

***OUTPUT SYNCHRONIZATION FOR TELEOPERATION OF
ROBOT MANIPULATORS***

by

PATRICK THOMAS MILLER

NOVEMBER 2008

A thesis submitted to the Faculty of the Graduate School of the State University of New York at Buffalo in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Department of Mechanical and Aerospace Engineering
State University of New York at Buffalo
Buffalo, New York 14260

For my Parents...
And their Patience...

Abstract

With the emergence of innumerable unmanned robotic systems, there is a greater push for a robust and stable means of teleoperated control. Teleoperation is extremely useful in two major ways. First, a remotely control robot may be able to enter an environment which is dangerous or inhabitable by a human operator. This scenario is pertains to autonomous vehicle or hazardous waste management applications. Secondly, robot manipulators are capable of detecting and compensating for environmental interactions and making more precise and consistent movements then their human counterparts. This becomes particularly useful robot assisted surgery.

In such remote control applications, it is also important to accurately reflect the remote environment back to the human user. This is typically achieved by adding force and torque sensors on the remote system and transmitting force data back to the operator. This is an ideal form of bilateral control since the sense of touch is significantly more sensitive then the human eye. However, problems arise when transmission delays are introduced into the link between host and remote system.

Transmission delays are an unavoidable part of remote communications which are typically compensated by adding physical or virtual compliance into the system. However, this has the deleterious effect on 'transparent' perception of the remote environment. Thus, it is the responsibility of the control algorithm to compensate for delay induced instability while providing a transparent feel to the human operator. Maintaining the stability of the overall teleoperation scheme at a minimal loss of

performance is the basis of this thesis. The case-study of telemanipulation of the end effector of a wheeled mobile manipulator using the stylus endpoint Phantom Omni haptic device will serve to form our efforts in this thesis. First and foremost the devices are kinematically and dynamically dissimilar. The WMM features considerable redundancy within its articulated chain which creates challenges for its control by a non-redundant Phantom Omni robot. These couple with the remaining issues of stable teleoperation over lossy, band limited, imperfect communication media to make this a challenging task. To this end we will study the application of a bilateral teleoperation framework developed [1] using nonlinear control and energy based stability theory to overcome some of the limitations. Locally an adaptive controller will identify and stabilize the dynamics of the master and slave systems. This local controller will render the dynamics passive to a secondary coupling input. A passive mapping is used to couple the output states of the master and slave systems. This mapping is shown to be insensitive to lossy and delayed communication mediums. A battery of simulations and real time experiments are used to verify the proposed controller. The controller will also be shown to work on a robotic arm and wheeled mobile robot.

Acknowledgment

I would like to express my most sincere gratitude towards my advisor, Dr. Krovi, for his guidance and support as his graduate student in the Department of Mechanical and Aerospace Engineering at the University at Buffalo. His enthusiasm and determination in the field of robotics has been an inspiration in these past two years. Through many trials and tribulations he has provided me with the necessary skills to be successful in today's competitive and diverse world.

Special thanks should be given to my committee members, Dr. John Crassidis and Dr. Tarunraj Singh, for reading my thesis and making a number of helpful suggestions. They have been excellent teachers in my six years at the university. Classes like *Optimal Estimation* and *Applied Nonlinear Control* have provided me with a set of essential tools for my future engineering endeavors.

Other special thanks should go to Dr. Das for coming all the way from Rochester to attend my thesis. His support and insight as my unofficial outside reader was very useful in the final stages of this thesis.

The ARMLab has been my second home for the past two years. I would like to acknowledge my colleagues that have made my stay such a memorable one. Special thanks should go to Chin Pei (CP) Tang and Qiushi Fu for always being there to lend a helping hand. Also, to Leng-Feng Lee, Anand Naik, Kun Yu, Hao Su, Yao Wang, Madusudanan S. Narayana, Srikanth Kannan, Xiabo Zhao and all the past lab members. Finally special thanks should go to Glenn White, whose design and physical prototype has been used extensively in this thesis for experimental validation.

Finally, I would like to thank my family, especially my parents, for their support through my graduate career. I could not have done it without them.

Contents

Abstract.....	iii
Acknowledgment.....	v
Contents.....	vii
List of Figures.....	x
List of Tables.....	xiii
1 Introduction.....	1
1.1 Challenges.....	3
1.2 Teleoperation Variants.....	5
1.3 Our System.....	6
1.4 Research Questions.....	7
1.5 Thesis Organization.....	8
2 Background.....	9
2.1 Skew Symmetry.....	9
2.2 Passivity.....	10
2.3 Homogenous Transformations.....	12
2.4 Euler-Lagrangian Equations of Motion.....	15
2.5 Robust Passivity Control.....	17
3 Dynamic Models.....	21
3.1 RR Manipulator Dynamics.....	21
3.2 Wheeled Mobile Robot Dynamics.....	24

3.2.1	Feasible Motion Dynamics	29
3.2.2	Constraint Analysis.....	31
4	Control	33
4.1	Adaptive Output Synchronization.....	33
4.1.1	Parameter Adaptation.....	38
4.1.2	Hybrid Teleoperation Control.....	38
4.2	Synchronization Control of a RR Manipulator.....	39
4.3	Synchronization Control of a WMR.....	41
4.3.1	WMR Decoupling Matrix.....	41
4.3.2	Zero Dynamics.....	42
4.3.3	Output Synchronization of a WMR	44
5	Implementation Framework.....	45
5.1	Virtual Environment Blocks	45
5.2	Simulation Framework.....	47
5.2.1	RR Simulation code	48
5.2.2	WMR Simulation code	53
5.3	Real time Implementation Framework	54
5.3.1	Network and Hardware Setup.....	55
5.3.2	Simulink Software	57
5.4	Test Cases	60
6	Results.....	63
6.1	RR Results	63
6.1.1	Case 1: Simulation Results	64

6.1.2	Case 1: Real time Results	65
6.1.3	Case 2: Simulation Results	67
6.1.4	Case 2: Real time Results	68
6.2	WMR Results.....	70
6.2.1	Case 3: Simulation Results	70
6.2.2	Case 4: Simulation Results	72
6.2.3	Case 5: Simulation Results	73
7	Conclusions.....	74
7.1	Summary	74
7.2	Future Work.....	76
8	Bibliography	78
	Appendix A.....	81
A.1	Hardware Interface Drivers.....	81

List of Figures

Figure 1-1: Examples of different teleoperation systems a) an exoskeleton developed at Jet Propulsion Labs NASA b) Stanford teleoperation demonstration.	1
Figure 1-2: Our mobile manipulator system as a solid model (a) and physical prototype (b). [16]	6
Figure 2-1: An example of a basic rigid body motion.	12
Figure 3-1: A kinematic schematic of the RR manipulator.	22
Figure 3-2: A kinematic schematic of the WMR from the a) top view and b) wheel view.	25
Figure 4-1: A schematic of bilateral synchronization for teleoperation.	33
Figure 5-1: The elements of a real-time virtual reality scenes: a) The Simulink block for linking real-time code to the VR scene b) a sample of the VRML scene.....	46
Figure 5-2: The Phantom Omni six degree of freedom haptic device. [25]	46
Figure 5-3: The RR simulation Simulink block diagram.....	48
Figure 5-4: The haptic control subsystem.....	49
Figure 5-5: The network simulator subsystem and ProSense network simulator block...	49
Figure 5-6: The slave manipulator dynamics and control subsystem.....	50
Figure 5-7: The haptic and output subsystem.	51
Figure 5-8: The WMR simulation Simulink block diagram.	53
Figure 5-9: An overview of the real time implementation framework.	55
Figure 5-10: A block diagram of the xPC boards and interconnections.....	56
Figure 5-11: The Simulink block diagrams for a) the PC haptic side and b) RR manipulator side.....	58

Figure 5-12: The master and slave UDP communication subsystem with ProSense network simulator	59
Figure 5-13: The slave manipulator control block and hardware/sensor interface.....	59
Figure 5-14: A pictorial representation of the test criteria for teleoperation schemes.....	60
Figure 6-1: A comparison of the simulated master and slave a) Cartesian state and b) Cartesian forces for Case 1.	64
Figure 6-2: The torque profile for the simulated slave manipulator for Case 1.	65
Figure 6-3: A comparison of the real master and slave a) Cartesian states and b) Cartesian forces for Case 1.	66
Figure 6-4: The torque profile for the real slave manipulator implementation for Case 1.	67
Figure 6-5: A comparison of the simulated master and slave a) Cartesian states and b) Cartesian forces for Case 2.	67
Figure 6-6: The real torque profile for the simulated slave manipulator for Case 2.	68
Figure 6-7: A comparison of the real master and slave a) Cartesian states and b) Cartesian forces for Case 2.	69
Figure 6-8: The torque profile for the real slave manipulator implementation for Case 2.	70
Figure 6-9: Simulation results for teleoperation of a WMR without time delay and packet loss.	71
Figure 6-10: The WMR teleoperated with 0.1 constant time delay and 55% random packet loss.....	72

Figure 6-11: The WMR teleoperated under variable time delay (mean = 0.4 s, std = 0.12 s) and 55% packet loss..... 74

List of Tables

Table 3-1: A summary of Kinematic Parameters of the two link planar manipulator.....	22
Table 3-2: A summary of kinematic parameters of the wheeled mobile robot	26
Table 5-1: A summary of technical specifications for the Phantom Omni.....	47
Table 5-2: RR manipulator kinematic and dynamic and haptic model simulation parameters.	50
Table 5-3: A summary of kinematic and dynamic parameters for a virtual box.	52
Table 5-4: Wheeled mobile robot kinematic and dynamic simulation parameters.	54
Table 5-5: A summary of kinematic and dynamic parameters for a virtual box in the WMR environment.	54
Table 5-6: A summary of all possible tests varying delay type and slave device.	61
Table 5-7: A summary of possible tests varying communication reliability and slave device.	61
Table 5-8: The proposed test cases for testing the bilateral teleoperation scheme ability.	62
Table 6-1: A summary of the proposed test cases.	63

1 Introduction

Ever since Raymond Goertz created the first master-slave manipulator for handling radioactive material, researchers have been studying remote manipulation to solve a number of emerging problems. Over the years, there have been a wide variety of proposed solutions for extending the sensing and manipulation abilities of humans.

Teleoperation is the intersection of human control and autonomous robotic operation and improves on both. Quintessentially, it can be defined as the interaction of a distant environment through the aid of an electromechanical subsystem. In terms of robotics, it is extension of the human user's ability to interact and manipulate an environment through the use of a robotic subsystem.



(a)



(b)

Figure 1-1: Examples of different teleoperation systems a) an exoskeleton developed at Jet Propulsion Labs NASA b) Stanford teleoperation demonstration.

Figure 1-1 contains just a glimpse of a few of the research devices created for teleoperation over the years.

The first figure is an exoskeleton designed by Jet Propulsion Laboratories in the early 90's. A two-armed telerobot was designed mimicking the actions of the human operator. Interaction forces were feedback to the glove of the human user as a form of tactile feedback. Figure 1-1(b) depicts a teleoperation framework developed at Stanford involving a 6 degree of freedom Puma robot remotely driven by a Phantom haptic device. This framework permitted user to set a dining room table remotely with nothing more than visual cues and tactile feedback.

In a typical teleoperation setup, a user controls with a master device (virtual manipulator) through the use of a pointing device, typically a joystick or multi-degree of freedom force reflecting (haptic) device. The user's commands are transmitted through a communication medium to the slave device (a distant robotic manipulator), which attempts to execute received commands while interacting within the remote environment. The experienced forces and motions from the slave device-environmental interactions are communicated back and reflected (in a scaled manner) back to the user. The reflected force allows the user to control the slave not only visually but with kinesthetic cues. In the most ideal case, the provided feedback (kinesthetic, auditory or visual) allows a sense of immersion in the distant environment, achieving what is known as telepresence. Unfortunately, imperfection in the communication medium (delay, etc.) can adversely affect this immersivity. Thus teleoperation control methods need to be developed to compensate for such imperfections and still maintain adequate performance. These aspects are formally discussed later in this chapter. However, informally and historically,

the presence of communication delays and instabilities were overcome by introducing compliance and damping in between the master and slave devices, reducing the responsiveness of the system. Thus, while the stability of the system could now be ensured, the quality of the experienced telepresence was compromised. And thus satisfying the competing/conflicting tradeoff between stability and transparency continues to pose challenges to teleoperation to this day.

Despite the challenges, teleoperation offers humans the ability to extend their own reach and senses over various length and timescales in innumerable applications. This technology has given doctors the ability of conducting minimally invasive surgeries or allows researchers to explore deeper into the ocean. Telerobotics can also extend the reach of human users, allows the user to be in a safe environment while the slave manipulator can be in a hazardous one. Telerobotics gives the user a chance to feel the interacting environment, despite the size or location of the remote environment.

For these reasons, modern day researchers continue to devote an extensive amount of effort to understanding to complexities of bilateral control over time delayed networks.

1.1 Challenges

Telerobotic systems are faced with many complex issues stemming from the teleoperation devices themselves, environmental factors, communication delays and control complexities.

First and foremost the master and slave systems are typically complex electromechanical systems individually which are typically optimized for their needs. However, this resulted in requiring kinematically and dynamically dissimilar systems to

interact and work together which can be challenging in idealized well modeled environments and when perfect idealized communication is available.

However, slave systems are often placed in highly unstructured and diverse environments, typically not understood a priori. Failure to compensate for environmental interaction or varying conditions could lead to destabilization.

Significant challenges also arise from the delays introduced by communication between the master and slave machines. Communication type and network loads, computational delays or simply distance between machines are all elements that add to communication latency. On a dynamic network, such as the Internet, these delays can be variable due to continually varying load conditions. These delays introduce phase lag into the control loop creating potential problems.

Last but not least is the contribution and role of the human user. Force feedback becomes transparent to the human sense of touch at sampling rates of 1000 Hz or faster. This sampling rate puts a significant burden on the control algorithm to be fast and efficient. Kinematic or velocity based algorithms tend to be computationally light weight and robust to environmental disturbances. However, since they do not directly command forces and torques, applied and interactive forces are hard to calculate and reflect back. Dynamic algorithms which can compute forces and accelerations become necessary but contribute a significant computational burden due to the nonlinear nature of robotic systems. Model based controllers can provide improved results, but depend significantly on accurate models. Introducing adaptivity to model based controllers to adapt to uncertainty can further increase the computational burden of the slave robot.

1.2 Teleoperation Variants

Over the years, there has been an abundant amount of control literature related to teleoperated Lagrangian systems. An extensive survey of bilateral teleoperation schemes was presented in [2]. Until recently, the wave variable based framework introduced in [3] and [4] has been at the fore front of teleoperation research. Since then there has been a plethora of variations based on the wave variable work [5], [6], [7], [8] and [9], to note a few. Unfortunately, since only velocity and force information is typically exchanged in between master and slave, these controllers would suffer from position drift if an initial position error exists between master and slave. [8] and [9] are an exception to this rule, and directly address the problem of position drift. This work has lead to the output synchronization frame work for Lagrangian Systems presented by Chopra, Spong and Lozano in [10]. This framework successfully incorporates adaptive nonlinear control, and drift less stable teleoperation without the use of wave variables. In [1] the stability of this framework for use in teleoperation was proven through the extensive use of Lyapunov function analysis.

We will adapt this framework and study its applicability to safely synchronize two heterogeneous Lagrangian systems for teleoperation. A wheeled mobile robot with nonholonomic constraints are considered in our implementation. A non-adaptive bilateral teleoperation scheme for wheeled mobile robots is also presented in [11]. The dynamic control of a mobile robot with constraints has been a widely research topic. [12], [13], [14] and [15] have provided significant insight into the control issues associated with nonholonomic constraints. Of these references [12] provides a unified framework for input/output control of nonholonomic systems using state transformations. When using

this framework, care should be taken to assess and analyze the stability of any zero dynamics are presented in [15].

1.3 Our System

All of the experimental results of this thesis were conducted on a custom built wheeled mobile manipulator (WMM) as in Figure 1-2.

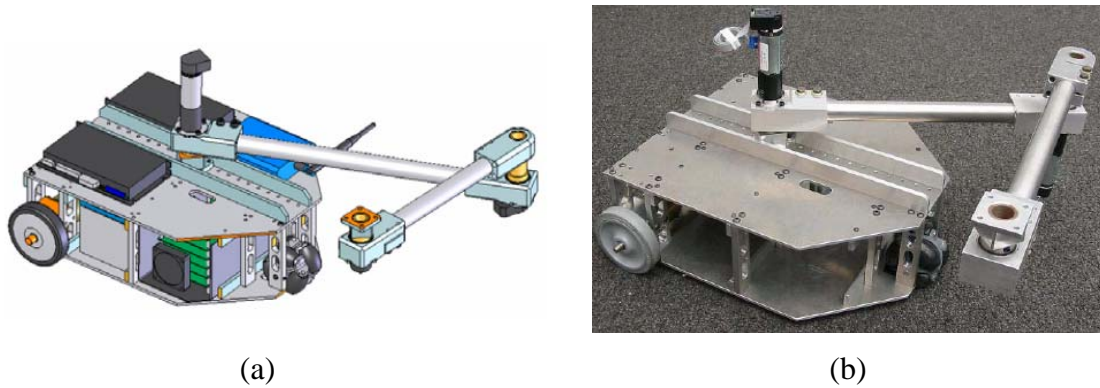


Figure 1-2: Our mobile manipulator system as a solid model (a) and physical prototype (b). [16]

The system is a differentially driven mobile base with a planar two link revolute arm rigidly attached. The base is driven by a rear differentially driven disk wheel pair with an omni-directional caster front wheel. Two high torque motors provide the low speed, high torque necessary to move the entire robot. A differentially driven system was chosen for its superior robustness over caster wheels to terrain irregularities. The omni-directional front caster (a passive Swedish wheel-type caster), was preferable over a conventional caster since it does not reduce the overall maneuverability of the robot. A manipulator arm is mounted on the top plate of the base wheeled robot. The mounting point is along the center of the robot conveniently placed away from the differentially driven wheel axis. The arm consists of a two link serial chain with powered revolute joints. Each wheel and joint is equipped with a digital optical encoder for precise angle

sensing. The axis of rotation for each joint is perpendicular to the base platform and ground making the overall system a planar robot. An onboard embedded Pentium class computer handles all of the sensor interfacing and control. A 10/100 Ethernet embedded controller acts the communication medium between the WMM and master machine.

A standard Pentium class laptop running Matlab will act as the master machine. A 3 degree of freedom Omni haptic device (from Sensable) will provide the position input and force feedback. This computer will communication over Ethernet to a dedicated router connected to the slave machine. The master and slave devices have the ability to connect wirelessly to router for simple wireless implementation or directly wired for reduced latency.

1.4 Research Questions

Given two complex robotic systems, such as a planar robotic arm or wheeled mobile robot, can a bilateral teleoperation scheme be develop which satisfies the following criteria:

- Primarily, the control scheme must be stable over a time delayed and lossy communication medium such as the Internet.
- This control scheme must also be insensitive to any passive external forces applied to the slave robot system.
- The Cartesian states of the master and slave systems should be kinematically coupled in such a way to prevent position drift in the presence of initial offset errors or environmental interactions.
- A transparent and immersive environment should be provided to the human operator to provide a sufficiently rich haptic experience.

- Finally, the model parameters of the slave manipulator should be adapted to guarantee consistently good tracking results over extended periods of time.
- The control algorithm should be verified through simulation and a battery of experiments. These tests should display the robustness of the controller to network latency and packet loss in free space and hard contact scenarios.

The goal of this thesis is to answer these questions.

1.5 Thesis Organization

The remainder of this thesis is organized in the following fashion.

In Chapter 2, a brief overview of the mathematics required for the thesis are given.

In Chapter 3, the mathematical models for each of the proposed robotic manipulators are derived. Dynamics are derived using the Euler-Lagrangian formulation. A reduced set of feasible motion dynamics will also be found if the robotic system contains any kinematic constraints.

In Chapter 4, the proposed bilateral teleoperation scheme is introduced and its stability analyzed. The scheme is adapted to the planar manipulator and wheeled robot slave systems.

In Chapter 5, the virtual reality display and implementation frame work is discussed. Matlab/Simulink code and block diagrams are provided as needed.

Chapter 6 will contain the simulation and experimental test scenarios. The results of each test are discussed and concluded upon.

Chapter 7 will contain a summary of the overall performance and robustness of the proposed teleoperation scheme. Finally, our work is concluded and future research direction are presented.

2 Background

The mathematical and control theory required for this thesis are outlined in this section. First, the concept of skew symmetry and passivity in dynamical systems is introduced. A homogenous matrix based Euler-Lagrangian modeling approach is outlined which insures passivity is conserved. Finally, the control method of robust time based passivity control is introduced.

2.1 Skew Symmetry

In this thesis, skew symmetric matrices are a common occurrence when analyzing the system dynamics and kinematics. From linear algebra we know a matrix is skew symmetric if and only if for all real vectors x the following property holds.

$$x^T \mathbf{A}x = 0$$

Skew symmetry is an inherent property of the equations of motion for Euler-Lagrangian dynamic systems. When analyzing the stability of these systems, the skew symmetric property of the dynamics (and their derivatives) can be taken advantage of. The skew symmetric operator is also useful when analyzing the angular Jacobian of homogenous coordinate transformations. The skew symmetric operator $[\bullet]_x$ converts a vector into a skew symmetric matrix. For example:

$$[a]_x = \begin{bmatrix} 0 & a_3 & -a_2 \\ -a_3 & 0 & -a_1 \\ a_2 & a_1 & 0 \end{bmatrix}, \text{ where } a = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

This operator is commonly used to when calculating the cross product between two vectors. However, the skew symmetric operator (and reverse skew symmetric operator)

can also be useful when analyzing the angular Jacobian of a rigid body transformation. A typical angular Jacobian from frame {A} to {B} might look like

$$\mathbf{J}_{\omega_{AB}} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \dot{q}, \text{ where } \dot{q} = [\dot{\theta}_1 \quad \dot{\theta}_2 \quad \dot{\theta}_3]^T$$

Applying the reverse skew symmetric operator, the angular Jacobian can be reduced to the following vector.

$$J_{\omega_{AB}} = [\mathbf{J}_{\omega_{AB}}]_{\times}^{-1} = [1 \quad 0 \quad 1] \dot{q}$$

2.2 Passivity

The concept of energy passivity is very beneficial when creating large and potentially complex control systems from small, simpler subsystems. When analyzing a system, it is common to use energy like function called a Lyapunov function to conclude on the system's stability. Since all physical systems are driven by the laws of energy conservation

$$\frac{\partial}{\partial t} [\text{Stored Energy}] = [\text{External Power Input}] + [\text{Internal Power Generation}]$$

it can be expected that the energy functions of combined systems are additive. The resulting combined Lyapunov function can then be analyzed to conclude on the stability of the new larger system. Passivity theory formalizes these observations and provides us with a set of rules to follow when creating feedback and parallel combinations of subsystems. [17]

The conservation of energy equation can be written generally in the following vector form.

$$\dot{V}(t) = y^T u - g(t) \quad (2.1)$$

where y is output effort, u is the input flow of the system, $g(t)$ is the power generated by system internally and $V(t)$ is the total energy stored in the system. The general system is said to be passive if $V(t)$ is positive (lower bounded) and the function $g(t) \geq 0$. [17]

Developing Lyapunov functions in the manner outlined by equation (2.1) can simplify our analysis of combined systems. When dealing with parallel and feedback combinations of passive subsystems (as defined by equation (2.1)), it can be found that the final combined systems will follow the same form as equation (2.1). For example when combining two systems in parallel, $u = u_1 = u_2$, and we get to following input output relationship.

$$y^T u = (y_1 + y_2)^T u = y_1^T u + y_2^T u = y_1^T u_1 + y_2^T u_2$$

The total energy of the system can be found to be the following equation.

$$\dot{V} = (\dot{V}_1 + \dot{V}_2) = y_1^T u_1 - g_1(t) + y_2^T u_2 - g_2(t)$$

$$\dot{V} = y^T u - (g_1(t) + g_2(t))$$

The overall stability of the combined system can now be concluded upon. A similar result can be shown for the feedback case where $u_2 = y_1$.

2.3 Homogenous Transformations

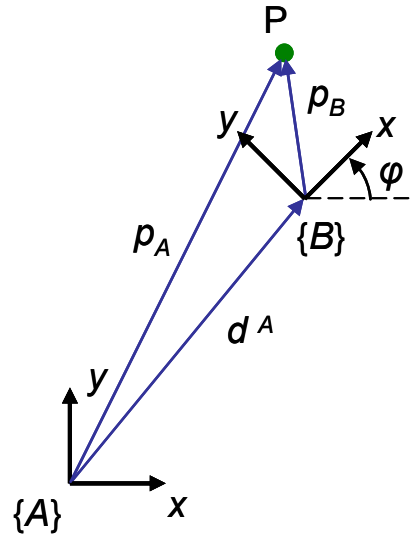


Figure 2-1: An example of a basic rigid body motion.

In Figure 2-1, a single point, P is expressed in the moving frame {B} by the vector p_B . Expressing the point P, defined in frame {B}, into a secondary frame {A}, p_A can be expressed as

$$p_A = {}_A \mathbf{R}^B p_B + d^A$$

$${}_A \mathbf{R}^B = \begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad d^A = \begin{bmatrix} d_x^A \\ d_y^A \\ d_z^A \end{bmatrix}$$

where ${}_A \mathbf{R}^B$ is a 3x3 rotation matrix defining the rotation of frame {B} with respect to frame {A} and d^A is the three dimensional displacement from frame {A} to frame {B}. A homogenous transformation describes this same relationship in a compact matrix form.

$${}_A \mathbf{T}^B = \begin{bmatrix} {}_A \mathbf{R}^B & d^A \\ \mathbf{0}_{1 \times 3} & 1 \end{bmatrix}$$

A planar transformation (such as Figure 2-1), it can be assumed that all rotations are considered in the z plane. With this assumption the row and column which correspond to the z axis can be taken out. The reduced form of the planar homogenous transform is given as

$${}^A\mathbf{T}^B = \begin{bmatrix} {}^A\mathbf{R}_{2 \times 2}^B & d_{2 \times 1}^A \\ \mathbf{0}_{1 \times 2} & 1_{1 \times 1} \end{bmatrix}$$

$${}^A\mathbf{R}_{2 \times 2}^B = \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix}, \quad d_{2 \times 1}^A = \begin{bmatrix} d_x^A \\ d_y^A \end{bmatrix}$$

For completeness the size of the each of the matrix elements was included in a lower right hand subscript. For the remainder of this thesis all homogenous coordinates will be planar transformations and the size subscript left out for brevity.

If a third frame, frame {C}, is introduced after frame {B} the final transformation link frame {A} to frame {C} is simply the multiplication of each of the intermediate relative homogenous transformations.

$${}^A\mathbf{T}^C = {}^A\mathbf{T}^B {}^B\mathbf{T}^C = \begin{bmatrix} {}^A\mathbf{R}^B & d^A \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} {}^B\mathbf{R}^C & d^B \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} {}^A\mathbf{R}^B {}^B\mathbf{R}^C & {}^A\mathbf{R}^B d^B + d^A \\ 0 & 1 \end{bmatrix}$$

$${}^A\mathbf{T}^C = \begin{bmatrix} {}^A\mathbf{R}^C & {}^A d^C \\ 0 & 1 \end{bmatrix}$$

Analyzing the final transformation ${}^A\mathbf{T}^C$, we see that it contains a planar rotation, ${}^A\mathbf{R}^C$ and planar displacement between frame {A} and frame {C}.

In robotic applications it is desirable to have a mapping between the joint space and the linear and angular velocities of a desired frame, {i}, with respect to an inertial frame, {0}. A homogenous matrix defined as a function of our joint variables (q) is represented by

$${}^0\mathbf{T}^i(q) = \begin{bmatrix} {}^0\mathbf{R}^i(q) & {}^0d^i(q) \\ 0 & 1 \end{bmatrix}$$

where $q = [q_1, \dots, q_n]$ are the joint variables corresponding to n-links. As the robot moves through space, the displacement and rotation between frames are a function of time. It is ideal to define the mapping between linear and angular velocity of the frame $\{i\}$ in the inertial frame $\{0\}$ as a function of the joint variables. These matrices are called the linear and angular Jacobian defined by

$$v_i = \mathbf{J}_{v_i}(q)\dot{q} \text{ and } \omega_i = \mathbf{J}_{\omega_i}(q)\dot{q}$$

[18] tells us the linear mapping between linear velocity and joint velocity is simply the derivative of the linear displacement vector.

$$v_i = \mathbf{J}_{v_i}(q)\dot{q} = {}^0\dot{d}^i(q)$$

Furthermore, the mapping linking the angular velocity to the joint space velocities is slightly more complicated. The angular rotation is related to the joint velocities through the following skew symmetric matrix.

$$\mathbf{S}(\omega_i) = {}^0\dot{\mathbf{R}}^i(q) \cdot [{}^0\mathbf{R}^i(q)]^T = \begin{bmatrix} 0 & \omega_z(q) \\ -\omega_z(q) & 0 \end{bmatrix}$$

In the planar case, where all linear movement is restricted to the x,y plane, all angular velocities are restricted to the perpendicular z plane. Using the skew symmetric operator defined in section 2.1, the skew symmetric matrix can be converted into a vector, representing $\mathbf{J}_{\omega_i}(q)\dot{q}$.

2.4 Euler-Lagrangian Equations of Motion

We review the processes of constructing the equations of motion for n-link rigid body systems. This formulation is based on the Lagrangian, a quantity defined as the difference between kinetic and potential energy,

$$L \equiv K_{energy} - P_{energy}$$

After defining the Lagrangian the system, the final equations of motion can be computed

$$\frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}} \right) - \frac{\partial L}{\partial q} = \frac{\partial \Pi}{\partial \dot{q}} - \frac{\partial \Delta}{\partial \dot{q}}$$

where, L represents the Lagrangian as defined above, q is a joint vector of the n-link system, Π is a vector of all input power sources (i.e. forces and torques) and Δ is a vector of all dissipative power sources (i.e. friction). The total kinetic energy of an n-link rigid body system can be defined by

$$K_{energy} = \frac{1}{2} \dot{q}^T \left\{ \sum_{i=1}^n \left[m_i \mathbf{J}_{v_i}(q)^T \mathbf{J}_{v_i}(q) + \mathbf{J}_{\omega_i}(q)^T \mathbf{R}_i(q) I_i \mathbf{R}_i(q)^T \mathbf{J}_{\omega_i}(q) \right] \right\} \dot{q}$$

where m_i is the mass of link i , I_i is the inertia matrix of link i , \mathbf{R}_i is the rotation matrix from inertial frame to the link frame of reference and $\mathbf{J}_{v_i}(q)$ and $\mathbf{J}_{\omega_i}(q)$ represent the linear and angular velocity mapping from the inertial frame to the location of mass i . For the planar case, where all rotations are restricted to the z plane, the above equation reduces to

$$K_{energy} = \frac{1}{2} \dot{q}^T \left\{ \sum_{i=1}^n \left[m_i \mathbf{J}_{v_i}(q)^T \mathbf{J}_{v_i}(q) + \mathbf{J}_{\omega_i}(q)^T I_i \mathbf{J}_{\omega_i}(q) \right] \right\} \dot{q}$$

The final kinetic energy can be succinctly written as:

$$K_{energy} = \frac{1}{2} \dot{q}^T \mathbf{M}(q) \dot{q} \quad (2.2)$$

where the matrix \mathbf{M} represents the inertial matrix of the n-link system. The Coriolis matrix (\mathbf{C}) can be calculated through an iterative relationship defined bellow.

$$C_{kj} = \sum_{i=1}^n c_{ijk}(q) \dot{q}_i$$

where, C_{kj} is the kth row and the jth column element of the Coriolis matrix, n is the length of q , and $c_{ijk}(q)$ is the Christoffel symbol of the first kind defined by:

$$c_{ijk}(q) = \frac{1}{2} \left\{ \frac{\partial m_{kj}}{\partial q_i} + \frac{\partial m_{ki}}{\partial q_j} - \frac{\partial m_{ij}}{\partial q_k} \right\}$$

where m_{ki} represents kth row and ith column element of the inertia matrix \mathbf{M} .

The energy input (Π) into the system is defined as a vector of the input energies provided by each of the actuators. This input relationship can be summarized as

$$\Pi = [\tau_1 \dot{\theta}_1, \dots, \tau_n \dot{\theta}_n]^T$$

The dissipative vector (Δ) is the sum of acting forces which take energy away from the system. This vector contains the various friction components of the system.

$$\Delta = [c_1(q, \dot{q}), \dots, c_n(q, \dot{q})]^T$$

The energy input and dissipative vectors are differentiated with respect to the joint vectors and added to the final result. Assembling the required matrices, the dynamics can be stated as

$$\mathbf{M}(q)_{n \times n} \ddot{q} + \mathbf{C}(q, \dot{q})_{n \times n} \dot{q} + \mathbf{D}(q, \dot{q})_{n \times n} q = \tau$$

$$\tau = [\tau_1, \dots, \tau_n]^T, \quad q = [q_1, \dots, q_n]^T$$

where $\mathbf{M}(q)_{n \times n}$ is an inertial matrix, $\mathbf{C}(q, \dot{q})_{n \times n}$ is a combination of the centrifugal and Coriolis terms, $\mathbf{D}(q, \dot{q})_{n \times n}$ is matrix of dissipative terms. The dynamics formed by this procedure have three important properties.

Property 1: $\mathbf{M}(q)_{n \times n}$ is a symmetric and positive-definite matrix

Property 2: There is a parametric vector θ on the dynamics which satisfies

$$\mathbf{M}(q)_{n \times n} \ddot{q} + \mathbf{C}(q, \dot{q})_{n \times n} \dot{q} + \mathbf{D}(q, \dot{q})_{n \times n} \dot{q} = \mathbf{Y}(\ddot{q}, \dot{q}, q)_I a_I + \mathbf{Y}(\ddot{q}, \dot{q}, q)_D a_D = \mathbf{Y}(\ddot{q}, \dot{q}, q) a$$

Property 3: The matrix $\dot{\mathbf{M}}(q)_{n \times n} - 2\mathbf{C}(q, \dot{q})_{n \times n}$ is skew-symmetric

2.5 Robust Passivity Control

In this section, robust passivity control for Lagrangian systems is reviewed. Many early nonlinear control techniques aim at precisely cancelling out the nonlinearities of the plant system in an effort to reduce the nonlinear system to a simpler linear system. Popular linear control techniques could then be applied. Passivity based controllers take advantage of the inherent (passive) structure of rigid body systems to achieve the desired behavior. Passivity based nonlinear controllers have been a major area of research for several decades. The formulation presented here is from [17], [10] and [18].

We will begin with the standard representation of an n-link robotic manipulator. For simplicity the matrix size subscripts are dropped for brevity.

$$\mathbf{M}(q) \ddot{q} + \mathbf{C}(q, \dot{q}) \dot{q} + \mathbf{D}(q, \dot{q}) \dot{q} = \tau$$

Our goal is to drive the manipulator end effector through a desired joint trajectory. The desired position, velocity and acceleration terms are denoted by q_d , \dot{q}_d and \ddot{q}_d . The

control law augments the existing state dynamics with the reference state vector \dot{q}_r given by:

$$\dot{q}_r = \dot{q}_d - \lambda \tilde{q} \quad \text{and} \quad \ddot{q}_r = \ddot{q}_d - \lambda \dot{\tilde{q}}$$

where, $\dot{\tilde{q}} = \dot{q} - \dot{q}_d$, $\tilde{q} = q - q_d$ and λ is a diagonal positive constant matrix. The manipulator control law is given by

$$\tau = \hat{\mathbf{M}}(q)\ddot{q}_r + [\hat{\mathbf{C}}(q, \dot{q}) + \hat{\mathbf{D}}(q, \dot{q})]\dot{q}_r + u$$

where $\langle \hat{\cdot} \rangle$ represents the estimated model parameter and u is a secondary control input.

Using *property 2* of the Lagrangian dynamics, the control law can be rewritten in a compact form

$$\tau = \mathbf{Y}(\ddot{q}_r, \dot{q}_r, q)\hat{a} + u$$

Plugging into the equations of motion,

$$\mathbf{M}(q)(\ddot{q} - \ddot{q}_r) + [\mathbf{C}(q, \dot{q}) + \mathbf{D}(q, \dot{q})](\dot{q} - \dot{q}_r) = \mathbf{Y}(\ddot{q}, \dot{q}, q)\tilde{a} + u$$

which can be succinctly written as:

$$\mathbf{M}(q)s + [\mathbf{C}(q, \dot{q}) + \mathbf{D}(q, \dot{q})]s = \mathbf{Y}(\ddot{q}, \dot{q}, q)\tilde{a} + u$$

where, $s = \dot{q} - \dot{q}_r = \dot{\tilde{q}} + \lambda \tilde{q}$, \tilde{a} is the estimated parameter error given by $\hat{a} - a$. The resulting closed loop equations can be rearranged into a state space form

$$\dot{x} = \mathbf{f}(x) + \mathbf{g}(x)u, \quad x = [q \quad s]^T$$

where,

$$\mathbf{f}(x) = \begin{pmatrix} s + \dot{q}_r \\ \mathbf{M}(q)^{-1} \{ \mathbf{Y}(\ddot{q}_r, \dot{q}_r, q)\tilde{a} - [\mathbf{C}(q, \dot{q}) + \mathbf{D}(q, \dot{q})]s \} \end{pmatrix}; \quad \mathbf{g}(x) = \begin{pmatrix} 0 \\ \mathbf{M}(q)^{-1} \end{pmatrix}$$

If the model parameters are perfectly known the above system is passive with respect to the control input u and output s using the storage function

$$V(x) = s^T \mathbf{M} s$$

In the case of uncertain dynamic parameters, this passive property can be maintained by adding an adaptation law for the model parameters.

The following control input was suggested by [17]

$$u = -\mathbf{K}s + \bar{u}$$

where \mathbf{K} is a constant positive diagonal matrix. An adaptation law must be created to insure global stability of the controller; a Lyapunov function is considered

$$V = \frac{1}{2} (s^T \mathbf{M}(q) s + \tilde{a}^T \Gamma \tilde{a}) + \tilde{q}^T \mathbf{P} \tilde{q}$$

where \mathbf{P} is a positive definite matrix that is yet to be determined. Differentiating the function with respect to time

$$\dot{V} = \frac{1}{2} s^T \dot{\mathbf{M}}(q) s + s^T \mathbf{M}(q) \dot{s} + 2\tilde{q}^T \mathbf{P} \dot{\tilde{q}} + \tilde{a}^T \Gamma \dot{\tilde{a}}$$

Plugging in the closed loop dynamics for $\mathbf{M}(q) \dot{s}$

$$\dot{V} = \frac{1}{2} s^T \dot{\mathbf{M}}(q) s + s^T [\mathbf{Y}(\ddot{q}, \dot{q}, q) \tilde{a} - [\mathbf{C}(q, \dot{q}) + \mathbf{D}(q, \dot{q}) + \mathbf{K}] s + \bar{u}] + 2\tilde{q}^T \mathbf{P} \dot{\tilde{q}} + \tilde{a}^T \Gamma \dot{\tilde{a}}$$

Taking out the Coriolis matrix and using the skew symmetric property defined above

$$\dot{V} = \frac{1}{2} s^T [\dot{\mathbf{M}}(q) - 2\mathbf{C}(q, \dot{q})] s + s^T [\mathbf{Y}(\ddot{q}, \dot{q}, q) \tilde{a} - \mathbf{D}(q, \dot{q}) s - \mathbf{K} s + \bar{u}] + 2\tilde{q}^T \mathbf{P} \dot{\tilde{q}} + \tilde{a}^T \Gamma \dot{\tilde{a}}$$

$$\dot{V} = s^T \mathbf{Y}(\ddot{q}, \dot{q}, q) \tilde{a} - s^T \mathbf{K}_D s + s^T \bar{u} + 2\tilde{q}^T \mathbf{P} \dot{\tilde{q}} + \tilde{a}^T \Gamma \dot{\tilde{a}}$$

where $\mathbf{K}_D = \mathbf{D}(q, \dot{q}) + \mathbf{K}$. Plugging in for s , in the gain terms reveals

$$\dot{V} = \tilde{a}^T \left[\Gamma \dot{\tilde{a}} + \mathbf{Y}(\ddot{q}, \dot{q}, q) s^T \right] - (\dot{\tilde{q}} + \lambda \tilde{q})^T \mathbf{K}_D (\dot{\tilde{q}} + \lambda \tilde{q}) + 2\tilde{q}^T \mathbf{P} \dot{\tilde{q}} + s^T \bar{u}$$

If we assume the model parameters vary slowly ($\dot{a} \sim 0$) then $\dot{\tilde{a}}$ will reduce to the change in the estimated parameters ($\dot{\hat{a}}$). The constant matrix \mathbf{P} is set to $\lambda \mathbf{K}_D$ as suggest by [19]. Plugging in and reassembling the equation gives:

$$\dot{V} = \tilde{a}^T \left[\Gamma \dot{\hat{a}} + \mathbf{Y}(\ddot{q}, \dot{q}, q) s^T \right] - \tilde{q}^T \lambda^T \mathbf{K}_D \lambda \tilde{q} - \dot{\tilde{q}}^T \mathbf{K}_D \dot{\tilde{q}} + s^T \bar{u}$$

The left side of the equation represents the estimate error. By properly choosing the parameter update law to be $\dot{\hat{a}} = \Gamma^{-1} \mathbf{Y}(\ddot{q}, \dot{q}, q) s^T$, the parameter error can be cancelled out.

$$\dot{V} = -(\tilde{q}^T \lambda^T \mathbf{K}_D \lambda \tilde{q} + \dot{\tilde{q}}^T \mathbf{K}_D \dot{\tilde{q}}) + s^T \bar{u}$$

The final equation proves the dynamic system is passive with respect to the secondary control input (\bar{u}) and output (s). It is important to note that the inclusion of the $-\mathbf{K}s$ term in the control input is not necessary to show the passive nature of the proposed control algorithm. However, $-\mathbf{K}s$ in conjunction the inherent system damping $\mathbf{D}(q, \dot{q})$ allows the closed loop system to represent a globally stable error dynamic with e ($e = [\tilde{q}, \dot{\tilde{q}}]$) being the origin. A formal proof of this can be found in [19]

It is interesting to note that inherent damping in the physical system ($\mathbf{D}(q, \dot{q})$) is added onto the damping induced by our choice of a control input increasing overall stability. This fortunate byproduct will allow us to tune the control damping lower then expected while still maintaining stability of the overall system.

3 Dynamic Models

In this section, the Euler Lagrangian models for the set of slave devices used in our teleoperation experiments are derived. These slave systems are a planar revolute pair (RR) robot manipulator and a wheeled mobile robot (WMR). The kinematic models for each system are found by defining the necessary coordinate frames and using homogenous transformations to describe the intermediate motions. From the kinematic models, a dynamic model for each system is developed using the methodology described in section 2.4. In the case of the wheeled mobile robot, physical constraints are added into the dynamic model giving the final feasible motion dynamic equations of motion.

3.1 RR Manipulator Dynamics

The first slave system is the RR robotic manipulator. The actuation space and joint space for the RR manipulator is given by the joint torque ($\tau = [\tau_1 \quad \tau_2]^T$) and joint angle ($\theta = [\theta_1 \quad \theta_2]^T$) of each link. Figure 3-1 outlines the kinematic parameters and necessary coordinate frames for the manipulator.

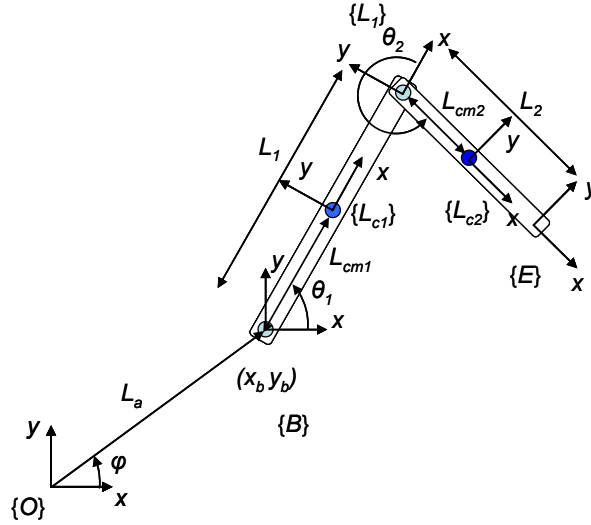


Figure 3-1: A kinematic schematic of the RR manipulator.

Variable	Description	Units
(x_b, y_b)	Cartesian location of the end effector base	m
L_1	Length of link 1	m
L_2	Length of link 2	m
θ_1	Relative angle of revolute joint 1	radians
θ_2	Relative angle of revolute joint 2	radians

Table 3-1: A summary of Kinematic Parameters of the two link planar manipulator

The manipulator has a two link planar arm connected by two revolute joints. A geared revolute motor is attached at the joint between the ground point and link 1 as well as link 1 and link 2. Each of the joints is perpendicular to the x y plane. The base frame of the RR manipulator, $\{B\}$ is located at an arbitrary constant location (x_b, y_b) away from the inertial frame $\{O\}$. A coordinate frame is attached to the center of mass locations of link 1, $\{L_{cm1}\}$ and link 2, $\{L_{cm2}\}$. Finally, coordinate frames are attached to end of link 1, $\{L_1\}$ and the end effector position, $\{E\}$. The joint angle of each revolute joint and is

defined in relative coordinates. The relevant kinematic parameters are summarized in Table 3-1.

After defining the desired frames of reference, relative homogenous transformations are used to describe the relative motion between subsequent frames. Analyzing the kinematic schematic reveals the following relative transformations.

$$\begin{aligned} {}_O\mathbf{T}^B &= \begin{bmatrix} 1 & 0 & x_b \\ 0 & 1 & y_b \\ 0 & 0 & 1 \end{bmatrix}, {}_B\mathbf{T}^{L_{cm1}} = \begin{bmatrix} c_1 & -s_1 & L_{cm1}c_1 \\ s_1 & c_1 & L_{cm1}s_1 \\ 0 & 0 & 1 \end{bmatrix}, \\ {}_B\mathbf{T}^{L_1} &= \begin{bmatrix} c_1 & -s_1 & L_1c_1 \\ s_1 & c_1 & L_1s_1 \\ 0 & 0 & 1 \end{bmatrix}, {}_{L_1}\mathbf{T}^{L_{cm2}} = \begin{bmatrix} c_2 & -s_2 & L_{cm2}c_2 \\ s_2 & c_2 & L_{cm2}s_2 \\ 0 & 0 & 1 \end{bmatrix}, \\ {}_{L_1}\mathbf{T}^E &= \begin{bmatrix} c_2 & -s_2 & L_2c_2 \\ s_2 & c_2 & L_2s_2 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

where, $c_1 = \cos(\theta_1)$, $s_1 = \sin(\theta_1)$, $c_2 = \cos(\theta_2)$, $s_2 = \sin(\theta_2)$. Combining the relative homogenous transformations, gives the final set of transformations

$$\begin{aligned} {}_O\mathbf{T}^{L_{cm1}} &= {}_O\mathbf{T}^B {}_B\mathbf{T}^{L_{cm1}} = \begin{bmatrix} c_1 & -s_1 & L_{cm1}c_1 + x_b \\ s_1 & c_1 & L_{cm1}s_1 + y_b \\ 0 & 0 & 1 \end{bmatrix} \\ {}_O\mathbf{T}^{L_{cm2}} &= {}_O\mathbf{T}^B {}_B\mathbf{T}^{L_1} {}_{L_1}\mathbf{T}^{L_{cm2}} = \begin{bmatrix} c_{12} & -s_{12} & L_{cm2}c_{12} + L_1c_1 + x_b \\ s_{12} & c_{12} & L_{cm2}s_{12} + L_1s_1 + y_b \\ 0 & 0 & 1 \end{bmatrix} \\ {}_O\mathbf{T}^E &= {}_O\mathbf{T}^B {}_B\mathbf{T}^{L_1} {}_{L_1}\mathbf{T}^E = \begin{bmatrix} c_{12} & -s_{12} & L_2c_{12} + L_1c_1 + x_b \\ s_{12} & c_{12} & L_2s_{12} + L_1s_1 + y_b \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

where $c_{12} = \cos(\theta_1 + \theta_2)$, $s_{12} = \sin(\theta_1 + \theta_2)$. After computing the desired transformations, the desired linear and angular Jacobian mappings can be extracted

$$\begin{aligned} \mathbf{J}_{v_{Lcm1}} &= \begin{bmatrix} -L_{cm1}s_1 & 0 \\ L_{cm1}c_1 & 0 \end{bmatrix}, & \mathbf{J}_{\omega_{Lcm1}} &= [1 \quad 0] \\ \mathbf{J}_{v_{Lcm2}} &= \begin{bmatrix} -L_{cm2}s_{12} - L_1s_1 & -L_{cm2}s_{12} \\ L_{cm2}c_{12} + L_1c_1 & L_{cm2}c_{12} \end{bmatrix}, & \mathbf{J}_{\omega_{Lcm2}} &= [1 \quad 1] \\ \mathbf{J}_{v_E} &= \begin{bmatrix} -L_2s_{12} - L_1s_1 & -L_2s_{12} \\ L_2c_{12} + L_1c_1 & L_2c_{12} \end{bmatrix}, & \mathbf{J}_{\omega_E} &= [1 \quad 1] \end{aligned}$$

The input power is the product of joint torques and angular velocities

$$\Pi = [\tau_1 \dot{\theta}_1 \quad \tau_2 \dot{\theta}_2]^T$$

Each joint is assumed to have a dissipative friction. A viscous friction model is used capture the energy loss from the system

$$\Delta = [c_{v_1} \dot{\theta}_1 \quad c_{v_2} \dot{\theta}_2]^T$$

After taking the necessary derivatives and collecting terms, and using the procedure developed in section 2.4, the dynamic equations are given by:

$$\mathbf{M}(q)\ddot{q} + \mathbf{C}(q, \dot{q})\dot{q} + \mathbf{D}(q, \dot{q})\dot{q} = \tau$$

$$\mathbf{M} = \begin{bmatrix} a_1 + a_2 + 2a_3c_2 + a_4 & a_3c_2 + a_4 \\ a_3c_2 + a_4 & a_4 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} -a_3s_2\dot{\theta}_2 & -a_3s_2(\dot{\theta}_1 + \dot{\theta}_2) \\ a_3s_2\dot{\theta}_1 & 0 \end{bmatrix}$$

$$\mathbf{D} = \begin{bmatrix} c_{v_1} & 0 \\ 0 & c_{v_2} \end{bmatrix}$$

$$\begin{aligned} a_1 &= m_1L_{cm1}^2 & a_3 &= m_2L_1L_{cm2} \\ a_2 &= m_2L_1^2 + I_1 & a_4 &= m_2L_{cm2}^2 + I_2 \end{aligned}$$

3.2 Wheeled Mobile Robot Dynamics

The next slave system is the WMR robotic manipulator. The actuation space and joint space for the system is given by the joint torque ($\tau = [\tau_R \quad \tau_L]^T$) and joint angle

($\theta = [\theta_R \ \theta_L]^T$) of each drive wheel. For the derivation of the dynamics of a WMR, an extended coordinate set is introduced.

$$q = [x_b \ y_b \ \phi \ \theta_R \ \theta_L]^T$$

Figure 3-2 outlines the kinematic parameters and necessary coordinate frames for the robot.

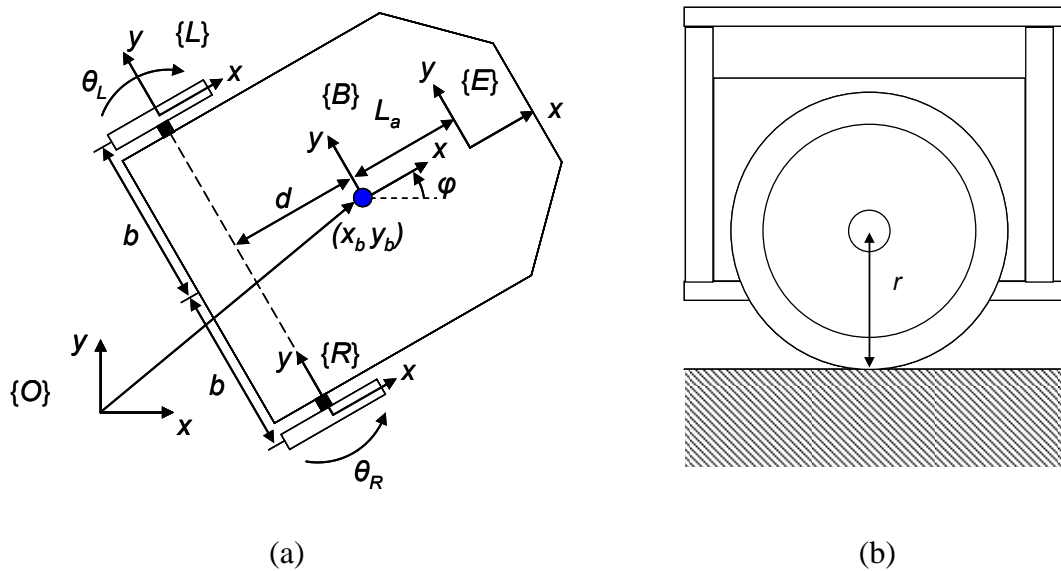


Figure 3-2: A kinematic schematic of the WMR from the a) top view and b) wheel view.

The wheeled robot is a differently driven base with two collinear drive wheels. A motor is attached to each of the wheels with the positive rotations defined. The wheels are setup parallel to the x y plane so that only planar movement is allowed. A base frame, {B} is attached to the center of mass of the mobile base and is located at point (x_b, y_b) with respect to the inertial frame, {O}. A coordinate frame is also attached on the left and right wheels with the axis of rotation collinear to the body fixed y axis. A look ahead point (and coordinate frame) is arbitrarily defined a distance L_a from the body fixed frame along the body fixed x axis. It is assumed the center of mass, wheel frames and

look ahead frames are all collinear on the z plane. The relevant kinematic parameters for the WMR are summarized in Table 3-2.

Variable	Description	Units
(x_b, y_b)	Cartesian location of the end effector base	m
b	Distance of half the axle length along the body fixed y axis	m
d	Distance from the axle to the center of mass along the body fixed x axis	m
r	Radius of the drive wheels	m
L_a	Distance from the center of mass to the look ahead point along the body fixed x axis.	m
θ_1	Relative angle of revolute joint 1	radians
θ_2	Relative angle of revolute joint 2	radians

Table 3-2: A summary of kinematic parameters of the wheeled mobile robot

Reviewing the schematic, the relative transformations between subsequent frames are defined as:

$$\begin{aligned}
 {}_O\mathbf{T}^B &= \begin{bmatrix} c_\phi & -s_\phi & x_b \\ s_\phi & c_\phi & y_b \\ 0 & 0 & 1 \end{bmatrix}, {}_B\mathbf{T}^L = \begin{bmatrix} 1 & 0 & -d \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix}, \\
 {}_B\mathbf{T}^R &= \begin{bmatrix} 1 & 0 & -d \\ 0 & 1 & -b \\ 0 & 0 & 1 \end{bmatrix}, {}_B\mathbf{T}^E = \begin{bmatrix} 1 & 0 & L_a \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}
 \end{aligned}$$

where, $c_\phi = \cos(\phi)$ and $s_\phi = \sin(\phi)$. Combining the relative homogenous transformations, the final set of transformations are defined

$$\begin{aligned}
{}_o\mathbf{T}^B &= \begin{bmatrix} c_\phi & -s_\phi & x_b \\ s_\phi & c_\phi & y_b \\ 0 & 0 & 1 \end{bmatrix} \\
{}_o\mathbf{T}^L &= {}_o\mathbf{T}^B {}_B\mathbf{T}^L = \begin{bmatrix} c_\phi & -s_\phi & x_b - bs_\phi - dc_\phi \\ s_\phi & c_\phi & y_b + bc_\phi - ds_\phi \\ 0 & 0 & 1 \end{bmatrix} \\
{}_o\mathbf{T}^R &= {}_o\mathbf{T}^B {}_B\mathbf{T}^R = \begin{bmatrix} c_\phi & -s_\phi & x_b + bs_\phi - dc_\phi \\ s_\phi & c_\phi & y_b - ds_\phi - bc_\phi \\ 0 & 0 & 1 \end{bmatrix} \\
{}_o\mathbf{T}^E &= {}_o\mathbf{T}^B {}_B\mathbf{T}^E = \begin{bmatrix} c_\phi & -s_\phi & x_b + L_a c_\phi \\ s_\phi & c_\phi & y_b + L_a s_\phi \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned}$$

From the homogenous coordinates, the Jacobian matrices are computed. The effect of the wheel states into the angular Jacobian for each wheel must be explicitly accounted for

$$\begin{aligned}
\mathbf{J}_{v_B} &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}, & \mathbf{J}_{\omega_B} &= [0 \ 0 \ 1 \ 0 \ 0] \\
\mathbf{J}_{v_L} &= \begin{bmatrix} 1 & 0 & ds_\phi - bc_\phi & 0 & 0 \\ 0 & 1 & -dc_\phi - bs_\phi & 0 & 0 \end{bmatrix}, & \mathbf{J}_{\omega_L} &= [0 \ 0 \ 0 \ 0 \ 1] \\
\mathbf{J}_{v_R} &= \begin{bmatrix} 1 & 0 & ds_\phi + bc_\phi & 0 & 0 \\ 0 & 1 & -dc_\phi + bs_\phi & 0 & 0 \end{bmatrix}, & \mathbf{J}_{\omega_R} &= [0 \ 0 \ 0 \ 1 \ 0] \\
\mathbf{J}_{v_E} &= \begin{bmatrix} 1 & 0 & -L_a s_\phi & 0 & 0 \\ 0 & 1 & L_a c_\phi & 0 & 0 \end{bmatrix}, & \mathbf{J}_{\omega_E} &= [0 \ 0 \ 1 \ 0 \ 0]
\end{aligned}$$

The input power is given by the product of joint torque and joint angular velocity.

$$\Pi = [\tau_R \dot{\theta}_R \quad \tau_L \dot{\theta}_L]^T$$

Again a viscous friction model is used model the dissipative elements in the system.

$$\Delta = [c_{v_R} \dot{\theta}_R \quad c_{v_L} \dot{\theta}_L]^T$$

When dealing with differentially driven wheeled robots, constraints must be imposed on the dynamics to prevent infeasible motions. The WMR system contains three constraints related to the drive system. The first restricts the base from moving in a direction perpendicular to the rolling direction of the drive wheels. In other words, the velocity perpendicular to the rolling direction must be zero which is given by:

$$-\dot{x}_b \sin \phi + \dot{y}_b \cos \phi - d\dot{\phi} = 0 \quad (3.1)$$

The final two constraints relate the wheel rotation to the forward and rotational velocity of the base.

$$\begin{aligned} \dot{x}_b \cos \phi + \dot{y}_b \sin \phi + b\dot{\phi} &= r\dot{\theta}_R \\ \dot{x}_b \cos \phi + \dot{y}_b \sin \phi - b\dot{\phi} &= r\dot{\theta}_L \end{aligned} \quad (3.2)$$

We can combine these three equations, to form a single constraint matrix, $\mathbf{A}(q)\dot{q} = 0$. \mathbf{A} is given by:

$$\mathbf{A} = \begin{bmatrix} -\sin \phi & \cos \phi & -d & 0 & 0 \\ \cos \phi & \sin \phi & b & -r & 0 \\ \cos \phi & \sin \phi & -b & 0 & -r \end{bmatrix}$$

These constraints are incorporated into the dynamics through a vector of Lagrange multipliers (λ). This vector is a state dependent term that will supply the necessary forces on the system to insure the constraints are not violated. After taking the necessary derivatives and applying the constraints the final dynamic equations are calculated as:

$$\mathbf{M}(q)\ddot{q} + \mathbf{C}(q, \dot{q})\dot{q} + \mathbf{R}(q, \dot{q})\dot{q} = \mathbf{E}\tau - \mathbf{A}(q)^T \lambda$$

$$\mathbf{M} = \begin{bmatrix} a_2 & 0 & 2a_1 ds_\phi & 0 & 0 \\ 0 & a_2 & -2a_1 dc_\phi & 0 & 0 \\ 2a_1 ds_\phi & -2a_1 dc_\phi & a_3 & 0 & 0 \\ 0 & 0 & 0 & a_4 & 0 \\ 0 & 0 & 0 & 0 & a_4 \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 0 & 0 & 2a_1 dc_\phi \dot{\phi} & 0 & 0 \\ 0 & 0 & 2a_1 ds_\phi \dot{\phi} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{R} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_{v_R} & 0 \\ 0 & 0 & 0 & 0 & c_{v_L} \end{bmatrix}, \mathbf{E} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\begin{aligned} a_1 &= m_w & a_3 &= I_b + 2m_w(d^2 + b^2) \\ a_2 &= m_b + 2m_w & a_4 &= I_w \end{aligned}$$

3.2.1 Feasible Motion Dynamics

For dynamic systems with constraints, the Lagrange multipliers add an extra level of complexity making it difficult to develop efficient control algorithms. It is more beneficial to develop a compact form of the equations of motion which implicitly account for the kinematic constraints. In this section, the constrained Euler-Lagrangian dynamics are projected into the feasible motion subspace to prevent the need to calculate the Lagrange multipliers directly. The constrained dynamics are projected into the feasible subspaces by filtering the dynamics through the null space of the constraint matrix (\mathbf{S}). By definition, the null space matrix has the property

$$\mathbf{AS} = 0 \text{ or } \mathbf{A}^T \mathbf{S}^T = 0$$

Since the derivation of the null space is not unique, an additional property can be introduced to the final matrix

$$\dot{q} = \mathbf{S} \nu \text{ and } \ddot{q} = \mathbf{S} \dot{\nu} + \dot{\mathbf{S}} \nu$$

where, ν is a vector of independent coordinates. Following the formulation presented in [12], the independent joint space is set to be the actuation space of the manipulator

$$\nu = [\dot{\theta}_R, \dot{\theta}_L]^T$$

Yielding the final null space matrix to be:

$$\mathbf{S} = \begin{bmatrix} c[b \cos \phi - d \sin \phi] & c[b \cos \phi + d \sin \phi] \\ c[b \sin \phi + d \cos \phi] & c[b \sin \phi - d \cos \phi] \\ c & -c \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \text{ where } c = \frac{r}{2b}$$

From above we know the dynamic equations of motion with constraints are of the form:

$$\mathbf{M}(q)\ddot{q} + \mathbf{C}(q, \dot{q})\dot{q} + \mathbf{R}(q, \dot{q})\dot{q} = \mathbf{E}\tau - \mathbf{A}(q)^T \lambda$$

The constrained dynamics are pre-multiplied with transpose of the null space mapping

$$\mathbf{S}^T \mathbf{M}(q)\ddot{q} + \mathbf{S}^T \mathbf{C}(q, \dot{q})\dot{q} + \mathbf{S}^T \mathbf{R}(q)\dot{q} = \mathbf{S}^T \mathbf{E}\tau - \mathbf{S}^T \mathbf{A}^T \lambda$$

Our choice of \mathbf{S}^T cancels out the constraint matrix, \mathbf{A}^T eliminating the need to solve for the constraint forces, λ .

$$\mathbf{S}^T \mathbf{M}(q)\mathbf{S}\dot{\nu} + \mathbf{S}^T \mathbf{C}(q, \dot{q})\dot{q} + \mathbf{S}^T \mathbf{R}(q)\dot{q} = \mathbf{S}^T \mathbf{E}\tau$$

Distributing the \mathbf{S}^T through and substituting in for \ddot{q} , gives the final set of equations

$$\mathbf{H}(q)\dot{\nu} + \mathbf{V}(q, \dot{q})\nu + \mathbf{D}(\dot{q})\nu = \mathbf{N}\tau \quad (3.3)$$

$$\begin{aligned}
\mathbf{H}(q) &= \mathbf{S}^T \mathbf{M}(q) \mathbf{S}, & \mathbf{V}(q, \dot{q}) &= \mathbf{S}^T \mathbf{M}(q) \dot{\mathbf{S}} + \mathbf{S}^T \mathbf{C}(q, \dot{q}) \mathbf{S} \\
\mathbf{D}(q) &= \mathbf{S}^T \mathbf{R}(q) \mathbf{S}, & \mathbf{N} &= \mathbf{S}^T \mathbf{E} = \mathbf{I} \\
\dot{v} &= [\ddot{\theta}_R, \ddot{\theta}_L]^T
\end{aligned}$$

Much like the infeasible motion dynamics, the feasible motion dynamics contain three important properties. [13]

Property 1: $\mathbf{H}(q)$ is a symmetric and positive-definite matrix

Property 2: There is a parametric vector a on the dynamics which satisfies

$$\mathbf{H}(q)\dot{v} + \mathbf{V}v + \mathbf{D}v = \mathbf{Y}(q, \dot{q}, v, \dot{v})a$$

Property 3: The matrix $\dot{\mathbf{H}}(q) - 2\mathbf{V}(q, \dot{q})$ is skew-symmetric

3.2.2 Constraint Analysis

The types of constraints (holonomic or nonholonomic) can be determined by analyzing the distribution spanned by the column vectors of the null space matrix, \mathbf{S} . The total number of nonholonomic constraints is equal to the total number of linearly independent Lie brackets which span the feasible directions of the distribution space. The constraint matrix, \mathbf{S} has two independent column vectors:

$$\mathbf{S} = [s_1(q), s_2(q)] = \begin{bmatrix} c[b \cos \phi - d \sin \phi] & c[b \cos \phi + d \sin \phi] \\ c[b \sin \phi + d \cos \phi] & c[b \sin \phi - d \cos \phi] \\ c & -c \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Computing the first Lie bracket gives:

$$\begin{aligned}
s_3(q) &= [s_1(q), s_2(q)] = \frac{\partial}{\partial q}(s_2(q))s_1(q) - \frac{\partial}{\partial q}(s_1(q))s_2(q) \\
&= [-rcs_\phi \quad rcc_\phi \quad 0 \quad 0 \quad 0]^T
\end{aligned}$$

Since $s_3(q)$ does not lie in the distribution spanned by $s_1(q)$ and $s_2(q)$, at least one of the constraints is nonholonomic. Computing the next bracket:

$$s_4(q) = [s_1(q), s_3(q)] = [-rc^2 c_\phi \quad -rc^2 s_\phi \quad 0 \quad 0 \quad 0]^T$$

We again find that $s_4(q)$ is linearly independent of $s_1(q)$, $s_2(q)$ and $s_3(q)$. Computing subsequent Lie brackets (with any two combinations of $s_1(q)$ through $s_4(q)$) yields that it is not longer possible to find another linearly independent direction in the distribution space so the space spanned by $s_1(q)$ through $s_4(q)$ is involutive. Counting the number of calculated Lie brackets, we conclude that the WMR system contains two nonholonomic constraints. The remaining constraint must then be holonomic and can be found by subtracting equations (3.2).

$$2b\dot{\phi} = r(\dot{\theta}_R - \dot{\theta}_L)$$

Integrating the resulting equations gives:

$$\phi = c(\theta_R - \theta_L) + \beta$$

where β is the constant of integration which can be made zero by properly setting θ_R and θ_L . This equation is the holonomic constraint linking the wheel angles to the rotation of the body fixed frame.

4 Control

The control section derives the teleoperation approach for each of the robotic slave systems. The output synchronization framework outlined in [1] is the basis for coupling the master and slave states together.

4.1 Adaptive Output Synchronization

We outline the output synchronization framework for two generic Lagrangian systems. The link between output synchronization and robust passivity control is established. This work is based largely on the work of [1], [20] and [21].

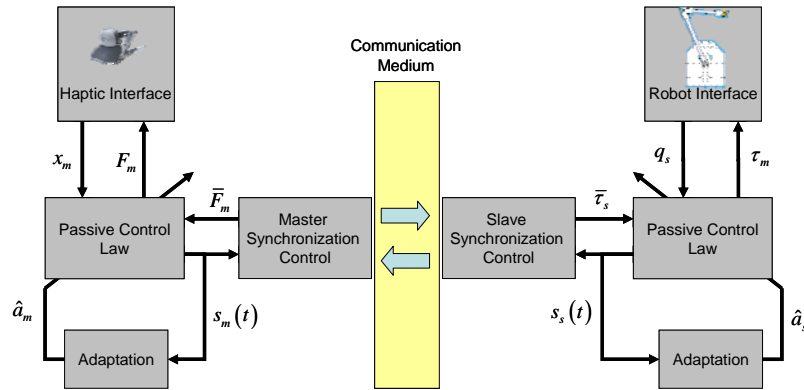


Figure 4-1: A schematic of bilateral synchronization for teleoperation.

The synchronization control strategy aims at solving two goals. A local nonlinear feedback control law is employed to render the local dynamics passive with respect to a secondary control input and output containing both position and velocity information. The secondary input is the kinematic coupling between master and slave which is passive in nature and insensitive to communication delay. This framework has been expanded to include a basic viscous friction term as outlined in [17].

Using the dynamic formulation above, the general dynamics for a master and slave system as:

$$\begin{aligned}\mathbf{M}_m(q)\ddot{q}_m + \mathbf{C}(q, \dot{q})_m \dot{q}_m + \mathbf{D}(\dot{q})_m \dot{q}_m &= \tau_m \\ \mathbf{M}_s(q)\ddot{q}_s + \mathbf{C}(q, \dot{q})_s \dot{q}_s + \mathbf{D}(\dot{q})_s \dot{q}_s &= \tau_s\end{aligned}$$

where the subscripts m and s represent the master and slave systems, respectively. The first step is to develop a local controller which renders the systems passive for a kinematic output. Using an estimated model, we apply the control:

$$\begin{aligned}\tau_m &= \hat{\mathbf{M}}(q)_m \ddot{q}_m^r + \hat{\mathbf{C}}(q, \dot{q})_m \dot{q}_m^r + \hat{\mathbf{D}}(\dot{q})_m \dot{q}_m^r + \bar{\tau}_m \\ \tau_s &= \hat{\mathbf{M}}(q)_s \ddot{q}_s^r + \hat{\mathbf{C}}(q, \dot{q})_s \dot{q}_s^r + \hat{\mathbf{D}}(\dot{q})_s \dot{q}_s^r + \bar{\tau}_s\end{aligned}$$

where τ_m and τ_s are the master and slave control inputs, $\langle \hat{\bullet} \rangle$ represents the estimated model parameter, $\bar{\tau}_m$ and $\bar{\tau}_s$ are a secondary control used for synchronization and the new joint reference states are given by:

$$\dot{q}_i^r = -\lambda q_i \text{ and } \ddot{q}_i^r = -\lambda \dot{q}_i$$

where λ is a positive constant matrix. This control law represents the same robust passive control law as in section 2.5 where the desired trajectory is the origin ($q_d = \dot{q}_d = \ddot{q}_d = 0$). The difference between the control laws is the addition of the secondary coupling control inputs. If the system dynamics are calculated using the method outlined about, the system dynamics can be written in a regressor form:

$$\begin{aligned}\tau_m &= \mathbf{Y}(q, \dot{q}_m^r, \ddot{q}_m^r)_m \hat{a}_m + \bar{\tau}_m \\ \tau_s &= \mathbf{Y}(q, \dot{q}_s^r, \ddot{q}_s^r)_s \hat{a}_s + \bar{\tau}_s\end{aligned} \tag{3.4}$$

Applying the control law yields the following closed loop system dynamics

$$\begin{aligned}
\dot{q}_m &= s_m - \lambda q_m \\
\mathbf{M}_m(q) \dot{s}_m + [\mathbf{C}(q, \dot{q})_m + \mathbf{D}(\dot{q})_m] s_m &= \mathbf{Y}(q_m, s_m)_m \tilde{a}_m + \bar{\tau}_m \\
\dot{q}_s &= s_s - \lambda q_s \\
\mathbf{M}_s(q) \dot{s}_s + [\mathbf{C}(q, \dot{q})_s + \mathbf{D}(\dot{q})_s] s_s &= \mathbf{Y}(q_s, s_s)_s \tilde{a}_s + \bar{\tau}_s
\end{aligned}$$

where $s_i = \dot{q}_i + \lambda q_i \forall i \in m, s$. Just like in the robust passive control case, the parameter error prevents the closed loop systems from being output passive. A similar parameter update law is introduced to maintain the passivity of the each local system.

$$\begin{aligned}
\dot{\hat{a}}_m &= -\Gamma^{-1} \mathbf{Y}(q_m, s_m)_m^T s_m \\
\dot{q}_m &= -\lambda q_m + s_m \\
\mathbf{M}_m(q) \dot{s}_m + [\mathbf{C}(q, \dot{q})_m + \mathbf{D}(\dot{q})_m] s_m &= \mathbf{Y}(q_m, s_m)_m \tilde{a}_m + \bar{\tau}_m \\
\dot{\hat{a}}_s &= -\Lambda^{-1} \mathbf{Y}(q_s, s_s)_s^T s_s \\
\dot{q}_s &= -\lambda q_s + s_s \\
\mathbf{M}_s(q) \dot{s}_s + [\mathbf{C}(q, \dot{q})_s + \mathbf{D}(\dot{q})_s] s_s &= \mathbf{Y}(q_s, s_s)_s \tilde{a}_s + \bar{\tau}_s
\end{aligned}$$

From section 2.5, we know that the above systems are passive with respect to the secondary control input $\bar{\tau}_i$ and output s_i (for $i = m, s$). This solves the first task of our desired output synchronization controller; a passive kinematic coupling must now be developed.

Two systems are said to be output synchronized if the following condition holds.

$$\lim_{t \rightarrow \infty} \|s_i(t-T) - s_j(t)\| = 0 \text{ for all } i, j$$

where

$$\begin{aligned}
s_i(t-T) - s_j(t) &= (\dot{q}_i(t-T) + \lambda q_i(t-T)) - (\dot{q}_j(t) + \lambda q_j(t)) \\
&= (\dot{q}_i(t-T) - \dot{q}_j(t)) + \lambda (q_i(t-T) - q_j(t)) \\
&= \dot{e}_i(t) + \lambda e_i(t)
\end{aligned}$$

where $e_i(t) = q_i(t-T) - q_j(t)$, T is the constant time delay and $\|\cdot\|$ denotes the Euclidean norm. This system represents an exponentially stable linear system with respect to the inputs $s_i(t-T)$ and $s_j(t)$. As the authors of [20] suggest, if $s_i(t-T) - s_j(t)$ converges asymptotically to zero and $e_i(t)$ is bounded, then the position and velocity states of each system will asymptotically approach each other. To achieve the desired output synchronization, the following coupling control is suggested:

$$\begin{aligned}\bar{\tau}_m &= \mathbf{K}(s_s(t-T) - s_m(t)) \\ \bar{\tau}_s &= \mathbf{K}(s_m(t-T) - s_s(t))\end{aligned}\tag{3.5}$$

where \mathbf{K} a constant positive diagonal matrix and T again represents the time delay introduced by the communication medium. The following positive semi-definite storage function is considered,

$$\begin{aligned}V &= \frac{1}{2}(s_m^T(t)\mathbf{M}_m s_m(t) + 2e_m^T \boldsymbol{\lambda} \mathbf{K} e_m + \tilde{a}_m^T \boldsymbol{\Gamma} \tilde{a}_m) + \tilde{q}_m^T \boldsymbol{\lambda} \mathbf{D}(q, \dot{q})_m \tilde{q}_m \\ &\quad \frac{1}{2}(s_s^T(t)\mathbf{M}_s s_s(t) + 2e_s^T \boldsymbol{\lambda} \mathbf{K} e_s + \tilde{a}_s^T \boldsymbol{\Lambda} \tilde{a}_s) + \tilde{q}_s^T \boldsymbol{\lambda} \mathbf{D}(q, \dot{q})_s \tilde{q}_s \\ &\quad + \int_{t-T}^t (s_m^T(\zeta) s_m(\zeta) + s_s^T(\zeta) s_s(\zeta)) d\zeta\end{aligned}$$

It should be noted that the master and slave candidate Lyapunov functions are identical to the function used in section 2.5, where $\mathbf{K}_D^i = \mathbf{D}(q, \dot{q})_i$. Taking the derivative of the storage function and using the analysis developed in section 2.5 yields:

$$\begin{aligned}\dot{V} &= -(\tilde{q}_m^T \boldsymbol{\lambda}^T \mathbf{D}(q, \dot{q})_m \boldsymbol{\lambda} \tilde{q}_m + \dot{\tilde{q}}_m^T \mathbf{D}(q, \dot{q})_m \dot{\tilde{q}}_m) + s_m^T(t) \bar{\tau}_m \\ &\quad -(\tilde{q}_s^T \boldsymbol{\lambda}^T \mathbf{D}(q, \dot{q})_s \boldsymbol{\lambda} \tilde{q}_s + \dot{\tilde{q}}_s^T \mathbf{D}(q, \dot{q})_s \dot{\tilde{q}}_s) + s_s^T(t) \bar{\tau}_s \\ &\quad + 2e_m^T \boldsymbol{\lambda} \mathbf{K} \dot{e}_m + 2e_s^T \boldsymbol{\lambda} \mathbf{K} \dot{e}_s \\ &\quad + \mathbf{K} [s_m^T s_m - s_m^T(t-T) s_m(t-T) + s_s^T s_s - s_s^T(t-T) s_s(t-T)]\end{aligned}$$

Substituting in for the synchronization control inputs and collecting terms gives:

$$\begin{aligned}
\dot{V} = & -\left(\tilde{q}_m^T \boldsymbol{\lambda}^T \mathbf{D}(q, \dot{q})_m \boldsymbol{\lambda} \tilde{q}_m + \dot{\tilde{q}}_m^T \mathbf{D}(q, \dot{q})_m \dot{\tilde{q}}_m\right) + s_m^T(t) \mathbf{K} (s_s(t-T) - s_m(t)) \\
& -\left(\tilde{q}_s^T \boldsymbol{\lambda}^T \mathbf{D}(q, \dot{q})_s \boldsymbol{\lambda} \tilde{q}_s + \dot{\tilde{q}}_s^T \mathbf{D}(q, \dot{q})_s \dot{\tilde{q}}_s\right) + s_s^T(t) \mathbf{K} (s_m(t-T) - s_s(t)) \\
& + 2e_m^T \boldsymbol{\lambda} \mathbf{K} \dot{e}_m + 2e_s^T \boldsymbol{\lambda} \mathbf{K} \dot{e}_s \\
& + \mathbf{K} \left[s_m^T s_m - s_m^T(t-T) s_m(t-T) + s_s^T s_s - s_s^T(t-T) s_s(t-T) \right]
\end{aligned}$$

$$\begin{aligned}
\dot{V} = & -\left(\tilde{q}_m^T \boldsymbol{\lambda}^T \mathbf{D}(q, \dot{q})_m \boldsymbol{\lambda} \tilde{q}_m + \dot{\tilde{q}}_m^T \mathbf{D}(q, \dot{q})_m \dot{\tilde{q}}_m\right) \\
& -\left(\tilde{q}_s^T \boldsymbol{\lambda}^T \mathbf{D}(q, \dot{q})_s \boldsymbol{\lambda} \tilde{q}_s + \dot{\tilde{q}}_s^T \mathbf{D}(q, \dot{q})_s \dot{\tilde{q}}_s\right) \\
& + 2e_m^T \boldsymbol{\lambda} \mathbf{K} \dot{e}_m + 2e_s^T \boldsymbol{\lambda} \mathbf{K} \dot{e}_s \\
& - \mathbf{K} (s_m(t-T) - s_s(t))^T (s_m(t-T) - s_s(t)) \\
& - \mathbf{K} (s_s(t-T) - s_m(t))^T (s_s(t-T) - s_m(t))
\end{aligned}$$

Substituting in for the state error term, e_i

$$\begin{aligned}
\dot{V} = & -\left(\tilde{q}_m^T \boldsymbol{\lambda}^T \mathbf{D}(q, \dot{q})_m \boldsymbol{\lambda} \tilde{q}_m + \dot{\tilde{q}}_m^T \mathbf{D}(q, \dot{q})_m \dot{\tilde{q}}_m\right) \\
& -\left(\tilde{q}_s^T \boldsymbol{\lambda}^T \mathbf{D}(q, \dot{q})_s \boldsymbol{\lambda} \tilde{q}_s + \dot{\tilde{q}}_s^T \mathbf{D}(q, \dot{q})_s \dot{\tilde{q}}_s\right) + \mathbf{K} (2e_m^T \boldsymbol{\lambda} \dot{e}_m + 2e_s^T \boldsymbol{\lambda} \dot{e}_s) \\
& - \mathbf{K} \left[2e_m^T \boldsymbol{\lambda} \dot{e}_m + 2e_s^T \boldsymbol{\lambda} \dot{e}_s + e_m^T \boldsymbol{\lambda}^2 e_m + e_s^T \boldsymbol{\lambda}^2 e_s + \dot{e}_m^T \dot{e}_m + \dot{e}_s^T \dot{e}_s \right]
\end{aligned}$$

$$\begin{aligned}
\dot{V} = & -\left(\tilde{q}_m^T \boldsymbol{\lambda}^T \mathbf{D}(q, \dot{q})_m \boldsymbol{\lambda} \tilde{q}_m + \dot{\tilde{q}}_m^T \mathbf{D}(q, \dot{q})_m \dot{\tilde{q}}_m\right) \\
& -\left(\tilde{q}_s^T \boldsymbol{\lambda}^T \mathbf{D}(q, \dot{q})_s \boldsymbol{\lambda} \tilde{q}_s + \dot{\tilde{q}}_s^T \mathbf{D}(q, \dot{q})_s \dot{\tilde{q}}_s\right) \\
& - \mathbf{K} \left[e_m^T \boldsymbol{\lambda}^2 e_m + e_s^T \boldsymbol{\lambda}^2 e_s + \dot{e}_m^T \dot{e}_m + \dot{e}_s^T \dot{e}_s \right] \leq 0
\end{aligned}$$

The above expression shows that all of the signals in the system are bounded. It can be shown that the states $\lim_{t \rightarrow \infty} \dot{\tilde{q}}_i = \lim_{t \rightarrow \infty} \tilde{q}_i = 0$ and the outputs $\lim_{t \rightarrow \infty} s_i(t-T) s_j(t) = 0$.

This implies that master and slave systems output synchronize and maintain stability in the presence of communication time delays.[20] This analysis has been expanded to prove the boundedness of the state trajectories when the slave system is in contact with a passive environment and the operator applies a non-passive bounded external force to the master system. [1]

4.1.1 Parameter Adaptation

The parameter update law for the adaptive synchronization law is the product of two time varying parameters.

$$\mathbf{Y}(q_i, s_i)_i^T s(t)_i \text{ where } i = m, s$$

It is pointed out in [21], these two parameters, despite being bounded, are not guaranteed to stop in steady state. This is due to the fact that our choice in $s(t)_i$ updates the parameters until the system state converges to the origin instead of the state of the synchronized system. Looking at the control law, (3.4) it is obvious that the estimated parameters affect the transient performance of the coupled systems. Therefore a gradient project method is implemented to restrict the estimates of \hat{a}_i to inside a bounded convex set as suggested in [21]. The gradient projection method developed in [22] is given by:

$$\dot{\hat{a}}_i = \begin{cases} \Gamma^{-1} \mathbf{Y}_i^T s_i & \text{if } \hat{a}_i^T \hat{a}_i < M_0^2 \text{ or} \\ & \text{if } \hat{a}_i^T \hat{a}_i = M_0^2 \text{ and } (\Gamma^{-1} \mathbf{Y}_i^T s_i)^T \hat{a}_i \leq 0 \\ \left(\mathbf{I} - \frac{\Gamma^{-1} \hat{a}_i \hat{a}_i^T}{\hat{a}_i^T \Gamma^{-1} \hat{a}_i} \right) \Gamma^{-1} \mathbf{Y}_i^T s_i & \text{otherwise} \end{cases}$$

where $\hat{a}(0)_i$ is chosen so $\hat{a}(0)_i^T \hat{a}(0)_i \leq M_0^2$ and $\mathbf{Y}_i = \frac{s_i}{m^2}$ where m^2 the normalizing constant $\dot{q}_r^T \dot{q}_r$, $\Gamma = \Gamma^T > 0$.

4.1.2 Hybrid Teleoperation Control

It is ideal to have the ability reposition the master device before activating the kinematic coupling control law when implementing a synchronization control law. In this section a hybrid controller which can turn on and off the coupling control between master

and slave is outlined. The hybrid system has two states; a coupling state and holding state which is triggered by a flag (H_{on}). When $H_{on} \geq 1$, the controller is in the coupling state and the master and slave are synchronized using the control law outlined in section 4.1. When $H_{on} < 1$ the force feedback on the master system is turned off and the slave system will follow the new secondary control:

$$\begin{aligned}\bar{\tau}_s &= -\mathbf{K}s_s(t) \\ \dot{q}_r &= -\lambda(q - q_c)\end{aligned}$$

where q_c is the actual joint state of the slave manipulator at the time H_{on} is switched off ($H_{on} < 1$). The new secondary control law is fed into the primary controller outlined above.

$$\tau_s = -\mathbf{Y}(q, \dot{q})_s \hat{a}_s + \bar{\tau}_s$$

The resulting control is identical to the robust passivity control law discussed in section 2.5. It will drive to the slave system to exponentially converge to the q_c joint state until the coupling control is turned on. The passivity controller will be turned off for the master system leaving it free to move under the users influence. This gives the user the ability to reposition the master system before engaging the kinematic coupling control law.

4.2 Synchronization Control of a RR Manipulator

In this section, the output synchronization framework is adapted to couple a Phantom Omni haptic device with the RR manipulator. The Cartesian position (x,y) is used as the output state linking the master Omni system to the RR slave. Taking the RR dynamics

from section 3.1 and modeling the Phantom Omni as a translational mass, the equations of motion are given as:

$$\begin{aligned}\mathbf{M}(q)_m \ddot{x}_m &= F_m \\ \hat{\mathbf{M}}(q)_s \ddot{q}_s + \hat{\mathbf{C}}(q, \dot{q})_s \dot{q}_s + \hat{\mathbf{D}}(\dot{q})_s \dot{q}_s &= \tau_s\end{aligned}$$

where \ddot{x}_m and F_m are vectors of planar Cartesian accelerations and forces, respectively.

We would like to adapt the output synchronization controller to this system using a passive Cartesian coupling between the RR and Omni end effector positions. In [23], the authors suggest an alternate way of calculating the reference state, \dot{q}_r , to implement the robust passivity based controller in Cartesian space.

$$\dot{q}_r = \mathbf{J}^{-1} \left[\dot{x}_d + \boldsymbol{\lambda} (x_d - x) \right]$$

where \mathbf{J} is the manipulator Jacobian, x_d and \dot{x}_d are the desired Cartesian states and x is the current Cartesian position. This transformation is only valid when the Jacobian is non-singular and non-redundant so care must be taken when during implementing. Using the Cartesian transformation, the following local controller is proposed.

$$\begin{aligned}F_m &= \mathbf{M}(q)_m \ddot{x}_m^r + \bar{F}_m \\ \tau_s &= \mathbf{M}(q)_s \ddot{q}_s^r + \mathbf{C}(q, \dot{q})_s \dot{q}_s^r + \mathbf{D}(\dot{q})_s \dot{q}_s^r + \bar{\tau}_s\end{aligned}$$

where \mathbf{J}_s is the linear Jacobian matrix of the RR end effector (\mathbf{J}_{v_E}) and \dot{x}_r^i and \ddot{x}_r^i are the new Cartesian reference states defined by:

$$\begin{aligned}\dot{x}_i^r &= -\boldsymbol{\lambda} x_i, & \dot{q}_s^r &= -\boldsymbol{\lambda} q_s \quad \forall i \in m, s \\ \ddot{x}_i^r &= -\boldsymbol{\lambda} \dot{x}_i, & \ddot{q}_s^r &= -\boldsymbol{\lambda} \dot{q}_s \quad \forall i \in m, s\end{aligned}$$

The Cartesian output vector for the master and slave devices are given by:

$$r_m = \dot{x}_m - \dot{x}_m^r = \dot{x}_m + \boldsymbol{\lambda} x_m \quad \text{and} \quad r_s = \dot{x}_s - \dot{x}_s^r = \dot{x}_s + \boldsymbol{\lambda} x_s$$

The synchronization control is defined as:

$$\begin{aligned}\bar{F}_m &= \mathbf{K} [r_s(t-T) - r_m(t)] \\ \bar{v}_s &= \mathbf{KJ}_s^{-1} [r_m(t-T) - r_s(t)]\end{aligned}$$

4.3 Synchronization Control of a WMR

In this section, a Cartesian based synchronization control law is introduced for teleoperation of a WMR. It has been shown that dynamic systems containing both holonomic and nonholonomic constraints, like the WMR, are not input-state linearizable by static state feedback.[24] However, as the authors of [12] suggest, given the proper set of output equations, this type of system may be input-output linearizable. To create an input-output mapping a decoupling matrix for the WMR must first be developed.

4.3.1 WMR Decoupling Matrix

The WMR system contains five generalized coordinates and three bilateral constraints leaving two degrees of freedom ($5 - 3 = 2$). This restricts the number of independent output equations to two. For teleoperation control, the Cartesian position ($x = [x_e, y_e]^T$) of the look ahead point of the WMR was chosen as system outputs. The input-output relationship between q and x is given by displacement vector of the ${}^o\mathbf{T}^E$ transformation.

$$x = h(q) = \begin{bmatrix} x_b + L_a c_\phi \\ y_b + L_a s_\phi \end{bmatrix}$$

Using the output equations outlined above, a decoupling matrix can be formed to map the set of independent joint velocities to the desired output velocities. The decoupling matrix is 2 by 2 and must be full rank.

$$\begin{aligned}
\Phi(q) &= \mathbf{J}_h(q)\mathbf{S}(q) \\
&= \begin{bmatrix} 1 & 0 & -L_a s_\phi & 0 & 0 \\ 0 & 1 & L_a c_\phi & 0 & 0 \end{bmatrix} \begin{bmatrix} c[b \cos \phi - d \sin \phi] & c[b \cos \phi + d \sin \phi] \\ c[b \sin \phi + d \cos \phi] & c[b \sin \phi - d \cos \phi] \\ c & -c \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \\
&= c \begin{bmatrix} b \cos \phi - (d + L_a) \sin \phi & b \cos \phi + (d + L_a) \sin \phi \\ b \sin \phi + (d + L_a) \cos \phi & b \sin \phi - (d + L_a) \cos \phi \end{bmatrix}
\end{aligned}$$

where $c = \frac{r}{2b}$ and $\mathbf{J}_h(q)$ is the output Jacobian given by $\mathbf{J}_{v_E}(q)$. Analyzing the determinate of the decoupling matrix, a singularity is found when $L_a = -d$. This constraint restricts the position of the look ahead point from being located on the axis of rotation of the drive wheels.

4.3.2 Zero Dynamics

Before developing a synchronizing control law, the input output linearization control law must be analyzed. It is possible that the choice of output equations may render a portion of the system dynamics unobservable. If so the stability of these internal dynamics must be assessed before the stability of a controller is known. If the internal dynamics are found to be unstable by our choice of output and feedback a new controller must be developed.

Following the procedure outline in [12], a general set of output equations ($h(q)$) are used to identify any potential unobservable dynamics and their stability. A generalized set of output equations and new state space variable, z is introduced:

$$z = \mathbf{T}(q) = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} h(q) \\ L_f h(q) \\ \tilde{h}(q) \end{bmatrix} = \begin{bmatrix} h(q) \\ \mathbf{\Phi}(q)v \\ \tilde{h}(q) \end{bmatrix}$$

where $\tilde{h}(q)$ is any function which insures $\begin{bmatrix} \mathbf{J}_h^T & \mathbf{J}_{\tilde{h}}^T \end{bmatrix}$ has full rank. $\mathbf{T}(q)$ is the diffeomorphism which maps the system state vector (q) to the new state vector, z . The new system is given by the following.

$$\begin{aligned} \dot{z}_1 &= \frac{\partial h}{\partial q} \dot{q} = z_2 \\ \dot{z}_2 &= \dot{\mathbf{\Phi}}(q)v + \mathbf{\Phi}(q)u \\ \dot{z}_3 &= \mathbf{J}_{\tilde{h}} \mathbf{S}(q)v \end{aligned}$$

Analyzing the new system, and recalling $z_2 = \mathbf{J}_h \mathbf{S}(q)v$, the final \dot{z}_3 state can be redefined as:

$$\dot{z}_3 = \mathbf{J}_{\tilde{h}} \mathbf{S}(q) (\mathbf{J}_h \mathbf{S}(q))^{-1} z_2$$

This shows the dependence of \dot{z}_3 on the state variable z_2 . Input-output linearization and decoupling can be achieved with the following feedback.

$$u = \mathbf{\Phi}(q)^{-1} [\psi - \dot{\mathbf{\Phi}}(q)v]$$

Analyzing the final system, we can see that the following states are observable:

$$\dot{z}_1 = z_2 \quad \dot{z}_2 = \psi \quad y = z_1$$

z_3 is in fact an unobservable internal state in our system and can not be directly controlled with our choice of outputs. However, assuming z_1 and z_2 tend to zero, it can be shown that \dot{z}_3 also tends to zero. Hence \dot{z}_3 is Lagrange stable but not necessarily asymptotically stable.[12] With the internal dynamics identified and determined stable, we are free to use the input output relationship developed in section 4.3.1 for the WMR.

4.3.3 Output Synchronization of a WMR

This section outlines a synchronize framework for coupling a WMR and Phantom Omni through a Cartesian space mapping. The Phantom Omni haptic device is modeled as a translational mass and the feasible dynamic formulation of the WMR is used

$$\begin{aligned}\mathbf{M}(q)_m \ddot{x}_m &= F_m \\ \mathbf{H}(q)_s \dot{v}_s + \mathbf{V}(q, \dot{q})_s v_s + \mathbf{D}(\dot{q})_s v_s &= \tau_s\end{aligned}$$

Using the local state output, the following passive control law is applied to each system.

$$\begin{aligned}F_m &= \mathbf{M}(q)_m \ddot{x}_m^r + \bar{F}_m \\ \tau_s &= \mathbf{Y}(q, v_s^r, \dot{v}_s^r) p_s + \bar{\tau}_s\end{aligned}$$

where

$$\begin{aligned}\dot{x}_i^r &= -\lambda x_i \quad \forall i \in m, s & v_i^r &= -\lambda \begin{bmatrix} \theta_R^i \\ \theta_L^i \end{bmatrix} \quad \forall i \in m, s \\ \ddot{x}_i^r &= -\lambda \dot{x}_i \quad \forall i \in m, s & \dot{v}_i^r &= -\lambda v_i \quad \forall i \in m, s\end{aligned}$$

The decoupling matrix provides a mapping between the independent joints space (v) and Cartesian output space, x .

$$\dot{x} = \Phi(q)v \quad \text{and} \quad v = \Phi(q)^{-1} \dot{x}$$

Using the decoupling mapping, a Cartesian synchronization control law can be found which passively couples the system irrespective of the local generalized coordinates.

$$\begin{aligned}\bar{F}_m &= \mathbf{K} [r_s(t-T) - r_m] \\ \bar{\tau}_s &= \mathbf{K} \Phi(q)_s^{-1} [r_m(t-T) - r_s(t)]\end{aligned}$$

where,

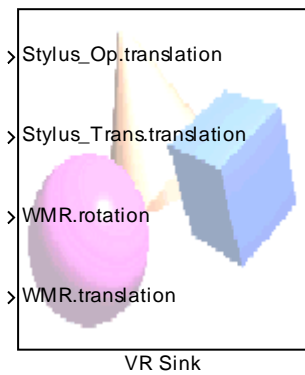
$$r_m = \dot{x}_m - \dot{x}_m^r = \dot{x}_m + \lambda x_m, \quad r_s = \dot{x}_s - \dot{x}_s^r = \dot{x}_s + \lambda x_s$$

5 Implementation Framework

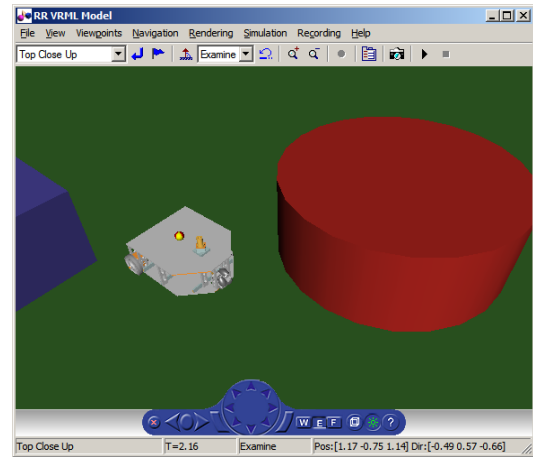
This section discusses the real time framework used for coupling the master and slave devices in simulation and experimentally. The virtual environment is introduced first. The simulation framework and its related subsystems will then be discussed. The real time implementation code and system layout is then introduced with descriptions of the new blocks. The final section will outline a battery of tests used to test the capabilities of the output synchronization controller.

5.1 Virtual Environment Blocks

In all teleoperation schemes it is important to provide feedback to the user in the most realistic fashion possible. A virtual reality scene has been prepared for this purpose. This scene will provide real-time visual feedback of the current state of the master and slave systems. It was programmed in the virtual reality modeling language (VRML), a markup language similar to HTML used for efficiently realizing three dimensional scenes over the internet. Our virtual reality scene has been rendered in a Matlab/Simulink environment and is run on a dedicated laptop. Figure 5-1 shows the VRML sink block, the code block which renders our scene in our code and an example of one of our developed scenes.



a)



b)

Figure 5-1: The elements of a real-time virtual reality scenes: a) The Simulink block for linking real-time code to the VR scene b) a sample of the VRML scene.

The master system position is indicated by a small solid sphere. The human user is free to move the sphere anywhere within the workspace of the haptic device. Separate VRML scenes were created for the RR and WMR slave systems. Each system contains a scaled three dimensional model of the slave system designed in Solidworks.



Figure 5-2: The Phantom Omni six degree of freedom haptic device. [25]

Visualization is only half of the haptic experience. To achieve remote environment immersion, the user must be able to command slave robot and feel the forces present on the system. To achieve this, a three degree of freedom Phantom Omni was used as an input/output device. The Omni captures the three dimensional position through digital encoders and orientation of the gimbal stylus using linear potentiometers.

Cartesian forces are fed back to user through a series of high torque motors driving the base and link joints. Two buttons located on the stylus allow the user to toggle the kinematic coupling between the Phantom Omni and slave systems. The Phantom Omni is shown in Figure 5-2.

The Omni interfaces directly with our Matlab/Simulink interface over a high speed IEEE 1394 (firewire) interface. It is capable of updating end effector forces and state feedback data at a rate at over 1000 Hz. A summary of the technical specifications of the Phantom Omni system can be found in Table 5-1.

Description	Value	Units
Force feedback workspace	160 W x 120 H x 70 D	<i>mm</i>
Nominal position resolution	> 450 (~ 0.055)	<i>dpi (mm)</i>
Maximum exertable force at the nominal position	0.75 (3.3)	<i>lbf (N)</i>
Continuous exertable force for 24 hrs	> 0.2 (0.88)	<i>lbf (N)</i>
Stiffness	X axis > 7.3 (1.26) Y axis > 13.4 (2.31) Z axis > 5.9 (1.02)	<i>lbf (N)</i>

Table 5-1: A summary of technical specifications for the Phantom Omni

5.2 Simulation Framework

This section highlights the key Simulink blocks required for simulated teleoperation control. All simulations were run on a single laptop computer using a virtual network interface between master and slave systems. The code can be broken down into several key subsystems; the haptic and slave control blocks, a stochastic network simulator and a

three dimensional visual output. The simulation Simulink block diagrams are presented for the RR and WMR systems.

5.2.1 RR Simulation code

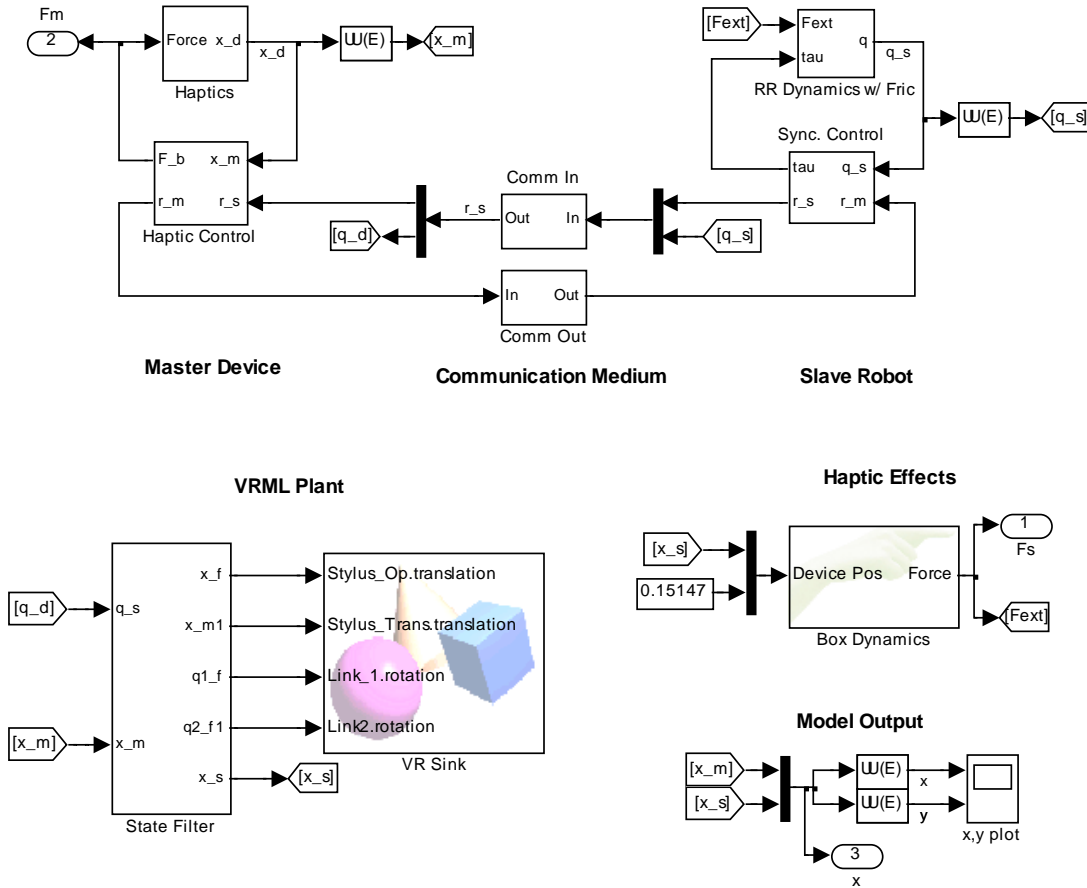


Figure 5-3: The RR simulation Simulink block diagram.

The RR simulation code can be broken down into four major sections. These sections include haptic control, communication simulation, slave control/dynamics and haptic/data feedback.

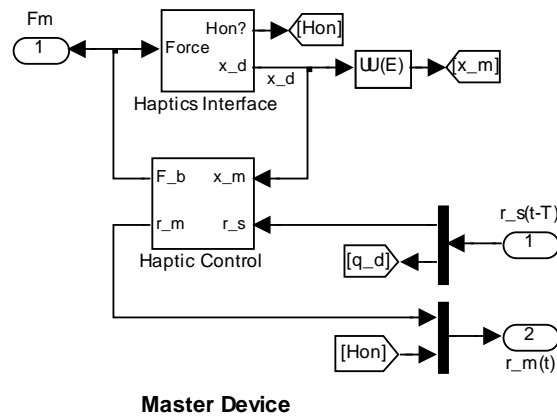


Figure 5-4: The haptic control subsystem.

The haptic controller contains the Phantom Omni driver interface, and the haptic force synchronization controller. (Figure 5-4) The driver interface gives Simulink direct access to the position/orientation output data of the haptic device. Cartesian force can also be commanded in three orthogonal directions. The Phantom Omni comes equipped with two buttons located on the stylus. These buttons are used to toggle the H_{on} state, activating or deactivating the passive coupling between master and slave. The haptic controller block contains the real time code synchronization code developed in section 4.2 and the haptic device dynamics.

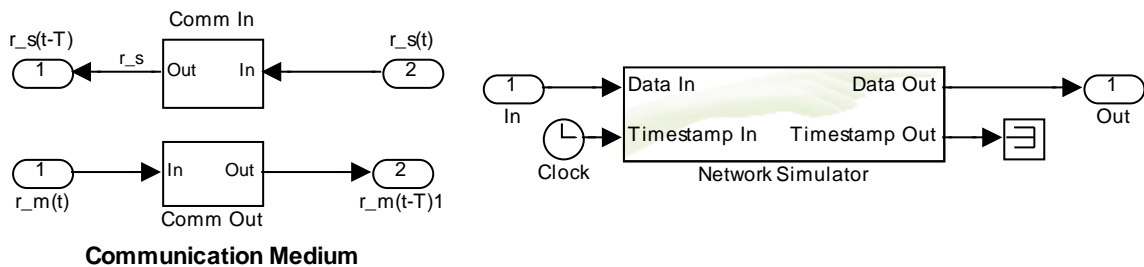


Figure 5-5: The network simulator subsystem and ProSense network simulator block.

The next subsystem will simulate the desired network conditions for time delayed teleoperation testing (Figure 5-5). The heart of this block is a stochastic network simulator. This block is a proprietary network simulator developed by ProSense which

can simulate lossy and delayed network conditions with known statistical performance. Constant, normal or uniform distributions of time delay and random or periodic packet loss conditions can all be simulated by this block. Provisions also exist to upload a text file containing the statistic information on the network delay of a preexisting network. ProSense also provides executable programs for recording the delay properties of existing networks.

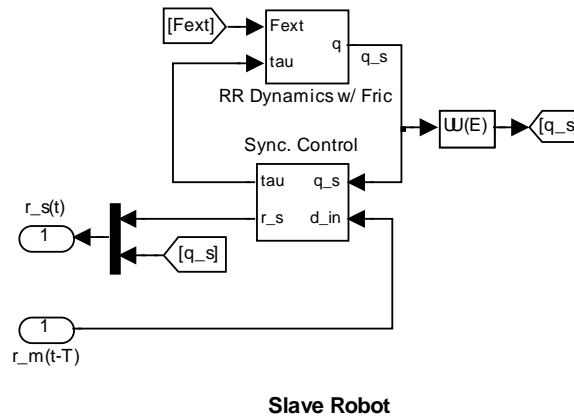


Figure 5-6: The slave manipulator dynamics and control subsystem

Parameter	Value	Parameter	Value
m_1	1.07 kg	I_1	$2.6e-2 \text{ kg} \cdot \text{m}^2$
m_2	0.5 kg	I_2	$6.5e-3 \text{ kg} \cdot \text{m}^2$
L_1	0.5144 m	L_{cm1}	0.2572 m
L_2	0.362 m	L_{cm2}	0.1810 m
c_1	$0.1 \text{ kg} \cdot \text{m}^2/\text{s}$	c_2	$0.1 \text{ kg} \cdot \text{m}^2/\text{s}$
M	$\text{diag}(1e-3, 1e-3) \text{ kg}$		

Table 5-2: RR manipulator kinematic and dynamic and haptic model simulation parameters.

The slave control/dynamics subsystem contains all the systems located on the embedded controller on the slave system (Figure 5-6). The synchronization controller block contains the synchronization control algorithm and gradient project adaptation law

developed in section 4. In the simulation framework, the synchronization controller directly drives the dynamic model of the RR system. The plant simulates the Euler-Lagrangian dynamics of the slave manipulator, developed in section 3. A third order Bogacki-Shampine ODE fixed time step solver running at 1000 Hz is used to solve the dynamics in real time. The kinematic and dynamic parameters for the RR manipulator and haptic model are summarized bellow.

The final subsystem contains the haptic visualization, the wall dynamic simulator and miscellaneous output state recording (Figure 5-7).

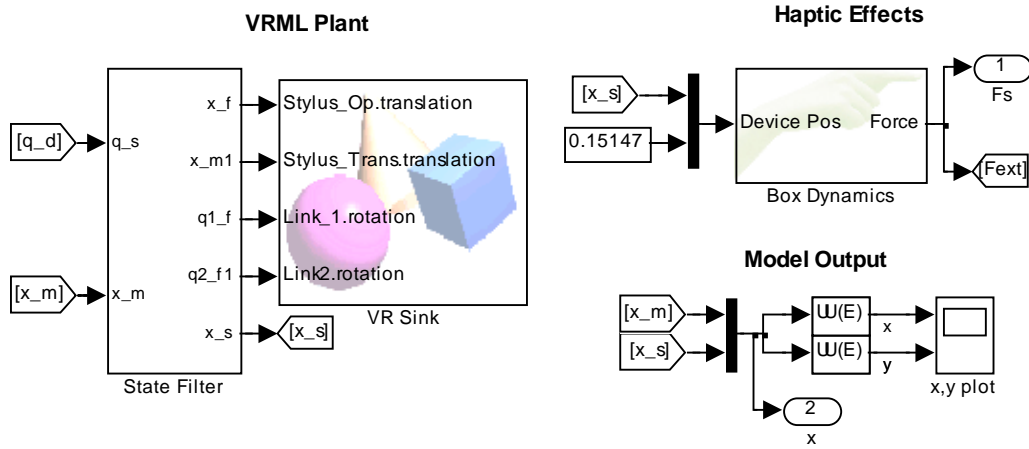


Figure 5-7: The haptic and output subsystem.

The haptic visual plant takes in the simulation state information and renders a three dimensional virtual scene. The user uses the Phantom Omni to drive the RR manipulator and interact with the virtual environment. The haptic scene filters the potentially noisy network data with a single pole linear filter ($\tau = 0.01s$) and updates a three dimensional scene at a rate of 33 Hz.

A virtual wall is added in the workspace of the RR manipulator providing an environment for the user to interact with. The box dynamics is simulated using a ProSense proprietary dynamic model. The block contains friction, inertia and damping

qualities providing the realistic force interaction at the RR end effector. The kinematic and dynamic parameters for the wall are summarized in tabular form in Table 5-3.

Parameter	Value	Parameter	Value
Box Size [w, h, d]	[1, 0.4, 0.5] m	Box Center Location	[0.8 0 0] m
Stiffness	3000 N/m	Coulomb Friction	5
Damping Factor	4	Velocity threshold for coulomb friction	20

Table 5-3: A summary of kinematic and dynamic parameters for a virtual box.

5.2.2 WMR Simulation code

The wheeled mobile robot teleoperation scheme is simulated using similar blocks to the RR simulation (Figure 5-8). Only the unique subsystems are discussed in this section.

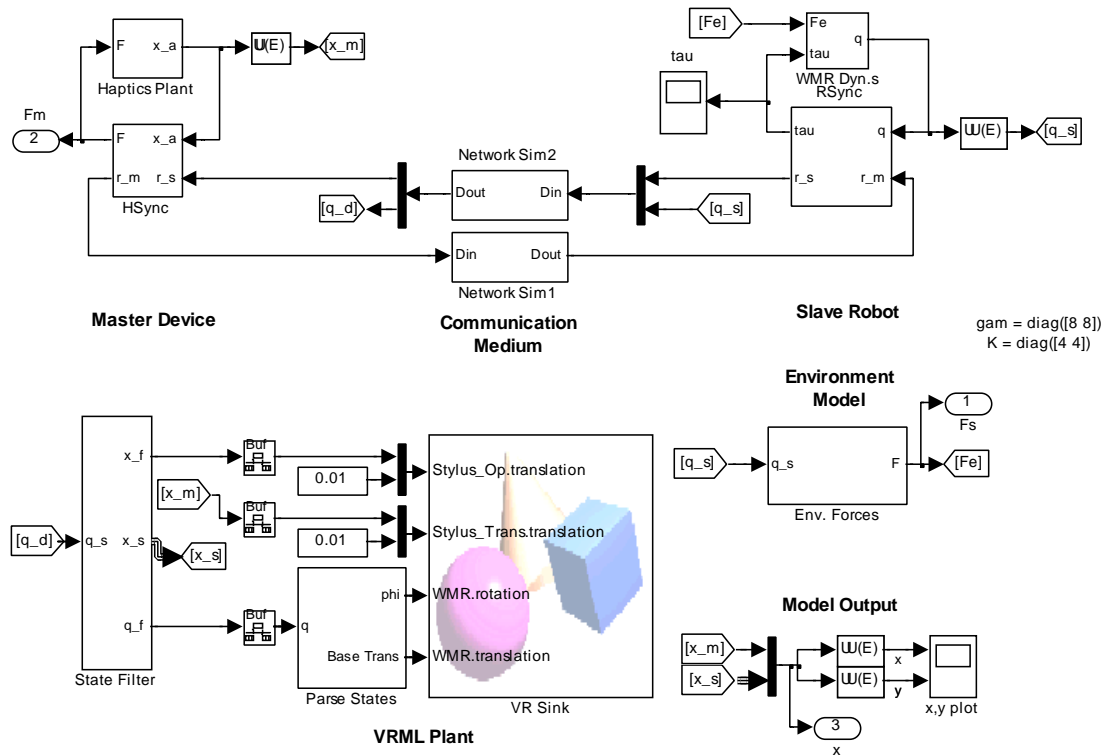


Figure 5-8: The WMR simulation Simulink block diagram.

The most notable differences would be in the slave dynamics and control and haptic output subsystems. The WMR output synchronization scheme and dynamics were used to simulation the slave manipulator. The kinematic and dynamic parameters for the haptic device and simulated WMR are summarized in Table 5-4.

Parameter	Value	Parameter	Value
m_w	0.1 kg	d	0.146 m
m_b	17.25 kg	b	0.168 m
I_w	$2e-4 \text{ kg} \cdot \text{m}^2$	c_R	$0.1 \text{ kg} \cdot \text{m}^2/\text{s}$
I_B	$17.25 \text{ kg} \cdot \text{m}^2$	c_L	$0.1 \text{ kg} \cdot \text{m}^2/\text{s}$
r	0.051 m	M	$\text{diag}(1e-3, 1e-3) \text{ kg}$

Table 5-4: Wheeled mobile robot kinematic and dynamic simulation parameters.

The haptic output subsystem contained a new model reflecting the WMR system. A virtual wall is added into the WMR virtual environment, providing a virtual surface the WMR can interact with. Again, the box dynamics are a proprietary box model created by ProSense with the following kinematic and dynamic parameters.

Parameter	Value	Parameter	Value
Box Size [w, h, d]	[1, 1, 0.5] m	Box Center Location	[0.9 0 0] m
Stiffness	100,000 N/m	Coulomb Friction	1.0
Damping Factor	40	Velocity threshold for coulomb friction	20

Table 5-5: A summary of kinematic and dynamic parameters for a virtual box in the WMR environment.

5.3 Real time Implementation Framework

In this section, we outline how the teleoperation scheme was implemented on a distributed master and slave setting. The hardware and network setup is first discussed. Next, any new Simulink software required for real time implementation is introduced.

5.3.1 Network and Hardware Setup

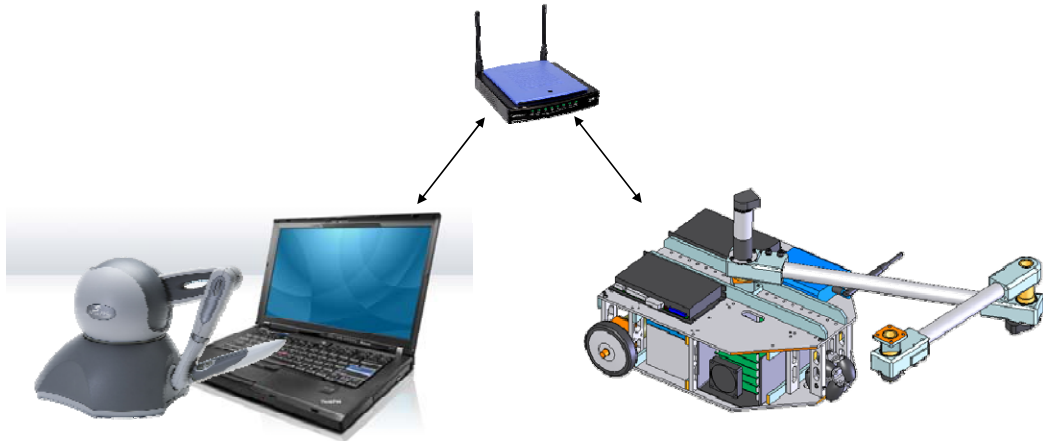


Figure 5-9: An overview of the real time implementation framework.

Unlike the simulation frame work previously introduced, the real time framework distributes the local synchronization controllers on separate master and slave computers (Figure 5-9).

The master system is a laptop running the Matlab/Simulink control and VRML visualization software. The laptop is a R52 ThinkPad laptop, with a single core Pentium M 1.73 GHz processor and integrated Intel GMA 900 graphics accelerator. An onboard IEEE 1394 (Firewire) port provides connectivity to the Phantom Omni haptic device. A spare VGA connector is used to attach a secondary display used for the VRML haptic scene rendering.

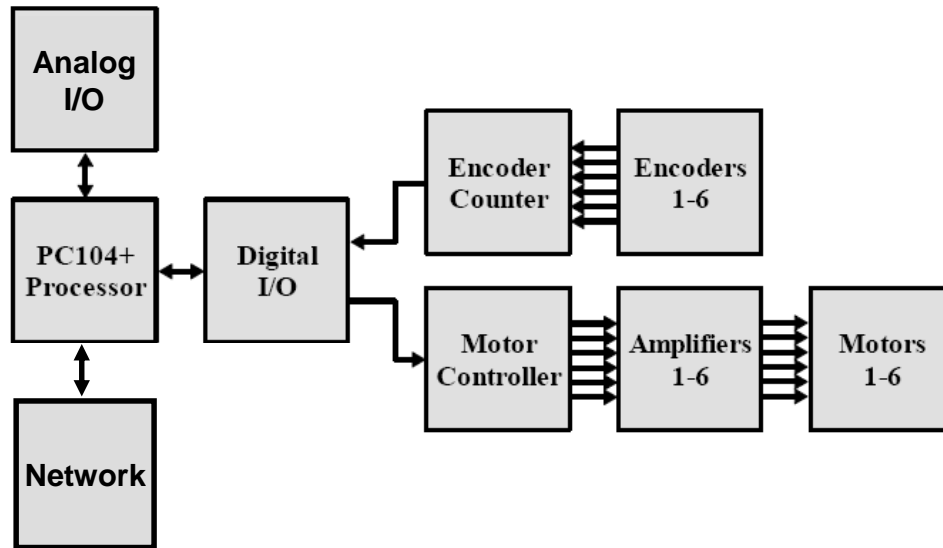


Figure 5-10: A block diagram of the xPC boards and interconnections.

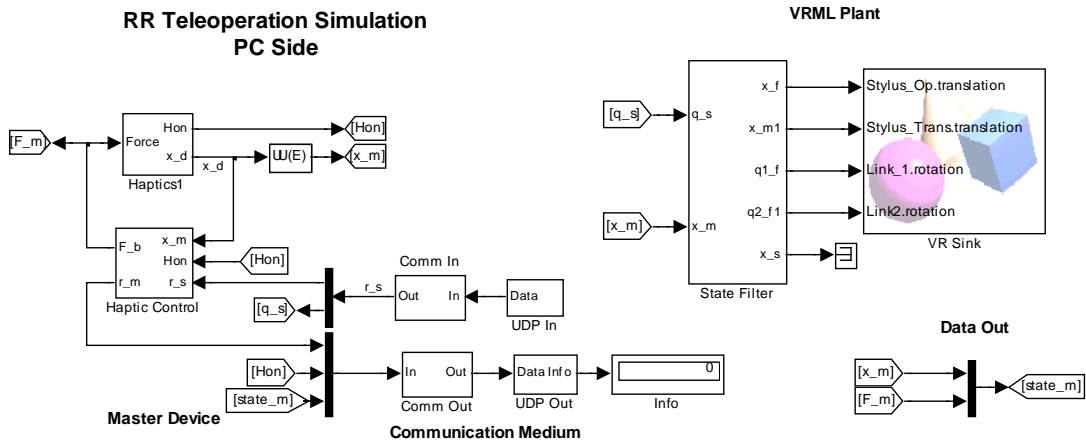
The slave system is an embedded PC-104 stack containing a CPU, power conditioning, and analog and digital interface boards (Figure 5-10). The onboard processor is an AMD ElanSC520 Pentium (486) microcontroller running at 133 MHz. It contains an onboard 10/100 Ethernet controller and has 64 MB of RAM. The PC-104 stack uses a parallel interface board to communicate a dedicated motor PWM generator and encoder reader board. Details on the capabilities and hardware/software specifications of these boards can be found in [16]. The motor PWM signals are transmitted to four off the self Copley current amplifier boards (403 and 4212z) which directly drive the manipulator actuators. The current demand and output of the currently amplifiers are monitored by the PC-104 stack system via an analog to digital board.

A wired Ethernet connection and dedicated router provides a fast and reliable communication link between the master and slave systems. Like in simulation, all network delay and packet loss is virtually introduced through the ProSense network simulator blocks instead of physically creating a lossy network connection. Using a

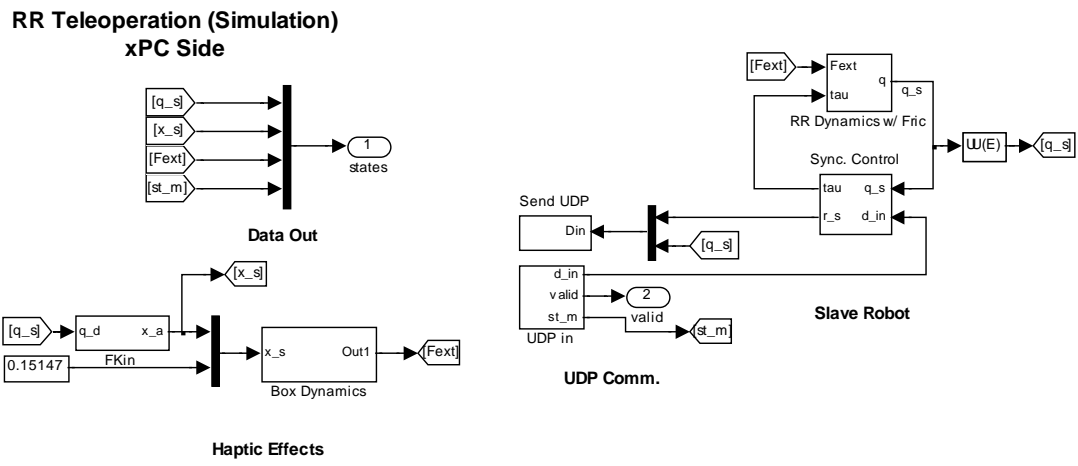
virtual network simulator is more reliable when attempting to create a repeatable unreliable network connection. The average latency and packet loss statistics are easily determine unlike when a real lossy physical network is used. The user datagram protocol (UDP) is used to send data along the network. Unlike TPC/IP, UDP does not guarantee that packets are delivered or arrive in the proper order to the receiving computer. However, these guarantees add significant overhead to the communication protocol significantly adding to network latency and memory requirements for implementation. The UDP protocol is simple and light weight by nature making it ideal for embedded and/or time sensitive applications. Multicasting and broadcasting capabilities are also supported in the UDP protocol making it ideal for groups of robotic systems.

5.3.2 Simulink Software

In this section, we will outline the new Simulink blocks used in the experimental setup. Figure 5-11 outlines the Simulink code used on the master and slave machines for real time implementation. The simulation code used in the previous section was designed to run in real time and is used for real time implementation when needed.



a)



b)

Figure 5-11: The Simulink block diagrams for a) the PC haptic side and b) RR manipulator side

However, some new blocks had to be introduced to accommodate for the UDP network and hardware interface layers which can be seen in Figure 5-12.

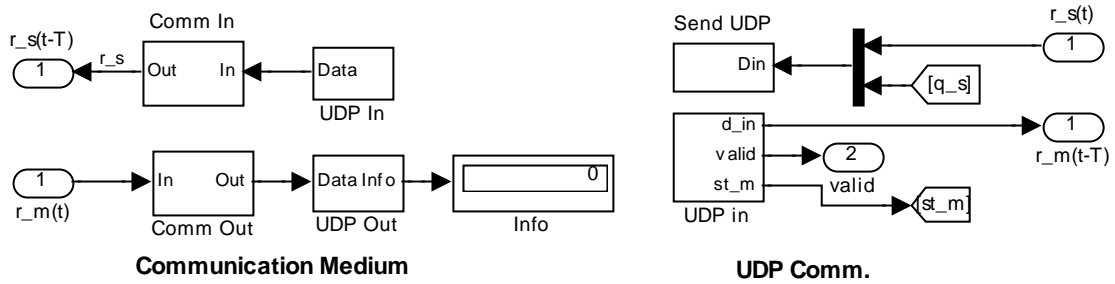


Figure 5-12: The master and slave UDP communication subsystem with ProSense network simulator

The UDP blocks create the link between the master and slave computers. The master UDP communication blocks contain both the ProSense network simulator and the UDP blocks. The UDP block must be given an IP address and port to listen and transmit data from. UDP packets are transmitted on the byte level, so the data type and size of the incoming information must be known to properly decode packets.

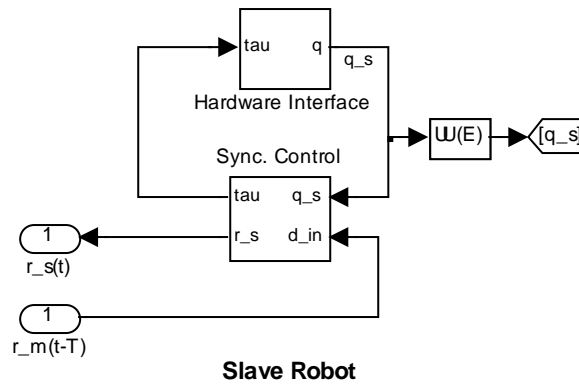


Figure 5-13: The slave manipulator control block and hardware/sensor interface

The xPC slave robot interface contains the synchronizing controller and hardware interface. Like in simulation, the synchronization control law contains the passive based output synchronization law and gradient projection adaptation laws developed in section 4. The hardware interface block contains two C style S functions which directly control the digital I/O card on the xPC stack. The two 24 bit parallel ports interface with the

custom PWM and encoder boards. The C style S function drivers for the digital I/O interface can be found in the appendix. A brief tutorial on creating non-inlined C style S functions is also included as a reference.

5.4 Test Cases

This section we will outline the many facets which may be used for testing a teleoperation scheme. In our testing we would like to expand this range to include several more common scenarios a remote robotic system is expected to experience. Figure 5-14 is a pictorial representation of some different testing scenarios. Each axis represents a variable which can be changed to assess the quality of a teleoperation scheme.

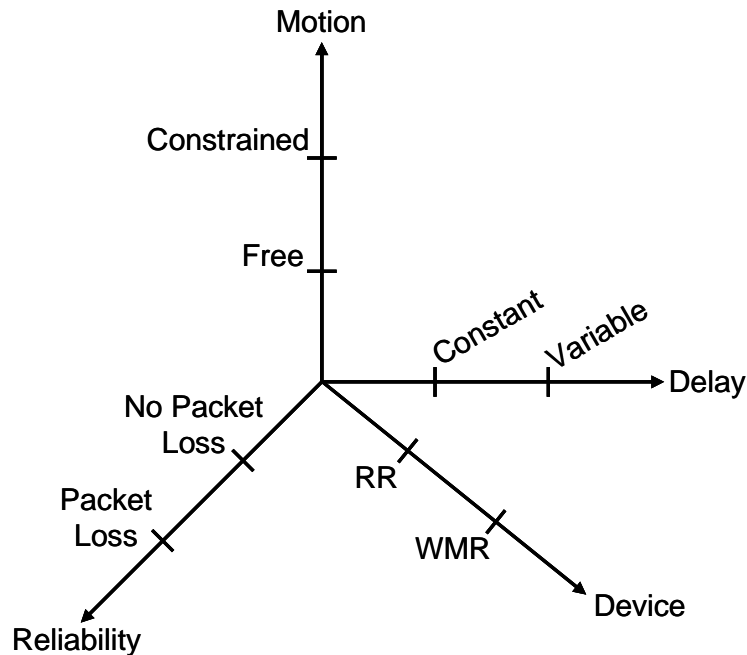


Figure 5-14: A pictorial representation of the test criteria for teleoperation schemes.

Typical testing of a teleoperation scheme utilizes constant time delay and no packet loss to test the stability of the controller. For example, packet loss can be a significant cause of controller instability when loss rates are high. In addition to varying

communication reliability and delay, we would like to assess the ability of the teleoperation scheme to handle different slave robotic systems. Analyzing these test criterions, a table of all the possible tests scenarios is created. A comparison of the performance of different slave systems with various time delay time will be discussed first. The possible tests are summarized in Table 5-6.

Device \ Delay	Constant	Time Varying
RR	E ₁	E ₂
WMR	E ₃	E ₄

Table 5-6: A summary of all possible tests varying delay type and slave device.

Similarly the device type can also be compared against the reliability of the communication channel. Table 5-7 summarizes these additional test possibilities.

Device \ Reliability	No Loss	Packet Loss
RR	E ₅	E ₆
WMR	E ₇	E ₈

Table 5-7: A summary of possible tests varying communication reliability and slave device.

Our goal is to illustrate the stability of the developed teleoperation scheme in an efficient manner. Of the introduced test possibilities, the following test cases are considered to verify our framework. In Table 5-8, we find the case number and a brief description of the case and network quality.

Case	Scenario	Device	Network Quality	Description
1	E ₁ & E ₅	RR	0.01 sec constant time delay, No packet loss	This will test algorithm's ability to compensate for network delays and overcome initial condition offsets. The manipulator will come into contact with a hard virtual and real structure
2	E ₁ & E ₆	RR	0.01 sec constant time delay, 55% Packet loss	This will illustrate the algorithms' robustness to time delay with packet loss. The RR end effector will come in contact with a virtual and real hard contact.
3	E ₂ & E ₇	WMR	0.001 sec delay, No packet loss	This will test the input/output synchronization ability of the algorithm. The kinematic coupling scheme is shown to compensate for position error despite an initial offset.
4	E ₂ & E ₇	WMR	0.1 sec constant time delay, 55% Packet loss	This test illustrates the robust capabilities of the algorithm in the presence of delay and packet loss. The WMR will come in contact with a virtual hard contact.
5	E ₄ & E ₈	WMR	Normal distributed time delay (mean = 0.4, std = 0.12), 55 % Packet loss	This test will demonstrate the algorithms ability to keep the master and slave safely coupled and stable despite extreme network conditions.

Table 5-8: The proposed test cases for testing the bilateral teleoperation scheme ability.

6 Results

In this section we will discuss the simulation and real-time output synchronization results. Each of the cases discussed above is highlighted again in Table 6-1. The delay type and communication reliability represent the network conditions from master to slave or visa versa. Total delay (from master back to master) is simply double of the delay values present here.

Case	Scenarios	Device	Delay Type	Communication Reliability
1	E ₁ & E ₅	RR	Constant 0.01 sec	No Loss
2	E ₁ & E ₆	RR	Constant 0.01 sec	55% Random Loss
3	E ₂ & E ₇	WMR	Constant 0.001 sec	0% Random Loss
4	E ₂ & E ₇	WMR	Constant 0.1 sec	55% Random Loss
5	E ₄ & E ₈	WMR	Normal Distribution mean = 0.4, std = 0.12	55% Random Loss

Table 6-1: A summary of the proposed test cases.

6.1 RR Results

In this case we discuss bilateral teleoperation for a RR manipulator. For trajectory consistency the haptic input device was virtually controlled by an embedded Matlab function to insure consistent movement between test cases. The slave device start with an initial offset with the coupling controller off. After a second, the controller is engaged and the slave and master positions should converge. The master device is commanded travel an ellipse described by the following equations.

$$x_d = -0.15 \cos(t) + 0.45$$

$$y_d = -0.20 \sin(t) - 0.30$$

The end effector will come in contact with a virtual and real hard contact during this trajectory. In the simulation, the box will have a damping and inertia as expressed in Table 5-3. The following control gains were used.

$$\lambda = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} 15 & 0 \\ 0 & 15 \end{bmatrix}$$

6.1.1 Case 1: Simulation Results

In the first simulation, a virtual RR manipulator was teleoperated by a virtual haptic device. The time delay for data being transmitted in either direction was set to a constant 0.01 second time delay without packet loss. This delay will produce a round trip time delay of approximately 0.02 seconds (neglecting computational delays caused by the controller).

After simulating the controller for 15 seconds, we get the following state and force outputs. The position and force profiles for the virtual master and slave systems are presented in Figure 6-1.

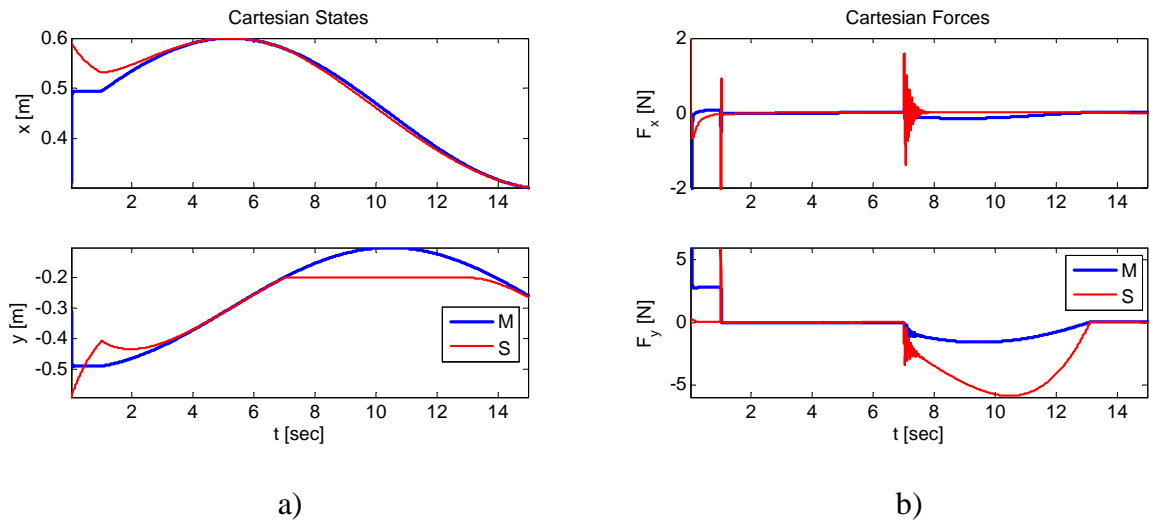


Figure 6-1: A comparison of the simulated master and slave a) Cartesian state and b) Cartesian forces for Case 1.

The simulation results present good tracking results for the Cartesian positions. The slave manipulator was able to converge to the master position despite an initial position offset. There is a slight lag between master and slave positions due to the communication induced time delay. The forces between master and slave resemble similar profiles despite a scaled difference. A small spike and oscillations appears in the force profile when the slave end effector encounters the virtual wall. Any oscillations are damped out with the synchronization controller within 1 second of contact. The torque profile for the slave manipulator is presented in Figure 6-2.

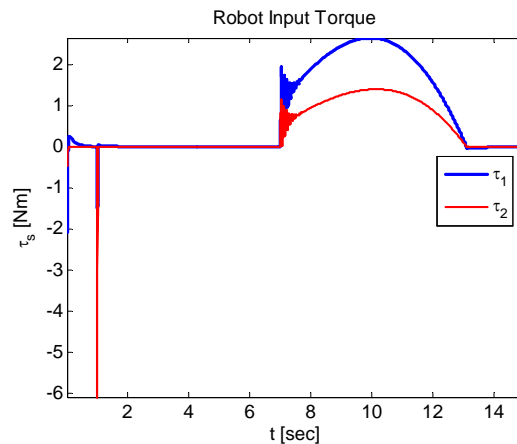


Figure 6-2: The torque profile for the simulated slave manipulator for Case 1.

The torque profiles are reasonable in size and can be easily achieved in a real time implementation framework. A large torque spike is present when the synchronization controller is turned on which is due to the large position error between master and slave. It is quickly dissipated as the master and slave states converge.

6.1.2 Case 1: Real time Results

During simulation, the chosen gain was found to be too high and resulted in chattering in the output torque. Although tracking results were good, a smaller set of

control gains was chosen as a trade off of position error for clean force and control signals. The lower control gains are:

$$\lambda = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

The virtual input was again provided to the Phantom Omni master device for 65 seconds. The position and force results for Case 1 are presented in Figure 6-3.

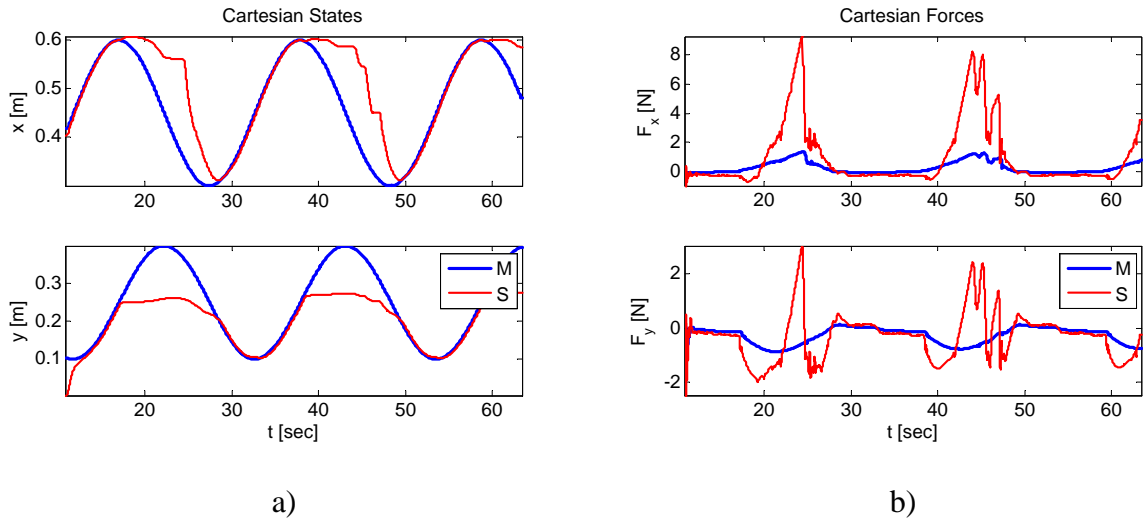


Figure 6-3: A comparison of the real master and slave a) Cartesian states and b) Cartesian forces for Case 1.

The real implementation of the output synchronization structure provided extremely good tracking results between the master and slave systems. The slave was able closely track the master position despite a constant time delay. The end effector of the slave device came in contact with a filled cardboard box twice during time 15 and 35. When the slave end effector came in contact with the environment, the system was able to provide noticeable tactile feedback to the user. The Cartesian force profiles tracked fairly well despite a difference in magnitude. Figure 6-4 represents the torque profile of the slave manipulator.

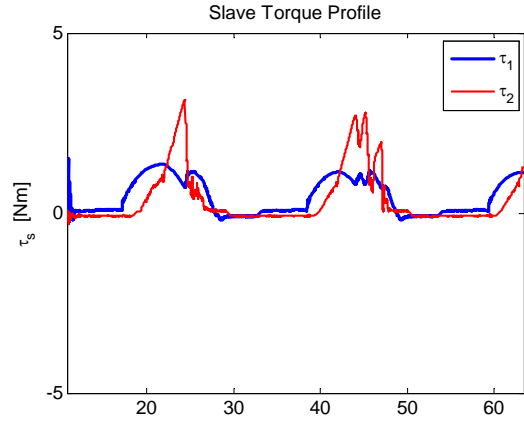


Figure 6-4: The torque profile for the real slave manipulator implementation for Case 1.

Once again the torque profile during teleoperation was found to be manageable for implementation on the slave robot system.

6.1.3 Case 2: Simulation Results

In the Case 2, the RR manipulator is command along the same elliptical path by the virtual haptic device. The simulated network conditions were a 0.01 second constant time delay with 55% random packet loss in the forward and backward communication lines.

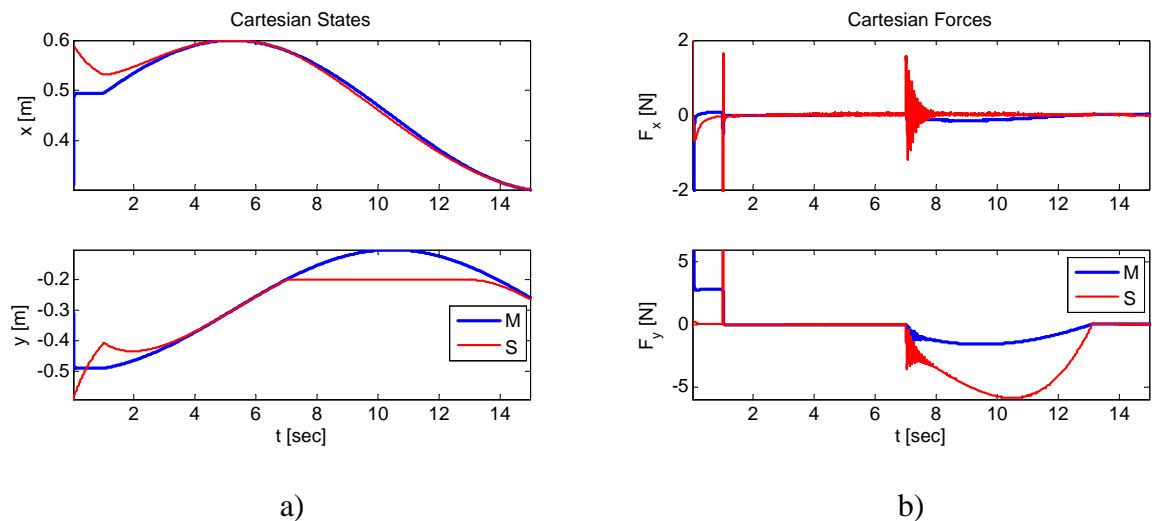


Figure 6-5: A comparison of the simulated master and slave a) Cartesian states and b) Cartesian forces for Case 2.

After simulating the controller for 15 seconds, we get the following state and force outputs (Figure 6-5).

Despite the high packet loss, the master and slave states were still able to converge and track. The force results also represent similar profiles despite a constant gain difference. A larger spike is present in the slave force profile when the end effector came in contact with the hard surface. This can be attributed to the packet loss induced delay. Again the oscillations created by the hard contact are quickly damped out within one second. A torque profile for the slave manipulator is presented in Figure 6-6.

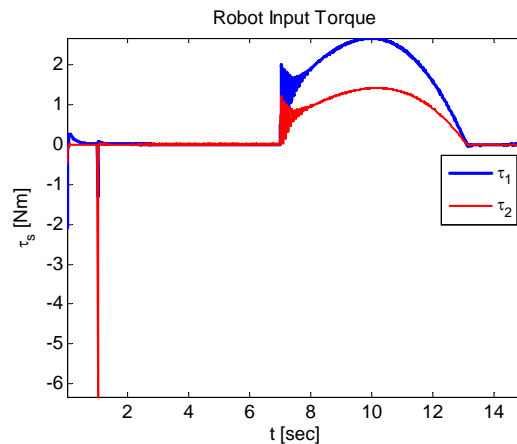


Figure 6-6: The real torque profile for the simulated slave manipulator for Case 2.

The slave torque profile is again reasonable for implementation on a real robotic system. A high torque appears due to the large position offset present between master and slave positions when the synchronization controller was turned on and commanded to converge to the master position.

6.1.4 Case 2: Real time Results

In case 2, the reduced set of control gains were also used as a compromise between high position accuracy and torque chatter.

$$\lambda = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

Figure 6-7 represents the Cartesian position and force profiles for the master and slave systems.

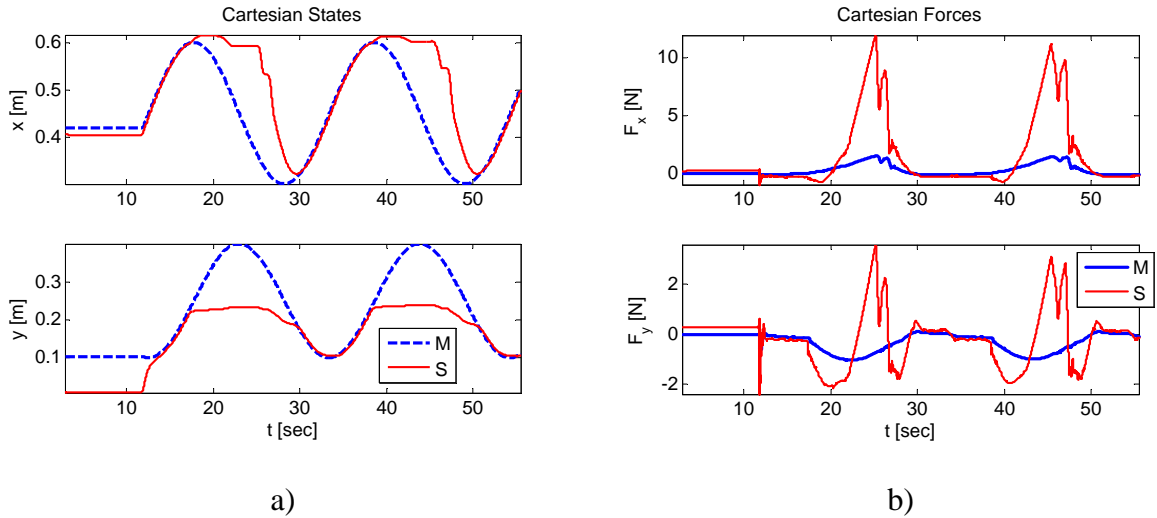


Figure 6-7: A comparison of the real master and slave a) Cartesian states and b) Cartesian forces for Case 2.

Good tracking performance was again seen between the master and slave end effector positions. The slave system was able to converge to the master position despite an unreliable network communication and position offset. At time 12 seconds, the kinematic coupling is turned on and the master and slave states converge. The slave end effector encountered a hard surface twice during this simulation at times 15 and 38 sec. The resulting force profiles showed an obvious increase in force at the slave end effector. The master force profile followed the slave profile well, but with a noticeable gain difference. The output torque profile of the slave manipulator was easily achievable by the robot hardware as seen from Figure 6-11. Peaks in output torque are present during initial setup when the slave robot came in contact with the rigid surface.

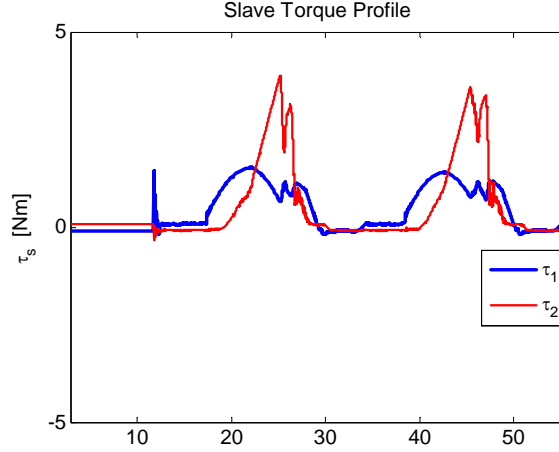


Figure 6-8: The torque profile for the real slave manipulator implementation for Case 2.

6.2 WMR Results

In this section, the teleoperation abilities for a wheeled mobile robot are explored. Cases 3, 4 and 5 will demonstrate the algorithms robustness to packet loss and constant and variable time delays.

The haptic device is command to following the following time varying Cartesian trajectories.

$$x = 3e - 4t^4 - 8.2e - 3t^3 + 0.052t^2 - 0.0351t + 9.4e - 3$$

$$y = 0$$

6.2.1 Case 3: Simulation Results

For Case 3, the WMR is teleoperated with a lossless communication channel and very low time delay. This test is meant to show the prove the proposed control scheme for differentially driven wheeled robots does in fact work. A position offset is introduced to prove the proposed algorithm can converge to the desired position in free space. The WMR is given an initial offset of $x = 0$ and $y = -0.05m$ and run for 15 seconds. The following control gains were used during simulation.

$$\lambda = \begin{bmatrix} 8 & 0 \\ 0 & 8 \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}$$

Figure 6-9 contains the Cartesian position and force tracking results for Case 3.

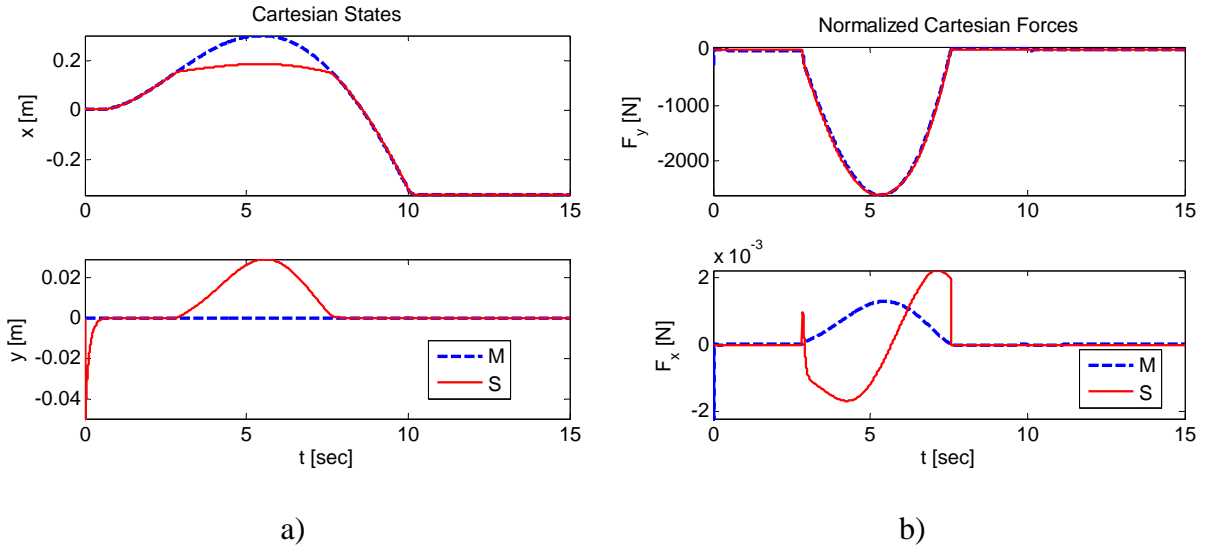


Figure 6-9: Simulation results for teleoperation of a WMR without time delay and packet loss.

In the absence of time delay, the master and slave systems have very good tracking results. Despite the initial y offset at the start of the simulation, the WMR slave robot successfully converged to the desired master trajectory. A spike in the y slave position appears while the WMR is driving backwards from the wall. This offset is due to the instability of the look ahead controller when driving backwards as pointed out in [15]. The WMR must instead quickly turn around and resume driving along the trajectory in the forward direction. The force results between master and slave force profiles are very closely matched. The Cartesian force figure represents normalized force profiles between the master and slave systems. The computed forces are normalized with respect to the maximum experienced force which occurred on the slave robot side. These large forces are a result of infinite actuation capacity and the strict enforcement of the no slip condition in the simulated dynamics. These large forces were not transmitted back the

virtual user. The Cartesian forces in the y direction are extremely small and are byproducts of the friction model of the block.

6.2.2 Case 4: Simulation Results

In Case 4, the stability capabilities of the proposed teleoperation scheme are demonstrated with respect to a lossy and delayed network conditions. Network delay is set to be a constant 0.1 second delay in either direction. A 0.1 second delay is typical for network communication which spans the entire length of the Earth. The network simulator was also programmed to drop random data packets at a rate of 55%. The haptic device was commanded in the predefined trajectory with an initial position of $x=0$ and $y=0$ and simulated for 15 seconds. The following control gains were used during simulation.

$$\lambda = \begin{bmatrix} 8 & 0 \\ 0 & 8 \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}$$

The position and force results for Case 4 are shown in Figure 6-10.

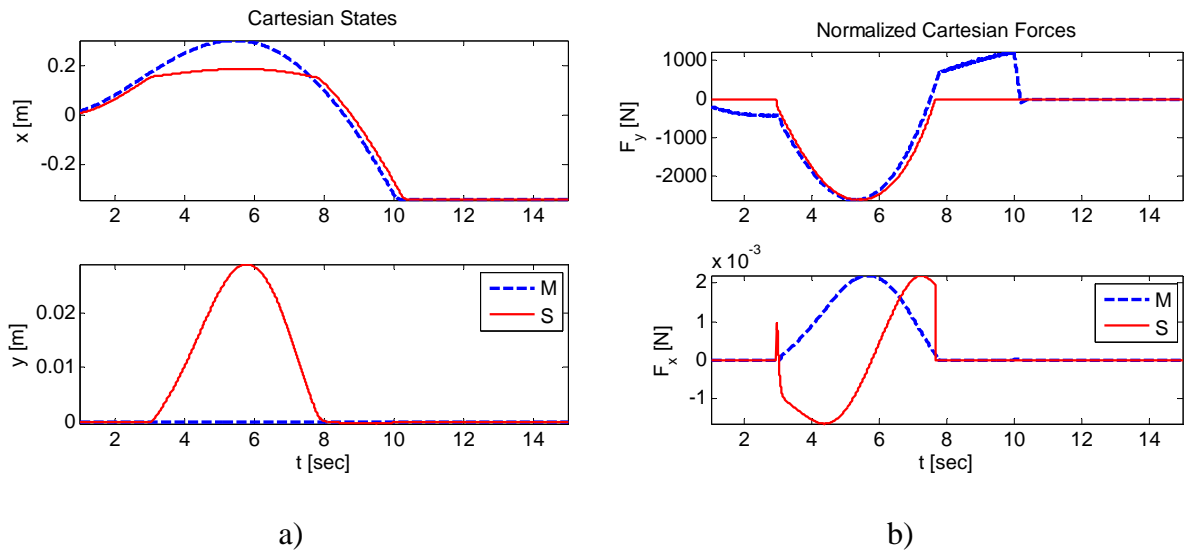


Figure 6-10: The WMR teleoperated with 0.1 constant time delay and 55% random packet loss.

Cartesian position tracking is again close between master and slave systems with a noticeable offset present due to the large constant time delay. Again, a spike is seen on the y position of the WMR due to the instability of look ahead control driving backwards. The normalized Cartesian forces show similar profiles. The y direction forces are again very small since the WMR is only commanded in the x direction. The residual force is due to the friction model on the surface of the box. The x directional forces for master and slave systems are good.

6.2.3 Case 5: Simulation Results

For the final test scenario, the WMR was teleoperated over an extremely lossy and delayed network environment. A variable network delay modeled by a normal distribution latency with an average value of 0.4 seconds and standard deviation of 0.12 seconds. This delay was mimicked in both receiving and transmitting directions. Data packets were randomly lost at a rate of 55%. The scenario proves that even in the most volatile of network conditions, the master and slave systems will still remain stable. The simulation was run for 15 seconds. The haptic device was commanded in the predefined trajectory with an initial position of $x = 0$ and $y = -0.05 m$ and simulated for 15 seconds. The gains λ and \mathbf{K} were set to the following values.

$$\lambda = \begin{bmatrix} 8 & 0 \\ 0 & 8 \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}$$

Figure 6-11 represents the simulated position and force results for Case 5.

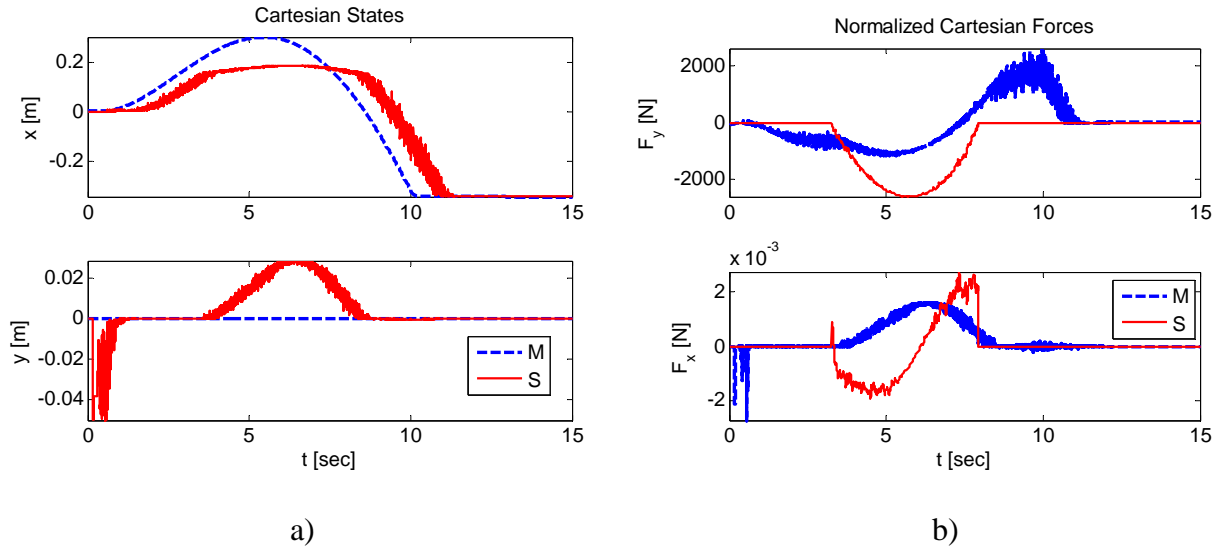


Figure 6-11: The WMR teleoperated under variable time delay (mean = 0.4 s, std = 0.12 s) and 55% packet loss.

The position results have an obvious offset in tracking performance due to the large variable delay and packet loss between master and slave. Despite an initial offset, the master and slave did states remained bounded and, in the case of the y Cartesian direction, tracked well. Position and force results are noisy but the controller did not destabilize despite the extreme network conditions.

7 Conclusions

7.1 Summary

In Chapter 1 we outlined the advantages of bilateral teleoperated robotic control and highlighted the difficulties associated implementing such a system in a real setting. These advantages can improve the precision and precision of the manipulation abilities of human operators. These advantages are extremely useful for a number of robotic

applications including remote navigation and telesurgery. Despite these advantages, a main disadvantage of such a scheme is that bilateral teleoperation can become unstable when the communication medium is delayed and inconsistent. We looked at the method of output synchronization proposed by [1] as a potential solution for teleoperation of two heterogeneous robot systems.

In Chapter 2 we presented the important mathematic tools used in the thesis. The concepts of skew symmetry and homogenous transformations where introduced for kinematic and dynamic analysis. The concept of passivity and its uses for creating complex control algorithms was touched on. Finally, the modeling and adaptive control of passive Lagrangian systems was developed for trajectory following applications.

In Chapter 3, we developed the kinematic and dynamic models for the planar revolute robotic arm and wheeled mobile robot.

In Chapter 3, the mathematical models for each of the proposed robotic manipulators are derived. Dynamics are derived using the Euler-Lagrangian formulation. A reduced set of feasible motion dynamics will also be found if the robotic system contains any kinematic constraints.

In Chapter 4, the output synchronization framework was introduced with the stability of the controller analyzed. A hybrid Cartesian bilateral teleoperation control scheme was developed to couple a simple mass system and the planar robot and wheeled mobile robot cases. An adaptive controller was used to estimate dynamic slave parameters in real time for improved tracking results. This hybrid controller has the ability to be turn the synchronization controller on and off at the press of a button.

In Chapter 5, we introduced the virtual environments for the planar arm and wheeled mobile robot. The simulation and real time code was introduced on a subsystem by subsystem basis. A set of experiments were introduced to test the capabilities of the hybrid controller. The free space and hard contact scenarios were proposed with delayed and lossy network conditions.

Chapter 6 contains the simulation and real time experiment results. The Cartesian position and force results are presented on a case by case basis. The results of each test was discussed and concluded upon.

7.2 Future Work

There are several ways in which the hybrid system presented in this thesis can be improved upon.

Implementation for Redundant Robotic Manipulators

The input/output synchronization framework has proven to be a powerful way of synchronizing two dissimilar systems. In this thesis we have only used robotic systems with non-redundant actuation. The first major goal would be to expand the output synchronization framework to work with redundant manipulators such as the wheeled mobile manipulator presented in [16]. It seems likely that with the use of the kinematically decoupled joint space decomposition (KD-JSD) proposed by [26] to map the Cartesian position into the redundant actuation space. A null space component could then be devised to implement a secondary cost or control algorithm.

Efficient implementation

The hybrid control algorithm devised in this thesis does run in real time; however the implementation can be improved on. In [27] a recursive implementation of the passivity

adaptive control scheme of [4] is presented. The proposed control strategy could be adaptive for faster implementation in simulation and real time bilateral teleoperation applications.

A composite adaptation law such as the ones presented in [17], could be implemented for faster convergence of the estimated parameters. The composite adaptation law combines the estimated output error with the actual sensor readings to speed up convergence significantly. Although the resulting controller is more complex, the increased convergence rate could allow for slow time steps to be used for the adaptation integration loop.

UDP Communication

In the Matlab/Simulink setting, UDP communication is handled as a background process to the running model. This means as the local control models become increasingly more complex and push the limit of the embedded CPU, the reliability of UDP communication will reduce. A different control architecture which equally weights the UDP communication and model processes might be considered for more robotic systems with more complex dynamics.

8 Bibliography

- [1] N. Chopra, M. W. Spong, and R. Lozano, "Synchronization of bilateral teleoperators with time delay," *Automatica*, vol. 44, p. 6, July 19 2007.
- [2] P. F. Hokayem and M. W. Spong, "Bilateral teleoperation: An historical survey," *Automatica*, vol. 42, p. 22, April 12 2006.
- [3] R. J. Anderson and M. W. Spong, "Bilateral Control of Teleoperators with Time Delay," *Transactions on Automatic Control*, vol. 34, p. 8, May 1989.
- [4] G. Niemeyer and J.-J. E. Slotine, "Stable Adaptive Teleoperation," *Journal of Oceanic Engineering*, vol. 16, p. 11, January 1991.
- [5] Y. Ye and P. X. Liu, "Improving Force Feedback Fidelity in Wave-Variable Based Teleoperation," in *IEEE International Conference on Robotics and Automation* Pasadena, CA, 2008, pp. 194 - 199.
- [6] J.-H. Ryu, D.-S. Kwon, and B. Hannaford, "Stable Teleoperation with Time-Domain Passivity Control," in *IEEE Transactions on Robotics and Automation*. vol. 20, 2004, pp. 365-373.
- [7] N. Chopra and M. W. Spong, "Adaptive Coordination Control of Bilateral Teleoperators with Time Delay," in *Conference on Decision and Control*, IEEE, Ed. Atlantis, Paradise Island, Bahamas, 2004, pp. 4540 - 4547.
- [8] G. Niemeyer and J.-J. E. Slotine, "Towards Force-Reflecting Teleoperation Over the Internet," in *International Conference on Robotics and Automation*, Leuven, Belgium, 1998, pp. 1909 - 1916.
- [9] N. Chopra, M. W. Spong, R. Ortega, and N. E. Barabanov, "On Tracking Performance in Bilateral Teleoperation," *Transactions on Robotics*, vol. 22, p. 6, August 2006.
- [10] M. W. Spong and N. Chopra, "Synchronization of networked Lagrangian Systems," *Lagrangian and Hamiltonian methods for nonlinear control*, p. 12, 2007.

- [11] D. Lee, O. Martinez-Palafox, and M. W. Spong, "Bilateral Teleoperation of a Wheeled Mobile Robot over Delayed Communication Network," in *International Conference on Robotics and Automation*, Orlando, Florida, 2006, p. 3298.
- [12] N. Sarkar, X. Yun, and V. Kumar, "Control of Mechanical Systems with Rolling Constraints: Application to Dynamic Control of Mobile Robots," University at Pennsylvania, Philadelphia, PA.
- [13] T. Fukao, H. Nakagawa, and N. Adachi, "Adaptive Tracking Control of a Nonholonomic Mobile Robot," *Transactions on Robotics and Automation*, vol. 16, p. 7, 2000.
- [14] Z. Li and J. Canny, *Nonholonomic Motion Planning*: Springer, 1992.
- [15] X. Yun and Y. Yamamoto, "Internal Dynamics of a Wheeled Mobile Robot," *Journal of Robotic Systems*, vol. 14, p. 12, 1997.
- [16] G. White, "Simultaneous Motion and Interaction Force Control of a Nonholonomic Mobile Manipulator," in *Mechanical Engineering*. vol. Masters of Science Buffalo: University at Buffalo, 2006, p. 200.
- [17] J.-J. E. Slotine and W. Li, *Applied Nonlinear Control*, 1 ed. Upper Saddle River: Prentice Hall International Inc., 1991.
- [18] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*: John Wiley & Sons, Inc., 2006.
- [19] D. M. Spong, R. Ortega, and R. Kelly, "'Comments on 'adaptive manipulator control: a case study' by J. Slotine and W. Li'," *IEEE Transactions on Automatic Control*, vol. 35, p. 2, 1990.
- [20] N. Chopra and Y. Liu, "Controlled Synchronization of Mechanical Systems," in *Dynamic Systems and Control Conference* Ann Arbor, Michigan, 2008, p. 8.
- [21] N. Chopra, "Output Synchronization of Networked Passive Systems." vol. Ph. D Urbana: University of Illinois at Urbana-Champaign, 2006, p. 155.
- [22] P. A. Ioannou and J. Sun, *Robust Adaptive Control*: Prentice-Hall, 1996.
- [23] J.-J. E. Slotine and W. Li, "Adaptive Manipulator Control: A Case Study," *IEEE Transactions on Automatic Control*, vol. 33, p. 8, November 1988.

- [24] X. Yun, V. Kumar, and N. Sarkar, "Control of multiple arms with rolling constraints," in *International Conference on Robotics and Automation*, IEEE, Ed. Nice, France, 1992, pp. 2193 - 2198.
- [25] "Sensable Technologies," 2008.
- [26] J. Park, W. Chung, and Y. Youm, "On Dynamical Decoupling of Kinematically Redundant Manipulators," in *International Conference on Intelligent Robots and Systems*, 1999, pp. 1495 - 1500.
- [27] G. Niemeyer and J.-J. E. Slotine, "Performance in Adaptive Manipulator Control," in *Proceedings on Decision and Control*, Austin, Texas, 1988, pp. 1585 - 1592.

Appendix A

This appendix contains the source code used for hardware interfacing with the wheeled mobile robot system.

A.1 Hardware Interface Drivers

This section provides the C code required for interfacing with Glenn White's custom PWM and encoder boards. This code has been updated for efficiency as a part of this thesis. The hardware interface and the software functionality have not been changed from [16].

C style S functions must be compiled within Matlab before being implemented in Simulink or Real-time Workshop. Mex file compilation is accomplished using the "mex" Matlab command. These mex files require access to the "io_xpcimport.h" header and source files for proper compilation. For example, to compile the function "encoder.c" the following command would be used.

```
mex -L<'C:\Program Files\MATLAB\R2006b\toolbox\rtw\targets\xpc  
\target\build\xpcblocks\include'> encoder.c
```

For more information on library and source code linking consult the help file for the "mex" function.

A.1.1 Send_PWM_Signals.c

```
// Send_PWM_Signals Non-inlined C style S function  
// Used to interface to Glenn White's PWM board
```

```

/*
 *Use the following to compile this driver.
 *mex -L<'C:\Program
Files\MATLAB\R2006b\toolbox\rtw\targets\xpc\target\build\xpcblocks\incl
ude'> Send_PWM_Signals.c
 */

#undef S_FUNCTION_NAME
#define S_FUNCTION_NAME Send_PWM_Signals
#define S_FUNCTION_LEVEL 2

#include <stddef.h>
#include <stdlib.h>
#include "simstruc.h" // Simulink data structure

#ifdef MATLAB_MEX_FILE
#include "mex.h"
#endif

#ifndef MATLAB_MEX_FILE
#include <windows.h>
#include "io_xpcimport.h"
#endif

/* Parameter List
 * 1) Base Address
 * 2) Sampling Time
 */

// Parameter Defines
#define BASE_ADDRESS_PARAM (ssGetSFcnParam(S,0))
#define BASE_ADDRESS ((uint_T) mxGetPr(BASE_ADDRESS_PARAM)[0])
#define SAMPLE_TIME_PARAM (ssGetSFcnParam(S,1))
#define SAMPLE_TIME (mxGetPr(ssGetSFcnParam(S,1))[0])

// Number of motors
#define NUM_CHANNELS (6)

// 8255-1 Offset Constants
#define PORT1_A (0)
#define PORT1_B (1)
#define PORT1_C (2)
#define PORT1_CONTROL (3)

// 8255-2 Offset Constants
#define PORT2_A (4)
#define PORT2_B (5)
#define PORT2_C (6)
#define PORT2_CONTROL (7)

/* Function Prototypes */

```

```

static void INIT_PWM(SimStruct *S);
static void DELAY();

/* Global Variables */
static char_T msg[256];

static void mdlInitializeSizes(SimStruct *S)
{
    uint_T i;

    #ifndef MATLAB_MEX_FILE
        #include "io_xpcimport.c"
    #endif

    ssSetNumSFcnParams(S, 2); /* Number of expected parameters */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)){
        sprintf(msg, "2 input args expected, %d passed",
                ssGetSFcnParamsCount(S));
        ssSetErrorStatus(S, msg);
        return;
    }

    ssSetNumInputPorts(S, NUM_CHANNELS);
    /* Number of inputs is now the number of channels. */

    /* We have no model outputs */
    ssSetNumOutputPorts(S, 0);

    /* Set direct feedthrough for all ports */
    for(i=0; i < NUM_CHANNELS; i++) {
        ssSetInputPortDirectFeedThrough(S,i,1); // Output is based on input
        ssSetInputPortWidth(S,i,1); // Set width to 1
        ssSetInputPortDataType(S, i, SS_UINT8); // Set inputs to uint8
    }

    ssSetNumSampleTimes(S, 1);
}

static void mdlInitializeSampleTimes(SimStruct *S)
{
    if (mxGetPr(SAMPLE_TIME_PARAM)[0] == -1.0) {
        ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
        ssSetOffsetTime(S, 0, FIXED_IN_MINOR_STEP_OFFSET);
    } else {
        ssSetSampleTime(S, 0, SAMPLE_TIME);
        ssSetOffsetTime(S, 0, 0.0);
    }
}

/* mdlStart should contain the hardware initialization routines
#define MDL_START
static void mdlStart(SimStruct *S)

```

```

{
#ifdef MATLAB_MEX_FILE
    /* Generated code calls function to initialize an PWM interface */
    INIT_PWM(S); /* This call accesses hardware */

#elif defined(MATLAB_MEX_FILE)
    /* During simulation, just print a message */
    if (ssGetSimMode(S) == SS_SIMMODE_NORMAL) {
        mexPrintf("\n Send_PWM_Signals.c: Simulating initialization\n");
    }

#endif
}

static void mdlOutputs(SimStruct *S, int_T tid)
{
#ifdef MATLAB_MEX_FILE
    InputUInt8PtrsType uPtrs = ssGetInputPortSignalPtrs(S,0);
    uint_T i,c;
    uint_T DataPort = BASE_ADDRESS + PORT2_B;
    uint_T AddressPort = BASE_ADDRESS + PORT2_C;

    //send motor data to atmel
    for (i = 0; i < NUM_CHANNELS; i++){
        //rl32eOutpB(port, *uPtrs[i]); // Test code

        //present data while CLK pin is high = invalid data
        rl32eOutpB(DataPort, *uPtrs[i]);

        //Present address data, make sure that CLK pin is not driven
low
        //addresses for 6 motors will just be 0->5
        rl32eOutpB(AddressPort, (i | 128));

        //set CLK pin low to indicate valid data
        rl32eOutpB(AddressPort, (i | 0));
        rl32eOutpB(AddressPort, (i | 0));
        rl32eOutpB(AddressPort, (i | 0));

        //set CLK pin high to indicate invalid data
        rl32eOutpB(AddressPort, (i | 128));

    }

#endif
}

/* Hardware Initialization Code */

static void INIT_PWM(SimStruct *S)

```

```

{
#ifdef MATLAB_MEX_FILE
    uint_T ControlPort = BASE_ADDRESS + PORT2_CONTROL;
    uint_T DataPort    = BASE_ADDRESS + PORT2_A;
    uint_T AddressPort = BASE_ADDRESS + PORT2_B;
    uint_T UnusedPort  = BASE_ADDRESS + PORT2_C;
    uint_T i = 0;

    // Set the Port1_A, B and C to outputs
    rl32eOutpB(ControlPort, 144); // (1000 0000) MSFLAG = 1

    for(i=0;i<254;i++){

        //Switch output pins from 0 to 255 to set up isolators
        rl32eOutpB(DataPort, 0);
        rl32eOutpB(AddressPort, 0);
        rl32eOutpB(DataPort, 255);
        rl32eOutpB(AddressPort, 255);
        rl32eOutpB(DataPort, 0);
        rl32eOutpB(AddressPort, 0);
        rl32eOutpB(DataPort, 255);
        rl32eOutpB(AddressPort, 255);
    }

    // Set the Port2_A, B and C to inputs
    //rl32eOutpB(port, 155);
#endif
}

static void DELAY()//DELAY(SimStruct *S, uint_T AddressPort, uint_T i,
uint_T Val)
{
#ifdef MATLAB_MEX_FILE
    uint_T ind;
    uint_T count = 0;

    for(ind = 0; ind < 35; ind++){
        //set CLK pin low to indicate valid data
        count = count + 1;
    }
#endif
}

// Set all of the PWM Channels to zero
static void mdlTerminate(SimStruct *S)
{
#ifdef MATLAB_MEX_FILE

    /* Send "0" to all motors */

#endif
}

#ifdef MATLAB_MEX_FILE    /* File being compiled as a MEX-file? */

```

```

#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfuns.h" /* Code generation registration function */
#endif

```

A.1.2 Read_Encoders.c

```

// Read_Encoder Non-inlined C style S function
// Used to interface to Glenn White's Encoder board
/*
 *Use the following to compile this driver.
 *mex -L<'C:\Program
Files\MATLAB\R2006b\toolbox\rtw\targets\xpc\target\build\xpcblocks\incl
ude'> Read_Encoders.c
 */

#undef S_FUNCTION_NAME
#define S_FUNCTION_NAME Read_Encoders
#define S_FUNCTION_LEVEL 2

#include <stddef.h>
#include <stdlib.h>
#include "simstruc.h" // Simulink data structure

#ifdef MATLAB_MEX_FILE
#include "mex.h"
#endif

#ifndef MATLAB_MEX_FILE
#include <windows.h>
#include "io_xpcimport.h"
#endif

/* Parameter List
 * 1) Base Address
 * 2) Sampling Time
 */

// Parameter Defines
#define BASE_ADDRESS_PARAM (ssGetSFcnParam(S,0))
#define BASE_ADDRESS ((uint_T) mxGetPr(BASE_ADDRESS_PARAM)[0])
#define SAMPLE_TIME_PARAM (ssGetSFcnParam(S,1))
#define SAMPLE_TIME (mxGetPr(ssGetSFcnParam(S,1))[0])
#define NUM_CHANNELS (7)

// 8255-1 Offset Constants
#define PORT1_A (0)
#define PORT1_B (1)
#define PORT1_C (2)
#define PORT1_CONTROL (3)

```

```

// 8255-2 Offset Constants
#define PORT2_A      (4)
#define PORT2_B      (5)
#define PORT2_C      (6)
#define PORT2_CONTROL (7)

/* Function Prototypes */
static void INIT(SimStruct *S);
static uint8_T SYNC_CLOCK(SimStruct *S);

/* Global Variables */
static char_T msg[256];

/* Initializations */
static void mdlInitializeSizes(SimStruct *S)
{
    // Variable Initializations
    uint_T num_channels;
    int_T port;

    #ifndef MATLAB_MEX_FILE
        #include "io_xpcimport.c"
    #endif

    ssSetNumSFcnParams(S, 2); /* Number of expected parameters */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)){

        sprintf(msg, "2 input args expected, %d passed",
                ssGetSFcnParamsCount(S));
        ssSetErrorStatus(S, msg);
        return;
    }

    // Set Port Number and Sampling Times
    ssSetNumInputPorts(S, 0);
    ssSetNumOutputPorts(S, NUM_CHANNELS);
    ssSetNumSampleTimes(S,1);

    for( port=0; port<NUM_CHANNELS; port++){
        ssSetOutputPortWidth(S,port,1); // One signal per
port
        ssSetOutputPortDataType(S, port, SS_DOUBLE); // Signals are
doubles
    }

    /*
    Note that by default, the ADC block has no direct feedthrough.
    The ADC output is calculated based on values read from hardware,
    not from data obtained from another block.
    */
}

static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, SAMPLE_TIME);
}

```

```

    ssSetOffsetTime(S, 0, 0.0);
}

// mdlStart should contain the hardware initialization routines
#define MDL_START
static void mdlStart(SimStruct *S)
{
#ifdef MATLAB_MEX_FILE
    /* Generated code calls function to initialize an hardware */
    INIT(S); /* This call accesses hardware */

#elif defined(MATLAB_MEX_FILE)
    /* During simulation, just print a message */
    if (ssGetSimMode(S) == SS_SIMMODE_NORMAL) {
        mexPrintf("\n Read_Encoders.c: Simulating initialization\n");
    }

#endif
}

static void mdlOutputs(SimStruct *S, int_T tid)
{
    real_T *y = ssGetOutputPortRealSignal(S,0);
    uint_T i;

#ifdef MATLAB_MEX_FILE

    /* Decoding Variables */
    uint8_T    Address = 0;
    uint8_T    EncoderDataLowBytes[12];
    uint8_T    EncoderDataHighBytes[12];
    uint32_T   EncoderIntegerVals[6];
    real_T     EncoderDoubleVals[6];

    /* Port Variables */
    uint_T     AddressPort   = BASE_ADDRESS + PORT1_B;
    uint_T     LowBytePort   = BASE_ADDRESS + PORT1_A;
    uint_T     HighBytePort  = BASE_ADDRESS + PORT1_C;

    if( SYNC_CLOCK(S) == 0 )

    /* Read encoder values if we're insync */
    {
        //First data now valid
        for(i=0; i<12; i++)
        {
            //only read Address once, takes a lot of time
            //CLK bit will be set, so bitwise "and" the Address byte
with 127 to clear it
            Address = r132eInpB(AddressPort)&127;
            //read data
            EncoderDataLowBytes[Address] = r132eInpB(LowBytePort);
            EncoderDataHighBytes[Address] = r132eInpB(HighBytePort);
            //Wait for CLK to be unasserted
            while( (r132eInpB(AddressPort) & 128) == 128 ){}
            //Wait until CLK is asserted again -> data is now valid

```

```

        while( (r132eInpB(AddressPort) & 128) == 0 ){
    }

    //Compose integer data values - exactly the same 32bit values
as those on the avr
    for(i=0; i<6; i++)
    {
        EncoderIntegerVals[i] =
((uint32_T)EncoderDataLowBytes[i*2]) +

((uint32_T)EncoderDataHighBytes[i*2])<<8) +

((uint32_T)EncoderDataLowBytes[i*2+1])<<16) +

((uint32_T)EncoderDataHighBytes[i*2+1])<<24);
    }

    //Compose double data values and add negative sign
    //At this point EncoderDoubleVals indicates the +/- absolute
encoder counts
    for(i=0; i<6; i++)
    {
        if( (EncoderIntegerVals[i] >= 0) && (EncoderIntegerVals[i]
< 2147483648) )
        {
            EncoderDoubleVals[i] = (real_T)EncoderIntegerVals[i];
        }
        else
        {
            EncoderDoubleVals[i] = -1.0*((real_T)(4294967295-
EncoderIntegerVals[i]+1));
        }
    }

    /* no error occured, send the good data out and error bit=0 */
    for (i = 0; i < NUM_CHANNELS-1; i++){
        y[i] = (real_T) EncoderDoubleVals[i];
    }
    y[NUM_CHANNELS-1] = (real_T) 0.0;    // Error bit = 0
}

/* Data is not insync */
else
{

    /* An error occured, send zeros out and error bit=1 */
    for (i = 0; i < NUM_CHANNELS-1; i++){
        y[i] = (real_T) 0.0;
    }
    y[NUM_CHANNELS-1] = (real_T) 1.0;    // Error bit = 1

}

#else
/* simulation code should just output zeroes to all channels */
for (i = 0; i < NUM_CHANNELS; i++){

```

```

        y[i] = 0.0;
    }
#endif
}

#ifdef MATLAB_MEX_FILE

/* Hardware Initialization Code */
static void INIT(SimStruct *S)
{
    uint_T port = BASE_ADDRESS + PORT1_CONTROL;

    // Set the Port1_A, B and C to inputs
    r132eOutpB(port, 155);
}

/* Synchronize to CLK signal */
/* Note: A benchmark indicated that the following while() loops
would be executed about 15-18 times, thus reading the CLK pin 15-18
times before it changed state. This means that we can make a
maximum of 15-18 port reads safely in 7e-6 seconds. This is
plenty of time since we only need to make about 3-4 port reads
to capture all the data.
*/

static uint8_T SYNC_CLOCK(SimStruct *S)
{
    uint32_T TimeoutCounter = 0;
    uint_T AddressPort = BASE_ADDRESS + PORT1_B;

    //Check if CLK is unasserted (low)
    /* Start TimeoutCounter, exit if TimeoutCounter > ~40 */
    while( ((r132eInpB(AddressPort) & 128) == 0) )
    {
        if(TimeoutCounter > 40){/*timed out*/return (uint8_T) 1;}
        TimeoutCounter++;
    }

    //Check if CLK is asserted (high)
    TimeoutCounter = 0;
    while( ((r132eInpB(AddressPort) & 128) == 128) )
    {
        if(TimeoutCounter > 40){/*timed out*/return (uint8_T) 1;}
        TimeoutCounter++;
    }

    //Check again if CLK is unasserted (low)
    /* The following while loop should now encompass the entire
unasserted
* CLK cycle. At the moment the CLK pin is asserted (high), this
* while loop will be exited and the first data will be valid
* and can be read in.
*/

```

```

TimeoutCounter = 0;
while( ((r132eInpB(AddressPort) & 128) == 0) )
{
    if(TimeoutCounter > 40){/*timed out*/return (uint8_T) 1;}
    TimeoutCounter++;
}

return (uint8_T) 0;

}
#endif

static void mdlTerminate(SimStruct *S){}

#ifdef MATLAB_MEX_FILE /* File being compiled as a MEX-file? */
#include "simulink.c" /* MEX-file interface mechanism */
#else
#include "cg_sfun.h" /* Code generation registration function */
#endif

```