motion into the necessary joint motions reducing the complexity of the solution. This method is known as the Resolved-Rate Method which provides a numerical solution in the end-effector space.

Considering Whitney's solution scheme, the Jacobian and the Inverse Jacobian of the manipulator are required to solve the inverse kinematics problem. The position and the linear velocity components or forces components of the robot's end-effector are specified. The linear velocity components of the end-effector must be transformed into joint velocities, and then into joint positions by simple numerical integration. Figure 3.3 shows a simplified diagram of the algorithm where the input to the block diagram corresponds to the linear velocity components of the robot end-effector, [51].

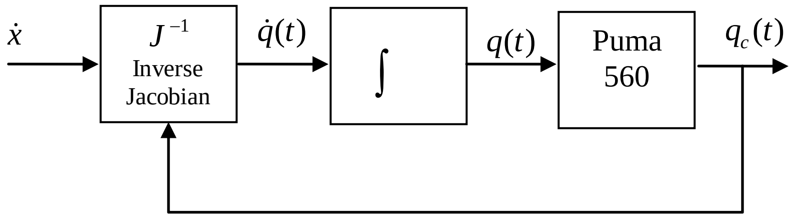


Figure 3.3 Simplified Resolved-Rate Algorithm Block Diagram

As shown in Figure 3.3, only the position vector $q(t)$ is known at this point. The 6-DoF of the Puma is controlled by six (6) brushed DC servo motors, each coupled with an encoder and a potentiometer. The current angular position of each joint can be

www.manaraa.com

obtained from the feedback signals from each encoder and potentiometer located at every joint. The required actuator torques τ are computed as a linearization feedback form of Eq. (3.8) based on the desired positions $q_d(t)$ and the desired joint rates $\dot{q}_d(t)$; i.e. the joint accelerations are not considered ($\ddot{q}_d = 0$). The computed components of Eq. (3.8) are defined as follows [55, 56]:

$\tau_c = 6 \times 1$ computed vector of generalized input forces,

$M_c(q) = 6 \times 6$ computed inertia matrix,

$V_c = 6 \times 6$ computed viscous friction diagonal matrix,

$F_c(\dot{q}, q) = 6 \times 1$ computed vector of Coriolis and centrifugal terms,

$G_c(q) = 6 \times 1$ computed vector of gravitational terms

Considering the computed values, the desired driving torque is computed as:

$$\tau_c = M_c(q)\{K_d(\dot{q}_d - \dot{q}) + K_v(q_{d.} - q)\} + V_c\dot{q} + F(\dot{q}, q) + G_c(q) \tag{3.9}$$

where K_d and K_v are the position and velocity gains, respectively. Eq. (3.9) gives an appropriate control action if $(q_d - q) = 0$. In practical implementation, there will be an error value defined as $e_q(t) = (q_d - q) \approx 0$. However, assuming that convergence is reached, then the elements of Eq. (3.9) would be equal to the actual elements in Eq. (3.8). The previous assumption results in the following set of equality constraints:

$$M_c(q) = M(q) \tag{3.10}$$

www.manaraa.com

$$V_c = V \tag{3.11}$$

$$F_c(\dot{q}, q) = F(\dot{q}, q) \tag{3.12}$$

$$G_c(q) = G(q) \tag{3.13}$$

If the constraints expressed by Eq. (3.10) to (3.13) are satisfied, then Eq. (3.9) yields to:

$$\tau = M(q)\{K_d\,(\dot{q}_d - \dot{q}) + K_v(q_{d.} - q)\} + V\dot{q} + F(\dot{q}, q) + G(q) \tag{3.14}$$

Equating (3.9) and (3.14) yields to the closed-loop system dynamics equation:

$$M(q)\{K_d\,(\dot{q}_d - \dot{q}) + K_v(q_d - q)\} = 0 \tag{3.15}$$

As can be observed in Eq. (3.15), this simplification does not include the joint accelerations, so it represents a set of independent first-order differential equations for each joint of the manipulator. The response characteristics of the systems of differential equations can be adjusted by the proper selection of the gains K_d and K_v. Eq. (3.15) can now be expressed as function of the error e_q and the error rate $\dot{e}_q$ as:

$$K_d e_q + K_v \dot{e}_q = 0 \tag{3.16}$$

Eq. (3.16) represents a linearized feedback form and it will be valid as long as the joint positions $q(t)$ converge to the desired joint positions $q_d(t)$. In this research work, the actual implementation of the manipulator's controller includes the gravitational term, $G(q)$ and the closed-loop system with a Proportional-Derivative (PD) feedback control law becomes:

المنارة للاستشارات

www.manaraa.com

$$\tau = K_d\left(\dot{q}_d - \dot{q}\right) + K_v\left(q_d - q\right) + G(q) \tag{3.17}$$

The PD controller with gravity compensation produces a global asymptotically stable closed-loop system through appropriate selection of the proportional and derivative set of gains [57] as long as the configuration of the robot arm is not singular. The calculation of the gravitational compensation terms requires the inertia values as well as the locations of the center of gravity of every link of the manipulator. Those parameters were experimentally determined by Armstrong et al [52] for the Puma 560 and are presented in Table 3.2.

The use of Lagrange's equation facilitates the derivation of the gravitational terms. The calculation of the required torques to compensate of the gravitational action will be a function of the joint-space configuration (pose) of the manipulator and the gravitational constant, g. The kinetic K_i and potential L_i energies for each link can be expressed in terms of the joint variables q_i and the link mass m_{li} located at the respective center of gravity of the link. The gravitational components will appear naturally in the final manipulator dynamics equation in the standard form given by Eq. (3.14). A detailed explanation of the procedure can be found in [52].

المنارة للاستشارات

www.manaraa.com

Table 3.2 Link Mass and Center of Gravity Locations [52]

Link i	mass (kg)	r_x (mm)	r_y (mm)	r_z (mm)
1	-	-	-	-
2	17.40	68	6	-16
3	4.80	0	-70	14
4	0.82	0	-143	14
5	0.34	0	0	0
6	0.09	0	0	32
Detached wrist	2.24	0	0	-64

In this research work, the gravitational compensation is applied to every joint of the manipulator. Using the DH parameters from Table 3.1 and the link mass and center of gravity locations from Table 3.2, the gravitational constant components g_i $(i = 1...6)$ corresponding to each joint are found to be:

$$\begin{aligned}
g_1 &= -g\left((m_{l3} + m_{l4} + m_{l5} + m_{l6})a_2 + m_{l2}\, r_{x2}\right) \\
g_2 &= g\left(m_{l3} r_{y3} - (m_{l4} + m_{l5} + m_{l6})d_4 - m_{l4}\, r_{z4}\right) \\
g_3 &= g\left(m_{l2}\right) r_{y2} \\
g_4 &= -g\, a_3 \left(m_{l4} + m_{l5} + m_{l6}\right) \\
g_5 &= -g\left(m_{l6}\right) r_{z6}
\end{aligned} \tag{3.18}$$

The gravitational terms as a function of the position vector G(q) can be obtained as follows:

$$\begin{aligned}
\tau_{g1} &= 0 \\
\tau_{g2} &= g_1 c_2 + g_2 s_{23} + g_3 s_2 + g_4 c_{23} + g_5\left(s_{23} c_5 + c_{23} c_4 s_5\right) \\
\tau_{g3} &= g_2 s_{23} + g_4 c_{23} + g_5\left(s_{23} c_5 + c_{23} c_4 s_5\right) \\
\tau_{g4} &= -g_5 s_{23} s_4 s_5 \\
\tau_{g5} &= g_5\left(c_{23} s_5 + s_{23} c_4 s_5\right) \\
\tau_{g6} &= 0
\end{aligned} \tag{3.19}$$

www.manaraa.com

Substituting all the terms in Eq. (3.19) into Eq. (3.17) gives the mathematical expression for calculating the driving torques of the manipulator in terms of the joint angle values at each time interval.

3.6 Generic Architecture for a Real-Time Robotic Controller

The components of a robotic system (robot arm, controller, sensors, user interface/input, signal conditioners, and amplifiers) must perform different activities and interchange information among different modules of the system to accomplish different desired tasks. This section describes the multithreaded PC-based implementation of a real-time controller for a haptically interfaced 6-DoF robot arm. To accomplish this, the feedback signals from the haptic device as well as the sensory information must be transferred to the arm controller in real-time in a deterministic fashion by the host computer.

The nature of this application demands a real-time response in order to be usable for enhancing the manipulation capabilities of users in cases where the haptic interface provides force feedback and is an integral part of the robot arm controller. For this to be possible, it is not acceptable to have delays in the haptic response. For example, it is not acceptable that the haptic device tip penetrates the rigid body rendered in the graphical scene during a haptic cycle [58]. In the other hand, the integration of sensory-assisted functions, SAF's, to assist the user's motion to execute a particular task requires the sensor datasets to be also available in a deterministic fashion even though the sensor update rates are smaller than the robotic control signals. In the case of humans, it has been determined that the transmission of realistic sensation of touch occurs at frequencies

المنارة للاستشارات
www.manaraa.com

over 1.0Khz [1, 3]. This corresponds to what was previously stated, the update rate of the feedback signals from the haptic device must be at least 1000Hz (1.0Khz) in order to generate rigid body sensations in the user's hands [1, 2].

An additional constraint of this type of application is the definition of the limits of the achievable stiffness in the environment for stable control of the haptic interface [3]. The platform implemented must ensure that the transmitted signals and the computed output torques are not delayed by a variable amount of time depending on the CPU system loads. To satisfy the forementioned requirements for any haptic control system for telerobotics applications, the following threads were defined:

1. The determination of the target position (in Joint or Cartesian space) from the haptic device interface,
2. The computation of the joint angles to reach the desired position,
3. A trajectory generation thread which computes position set-point commands, and
4. The computation of the torques (a PD software controller with gravity compensation) required to drive the motors (manipulator control program) based on the positional error signals. The error-based control signals of the robot arm (used for Joint-Torque actuation control) are computed at the same update rate as the haptic signals.

It must be taken into account that since there are multiple threads running at the same time, there is a chance of conflict when accessing shared memory or data structures. For example, the case when one thread is writing data to the memory and a second thread is reading from that same memory. In order to avoid data corruption ("mutual exclusion"), a synchronization method is required to ensure exclusive access to shared

المنارة للاستشارات
www.manaraa.com

resources. QNX RTOS was chosen for this platform because it is a fully compliant Portable Operating System Interface (POSIX) operating system and it provides multiple synchronization primitives, such as mutexes, real-time semaphores, conditional variables, joining, and barriers [50]. The POSIX standard is maintained by the IEEE and it is recognized by ISO and ANSI. All of these primitives implement mutual exclusion but have varying performance benefits and usage models [59]. The synchronization mechanism implemented is based on real-time semaphore signals and message passing, [50, 59].

Figure 3.4 shows the multithreaded architecture of the telerobotic control system. As shown, only the robotic controller side of the design is illustrated in this figure.

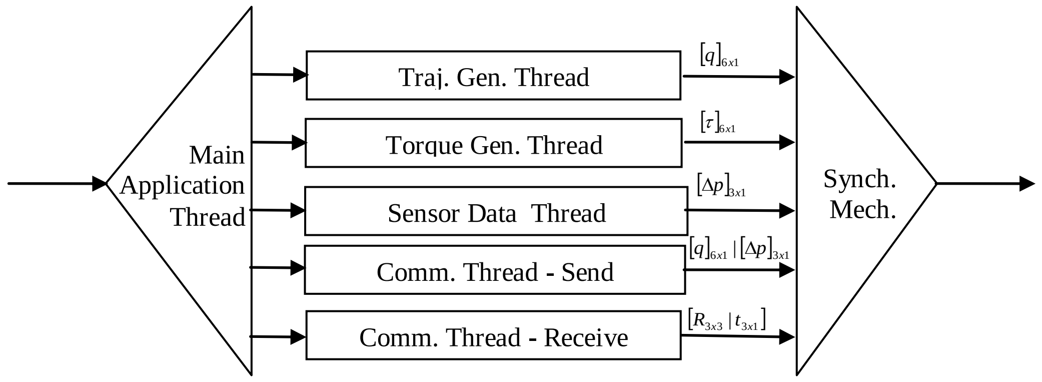


Figure 3.4 Multithreaded Robot Arm Controller Architecture

The telerobotic control system implemented in this work requires the interaction of three fundamental components or subsystems: sensory, control, and actuation subsystems. The sensory subsystem handles the measurements of physical quantities and “state” of the environment. At this level, the camera and the laser input, the joint encoder readings, as well as the haptic interface information, are gathered and processed. The

www.manaraa.com

control subsystem uses the sensors input to compute an action command to drive the actuators. The actuation subsystem (motors and transmission mechanisms) is responsible for physically changing the manipulator position and orientation. In order to control the robotic system and to achieve a desired configuration, the sensing and the corresponding commanded actuation must meet strict timing constraints. In other words, the scheduled activities of the different subsystems must not be delayed before a relatively short deadline for stable control of the robot arm. So, consistency and predictability are fundamental requirements for the sensor-based telerobotic control system to be "controllable".

The generic architecture described in the present work is a multithreaded implementation, where the shared resources (critical section or region) are accessed by multiple threads concurrently. The QNX thread programming model allows multiple threads to access the CPU simultaneously with priority-based scheduling. This means that the kernel will block the threads based on priorities and scheduling policies defined for every thread created, [50]. The priority levels are defined by QNX from 0 as the lowest priority to 63 as the highest. These priority levels are strictly enforced by the operating system. This way, the thread with the highest priority that is ready to run will be running until it is blocked. At each priority, the threads in QNX are scheduled according to one of the available policies (First-Input-First-Output, FIFO, and Round-Robin, RR). These policies are only activated when more than one thread is ready to run at the same priority.

Figure 3.5 shows a diagram of the data flow. As illustrated, threads T1, T3, and T4 are at the highest priority which means that they will share the CPU based on the

www.manaraa.com

thread's scheduling policy assigned to each particular thread, [50]. The scheduler selects the next thread to run by looking at the priority assigned to the thread in the READY state. The thread with the highest priority that's at the head of its priority's queue is selected to run. For instance, as shown in Figure 3.5, T1 is "active" and "READY" to run because it has the highest priority and it is at the "head of the queue". As stated before, the scheduling policy will be applied only when threads with the same priority are ready to run and a decision is required.

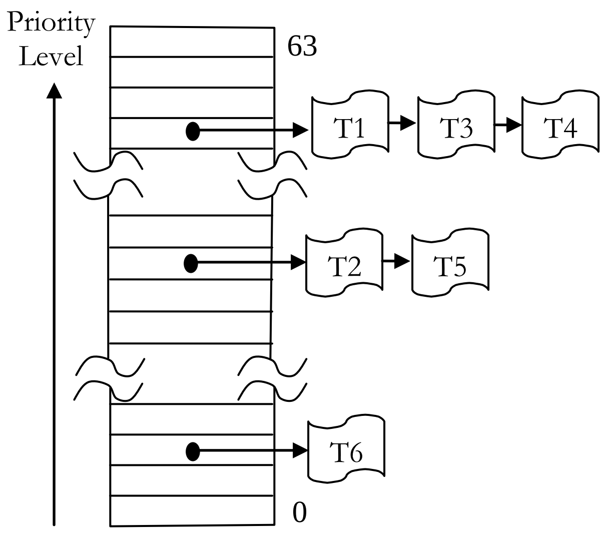


Figure 3.5 "Ready/Blocked" States, Adapted from [50]

As multiple threads are running at the same time, there is a possibility of data corruption. In this research work, semaphore signals (a variable that indicates the status of a shared resource) and message passing [50] is used as the synchronization mechanism to prevent data corruption. The semaphore signaling mechanism used for synchronization is set up before starting any of the implemented threads shown in Figure 3.4. If any previously defined thread is currently blocked waiting for the semaphore, the

www.manaraa.com

next thread to be unblocked is determined in accordance with the scheduling policy defined for the blocked thread. If the situation arises where multiple threads are blocked waiting for the semaphore, then the highest priority thread that has been waiting the longest is unblocked; i.e. access is granted based on priority and scheduling policy.

In general, when the supervisory control scheme (“human-in-the-loop”) is used, the sensory information can be used for adjusting the trajectory of the end-effector of the robot arm to guide the user’s motion through a haptic interface. In order to combine the camera, the laser, encoder readings, and haptic sensory inputs to assist the user during task execution, the telerobotic system must meet tightly defined response constraints to avoid instability caused by time delays such as oscillations, collisions, and the loss of rigid body sensations while touching objects. The correctness of the system response depends not only on the logical result of computations, but also on the time at which the results are produced [7]. At the control level of the telerobotic system, the different computational processes to execute a particular motion in 3D space, such as trajectory following and the required torque computations need to interchange information. In this work, multiple threads were designed to handle the signals of the robot controller as well as the visual and haptic information.

The following is a summary of the key aspects of the generic architecture for the real-time telerobotic controller proposed in this work. The real-time application design enables the possibility to communicate between different running threads. This allows the different subsystems to interact with each other and share the same data structure. Even though this inter-process communication is a highly desirable design feature of the telerobotic system, there might be a chance of data corruption when a running thread

المنارة للاستشارات
www.manaraa.com

attempts to change data while another thread is using the same data. For instance, when the "Trajectory Generation Thread" is accessing its data structure for writing and the "Torque Generation Thread" is accessing the same data structure for reading. In such case, the concept of "mutual exclusion" of the data can be accomplished in RTOS's by the use of real-time semaphores (a variable that indicates the status of a shared resource) without affecting the responsiveness of the operating system [50]. Another important aspect is the preemptive scheduling of threads based on predefined priority level of each thread.

Figure 3.6 illustrates the integration of the different subsystems encompassing the system architecture. As shown, the system conforms to a modular design which facilitates scalability and application of the multithreading programming paradigms to other telerobotic applications in rehabilitation, training, surgery, defense, research, device testing, and assistive technology solutions.

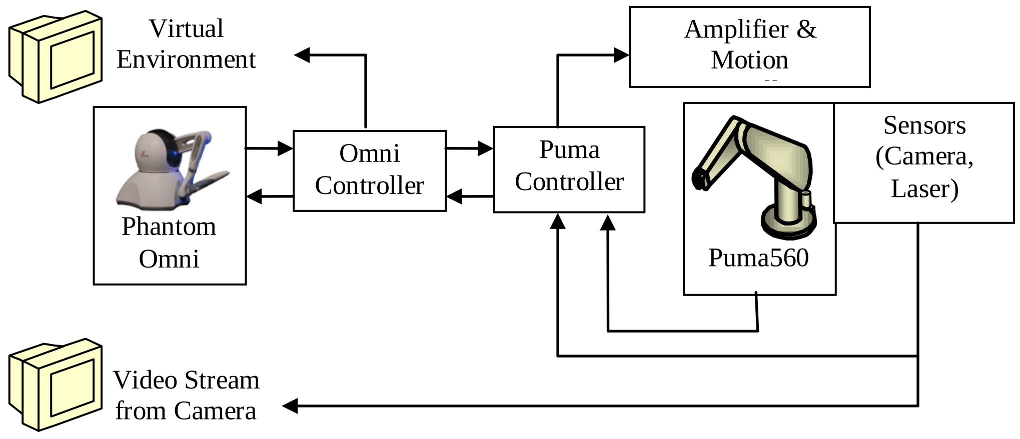


Figure 3.6 Block Diagram of the System Architecture

www.manaraa.com

3.7 Cartesian Trajectory Generation Thread

The trajectory generation thread solves the inverse kinematic equations of the robotic arm for non-redundant robot arms and an inverse Jacobian approach for redundant robot arms, as discussed later this section. For the case of the Puma 560, both implementations are available in the proposed system. The inverse kinematics solution gives the joint values corresponding to positions and orientations of the end-effector. For the non-redundant case, the trajectory generation thread is composed of the following steps:

1. At every time step, define $t = t + \Delta t$.
2. Obtain the position and orientation of the end-effector corresponding to the desired trajectory function (a straight-line, for example) as explained below.
3. Solve the inverse kinematic problem to obtain the joint values corresponding to the position and orientation obtained in (2).
4. Compute the driving torque based on the controller scheme being used. In this particular implementation a Proportional-Derivative-Plus-Gravitational Compensation.
5. Send the computed torques to the robotic controller.
6. Repeat the loop until the final destination is reached.

The straight line motion in the trajectory generation thread is accomplished by computing the total transformation required to move the robotic arm from point *i* (defined as the initial) to *j* (defined as the destination). Once the total transformation is calculated, it must be divided into smaller segments to obtain the intermediate points for a smooth

المنارة للاستشارات

www.manaraa.com

transition. The total transformation, T, defined between the initial position and orientation, T_i and the final position and orientation T_f is derived as follows:

$$T_f = T_i T \tag{3.20}$$

Pre-multiplying by the inverse of T_i yields to:

$$T_i^{-1} T_f = T_i^{-1} T_i T \tag{3.21}$$

So, the required total transformation between points A and B is given as:

$$T = T_i^{-1} T_f \tag{3.22}$$

In order to compute the intermediate points, the total transformation can be decomposed into a translation for moving the origin of the initial end-effector frame to the destination frame and a rotation about a single axis $\hat{\kappa}$ to align the end-effector frame to the desired goal frame. In the literature, this method is known as the single-axis rotation method [60]. In the method, the translation component can be easily divided into smaller linear segments. However, the rotational components are nonlinear and a procedure to ensure orthogonality of the axes is required as well as provisions to avoid representational singularities (See Appendix B).

المنارة للاستشارات
www.manaraa.com

3.8 Resolved-Rate Thread

This thread deals with implementation of the resolved-rate algorithm described in [53, 54, 56]. The joint velocities are determined from the Cartesian velocities as follows:

$$\dot{\theta} = J^{+}\dot{X} \tag{3.23}$$

where,

$\dot{\theta} = 6\times 1$ desired vector of joint velocities,

$\dot{X} = 6\times 1$: commanded vector of Cartesian velocities (from the haptic device interface)

$J^{+} = 6\times 6$: is the pseudo-inverse of the Jacobian of the robot arm.

The pseudo-inverse J^{+} is given by $J^{+} = J^{T}\left(JJ^{T}\right)^{-1}$. However, rather than directly performing a pseudo-inverse calculation, the following relationship is defined:

$$\dot{X} = JJ^{T}y \tag{3.24}$$

The $y = 6\times 1$vector of independent coefficients can be solved with a LU decomposition method avoiding the computationally expensive process of the inverse of matrix defined as $\left(JJ^{T}\right)^{-1}$. Once the vector y is known, the required angle rates $\dot{\theta}$ are obtained from:

$$\dot{\theta} = J^{T}y \tag{3.25}$$

The resulting $\dot{\theta}$ is the least-norm joint velocity vector (or joint rate) which produces the required end-effector Cartesian velocity vector $\dot{X}$, [56]. The numerical techniques associated with the calculation of resolved rate algorithm are all implemented in C++ to run under QNX. Figure 3.7 illustrates the process.

المنارة للاستشارات

www.manaraa.com

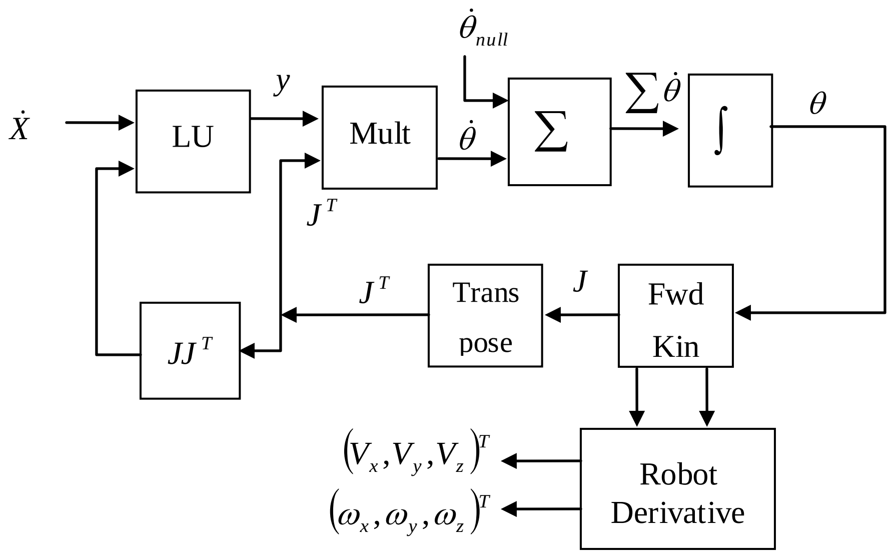


Figure 3.7 Cartesian to Joint Space Conversion in the Robotic Workspace

3.9 Sensory Information Threads

Sensors give the robot the ability to interact with an unknown or unstructured environment [61]. In practice, the robot will not be able to "view" the entire environment. If the workspace is defined as a matrix of a determined size, the robot arm will reach only a set of local matrix cells around the robot. Sensors return information about their environment by physically interacting with the real world. The nature of this interaction may be "passive" or "active". Passive sensors simply record emissions already present in the environment. Active sensors emit a signal and measure how the environment modifies the signal. In this research work, a CCD camera and a laser range finder are passive-type sensors used for the location of objects of interest. The sensory information threads are in charge of data acquisition and post-processing of the sensory datasets. It consists of six (6) concurrent threads with different update rates of their respective data structures:

www.manaraa.com

1. The collection of image information and processing: This thread is responsible for capturing the images and image processing (binarization, edge detection, and feature extraction).
2. The laser ranger sensor thread: This thread reads the analog signals coming from the laser sensor. The output from laser finder is a voltage value which is proportional to the range or distance measured. To have access to this analog signal from a PC, it needs to be calibrated and converted to digital signals using an Analog to Digital Converter as described in Appendix G.
3. The haptic Servo-loop thread: This thread implements the haptic effects (spring-force model, spring-damper model, Coulomb's friction, among others) in simulation. This thread requires an update rate over 1000Hz for a realistic sensation of the particular effect through the actuators of the Phantom Omni. The differential transformation matrices (position and orientation) corresponding to the haptic tip are updated at this rate.
4. The collision-detection thread (user and virtual objects interaction)
5. The graphic thread: displays the 3D virtual reality model on the screen and communicates with the haptic servo loop to update the display accordingly.
6. The communication thread: implements a low-level User Datagram Protocol (UDP) packet protocol with provision for data losses and order of arrival of the sensory datasets.

These threads are run as six (6) separate threads concurrently or simultaneously, but with different update rates of their respective data structures. The sensory datasets

www.manaraa.com

fusion as well as the velocity and differential transformations of the haptic end effector is then transferred to the manipulator controller. The QNX software design uses a scheduled thread for communication. This communication thread consists of a low-level network protocol based on UDP packets. The UDP protocol is flexible in its data structure, it can be extended to prevent data losses, ensure the order of arrival of the data transmitted and has reduced latency. These properties are desirable for transmission of data from diverse sensors. In this particular implementation, a single packet contains the data fusion from the visual and the laser range finder information. The design takes into account that datasets could be sent to multiple machines at once (for physical and virtual reality simulations, for example) by using the multicasting and broadcasting properties of the UDP transmission protocol. Due to the connectionless nature of the UDP protocol and its disregard for network congestion, the derived protocol implements programmatic features to assure the order of arrival of the data and mechanisms to handle data loses, if any.

3.10 Summary

In this chapter, the distinctive features of real-time operating system and real-time applications are presented in relation to the multithreading tasks of the telerobotic system. The forward kinematics of the 6-DoF manipulator is formulated in terms of the homogenous transformations and the Denavit-Hartenberg (DH) parameters. The inverse kinematic formulations are developed using Whitney's resolved rate approach in order to make the solution extensible to redundant robot arms. A linearized mathematical model of the control system is described in terms of the error signals between the actual

المنارة للاستشارات
www.manaraa.com

positions and the desired positions with gravitational compensation. The implemented multi-threading approach is explained and the threads defined for executing a particular motion, the trajectory following, sensory data fusion, as well as the torques required to drive the arm are discussed. The multiple threads designed to handle the signals of the robot controller as well as the visual and haptic data fusion with provisions for inter-processing communication; priority-based execution and data corruption avoidance are explained.

المنارة للاستشارات
www.manaraa.com

Chapter 4

Sensor-Based Assistance, Autonomous and Teleoperation Control

4.1 Introduction

In general, a telerobotic system consists of a master user-input device operated by a human and the slave robot placed at a remote location and controlled using a supervisory control scheme. This form of teleoperation requires the human to be in the control loop at all times. Autonomous and teleoperation control modes enable the system to combine human high level decisions with the computer-based intelligence control. The idea of incorporating sensor-based assistance to the system is to facilitate task executions and to remove the skills required for operating the system. This work focuses on enhancing the capabilities of users using intelligent autonomous and teleoperation (telerobotic) control to combine human high level decisions with computer intelligence on a hard real-time master-slave system that will help users to execute different tasks in an easier and faster manner. The human decision making component comes from locating the target objects in the environment using simple sensors and selecting a combination of different modes of operation like the autonomous control, scaled, virtual fixture based, position or velocity based teleoperation control modes.

In this chapter, the concept of assist function is defined in relation to the basic haptic parameters and the control law equations required to determine the intended path based on the master‘s end-effector position and sensory input are outlined. The different operation modes derived from the implementation of the autonomous control mode and

المنارة للاستشارات
www.manaraa.com

teleoperation control scheme are also described. The concept of the centroid of the object used in the derivation of the scaled and virtual fixture constraints is assumed to be known and the details of its determination will be presented in Chapter 5.

4.2 Sensor-Based Telerobotic Control Theory

The sensor-based assistance and telerobotic control implementations depend on either position or velocity control variables. For position-based assistance a simple form is scaling, in which the motion of the slave's end-effector is scaled up in the desired direction and scaled down in any other direction. Similarly, in the case of velocity assistance, the velocity is scaled according to whether the motion in a particular direction is serving to further accomplishing the desired effect of the motion, for example, when moving towards a target object. For instance, the 3D Cartesian based mapping from master to slave makes it very easy and quick for the users to point to objects in the environment with the laser range finder. Once the object is located by pointing the laser, it is locked by the system by the press of a key and then the slave can proceed towards the object in automatic mode or by teleoperation.

4.2.1 Autonomous Control Mode

Before the activation of the autonomous control mode, the user points the laser to an object in the environment by teleoperating the slave robot arm. Then the user selects the automatic mode option to move the gripper towards the object along the linear trajectory (line of sight) generated by the laser as shown in the Figure 4.1. After reaching

المنارة للاستشارات

www.manaraa.com

a certain threshold distance, the arm moves along a secondary trajectory to account for the laser offset distance from the gripper as shown in Figure 4.1.

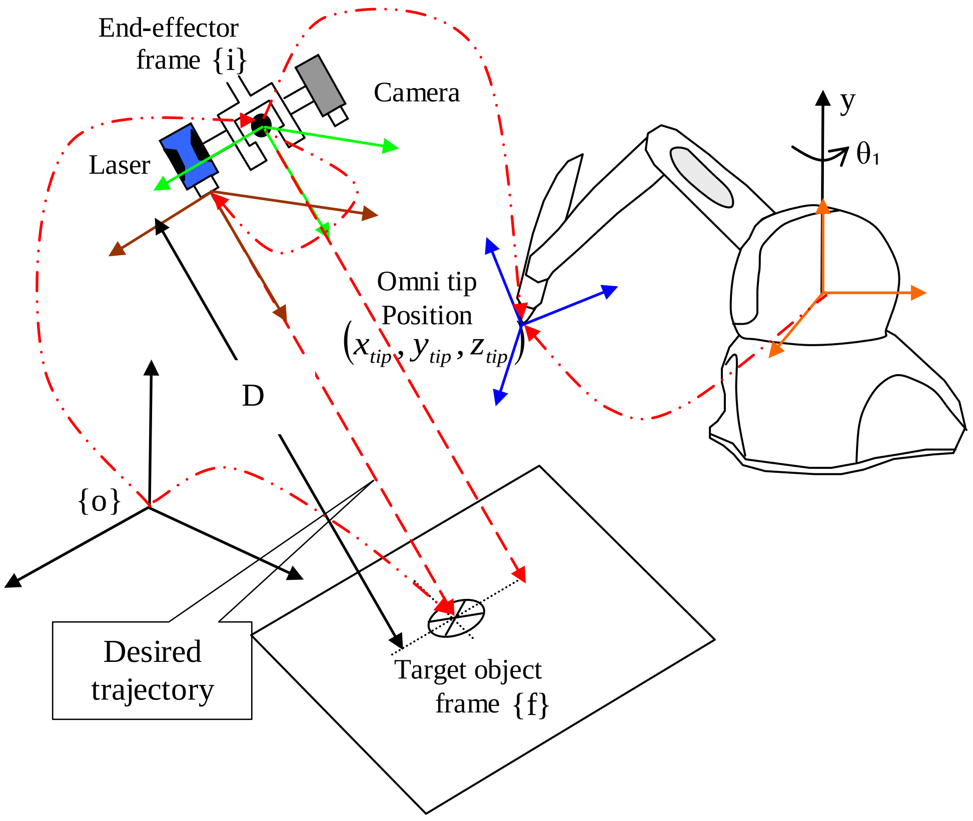


Figure 4.1 Conceptual Representation of Autonomous Control Mode

As explained in Chapter 3, the resolved-rate approach for Cartesian motion is used to compute required joint velocities from the Cartesian velocities of the end-effector. When the user selects the ‘Automatic Mode’, a linear trajectory in the form of differential transformation matrices at each of the sampling points is computed between the current end-effector position and the target object position in hand coordinates. Then, the resulting transformations are transformed to base coordinates before their use in the resolved-rate algorithm.

www.manaraa.com

If the transformation of the current end effector position with respect to the base, obtained from the solution of the forward kinematics of the manipulator, is denoted by ${}^{0}_{i}T$, then the transformation of the target object with respect to the base ${}^{0}_{f}T$ can be computed by the following operation:

$$ {}^{0}_{f}T = {}^{0}_{i}T * {}^{i}_{f}T \tag{4.1}$$

where ${}^{i}_{f}T$ is given by Eq. (4.2) and D is the measured distance from the laser.

$$ {}^{i}_{f}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & D \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

The equivalent angle-axis method [22] is used for obtaining the rotation part, and linear interpolation to obtain the linear part of transformations at the sampling points or "via points". A Cartesian velocity vector, V, is computed from two consecutive sampling transforms taken from the set above every 200 Hz which is the refresh rate of the trajectory generation thread, as explained before. If T_1 and T_2 are two consecutive transformations defined as $T_1 = [\overline{n}_1 \quad \overline{o}_1 \quad \overline{a}_1 \quad \overline{p}_1]$ and $T_2 = [\overline{n}_2 \quad \overline{o}_2 \quad \overline{a}_2 \quad \overline{p}_2]$, then the velocity "screw" approximation can be used to obtain the Cartesian velocity vector V as follows:

$$ V = [\overline{v} \quad \overline{w}]^T \tag{4.3}$$

where

$$ \overline{v} = [\overline{p}_2 - \overline{p}_1] \tag{4.4}$$

and

www.manaraa.com

$$\overline{w} = \left[\frac{1}{2} \left(\overline{n}_1 \times \overline{n}_2 + \overline{o}_1 \times \overline{o}_2 + \overline{a}_1 \times \overline{a}_2 \right) \right] \tag{4.5}$$

The required joint angle rates are computed using the inverse of the Jacobian of the manipulator as follows:

$$\dot{q} = J_0^{-1} * V \tag{4.6}$$

After integration of the joint rates, the current joint angles are sent to the "Torque Generation" thread to calculate joint torques to drive the arm.

4.2.2 Position-Based Teleoperation Control Mode

Position-based teleoperation is the default control mode of the telerobotic system. In this mode, as the Phantom Omni is moved in its workspace by the user, its transformation matrices are computed by solving the forward kinematics problem, and mapped to the PUMA base frame. The differential rotations, *dR*, and differential translations, *dP*, of the Phantom Omni are computed between every two consecutive sampling points by (4.7) and (4.8), respectively.

$$dR = R_i^T * R_{i+1} \tag{4.7}$$

$$dP = P_{i+1} - P_i \tag{4.8}$$

Knowing the current PUMA POSE, T_{P1}, the new end-effector POSE of the PUMA is computed as:

$$T_{P2} = T_{P1} * \begin{bmatrix} dR \,|\, dP \\ \underline{0} \,|\, 1 \end{bmatrix} \tag{4.9}$$

www.manaraa.com

For teleoperation, a closed-form solution of the inverse kinematics problem is used to yield the joint angles which are then sent to the torque generator for computing joint torques.

4.2.3 Velocity-Based Teleoperation Control Mode

In this mode of teleoperation, the Phantom Omni position determines the PUMA end-effector speed and direction. In other words, when velocity control is used, the PUMA end-effector speed changes proportionally to the Phantom Omni changing position. When the specified velocity is reached, it is maintained until the command from the Omni is changed. Under velocity control mode, the user will move the Omni's end-effector once to select a direction and speed for the Puma end-effector. Then, the user will hold the Omni's end-effector steady until the gripper mounted on the PUMA is close to the target object, then move the Omni's end-effector back to its initial position in order to stop close to the target.

The implementation of the velocity-based teleoperation is similar to the position-based teleoperation mode except that the differential rotations *dR* and differential translations *dP* of the Omni are computed between the initial Omni stylus position when its button is pushed, and its current position. This way, the Omni pen behaves like a joystick; the further the joystick moves away from the center, the faster the PUMA end-effector moves. This is also suitable to wheelchair bound users who are accustomed to using a wheelchair for mobility.

In this mode, the Phantom Omni end-effector transformation is recorded when the user clicks the stylus button. The recorded transformation is referred to as in (4.10):

المنارة للاستشارات
www.manaraa.com

$$T^{ref} = \begin{bmatrix} dR^{ref} & \mid P^{ref} \\ \underline{0} & \mid 1 \end{bmatrix} \tag{4.10}$$

Again, as the Omni's stylus is moved in its workspace by the user, the current transformations are sent to the PUMA controller and are mapped to the PUMA base frame. The differential translation is computed as:

$$dP = \left(P_2 - P^{ref}\right) * V_{factor} * dt \tag{4.11}$$

where

V_{factor} = a constant velocity factor and,

dt = the real time clock refresh rate.

This means that the farther the Omni pen is from the start position, the faster the PUMA moves as P^{ref} is constant and only P_2 is updated at the cycle refresh rate. The differential rotation *dR* is computed as:

$$dR = \left[R^{ref}\right]^T * factor_R * R_2 \tag{4.12}$$

where $\left[R^{ref\,T}\right]$ corresponds to the transpose of R_{ref} and $factor_R$ is a scaling rotation factor. Then, small increments of *dR* are computed from equivalent angle-axis method and are used to transform R^{ref} at the cycle refresh rate to yield new rotational components of the PUMA end-effector transformation. These new transformations are computed in the same way as in position-based teleoperation and the inverse kinematics yields joint angles at the cycle refresh rate, as explained in Chapter 3.

المنارة للاستشارات
www.manaraa.com

4.2.4 Scaled Teleoperation

Scaled teleoperation is used to scale up or down the user's input for assistance and create virtual constraint using the sensory data. After the user selects the target object from the environment by pointing the laser, the reference trajectory vector is calculated. As the user moves the Phantom Omni in its workspace, the translation vectors $\bar{k}_{via}$ are computed from the Omni's tip transformations and sent to the PUMA controller at every cycle step. If P_i and P_{i+1} are the translation vectors of the homogenous transformations of two consecutive Omni's tip points, then the translation vector $\bar{k}_{via} = \bar{P}_{i+1} - \bar{P}_i$ can be projected on the reference vector $\bar{k}$ to obtain a new vector $\bar{P}$ as follows:

$$\bar{P} = \frac{\left(\bar{k} \bullet \bar{k}_{via}\right)}{\left|\bar{k}\right|}\bar{k} \tag{4.13}$$

The projected vector resulting from (4.13) is then scaled up by multiplying it by a scaling matrix $Kscale$ given by:

$$\bar{P}_n = \begin{bmatrix} KscaleX & 0 & 0 \\ 0 & KscaleY & 0 \\ 0 & 0 & KscaleZ \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} \tag{4.14}$$

Similarly, the projections of the current translation vectors are determined on the other two axes perpendicular to the reference vector $\bar{k}$. However, the components of these vectors are scaled down. As the computations continue, $\bar{P}_n$ becomes the new differential translation vector computed every cycle. The inverse kinematics on the new transformation yields the new joint angles that are sent to the torque generator as before.

المنارة للاستشارات
www.manaraa.com

4.2.5 Virtual Fixture Teleoperation

The virtual fixture constraints are created by completely constraining the PUMA motion along the reference trajectory vector $\bar{k}$ locked by the laser. This is done by scaling up the components of the current projected vector $\bar{P}$ on the reference vector $\bar{k}$ and scaling down to zero the components of the current projected vector $\bar{P}$ on the axes perpendicular to $\bar{k}$. At the same time, the orientation of the PUMA end-effector frame is maintained constant throughout the teleoperation. This way the user's motion is completely constrained in the Cartesian space except along the axis parallel to the desired trajectory. The differential translation vectors to be sent to the PUMA are computed in a manner similar to the Scaled Teleoperation discussed in 4.2.3, keeping the rotation fixed and the new transformations yield joint angles at the cycle refresh rate to drive the PUMA robot arm**.**

4.3 The Phantom Omni Haptic Interface

A haptic interface, such as the Phantom Omni, has sensors to measure the (6 x 1) vector corresponding to the position and orientation of its end-effector (3 rotations and 3 translations) as well as the built-in 3-DoF force feedback (F_x, F_y, F_z)capabilities. The haptic device used in this work is manufactured by SensAble Technologies® and it is shown in Figure 4.2.

The positional feedback is obtained from the encoders placed at the motors and the force measurements are obtained from the actuators of the Phantom Omni interface. This information can be manipulated to express the assistive forces not just as function of the end-effector position of the Phantom Omni (also known as the stylus or thimble), but

www.manaraa.com

also as a combination of the latter and external visual information provided by sensors such as a camera and a laser range finder. Assuming that there is an object of interest in the field of view of the user, when the user points to the object with the laser, the line of sight (LoS), which passes through the centroid feature of the object or region of interest and the manipulator's end-effector, provides a visual indication of its location with respect to a fixed 3-D world reference frame. On the other hand, if the object of interest is partially or totally occluded from the user's point of view, the sensors (camera and laser range finder) can provide the location of the centroid. In this case, the "LoS" depends on the robot-mounted camera's position in space (known as the camera frame), the distance and direction of sight. In practice, there will be measurement errors between the desired position and orientation and the user's input interacting with the system. These error signals can be used to compute force constraints for correcting the deviations from the intended path and for guiding the user towards the goal.

As previously stated, the Phantom Omni shown in Figure 4.2 provides six (6) positional degree-of-freedom inputs and three (3) force degree-of-freedom output (See Appendix F). The Omni model allows users to have the "sensation of touch" of virtual objects by means of the forces transmitted to the users through the actuators mounted on the device. It allows for the control of the x, y, and z linear components of the feedback force, but does not allow for torsional feedback when users rotate the stylus. The stylus has two buttons (white and blue) such that it can be used as a mouse for "click and drag", for example.

المنارة للاستشارات
www.manaraa.com

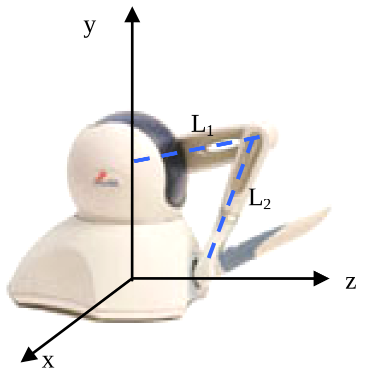


Figure 4.2 Phantom Omni Haptic Device

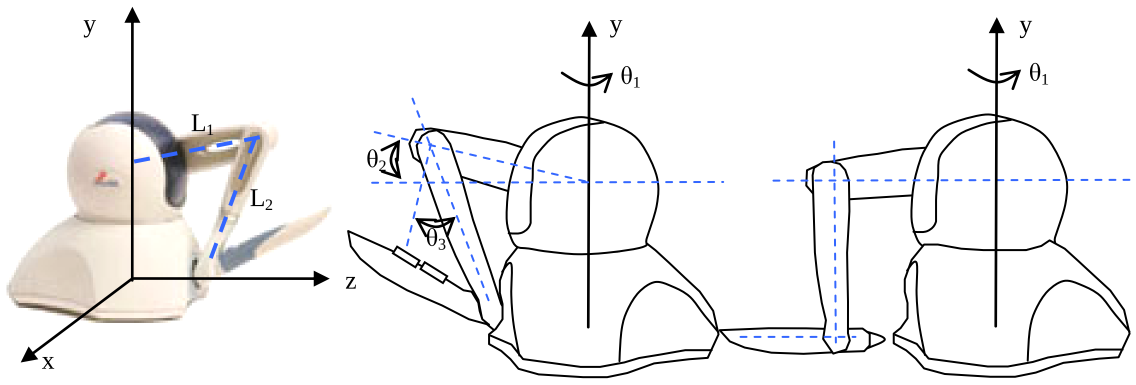
The Phantom Omni software uses the OpenHaptics software development kit (SDK) that runs on Windows XP OS. The OpenHaptics SDK consists of a set of two libraries known as the HDAPI and HLAPI. The HLAPI is a high-level library for haptics scene rendering. It is best suited for adding haptic interactions to existing OpenGL graphics applications. On the other hand, the HDAPI provides access to low-level haptic functions to handle direct force rendering to the actuators of the haptic interface. The type of feedback force rendered by the haptic device can be time dependant, motion dependant, or a combination of both. In this work, the motion dependant feedback combined with the concept of the sensor-based assist functions is used to control the six (6) Puma 560 robot arm in both, joint and Cartesian spaces.

4.4 Joint and Cartesian Control through the Haptic Interface

The Puma 560 robot arm can be controlled in joint and Cartesian spaces. Joint space haptic control means that the six (6) joints of the Phantom Omni are mapped to the corresponding joint angles of the robot arm. The forward kinematic equations of the haptic and the robot arm are used at this point to obtain a set of joint angles. After

المنارة للاستشارات
www.manaraa.com

mapping, the manipulator's controller is directed to drive the robot arm to the appropriate configuration. Figure 4.3 (c) shows the zero configuration position of the Phantom Omni. When the device is placed as shown in (c), the first three joint angles $(\theta_1, \theta_2, \theta_3)$ are zero. The gimbals' angles of the device are not shown in this configuration. On the other hand, Cartesian space haptic control deals with the determination of the joint angle values to place the manipulator at a desired position and orientation at the specified velocity. The input velocity components are provided by the haptic device, as shown in Figure 4.4.



(a) Phantom Omni (b) Measured Joint Angles (c) Zero Configuration

Figure 4.3 Phantom Omni Reference Configurations

4.5 Telerobotic Control System

The control strategy is a form of generalized bilateral control, which maps positions and velocity components between the haptic workspace and the Puma 560 workspace [17]. Figure 4.4 shows a block diagram of the control strategy where the linear velocity components of the Omni's tip are mapped to the linear velocity of the robot arm through the Jacobian J_u. As shown, the inverse of the Jacobian J_u^{-1} is not calculated directly (through the inverse or pseudo-inverse methods). Instead, the

المنارة للاستشارات
www.manaraa.com

calculation is performed following the procedure illustrated in section 3.5. This approach provides an improvement to the computational efficiency of the control strategy algorithm.

When joint space control is used, the direct measurements from the optical encoders mounted on the haptic device are used to determine the joint angles. The corresponding transformation matrices are then used to represent the haptic's reference frame relative to the manipulator's reference frame. Given the numerical values of the haptic joint angles is relatively easy to map to the manipulator's reference frames.

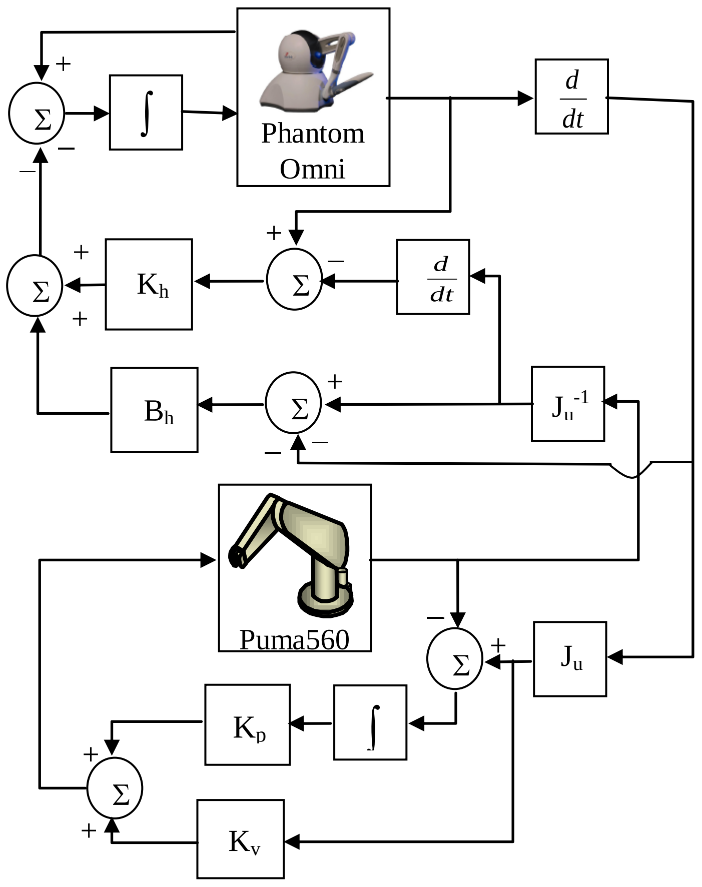


Figure 4.4 Telerobotics System Block Diagram

www.manaraa.com

4.6 Indexing with the Haptic Device

The kinematics of the Phantom Omni is very different from the robot arm kinematics that it is controlling. A technique known as “indexing” is used to extend the workspace of the haptic-manipulator interface. The most appropriate way to implement “indexing” is in Cartesian space. The stylus buttons are used for the user interaction, as follows: With the white button, the user can only “drag and drop” the virtual object on the screen, just like a standard mouse, to place the virtual object away from the limits of the workspace or to re-position the stylus to a more comfortable orientation. On the other hand, the blue button is used to re-engage the motion of the manipulator through the Phantom Omni interface. The implementation of switching between these two “states” in real-time is a challenge because, if it is not done predictably, and/or the commanded control signals from the haptic are delayed, the telerobotic system can go out of control or automatically shutdown. This safety feature is built in the hardware of the manipulator’s controller in the form of a “watchdog” timer. In addition, the software controller is designed to expect a specified difference between the current and the next commanded configuration of the manipulator. If this difference is outside the specified range, the system is shutdown.

4.7 Assistance Function (SAF) Concept

As previously mentioned, the haptic interface allows the user to have the "sensation of touch" of virtual objects through time dependant, motion dependant or a combination of both feedback forces. The idea of combining those types of forces with “force assistance” along a trajectory serves the purpose of augmenting the user’s

المنارة للاستشارات
www.manaraa.com

dexterity by scaling or by imposing virtual constraints. Also, attractive or repulsive potential fields can be defined as virtual constraints that are implemented in the haptic control software to modify the control action provided by the actuators of the haptic interface, [24].

As shown in Figure 4.5, the SAF constrains the motion of the robot arm to a desired linear path by constraining the robot end-effector motion along a line defined between the initial position of the manipulator and the position of the goal point, both defined in Cartesian space. This way, the calculation of the SAF is based on the projected line from the end-effector of the manipulator to the intended destination of the user defined by “pointing” to the object of interest or target. In this discussion, it is assumed that the location of the centroid that the user is pointing to is known for the development of the assist function equations. The required computations to identify the position and orientation of an object in the 3D space are the topic of the next chapter where the centroid location in Cartesian coordinates is the result of the data fusion of the optical sensors, camera and laser.

A common application of the assist function concept results from the situation where the object of interest is partially or totally occluded from the user’s point of view, but it is still visible from the sensors point of view (camera and laser range finder combined model). In this situation, the sensors can provide the location of the centroid from the images of the object captured by the vision system, the image processing techniques (binarization, edge detection, and feature extraction), and the inverse mapping solution. Another application results from the possibility that the user was shaking, due to tremor illness, for example, and was unable to point the laser range finder precisely on

www.manaraa.com

the object of interest. In this case, the camera information can be used to determine the location of the centroid of the object and the "offset" can be computed to compensate the erroneous user input. During the execution of a task, the user is provided with position and velocity based control schemes as well as autonomous control with the possibility of switching between them. For instance, the user may choose to approach the target object in autonomous mode and then switch from autonomous to regular teleoperation for fine tuning the orientation of the end-effector before grasping. Any combination between regular, scaled, and virtual fixture modes can be selected by the user to complete the task.

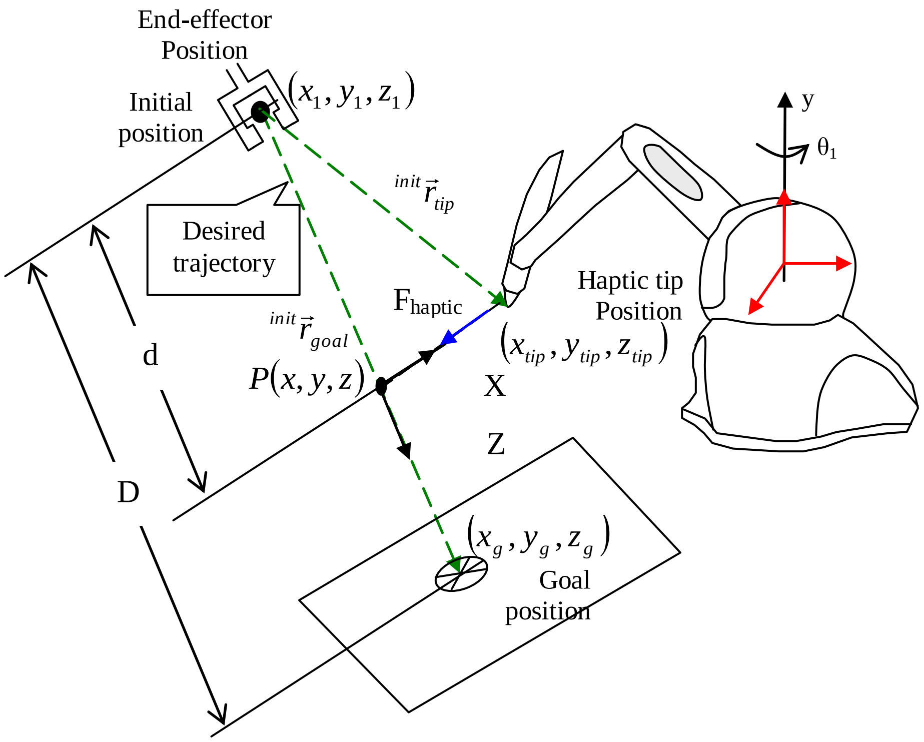


Figure 4.5 Representation of the Sensor-Based Assistance Function

Figure 4.5 illustrates the line of sight vectors defined between the manipulator's end-effector and the region of interest (ROI). At this point, there are two types of

المنارة للاستشارات
www.manaraa.com

assistive forces. One type will be attractive or repulsive to assist the user while moving along the trajectory path and the second type will assist the user motion to follow the prescribed linear path. The latest updates of the position vector obtained in the haptic thread are used to compute the new positions of the virtual object and to display the effect of attraction or repulsion. The linear trajectory is defined by the line of sight vector. Once the user's motion is along the prescribed path, an assist function is generated to guide the user to follow the trajectory with ease.

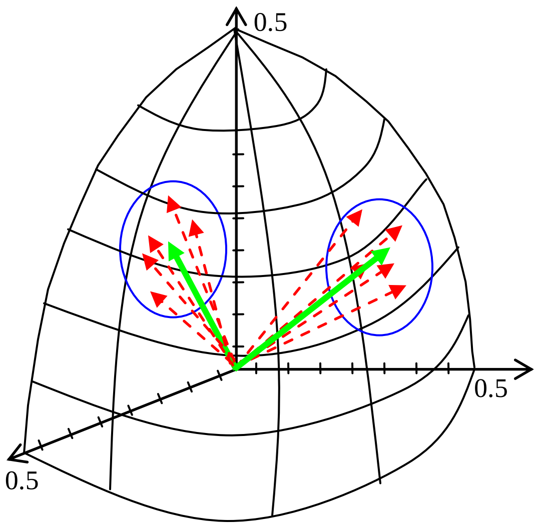


Figure 4.6 A Set of Line of Sight Vectors (in Red) Placed Closed to the Centroid of the Region of Interest (ROI)

The goal or destination of the robot arm is defined as the centroid of the object of interest. The coordinates of the centroid feature are computed in pixels relative to the image plane. As it will be discussed later, sequences of transformations are required to represent the centroid coordinates relative to the world coordinate system. Also, the transformation from image space to joint space of the robot arm requires the knowledge of the kinematic equations of the robot arm. In the case of a robot-mounted camera-laser

المنارة للاستشارات
www.manaraa.com

suite, the visual information is produced as an input signal defined in the image space. Therefore, a conversion is necessary for the transformation. The inverse projection transformation obtained from data provided by the sensory suite (camera and laser range finder) is used to generate a linear trajectory in joint space using the single axis rotation method described in [24]. Since the human is in the control loop, rather than attempting to drive the arm along this path autonomously, the difference between this trajectory and the user's motion as sensed by the haptic device is obtained. Figure 4.6 illustrates the method implemented to generate the linear trajectory in joint-space.

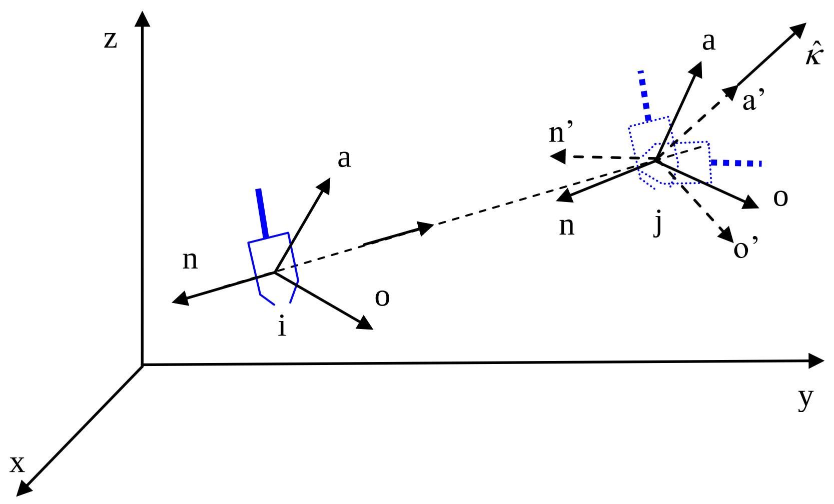


Figure 4.7 Line of Sight Using Single Axis Rotation [60]

In cases where the user wants to switch to autonomous control mode to reach the object of interest, a linear trajectory path is automatically generated using the location of the centroid of the object calculated using information obtained from the sensor datasets.

المنارة للاستشارات
www.manaraa.com

4.8 Summary

In this chapter, the concept of assist function was defined. The control law equations required to calculate the haptic feedback based on the haptic position were developed. The connecting line between the end-effector of the robot arm and the centroid feature of the image of an object extracted from the optical sensor data fusion was developed as well. Two types of functions to assist the user were described. One while approaching the path, and a second for following the prescribed path. The latter is given by the “line of sight” connection of the end-effector of the manipulator and the centroid of the object of interest. In order to reduce the burden of tasks execution over long periods of time, an automatic mode is developed by the generation of a linear trajectory path using the location of the centroid of the object and the current position of the end-effector of the manipulator. In the development of the control law, the location of the centroid was assumed to be known. The procedure to extract this information from images of the object is the topic of the next chapter as well as the sensor-based assist functions calculations.

المنارة للاستشارات
www.manaraa.com

Chapter 5

Visual and Haptic Data for Motion Scaling and Virtual Constraint Definition

5.1 Introduction

In the previous chapter, the concept of the centroid of the object was used to determine the "line of sight" between the end-effector position of the robot arm and the object of interest without detailing the procedure followed for its computation. The centroid calculation is based on information extracted from images of the object of interest which involves computer vision processes such as edge detection and feature extraction techniques. In computer vision, CCD cameras are used as passive sensors to extract data from the captured images. The intensity of the light is used to process the image information and to extract a model of what the camera "sees". In practice, a complication arises from the extraction of 3-dimensional coordinates of an object given 2-dimensional information from the camera's image plane. Data fusion from two different sensors (camera and laser range finder) provides a unique solution to the problem of reconstructing the 3D object position and orientation with respect to a fixed coordinate system based on 2-dimensional datasets. In this combined system, the laser range sensor is used to determine the distance to the observed target object.

This chapter describes the methodology necessary to calculate the location of the centroid and its relation to motion scaling and virtual constraints. The detailed procedures for handling the images, camera calibration, space domain processing, and

www.manaraa.com

mapping of the camera frame with respect to the base reference frame of the robot arm is also presented.

5.2 Spatial Domain Pre-Processing

In order to accurately predict the position and orientation of an object or region of interest, the pixel coordinates of the point in 3D given the points in world coordinates need to be matched. To accomplish this, the computation of the internal ("intrinsic") and external ("extrinsic") parameters of the camera is required. The Tsai's camera model as described in [62] is used to obtain those parameters. The model includes 3D-2D perspective projection with radial lens distortion compensation. This camera model defines a total of eleven (11) parameters: five (5) intrinsic or internal parameters and six (6) extrinsic or external parameters.

The internal parameters describe how the camera forms an image while the external parameters describe the camera position and orientation with respect to the world coordinate frame. The internal parameters include the focal length, the center of projection, and the CCD sensor array dimensions and they are specified by the manufacturer's design. The intrinsic parameters might vary from device to device even if they belong to the same manufacturing batch. The specifications might also be affected by environmental conditions such as distance between the camera and the scene and level of illumination available.

The intrinsic parameters are defined as follows [62, 63, 64]:

1. Principal point (C_x, C_y): intersection coordinates of the optical axis with the image plane as shown in Figure 5.1.

www.manaraa.com

2. Scale factors (d_x, d_y): scaling factors for the x and y pixel dimensions; i.e., the horizontal and vertical size of a single pixel in engineering units (millimeters, inches, meters, etc).
3. Aspect distortion factor (s_x): a scale factor to account for the model distortion in the aspect ratio of the camera.
4. Focal length (f): defines the distance from the optical center (or projection center) to the image plane as defined in a pinhole camera model (this is different from the focal length printed on the lens of the camera by the manufacturer).
5. Lens distortion factor (κ_1): first order radial lens distortion coefficient.

The extrinsic or external parameters of the camera define the transformation of the pose of the camera with respect to a local coordinate system represented by the chessboard pattern's local coordinate system. The six (6) extrinsic camera parameters are:

1. (R_x, R_y, R_z) - defines rotation angles necessary to obtain the rotational transformation between the world and camera coordinate frames.
2. (T_x, T_y, T_z) - corresponds to the translational components between the world and camera coordinate systems.

Figure 5.1 shows the assigned frames of the Tsai's camera model. Calibration data for the Tsai's camera model consists of 3D world coordinates of a feature point (x^w, y^w, z^w) in engineering units (in mm, for example), and corresponding 2D coordinates (X_f, Y_f) in pixels of the corresponding feature point in the image.

المنارة للاستشارات

www.manaraa.com

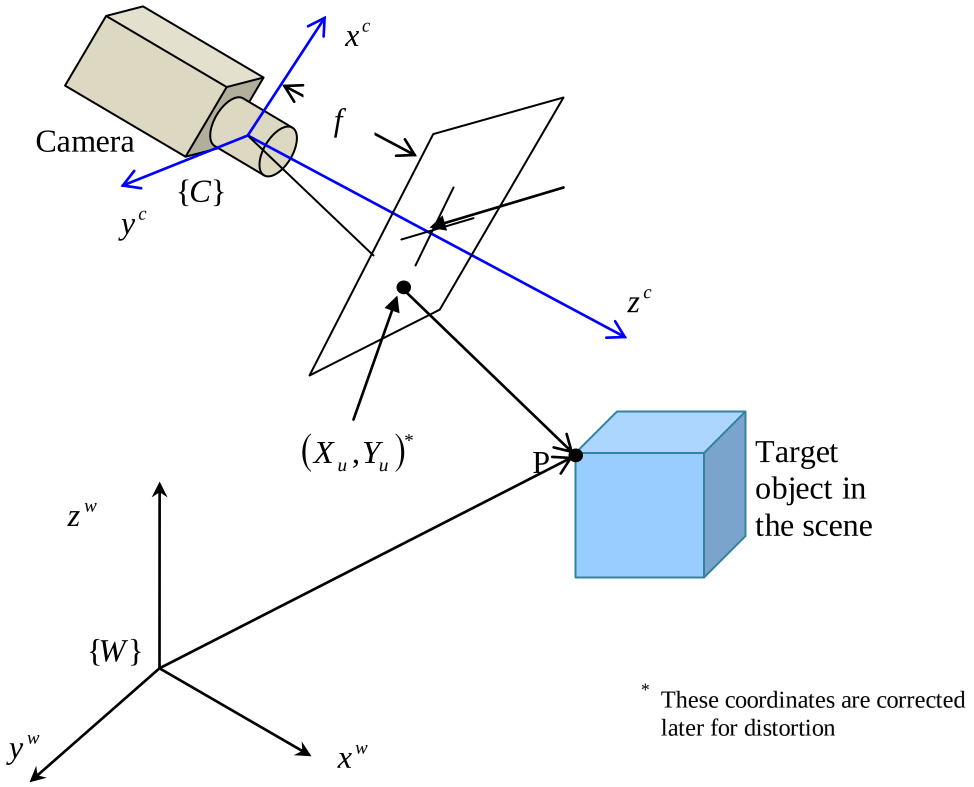

x^c
f
Camera
$\{C\}$
y^c
z^c
$(X_u, Y_u)^*$
P
Target object in the scene
z^w
$\{W\}$
y^w
x^w
* These coordinates are corrected later for distortion

Figure 5.1 Camera Model Geometry

As shown in Figure 5.1, a sequence of transformations is required to define the relationship between the position of a point P in world coordinates, (x^w, y^w, z^w), and the same point as projected in the camera reference frame (X_f, Y_f). The first transformation is a rigid body transformation from the world coordinate system (x^w, y^w, z^w) to the camera-centered coordinate system defined as (x^c, y^c, z^c). This transformation is expressed as follows:

$$\begin{bmatrix} x_C \\ y_C \\ z_C \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \tag{5.1}$$

المنارة للاستشارات
www.manaraa.com

where r_{ij} are the elements of the rotation (orientation) of the camera and $\begin{bmatrix} T_x & T_y & T_z \end{bmatrix}^T$ corresponds to the translation vector in the world coordinate system.

Once this transformation is known, a second transformation relates the $\left(x^c, y^c, z^c\right)$ to the ideal (un-distorted) pinhole camera model $\left(X_u, Y_u\right)$. This is accomplished by using the projective transformation formulas. In other words, the 3D camera point is projected into a 2D-plane $\left(X_u, Y_u\right)$ where the subscript u means "undistorted", because, at this point, there is no correction for lens distortion of the projected point. The projected transformation is given by Eq. (5.2) and (5.3) as follows:

$$X_u = f\frac{x_c}{z_c} \tag{5.2}$$

$$Y_u = f\frac{y_c}{z_c} \tag{5.3}$$

Expanding (5.1) and substituting into Eq. (5.2) and (5.3) yields to:

$$X_u = f\frac{r_{11}x_w + r_{12}y_w + r_{13}z_w + T_x}{r_{31}x_w + r_{32}y_w + r_{33}z_w + T_z} \tag{5.4}$$

$$Y_u = f\frac{r_{21}x_w + r_{22}y_w + r_{23}z_w + T_y}{r_{31}x_w + r_{32}y_w + r_{33}z_w + T_z} \tag{5.5}$$

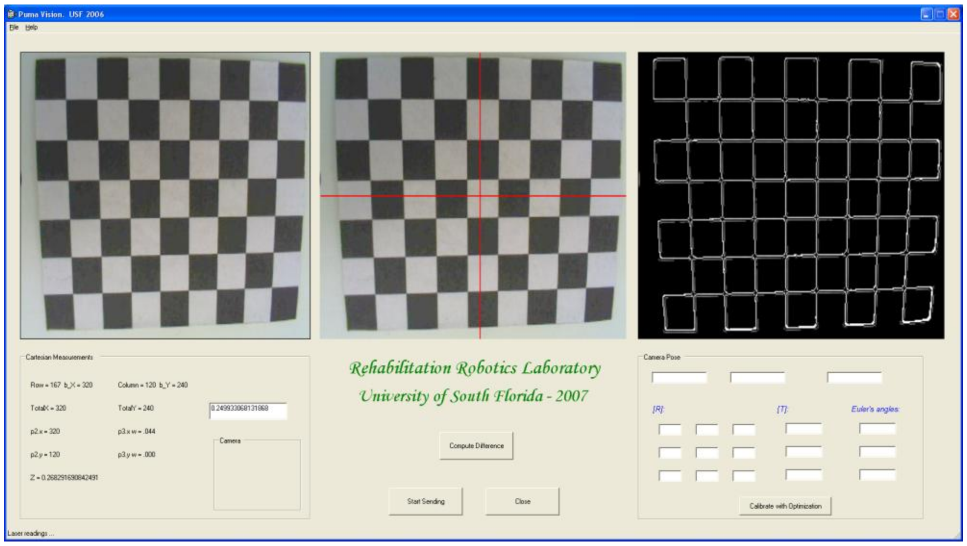
Equations (5.4) and (5.5) represent the undistorted coordinates of the point P. Next, the 1st order radial distortion model is applied to transform the undistorted points $\left(X_u, Y_u\right)$ to the "true" position of the point's image $\left(X_d, Y_d\right)$. The corrected coordinates $\left(X_u, Y_u\right)$ for distortion are:

$$X_u = \left(1.0 + \kappa_1\left(X_d^{\,2} + Y_d^{\,2}\right)\right)X_d \tag{5.6}$$

$$Y_u = \left(1.0 + \kappa_1\left(X_d^{\,2} + Y_d^{\,2}\right)\right)Y_d \tag{5.7}$$

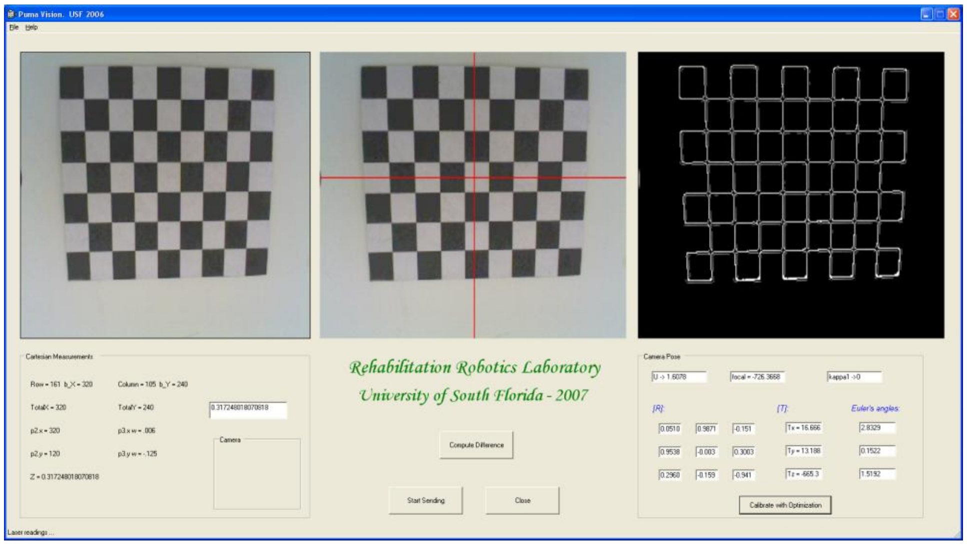
المنارة للاستشارات
www.manaraa.com

Figures 5.2 and 5.3 show some of the results presented to the user through a graphical user interface. Figure 5.4 show the chessboard pattern used for calibration and a typical Puma 560 configuration during calibration.



```
Row = 167  b_X = 320      Column = 120  b_Y = 240
TotalX = 320              TotalY = 240             0.249933068131868
p2.x = 320                p3.x w = .044
p2.y = 120                p3.y w = .000
Z = 0.268291690842491
```

Figure 5.2 Graphical User Interface with Chessboard Calibration Pattern



```
Row = 161  b_X = 320      Column = 105  b_Y = 240
TotalX = 320              TotalY = 240             0.317248018070818
p2.x = 320                p3.x w = .006
p2.y = 120                p3.y w = -.125
Z = 0.317248018070818

U -> 1.6078      focal = -726.3668      kappa1 ->0

0.0510   0.9871   -0.151    Tx = 16.666   2.8329
0.9538   -0.003   0.3003    Ty = 13.188   0.1522
0.2960   -0.159   -0.941    Tz = -665.3   1.5192
```

Figure 5.3 Chessboard Calibration Pattern at a Different Pose of the Robot Arm

www.manaraa.com

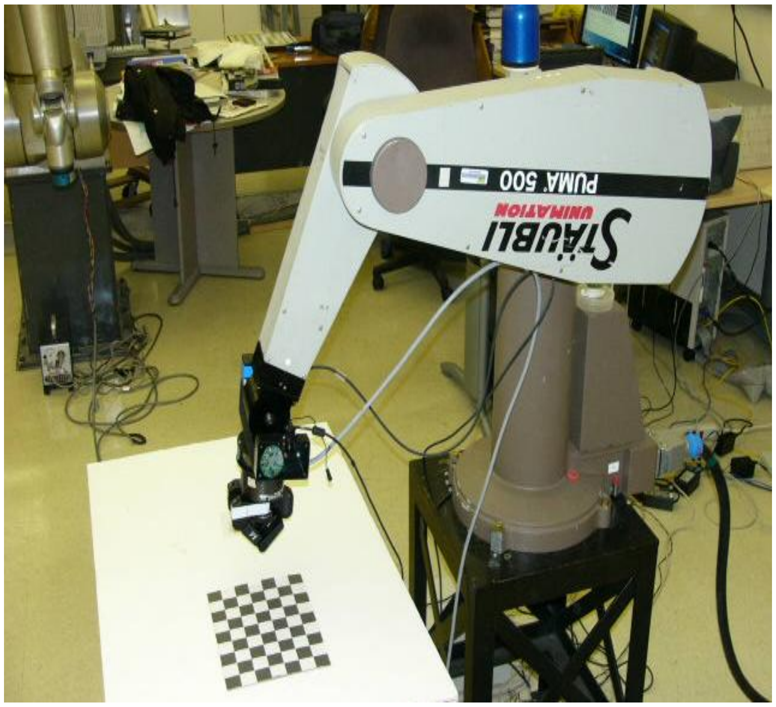


Figure 5.4 Calibration Pattern in the Camera-Mounted Field View

As shown in Figure 5.5, a sequence of conversions is necessary to obtain "true" representation of the position of the image points and their coordinates in the camera's image frame (X_f, Y_f).

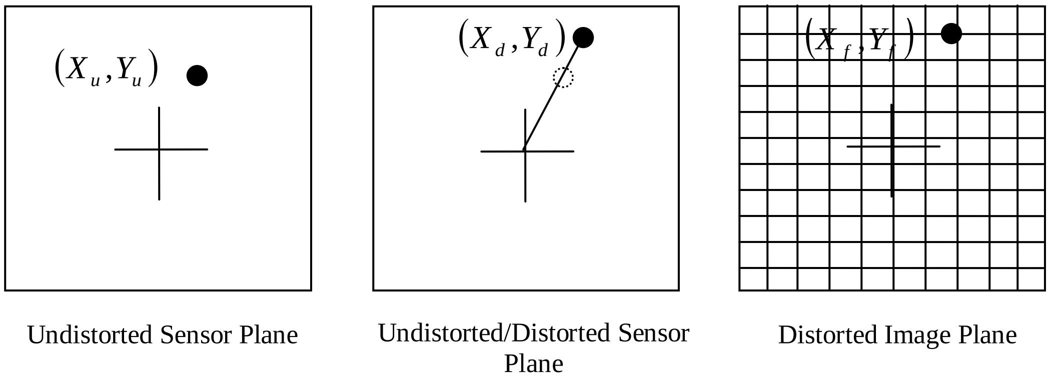


Figure 5.5 Distorted and Undistorted Sensor and Image Coordinates

www.manaraa.com

These conversions are obtained by the evaluation of Eq. (5.8) and (5.9), as follows [65]:

$$X_f = \left(\frac{X_d}{d_x}\right) s_x + C_x \tag{5.8}$$

$$Y_f = \left(\frac{Y_d}{d_y}\right) + C_y \tag{5.9}$$

Now, given a set of points of the object of interest in the world coordinate system (x_w, y_w, z_w) and the corresponding measured position in the image (X_f, Y_f), after the distortion factor has been applied, an error-based objective function can be defined in terms of the difference between the point's image coordinates and the coordinates predicted by the camera model as expressed in Eq. 5.10:

$$\sum_{i=1}^{N}(X_{Ii} - X_{Pi})^2 + \sum_{i=1}^{N}(Y_{Ii} - Y_{Pi})^2 \tag{5.10}$$

where (X_{Ii}, Y_{Ii}) are the observed image positions and (X_{Pi}, Y_{Pi}) are the predicted positions based on the known 3D world coordinates (X_w, Y_w, Z_w) after correction of the radial distortion. The solution is found through the use of a nonlinear optimization technique known as the Levenberg-Marquardt (LM) method [62, 63, 64] as discussed next.

5.3 Numerical Optimization Approach for Estimation of the Camera Parameters

The nonlinear optimization for the determination of camera intrinsic and extrinsic parameters is based on a modified Levenberg-Marquardt (LM) algorithm with a Jacobian calculated by a forward-difference approximation [62]. The LM method increases the computational efficiency by combining gradient descend and Gauss-Newton optimization

www.manaraa.com

methods. Initially, the implementation uses a closed-form least squares estimation of three parameters, the focal length *f*, z-axis translational component T_z and the distortion coefficient κ_1. Using the obtained values as the starting point, an iterative nonlinear optimization of all parameters simultaneously is executed using the LM algorithm one more time.

The intrinsic camera parameters will be constants when the camera is moved with respect to the world reference frame. However, the extrinsic parameters defined by the position and orientation of the camera with respect to the world coordinate system will change and, therefore, Eq. (5.1) must be recomputed. This situation will arise every time the user points to an object and/or rotates the haptic stylus, for example. In this case, the knowledge of the extrinsic camera parameters is fundamental to determine the transformations required to map the position and orientation of an object with respect to the robot arm's end effector frame where the camera and laser ranger are mounted. The procedure involves supplying parameters like window size and number of squares along each axis (X, Y) of the calibration pattern (chessboard pattern in this work) used for calibration and identifying the corners of the calibration grid in each of the images. Then, the Inverse Perspective Mapping (IPM) problem can be addressed.

Figures 5.6 and 5.7 show simulated world-centered and camera-centered reference frames, respectively, after the optimization.

www.manaraa.com

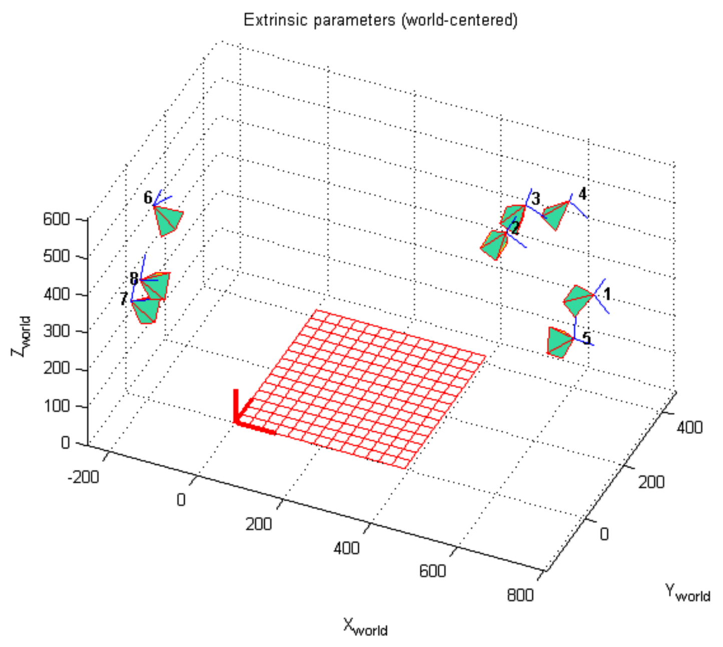


Figure 5.6 World Centered Camera Calibration using Bouguet‘s Toolbox [63]

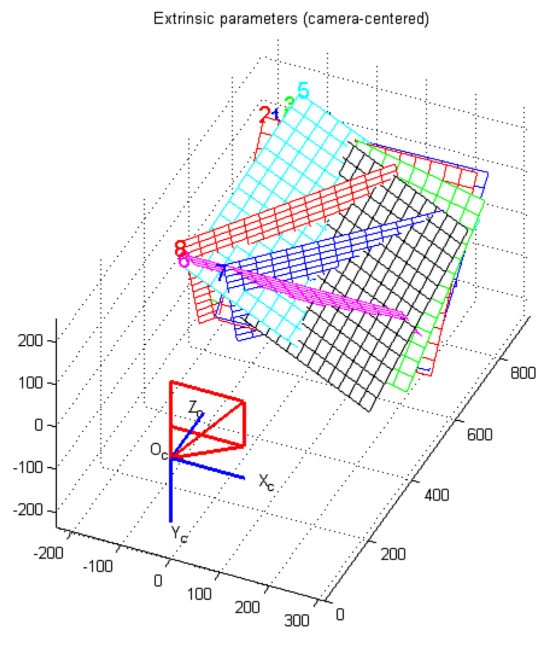


Figure 5.7 Camera Centered Calibration using Bouguet‘s Toolbox [63]

www.manaraa.com

5.4 Inverse Perspective Mapping (IPM)

The inverse perspective mapping IPM is the key to use the visual information for driving the manipulator using supervisory control by the determination of the line of sight defined between the end-effector of the robot arm and the centroid of the object of interest measured by the sensors. It can be also used for planning the straight line motion of the end-effector in autonomous mode. The IPM is the opposite problem regarding the projective projection used during calibration. Figure 5.8 illustrates possible errors between the calibrated camera model predictions and the actual position of the observed image points.

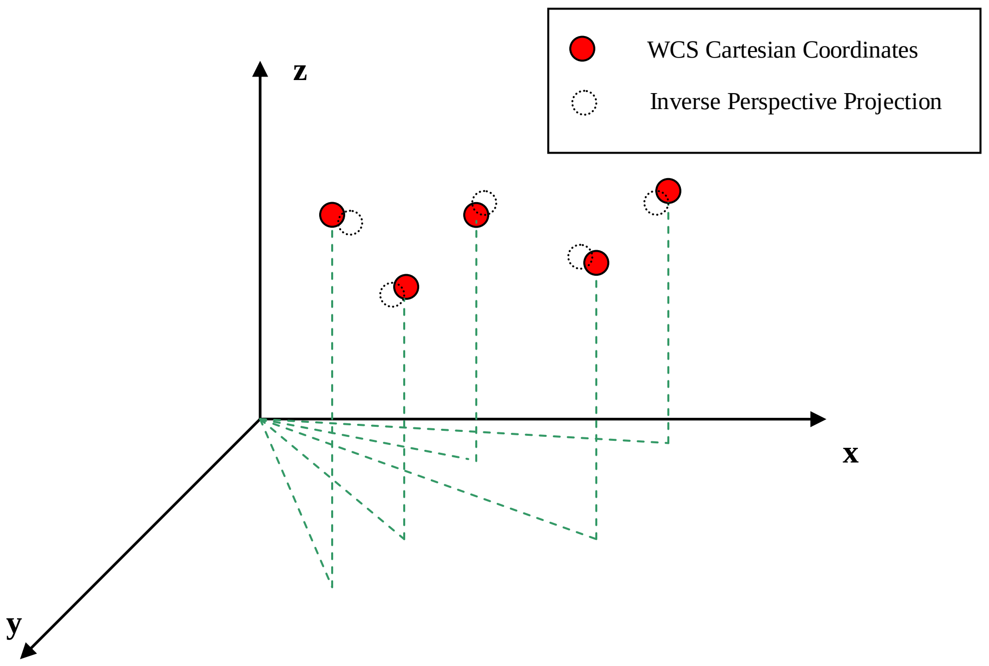


Figure 5.8 Illustration of the Error between Predicted and Observed Image Points

During calibration, a set of N image points ($N > 5$) are matched to the corresponding points in the world coordinate system and the intrinsic and extrinsic parameters required for this matching are calculated. On the other hand, the inverse perspective problem uses the calibration data to determine the position and orientation of

المنارة للاستشارات
www.manaraa.com

points on the image relative to the world coordinate system. Similarly to the calibration problem, the methodology implemented to solve the inverse perspective problem is once again the Tsai's method [62] and the Levenberg-Marquardt (LM) numerical technique is also used to solve the optimization problem in a least-square sense. For the application to this particular problem, input to the Tsai's algorithm is the predicted position and orientation of the end-effector using the camera and the object position relative to the base and data from forward kinematics solution of the robot arm. Figure 5.9 shows some of the coordinate frames assigned in order to obtain the required transformations of the points in the image plane with respect to the camera plane.

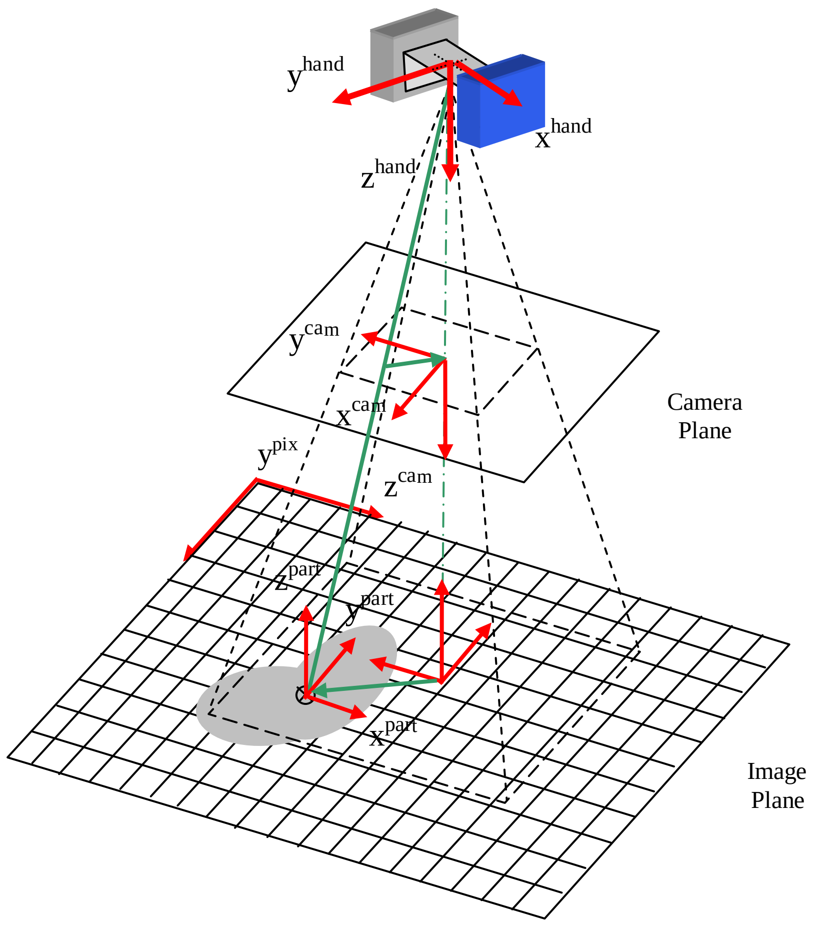


Figure 5.9 Camera and Image Planes Geometrical Relationships

www.manaraa.com

5.5 Edge Detection and Feature Extraction

In order to recognize an object from an image, it is assumed that the object can be segmented out of the image background after binarizing the captured image. A histogram equalization post-processing is performed to make an even distribution of the grayscale pixel colors. For edge detection, the “Sobel” method is used to compute the edges [64] as well as the “Canny” method described in [66]. The Canny method is the preferred method in this work because it is more efficient in reducing noise from the captured image. Both methods are standard image processing techniques; the details of their implementations are described in [64] and [66].

The methodology for the segmentation is that for each segmented object, the feature extraction component of the vision system computes the object’s geometric features, such as the centroid, perimeter, or area. For the computation of the centroid, the following two equations are used: $C_x = \frac{1}{n}\sum_{i=1}^{n} x$ and $C_y = \frac{1}{n}\sum_{i=1}^{n} y$ where x and y represents each individual pixel coordinates, and n defines the total number of pixels in the 2D region of interest (ROI) [64]. As a result of the image projection and transformation, only 2D datasets are available which correspond to the x-y plane. However, in order to drive the robotic system to reach a particular object of interest, the triple (x, y, and z) Cartesian coordinates are required. So, the additional information, which corresponds to the z-dimension or depth, is provided by the laser range finder measurements.

The acquisition and digitalization processes of the images produce distortions of the original region of interest (ROI), especially when viewing objects from a large

المنارة للاستشارات
www.manaraa.com

distance. These distortions increase the uncertainty of the datasets, the complexity of the image recognition process as well as the computational expense. For applications involving the location of objects of interest at large distances, the procedure implemented provides for distortion removal introduced by the lens and the aspect ratio of the camera, respectively. As stated before, the methodology for the perspective projection camera model was devised by R. Tsai [62] and implemented by Bouguet [63] as a MatLab toolbox. This toolbox was used for validating the results of the multithreaded implementation of this algorithm which is included as a module of the vision system. An optimized algorithm for the camera calibration is also described in [67].

5.6 Mapping to the Robot Arm Reference Frame

In order to use the robot-mounted camera (hand-eye) information and the laser range finder sensor for the robot pose estimation, both intrinsic and extrinsic parameters of the camera needs to be obtained first. Then, the transformations for mapping the grid's local coordinate system of sensing array with respect to the manipulator's base frame are required. It is important to note that, in practice, an intermediate step, known as the pixel-to-camera transformation, will also be required because points on the object or region of interest are known at the pixel level. This means that image pixel pairs ($\text{pixel}_{\text{row}}$, $\text{pixel}_{\text{col}}$) representing row and column numbers, respectively, are available with respect to a fixed pixel coordinate frame attached to the sensing array.

From Figure 5.2, the geometrical relationships between the coordinate points in the camera and image planes can be described. Note that the origin of the image plane is defined at the left-upper corner of the image window. On the other hand, the origin of

www.manaraa.com

the camera plane is considered to be at the center of the camera plane (the principal point) which corresponds to one of the intrinsic or internal parameter of the particular camera in use. For a robot-mounted camera, the offset between the end-effector of the manipulator and the camera is constant (it does not change between views), but it is unknown. The assembled homogenous transformation is then represented relative to the end-effector of the robotic arm given their relative position as illustrated in Figure 5.10. A detailed procedure of the mapping of the different reference frames can be found in [63].

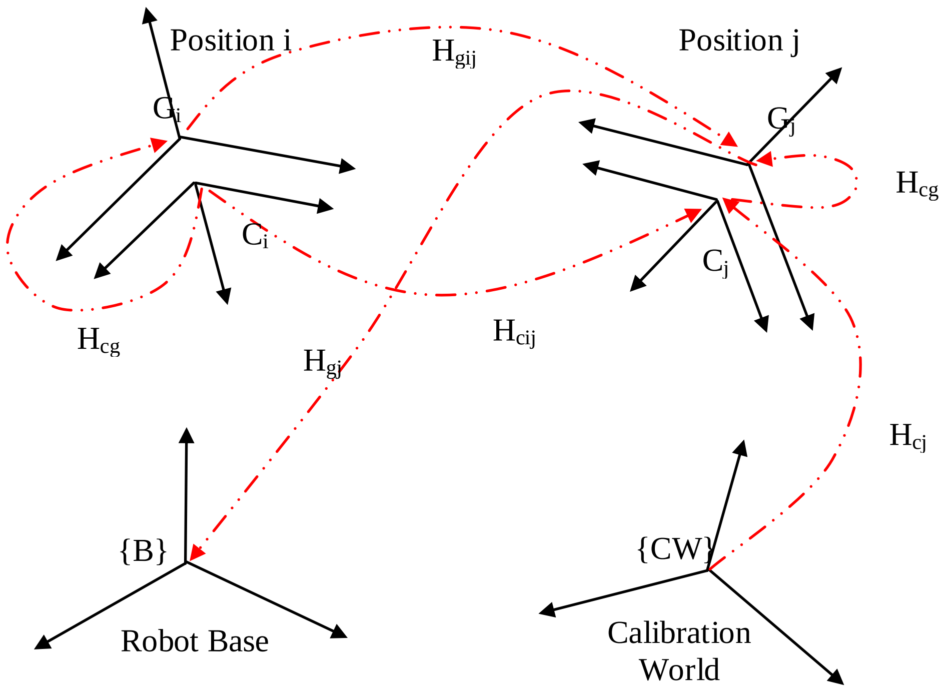


Figure 5.10 Relationships between the Different Coordinate Frames [63]

In order to be able to drive the robot arm using the sensor information from the laser and the camera combination, the pose transformation of the robot arm with respect to the manipulator's base frame is required.

www.manaraa.com

From Figure 5.10 the following relationship for the homogeneous transformation can be extracted:

$$H_{gij} H_{cg} = H_{cg} H_{cij} \tag{5.11}$$

where,

H_{gij}: (4x4) homogenous transformation of the gripper or end-effector between views.

H_{cg}: (4x4) homogenous transformation of the gripper or end-effector with respect to the camera.

H_{cij}: (4x4) homogenous transformation of the camera between views.

As stated previously, at this point the Tsai's approach is once again used to solve (5.11) and to determine the position of the camera with respect to the robot hand coordinate frame. For a full description of the method refer to [62]. The result of the method will be the transformation matrix H_{cg}. The homogeneous transformations H_{gij} and H_{cij} are known from the robot forward kinematic equations and from the extrinsic parameters of the camera calibration procedure discussed earlier. The transformation H_{grid2c} which defines the calibration grid frame with respect to the camera frame can be found from the inverse of the extrinsic parameters of the camera (*R*$_c$, *T*$_c$), as follows:

$$H_{grid2c} = \begin{bmatrix} {}^{c}r_{11} & {}^{c}r_{12} & {}^{c}r_{13} & {}^{c}t_x \\ {}^{c}r_{21} & {}^{c}r_{22} & {}^{c}r_{23} & {}^{c}t_y \\ {}^{c}r_{31} & {}^{c}r_{32} & {}^{c}r_{33} & {}^{c}t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \tag{5.12}$$

www.manaraa.com

where r_{ij} are the elements of the rotation matrix R_c and (t_x, t_y, t_z) are the components of the translation vector T_c.

At a particular position and orientation of the robot manipulator the transformation H_{gij} is stored and the corresponding extrinsic parameters of the camera are retrieved given the image of the region of interest (ROI). The camera transformation in the manipulator base reference frame $H_{c2b_{ij}}$ is:

$$H_{c2b_{ij}} = H_{gij} H_{cg} \tag{5.13}$$

The calibration grid transformation $H_{grid2b_{ij}}$ can also be obtained with respect to the robot base frame as:

$$H_{grid2b_{ij}} = H_{c2b_{ij}} H_{cij} \tag{5.14}$$

The fixed transformation between the end-effector and the robot-mounted camera can be verified using the following expression:

$$H_{cg} = \left(H_{gj}\right)^{-1} H_{c2b_{ij}} \tag{5.15}$$

As an additional check to verify the solution, the result of (5.15) must reflect the fact that the homogeneous transformation of the camera with respect to the gripper or end-effector frame is constant for all calibration points given that the camera is attached to the end-effector of the robot arm. Table 5.1 shows the rotation and translation components of the camera and the predicted manipulator's end-effector obtained from Eq. (5.11) using the Tsai's approach corresponding to ten (10) calibration points. This table was generated using simulation software in MatLab and compared to the recorded

www.manaraa.com

transformation matrices of the end-effector of the Puma robot arm from the forward kinematics.

Table 5.1 Extrinsic Camera Parameters ($[R_c],[T_c]$) and End-effector Rotation and Translation Matrices ($[R],[T]$)

Image Rotation Matrix $[R_c]$			Image Translation $[T_c]$, mm	End-effector Rotation Matrix $[R]$			End-effector Translation $[T]$, mm
0.1179	0.9928	-0.0232	-126.5395	-0.6862	0.6945	0.2163	92.8000
0.9902	-0.1158	0.0779	-66.5448	0.7274	0.6530	0.2110	635.6000
0.0747	-0.0322	-0.9967	235.2563	0.0053	0.3021	-0.9533	-326.6000
0.0124	0.9996	-0.0263	-143.9736	-0.6221	0.7609	0.1843	115.9000
0.9937	-0.0094	0.1119	-76.1713	0.7794	0.5796	0.2378	625.8000
0.1116	-0.0275	-0.9934	224.2457	0.0741	0.2916	-0.9537	-338.0000
-0.0849	0.9957	-0.0378	-135.1690	-0.5437	0.8188	0.1844	115.8000
0.9900	0.0886	0.1100	-83.2790	0.8327	0.4990	0.2399	626.7000
0.1129	-0.0281	-0.9932	225.3835	0.1045	0.2840	-0.9531	-336.4000
-0.1185	0.9921	-0.0422	-131.9680	-0.5163	0.8363	0.1843	115.8000
0.9864	0.1225	0.1093	-85.4977	0.8487	0.4709	0.2407	627.0000
0.1136	-0.0287	-0.9931	225.6934	0.1145	0.2807	-0.9529	-335.9000
-0.0856	0.9959	-0.0293	-138.7425	-0.5461	0.8153	0.1925	115.8000
0.9896	0.0884	0.1138	-85.7883	0.8307	0.4976	0.2496	627.6000
0.1159	-0.0193	-0.9931	225.9435	0.1076	0.2962	-0.9490	-334.5000
-0.1478	0.9874	-0.0567	-124.5867	-0.4882	0.8550	0.1749	115.7000
0.9824	0.1532	0.1067	-88.7926	0.8637	0.4447	0.2372	628.1000
0.1140	-0.0400	-0.9927	227.9899	0.1250	0.2669	-0.9556	-333.7000
-0.1117	0.9921	-0.0570	-102.6291	-0.5192	0.8382	0.1666	93.5000
0.9870	0.1174	0.1099	-88.6900	0.8454	0.4752	0.2439	632.0000
0.1157	-0.0440	-0.9923	227.1811	0.1253	0.2675	-0.9554	-333.3000
-0.1417	0.9874	-0.0699	-94.6773	-0.4910	0.8568	0.1574	92.6000
0.9830	0.1487	0.1076	-88.9109	0.8609	0.4497	0.2378	632.2000
0.1167	-0.0535	-0.9917	227.5109	0.1329	0.2523	-0.9585	-333.1000
-0.1053	0.9923	-0.0655	-98.0316	-0.5214	0.8387	0.1574	92.6000
0.9874	0.1121	0.1119	-90.0564	0.8440	0.4796	0.2400	633.0000
0.1183	-0.0529	-0.9916	228.2876	0.1258	0.2580	-0.9579	-331.7000
-0.1066	0.9920	-0.0676	-96.7395	-0.5256	0.8398	0.1359	92.6000
0.9832	0.1153	0.1414	-102.9621	0.8354	0.4793	0.2691	633.2000
0.1480	-0.0514	-0.9876	225.5877	0.1609	0.2550	-0.9535	-331.3000

Once the end-effector transformation is determined based on the sensors data, the connecting line between the end-effector of the robot arm and the position and orientation

المنارة للاستشارات

www.manaraa.com

of the centroid feature of the object with respect to the manipulator's base is defined as the desired straight line trajectory.

As explained in Chapter 4, the z-component of the "LoS" is found using the orthonormal constraint via the cross product:

$$\begin{vmatrix} \hat{x} & \hat{y} & \hat{z} \\ (r_{11} - X_{im} r_{31}) & (r_{12} - X_{im} r_{32}) & (r_{13} - X_{im} r_{31}) \\ (r_{21} - Y_{im} r_{31}) & (r_{22} - Y_{im} r_{32}) & (r_{23} - Y_{im} r_{31}) \end{vmatrix} \tag{5.16}$$

Eq. (5.16) needs to be transformed to coincide with the origin of the end-effector reference frame for grasping. The necessary transformation correspond to a translation to specify the line of sight relative to the end-effector frame (the z-axis of the camera is parallel to the z-axis of the end-effector). The method to calculate the assist function based on the "LoS" of the camera is discussed in detail in Chapter 6.

5.7 Summary

This chapter describes the procedure for using the camera and laser information to compute the centroid location as well as the position and orientation of an object of interest in a 3D space. The principal utility of the sensory information (camera and laser range finder) at this level is to provide an automated system for measuring and digitally processing the content of the images of an object of interest. This information is then used for calculating the line of sight (LoS) defined between the end-effector position and the object. Then, the LoS defines a linear trajectory for guiding the user's motion towards the object of interest. The Levenberg-Marquardt (LM) nonlinear optimization method is

المنارة للاستشارات

www.manaraa.com

described for the camera and the laser range finder calibration. The LM is also used for solving the inverse perspective mapping (IPM) to transform from measured points in the image's plane to the base reference frame of the manipulator.

المنارة للاستشارات
www.manaraa.com

Chapter 6

Sensor-Based Assistance Function Calculations

6.1 Introduction

The architecture proposed in this work incorporates assistance to the user's motion using simple sensors (a camera and a laser range finder). The visual information is combined with the human inputs and the deviations are corrected by the calculation of assistive or resistive forces. The line of sight vector defined between the manipulator's end-effector and the object of interest is used as a constraining line. Once the object is in the view of the eye-in-hand camera, the vision system is activated and all the required transformations are determined as explained in Chapter 5.

In the image pre-processing part, the case in which all objects are on the top of a table is considered. In this situation, the control input is the position and orientation commands calculated from the visual input as well as the commands of the haptic input device. This chapter describes the determination of the forces required to provide the appropriate feedback to guide the user's motion, which are identified here as the sensor-based assistance functions.

6.2 Generic Scheme for Motion-Dependent Force Feedback Calculation

The feedback force, F, is computed to maintain the haptic tip constrained to the user's intended path (see Figure 6.1). This force feedback is generated according the following control law:

المنارة للاستشارات

www.manaraa.com

$$F = -K_1 \Delta p - K_2 \Delta \dot{p} \qquad (6.1)$$

where,

F = force feedback through the haptic interface

K_1 = proportional gain

K_2 = derivative gain

Δp = difference between the haptic tip position and target's centroid

$\Delta \dot{p}$ = rate of change of Δp

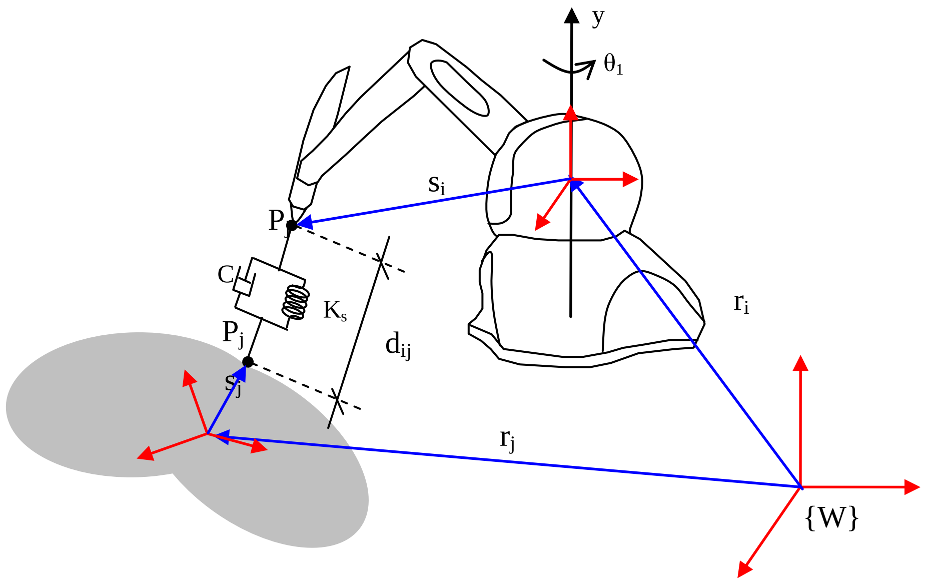


Figure 6.1 Translational Distance, d_{ij}, Used for Feedback Force Control Law

From equation 6.1, the translational spring-damper virtual model is used for the force computation where d_{ij} represents a displacement vector connecting points P_i and P_j. P_i corresponds to the tip of the haptic stylus, and P_j correspond to a contact node on a path or contact point on an object of interest. As previously explained, the object's centroid as well as the line of sight are used as geometric features to have a visual

www.manaraa.com

indication of the user's intended path. The displacement vector from P$_i$ to P$_j$ is obtained as:

$$d_{ij} = r_j + T_j s_j - r_i - T_i s_i \tag{6.2}$$

where T_i and T_j are homogenous transformation matrices expressed with respect to the world coordinate system, {W}.

The corresponding length of the spring-damper, ℓ, is now defined as:

$$\ell^2 = d_{ij}{}^T d_{ij} \tag{6.3}$$

The damping force component is a function of the displacement rate which is obtained by differentiating Eq. (6.3) with respect to time:

$$2\ell\dot{\ell} = 2d_{ij}{}^T \dot{d}_{ij} \tag{6.4}$$

After substitution and simplification, Eq. (6.4) yields:

$$\dot{\ell} = \left(\frac{d_{ij}}{\ell}\right)^T \left(\dot{r}_j + \dot{T}_j s_j - \dot{r}_i - \dot{T}_i s_i\right) \tag{6.5}$$

It can be shown that the time derivatives of the transformation matrices can be expressed in terms of angular velocities, ω_i and ω_j (see Appendix E for details) as:

$$\dot{\ell} = \left(\frac{d_{ij}}{\ell}\right)^T \left(\dot{r}_j - T_j s_j \omega_j - \dot{r}_i + T_i s_i \omega_i\right) \tag{6.6}$$

Finally, the magnitude of the force applied to the user's hand through the haptic device is found to be:

$$F = -K_1(\ell - \ell_0) - K_2\dot{\ell} \tag{6.7}$$

المنارة للاستشارات

www.manaraa.com

Comparing Eq. (6.1) and Eq. (6.7), it is observed that $\Delta p = (\ell - \ell_0)$ and $\Delta \dot{p} = \dot{\ell}$; i.e., the shortest distance between the haptic tip position and any point on the connecting line, as shown in Figure 6.1, is taken to be equivalent to the change in length of a virtual spring. Similarly, the rate of change $\Delta \dot{p}$ is equivalent to the rate of change of the virtual spring length.

As it is obvious from this derivation, the torsional components were not taken into consideration in the calculation. The Phantom Omni device used in this research does not have built-in actuators such that it can exert torsional forces with the thimble. In the case of a device with such capabilities, the generalized forces can be calculated using the principle of virtual work where the virtual displacements can be obtained from the differential equation expressed in Eq. (6.7) and virtual rotations components can be obtained in terms of the Euler angles orientation coordinates [68, 69]. The next section discusses additional forces and effects used to constrain or guide the user's motion. These forces are sent to the haptic device in real-time.

6.3 Sensor-Based Assistance

The sensors (camera and laser range finder) information needs to be mapped to the Cartesian space of the manipulator in order to generate an attractive or repulsive force to guide the user until the object of interest is between the gripper fingers in real time.

As stated before, the line of sight (LoS) is considered to be the intended or desired user's motion. A constraint frame for the end-effector of the manipulator is defined along the LoS of the camera considering the z-axis pointing in the direction of the camera axis, the x-axis along the line defined between the initial position of the haptic tip

المنارة للاستشارات

www.manaraa.com

$\left(x_{tip}, y_{tip}, z_{tip}\right)$ and the projection defined by $P(x, y, z)$ as shown in Figure 4.3. There will be measurement errors between the line of sight and the user's input possibly due to the reduced physical performance due to fatigue of the person interacting with the system or tremor illness. These error signals are used to compute force constraint's to guide the user towards the destination. As mentioned, the force constraints are defined by two different models: a) an attractive or repulsive force to guide the user towards the trajectory, and b) an assistive force to guide the user along the trajectory path. In the case of approaching the surface of a table, the contact force can be computed as a function of the remaining distance to the surface.

The Cartesian motion between the initial position of the manipulator and the goal position is described in terms of robot arm transformations with respect to the base frame of the manipulator. One way to accomplish this is to define a translation along a straight line and a rotation about a fixed axis $\kappa = \left(k_x, k_y, k_z\right)^T$ by an equivalent angle θ [51, 60] (See Appendix B). As shown in Figure 4.3, the two constraint points are defined by the coordinates $\left(x_1, y_1, z_1\right)$ and $\left(x_g, y_g, z_g\right)$, respectively. The equation of the 3D line is given by:

$$\frac{x - x_1}{x_g - x_1} = \frac{y - y_1}{y_g - y_1} = \frac{z - z_1}{z_g - z_1} = k \tag{6.8}$$

The projection of the initial position of the end-effector is:

$$\begin{aligned} x &= k(x_g - x_1) + x_1 \\ y &= k(y_g - y_1) + y_1 \\ z &= k(z_g - z_1) + z_1 \end{aligned} \tag{6.9}$$

المنارة للاستشارات
www.manaraa.com

The distance between the projected point $P(x, y, z)$ and the initial point is given by

$$d = \sqrt{(x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2} \qquad (6.10)$$

Substituting (4.9) into (4.10) yields:

$$d = \sqrt{k^2[(x_g - x_1)^2 + (y_g - y_1)^2 + (z_g - z_1)^2]} \qquad (6.11)$$

If D is defined as the distance measured using the laser range finder, and it is expressed in terms of the initial and goal Cartesian coordinates, then $D = \sqrt{(x_g - x_1)^2 + (y_g - y_1)^2 + (z_g - z_1)^2}$. The following computation is performed:

$$d = k\,D \quad \rightarrow \quad k = \frac{d}{D} \qquad (6.12)$$

The projection of the haptic tip's initial position $P(x, y, z)$ can be obtained by substituting (6.11) into (6.9). The constraint frame for the end-effector of the manipulator can now be obtained by defining the axes as shown in Figure 4.3 where the z-axis points in the direction of the constraint line, the x-axis along the line defined between the initial position of the haptic tip $(x_{tip}, y_{tip}, z_{tip})$ and the projection defined by $P(x, y, z)$. The direction of the y-axis can be found using the right-hand rule and orthogonality condition $\vec{Y} = \vec{X} \times \vec{Z}$. After normalization, the transformation matrix R in terms of the directional cosines $[\hat{n} \quad \hat{o} \quad \hat{p}]$ can be found as:

المنارة للاستشارات
www.manaraa.com

$$R = \begin{bmatrix} \frac{n_x}{\sqrt{n_x{}^2 + n_y{}^2 + n_z{}^2}} & \frac{o_x}{\sqrt{o_x{}^2 + o_y{}^2 + o_z{}^2}} & \frac{a_x}{\sqrt{a_x{}^2 + a_y{}^2 + a_z{}^2}} \\ \frac{n_y}{\sqrt{n_x{}^2 + n_y{}^2 + n_z{}^2}} & \frac{o_y}{\sqrt{o_x{}^2 + o_y{}^2 + o_z{}^2}} & \frac{a_y}{\sqrt{a_x{}^2 + a_y{}^2 + a_z{}^2}} \\ \frac{n_z}{\sqrt{n_x{}^2 + n_y{}^2 + n_z{}^2}} & \frac{o_z}{\sqrt{o_x{}^2 + o_y{}^2 + o_z{}^2}} & \frac{a_z}{\sqrt{a_x{}^2 + a_y{}^2 + a_z{}^2}} \end{bmatrix} \tag{6.13}$$

As previously stated, the equivalent single axis-angle method is used to represent a rotation about a single axis $\hat{\kappa}$ to align the end-effector frame to the desired goal configuration. This is also the basis for planning the linear motion for autonomous execution at the user's command. In this case, the linear trajectory is divided into *N* smaller segments, where *N* depends on the distance of travel, nominal linear velocity of the end-effector and the update rate of the trajectory generation thread. To accomplish this task, the inverse kinematic equations of the manipulator are solved at each intermediate position.

Two different approaches to solve the inverse kinematic equations are implemented in this work. One approach considers the closed-form solution to obtain the required joint variables to drive the robot arm to the next segment along the linear trajectory. This solution is appropriate when the robot arm is kinematically non-redundant. The second approach is to obtain the joint rates using the inverse Jacobian, followed by integration to obtain a set of joint angles by the application of Whitney's resolved-rate algorithm. This allows added flexibility for dealing with kinematically redundant robots. As stated before, the benefit of switching control between the human

المنارة للاستشارات
www.manaraa.com

user and the automatic control is to reduce the burden of executing repeated tasks and to provide an appropriate level of assistance to the user by scaling the motion.

As an example of constrained motion, the haptic end-effector linear velocity can be assigned to the robot end-effector velocity as $\vec{V}_{robot} = R^T \vec{V}_{haptic}$. This velocity can be scaled using a scaling factor in the constrained direction as follows:

$$\vec{V}_{robot} = \begin{bmatrix} K_v & 0 & 0 \\ 0 & K_v & 0 \\ 0 & 0 & 1 \end{bmatrix} R^T \vec{V}_{haptic} \tag{6.14}$$

Notice that the Z-axis component is not affected by the scale factor because the constrained frame is defined along the desired path. However, the X and Y directions are scaled by the scaling factor $0 < K_v < 1$. The resulting velocity components are then used as the input to the resolved-rate algorithm as shown in the simplified version of the Whitney's algorithm in Figure 3.7, which shows an expanded version as implemented in the real-time telerobotic controller.

The current position in the base frame of the haptic device is obtained, the vector ${}^{init}\vec{r}_{tip}$ defined from the starting point to the haptic device position is calculated as

$${}^{init}\vec{r}_{tip} = \left(x_{tip} - x_1, y_{tip} - y_1, z_{tip} - z_1\right) \tag{6.15}$$

Similarly, the vector between the starting and goal (destination) points is obtained as:

$${}^{init}\vec{r}_{goal} = \left(x_{goal} - x_1, y_{goal} - y_1, z_{goal} - z_1\right) \tag{6.16}$$

Finally, the projection of the haptic position on the desired path is obtained through the use of the dot product as:

www.manaraa.com

$$\vec{r}_{projected} = \frac{\left(\left({}^{init}\vec{r}_{tip}\right)\bullet\left({}^{init}\vec{r}_{goal}\right)\right)}{\left\| {}^{init}\vec{r}_{goal} \right\|} {}^{init}\vec{r}_{goal} \tag{6.17}$$

In Eq. (6.17), the vector ${}^{init}\vec{r}_{goal}$ is equivalent to ${}^{init}\vec{r}_{obj}$ defined by:

$${}^{init}\vec{r}_{obj} = \left(x_w - x_1, y_w - y_1, z_w - z_1\right) \tag{6.18}$$

where the Cartesian coordinates of the object $\left(x_w, y_w, z_w\right)$ are represented in the world space following the procedure explained in Chapter 5.

The trajectory path or control surface is surrounded by an attractive potential field the amplitude of which increases with the distance between the end-effector and the projected point. The assistance force vector is calculated as:

$$F_{haptic} = K\left({}^{init}\vec{r}_{tip} - \vec{r}_{projected}\right) \tag{6.19}$$

For a motion task along the X-axis, a general scheme is to constrain the Y and Z axis directions. If the assisted motion is along the Y axis, then the X and Z directions are constrained. Table 6.1 shows the different cases for constrained directions in a motion task.

Table 6.1 Constrained Directions in a Motion Task

X-dir Free	Y-dir Free	Z-dir Free
$\begin{cases} X = hapticPos_X \\ Y = f\left(F_{haptic_Y}\right) \\ Z = f\left(F_{haptic_Z}\right) \end{cases}$	$\begin{cases} X = f\left(F_{haptic_X}\right) \\ Y = hapticPos_Y \\ Z = f\left(F_{haptic_Z}\right) \end{cases}$	$\begin{cases} X = f\left(F_{haptic_X}\right) \\ Y = f\left(F_{haptic_Y}\right) \\ Z = hapticPos_Z \end{cases}$

المنارة للاستشارات
www.manaraa.com

where, $(hapticPos_X, hapticPos_Y, hapticPos_Z)$ corresponds to the current user's position in Cartesian space and $f(F_{haptic_X})$, $f(F_{haptic_Y})$, $f(F_{haptic_Z})$ are the new position after the constraint force is applied.

Equation (6.19) includes only the spring-type force feedback. Considering the force feedback control law represented by Eq. (6.7), it can be observed that this control law not only compensates for the difference (error signals) between the computer-generated desired path and the deviation from this path caused by the user input, but it can also includes a dampening effect. This effect is directly proportional to the velocity component in the opposite direction of the motion. The combined spring-type and damping-type feedback forces help the user to stay in the straight trajectory.

Once the user is moving along the path, additional assistance is provided in the direction along the linear trajectory as illustrated in Figure 6.2. The linear velocity components are scaled up or down depending upon the user's motion along the trajectory. In the illustration, $\vec{V}_{\text{scaled}}$ corresponds to the scaled velocity vector, $\vec{V}_{\text{user}}$ is the current user's motion velocity vector, and $\vec{V}_{\text{proj}}$ is the projection of the user's velocity vector in the direction of the desired resultant velocity.

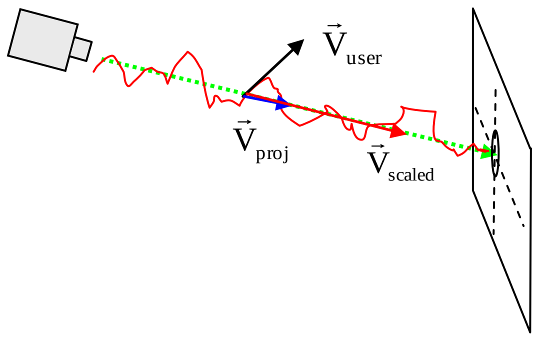


Figure 6.2 Desired Path and "Noisy" Trajectory Input

www.manaraa.com

The Phantom Omni has built-in force feedback capabilities, and an attractive or repulsive force can be rendered through the haptic device interface to constrain the user's motion using the control law defined by Eq. 6.5. The level of assistance can be modified as the user's skills in executing a particular task increase by modifying the scaling factor *K* (gain) in the haptic control strategy.

6.4 Comments

The Cartesian trajectory generated by positioning and orienting the end-effector toward the object (destination point) is monitored by a separate computational thread. By separating the data acquisition processing and communication process, a highly responsive interaction was attained. Even though the manipulation of objects can be driven through the sense of touch and the optical sensory information while the human is in the loop, the multithreaded implementation at the sensory suite level allows for the possibility to switch supervisory control of the robotic arm to an autonomous mode at the user's command with ease. This transition between a supervisory control mode to an autonomous control mode reduces the burden on the user and reduces the possibility of fatigue during long time interactions with the system.

6.5 Summary

In this chapter, the concept of sensor-based assistance is defined. The assistance function calculations are described as well as the force feedback required to provide the appropriate sensor assisted function to guide the user's motion. The line of sight concept is considered as a visual indication of the intended linear trajectory of the user. The

المنارة للاستشارات
www.manaraa.com

assistance function was generated to constraint the user's motion based on the measured differences between the LoS, determined through the use of the sensor data fusion, and the current position of the user, provided by the haptic's tip.

المنارة للاستشارات
www.manaraa.com

Chapter 7

Experimental Methodology and Testbed for Interactive Simulation

7.1 Introduction

The implementation of a PC-based multithreaded architecture made possible the design and realization of a real-time robotic system with the capabilities to provide sensor-based assistance and haptic manipulation of real and virtual objects. In this chapter, the experiments conducted to validate the control strategies with the actual hardware are described. The testing of the system was conducted on healthy people performing a "pick-and-place" task, which is a common activity of daily living (ADL) task. Three people were trained to use the Phantom Omni interface and to teleoperate the PUMA manipulator in all control modes to familiarize themselves with the system.

This Chapter presents the methodology used for the experiments with the actual hardware: a 6-DoF Puma 560 manipulator, a Phantom Omni haptic interface and the sensory suite consisting of a CCD camera, a Sick DT60 laser range finder and the PUMA encoders. The performance measures are defined by the "Absolute Position Error" (APE), the "Absolute Orientation Error" (AOE) indicators, and the task-completion time which are calculated using the recorded data sets for each experiment. The following list shows the different comparisons made using the APE and the AOE indicators for position and velocity based control modes:

www.manaraa.com

1. Autonomous Control Mode
2. Position-Based Regular Teleoperation
3. Position-Based Virtual Fixture Teleoperation
4. Position-Based Scaled Teleoperation
5. Velocity-Based Regular Teleoperation
6. Velocity–Based Virtual Fixture Teleoperation
7. Velocity-Based Scaled Teleoperation
8. Force-Based Virtual Fixture Teleoperation

Chapter 9 discusses and analyses the experimental data gathered for validating the trajectory tracking and assistive capabilities of the system for guiding the user's motion during execution and successful completion of the task.

7.2 Methodology for Experiments

As previously stated, the testing of the system was conducted on three healthy people performing a “pick-up-a-cup” task. After training the subjects to use the Phantom Omni interface, they moved the PUMA manipulator in all control modes. The test setup included a platform in front of the arm, with two markers indicating the pick-up position and the drop-off (destination) position. These two positions were offset from each other in all the three Cartesian directions as shown in Figure 7.1. A coffee cup was used as the intended target to be grasped and moved from the start to the end positions. The start position for all the experiments is kept constant and it is defined as the start position.

المنارة للاستشارات
www.manaraa.com

For each test, the position and velocity based teleoperation modes were compared to regular, scaled and virtual fixture based teleoperation modes in the following way:

1. Position-Based Regular teleoperation vs. Scaled teleoperation
2. Position-Based Regular teleoperation vs. Virtual Fixture
3. Position-Based Regular teleoperation vs. Autonomous
4. Velocity-Based Regular teleoperation vs. Scaled teleoperation
5. Velocity-Based Regular teleoperation vs. Virtual Fixture
6. Velocity-Based Regular teleoperation vs. Autonomous
7. Position-Based Regular teleoperation vs. Force-Based

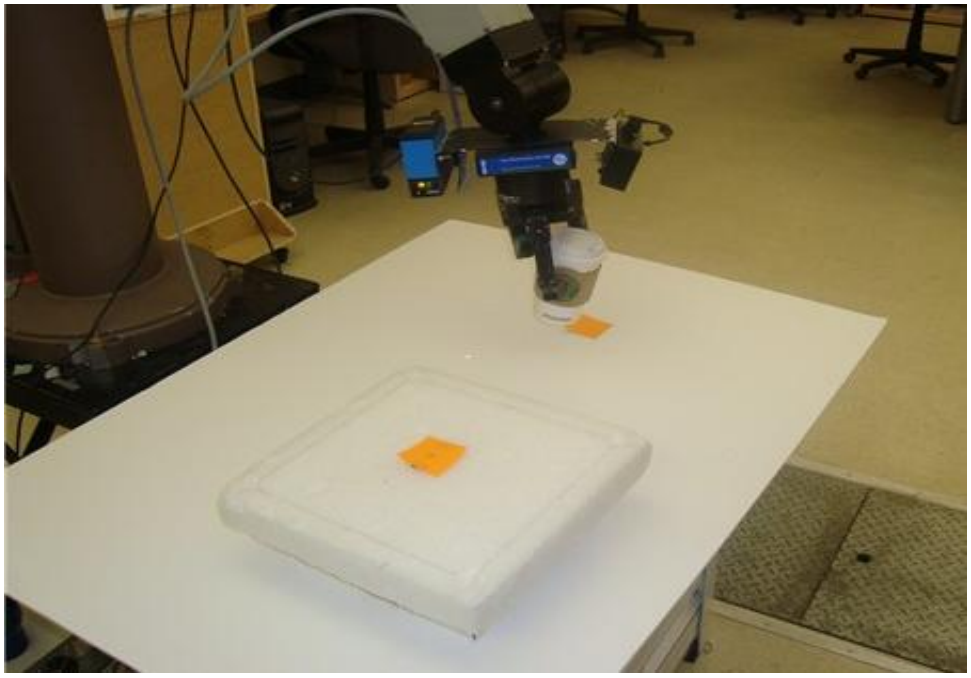


Figure 7.1 ‘Pick-up-a-cup’ Task Experimental Setup

المنارة للاستشارات
www.manaraa.com

When the user starts the operation under the supervision and observation of the attendant, the robot is commanded to go from the “parked” position to the “ready” position by the attendant. The user starts to control the arm from the “ready” position. The user always starts with the position-based teleoperation mode and then switches the test mode. While performing an ADL task the user can switch to any mode, however, for the purposes of testing the user toggles between the position-based teleoperation and the tested mode. The user has to toggle to position based teleoperation every time to orient the hand so that it is able to point to target objects, grasp the cup and drop the cup at the destination point as these steps require re-orientation of the end-effector. For automatic, scaled and virtual fixture based teleoperation modes, once the object is located by teleoperation, the user pushes the Phantom Omni stylus button to lock the target and generate the desired trajectory. Once the user reaches the target vicinity, the user teleoperates the arm to adjust the gripper and grasp the object. The user then points to the destination marker and pushes the Omni stylus button again to lock the destination coordinates and move in the same fashion to the drop-off point and release the object.

In the Scaled Teleoperation mode, the user input was scaled 3X when it was along the trajectory generated by the laser, and 0.2X when it was perpendicular to the trajectory. In the case of virtual fixtures, all positions and orientations coming from the user input were locked (scaled down to 0X) except the position parallel to the trajectory, which was scaled to 3X. Each control mode was tested five times, and the elapsed-time to complete the task was recorded. The trajectory generator thread generates a log file recording the transformation matrices of the tip, the elapsed time and the gripper status at every loop. Data from this file were conditioned, and used for data analysis.

للاستشارات المنارة
www.manaraa.com

7.3 Visual and Haptic Testbed to Control a 6-DoF Robot Arm

In the experiments the Phantom Omni Haptic interface from SensAble Technologies is used as the master. It is run on a Pentium computer, with 1GHz single processing unit. The Phantom Omni device uses the OpenHaptics software which runs on Windows XP OS. A Microsoft Visual Studio C++ program was developed to run the Phantom Omni controller and render the virtual environment using OpenHaptics [70] and OpenGL library functions as well as APIs. The commands for creating and interfacing the PUMA software controller and the Phantom Omni controller were also embedded in the same program. The protocol for sending and receiving information between the Omni and the PUMA controller is based on User Datagram Protocol (UDP) sockets. The UDP socket programming class implemented is a derived class from the Microsoft socket programming library.

The program running on the Omni controller is multithreaded. These threads include the main application thread, the graphics thread, the haptics thread, the collision detection thread (this thread runs on the background and it is responsible for collision among objects on the virtual environment and no real objects) and the communications thread for receiving data from PUMA controller. The main application thread starts the other threads, initializes the Phantom Omni, creates sockets for communication and integrates the whole application. The graphics thread renders the graphics scene at approximately 30 Hz refresh rate. This graphics scene is a virtual environment that helps the user to engage and disengage the PUMA in teleoperation (Figure 7.2). The haptics thread provides the haptics feedback to the user at a refresh rate of 1000 Hz and the collision detection thread does the computations for haptics force rendering.

www.manaraa.com

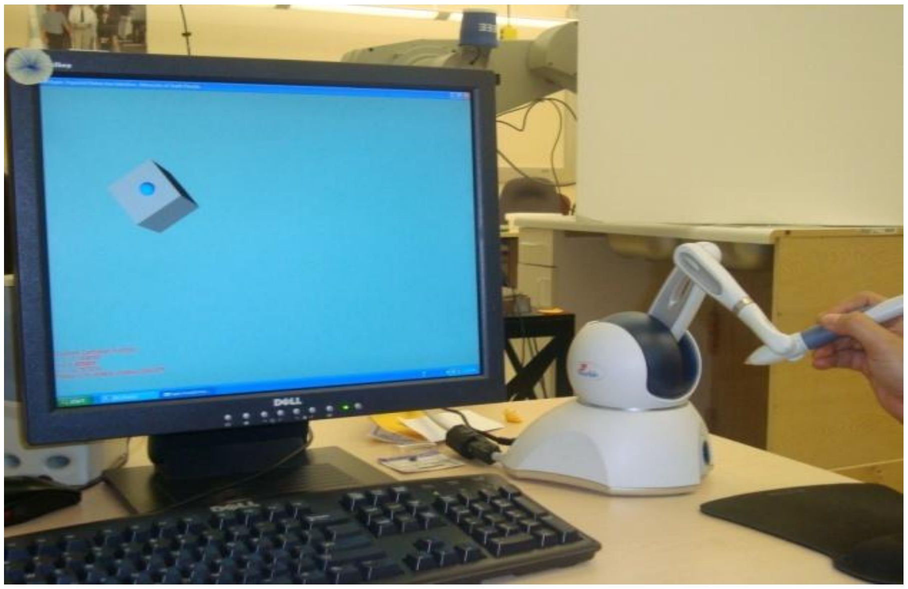


Figure 7.2 Virtual Environment for Teleoperation of the PUMA Manipulator

The teleoperated robot consists of a 6-DoF Puma 560 manipulator. As explained in Chapter 3, the Puma software controller is a form of a PD plus gravitational compensation strategy controller. The robot arm is equipped with a modified QC MP Orbit camera (an off-the-shelf USB camera) and a Sick DT60 laser range finder (See Appendix G) as shown in Figure 7.3. In its original format, the camera was not suitable to be mounted at the wrist of the robot arm and a new case was built to accommodate the integrated circuit, the lens and cables. Also, the face detection and auto-zoom features of the MP Orbit model were turned off in order to implement the calibration procedure described in Chapter 5. This software runs on a Dual-core computer with Windows XP OS. The sensors (the camera and Sick DT60 laser range finder) and a 4-DoF Barrett Hand (Figure 7.3) were attached to the wrist of the Puma 560 manipulator.

www.manaraa.com

Logitech MP Orbit™ CCD Camera	Sick DT60 Laser Range Finder	Phantom Omni Haptic Device	3- Fingers Barrett Hand

Figure 7.3 Sensory Suite Devices

Figure 7.4 shows the camera and the DT60 laser as they are mounted on the wrist of the Puma 560 robot arm in the experimental setup. The Barrett hand is also shown.

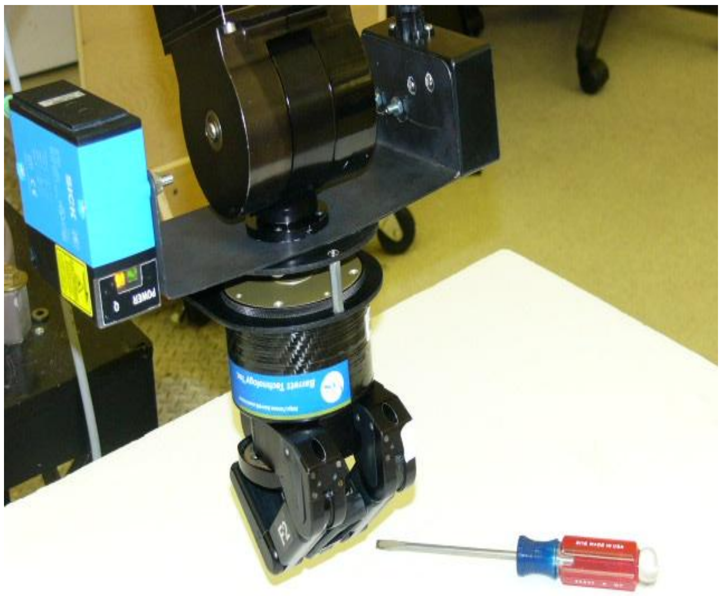
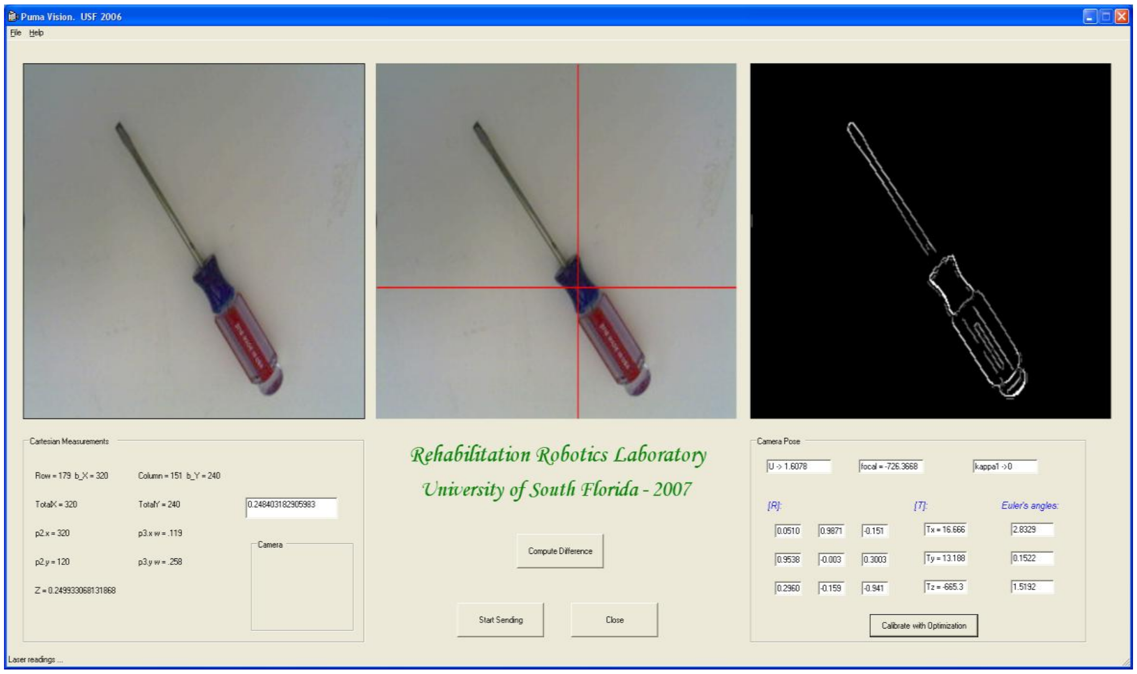


Figure 7.4 Camera and the Sick DT60 Laser Range Finder Mounted at the Puma's End-Effector

www.manaraa.com

As shown in Figure 7.5, when the user operates the robot arm and locates an object of interest, a stream of images of the object in the field of view is processed for geometrical information computations.


Puma Vision. USF 2006
File Help
Rehabilitation Robotics Laboratory
University of South Florida - 2007
Compute Difference
Start Sending
Close
Calibrate with Optimization
Laser readings ...

Cartesian Measurements

Row = 179 b_X = 320 Column = 151 b_Y = 240

TotalX = 320 TotalY = 240 0.248403182905983

p2.x = 320 p3.x w = .119

p2.y = 120 p3.y w = .258

Z = 0.249933068131868

Camera

Camera Pose

U -> 1.6078 focal = -726.3668 kappa1 ->0

[R]:			[T]:	Euler's angles:
0.0510	0.9871	-0.151	Tx = 16.666	2.8329
0.9538	-0.003	0.3003	Ty = 13.188	0.1522
0.2960	-0.159	-0.941	Tz = -665.3	1.5192

Figure 7.5 Results of the Segmentation and Feature Extraction Processes

The segmentation and the feature extraction processes that take place are also shown in Figure 7.5. As shown, the first window to the left presents the object as seen from the camera. The crosshair lines, overlaid in the centered image, are used to emphasize the centroid of the object of interest with respect to the screen coordinate system located at the top-left corner of the viewport. The black and white image to the right is the image resulted after applying the edge detection algorithm. As mentioned, the system includes two algorithms for edge detection for added flexibility: Sobel and Canny. However, only one of these edge detection algorithms must be active when the experiments are

المنارة للاستشارات
www.manaraa.com

performed. The Canny edge detector is used in the presented computations because of its capabilities to smooth the image and to filter noise in the original image.

7.4 Haptic Interface and Cartesian Motion

During teleoperation of the robot arm through the haptic interface, the real-time controller receives the latest position and velocity updates from a virtual environment as shown in Figure 7.6.

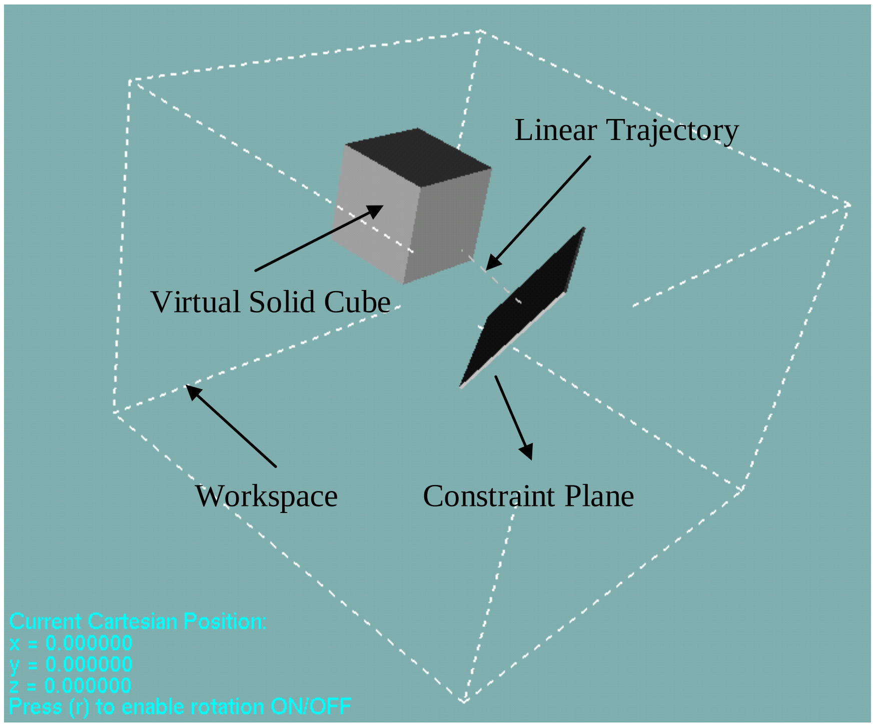


Figure 7.6 Virtual Environments and 3D Constraint Plane for Haptic Control

As explained before, the user engages the Puma using the toggle buttons available to the user. The Phantom Omni control software uses the input from the two buttons located on the Phantom Omni stylus. The “white button” is used for

www.manaraa.com

teleoperating the Puma manipulator and the “blue button” for indexing. For instance, the user can use the “blue button” on the stylus to index the virtual cube as shown in Figure 7.6. This way, the user can move the cube to the center of the screen when it is needed and re-engage the manipulator with more screen space available in the virtual environment.

7.5 Performance Measures

The performance measures defined in this work are associated with the trajectory tracking when position-based or velocity-based control modes are active. In this case, two performance indices were used to measure the error associated with the position and orientation in regular, scaled, and virtual fixture teleoperation. The performance measures were defined by the "Absolute Position Error" (APE) and the "Absolute Orientation Error" (AOE) indicators. The following list shows the different comparisons made between the different APE and the AOE indicators for position and velocity based control modes:

1. Autonomous, Force-based, and Motion-based Virtual Fixture Teleoperation
2. Force-based Virtual Fixture, Regular, Scaled, and Virtual Fixture Teleoperation
3. Autonomous, Velocity-Based Scaling, Velocity-Based Virtual Fixture, and Force-based Virtual Fixture
4. Position-Based Regular teleoperation vs. Scaled teleoperation
5. Position-Based Regular teleoperation vs. Virtual Fixture
6. Position-Based Regular teleoperation vs. Autonomous

www.manaraa.com

7. Velocity-Based Regular teleoperation vs. Scaled teleoperation
8. Velocity -Based Regular teleoperation vs. Virtual Fixture
9. Velocity -Based Regular teleoperation vs. Autonomous

Each task was repeated five times for each mode of operation and the calculations for the associated indicators of the Absolute Position Error as well as the Absolute Rotation Error were based on the following definitions.

7.5.1 The Absolute Position Error (APE)

This performance measure defines the error between the commanded linear position components (x_i^c, y_i^c, z_i^c) and the actual position achieved by the software controller (x^f, y^f, z^f). In other words, the APE is the Cartesian distance between the desired and the actual end-effector position [70]. This measure is obtained by the evaluation of Eq. 7.1 as follows:

$$error_{pos} = APE = \sqrt{\left(x_i^c - x^f\right)^2 + \left(y_i^c - y^f\right)^2 + \left(z_i^c - z^f\right)^2} \tag{7.1}$$

where (x_i^c , y_i^c , z_i^c) are the current 3D coordinates of the robot's end-effector in the base frame of the manipulator and (x^f , y^f , z^f) are the achieved 3D coordinates of the drop-off point (destination), also with respect to the base frame. Figure 7.7 shows the absolute position error when the robot arm is commanded in simulation to follow a straight line trajectory between the goal position and a target situated 15.0 cm away from the initial position of the end-effector.

المنارة للاستشارات
www.manaraa.com

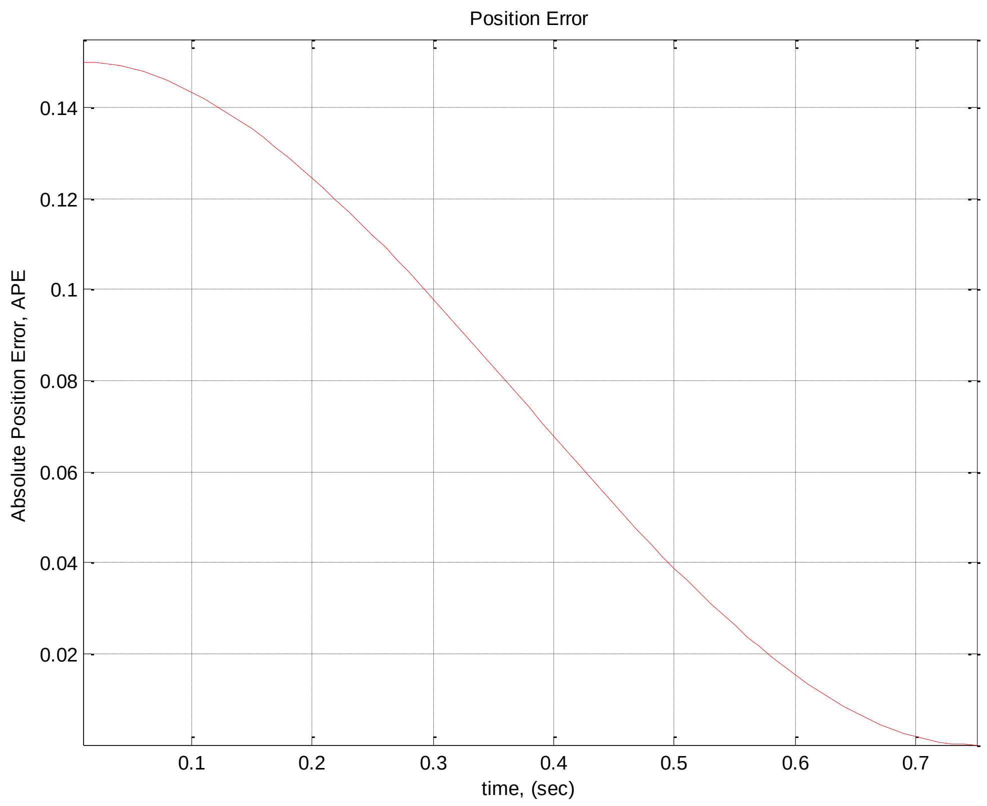


Figure 7.7 Absolute Position Error (APE)

7.5.2 The Absolute Orientation Error (AOE)

This performance measure defines the error related to the rotation matrix elements (r_{ij}) as described in Chapter 3. It specifies an equivalent single axis rotation angle about a vector defined between the desired and the current rotation of the end-effector of the robot arm [70]. Equation 7.2 defines the rotation error:

$$error_{ori} = AOE = abs\left(\cos^{-1}\left(\frac{trace\left(R_f R_c^T\right)-1}{2}\right)\right) \tag{7.2}$$

where

المنارة للاستشارات
www.manaraa.com

R_f = (3x3) achieved rotation matrix at the destination (defined as the DROP-OFF POINT) and

R_c = (3x3) current rotation matrix evaluated at each time interval.

The trace function in Eq. (7.2) corresponds to the sum of the diagonal elements of the product of the achieved R_f and current rotation R_c matrices, which is also the sum of the eigenvalues of the product $R_f R_a^T$. The angle expressed by $\left(\frac{trace\left(R_f R_c^T\right)-1}{2}\right)$ specifies an equivalent single angle rotation about a vector defined between the final and the current orientation of the end-effector of the manipulator.

Figure 7.8 shows the results of the evaluation of Eq. 7.2 in an offline program in MatLab. As before, absolute orientation error is calculated for the straight line trajectory defined between the goal position and a target situated 15.0 cm away from the initial position of the end-effector. As can be observed, the maximum orientation error obtained is about 0.000001 radians. Given that the initial orientation was zero, it should be expected that the orientation error to also be zero. However, accumulated errors in the computation prevent this from happening in the simulation.

المنارة للاستشارات
www.manaraa.com

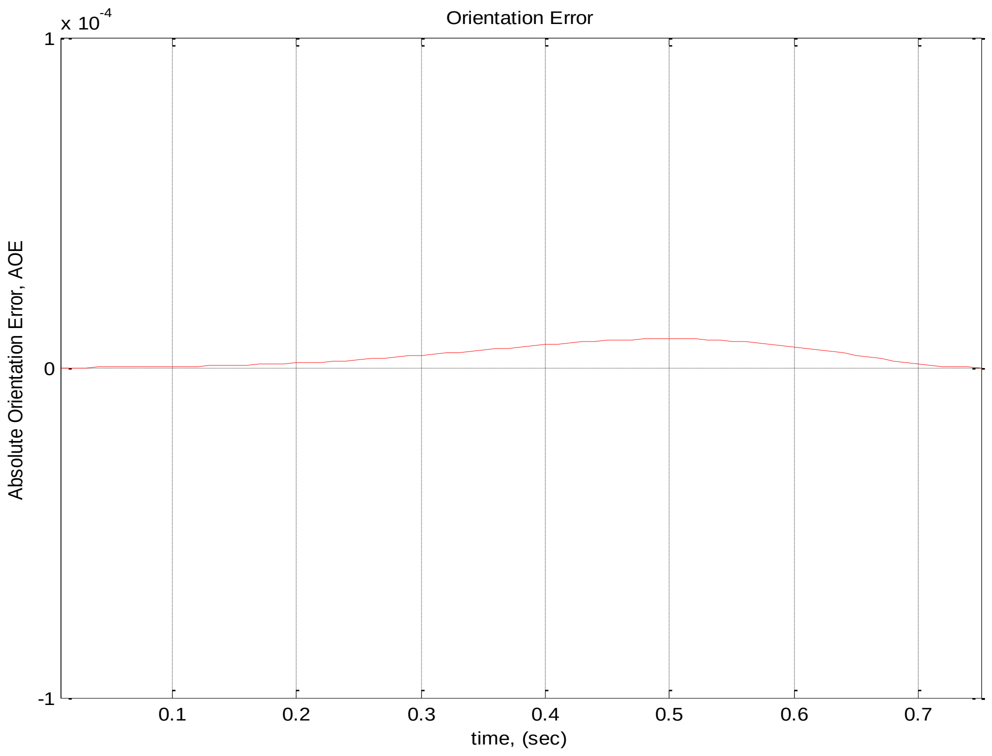


Figure 7.8 Absolute Orientation Error (AOE)

The following steps describe the process after recording every user interaction in autonomous and teleoperation control modes:

1. During regular teleoperation, the system does not use the external sensory input for assisting the user's motion. For automatic, scaled and virtual fixture teleoperation modes, once the object is located by using teleoperation mode, the user pushes the Omni stylus button to lock the target and generate the desired trajectory based on the sensory input. The user then teleoperates the robot arm using autonomous, scaled, or virtual fixture mode until the gripper reaches the target vicinity. Once the gripper reaches the target vicinity, the user teleoperates the arm to adjust the gripper and grasp the object. Then, the user uses regular

www.manaraa.com

teleoperation and points to the destination marker and pushes the Omni stylus button again to lock the destination coordinates and move in the same fashion to the drop-off point and release the object. In the case of force-based virtual fixture a “stick force” effect keeps the user on the straight line trajectory generated using the laser input.

2. The position (X, Y, Z) and the orientation angles (α, β, γ) of the end-effector of the Puma manipulator, as well as, the real-time timing are recorded in text files by the real-time application for all the experiments: autonomous, position-based, and velocity-based (regular, scaled and virtual fixture) teleoperation. The initial (START POINT), the pick-up point (PICKUP POINT) and the drop off (DROP POINT) are also recorded in the text file.
3. The recorded data $(X, Y, Z, \alpha, \beta, \gamma)$ are then transferred to the visualization application in MatLab for plotting and further analysis. The transferring of the angles is more efficient than transferring the assembled (3x3) rotation matrix as registered by the real-time software.
4. For every recorded configuration, a 3D plot showing the 3D Cartesian position (X, Y, Z) is obtained. It is important to mention that, even if the autonomous mode is being tested, there is a small part of the trajectory for which the user needs to switch back to regular teleoperation in order to re-orient and to avoid an obstacle intentionally placed between the pick-up and drop points. Once the obstacle is avoided, the user can switch back to autonomous, or any of the tested modes. For instance, Figure 9.3 presents the case where the user switched back to autonomous mode for the last portion of the path to the drop-off point.

www.manaraa.com

5. The (X, Y, Z) coordinates of the end-effector from the START POINT to DROP POINT are used to calculate the "Absolute Position Error", APE, as given by Eq. (7.1). The result from Eq. (7.1) will then correspond to the traveled distance from start to destination. This value can be used as an indicator to measure which teleoperation mode can achieve the destination by traveling the lesser distance as a function of time. For instance, this measure is used to compare the regular teleoperation mode, which provides no assistance, to the autonomous, scaled, force-based and motion-based virtual fixture teleoperation modes.
6. The calculation of the "Absolute Orientation Error" (AOE) is more involved. First, the Euler's angles (α, β, γ) are used in the offline program to compute the rotation matrix (the details are shown in Appendix E). Eq. (7.2) is then evaluated at every sampled point recorded in the text file.
7. The APE and AOE measures of the tested control modes described in section 7.2 are plotted versus time and comparisons are made to determine the effectiveness of the assistance provided to guide the user's motion to accomplish the task.

For both performance indicators the area under the curve represents a measurement of the distance traveled (START POINT to the DROP POINT) and the time to complete the pick-up-a-cup task. By comparing the area covered autonomous control mode, force and motion-based virtual fixtures, and scaled teleoperation experiments it is possible to determine the effectiveness of each form of control for completing the pick-up-a-cup task and others ADL tasks. This area can be determined by numerically integration of the APE curve using a fixed increment of time Δt as registered by the real-

المنارة للاستشارات
www.manaraa.com

time system. The smaller the area, the better the effectiveness of the method for accomplishing the pick-up-a-cup task.

7.6 Summary

In this chapter, the methodology followed to conduct the experiments as well as the experimental testbed was described. The performance measures were also defined. A pick-up-a-cup task, a common activity of daily living (ADL), is used as the testing task. Eight testing scenarios were defined for position-based and velocity-based control modes for later analysis. The performance corresponding to autonomous control, regular, scaled, force-based and motion-based virtual fixture teleoperation modes is defined in terms of the “Absolute Position Error” (APE) and the “Absolute Orientation Error” (AOE). The area under APE curve can be used as a qualitative indicator for comparing each of the operation modes. Results including these comparisons are presented later in Chapter 9.

المنارة للاستشارات
www.manaraa.com

Chapter 8

Virtual Reality Simulation Testing

8.1 Introduction

In robotics, once the governing equations of robot arm motion are defined in terms of the virtual object variables, a computer-generated version of the real robot arm can be used for testing the control strategies without the dangers of damaging the hardware. Virtual Reality, VR, provides a widely accepted computer interface that enables realistic simulations of physical systems.

In the case of a robot arm, both the forward and inverse kinematics solutions can be defined in terms of the joint angles of the virtual reality standard transformations defined by the scripting language known as Virtual Reality Markup Language (VRML). In practice, the appropriate mapping of the Cartesian axes between the reference frames defined for the robot arm and the haptic device can be easily visualized in the virtual environment by moving the haptic stylus or through a graphical user interface. This way, the inherent complexity of the design and testing of a real-time controller with a haptic interface directly on the physical system can be reduced by performing probe of concepts of many of the programming tasks with realistic and believable visualizations and simulations. In this chapter, the haptic control of the Puma 560 model using the VR techniques is presented as well as the communication protocol developed in order to resolve the high timing demands of the haptic loop and the integration of the different programming workspaces.

www.manaraa.com

8.2 Virtual Reality Simulation of the Puma 560 Manipulator

Virtual Reality simulation of the robot arm enables the design and testing of sophisticated control strategies in a "proof of concept" sense without the dangers of damaging the real robot arm. As discussed in Chapter 4, the teleoperation tasks are executed through the use of the Phantom Omni for force feedback and the Puma 560 robot arm interface which has a very different kinematics compared to the Omni. The resulting transformations from the evaluation of their respective kinematics equations need to be mapped (in joint space or Cartesian space). For simulation of the VR robot arm motion, both the forward and inverse kinematics solutions can be defined in terms of the joint angles of the virtual reality transformations (known as a "Transform" object in the VRML script language). The appropriate mapping of the Cartesian axes between the reference frames defined for the robot arm and the haptic device can be easily visualized in the virtual environment.

In this work, the visualizations of the motion of the Puma 560 (with and without haptic control) were realized using VR toolbox as shown in Figure 8.1. The VR toolbox is an add-in library used for the creation and visualization of virtual models within the MatLab/Simulink workspace. This toolbox allows complete control of the scripting files associated to the different parts of the robot construction (links, joints, base stand, and end-effector). The VR toolbox follows the VRML97 standard which means that 3D CAD modeling software such as SolidWorks can be used to create the solid models. The CAD model (parts and assembling) can then be ported to the VRML97 format following a straightforward procedure.

المنارة للاستشارات
www.manaraa.com

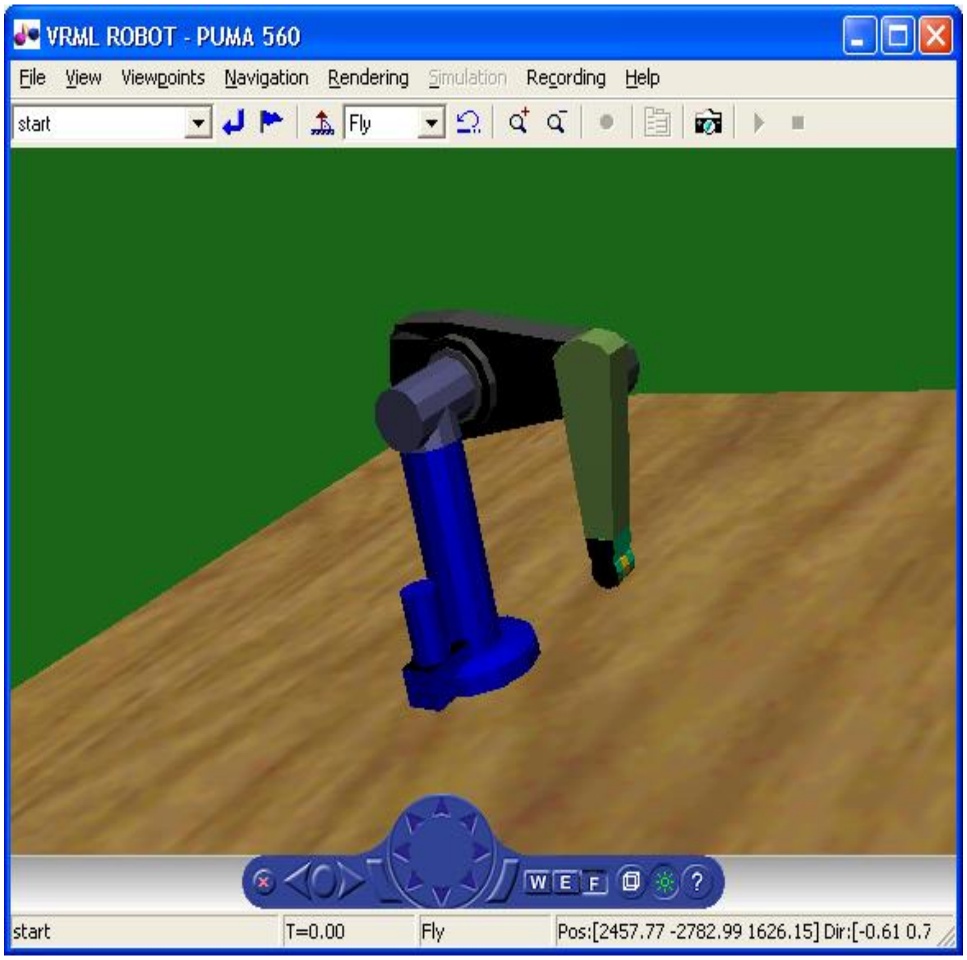

VRML ROBOT - PUMA 560
File View Viewpoints Navigation Rendering Simulation Recording Help
start
Fly
start
T=0.00
Fly
Pos:[2457.77 -2782.99 1626.15] Dir:[-0.61 0.7

Figure 8.1 Virtual Reality Model of the Puma 560 Robot Arm

8.3 Control of the VR Model of the Puma 560 Manipulator

The VR model of the Puma 560 can be driven in two different ways. One way is using a simple graphical user interface (GUI) as shown in Figure 8.2. This option enables the user to perform the virtual simulations of the robot arm using purely robotic mode (without the haptic interface). The GUI was developed as a control panel with toggle buttons and scroll bars for this form of operation. As shown, the graphical user interface (GUI) presents toggle buttons for the selection of the type of control, either joint or Cartesian space. This GUI provides an intuitive interface to the user and the toggle

المنارة للاستشارات
www.manaraa.com

bottom action prevents from trying to activate the two types of available control modes simultaneously.

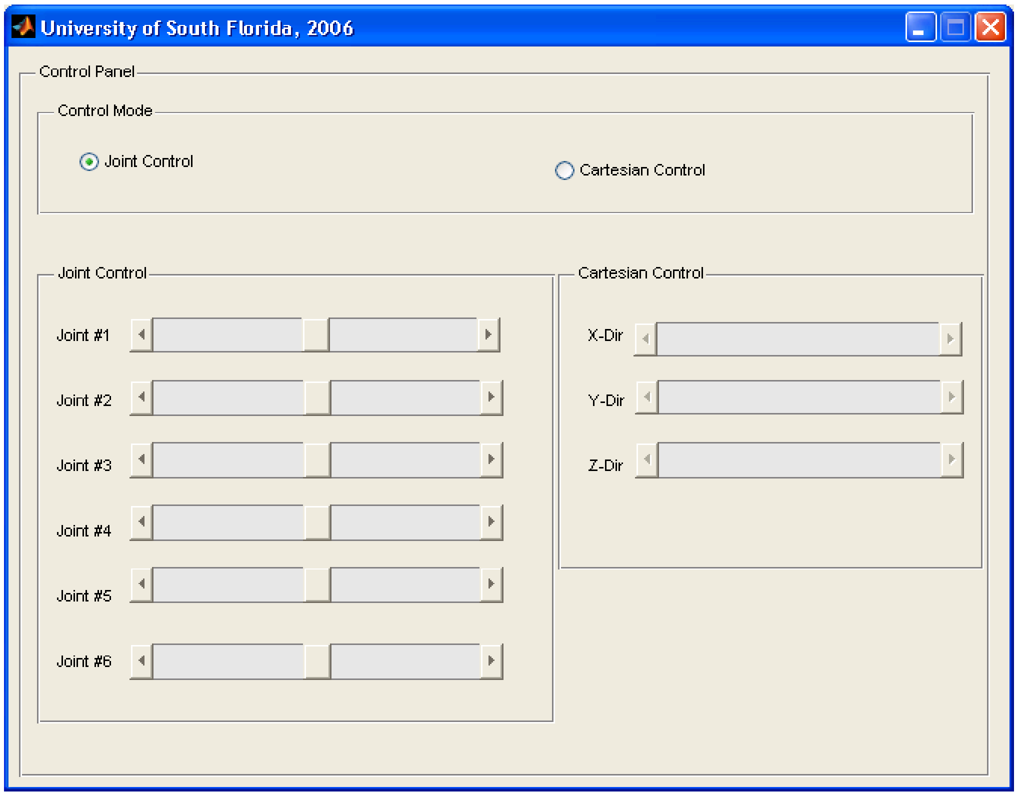


Figure 8.2 Control Panel for Joint and Cartesian Space VR Simulations

If the "Joint Control" toggle bottom is activated on the control panel, the scroll bars can be used to change each individual joint angle value in increments of 1 deg. The minimum value of the scroll bar is zero and the maximum value corresponds to the joint limit as defined in the real robot arm configuration files. In this case, the homogenous transformation matrices are evaluated (See Appendix A) and the results are assigned to the corresponding joint transformation matrix in the VRML script file. On the other hand, if the "Cartesian Control" toggle bottom is activated, the user is able to move the

المنارة للاستشارات
www.manaraa.com

end-effector along the 3D axis directions (X,Y,Z) and the solution of the inverse kinematics problem is required. In this case, two solutions were implemented. The first one is a "closed-form" solution available for the Puma560 and the resolved-rate algorithm based on the inverse Jacobian of the robot arm. This solution is more convenient when a closed-form solution is not available, as it is the case for kinematically redundant-robot arms. The details of this algorithm can be found in Chapter 3. The second one is using the haptic device for teleoperation of the virtual model of the robot arm as shown in Figure 8.3. In this case, the user is provided with a virtual environment where a solid object (red) is displayed and the user can "touch" with the Omni's stylus. A separate window is then shown with the VR model of the Puma 560 tracking the "haptic tip" of the Phantom Omni device when the cube is "grasped" with the stylus.

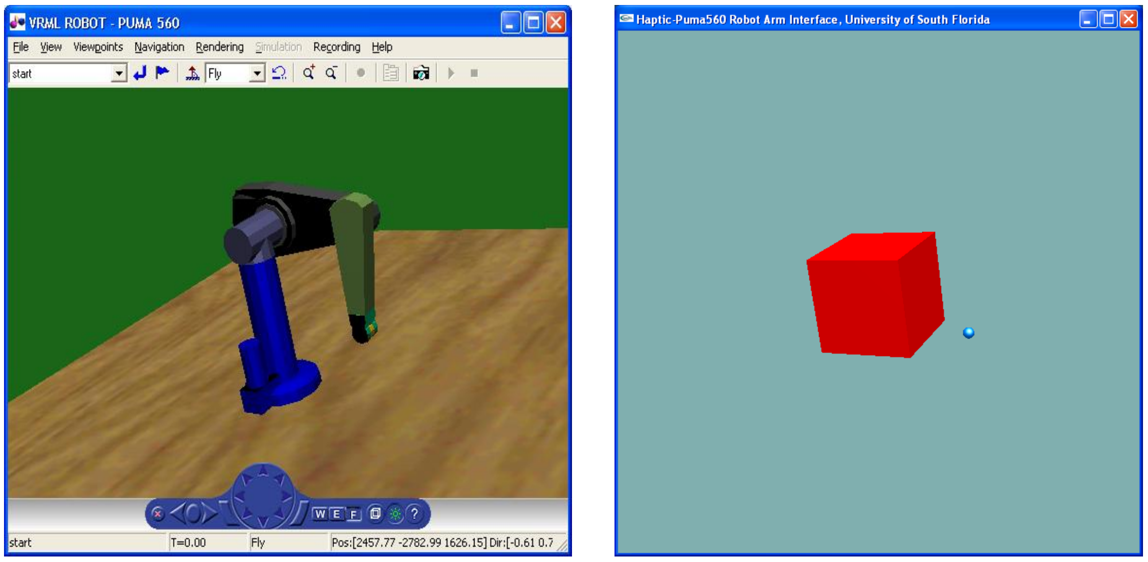

VRML ROBOT - PUMA 560
File View Viewpoints Navigation Rendering Simulation Recording Help
start
Fly
start
T=0.00
Fly
Pos:[2457.77 -2782.99 1626.15] Dir:[-0.61 0.7
Haptic-Puma560 Robot Arm Interface, University of South Florida

Figure 8.3 Haptic-VR Puma 560 Graphical User Interface

www.manaraa.com

8.4 VR Linear Trajectory Simulation

A major benefit of the VR toolbox in MatLab, in addition to the visualization capabilities, is the availability of robust built-in numerical functions for linear algebra, inverse and pseudo-inverse algorithms, optimization and singular value decomposition, among others. Taking advantage of these capabilities and, in preparation for the implementation of the real-time trajectory generation in QNX, a MatLab script program was developed in order to compare the results from the VR simulation and the actual physical implemented in C++ code.

The algorithm for the linear trajectory is based on the Equivalent Single Axis Rotation Method with provisions taken to avoid representational singularities (See Appendix B). Once the linear trajectory is generated, the required torques to drive the arm to the final destination needs to be computed. As discussed in Chapter 3, the implementation of the resolved-rate control technique involves the computation of the Jacobian and the inverse of the Jacobian of the robotic arm.

In QNX, all required numerical solutions must be implemented in C++ and the results need to be validated. The availability of the results from the simulation makes it easier to debug potential errors during the computation of the different numerical algorithms in C++ running under QNX O/S.

In MatLab, the script requires a homogenous transformation matrix defining the initial position and orientation of the end-effector and the final transformation matrix defining the desired (goal) destination as input arguments. Both transformation matrices are described relative to the base reference frame of the manipulator. Also, the script expects the desired linear speed of the end-effector as an input argument (0.2 m/s in this

www.manaraa.com

simulation). The following results were obtained by commanding the VR model of the Puma 560 to travel from its predefined ready (initial) position to the predefined destination. The corresponding homogenous transformation matrices are:

$$^{0}T_{initial} = \begin{bmatrix} -0.6206 & 0.7621 & 0.1843 & 0.1158 \\ 0.7806 & 0.5785 & 0.2365 & 0.6254 \\ 0.0736 & 0.2907 & -0.9540 & -0.3387 \\ 0.0000 & 0.0000 & 0.0000 & 1.0000 \end{bmatrix} \qquad (8.1)$$

$$^{0}T_{goal} = \begin{bmatrix} -0.6206 & 0.7621 & 0.1843 & 0.1434 \\ 0.7806 & 0.5785 & 0.2365 & 0.6609 \\ 0.0736 & 0.2907 & -0.9540 & -0.4818 \\ 0.0000 & 0.0000 & 0.0000 & 1.0000 \end{bmatrix} \qquad (8.2)$$

The specified initial and goal transformations correspond to 15.0 cm displacement of the end effector from its initial position along its own z-axis. Figure 8.4 shows the required joint angles of the manipulator and Figure 8.5 shows the commanded linear trajectory. This is an important validation phase before using the Phantom Omni differential transformations are used to command motion actions to the Puma manipulator.

www.manaraa.com

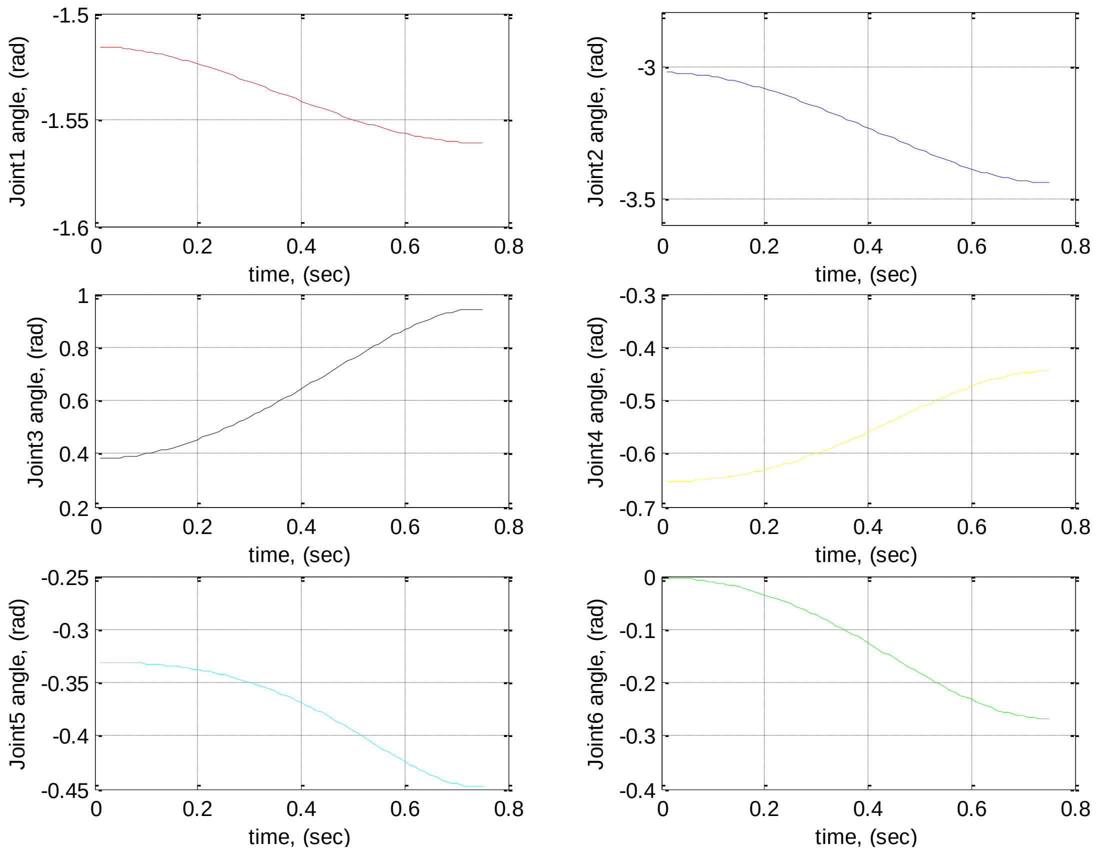


Figure 8.4 Required Joint Angles for the Predefined Linear Trajectory Path

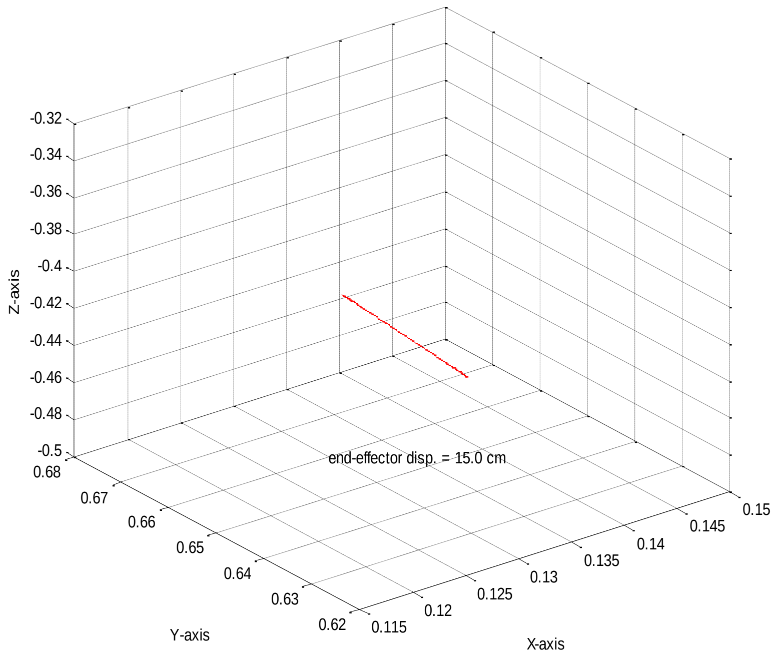
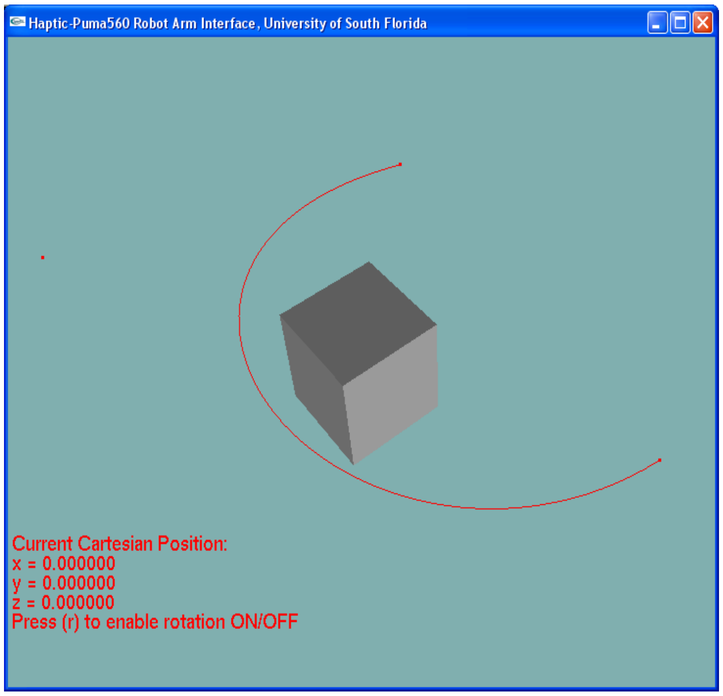


Figure 8.5 End-Effector Displacements from Initial to Goal Position

المنارة للاستشارات
www.manaraa.com

8.5 Haptic Feedback and Assist Functions in Simulation

Figure 8.6 shows a simulation of a haptically rendered cube and Bezier-type curve trajectory where features of OpenGL, HLAPI and HDAPI libraries are combined for the simulation of a teleoperation task. The solid cube was created using graphic functions available through the OpenGL graphic and HLAPI libraries. On the other hand, the Bezier points were generated using the classical algorithm in C++, and then, displayed using OpenGL vertex structures.


Haptic-Puma560 Robot Arm Interface, University of South Florida

Current Cartesian Position:
x = 0.000000
y = 0.000000
z = 0.000000
Press (r) to enable rotation ON/OFF

Figure 8.6 Bezier Curve Trajectory and Haptically Rendered Cube

During the interaction, the user will approach the Bezier trajectory. The assistance provided at this instant is a "stick" friction effect, running at the haptic servo loop update rates, which is activated when the user is at a close proximity (a distance

www.manaraa.com

equivalent to the radius of the sphere representing the haptic tip in the virtual environment) to the trajectory and a spring-damper force activated once the user is following the path. In other words, the haptic interface provides guidance by constraining the user's motion along the trajectory. The resultant force is transmitted to the user's hands through the Phantom Omni using the method explained in Chapter 4. In this simulation, the haptic device is used for sensing proximity and for actuation in the form of force feedback to the user's hand. Typical "stick" friction forces are shown in Figure 8.7. Both original and filtered data are shown.

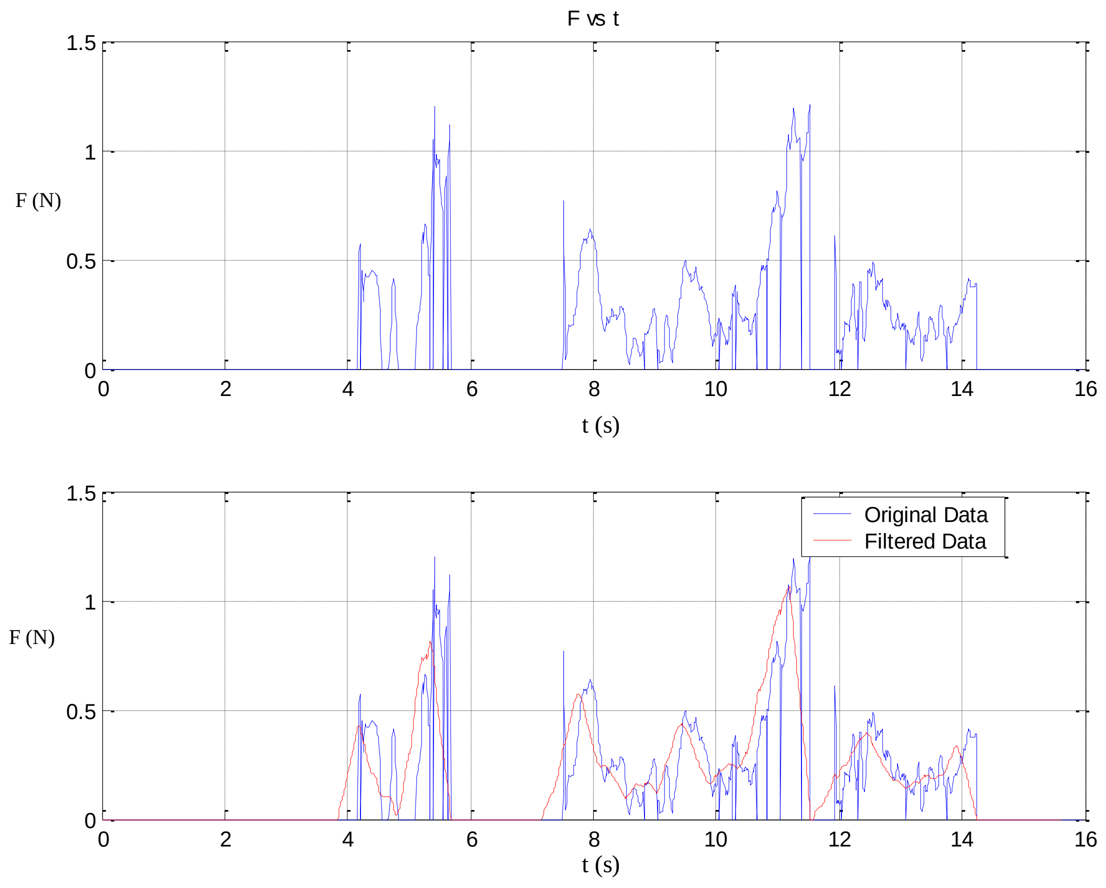


Figure 8.7 Experimental Data of Forces Resulting from a Typical Interaction

www.manaraa.com

8.6 Comments on the Haptic and VR Model Simulations

The integration of the VR toolbox with the different motion algorithms to drive the VR robot arm model in pure robotic mode occurs within the same MatLab workspace. Therefore, there is no communication issues involved. However, when the haptic control is integrated with the virtual reality environment (solid cube created with OpenGL) and the VR toolbox in MatLab, a different approach is required in order to make the virtual simulations responsive and stable at both ends.

As discussed in Chapter 4, the Phantom Omni model uses the OpenHaptics libraries for the Windows OS. To have access to those libraries, the C++ programming language is used. The VR simulation running on the MatLab environment needs to be interfaced with the HDAPI/HLAPI libraries for haptically rendering the OpenGL virtual objects in C++. A multithreaded application interface was developed to make the separate workspaces to communicate back and forth for data interchange. This component of the application is based on UDP sockets running as separate thread and the technique is further explained next.

8.7 Communication Protocol

As previously stated, the VR simulation and the haptic control software run in two different workspaces. A network protocol based on User Datagram Packets (UDP) was developed in order to interface the MatLab workspace used for the VR simulations and the C++ programming language used for the haptic control. A single packet contains the joint angles and the Cartesian position of the Phantom Omni's end effector needed to be

المنارة للاستشارات
www.manaraa.com

transferred to the MatLab workspace. As stated in Chapter 3, the protocol design includes features to deal with the possibility of data loses or out of order sequences.

For this particular implementation, a time-stamp variable was used to prevent these problems. The interfacing of haptic control and the VR simulation software implements four (4) main threads in C++ running simultaneously with different update rates. The different threads are:

1. The graphics thread.
2. The haptic loop thread.
3. The collision-detection thread.
4. The communication thread

Of these four threads, only the communication thread implementation is different from the physical simulation (as discussed in Chapter 3). This is due to the fact that MatLab does not provide functionalities for handling real-time clocks or synchronization mechanisms. The solution was to use regular timers and standard UDP-based socket programming techniques in the MatLab programming environment.

8.8 Comments on the Communication Protocol in the Simulation Program

The communication thread provided a stable and acceptable response time for interfacing VR simulations with the Phantom Omni controller when used for short periods of time. However, when the interface is used for extended time, the communication between the C++ application and the MatLab simulation is inconsistent and unreliable. The dynamic data exchange API responsible for transferring the UDP packets between the MatLab workspace and the sockets program in C++ fails to meet the

المنارة للاستشارات
www.manaraa.com

high timing constraints of the Phantom Omni and, at the same time, to update the virtual environment during the simulation. However, the interfacing between the VR simulation in MatLab and the OpenHaptics libraries in C++ creates a realistic look and appearance of the robot arm as well as a friendlier graphical user interface (GUI) for testing and debugging.

8.9 Summary

The use of the VR simulation provides a flexible visualization tool for testing the purely robotic control mode as well as the haptically driven manipulator. The virtual simulations allow validating the actual algorithms for teleoperation developed in C++ and the QNX RTOS. The capability of matching the homogeneous transformations resulting from the kinematics analysis and the transformations programmed in VRML scripts permits to experiment and develop more efficient interfaces and communication techniques. The implementation as well as the debugging processes of the different control algorithms and the required numerical approximation methods, both closed-form and resolved-rate, are greatly facilitated due the built-in linear algebra scripts available in MatLab and the visualization facilities available in the Virtual Reality Toolbox.

المنارة للاستشارات
www.manaraa.com

Chapter 9

Results and Discussion

9.1 Introduction

To evaluate the assistance enabled by the system to guide the user's motion, the proposed model was tested in eight different modes of operation. These modes consisted of regular, scaled and virtual fixtures using position based and velocity based control, autonomous mode, and force-based virtual fixture (for a total of 8), as described in Chapter 7. Each of these modes of operation comprised five repetitions of each experiment, for a total of forty (40) experiments. Three users executed these experiments for a total of 120 experimental data sets.

This Chapter presents the results of these experiments. Results and discussions of the virtual reality simulation are also presented in this chapter.

9.2 Interactive Simulations Results

The experiments were conducted based on the methodology presented in section 7.2. In all these experiments, when position-based control is activated, the user teleoperates the Phantom Omni interface to move the PUMA to the desired position and orientation. For instance, in order to select a target object using the laser pointer, the user will move the Omni tip to a configuration so that the PUMA end-effector points to the target object. On the other hand, when velocity-based control is activated, the Phantom Omni interface position determines the Puma end-effector speed and direction. In other

www.manaraa.com

words, when velocity control is used, the Puma end-effector speed changes proportionally to the Phantom Omni interface changing position. When the specified velocity is reached, it is maintained until the command from the Omni is changed. Under velocity control mode, the user will move the Omni's end-effector once to select a direction and speed for the Puma end-effector. Then, the user will hold the Omni's end-effector steady until the gripper mounted on the Puma is close to the target object, then move the Omni's end-effector to the center in order to stop close the target. The definitions of these experiments are described as follows:

a) Regular Teleoperation Mode: the user does not receive any assistance from the sensor-based assist system.

b) Scaled Teleoperation Mode: the user input is scaled 3x when it is along the trajectory generated by the laser, and 0.2X when it is perpendicular to the trajectory.

c) Virtual Fixture Teleoperation Mode: all positions and orientations coming from the user input are locked except the position parallel to the trajectory, which is scaled to 3X.

d) Autonomous Mode: the user points the laser in the direction of the target object and commands the Puma manipulator to follow the trajectory.

e) Force-based Virtual Fixture Mode: a "stick" force effect is used for maintaining the user moving along the straight line trajectory defined by the "line of sight" using the laser range finder.

المنارة للاستشارات
www.manaraa.com

Table 9.1 shows collected data of the time to complete the pick-up-a-cup task for ten repetitions using autonomous, regular, scaled and virtual fixtures using position based and velocity based control, and force-based virtual fixture teleoperation modes. The variables are defined as follows:

1. C1 = autonomous control mode
2. C2 = position-based regular teleoperation mode
3. C3 = position-based scaled teleoperation mode
4. C4 = position-based virtual fixture constraint
5. C5 = velocity-based regular teleoperation
6. C6 = velocity-based scaled teleoperation
7. C7 = velocity-based virtual fixture constraint
8. C8 = force-based virtual fixture constraint

Table 9.1 Completion Time (in seconds) for the Pick-up-a-cup Task

Experiment No.	C1	C2	C3	C4	C5	C6	C7	C8
1	86.549	82.058	69.243	74.322	71.230	82.288	78.382	80.949
2	86.214	88.105	102.300	92.718	80.681	79.143	79.990	66.764
3	98.342	87.114	95.975	79.582	70.778	81.129	80.849	68.850
4	85.255	92.069	69.630	86.085	74.315	88.941	76.833	79.776
5	94.995	77.443	71.129	53.457	63.775	71.469	64.575	68.552
6	68.592	86.214	109.892	78.522	76.064	84.615	84.835	78.213
7	73.647	88.105	90.282	96.207	93.846	77.063	74.046	84.389
8	65.670	94.862	91.182	98.683	76.953	83.948	82.158	77.473
9	67.654	109.590	89.762	101.060	60.270	78.322	64.525	94.596
10	65.097	88.848	84.878	80.340	62.398	67.932	71.958	79.910

المنارة للاستشارات
www.manaraa.com

Table 9.2 Completion Time Descriptive Statistics

Variable	N	N*	Mean	SE Mean	Std. Dev.	Minimum	Q1	Median	Q3	Maximum
C1	10	0	79.20	3.96	12.54	65.10	67.16	79.45	88.66	98.34
C2	10	0	89.44	2.71	8.57	77.44	85.18	88.10	92.77	109.59
C3	10	0	87.43	4.40	13.93	69.24	70.75	90.02	97.56	109.89
C4	10	0	84.10	4.50	14.24	53.46	77.47	83.21	96.83	101.06
C5	10	0	73.03	3.14	9.93	60.27	63.43	72.77	77.89	93.85
C6	10	0	79.49	1.98	6.25	67.93	75.66	80.14	84.12	88.94
C7	10	0	75.82	2.22	7.02	64.53	70.11	77.61	81.18	84.84
C8	10	0	77.95	2.65	8.38	66.76	68.78	78.99	81.81	94.60

Data from Table 9.2 were used to verify if the average time to complete the pick-up-a-cup task can be used as predictive parameter. For this purpose, a “boxplot” type of chart was used. The “boxplot” is a standard graphical tool used in descriptive statistics, to show the variability of a set of input variables without assuming any probability distribution of the underlying data [71].

The boxplot in Figure 9.1 shows that the time parameter will be a poor parameter if it is used as the only prediction parameter to identify which of the methods of control used to execute the task would perform better for this task. Also shown in Figure 9.1, is that the variability in the completion time of the pick-up-a-cup task is too large when comparing the different modes described as C1 to C8. Therefore, a different method of evaluation of results must be used to better explain the performance of the sensor-based assistive system.

المنارة للاستشارات
www.manaraa.com

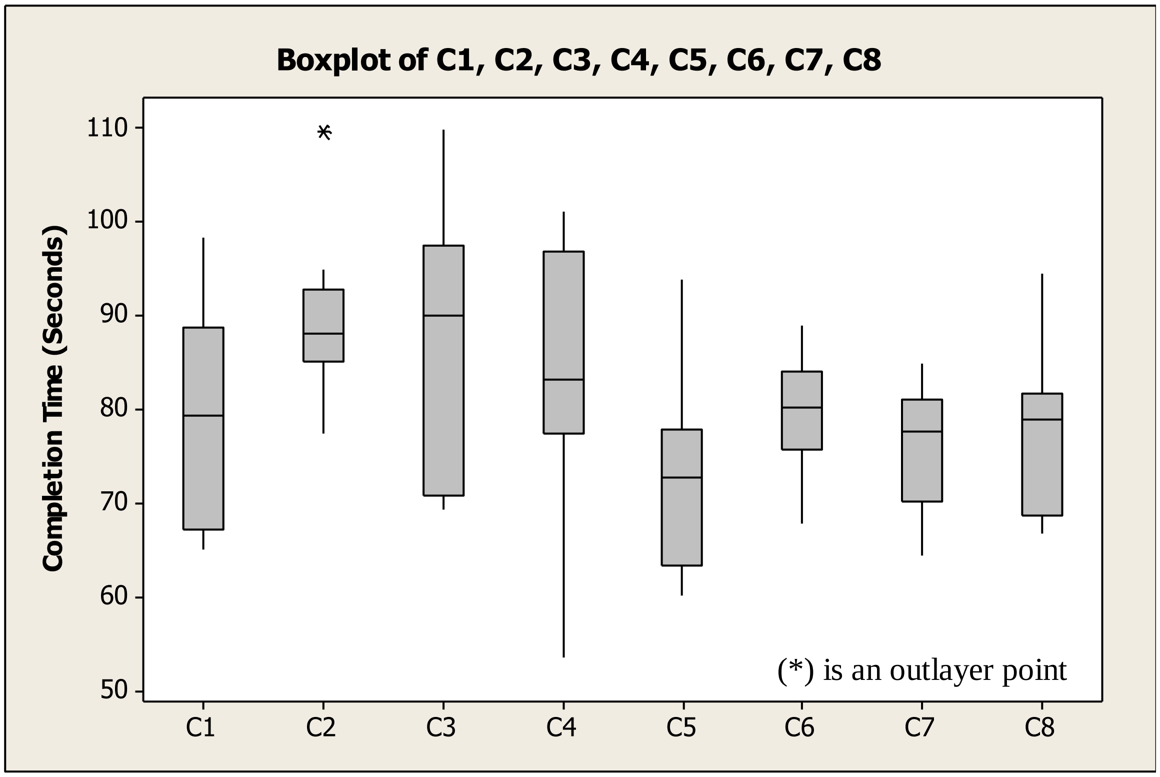


Figure 9.1 Boxplot of Autonomous (C1), Position-based Regular Teleoperation (C2), Position-based Scaled Teleoperation (C3), Position-based Virtual Fixture (C4), Velocity-Based Regular Teleoperation (C5), Velocity-Based Scaled Teleoperation (C6), Velocity-Based Virtual Fixture (C7), Force-based Virtual Fixture (C8)

In section 7.5 a definition of performance indicators was presented. By using these indicators, eight combinations of the operation modes can be defined. Each mode of operation was compared, and the associated Absolute Position Error (APE) and the Absolute Orientation Error (AOE) were plotted for one repetition of the experiment of the pick-up-a-cup task.

A qualitative assessment of results when the performance indicators were used is shown in Figures 9.2 to 9.20 for position-based control and Figures 9.21 to 9.39 for velocity-based control. The figures show the comparison between each of the four modes and the corresponding Absolute Position and Orientation Errors. From this qualitative

www.manaraa.com

comparison of the absolute errors in position and orientation, it is recognized that 1) scaling and virtual fixture teleoperation modes perform better than regular teleoperation and 2) autonomous mode performs better than regular, scaled, and virtual fixture either in position-based or velocity-based control forms. These are expected results from an assistive system where the user's motion is guided during the task execution.

9.2.1 Position-Based Control Interactive Simulations Results

The position-based teleoperation is the default control mode of the telerobotic system. In this case, the Phantom Omni is moved in its workspace by the user and transformation matrices are computed by solving the forward kinematics problem. The resulting transformations and then mapped to the PUMA base frame following the procedure discussed in section 4.2.2.

Although the same task was performed using different modes of operation, when Regular teleoperation mode was used, the trajectory was not as smooth and fast as it was in the case of Autonomous, Scaled and Virtual Fixture modes (Figures 9.2 to 9.4). Also, the trajectory is longer in Regular mode. Nevertheless, the trajectory in the Autonomous compared to Virtual Fixture mode and also in the Scaled compared to Virtual Fixture, is similar (Figures 9.5 and 9.7). When comparing Autonomous to Scaled, the trajectory is shorter and smoother for the Autonomous mode (Figures 9.6). This latter is mostly due to the fact that in Autonomous mode the input from the user is partly removed and only used for re-orienting the end-effector of the manipulator. These same results were obtained when comparing the Absolute Position Error (Figures 9.8 to 9.13).

المنارة للاستشارات

www.manaraa.com

As for the Absolute Orientation Error, the errors in the Regular mode for the complete task are mostly higher than the Autonomous, Scaled and Virtual Fixture (Figures 9.14 to 9.16). In the Autonomous and Virtual Fixture modes, some portions of the errors are constant (Figures 9.16 to 9.20). The explanation for this behavior is that those portions represent the sections of the trajectory where the orientation of the end effector of the Puma manipulator remains unchanged.

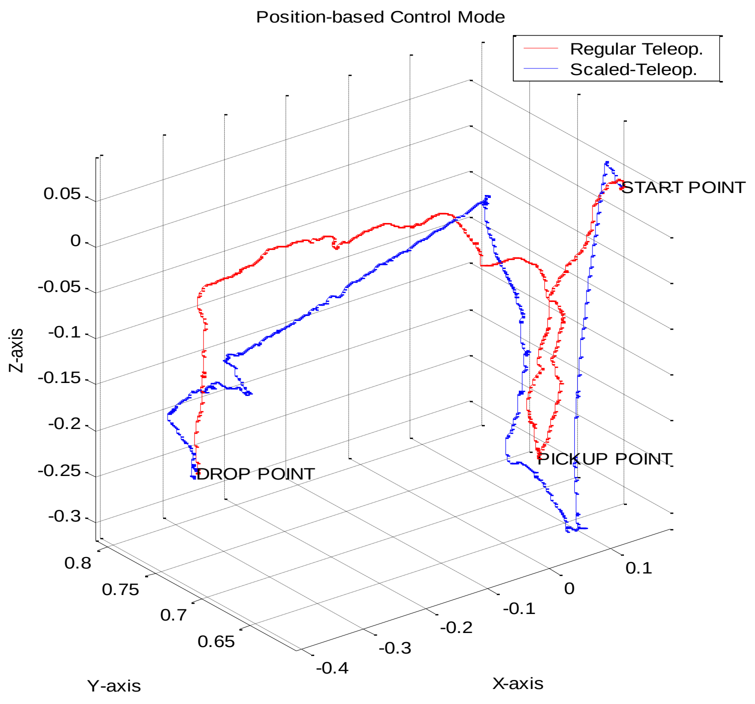


Figure 9.2 Position-Based Regular Teleoperation vs. Scaled Teleoperation

www.manaraa.com

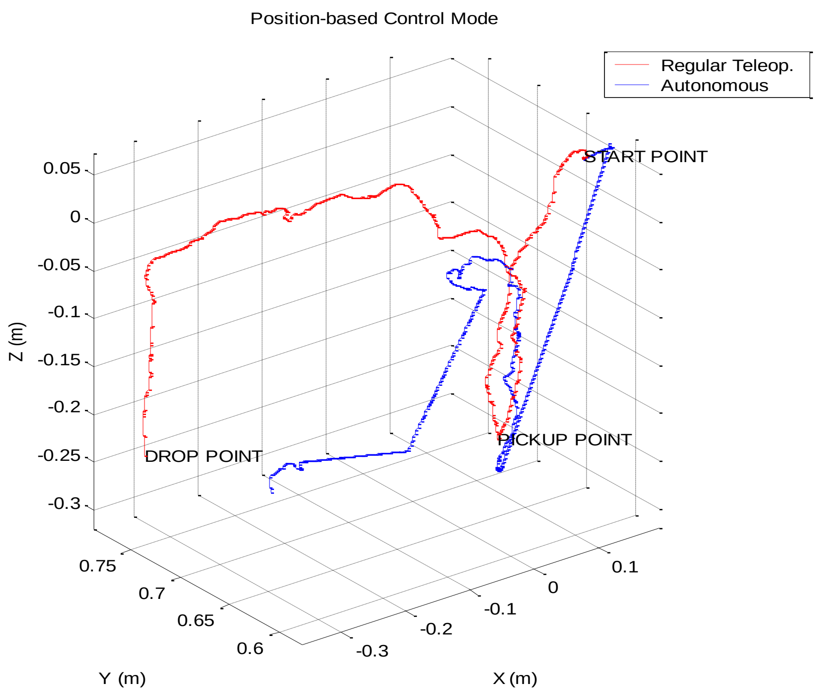


Figure 9.3 Position-Based Regular Teleoperation vs. Autonomous Control

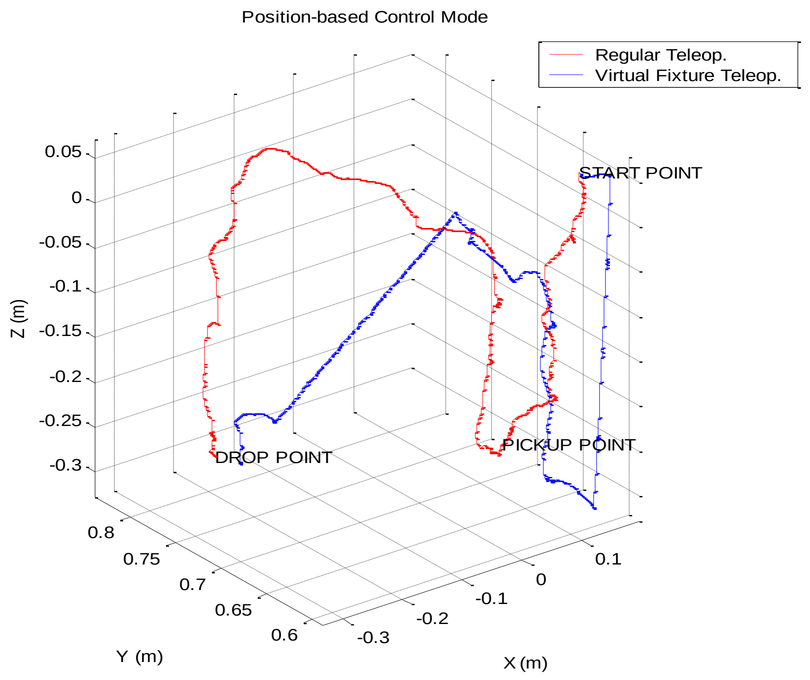


Figure 9.4 Position-Based Regular Teleoperation vs. Virtual Fixture Teleoperation

www.manaraa.com

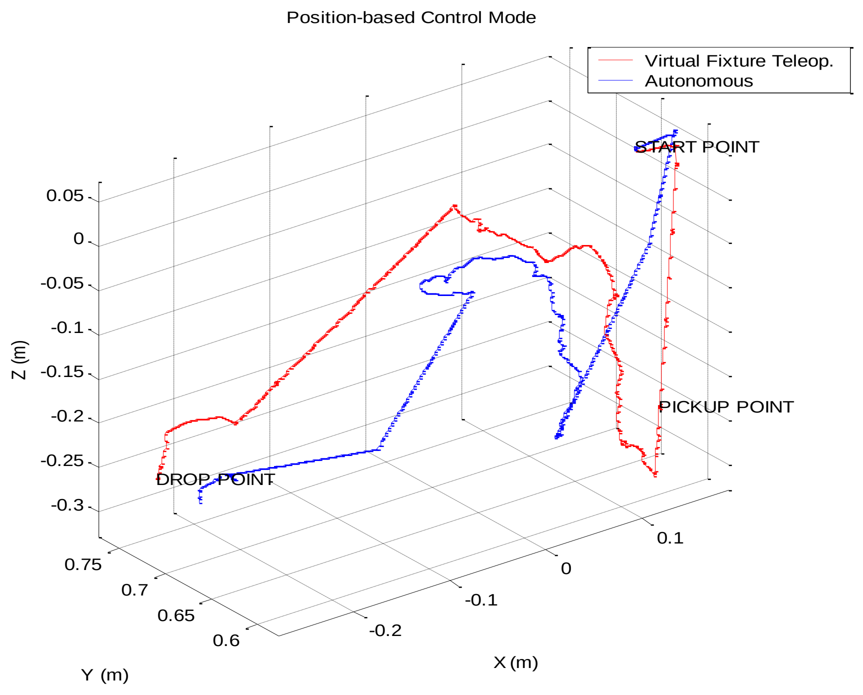


Figure 9.5 Position-Based Virtual Fixture Teleoperation vs. Autonomous Control

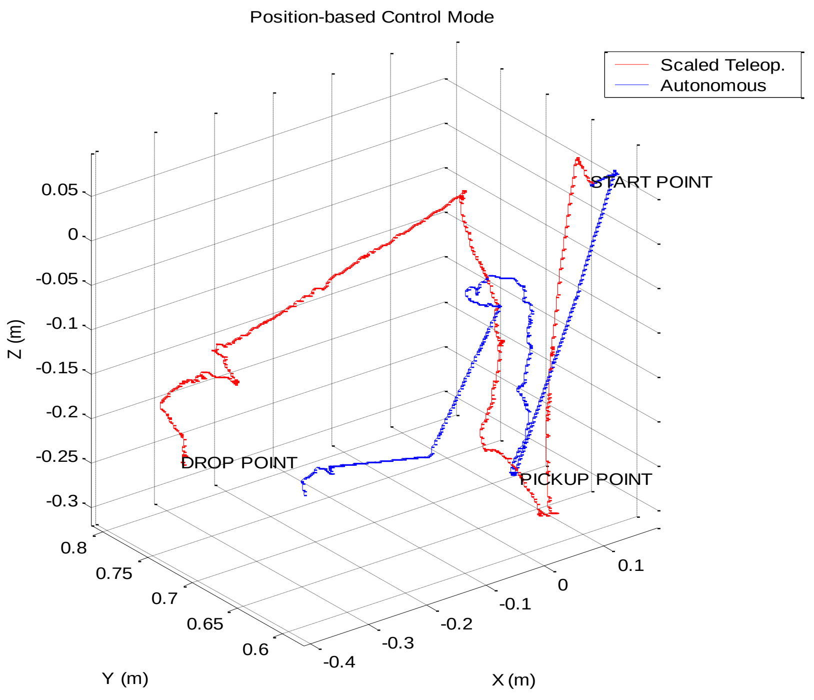


Figure 9.6 Position-Based Scaled Teleoperation vs. Autonomous Control

www.manaraa.com

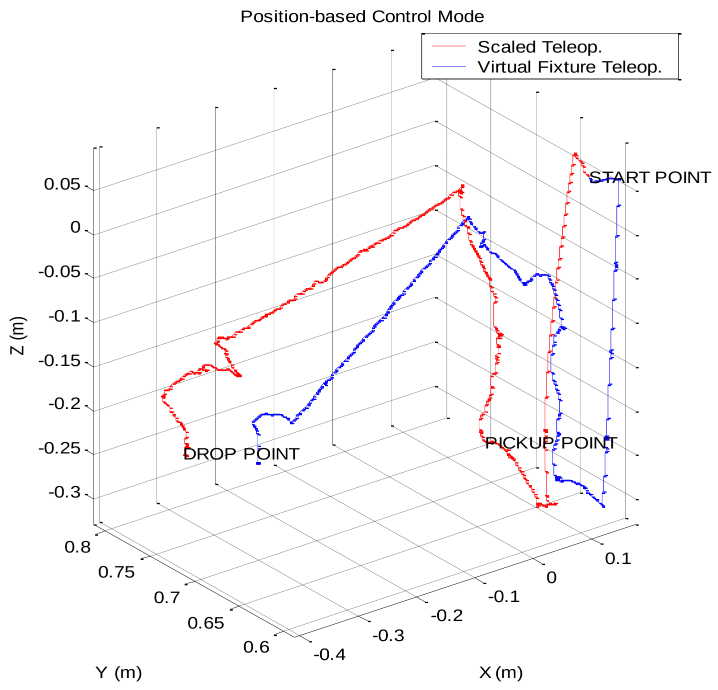


Figure 9.7 Position-Based Scaled Teleoperation vs. Virtual Fixture Teleoperation

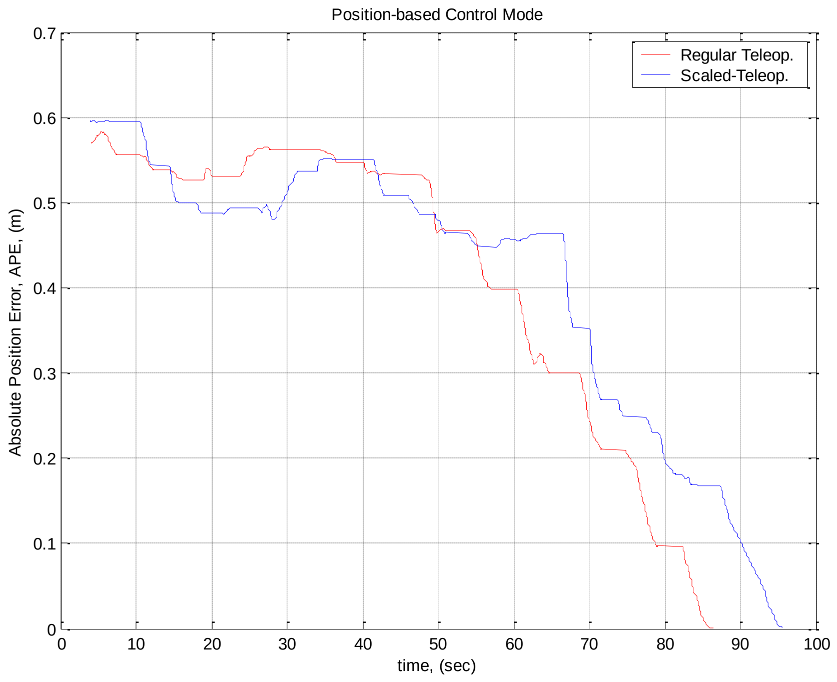


Figure 9.8 Absolute Position Error in Position-Based Regular vs. Scaled Teleoperation

المنارة للاستشارات
www.manaraa.com

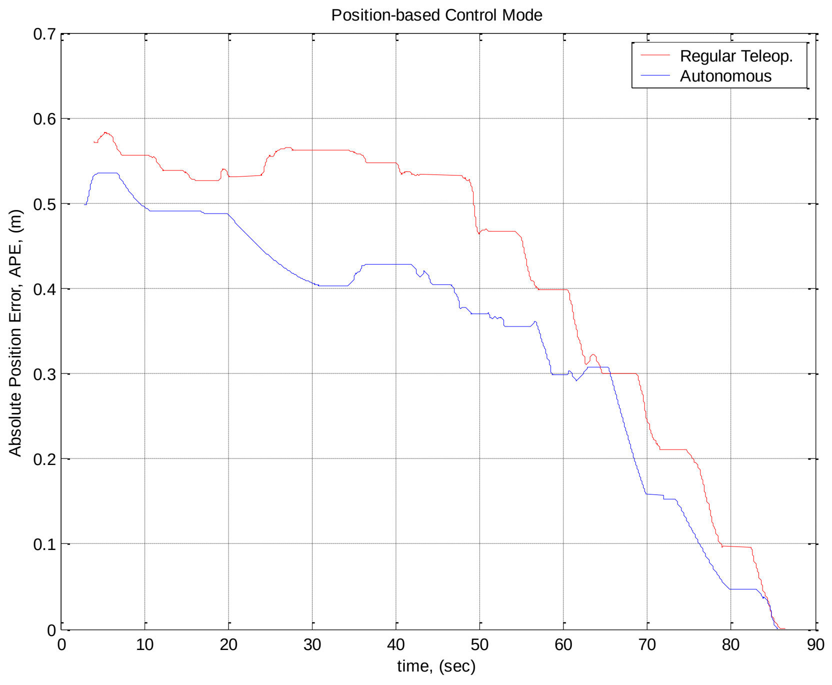


Figure 9.9 Absolute Position Error in Position-Based Regular Teleoperation vs. Autonomous Control

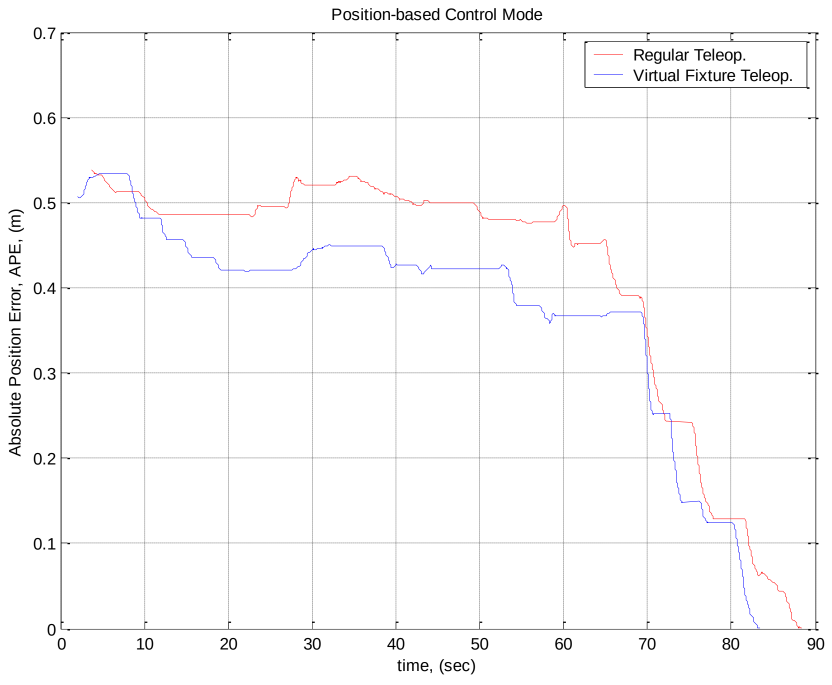


Figure 9.10 Absolute Position Error in Position-Based Regular vs. Virtual Fixture Teleoperation

المنارة للاستشارات
www.manaraa.com

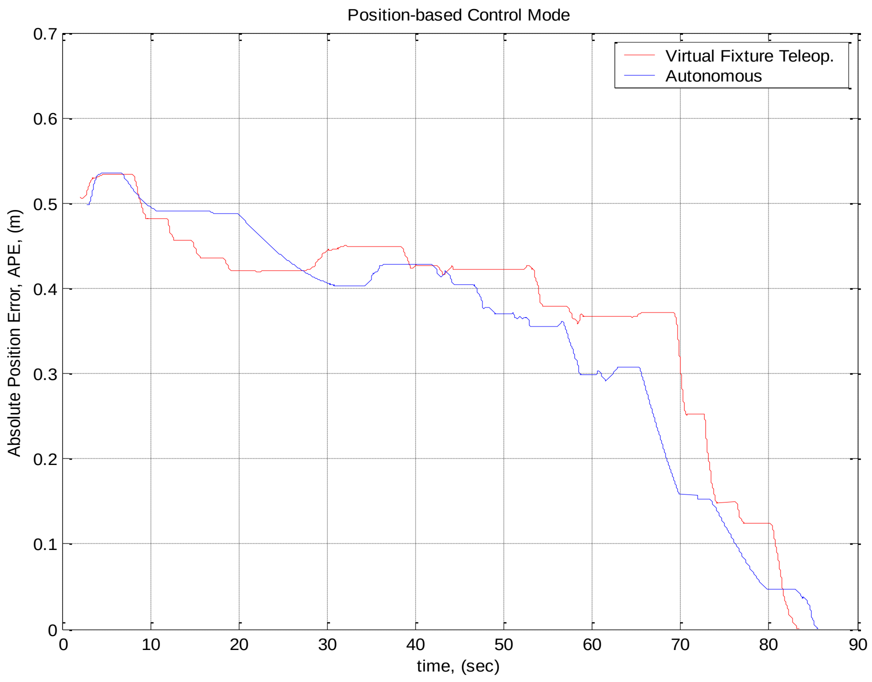


Figure 9.11 Absolute Position Error in Position-Based Virtual Fixture Teleoperation vs. Autonomous Control

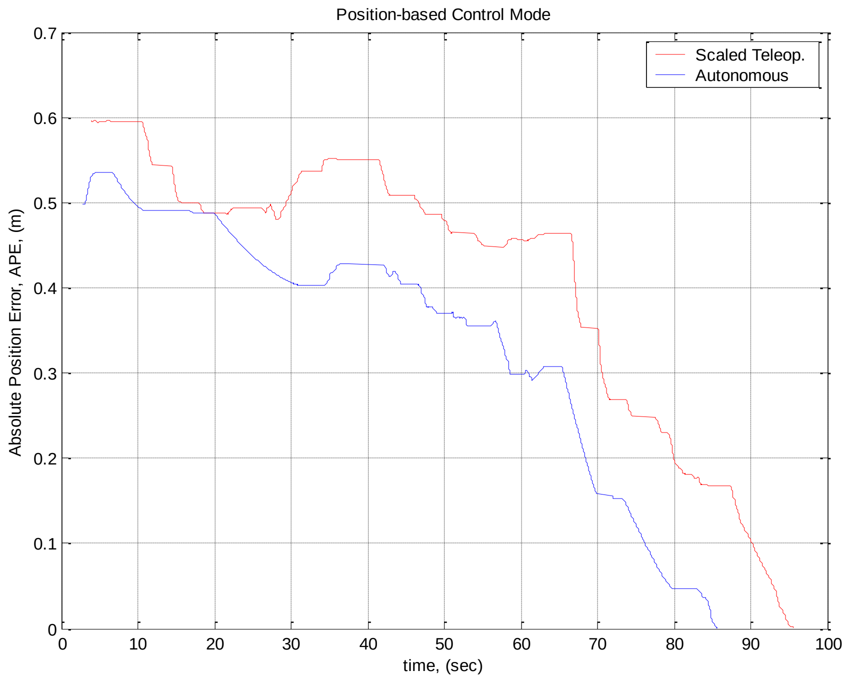


Figure 9.12 Absolute Position Error in Position-Based Scaled Teleoperation vs. Autonomous Control

www.manaraa.com

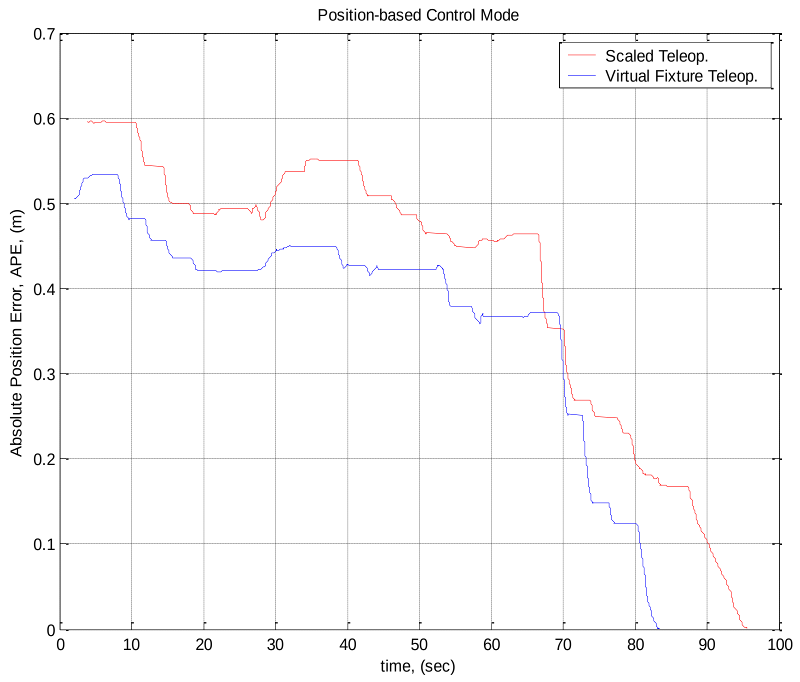


Figure 9.13 Absolute Position Error in Position-Based Scaled vs. Virtual Fixture Teleoperation

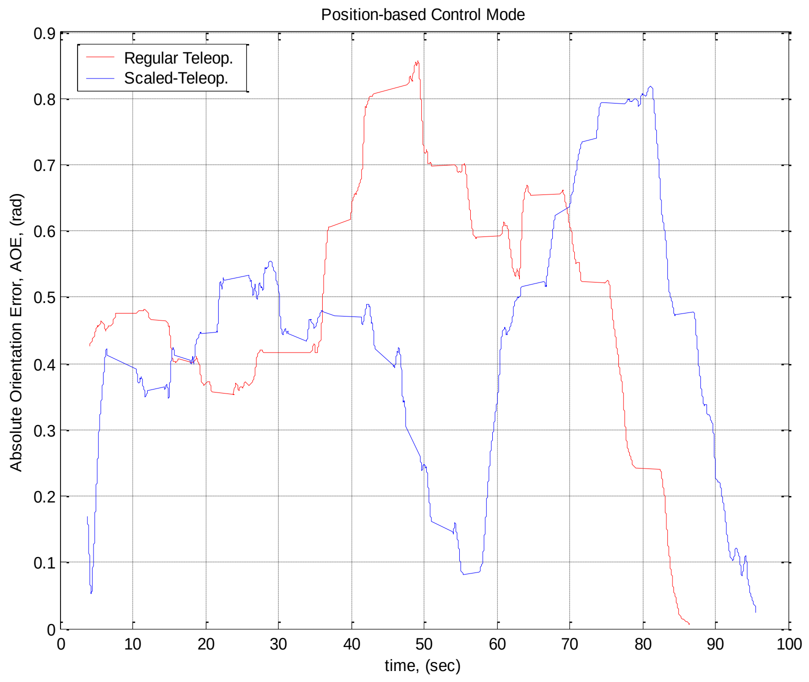


Figure 9.14 Absolute Orientation Error in Position-Based Regular vs. Scaled Teleoperation

www.manaraa.com

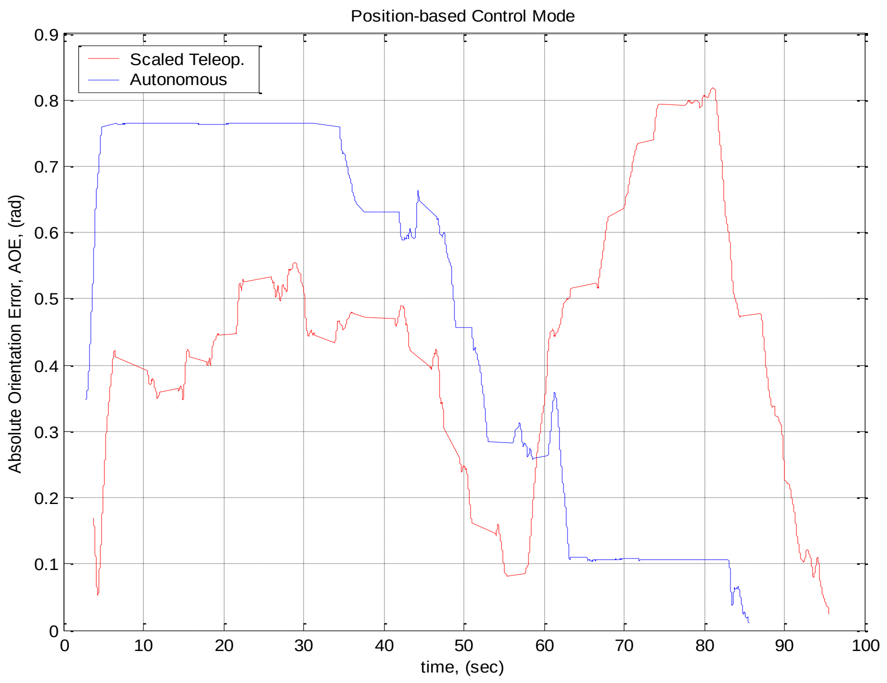


Figure 9.15 Absolute Orientation Error in Position-Based Scaled-Teleoperation vs. Autonomous Control

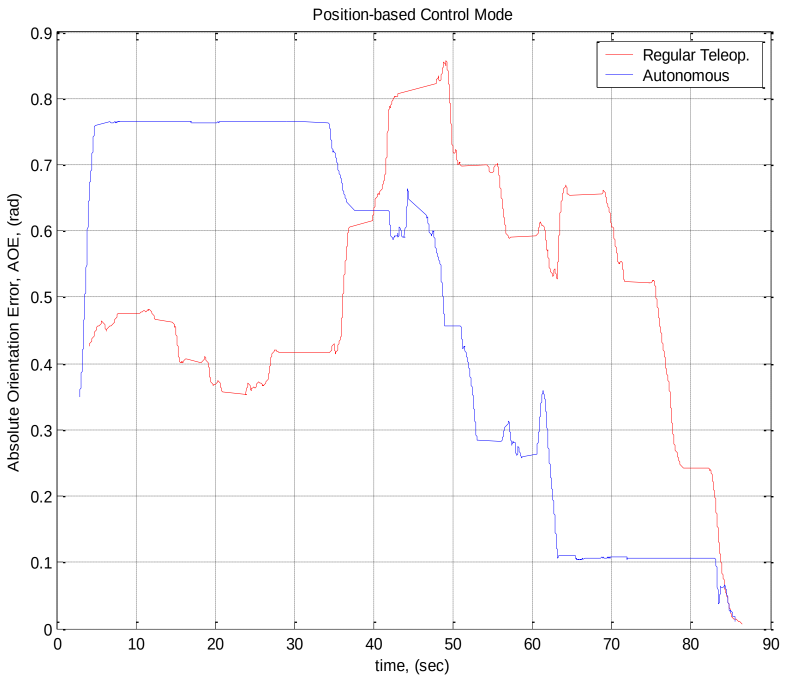


Figure 9.16 Absolute Orientation Error in Position-Based Regular Teleoperation vs. Autonomous Control

www.manaraa.com

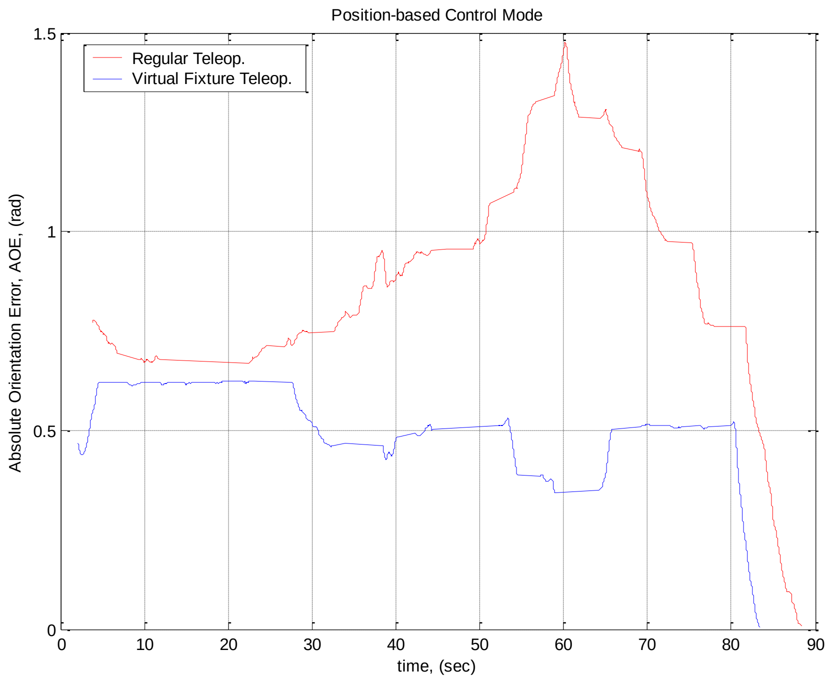


Figure 9.17 Absolute Orientation Error in Position-Based Regular vs. Virtual Fixture Teleoperation

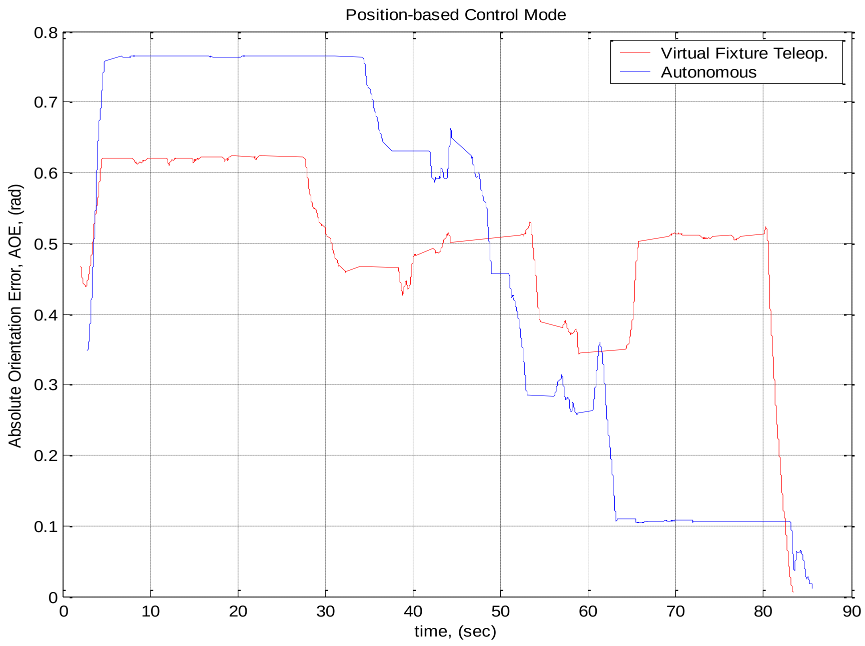


Figure 9.18 Absolute Orientation Error in Position-Based Virtual Fixture Teleoperation vs. Autonomous Control

المنارة للاستشارات
www.manaraa.com

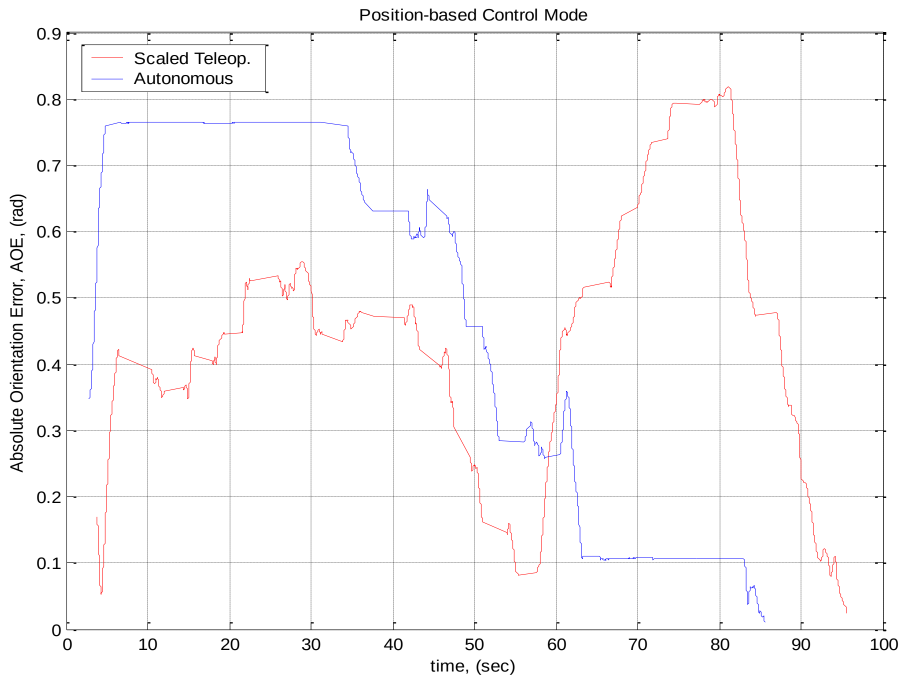


Figure 9.19 Absolute Orientation Error in Position-Based Scaled Teleoperation vs. Autonomous Control

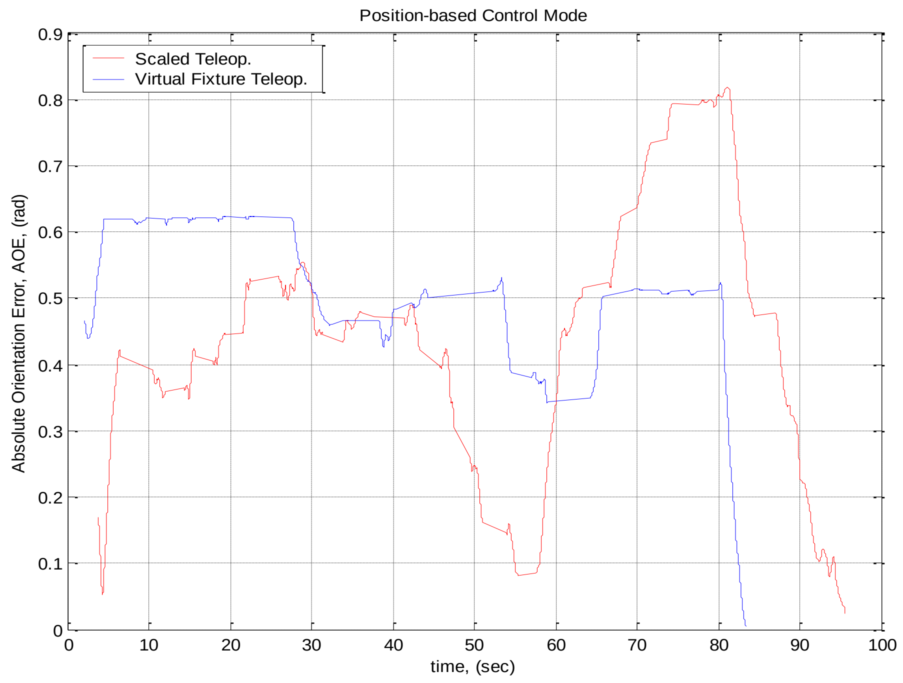


Figure 9.20 Absolute Orientation Error in Position-Based Scaled vs. Virtual Fixture Teleoperation

المنارة للاستشارات

www.manaraa.com

9.2.2 Velocity-Based Control Interactive Simulations Results

In this mode of teleoperation, the PUMA end-effector speed changes proportionally to the Phantom Omni changing position. The user will move the Omni's end-effector once to select a direction and speed for the PUMA end-effector. As discussed in section 4.2.3, the user holds the Phantom Omni's end-effector steady to fix the speed until the gripper is in the vicinity of the target object. Then, the user moves the Phantom Omni's end-effector back to its initial position for stopping close to the target.

The testing results for the Velocity-Based control simulations are very similar to those obtained for the Position-based control simulations. Figures 9.21 to 9.23 show that the trajectory in Regular teleoperation mode is not as smooth, fast and shorter as it is Autonomous control mode and Scaled, Virtual Fixture control modes. The trajectories in the Autonomous, Virtual Fixture and Scaled modes are similar (Figures 9.24 and 9.26). And comparing Autonomous to Scaled, the trajectory is shorter and smoother for the Autonomous mode (Figures 9.25). This is also the case for the Absolute Position Error (Figures 9.27 to 9.32).

As for the Absolute Orientation Error, for the Velocity-Based control, the errors in the Regular mode for the complete task are mostly smaller than for the Autonomous, Scaled and Virtual Fixture (Figures 9.33 to 9.35). Similarly, the orientation errors for the Virtual Fixture and Scaled modes are smaller than for the Autonomous (Figures 9.36 to 9.39). This can be explained by the condition imposed in the velocity control mode for which a particular Omni end-effector position does not have to remain mapped to a specific configuration of the slave, but only to the magnitude and direction of the slave of

www.manaraa.com

the end-effector velocity. This means that there is no need to precisely reorient the gripper for grasping when the velocity control mode is active.

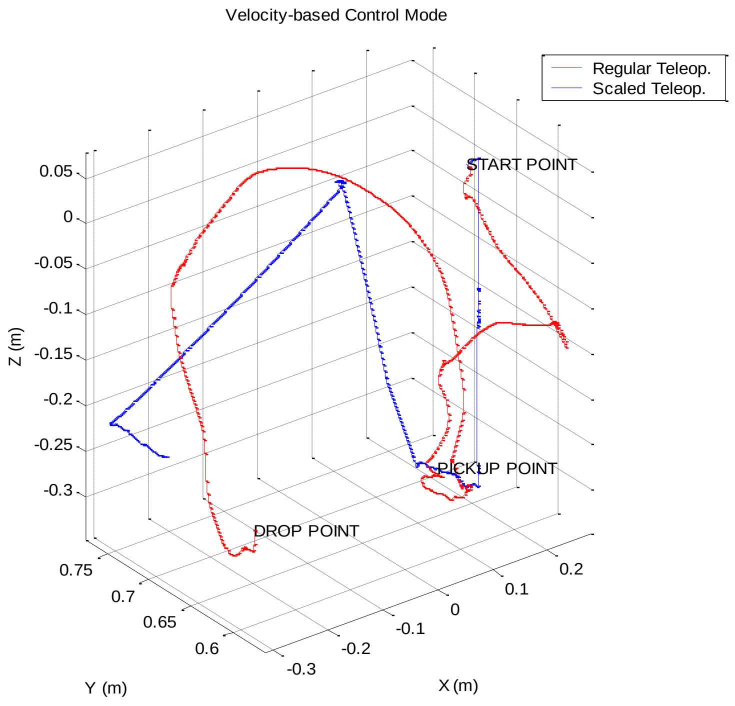


Figure 9.21 Velocity-Based Regular Teleoperation vs. Scaled Teleoperation

www.manaraa.com

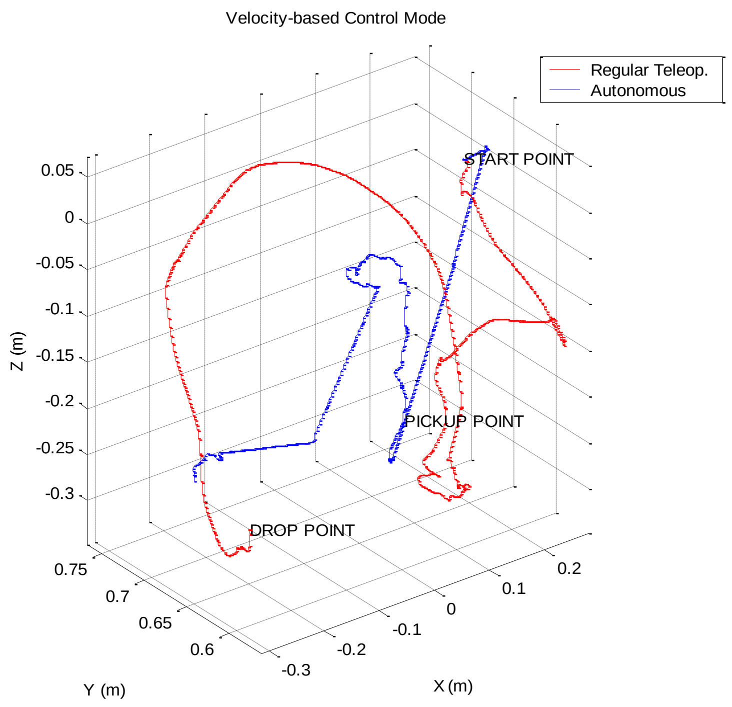


Figure 9.22 Velocity-Based Regular Teleoperation vs. Autonomous Control

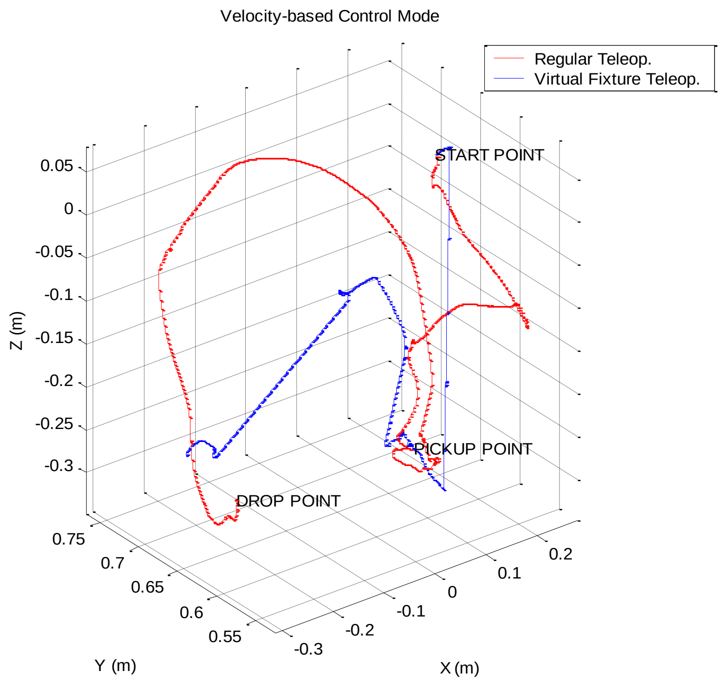


Figure 9.23 Velocity-Based Regular Teleoperation vs. Virtual Fixture Teleoperation

www.manaraa.com

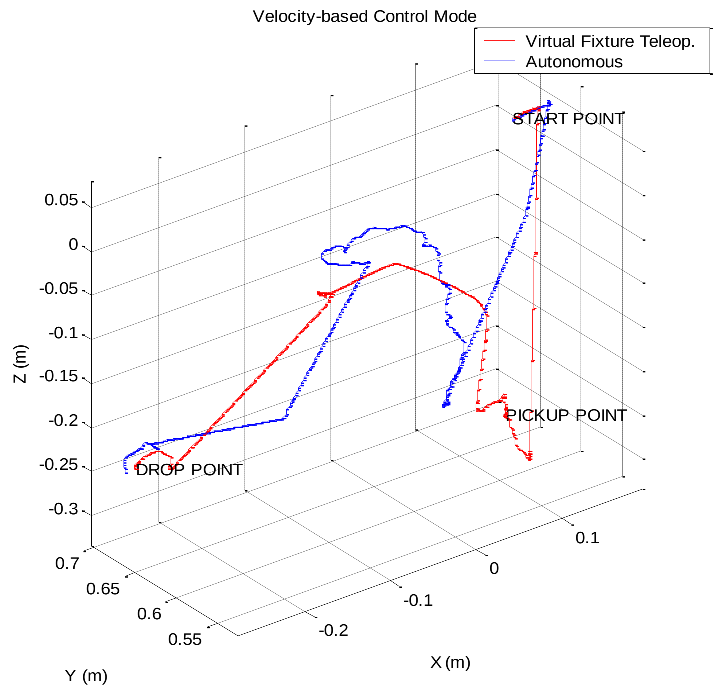


Figure 9.24 Velocity-Based Virtual Fixture Teleoperation vs. Autonomous Control

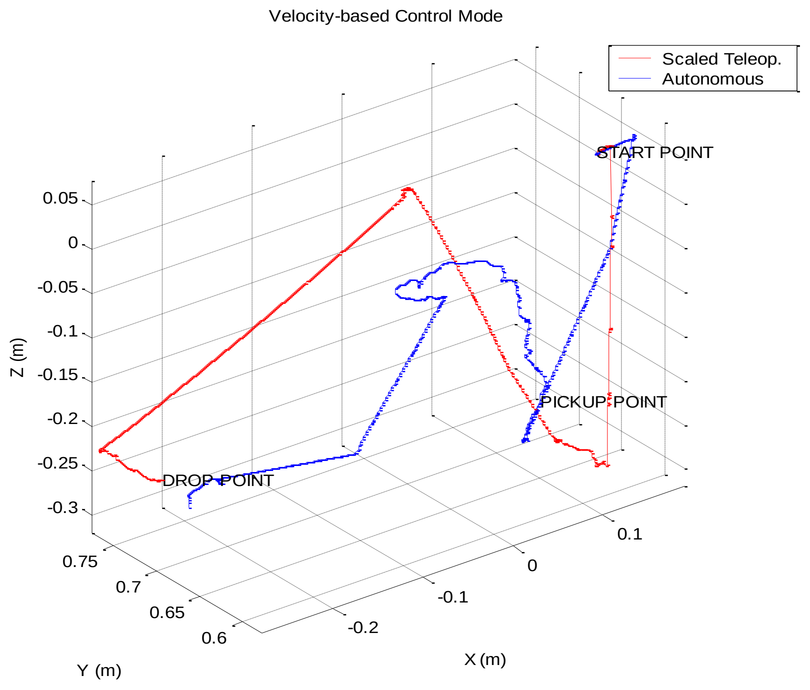


Figure 9.25 Velocity-Based Scaled Teleoperation vs. Autonomous Control

www.manaraa.com

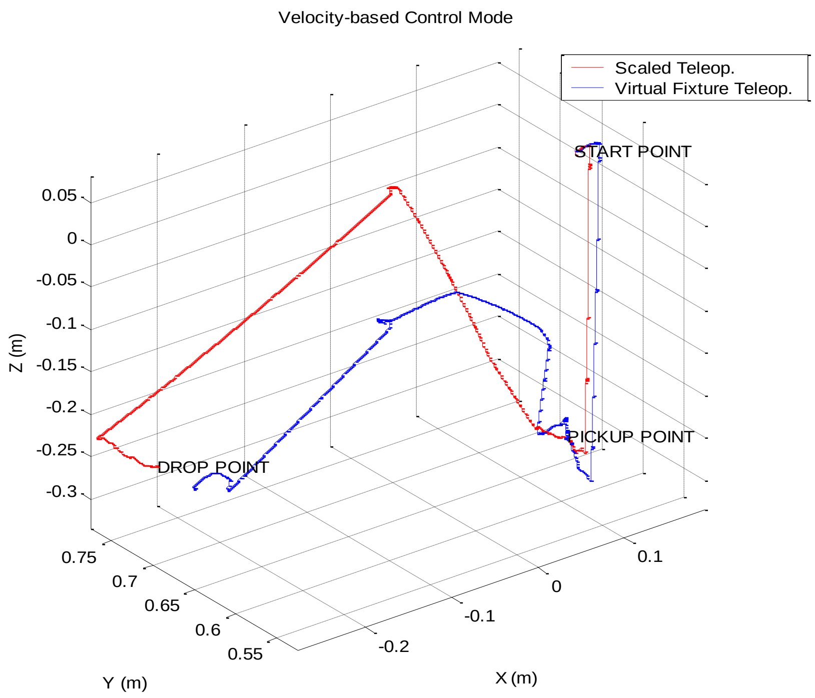


Figure 9.26 Velocity-Based Scaled Teleoperation vs. Virtual Fixture Teleoperation

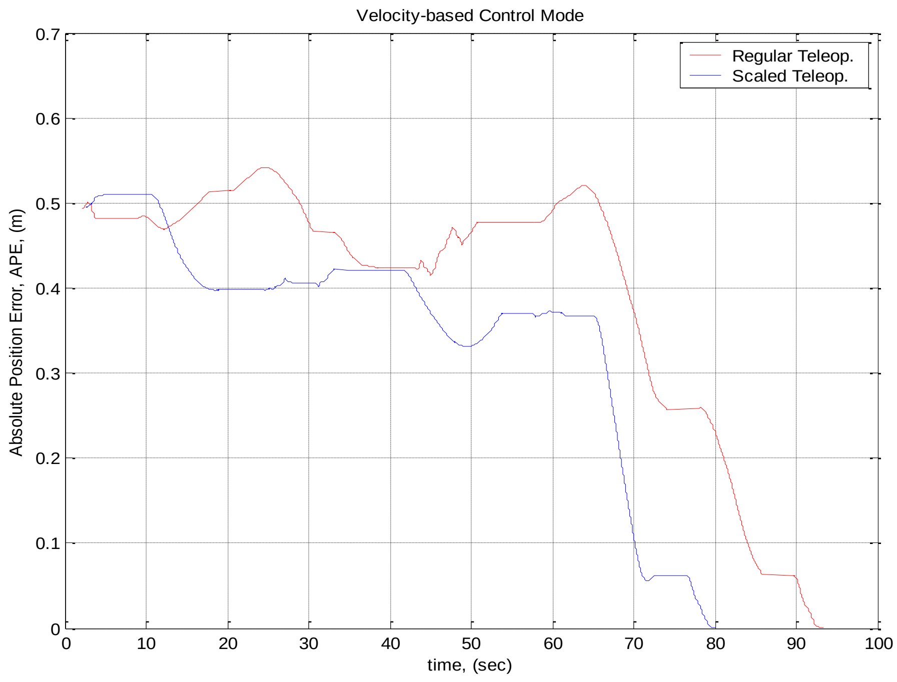


Figure 9.27 Absolute Position Error in Velocity-Based Regular vs. Scaled Teleoperation

www.manaraa.com

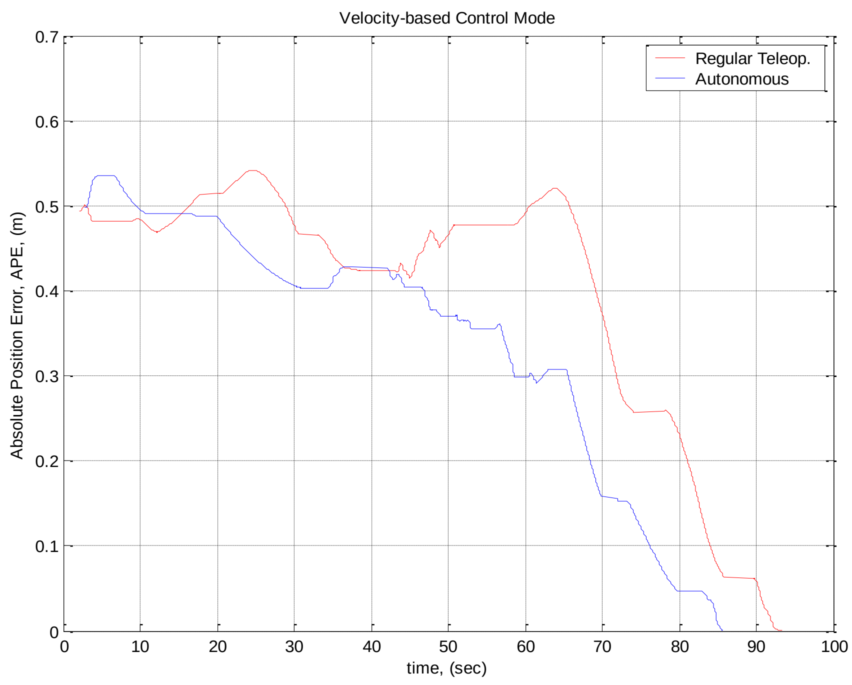


Figure 9.28 Absolute Position Error in Velocity-Based Regular Teleoperation vs. Autonomous Control

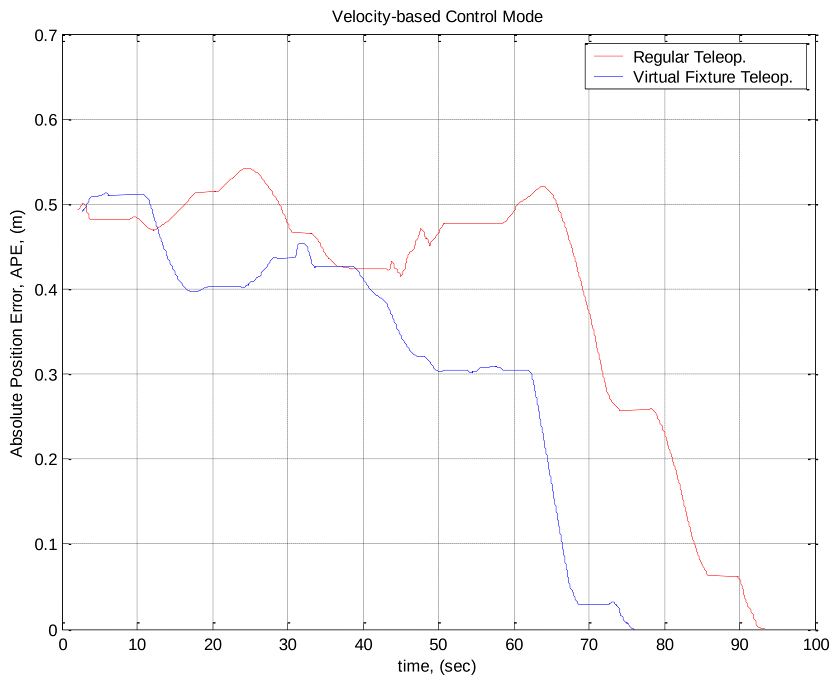


Figure 9.29 Absolute Position Error in Velocity-Based Regular vs. Virtual Fixture Teleoperation

المنارة للاستشارات
www.manaraa.com

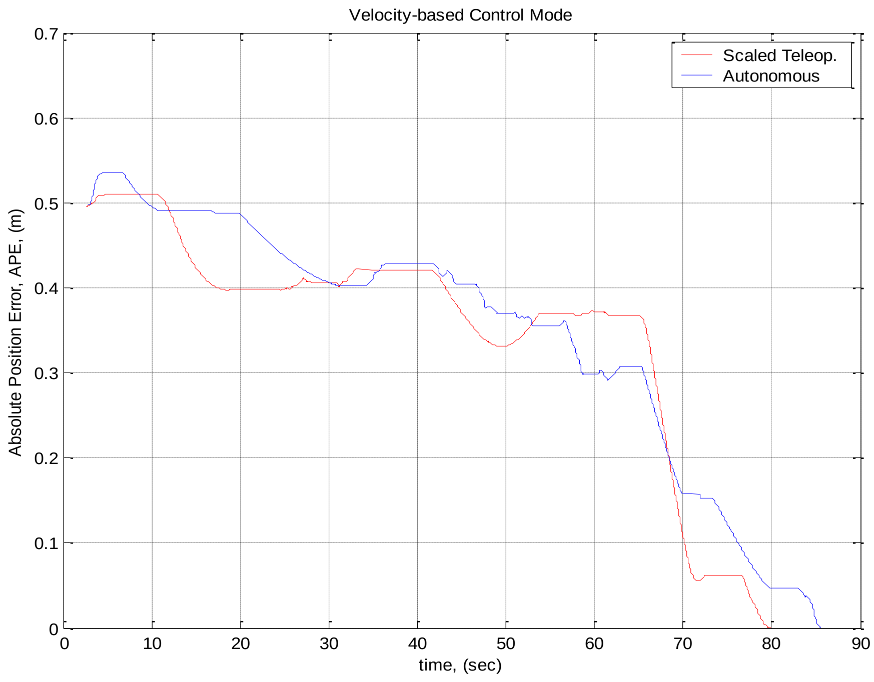


Figure 9.30 Absolute Position Error in Velocity-Based Virtual Fixture Teleoperation vs. Autonomous Control

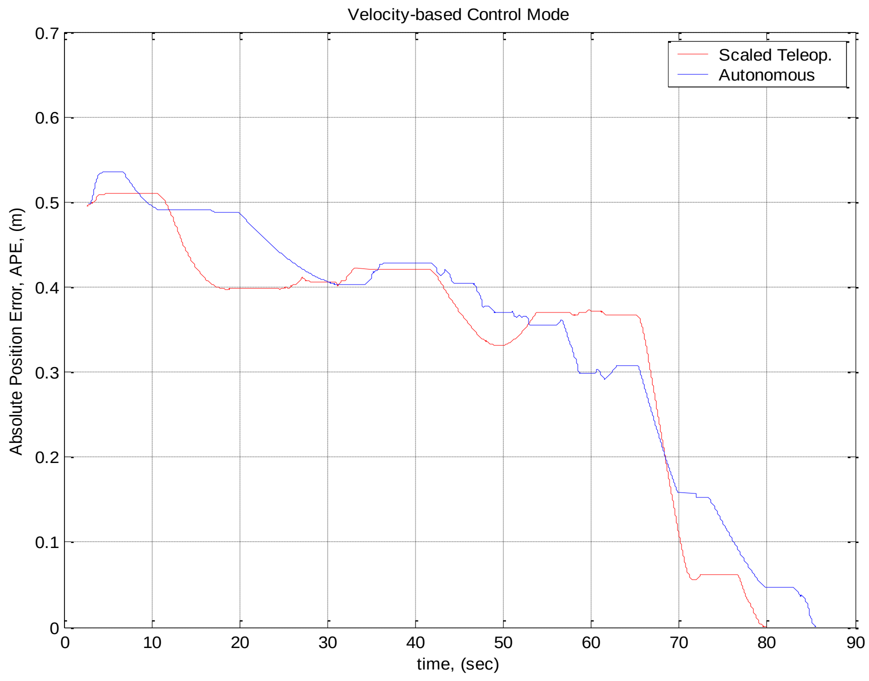


Figure 9.31 Absolute Position Error in Velocity-Based Scaled Teleoperation vs. Autonomous Control

المنارة للاستشارات
www.manaraa.com

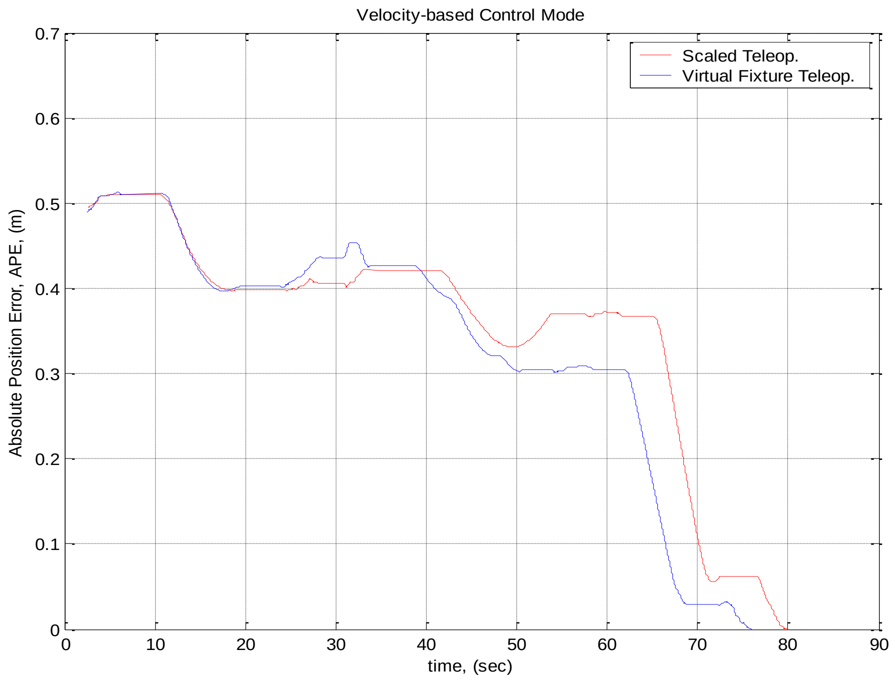


Figure 9.32 Absolute Position Error in Velocity-Based Scaled vs. Virtual Fixture Teleoperation

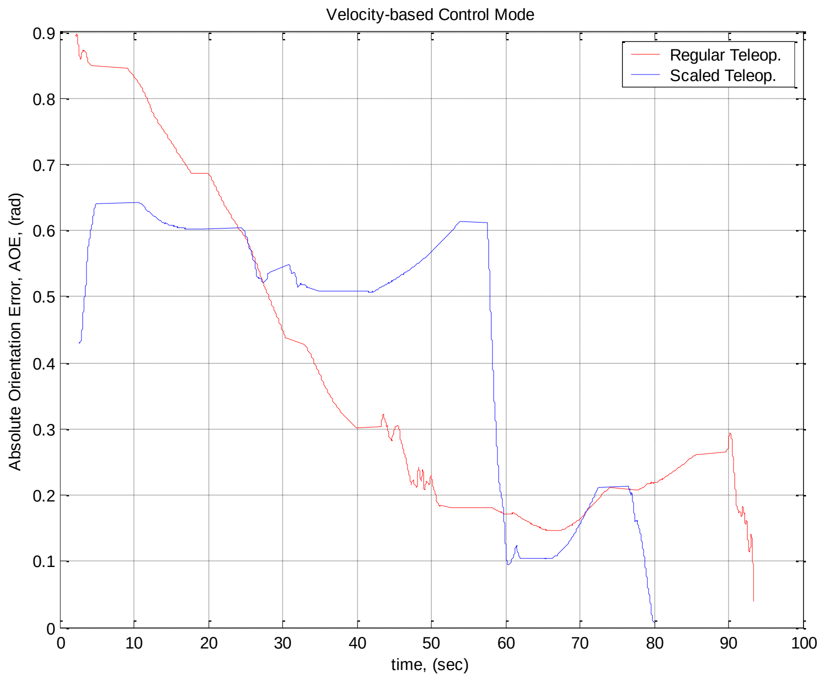


Figure 9.33 Absolute Orientation Error in Velocity-Based Regular vs. Scaled Teleoperation

www.manaraa.com

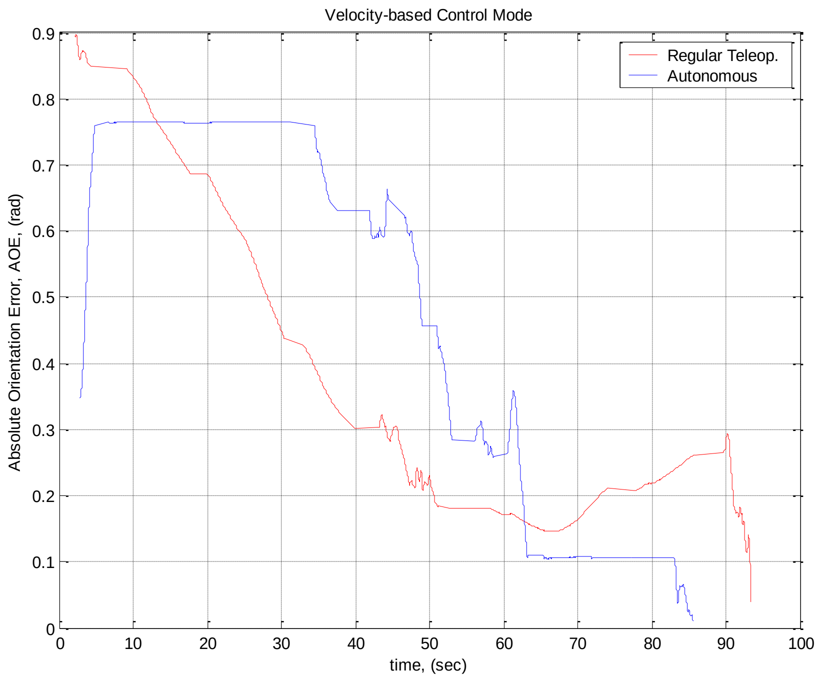


Figure 9.34 Absolute Orientation Error in Velocity-Based Scaled-Teleoperation vs. Autonomous Control

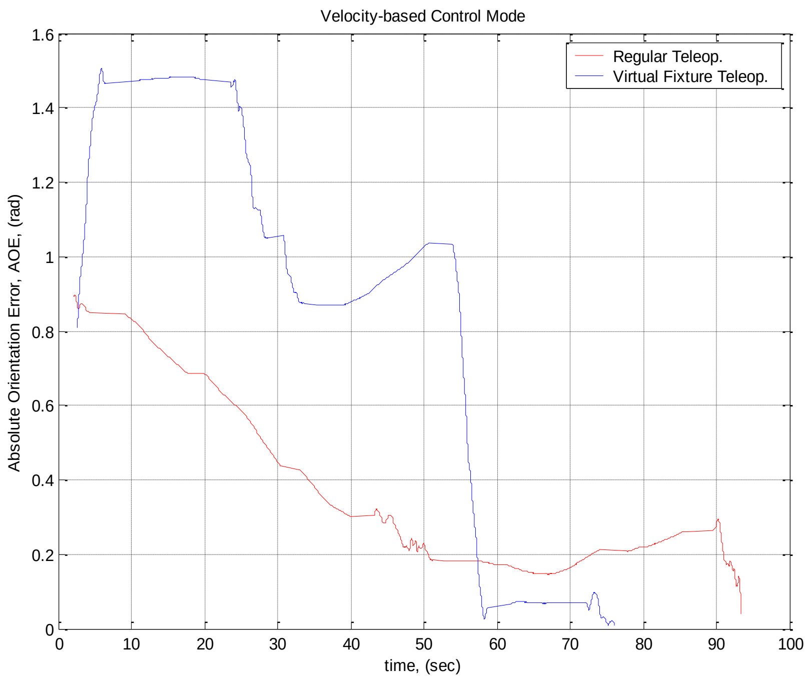


Figure 9.35 Absolute Orientation Error in Velocity-Based Regular Teleoperation vs. Autonomous Control

www.manaraa.com

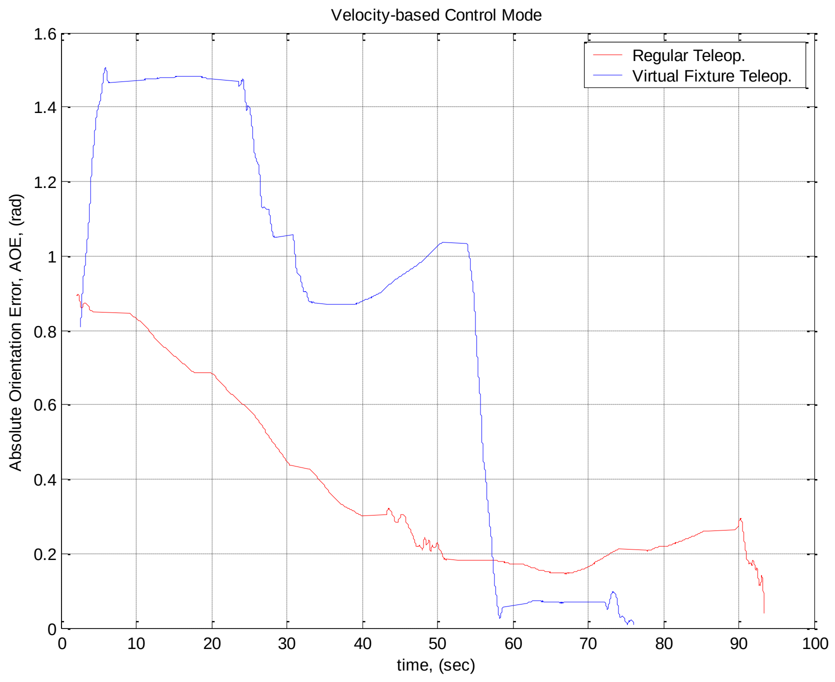


Figure 9.36 Absolute Orientation Error in Velocity-Based Regular vs. Virtual Fixture Teleoperation

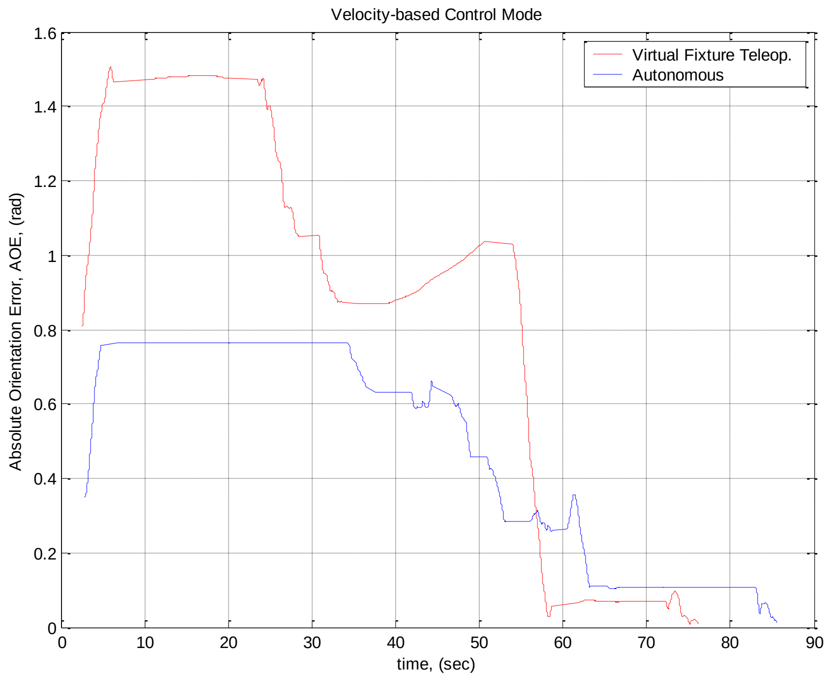


Figure 9.37 Absolute Orientation Error in Velocity-Based Virtual Fixture Teleoperation vs. Autonomous Control

www.manaraa.com

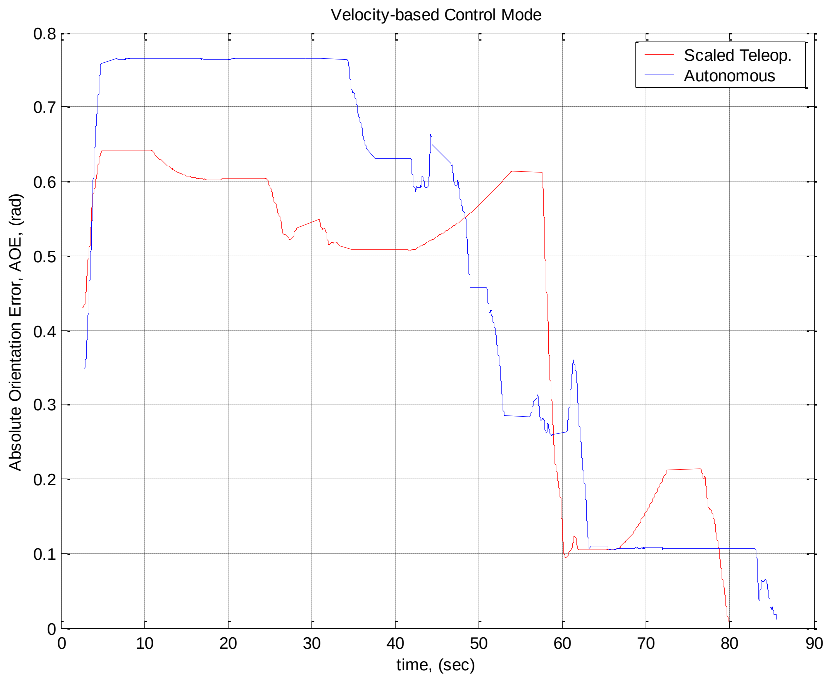


Figure 9.38 Absolute Orientation Error in Velocity-Based Scaled Teleoperation vs. Autonomous Control

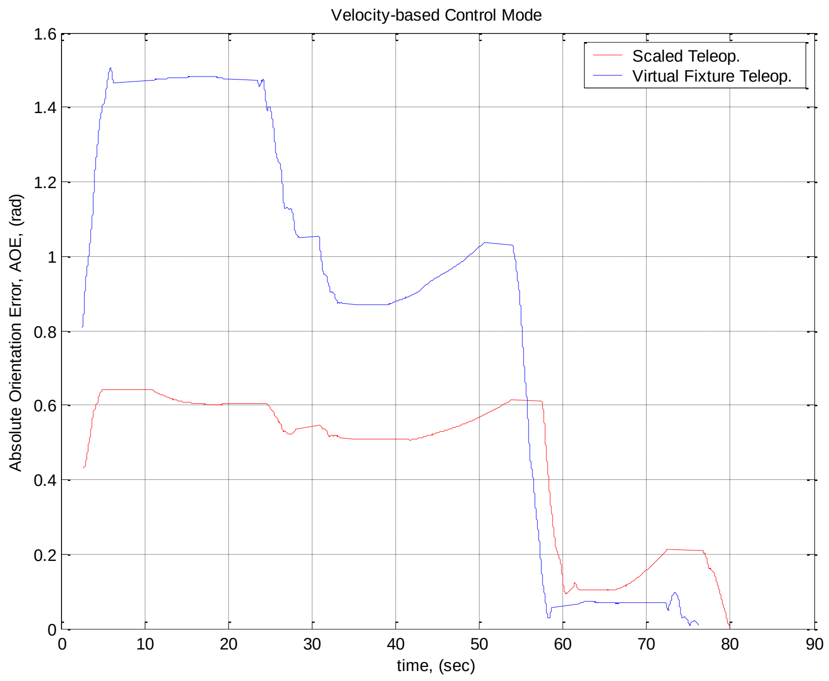


Figure 9.39 Absolute Orientation Error in Velocity-Based Scaled vs. Virtual Fixture Teleoperation

www.manaraa.com

The effectiveness of the assistive system during the execution of the pick-up-a-cup task presented in Figures 9.2 to 9.39 is summarized in Figures 9.40 to 9.45. The testing of force based virtual fixture is included in these figures as an additional parameter of comparison between the different modes of teleoperation.

A comparison of the APE and the AOE indicators for the force-based and position-based, regular (teleoperation without assistance) and scaled teleoperation modes is shown in Figures 9.40 and 9.41. The APE and AOE comparisons corresponding to Regular (Teleoperation without Assistance), Position-based Scaled Teleoperation (Motion-based Scaling), Position-based Virtual Fixture (Motion-based Virtual Fixture) and Force-based Virtual Fixture are depicted in Figures 9.42 and 9.43.

Figures 9.44 and 9.45 show the APE and AOE indicators for Autonomous, Velocity-Based Scaling, Velocity-Based Virtual Fixture and Force-based Virtual Fixture. As can be observed, Autonomous mode performs better than any other method, as shown in previous figures. The assistance provided in the form of scaled and virtual fixture is shown to be better than regular teleoperation (without assistance), as expected. The force-based virtual fixture is more effective in assisting the user's motion along the straight line trajectory when compared to motion-based virtual fixture.

المنارة للاستشارات
www.manaraa.com

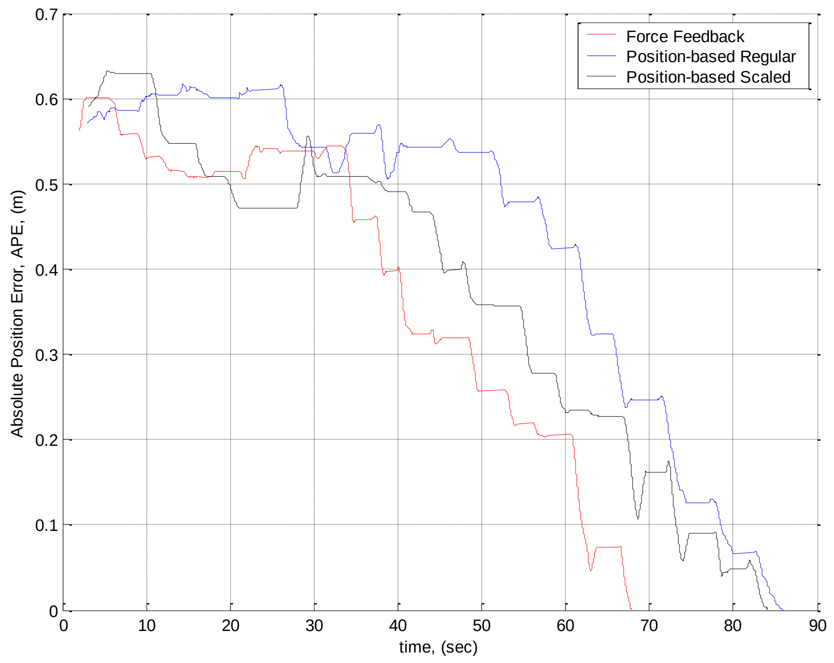


Figure 9.40 APE for Force, Position-Based Regular and Scaled Teleoperation

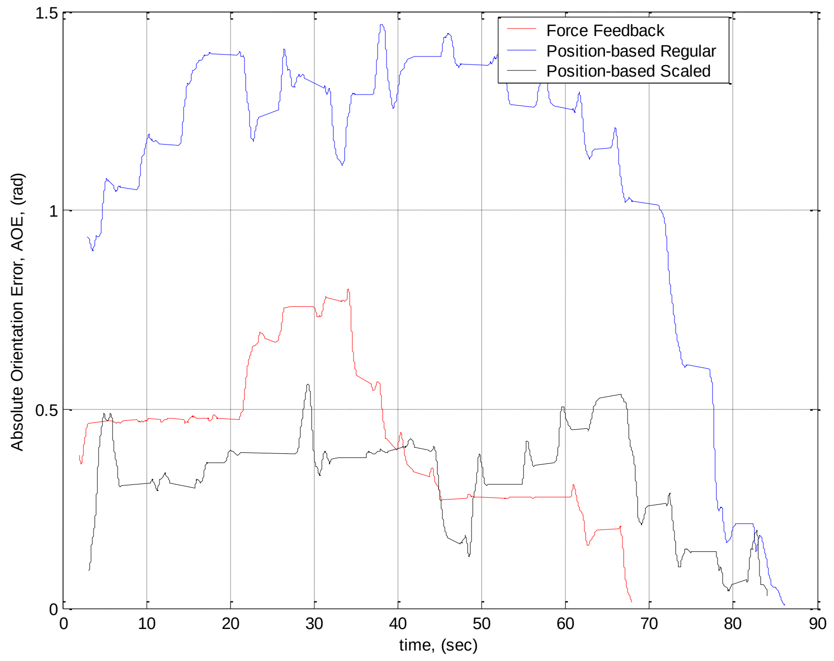


Figure 9.41 AOE for Force, Position-Based Regular and Scaled Teleoperation

www.manaraa.com

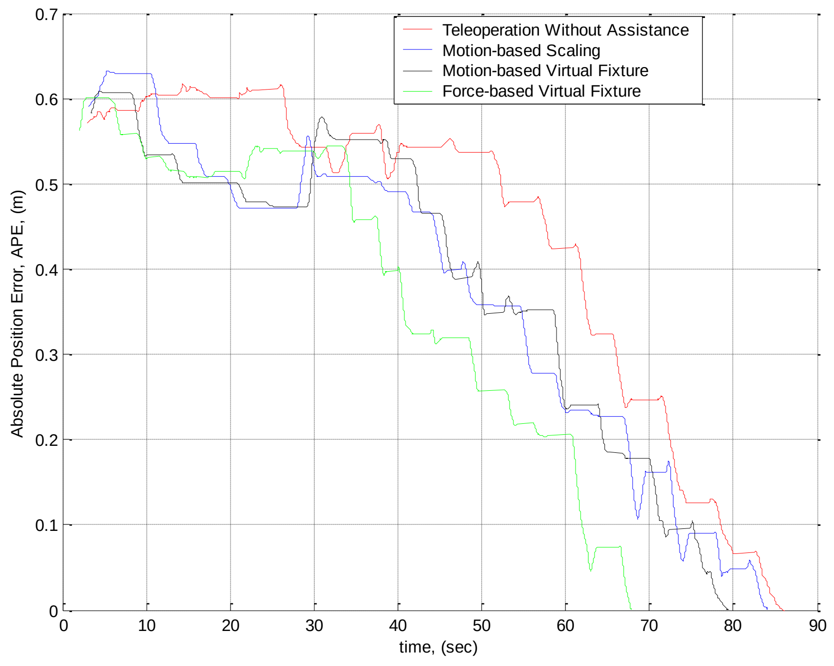


Figure 9.42 APE for Teleoperation without Assistance, Motion-based Scaling, Motion-based Virtual Fixture and Force-based Virtual Fixture

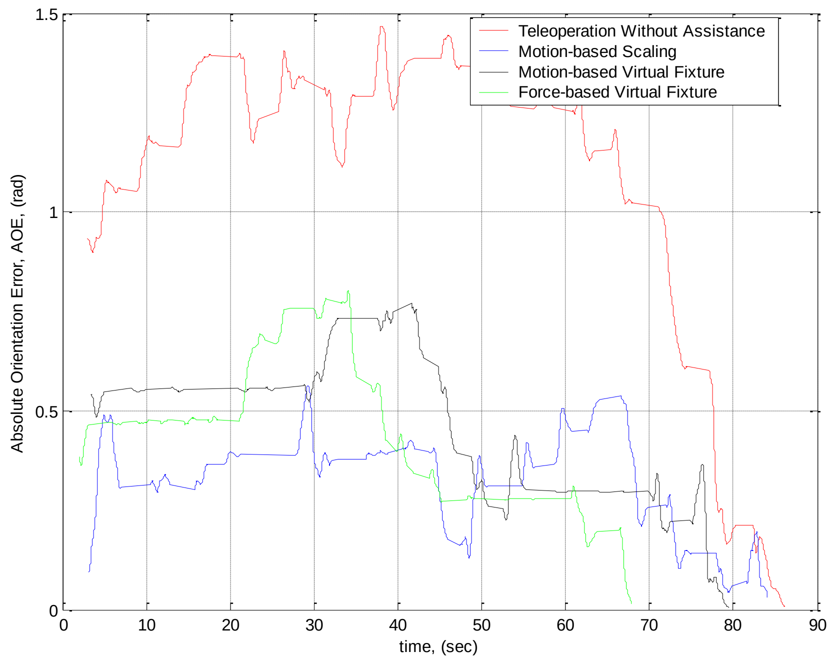


Figure 9.43 AOE for Teleoperation without Assistance, Motion-based Scaling, Motion-based Virtual Fixture and Force-based Virtual Fixture

المنارة للاستشارات
www.manaraa.com

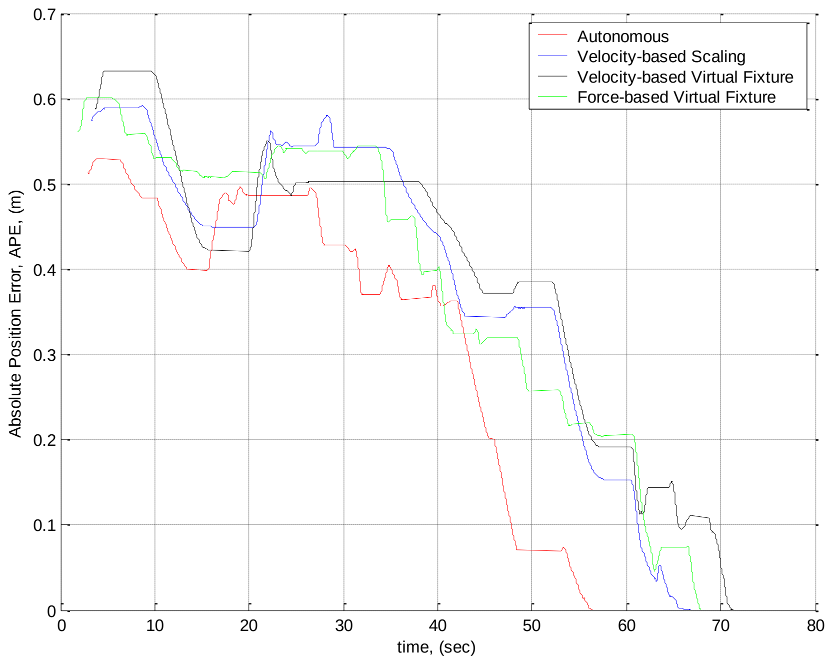


Figure 9.44 APE for Autonomous, Velocity-Based Scaling, Velocity-Based Virtual Fixture and Force-based Virtual Fixture

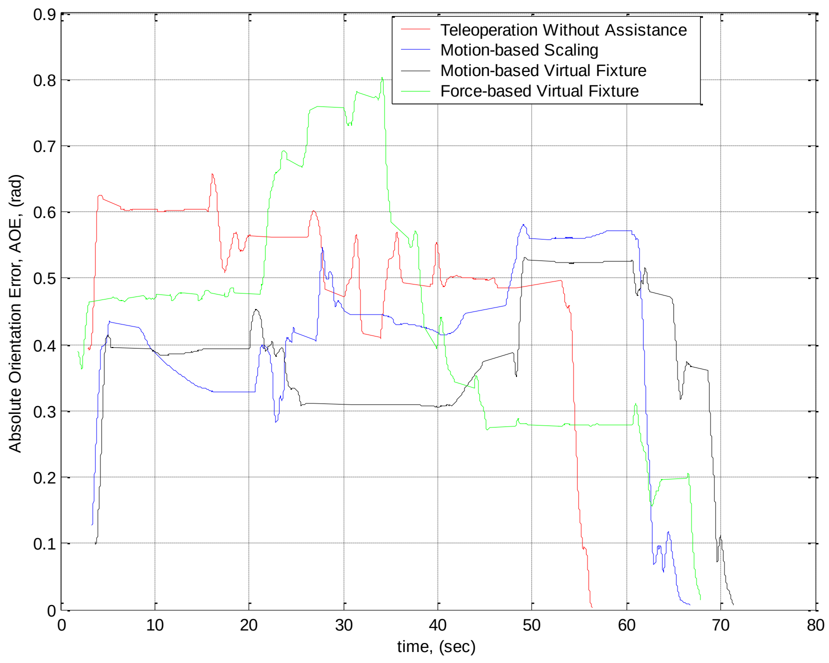


Figure 9.45 AOE for Autonomous, Velocity-Based Scaling, Velocity-Based Virtual Fixture and Force-based Virtual Fixture

المنارة للاستشارات
www.manaraa.com

9.3 Virtual Reality Simulation Results

The implemented multithreaded approach was also tested using Virtual Reality (VR) model of the PUMA manipulator. The software-based controller of the robot arm was interfaced to the real Phantom Omni hardware controller using the socket programming technique explained in section 7. Figure 9.40 shows the Cartesian coordinates of the PUMA's end-effector and the Phantom Omni's end-effector. As shown, the implemented multithreaded design allowed the execution of the telerobotic without event mismatch. However, the communication between the Phantom Omni hardware controller and the software-based controller was unstable and it stopped responding abruptly. The problem with that is the unpredictability and unreliability of the third-party MatLab socket API used to integrate the C++ implementation of . For the case shown in Figure 9.40, the PUMA's end-effector follows the position in Cartesian space are negligible, and for plotting purposes, an offset of 10.0 mm in each direction was introduced so that the traces are distinguishable from one another. This shows that the multi-threaded implementation allows the associated tasks for controlling the telerobotic system to be executed concurrently without delays, increasing the overall performance.

المنارة للاستشارات
www.manaraa.com

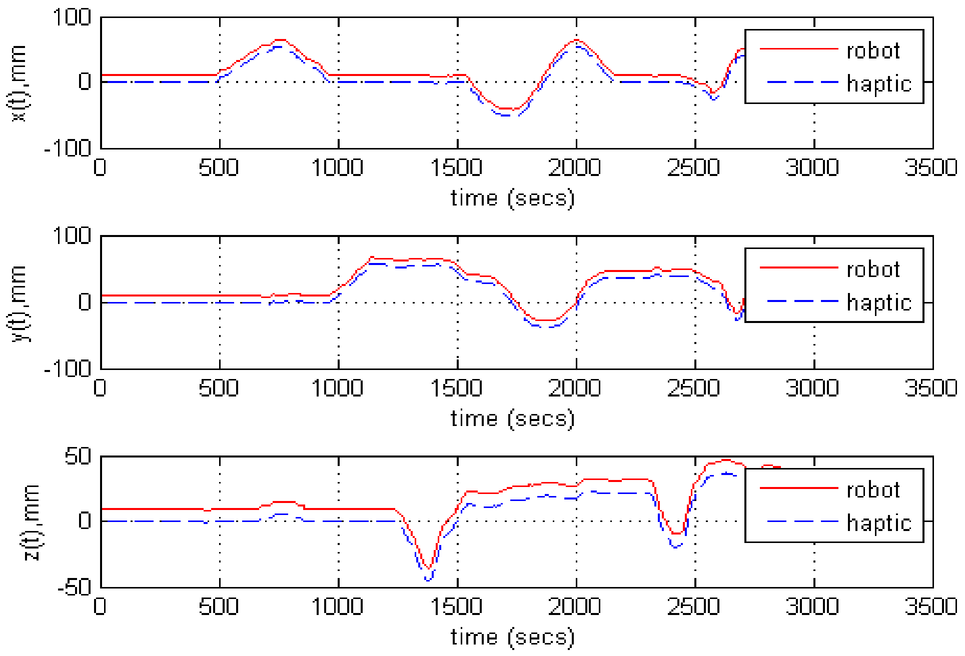


Figure 9.46 Position Results of Circular Path in Cartesian Space

Figures 9.47 and 9.48 illustrate the planar (X,Y) components of the trajectory using datasets from the circular path corresponding to the robot arm and the haptic plotted individually versus time in Figure 9.46.

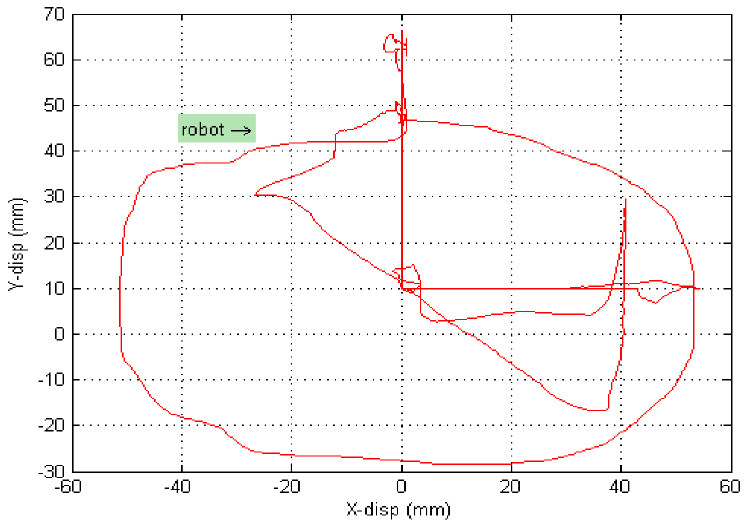


Figure 9.47 Robot Position Tracking of the Circular Path in the X-Y Plane.

المنارة للاستشارات
www.manaraa.com

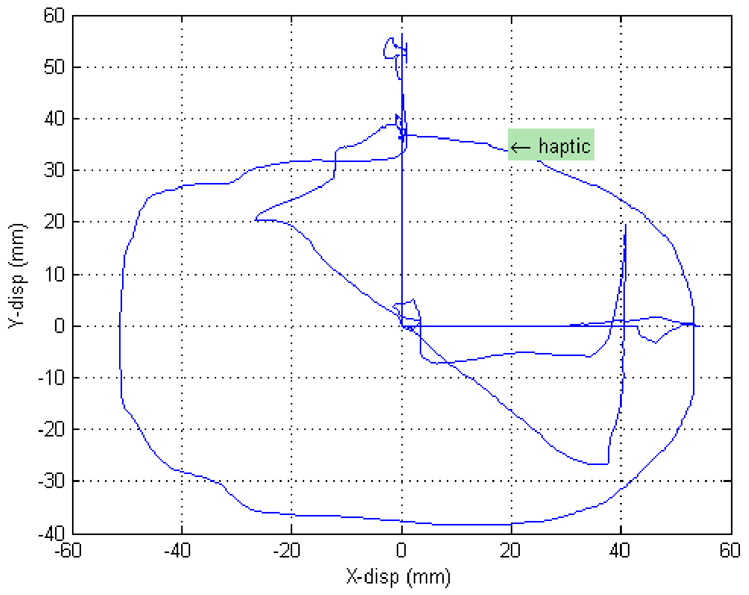


Figure 9.48 Haptic Position Tracking of the Circular Path in the X-Y Plane

For testing the sensor-based assist force (SAF's), the "haptic tip" was made to follow a linear trajectory generated between the Puma end-effector and a target. As mentioned previously, this trajectory is generated from the information gathered by the camera and the laser. The virtual environment that consisted of a simulated target and an end-effector along with a linear trajectory was available for the user to view on the PC that runs the Phantom Omni. A graph of forces that the user experiences while deviating from the trajectory versus time is shown in the Figure 9.49.

المنارة للاستشارات
www.manaraa.com

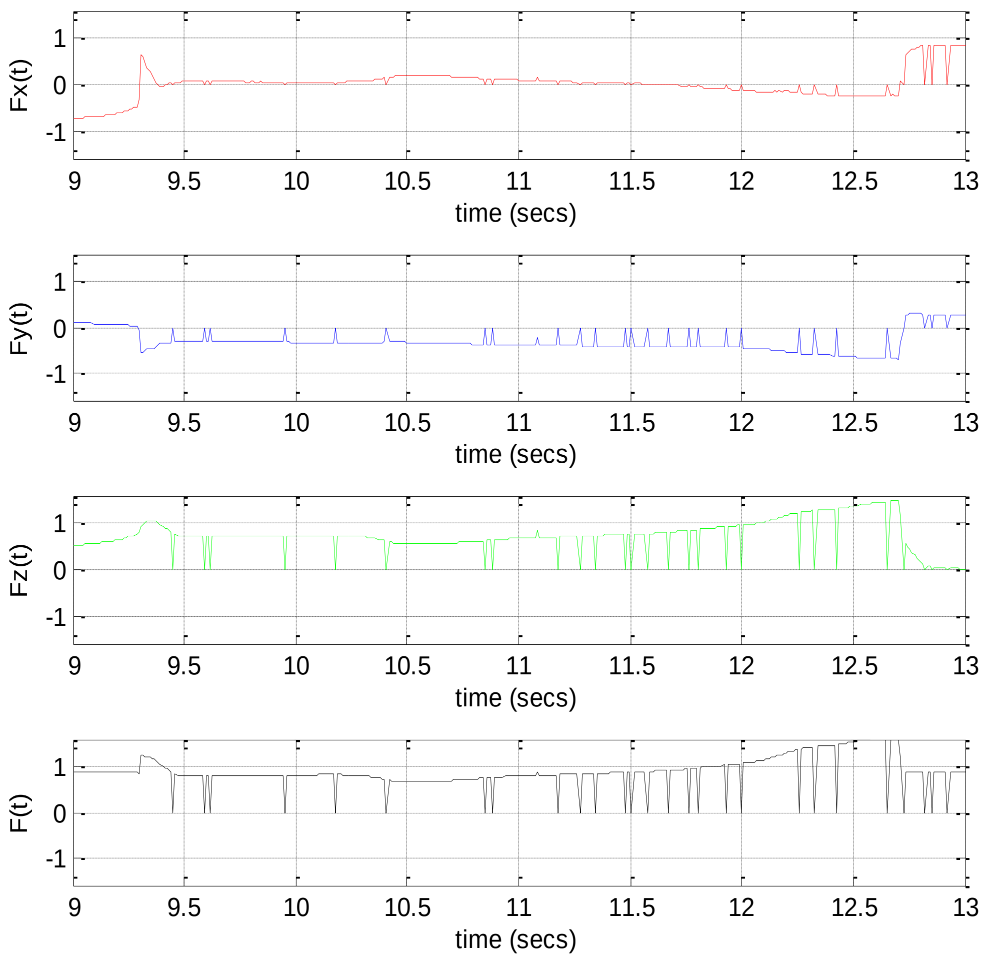


Figure 9.49 Typical Assistive Force Feedback Experienced by the User

It can be observed from this graph that the user begins to deviate from the target at the 12.0 second mark. As this happens the feedback forces increase trying to put the user back on the trajectory. At around 12.7 second mark the user experiences the maximum force as the user has deviated maximum from the trajectory. This way the user is given force assistance to move along the trajectory. It should be also noted that the user experiences the forces only if the user is at a certain radius near the trajectory. The user experiences maximum forces at the outer periphery of the circle defined by the radius and fails to experience any forces once the user leaves the periphery. The response of the

المنارة للاستشارات
www.manaraa.com

system is real time i.e., the user experiences the forces as soon as the user tries to move away from the trajectory. This real time response has been possible because of the multithreading strategies described previously. Using traditional signal processing techniques, it was found that the short period deviations ("spikes") shown in Figure 9.50 correspond to frequencies between 5.0 to 10.0 Hz. This figure also shows a simple moving average filter used to remove those "noisy" signals. A second order Butterworth filter was also implemented for this purpose with acceptable results which are not included in this document.

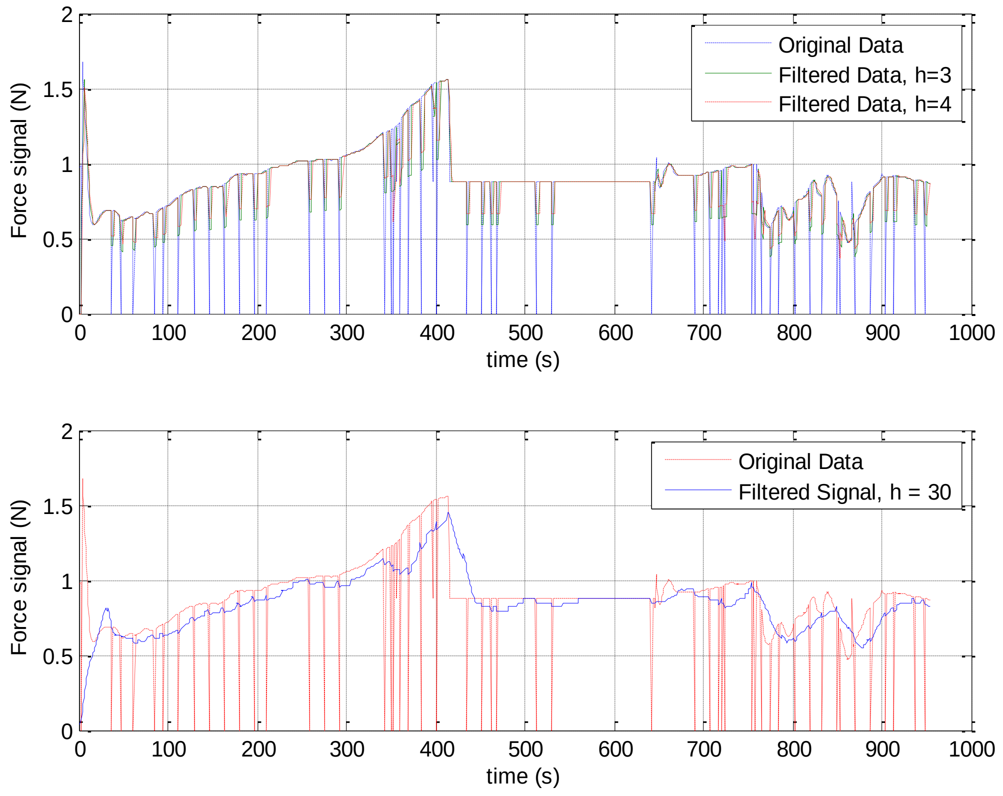


Figure 9.50 Typical Results of the Moving Average Filter Implementation

المنارة للاستشارات
www.manaraa.com

9.4 Summary

The results of the interactive or physical simulations for the pick-up-a cup task were presented and the performances for autonomous control mode, force and motion-based virtual fixtures, and scaled teleoperation modes of assistance were compared. The performance measures as shown in Figures 9.2 to 9.20 clearly indicate that the autonomous, scaled and virtual fixture teleoperation modes enable appropriate assistance to guide the user’s motion during the execution of the pick-up-a-cup task. The experiments conducted to validate the control strategies with the actual hardware show that the errors in both position and orientation are acceptable. The results of the experiments with the Puma 560, the Phantom Omni and the sensory suite (camera and a laser range finder) for trajectory tracking as well as the force assistance for guiding the user's motion were satisfactory. It was found that the variability shown by the boxplot indicates that the completion time is not a sufficient parameter for comparison of the autonomous and teleoperation modes. The performance measures also indicate that the real-time performance of robotic system provides adequate assistance for trajectory tracking, the manipulation of objects and completion of the pick-up-a-cup task. The experiments conducted to validate the control strategies with the actual hardware show that the errors in both position and orientation are acceptable. The results of the experiments with the PUMA 560, the Phantom Omni and the sensory suite (camera and a laser range finder) for trajectory tracking for guiding the user's motion were satisfactory. It is shown that the system provides the sensor-based assistance to guide the user’s motion.

المنارة للاستشارات
www.manaraa.com

Chapter 10

Conclusions and Recommendations

10.1 Overview

A PC-based multithreaded, hard real-time controller for a sensor-assisted telerobotic system was developed. The implemented assistive force feedback system used simple sensors such as a laser range finder to guide the user's motion and a CCD camera for visual feedback. The user gets visual as well as haptic feedback on the remote PC that has Phantom Omni as the master. It was shown that the force feedback provided by the telerobotic controller and the sensors is consistent and in real-time, even though the computational resources used for the implementation were purposely limited to support a wide range of users. In order to coordinate the parallel execution of the telerobotic tasks to run in real-time, a multithreaded architecture was developed. This approach allowed the telerobotic control of the arm, sensory integration, and the computations of the different forms of assistance without incurring in high costs, increased complexity and scalability problems associated with multiprocessor workstation systems.

The control strategy described in this dissertation used sensory signals for regular, scaled and virtual fixtures using position based and velocity based control, autonomous mode, and force-based virtual fixture teleoperation during user interactions. The user was enabled to switch between autonomous control mode, force and motion-based virtual

www.manaraa.com

fixtures, and scaled teleoperation modes. Several experiments were conducted to validate the trajectory following capabilities of the telerobotic system as well as the sensor-based assistance to guide the user's motion. A virtual environment for object manipulation was provided to the user in the form of a virtual cube, and a sphere was displayed as a visual cue of the position and orientation of the tip of the haptic device. In addition to the virtual environment, three (3) graphical views presented the sensory information to the user for enhanced visual perception of the object's location relative to the end-effector of the robot manipulator.

A testbed was created for conducting both simulated and physical experiments. The simulation was developed using a virtual reality model of the Puma 560 arm in MatLab and the Virtual Reality Toolbox. The C++ programming software was developed to interface the Phantom Omni software and the virtual reality simulations. For the physical experiments, the Phantom Omni Haptic device from SensAble Technologies is used as the master. It runs on a Pentium computer, with 1GHz single processing unit. The Phantom Omni device uses the OpenHaptics software which runs on Windows XP OS. The robot arm was equipped with a CCD camera and a Sick DT60 laser range finder. A Pentium II-666 MHz single processor computer was used to run the QNX Real-time Operating System. The Puma 560 software-based control strategy is a form of a PD plus gravitational compensation controller. The testing procedures of the supervisory control scheme included circular, polynomial, Bezier curves, and linear trajectories with force feedback along the Cartesian axes (X, Y, Z) as the user deviates from any of those trajectories. During those interactions, the virtual environment described previously as well as the camera views were displayed simultaneously on the

www.manaraa.com

screen for visualization of the telerobotic environment. The control system architecture designed to satisfy the real-time constraint consists of the following main threads:

1. The determination of the target position and orientation with respect to the Puma end-effector (in joint or Cartesian space) and mapping this position and orientation to the Phantom Omni tip.
2. A trajectory generation thread which computes intermediate points of the trajectory to reach the target.
3. The computation of the joint angles of the PUMA for trajectory-following using inverse kinematics based on the resolved-rate algorithm.
4. The computation of the torques using a proportional-derivative (PD) controller with gravity compensation which was required to drive the motors in the PUMA.
5. The sensor information from the camera and the laser was fused to determine the position and orientation of the target with respect to the PUMA's end-effector and this data was sent to the Phantom Omni for further processing.
6. The communication thread handles the position and orientation information of the Phantom Omni's end-effector. This information was used by the PUMA software controller for position-based and velocity-based teleoperation modes.

Also the processor that handles the Phantom Omni device has the following threads:

1 The graphics thread: It renders a virtual target, end-effector position and a trajectory on the user screen that is similar to the PUMA environment at a refresh rate that conforms to the PUMA and Phantom end-effector movement.

المنارة للاستشارات
www.manaraa.com

2 The haptic thread: This thread computes the feedback forces based on the sensory information about the trajectory of the PUMA and the users' movement of the Phantom Omni. As the user deviates from the trajectory, the assistive forces required to bring the user back on the trajectory were calculated and delivered to the user using the OpenHaptics software and the actuators of the Phantom Omni interface.

3 The communication thread handles the packets containing the Cartesian position and the Euler's angles sent to the Phantom Omni application from the PUMA software controller.

10.2 General Discussion

The integration of haptic feedback and the generation assistance based on sensory information is a challenge due to the strict timing constraints for a realistic sensation of touch and high update rates of the sensory inputs. Additionally, the combination of visual and haptic information depends on computationally intensive pre-processing to obtain the digital features from the images. In this dissertation a multithreaded architecture was designed and implemented to deal with the timing constraints and high update rates imposed by separating the computational tasks into different running threads with synchronization mechanisms for inter-processing communication to achieve real-time performance. The following is a list of the major contributions made in this dissertation:

1. A multithreaded PC-based control scheme capable of real-time haptic and visual feedback

المنارة للاستشارات
www.manaraa.com

2. Implementation of sensor-based assist functions (SAF's) for guiding the user's motion in the form of scaling, motion-based and force-based virtual fixture
3. The development of an automatic control mode to enhance the manipulation capabilities of the users and for reducing the possibility of fatigue over long periods of times
4. The integration of a laser-range finder for the determination of the desired trajectory by pointing the laser to the object of interest
5. An integrated approach for handling diverse sensor datasets and data acquisition

10.3 Recommendations

It is recommended to improve the computer vision sub-system to include more sophisticated feature extraction algorithms and object recognition techniques. The experimental tests were performed successfully for a single object in the field of view of the camera and laser range finder and the computation of the centroid of the object of interest, however, it is recommended to include "blobs" detection capabilities in order to detect and to label multiple objects in the field of view of the camera, and then, use probabilistic techniques for object recognition. Some geometrical features such as the centroid, area, perimeter, and roundness of the detected objects can be compared with existing geometrical features enumerated in a database for this purpose. This would add flexibility to the trajectory generation in the presence of multiple objects as well as to the autonomous mode control of the telerobotic system. Also, another recommendation is to enable the laser-tracking of moving objects by using the current capabilities of the system

المنارة للاستشارات
www.manaraa.com

for image processing and data fusion of the sensory information from the camera, laser range finder, and encoder readings. The multithreaded approach used proved to support high update rates of the sensory data which are fundamental for the tracking of moving objects.

It is also recommended to extend the sensor-based assist force (SAF's) concepts to include torque feedback. This requires force feedback in six degrees of freedom. In the current implementation, the SAF's are 3-DoF output and, therefore, the assistance provided corresponds to force components along the Cartesian axes. However, for enhanced manipulation in 3D space, assisting or resisting torques may also be useful for certain tasks. In the hardware side, the Phantom Omni will need to be replaced by a 6-DoF haptic interface capable of reflecting torques. Commonly ADL tasks requiring user's actions such as “turn”, “push”, “insert” can also be enhanced by a 6-DoF force-based virtual fixture teleoperation mode.

للاستشارات المنارة

www.manaraa.com

Chapter 11

Future Work

11.1 Introduction

As previously discussed, the methods developed in this dissertation allowed the execution of telerobotic manipulation tasks by the combination of visual information using simple sensors and haptic force feedback to calculate assistive functions in real-time. In the current version of the telerobotic control system, the calculation of the assistive force for guiding the user's motion and the determination of the position and orientation of an object of interest as "seen" by the sensors (eye-in-hand camera and laser range finder) is based on a fixed reference frame located at the Puma 560 base. Having this system controlling a robot on a mobile platform with sensor-based assist functions such as the Wheelchair Mounted Robotic Arm (WMRA) may increase the flexibility of such system as an assistive device. This chapter describes potential research problems that the development of a real-time telerobotic control system with sensor-based assist functions for a robot-mobile platform would entail.

11.2 Combined Mobility and Manipulation with Time-dependant Sensory Calibration Functions in Real-time

The idea is to design a real-time control scheme which combines the control strategies required for maximizing the combined mobility and manipulation capabilities as implemented in [72], and, at the same time, implement the time-dependent sensory

www.manaraa.com

calibration functions required to calculate the sensor-based assist functions (SAF's) as described in this dissertation. The integration of a real-time telerobotic control system with sensor-based assist functions and the "Wheelchair Mounted Robotic Arm", WMRA, entails the implementation of optimized numerical approaches to deal with the redundancy of the WMRA system as well as the online calibration functions to determine the feedback force to guide the user's motion based on the sensor readings. Such development would benefit users who are vision-impaired and also forced to use a wheelchair.

11.3 Autonomous Navigation

The implementation of navigational technologies with advanced perception through the use of sensor fusion, autonomy and learning techniques might benefit from the development of a Hybrid-Deliberative Architecture (HDA). HDA techniques might provide a suitable solution when the environment can not be altered to accommodate the robot's needs. Behavior-based robotics and Neuro-Fuzzy techniques for inference and learning might be combined. In this scenario, Neural Networks (NN) might be extended to automatically extract fuzzy rules from sensory information (or numerical data) while Fuzzy Logic (FL) techniques might be used to resolve conflicts and control of primitive behaviors. Hybrid-Deliberative systems and methods are not commonplace and correspond to efforts of current research. Such implementation will require highly responsive and stable computer and software architectures. The multithreading framework developed for this work has the capabilities to perform in real-time and implements a high-level communication protocol to deal with different sensory input

www.manaraa.com

formats (RS232, RS485, parallel, USB, IEEE1392, among others). These capabilities could serve as the foundation of the Hybrid-Deliberative approach.

11.4 Remote Assistance

As already implemented, the system provides force assistance based on the visual feedback and laser readings. A similar setup can be implemented with the added capability for monitoring of the WMRA from a remote location using communication channels over the Internet-based protocol. The sensory suite can be mounted at the end-effector of the wheelchair-mounted robot arm, similar to the current version of the Puma 560 testbed. The present user interface will have to be modified to accommodate the visual information from the optical sensors and the haptic graphical display interfaces to be available online to the remote assistant. This way the remote human user will be able to observe the environment around the WMRA. Using a haptic device as an input, the remote assistant can specify the desired motion to assist the disable person remotely. Several of the methods described in this thesis will be useful for this application.

المنارة للاستشارات
www.manaraa.com

References

[1] J. K. Salisbury, and M. A. Srinivasan**,** 1997, "Phantom-based Haptic Interaction with Virtual Objects", IEEE Computer Graphics and Applications, Vol.17, Issue 5, Sept-Oct, 6-10.

[2] N. Diolaiti, G. Niemeyer, F. Barbagli, J. K. Salisbury, and C. Melchiorri, 2005, "The Effect of Quantization and Coulomb Friction on the Stability of Haptic Rendering" in Proc. 1st World Haptics Conference, Pisa, Italy, Mar, 237-246.

[3] T. Massie and J. Salisbury, 1994, "The PHANTOM Haptic Interface: A Device for Probing Virtual Objects" in Proc. ASME Winter Annual Meeting, Vol. 55-1, New Orleans, LA, 295–300.

[4] Z.Y. Yang, Y.H. Chen, 2003, "Haptic Rendering of Milling Encoding", Proceedings of the EuroHaptics 2003, Dublin, Ireland, 6–9 July, 2003, 206–217.

[5] P. Leskovsky, M. Harders, and G. Szekely, 2006, "Assessing the Fidelity of Haptically Rendered Deformable Objects", IEEE Haptics Symposium, Virginia, USA, March 25-26, 19-25.

[6] Y. Guangqi, J.J. Corso, G.D Hager, and A.M. Okamura, "VisHap: Augmented Reality Combining Haptics and Vision", IEEE International Conference on Systems**,** Man and Cybernetics**,** Vol. 4, 5-8 October, 2003, pp. 3425 – 3431.

[7] T. L. McDaniel, and S. Panchanathan, "A Visio-Haptic Wearable System for Assisting Individuals Who Are Blind", SIGACCESS Accessibility Computing, September, 2006.

[8] C. R. Wagner, S. J. Lederman, and R. D. Howe, "A Tactile Shape Display Using RC Servomotors", Proceedings of the 10th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, pp. 354-355, 2002.

[9] V. Hayward and M. Cruz-Hernandez, "Tactile Display Device Using Distributed Lateral Skin Stretch", Proceedings of the 8th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, ASME IMECE, DSC-69-2, pp. 1309-1314**,** 2000.

[10] T.B. Sheridan, 1992, "Telerobotics, Automation and Human Supervisory Control", MIT Press, ISBN: 0-262-19316-7.

المنارة للاستشارات
www.manaraa.com

[11] J. Vertut and P. Coiffet, “Teleoperation and Robotics: Evolution and Development”, Hermes Publishing, ISBN: 1-85121-002, Vol. 3A, pp23-63.

[12] W.R. Ferrell and T.B. Sheridan, 1967, “Supervisory Control of Remote Manipulation”, IEEE Spectrum, 4, No 10, October, pp 81-88.

[13] T.B. Sheridan, 1993, “Space teleoperation through time delay: review and prognosis”, IEEE Transactions on Robotics and Automation, Vol. 9, No. 5, October 1993, pp. 592-606.

[14] D.R. Yoerger and J. E. Slotine, 1991, “Adaptive sliding control of an experimental underwater vehicle”, in Proceedings of 1991 IEEE International Conference on Robotics and Automation, Vol. 3, 9-11 April, pp. 2746-2751.

[15] D. R. Yoerger, J. Newman, and J. E Slotine, 1986, **“**Supervisory control system for the JASON ROV”**,** IEEE Journal of Oceanic Engineering, Vol. 11, Issue 3, July1986, pp. 392 – 400.

[16] Young S. Park, Hyosig Kang, Tomas F. Ewing, Eric L. Faulring, J. Edward Colgate, Michael A. Peshkin, " Enhanced Teleoperation for D & D”, in the 2004 IEEE International Conference on Robotics and Automation, New Orleans, 2004.

[17] T. F. Chan and R. V. Dubey, “Design and Experimental Studies of a Generalized Bilateral Controller for a Teleoperator System with a Six DoF Master and a Seven DoF Slave”, in Proc. of the 1994 IEEE International Conference on Robotics and Automation, San Diego, CA, USA, May 1994, Volume 3, pp 2612-2619.

[18] R. V. Dubey, T. F. Chan and S. E. Everett, “Variable Damping Impedance Control of a Bilateral Telerobotic System,” IEEE Control System Magazine, February 1997.

[19] Luc D. Joly and Claude Andriot, “Imposing motion constraints to a force reflecting telerobot through real-time simulation of a virtual mechanism”, in Proceedings of the 1994 IEEE International Conference on Robotics and Automation, San Diego, CA, May 1994, pp. 357-362.

[20] T. F. Chan and R. V. Dubey, “Generalized Bilateral Controller for a Teleoperator System with a Six DoF Master and a Seven DoF Slave”, Proceedings of the IEEE International Conference on Robotics and Automation, San Diego, California, May 8-13, 1994, pp. 2612-2619.

[21] K. Sato, M. Kimura, and A. Abe, “Intelligent Manipulator System with Nonsymmetric and Redundant Master-Slave”, Journal of Robotic System, 9(2), pp. 281-290, 1992.

المنارة للاستشارات
www.manaraa.com

[22] E.J. Veras, R. Swaminathan, and R. Dubey, "A Multithreaded Implementation of Assist Functions to Control a Virtual Reality Model of a 6-DoF Robot Arm for Rehabilitation Applications", Florida Conference on Recent Advances in Robotics and Robot Showcase, FCRAR, 2007, Tampa, Florida, May 31-June 01, 2007.

[23] G. Bolmsjo, H. Neveryd, and H. Ftring. "Robotics in Rehabilitation", IEEE Transactions on Rehabilitation Engineering, Vol. 3, No. 1, March, 1995.

[24] N. Pernalete, "Development of a Robotic Haptic Interface to Perform Vocational Tasks by People with Disabilities", Ph.D. Dissertation, Department of Electrical Engineering, University of South Florida, December 2001.

[25] C. Stanger, C. Angling, W. Harwin, D. Romilly, "Devices for Assisting Manipulation: A Summary of User Task Priorities", 1994 IEEE Transactions on Rehabilitation Engineering, Vol 2, No 4, December 1994.

[26] N. Pernalete, W. Yu, R. V. Dubey, and W.A. Moreno, "Development of an Intelligent Mapping Based Telerobotic Manipulation System To Assist Persons With Disabilities", In Proc. of the 2002 IEEE International Conference on Robotics & Automation, Washington, DC USA., May 2002.

[27] K. Kawamura, S. Bagchi, M. Iskarous, and M. Bishay, "Intelligent Robotic Systems in Service of Disabled", 1995 IEEE Transactions on Rehabilitation Engineering, Vol. 3, No. 1, March 1995.

[28] S. E. Everett, R.V. Dubey, Y. Isoda, and C. Dumont, "Vision-Based End-Effector Alignment Assistance for Teleoperation". Proceedings of the IEEE International Conference on Robotics and Automation, Detroit, MI.. May 1999, pp. 543-549.

[29] W. Yu, B. Fritz, N. Pernalete, M. Jurczyk, and R. V. Dubey, "Sensors Assisted Telemanipulation for Maximizing Manipulation Capabilities of Persons With Disabilities", Haptics Symposium 2003, Los Angles, CA, 2003.

[30] N. Pernalete, W. Yu, and R. Dubey, "Augmentation of manipulation Capabilities of Persons with Disabilities Using Scaled Telemanipulation", IEEE/RSJ International Conference on Intelligent Robots and Systems, Lausanne, Switzerland, October 2002.

[31] Steven Edward Everett, "Human-Machine Cooperative Telerobotics Using Uncertain Sensor and Model Data". Ph.D. Dissertation, University of Tennessee, Knoxville, 1998.

[32] L.B Rosenberg, Virtual fixtures: Perceptual tools for telerobotic manipulation", IEEE Virtual Reality Annual International Symposium, 18-22 Sep 1993, pp. 76 -82.

المنارة للاستشارات

www.manaraa.com

[33] Kazuhiro Kosuge, Koji Takeo, and Toshio Pukuda, "Unified approach for teleoperation of virtual and real environment manipulation based on reference dynamics", in Proc. of the 1995 IEEE International Conference on Robotics and Automation, Nagoya, Japan, May1995.

[34] P. Marayong, A. Bettini and A. Okamura, "Effect of Virtual Fixture Compliance on Human-Machine Cooperative Manipulation", in IEEE/RSJ Proc., Lausanne, Switzerland, Oct 2002.

[35] Z. Stanisic, S. Payandeh, and E. Jackson, "Virtual Fixture as an Aid for Teleoperation", in 9th Canadian Aeronautic and Space Inst. Conference, 1996.

[36] S, Payandeh, and Z. Stanisic, "On Application of Virtual Fixtures as an Aid for Telemanipulation and Training", 10th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2002.

[37] N. Costescu, M. Loffler, E. Zergeroglu, D. Dawson, "QRobot – A Multitasking PC Based Robot Control System", Microcomputer Applications Journal Special Issue on Robotics, Vol. 18 No. 1, pages 13-22.

[38] E. Jung, C. Kapoor, and D. Batory, 2005, "Automatic Code Generation for Actuator Interfacing from a Declarative Specification", IEEE International Conference on Robotics and Systems (IROW), Edmonton, Canada.

[39] Herman Bruyninckx and Peter Soetens, "Generic Real-Time Infrastructure for Signal Acquisition, Generation and Processing", 4th Real-time Linux Workshop, Boston, MA, December, 2002.

[40] Microsoft Robotics Studio http://msdn2.microsoft.com.

[41] S. Cherry, "Robots, Incorporated", IEEE Spectrum Magazine, August, 2007, pp 24-29.

[42] N. Turro, O. Khatib, and E. Coste-Maniere, "Haptically Augmented Teleoperation", in Proc. IEEE International Conference on Robotics and Automation, Seoul, Korea, May 21-26, 2001, pp 386-392.

[43] N. Hogan, H. I. Krebs, J. Charnnarong, P. Srikrishna, and A. Sharon, "MIT-MANUS: a workstation for manual therapy and training II", International Society for Optical Engineering, 2005.

[44] S. Charles, H. Das, T. Ohm, C. Boswell, G. Rodriguez, and R. Steele, "Dexterity-enhanced Telerobotic Microsurgery", MicroDexterity Systems, Inc., Charles Retina Institute, Memphis, TN, and JPL/NASA, California Institute of Technology, Pasadena, CA, 1997.

www.manaraa.com

[45] S. E. Everett and R. V. Dubey, “Sensor-Assisted Variable Trajectory Mapping for Telerobotic Task Execution”, in Proc. of IEEE International Conference on Robotics and Automation, 1998.

[46] R.V. Dubey, S.E. Everett, N. Pernalete, and K.A, Manocha**,** 2001**,** "Teleoperation Assistance through Variable Velocity Mapping" IEEE Transactions on Robotics and Automation, Vol. 17, Issue 5, Oct. 2001, 761–766.

[47] W. Yu, R. Dubey and N. Pernalete, “Robotic Therapy for Persons with Disabilities Using Hidden Markov Model Based Skill Learning”, First Conference on Intelligent Manipulation and Grasping, Genova, Italy, July 1-2, 2004.

[48] L. L. Kovács and G. Stépán, “Dynamics of Digital Force Control Applied in Rehabilitation Robotics”, Meccanica Journal, Springer Netherlands, Vol. 38, No. 2, March, 2003, pp. 213-226.

[49] A. Steinfeld, T. Fong, D. Kaber, M. Lewis, J. Scholtz, A. Schultz, and M. Goodrich, "Common metrics for human-robot interaction", in Proc. 2006 ACM Conference on Human-Robot Interaction, 2006, pp 33-40.

[50] QNX RTOS Soft Systemswww.qnx.com/index.html.

[51] J. Craig, 2003, "Introduction to Robotics Mechanics and Control", 3rd Edition, Addison-Wesley Publishing, ISBN 0201543613.

[52] B. Armstrong, O. Khatib, and J. Burdick, “The explicit dynamic model and inertial parameters of the Puma 560 arm” in *Proc.* IEEE International Conference on Robotics and Automation, Vol. 1, San Francisco, USA, pp. 510–8, 1986.

[53] D. E. Whitney, 1969, "Resolved motion rate control of manipulators and human prostheses", IEEE Transactions on Man and Machine Systems, Vol. MMS-10, June, 47-53.

[54] D. E. Whitney, 1972, "The mathematics of coordinated control of prosthetic arms and manipulators", ASME Journal of Dynamic Systems, Measurement, and Control, Dec., 303-309.

[55] J.Y.S Luh, M.W. Walker, and R.P. Paul, “Resolved-acceleration Control of Mechanical Manipulators”, IEEE Transactions on Automatic Control, Vol. AC-25, No.3, June, 1980, pp 468-474.

[56] R. Dubey, and J. Y. S Luh, 1987, "Redundant Robot Control for Higher Flexibility", in Proc. of 1987 IEEE International Conference on Robotics and Automation, Vol.4, March 1987, 1066-1072.

المنارة للاستشارات
www.manaraa.com

[57] M. Takegaki and S. Arimoto, 1981, "A New Feedback Method for Dynamic Control of Manipulators", J. Dyn. Sys. Meas. Control Transaction. ASME,103 119-125, 1981.

[58] A. Fisher, and J. M. Vance, 2003, "PHANTOM Haptic Device Implemented in a Projection Screen Virtual Environment", 7th International Immersive Projection Technologies Workshop, Eurographics Workshop on Virtual Environments, 225-230.

[59] R. Krten, 2001, "Getting Started with QNX Neutrino 2: A Guide for Realtime Programmers", Parse Software Devices, Ontario, Canada, ISBN 0-9682501-1-4.

[60] R. P. Paul, "Robot Manipulators: Mathematics, Programming and Control", MIT Press, Boston, 1981, ISBN: 0-262-16082-X.

[61] R. R. Murphy, "Introduction to AI Robotics", The MIT Press, Cambridge, Massachusetts, 2000. ISBN: 0-262-13383-0.

[62] Roger Y. Tsai, "A versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses", IEEE Journal of Robotics and Automation, Vol. RA-3, No. 4, August 1987, 323-344.

[63] J. Y. Bouguet, "A Camera Calibration Toolbox for MatLab", www.vision.caltech.edu/bouguetj/calib_doc.

[64] R.M. Haralick, and L.G. Shapiro, "Computer and Robot Vision", Addison Wesley Publishing Co., ISBN: 0-201-10877-1, Vol. I, pp 60-93.

[65] R. Willson, "Modeling and Calibration of Automated Zoom Lenses", in Proceedings of the SPIE #2350, Videometrics III, October 1994, pp.170-186.

[66] J. Canny, 1986, "A Computational Approach to Edge Detection", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 8, Issue 6, Nov, 679-698.

[67] Z. Zhang, "Flexible Camera Calibration by Viewing a Plane from Unknown Orientations", Microsoft Research, One Microsoft Way, Redmond, Washington, 98052-6399, USA.

[68] Ortega, M., Redon, S., and Coquillart, S., 2006, "A Six Degree-of-Freedom God-Object Method for Haptic Display of Rigid Bodies", Virtual Reality Conference, 25-29 March 2006, 191-198.

[69] Paljic, A., Burkhardtt, J.M., and Coquillart, S., 2004, "Evaluation of pseudo-haptic feedback for simulating torque: a comparison between isometric and elastic input devices", Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2004. 12th International Symposium on Haptics, 27-28 March, 2004, 216-223.

المنارة للاستشارات
www.manaraa.com

[70] Tarn, T.J., Bejczy, A.K., Marth, G.T. and Ramadorai, A.K. "Performance Comparison of Four Manipulator Servo Schemes", IEEE Control Systems Magazine, February, 1993, pp 22-29.

[71] Douglas C. Montgomery, and George C. Runger, “Applied Statistics and Probability for Engineers”, 3rd Edition, John Wiley & Sons, Inc. 2002, ISBN: 0-471-20454-4.

[72] R. M. Alqasemi, "Maximizing Manipulation Capabilities of Persons with Disabilities Using a Smart 9-Degree-of-Freedom Wheelchair-Mounted Robotic Arm System", Ph.D. Dissertation, University of South Florida, 2007.

المنارة للاستشارات
www.manaraa.com

Bibliography

The following bibliography was revised and studied for learning fundamentals of some robotics and haptic concepts during the course of this research. It is listed for the benefit of potential readers.

A. J. Davison, "Real-Time Simultaneous Localization and Mapping with a Single Camera", ICCV 2003.

C. S. G. Lee, “Robot arm kinematics, dynamics and control,” IEEE J. Computer, Vol. 15, pp. 62–80, Dec. 1982.

D. K. Swanson and W. J. Book, “Path-Following Control for Dissipative Passive Haptic Displays”, 11th International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, Los Angeles, California, pp 101-108, March 22-23, 2003.

E. Chen, 1999, "Six degree-of-freedom Haptic System for Desktop Virtual Prototyping Applications, Proceedings of the First International Workshop on Virtual Reality and Prototyping, Laval, France, June, 1999, 97–106.

Ho, C., Basdogan, C., Srinivasan, M.A., 1998, "An Efficient Haptic Rendering Technique for Displaying Polygonal Objects with Surface Details in Virtual Environments" submitted to Presence: Teleoperators and Virtual Environments.

H.T. Yau, C.H. Menq, 1995, "Automated CMM Path Planning for Dimensional Inspection of Dies and Molds Having Complex Surfaces", International Journal of Machine Tools and Manufacture 35 (6) (1995) 861–876.

J. M. Hollerbach, “A Survey of Kinematic Calibration”, In O. Khatib, J. J. Craig, and T. Lozano-Perez, editors, Robotics Review 1, The MIT Press, Cambridge, MA, 1989.

J. M. Prager, 1980, "Extracting and labeling boundary segments in natural scenes", IEEE Transactions PAMI 2, January 1st, 16-27.

J. T. Feddema, C. S. George Lee, and O. R. Mitchell, “Weighted selection of image features for resolved rate visual feedback control”, IEEE Transactions on Robotics and Automation, 7:31-47, 1991.

المنارة للاستشارات
www.manaraa.com

J. Weng, P. Cohen, and M. Herniou, “Camera calibration with distortion models and accuracy evaluation”, IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(10), Oct. 1992, pp. 965–980.

L.G. Shapiro and G.C. Stockman, “Computer Vision”, Prentice Hall, ISBN: 0-13-030796-3, pp. 422-453.

M. Ikits, C. Hansen, and C. R. Johnson, 2003, “A Comprehensive Calibration and Registration Procedure for the Visual Haptic Workbench”, Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, Utah, U.S.A.

M. Li and A. M. Okamura, “Recognition of Operator Motions for Real-Time Assistance using Virtual Fixtures”, 11th International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, Los Angeles, California, pp 125-131, March 22-23, 2003.

M. Skubic, and R.A. Volz, 2004, "Learning Force Sensory Patterns and Skills from Human Demonstration", International Journal of Machine Tools and Manufacture 44, 18 March, 2004, 1009–1017.

N.I. Durlach, A.S. Mavor, 1995, Virtual Reality: Scientific and Technological Challenges, National Academy Press, Washington, DC, 1995.

P. I. Corke, 1996, "A Robotics Toolbox for MatLab," IEEE Robotics and Automation Magazine, Vol. 3, 24-32.

P. I. Corke, and B.Armstrong-Helouvry, 1994, "A Search for Consensus among Model Parameters Reported for the PUMA 560 Robot" in Proc. of 1994 IEEE International Conference on Robotics and Automation, Vol. 2, 8-13 May, 1994, 1608-1613.

R. B. Gillespie, J. E. Colgate, and M. A. Peshkin, “A general framework for Robot Control”, IEEE Transactions on Robotics and Automation, Vol. 17, Num 4, pp 391-401, August 2001.

R. Hartley and A. Zisserman, 2004, "Multiple View Geometry in Computer Vision", Cambridge, MA, 2nd Edition, Cambridge University Press, ISBN 0521540518.

R. M. Haralick,H., Joo, C. Lee, X. Zhuang, V.G. Vaidya, and M. B. Kim, 1989, "Pose Estimation from Corresponding Point Data", IEEE Transactions On Systems, Man. And Cybernetics, Vol. 19, No. *6*, November/December 1989.

R.P. Paul, and C. N. Stevenson**,** 1983, "Kinematics of Robot Wrists", IEEE International Journal of Robotics Research, Vol. 1, No**.** 2, 33-38.

R. Paul, M. Rong, and H. Zhang, “Dynamics of Puma manipulator,” in American Control Conference, San Francisco, CA, 22-24 June, 1983, pp. 491-496.

المنارة للاستشارات

www.manaraa.com

R. P. Paul and H. Zhang, “Computationally efficient kinematics for manipulators with spherical wrists,” International Journal of Robotics Research, Vol. 5, No. 2, 1986.

S. E. Salcudean, N. M. Wong, and R. L. Hollis, “Design and Control of a Force-Reflecting Teleoperation System with Magnetically Levitated Master and Wrist”, IEEE Transactions on Robotics and Automation, Vol. 11, No. 6, December 1995, pp. 844-858.

T. Gutierrez, J.I. Barabero, M. Aizpitarte, A.R. Cariilo, A. Eguidazu, 1998, "Assembly Simulation Through Haptic Virtual Prototypes", Proceedings of the 3rd Phantom Users Group Workshop, Dedham, MA, 3–6 October, 1998.

T. Yoshikawa, 1990, "Foundations of Robotics: Analysis and Control", MIT Press, ISBN: 0262240289.

W.H. Press, B.P. Flannery, S. A. Teukolsky, and W.T. Vetterling, 1988, “Numerical Recipes in C”, 2nd Edition, Cambridge University Press, Cambridge, MA, USA, ISBN: 0-521-43108-5.

Y. Abdel-Aziz, and H. Karara, 1971, "Direct linear transformation from comparator coordinates into object space coordinates", in Proc. of ASP/UI Symposium on Close-Range Photogrammetric Systems, Urbana, IL., pp. 1-48.

Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, "Model-based and image-based 3D scene representation for interactive visualization", Computer Vision and Image Understanding, Vol. 96, Issue 3, December 2004, 274 – 293.

المنارة للاستشارات

www.manaraa.com

Appendices

www.manaraa.com

Appendix A: Puma 560 Homogeneous Transformations

The homogeneous transformations are obtained from the substitution of the DH parameters in Table 3.1 into the transformation equation given by Eq. [6] yields to:

$$ {}^{0}_{1}T = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(A.1)} $$

$$ {}^{1}_{2}T = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\sin\theta_2 & -\cos\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(A.2)} $$

$$ {}^{2}_{3}T = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & a_2 \\ \sin\theta_3 & \cos\theta_3 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(A.3)} $$

$$ {}^{3}_{4}T = \begin{bmatrix} \cos\theta_4 & -\sin\theta_4 & 0 & a_3 \\ 0 & 0 & 1 & d_4 \\ -\sin\theta_4 & -\cos\theta_4 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(A.4)} $$

$$ {}^{4}_{5}T = \begin{bmatrix} \cos\theta_5 & -\sin\theta_5 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ \sin\theta_5 & \cos\theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(A.5)} $$

$$ {}^{5}_{6}T = \begin{bmatrix} \cos\theta_6 & -\sin\theta_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\sin\theta_6 & -\cos\theta_6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{(A.6)} $$

المنارة للاستشارات
www.manaraa.com

Multiplying (A.1) – (A.6), the homogeneous transformation matrix of the end-effector frame, {6}, in terms of the reference frame {0} corresponding to the base of the robot (See Figure 3.1) as can be now be calculated:

$$ {}^{0}_{6}T = {}^{0}_{1}T \cdot {}^{1}_{2}T \cdot {}^{2}_{3}T \cdot {}^{3}_{4}T \cdot {}^{4}_{5}T \cdot {}^{5}_{6}T \tag{A.7} $$

The symbolic evaluation of Eq. (A.7) can be written as:

$$ {}^{0}_{6}T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.8} $$

where,

$$ r_{11} = c_1[c_{23}(c_4c_5c_6 - s_4s_5) - s_{23}s_5c_5] + s_1(s_4c_5c_6 + c_4s_6) \tag{A.9} $$

$$ r_{21} = s_1[c_{23}(c_4c_5c_6 - s_4s_6) - s_{23}s_5c_6] - c_1(s_4c_5c_6 + c_4s_6) $$

$$ r_{31} = -s_{23}(c_4c_5c_6 - s_4s_6) - c_{23}s_5c_6 $$

$$ r_{12} = c_1[c_{23}(-c_4c_5s_6 - s_4c_6) + s_{23}s_5s_6] + s_1(c_4c_6 - s_4c_5s_6) $$

$$ r_{22} = s_1[c_{23}(-c_4c_5s_6 - s_4c_6) + s_{23}s_5s_6] - c_1(c_4c_6 - s_4c_5s_6) $$

$$ r_{32} = -s_{23}(-c_4c_5s_6 - s_4c_6) + c_{23}s_5s_6 $$

$$ r_{13} = -c_1(c_{23}c_4s_5 + s_{23}c_5) - s_1s_4s_5 $$

$$ r_{23} = -s_1(c_{23}c_4s_5 + s_{23}c_5) - c_1s_4s_5 $$

$$ r_{33} = s_{23}c_4s_5 - c_{23}c_5 $$

$$ p_x = c_1(a_2c_2 + a_3c_{23} - d_4s_{23}) - d_3s_1 $$

$$ p_y = s_1(a_2c_2 + a_3c_{23} - d_4s_{23}) + d_3c_1 $$

$$ p_z = -a_3s_{23} - a_2s_2 - d_4c_{23} $$

المنارة للاستشارات
www.manaraa.com

Appendix B: Equivalent Single Angle-Axis Representation

The homogeneous transformation matrix, T, which describes a rotation around an arbitrary axis vector κ and an angle defined as θ is given by the following matrix [48].

$$T = \begin{bmatrix} \kappa_x\kappa_x V_\theta + c_\theta & \kappa_y\kappa_x V_\theta - \kappa_z s_\theta & \kappa_z\kappa_x V_\theta + \kappa_y s_\theta & P_x \\ \kappa_x\kappa_y V_\theta + \kappa_z s_\theta & \kappa_y\kappa_y V_\theta + c_\theta & \kappa_z\kappa_y V_\theta - \kappa_x s_\theta & P_y \\ \kappa_x\kappa_z V_\theta - \kappa_y s_\theta & \kappa_y\kappa_z V_\theta + \kappa_x s_\theta & \kappa_z\kappa_z V_\theta + c_\theta & P_z \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{B.1}$$

where, $s_\theta = \sin(\theta)$, $c_\theta = \cos(\theta)$, and $V_\theta = 1 - \cos(\theta)$, and $(\kappa_x, \kappa_y, \kappa_z)$ are the directional components of the rotational axis κ. The (3x3) rotation matrix is, then:

$$R = \begin{bmatrix} \kappa_x\kappa_x V_\theta + c_\theta & \kappa_y\kappa_x V_\theta - \kappa_z s_\theta & \kappa_z\kappa_x V_\theta + \kappa_y s_\theta \\ \kappa_x\kappa_y V_\theta + \kappa_z s_\theta & \kappa_y\kappa_y V_\theta + c_\theta & \kappa_z\kappa_y V_\theta - \kappa_x s_\theta \\ \kappa_x\kappa_z V_\theta - \kappa_y s_\theta & \kappa_y\kappa_z V_\theta + \kappa_x s_\theta & \kappa_z\kappa_z V_\theta + c_\theta \end{bmatrix} \tag{B.2}$$

The first three elements of column fourth of T are the components of the position vector, P:

$$P = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} \tag{B.3}$$

A linear trajectory in Cartesian space can now be generated between two points defined by their corresponding homogenous transformation matrices, T_1 and T_2, where:

المنارة للاستشارات
www.manaraa.com

$$T_1 = \begin{bmatrix} {}^1n_x & {}^1o_x & {}^1a_x & {}^1P_x \\ {}^1n_y & {}^1o_y & {}^1a_y & {}^1P_y \\ {}^1n_z & {}^1o_z & {}^1a_z & {}^1P_z \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \qquad \text{(B.4)}$$

and

$$T_2 = \begin{bmatrix} {}^2n_x & {}^2o_x & {}^2a_x & {}^2P_x \\ {}^2n_y & {}^2o_y & {}^2a_y & {}^2P_y \\ {}^2n_z & {}^2o_z & {}^2a_z & {}^2P_z \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix} \qquad \text{(B.5)}$$

If N intermediate points are desired between the initial point defined by the homogeneous transformation T_1 and the destination position defined by T_2, the linear components can be found as:

$$dx = \frac{{}^2P_x - {}^1P_x}{N} \qquad dy = \frac{{}^2P_y - {}^1P_y}{N} \qquad dz = \frac{{}^2P_z - {}^1P_z}{N} \qquad \text{(B.6)}$$

For the rotational components, the following calculations are required. Notice that the transform is used instead of the inverse because the rotation matrix is orthogonal:

$$R = \begin{bmatrix} {}^1n_x & {}^1o_x & {}^1a_x \\ {}^1n_y & {}^1o_y & {}^1a_y \\ {}^1n_z & {}^1o_z & {}^1a_z \end{bmatrix}^T \begin{bmatrix} {}^2n_x & {}^2o_x & {}^2a_x \\ {}^2n_y & {}^2o_y & {}^2a_y \\ {}^2n_z & {}^2o_z & {}^2a_z \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \qquad \text{(B.7)}$$

Before proceeding, it is convenient to ensure that the elements of the resulting matrix define an orthogonal matrix. This is accomplished by the cross product and taking any two columns as follows:

$$r_{13} = r_{21}r_{32} - r_{22}r_{31} \qquad r_{23} = r_{11}r_{32} - r_{12}r_{31} \qquad r_{33} = r_{11}r_{22} - r_{12}r_{21} \qquad \text{(B.8)}$$

المنارة للاستشارات
www.manaraa.com

Now, the equivalent single rotation angle θ can be found from the r_{ij} elements of the rotation matrix given by Eq. (B.7) and (B.8), as follows:

$$\theta = \text{atan2}\left(\sqrt{(r_{32}-r_{23})^2+(r_{31}-r_{13})^2+(r_{21}-r_{12})^2},(r_{11}+r_{22}+r_{33}-1)\right) \tag{B.9}$$

Using the equivalent angle, the directional components of the single axis $(\kappa_x,\kappa_y,\kappa_z)$ can now be found using the following set of equations. Notice that these equations include provisions to avoid the representational singularities (i.e. the axis is poorly defined) arising from situations where the angle of rotation is very small (defined by a tolerance, Toler), or 180°. The following equations are evaluated:

$$\text{If} \quad \theta <= \text{Toler} \left\{ \kappa = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right. \tag{B.10}$$

$$\text{If} \quad \theta < 90° \left\{ \kappa = \begin{bmatrix} \dfrac{(r_{32}-r_{23})}{2*s_\theta} \\ \dfrac{(r_{13}-r_{31})}{2*s_\theta} \\ \dfrac{(r_{21}-r_{12})}{2*s_\theta} \end{bmatrix} \right. \tag{B.11}$$

$$\text{If} \quad \theta >= 90° <= 180° \begin{cases} \kappa_x = \text{sign}(r_{32}-r_{23}).\sqrt{\dfrac{(r_{11}-c_\theta)}{V_\theta}} \\ \kappa_y = \text{sign}(r_{13}-r_{31}).\sqrt{\dfrac{(r_{22}-c_\theta)}{V_\theta}} \\ \kappa_z = \text{sign}(r_{21}-r_{12}).\sqrt{\dfrac{(r_{33}-c_\theta)}{V_\theta}} \end{cases} \tag{B.12}$$

www.manaraa.com

In Eq. B.12, the following substitutions are needed to ensure the most positive components of are $\left(n_x, o_y, a_z\right)$ used:

$$\text{If } \left(\kappa_x > \kappa_y\right) \cap \left(\kappa_x > \kappa_z\right) \begin{cases} \kappa_y = \dfrac{\left(r_{21} + r_{12}\right)}{2\kappa_x V_\theta} \\ \kappa_z = \dfrac{\left(r_{13} + r_{31}\right)}{2\kappa_x V_\theta} \end{cases} \tag{B.12a}$$

$$\text{If } \left(\kappa_y > \kappa_x\right) \cap \left(\kappa_y > \kappa_z\right) \begin{cases} \kappa_y = \dfrac{\left(r_{21} + r_{12}\right)}{2\kappa_y V_\theta} \\ \kappa_z = \dfrac{\left(r_{32} + r_{23}\right)}{2\kappa_y V_\theta} \end{cases} \tag{B.12b}$$

$$\text{If } \left(\kappa_z > \kappa_x\right) \cap \left(\kappa_z > \kappa_y\right) \begin{cases} \kappa_y = \dfrac{\left(r_{31} + r_{13}\right)}{2\kappa_z V_\theta} \\ \kappa_z = \dfrac{\left(r_{32} + r_{23}\right)}{2\kappa_z V_\theta} \end{cases} \tag{B.12c}$$

Now, a rotation matrix can be obtained for every intermediate point by dividing the equivalent rotation angle θ into (N-1) equally spaced values by substitution of the corresponding components $\left(\kappa_x, \kappa_y, \kappa_z\right)$ of the single axis rotation, Eq. B.10 to B.12, and the evaluation of the conditions to avoid representational singularities in B.12a to B.12c. This procedure will allow having well-defined intermediate transformations between the initial and the goal (destination) transformations.

www.manaraa.com

Appendix C: MatLab Script for the Symbolic Jacobian Matrix

```
function Jac = symJacobn()
%symJacobn calculates the symbolic form of the Jacobian of the manipulator
%with respect to the end-effector frame.
puma560akb;
syms th1 th2 th3 th4 th5 th6 real;
syms th2d th3d th4d th5d th6d real;
syms a3 a4 d2 d3 d4 real;
th=sym('[th1; th2; th3; th4; th5; th6]');

%Symbolic values:
DH=[ 0 0   th(1) 0;  -pi/2  0  th(2) d2;0 a3  th(3) d3; pi/2 a4  th(4) d4;  -pi/2  0 th(5) 0; pi/2   0   th(6) 0];
U=sym('[1 0 0 0;0 1 0 0;0 0 1 0; 0 0 0 1]');

for i=6:-1:1
   dx = [-U(1,1)*U(2,4)+U(2,1)*U(1,4);
        -U(1,2)*U(2,4)+U(2,2)*U(1,4);
        -U(1,3)*U(2,4)+U(2,3)*U(1,4)];

   delt = [U(3,1); U(3,2); U(3,3)];

   Jac(1,i) = dx(1);
             Jac(2,i) = dx(2);
             Jac(3,i) = dx(3);
             Jac(4,i) = delt(1);
             Jac(5,i) = delt(2);
             Jac(6,i) = delt(3);

   TT=rotx(DH(i,1))*transl(DH(i,2),0,0)*rotz(DH(i,3))*transl(0,0,DH(i,4));
   U = TT*U;
end

%The Solution using symbolic approach is:
% ans =
%     0.4995    0.2394    0.3162         0         0         0
%    -0.4457    0.3319    0.2813         0         0         0
%    -0.0303   -0.5160   -0.0941         0         0         0
%     0.4504   -0.6164   -0.6164    0.3309   -0.0479         0
%     0.5524    0.7607    0.7607    0.0159    0.9989         0
%    -0.7014    0.2034    0.2034    0.9435         0    1.0000

% Solution using Corke's toolbox
% jacobn(p560m,qready)
% ans =
%     0.4995    0.2394    0.3162         0         0         0
%    -0.4457    0.3319    0.2813         0         0         0
%    -0.0303   -0.5160   -0.0941         0         0         0
%     0.4504   -0.6164   -0.6164    0.3309   -0.0479         0
%     0.5524    0.7607    0.7607    0.0159    0.9989         0
%    -0.7014    0.2034    0.2034    0.9435    0.0000    1.0000
```

www.manaraa.com

Appendix D: Singularity-Robust (SR) Inverse

The SR inverse [16] is also known as damped pseudoinverse [18]. Considering a linear system of equations as the form:

$$[A]\{x\} = \{b\} \tag{D.1}$$

If the matrix of coefficients $[A]$ is not square, the pseudoinverse A^{+} may be used to compute the least-square solution with the objective function defined as the minimal norm. The pseudo-inverse solution avoids the problem of extremely large amplitude in the neighborhood of singular points by minimizing the sum of the norms of the error (defined as $|b - Ax|$) and the solution $|x|$. For an m-by-n (where m < n) matrix A, its pseudoinverse is computed by:

$$A^{+} = A^{T}\left(A A^{T}\right)^{-1} \tag{D.2}$$

The resulting matrix A^{+} may have extremely large elements when $\left(A A^{T}\right)$ is nearly singular. The SR inverse uses the following equation instead:

$$A^{*} = A^{T}\left(A A^{T} + \kappa I\right)^{-1} \tag{D.3}$$

Where A^{*} is the SR inverse of $[A]$, I is the identity matrix, and κ is the parameter that determines the weighting between the norm of the solution and the error. If a small κ is used, then the error gets small, but the solution might get large around singular points and vice versa [19].

للاستشارات المنارة
www.manaraa.com

Appendix E: Angular Velocities Components of the End-Effector

The Euler's rotation theorem states that any rotation can be defined using three angles (ϕ, θ, ψ), as shown in Figure ZZ. These angles (θ, ϕ, ψ) are called Euler angles.

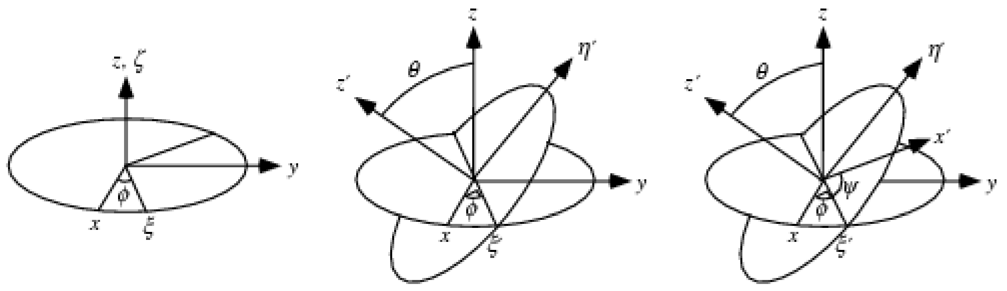


Figure E.1 Definition of the Euler Angles

In robotics it is more convenient to write the Euler's rotation in terms of rotation matrices. For the case of the angular velocity components of the end-effector, the equation that describes the total rotation is $R(\phi,\theta,\psi) = R_z(\psi)\, R_x(\theta)\, R_z(\phi)$. The corresponding rotation matrices in terms of the Euler's angles are:

$$R_z(\phi) = \begin{bmatrix} \cos(\phi) & \sin(\phi) & 0 \\ -\sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix},\ R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix},\ \text{and } R_z(\psi) = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{E.1}$$

Now, the total rotation matrix, R, is found to be:

$$R(\phi,\theta,\psi) = R_z(\psi)\, R_x(\theta)\, R_z(\phi) = \begin{bmatrix} c\psi c\phi - c\theta s\phi s\psi & c\psi s\phi + c\theta c\phi s\psi & s\psi s\theta \\ -s\psi c\phi - c\theta s\phi c\psi & -s\psi s\phi + c\theta c\phi c\psi & c\psi s\theta \\ s\theta s\phi & -s\theta c\phi & c\theta \end{bmatrix} \quad \text{E.2}$$

where $c\psi = \cos(\psi)$, $c\phi = \cos(\phi)$, $c\theta = \cos(\theta)$, $s\psi = \sin(\psi)$, $s\phi = \sin(\phi)$, and $s\theta = \sin(\theta)$.

www.manaraa.com

In the end-effector axis, the components of the angular velocity $\{\omega\}$ are obtained by writing the total rotation matrix as:

$$[R]=\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}=\begin{bmatrix} R_1 & R_2 & R_3 \end{bmatrix} \text{ and } \{\omega\}=\begin{Bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{Bmatrix} \tag{E.3}$$

$$[R]\{\omega\}=\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}\begin{Bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{Bmatrix}=\begin{bmatrix} r_{11}\omega_x & r_{12}\omega_y & r_{13}\omega_z \\ r_{21}\omega_x & r_{22}\omega_y & r_{23}\omega_z \\ r_{31}\omega_x & r_{32}\omega_y & r_{33}\omega_z \end{bmatrix}=\begin{bmatrix} R_1\omega_x & R_2\omega_y & R_3\omega_z \end{bmatrix} \tag{E.4}$$

where ω_z is the rotation about the z- axis by angle ϕ and it is obtained from the total rotation given by Eq. (TT). Taking the z-component as $[R_3]\{\omega_z\}$ yields to:

$$\omega_\phi=\begin{bmatrix} s\psi s\theta \\ c\psi s\theta \\ c\theta \end{bmatrix}\dot{\phi} \tag{E.5}$$

Next, the rotation about the ξ-axis by angle θ, is obtained from ω_ξ given by second column vector of $R_z(\psi)\{\omega\}$:

$$\omega_\theta=\begin{bmatrix} c\psi \\ -s\psi \\ 0 \end{bmatrix}\dot{\theta} \tag{E.6}$$

Similarly, the rotation by angle ψ is given by the third column vector of $R_z(\psi)\{\omega\}$ as:

$$\omega_\psi=\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}\dot{\psi} \tag{E.7}$$

www.manaraa.com

The end-effector angular velocity components in matrix form are:

$$\omega = \begin{bmatrix} s\psi s\theta & c\psi & 0 \\ c\psi s\theta & -s\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{Bmatrix} \tag{E.8}$$

www.manaraa.com

Appendix F: Specifications for the PHANTOM Omni Haptic Device

The Phantom® Omni is a haptic device model developed by SensAble Technologies. It offers six (6) positional DoF as input and three (3) forces DoF output. The specifications for this device are shown in Table F.1

Table F.1 Specifications for the Omni Haptic Device

Model	The PHANTOM Omni Device
Force feedback workspace:	~6.4 W x 4.8 H x 2.8 D in > 160 W x 120 H x 70 D mm
Footprint: Physical area the base of device occupies on the desk	6 5/8 W x 8 D in ~168 W x 203 D mm
Weight (device only):	3 lb 15 oz
Range of motion:	Hand movement pivoting at wrist
Nominal position resolution:	> 450 dpi ~ 0.055 mm
Backdrive friction:	<1 oz (0.26 N)
Maximum exertable force at nominal (orthogonal arms) position:	0.75 lbf. (3.3 N)
Continuous exertable force (24 hrs.)	> 0.2 lbf. (0.88 N)
Stiffness:	X axis > 7.3 lb/in (1.26 N/mm) Y axis > 13.4 lb/in (2.31 N/mm) Z axis > 5.9 lb/in (1.02 N/mm)
Inertia (apparent mass at tip):	~0.101 lbm. (45 g)
Force feedback:	x, y, z (3Dof Output)
Position sensing: [Stylus gimbal]:	x, y, z (digital encoders) [Pitch, roll, yaw (± 5% linearity potentiometers)] (6Dof Input)
Interface:	IEEE-1394 FireWire® port
Supported platforms:	Intel-based PCs
GHOST® SDK compatibility:	No
3D Touch™ SDK compatibility:	Yes
Applications:	Selected Types of Haptic Research and The FreeForm® Concept™ system

www.manaraa.com

Appendix G: Custom Made Sick DT60 Data Acquisition Module

The Sick DT60 is distance sensor that uses a laser diode to produce red light which is a reflected from the target object to generate an analogue signal proportional to the distance from the target. The DT60 sensor has a range of 200mm to 6m and is designed to be used with any target material. According to the documentation provided by the manufacturer, the visible red light is an eye-safe light beam, however, it is highly recommended to avoid direct exposure to the laser light. Power and signal connections to the laser are via a standard M12, 5-pin plug. Accuracy is ±10mm with a typical reproducibility of around 7mm. The output signal is a current varying from 4.0mA to 20.0mA proportional to the measured distance. Before Analog-to-Digital conversion using the 232 SDA12, a high precision resistor must be used to convert to a voltage signal with 0-5 VDC range (See Figure G.1).

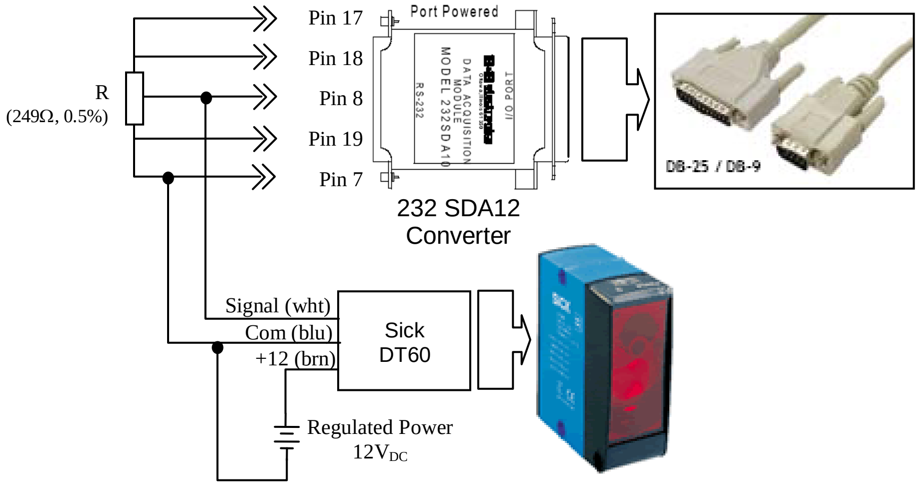


Figure G.1 Custom-made ADC Module for the DT60 Sick Laser Sensor

المنارة للاستشارات
www.manaraa.com

About the Author

Eduardo J. Veras was born on August 9, 1963 in Santiago, Dominican Republic. He received a Bachelor's Degree in Mechanical Engineering from Pontificia Universidad Católica Madre y Maestra University in 1987 and a M.S. in Design and Manufacturing from University of Puerto Rico at Mayaguez in 1992. He started teaching at Polytechnic University of Puerto Rico until he entered the Ph.D. program at the University of South Florida in 2004. Mr. Veras was a teaching assistant in the USF Mechanical Engineering Laboratory II and a research assistant in the Rehabilitation Robotics Center at USF. His responsibilities as RA included development of a haptic controller to drive a VR of the Puma 560, and interfaces for a Spaceball, Barrett Hand, and Sick LRF for MatLab, and a BCI-2000 program to drive a wheelchair-mounted robotic arm. He accepted a position as a faculty member at the Polytechnic University of Puerto Rico, Orlando.

المنارة للاستشارات

www.manaraa.com