8-31-2011

Cooperative impedance control with time-varying stiffness

Matthew Courtney

Matthew Courtney

Candidate

Mechanical Engineering

Department

This thesis is approved, and it is acceptable in quality and form for publication:

Approved by the Thesis Committee:

[illegible] ,Chairperson

[illegible]

[illegible]

Cooperative Impedance Control with Time-Varying Stiffness

by

Matthew Courtney

B.S., Mechanical Engineering, University of New Mexico, 2008

THESIS

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science
Mechanical Engineering

The University of New Mexico

Albuquerque, New Mexico

July 2010

Dedication

To my loving fiancee, Kim.

Acknowledgments

I would like to thank my advisor, Dr. Gregory Starr, for the opportunity to (as my fiancee says)"go to work everyday and play." His internship provided me with thought-provoking problems to tackle and the experience necessary to be a marketable engineer. Without Dr. Starr's support and guidance, this thesis would not have been possible.

Stipend and equipment support for this thesis work was made available by the DOE University Research Program in Robotics (URPR),
via grant # DE-FG52-04NA25590.

Cooperative Impedance Control with Time-Varying Stiffness

by

Matthew Courtney

ABSTRACT OF THESIS

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science
Mechanical Engineering

The University of New Mexico

Albuquerque, New Mexico

July 2010

Cooperative Impedance Control with Time-Varying Stiffness

by

Matthew Courtney

B.S., Mechanical Engineering, University of New Mexico, 2008

M.S., Mechanical Engineering, University of New Mexico, 2010

Abstract

The focus of much automation research has been to design controllers and robots that safely interact with the environment. One approach is to use impedance control to specify a relationship between a robot's motion and force and control a grasped object's apparent stiffness, damping, and inertia. Conventional impedance control practices have focused on position-based manipulators — which are inherently non-compliant — using constant, task-dependent impedances. In the event of large trajectory tracking errors, this implementation method generates large interaction forces that can damage the workcell. Additionally, these position-based devices require dedicated force/torque sensors to measure and apply forces. In this paper, we present an alternative impedance controller implemented on cooperating torque-based manipulators. Through the use of time-varying impedance parameters, this controller limits the interaction forces to ensure harmless manipulation. Successful completion of transport and insertion tasks demonstrated the effectiveness of the controller.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In a traditional industrial setting, all locations are known and error tolerances are low; this is a structured environment. The robots appropriate for these environments are stiff and accurate, controlled by a position-based device. Take one of these heavy, rigid robots off of the assembly line and onto the battlefield or into the operating room and havoc will ensue. With an unstructured environment, manipulation tasks will inevitably collide with or fail to grasp objects, breaking valuable commodities along the way - possibly including the robots themselves. To properly use a rigid, position-based robot in this situation, a large amount of sensing and computation must be dedicated to perception of the environment. Rather than incorporate cameras and sensors and additional controllers for each new subsystem, it may be preferable to design a controller that can work in both settings.

Using robots for combat has been the "wave of the future" for years. But, only recently, the notions of robot couriers, barricade builders, and scouts are being realized. In these situations, smaller is better: easier to move and easier to hide. So, what happens when a heavy or awkward object must be manipulated? The payload limits of an individual robot will be exceeded and cooperation is needed between multiple platforms to accomplish the job. It is clear that compliance and cooperation leads to much needed versatility in robotic tasks.

1.2 Literature Review

1.2.1 Force Control

Decades of research have been focused on the challenge of controlling robot-environment interactions. Robust and reliable manipulation tasks have been successfully demonstrated using various force control methods. The two fundamental approaches are force tracking control (explicit force control) and impedance control (implicit force control).

Explicit force control tracks a reference force to perform a desired task. By properly following the desired contact forces, the manipulator is ensured a safe interaction with the environment. However, this method is only designed for manipulation during contact. While in free space, the manipulator must use an alternate controller. Also, the desired position trajectory of the manipulator is not directly known. The expected path can be inferred, only through knowledge of the environment stiffness and position and the manipulator's dynamic model.

Conversely, impedance control specifies the interaction dynamics by relating the contact forces to the motions of the manipulator and thereby indirectly controls the environmental forces. The concept of stiffness control was first proposed by Salisbury [2] in 1980 and later formalized by Hogan [3] in 1985 to include the control of all three forms of impedance: stiffness, damping, and inertia. Through impedance control, the stiffness, damping, and inertial parameters of the robot can be tuned to be rigid for accurate maneuvers or compliant for interactions with the environment depending on the desired task. However, due to the indirect nature of this scheme, the environment must be known to accurately control the interaction forces. When impacts with unexpected obstacles cause large errors in the trajectory, large forces are generated - which can still be damaging.

Both implicit and explicit force controllers have their advantages. Previous work on force-tracking impedance control seeks to combine the benefits of both control schemes. In 1997, Seraji and Colbaugh demonstrated an adaptive scheme that modified the reference trajectory to maintain a desired force [4]. Two methods were proposed: the first uses the Model-Reference Adaptive Control (MRAC) framework to select the reference position based on the desired force tracking dynamics; the second estimates the position and stiffness of the environment and uses them to calculate a reference trajectory. Both methods utilize an accurate position controller to realize the new reference trajectory. Tsumugiwa *et. al.*, in 2002, estimated the human arm stiffness during robot-human interactions and varied the desired damping parameters accordingly [5]. This method only uses inertial and damping terms in the controller; therefore, it is strictly passive and unable to track free space trajectories. Jung *et. al.* extended this idea in 2004 using a two phase adaptive approach that alternates between stiffness and admittance control depending on the contact state of the manipulator [6]. Jung's method was designed as a hybrid position/force controller. Hybrid control is a method that partitions the Cartesian axes into force controlled and position controlled. Though this scheme is useful for position-based manipulators, the interaction forces must arise in the predetermined 'force axes' to be utilized.

1.2.2 Cooperative Control

When manipulating large, heavy, and awkward objects, it may also be necessary to use multiple robots to safely accomplish motions. This is especially the case where a single manipulator's acceptable payload is exceeded. Rather than purchasing a larger robot for the task, one could utilize cooperation between multiple, smaller arms. In the hope of incorporating compliance and cooperation, much theoretical work has been performed. The first work in multi-robot impedance control was performed by Schneider and Cannon in 1992[7]. They formulated a dual-arm Object Impedance Control (OIC) scheme that propagates the necessary forces to the manipulators and thereby controls a commonly-held object's impedance. Their experimental work used planar two-link Selective Compliant Articulated Robotic Arm (SCARA) manipulators connected to the 'object' via pins so that the reactions at the manipulator/object interfaces were strictly forces. This control algorithm was modified by later researchers to improve trajectory tracking, and was applied to other robots. Moosavian and Papadopoulos [8] presented an extension of OIC called Multiple Impedance Control (MIC). This control scheme not only imposes an impedance on the object but on the manipulator end-effectors as well. Simulations of both OIC and MIC were performed using a single arm to demonstrate the improved behavior of the MIC law in free space and in contact with the environment. Many other variations for cooperative control have been formulated, including: extensions of hybrid impedance control [9], inner force and outer PID control loops [10], and adaptive control [11], to name a few. Each of these methods, however, is inappropriate for the use on a torque-based manipulator. All of these controllers, as well as most found in the literature, utilize force/torque sensors to estimate the contact forces. Commonly mounted to the robot wrist, only interactions with the tool will provide measurements suitable for feedback control. These forces and torques can be sensed by a back-drivable torque-based manipulator without an explicit sensor by examining the disturbance torques. Additionally, the previous works in cooperative compliance focus on position-based implementation which adds further unnecessary complexity to the system.

Another aspect of cooperative control unites manipulators and mobile bases. To increase a robotic arm's versatility, it is often mounted upon a mobile platform for extra degrees of freedom. Much research has been performed on coordination and cooperation between manipulator and base. Early research focused on decoupling the mobility and manipulability of the mobile robot system and supplying different coordinated trajectories for each subcomponent. However, more recently unified systems have become desirable: [12] [13] [14] [15], to name a few. Hootsman proposed the simple extended Jacobian method that will be used here. This method modifies the standard Jacobian matrix to include the mobile platform's kinematics - effectively unifying the manipulator and base into a single system.

1.3 Problem Statement

Despite the broad body of research, the problem remains to derive a cooperative time-varying impedance controller that does not rely on a force/torque sensor. This improvement will afford greater controller versatility while ensuring safe handling in uncertain conditions. The workcell at UNM is one of only a handful with two redundant, torque-based, back-drivable manipulators - the Whole Arm Manipulator (WAM) from Barrett Technologies. In the few years that the WAM has been available, a large amount of research and support has arisen. From conference workshops to journal articles, from academic theses to R&D corporations, the development of the WAM has reinvigorated the force-control community. Taking advantage of this unique resource, we propose a control scheme that eliminates the force/torque sensor at the wrist, has the ability to track forces under impedance control, and employs cooperation between multiple manipulators. Conventional impedance controllers use task-dependent impedances that remain constant for the duration of the task. By enforcing a nonlinear, time-varying impedance function, the new controller varies the impedance parameters throughout the task to reduce the interaction forces. Limited work has been performed on time-varying impedance. Research by Tsetserukou [16] most closely resembles the work performed here, however it is joint impedances that are varied, and the controller is restricted to single arm applications. No method could be found that uses time-varying impedances in task coordinates capable of autonomous control; nor was there evidence of cooperative force-tracking impedance controllers in the literature.

1.4 Thesis Organization

We began by presenting motivation for improvements in existing impedance control theory. The remainder of the paper is structured as follows: Chapter 2 formulates the cooperative impedance control law for torque-based manipulators and provides an overview of the theory needed for generalized implementation; the workcell's hardware and software are described in Chapter 3; Chapter 4 delves into the specific control tasks and results for computer simulation and experimental validation; and the conclusions and future work are laid out in Chapter 5.

Chapter 2

Theory

The previous chapter discussed the need for a single robot to cooperatively interact with other robots and the environment to safely perform a wide variety of manipulation tasks. Documented approaches to force-tracking impedance control focus on single position-based manipulators with constant impedance parameters. In this work, an alternative method of cooperative force-tracking impedance control is proposed, for use on torque-based manipulators, that simplifies implementation and provides safe, versatile handling of objects in uncertain environments.

2.1 Control Law

Cooperative impedance control is a manipulation method wherein multiple robots interact with a common object to simultaneously manipulate the object and give it a desired dynamic response - whether it be high stiffness for accuracy or low stiffness for compliance. Just as in single-arm impedance control, force relates to motion through a second order linear differential equation,

$$\mathbf{F} = \mathbf{M_d}\ddot{\mathbf{x}} + \mathbf{K_v}\dot{\mathbf{x}} + \mathbf{K_p}\mathbf{x}. \tag{2.1}$$

However, in object impedance control, the motion is defined relative to a grasped object rather than to a single manipulator's end effector. The forces in (2.1) are a combination of all forces on the object, including but not limited to: gravity, inertia, external, centripetal, and Coriolis effects. To obtain the desired object response, cooperating manipulators apply additional forces to compensate for errors in the motion trajectory and contact with the environment. Initially these impedance forces ($\mathbf{F}$) are computed with respect to the object. Then they are propagated to the robots through a grasp matrix ($\mathbf{W}$). Finally, the Jacobian transpose is used to convert each arm's force into joint torques to be realized by the manipulator's motors.

Let us begin to formulate the cooperative impedance control law by first stating the desired object behavior,

$$\mathbf{M_d}\ddot{\mathbf{x}} + \mathbf{K_v}\dot{\mathbf{e}} + \mathbf{K_p}\mathbf{e} = \mathbf{F}. \tag{2.2}$$

Define trajectory tracking error ($\mathbf{e}$) as the difference between the actual and desired position, $\mathbf{x_{des}} - \mathbf{x_{act}}$. The stiffness ($\mathbf{K_p}$), damping ($\mathbf{K_v}$) and inertia ($\mathbf{M_d}$) coefficients can be selected to impart a desired impedance to the object. In this approach, interaction forces between the object and the environment are generated from position and velocity tracking errors and object accelerations.

Using the definition of $\mathbf{e}$, we can solve (2.2) for the object's acceleration,

$$\ddot{\mathbf{x}} = -\mathbf{M_d^{-1}}(\mathbf{K_v}\dot{\mathbf{e}} + \mathbf{K_p}\mathbf{e} - \mathbf{F_{ext}}). \tag{2.3}$$

The equation of motion for an object moving in a Cartesian coordinate frame, which relates inertial forces to external forces, is given as,

$$\mathbf{M_0}\ddot{\mathbf{X}} + \mathbf{B_0} = \mathbf{F_{ext}} + \mathbf{F_{robot}}. \tag{2.4}$$

To clarify the notation: the object's motion ($\mathbf{X}$) is a concatenation of its position ($\mathbf{x}$) and orientation ($\boldsymbol{\theta}$); the error in $\mathbf{X}$ is $\mathbf{E}$; the angular velocity ($\boldsymbol{\omega}$) is the rate of change of the orientation; the inertia matrix ($\mathbf{M_0}$) refers to the object's mass (m) and

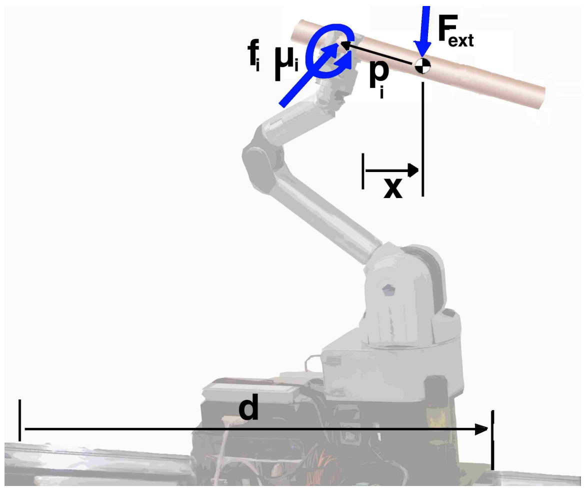


Figure 2.1: Impedance Control Diagram

inertia tensor ($\mathbf{I}$); body forces ($\mathbf{B_0}$) are caused by gravitational acceleration ($\mathbf{g}$) and centripetal and Coriolis effects; external forces ($\mathbf{F_{ext}}$) are generated during contact with the environment and are represented by an equivalent force ($\mathbf{f_{ext}}$) and moment ($\boldsymbol{\mu}_{ext}$) acting on the object; the total robot force ($\mathbf{F_{robot}}$) relates the i^{th} manipulator's forces ($\mathbf{f_i}$) and moments ($\boldsymbol{\mu}_\mathbf{i}$) to the object frame through a grasp vector ($\mathbf{p_i}$). $\mathbf{I_{3x3}}$ and $\mathbf{0_{3x3}}$ are the identity and zero matrices, respectively. These terms are calculated by the following equations,

$$\mathbf{M_0} = \begin{bmatrix} m\mathbf{I_{3x3}} & \mathbf{0_{3x3}} \\ \mathbf{0_{3x3}} & \mathbf{I} \end{bmatrix} \tag{2.5}$$

$$\mathbf{B_0} = \begin{bmatrix} -m\mathbf{g} \\ \boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega} \end{bmatrix} \tag{2.6}$$

$$\mathbf{F_{ext}} = \begin{bmatrix} \mathbf{f_{ext}} \\ \boldsymbol{\tau}_\mathbf{ext} \end{bmatrix} \tag{2.7}$$

$$\mathbf{F_{robot}} = \begin{bmatrix} \Sigma\mathbf{f_i} \\ \Sigma(\mathbf{p_i} \times \mathbf{f_i}) + \Sigma\boldsymbol{\mu}_\mathbf{i} \end{bmatrix}. \tag{2.8}$$

Taking the acceleration, $\ddot{\mathbf{X}}$, of (2.3), substituting it into (2.4), and solving for the total robot force, we obtain the force necessary to impose the impedance behavior set forth in (2.2),

$$\mathbf{F_{robot}} = -\mathbf{M_0}\mathbf{M_d}^{-1}(\mathbf{K_v}\dot{\mathbf{E}} + \mathbf{K_p}\mathbf{E} - \mathbf{F_{ext}}) + \mathbf{B_0}(\dot{\mathbf{X}}, \mathbf{X}) - \mathbf{F_{ext}}. \tag{2.9}$$

2.2 Extension to Multiple Manipulators

For cooperative tasks, the control force computed in (2.9) must be partitioned between all robots. A grasp matrix ($\mathbf{W}$) is used to accomplish this. This matrix combines the forces at the grasp locations ($\mathbf{F_{cmd}}$) and computes the equivalent force acting on the object frame ($\mathbf{F_{robot}}$),

$$\mathbf{F_{robot}} = \mathbf{W}\mathbf{F_{cmd}} \tag{2.10}$$

$$\mathbf{W} = \left[\begin{array}{cc|cc|c} \mathbf{I_{3x3}} & \mathbf{0_{3x3}} & \mathbf{I_{3x3}} & \mathbf{0_{3x3}} & \vdots \; \cdots \\ \mathbf{P_1} & \mathbf{I_{3x3}} & \mathbf{P_2} & \mathbf{I_{3x3}} & \end{array}\right]. \tag{2.11}$$

$\mathbf{P_i}$ is a skew symmetric matrix that performs the cross product between $\mathbf{p_i}$ and $\mathbf{f_i}$; $\mathbf{F_{cmd}}$ is a column vector concatenating the forces and moments caused by the manipulators at the grasp points,

$$\mathbf{P_i} = \begin{bmatrix} 0 & -p_{i3} & p_{i2} \\ p_{i3} & 0 & -p_{i1} \\ -p_{i2} & p_{i1} & 0 \end{bmatrix} \tag{2.12}$$

$$\mathbf{F_{cmd}} = \begin{bmatrix} \mathbf{f_1} \\ \boldsymbol{\mu_1} \\ \mathbf{f_2} \\ \boldsymbol{\mu_2} \\ \vdots \end{bmatrix}. \tag{2.13}$$

Theoretically, the inverse of the grasp matrix computes the necessary manipulator forces to impose a desired object force. However, because $\mathbf{W}$ is a non-square matrix, we must use the right pseudoinverse of the grasp matrix, $(\mathbf{W}^{+})$, to perform this calculation. This pseudoinverse exists because the grasp matrix always has full row rank,

$$\begin{aligned}
\mathbf{F_{cmd}} &= \mathbf{W}^{+}\mathbf{F_{robot}} && (2.14)\\
\mathbf{F_{cmd}} &= \mathbf{W}^{+}\{-\mathbf{M_0}\mathbf{M_d^{-1}}(\mathbf{K_v}\dot{\mathbf{E}}+\mathbf{K_p}\mathbf{E}-\mathbf{F_{ext}})+\mathbf{B_0}(\dot{\mathbf{X}},\mathbf{X})-\mathbf{F_{ext}}\} && (2.15)\\
\mathbf{W}^{+} &= \mathbf{W^T}(\mathbf{W}\mathbf{W^T})^{-\mathbf{1}}. && (2.16)
\end{aligned}$$

Although (2.15) embodies the object's desired dynamics, it does not consider the dynamic effects of the manipulators. Manipulator dynamics are modeled by the following equation,

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}}+\mathbf{C}(\dot{\mathbf{q}},\mathbf{q})\dot{\mathbf{q}}+\mathbf{G}(\mathbf{q})=\boldsymbol{\tau}-\mathbf{J^T}\mathbf{f}+\boldsymbol{\tau}_{\mathbf{n}}, \qquad (2.17)$$

in which $\mathbf{M}(\mathbf{q})$ is the robot's inertia matrix, $\mathbf{C}(\dot{\mathbf{q}},\mathbf{q})$ accounts for the Centripetal and Coriolis torques, and $\mathbf{G}(\mathbf{q})$ adds the gravitational effects. In (2.17), the force exerted on the environment $(\mathbf{f})$ is converted into a moment using the Jacobian transpose. This exerted force is the decoupled command force found in (2.15). $\boldsymbol{\tau}_n$ is a set of joint torques that act in the nullspace of the Jacobian; these torques generate no forces at the robot end-effector and only exist if the manipulator degrees of freedom are greater than the controlled degrees of freedom (in general, 6). Therefore, each robot's joint torques, necessary to impart a specified impedance on the object while compensating for the object and manipulator dynamic models, are given by

$$\begin{aligned}
\boldsymbol{\tau} = {} & \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}}+\mathbf{C}(\dot{\mathbf{q}},\mathbf{q})\dot{\mathbf{q}}+\mathbf{G}(\mathbf{q})+\\
& \mathbf{J^T}\mathbf{W}^{+}[-\mathbf{M_0}\mathbf{M_d^{-1}}(\mathbf{K_v}\dot{\mathbf{E}}+\mathbf{K_p}\mathbf{E}-\mathbf{F_{ext}})+\\
& \mathbf{B_0}(\dot{\mathbf{X}},\mathbf{X})-\mathbf{F_{ext}}]+\boldsymbol{\tau}_{\mathbf{n}}. \qquad (2.18)
\end{aligned}$$

2.3 General Implementation

2.3.1 Forward Kinematics

The first step in realizing (2.18) is to accurately calculate the object's position and orientation with respect to a world coordinate frame. Transformation operators are one such method for solving this kinematics problem. Each transform consists of a rotation matrix ($\mathbf{R}$), a position vector ($\mathbf{p}$), and a scaling factor (1). Denote the homogeneous transformation matrix of frame {B} with respect to frame {A} as ${}^{\mathbf{A}}_{\mathbf{B}}\mathbf{T}$. These homogeneous transform matrices have the special properties that ${}^{\mathbf{C}}_{\mathbf{A}}\mathbf{T} = {}^{\mathbf{C}}_{\mathbf{B}}\mathbf{T}{}^{\mathbf{B}}_{\mathbf{A}}\mathbf{T}$ and ${}^{\mathbf{A}}_{\mathbf{B}}\mathbf{T}^{-\mathbf{1}} = {}^{\mathbf{B}}_{\mathbf{A}}\mathbf{T}$. Therefore, the matrices can be combined to determine the relationship between two frames that are not necessarily adjacent. Figure 2.2 shows an example workcell with the relevant coordinate frames. Through transformation operators, we

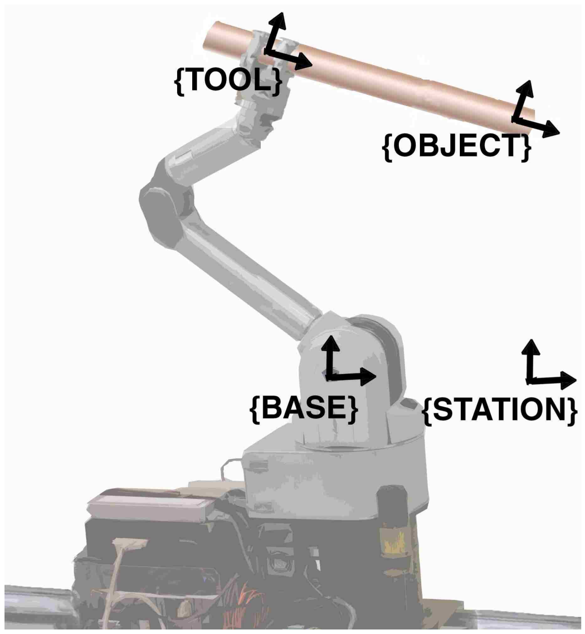


Figure 2.2: Frame Assignment Diagram

find that the object frame {O} is related to the stationary, world coordinate frame

{S} (called the station frame) by

$$ {}^{\mathbf{S}}_{\mathbf{O}}\mathbf{T} = {}^{\mathbf{S}}_{\mathbf{B}}\mathbf{T}\,{}^{\mathbf{B}}_{\mathbf{T}}\mathbf{T}\,{}^{\mathbf{T}}_{\mathbf{O}}\mathbf{T}. \tag{2.19} $$

To calculate each manipulator's transformation between the base {B} and tool {T} frames, we use the standard Denavit-Hartenberg method. This technique assigns link frames and defines them using a table of link parameters. The parameters include (a_i) the length of the common normal between two link frames, (α_i) the angle about the common normal from z_{i-1} to z_i, (d_i) the distance along z_{i-1} to the common normal, and (θ_i) the angle about z_{i-1} from x_{i-1} to x_i.

Using these parameters, we are able to formulate transformation matrices between adjacent joint frames,

$$ {}^{\mathbf{i-1}}_{\mathbf{i}}\mathbf{T} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i)\cos(\alpha_i) & \sin(\theta_i)\sin(\alpha_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i)\cos(\alpha_i) & -\cos(\theta_i)\sin(\alpha_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.20} $$

The relationship between the station frame and the base frame (${}^{\mathbf{S}}_{\mathbf{B}}\mathbf{T}$) and the tool frame and object frame (${}^{\mathbf{T}}_{\mathbf{O}}\mathbf{T}$) is subject to the workcell configuration and specific experiment. The transformations for our specific hardware arrangement will be addressed in Chapters 3 and 4. After all necessary transformations are determined, (2.19) can be computed.

2.3.2 Error Calculations

The next stage of computation determines the position and velocity error in the object's trajectory. Different methods were used to calculate the translational and rotational errors. First redefine the transformation matrix into a more computationally usable vector form,

$$\mathbf{T} = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{n} & \mathbf{o} & \mathbf{a} & \mathbf{p} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.21}$$

The desired and actual translations can be compared using the standard Euclidean difference:

$$\mathbf{d} = \mathbf{p_{des}} - \mathbf{p_{act}}. \tag{2.22}$$

The rotations, however, are more complex. The relationship between two 3-space rotations can be found from

$$\mathbf{R_\Delta} = \mathbf{R_{des}} \mathbf{R_{act}}^T. \tag{2.23}$$

Using the notation from (2.21) and comparing the resulting off-diagonal terms in (2.23), we find that the rotational error is

$$\boldsymbol{\partial} = \frac{\mathbf{1}}{\mathbf{2}}(\mathbf{n_{act}} \times \mathbf{n_{des}} + \mathbf{o_{act}} \times \mathbf{o_{des}} + \mathbf{a_{act}} \times \mathbf{a_{des}}). \tag{2.24}$$

The full derivation of this equation can be found in Appendix B.1.

From [17], a skew symmetric matrix of angular velocities is given by

$$\boldsymbol{\Omega} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} = \dot{\mathbf{R}}\mathbf{R^T}. \tag{2.25}$$

Thus the angular velocities are the resulting elements of $\boldsymbol{\Omega}$. Angular velocity error can then be calculated with the difference,

$$\dot{\boldsymbol{\partial}} = \boldsymbol{\omega}_{\mathbf{des}} - \boldsymbol{\omega}_{\mathbf{act}}. \tag{2.26}$$

2.3.3 Impedance Forces

In (2.2) we calculate the stiffness, damping, and inertia forces caused by the object's trajectory errors. In an ideal impedance controller, the desired trajectory is closely tracked during free-space motions (high stiffness) and the forces are minimized during environmental interactions (low stiffness). A problem with typical impedance controllers is that impedance gains remain constant for an entire task. If a compliant interaction is desired, the motion has poor tracking performance - even in free-space. Conversely, if an accurate motion is desired, large forces are applied during contact. Task-dependent impedances have been used to avoid this problem, but these have relied on either a predefined impedance trajectory that is strictly time-dependent or on a state (contact or free-space) observer with abrupt adjustments. Both of these are incapable of quickly and stably adjusting to unexpected collisions. Continuous variation of the impedance parameters is needed to cause both stiff and compliant interactions during a single task. Because the goal is to allow interactions with an unstructured environment, the variance cannot simply be a time-dependent adjustment; it must correspond to a controlled variable that fluctuates with task state. We propose a solution to this problem by varying the controller's impedance parameters with respect to the tracking error. Our response behavior is achieved in the force-error graph of Figure 2.3.

This relationship maintains the impedance concept of relating motions to forces and is governed by the equation

$$F = \text{sign}(e) F_{max}[1 - \exp(-\frac{|e|}{\xi})]. \tag{2.27}$$

The correlation is defined by two constants: the maximum force (F_{max}), chosen to limit the force on the environment caused by trajectory tracking errors, and the rise constant (ξ), chosen for a desired stiffness near the origin. From Hooke's law, the

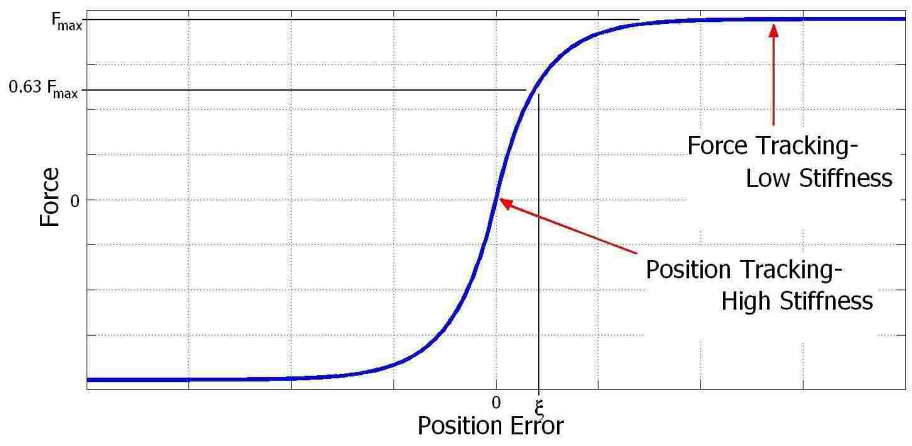


Figure 2.3: Proposed Relationship Between Force and Error

stiffness is obtained by $K = \frac{\partial F}{\partial e}$. Which, when $e << 1$, can be estimated by

$$K_O = \frac{F_{max}}{\xi}. \tag{2.28}$$

We know that, when the error equals the rise constant, the exerted force will be $0.63F_{max}$; which is another possible method for choosing ξ. With this saturating force function, the maximum stiffness force will be tracked once the error threshold has been reached. By using this particular stiffness relationship, there is always a reactionary force driving the position error to zero.

2.3.4 Jacobian

The impedance forces found from the object errors must next be converted into joint torques using the transpose of the Jacobian matrix. The Jacobian matrix ($\frac{\partial X}{\partial q}$) was found using the kinematic parameters of the robot. In [18], Paul defines a method whereby the Jacobian is found during the forward kinematic computations. First calculate the intermediate transformations,

$${}^{\mathbf{7}}_{\mathbf{i}}\mathbf{T} = {}^{\mathbf{7}}_{\mathbf{6}}\mathbf{T}{}^{\mathbf{6}}_{\mathbf{5}}\mathbf{T} \ldots {}^{\mathbf{i+1}}_{\mathbf{i}}\mathbf{T}. \tag{2.29}$$

Again denoting the transformation matrix by (2.21), we begin by formulating the manipulator's 6x7 Jacobian matrix that relates differential joint-space to differential task-space with respect to the tool frame ($^{\mathbf{T}}\mathbf{J}_{\mathbf{manip}}$). The i^{th} column vector of the manipulator's tool frame Jacobian ($^{\mathbf{T}}\mathbf{J}_{\mathbf{manip,i}}$) is computed using the following equation for revolute joints,

$$^{\mathbf{T}}\mathbf{J}_{\mathbf{manip,i}} = \begin{bmatrix} (\mathbf{p} \times \mathbf{n})_{\mathbf{z}} \\ (\mathbf{p} \times \mathbf{o})_{\mathbf{z}} \\ (\mathbf{p} \times \mathbf{a})_{\mathbf{z}} \\ \mathbf{n_z} \\ \mathbf{o_z} \\ \mathbf{a_z} \end{bmatrix}, \tag{2.30}$$

where the vectors **n**, **o**, **a**, and **p** are obtained from $^{\mathbf{7}}_{\mathbf{i}}\mathbf{T}$. Next, the RTU translational degree of freedom must be added to the Jacobian. This translational joint axis is parallel to the x-axis of the manipulator base frame, so a differential change in the RTU joint corresponds to a differential change in the x-axis. This means that the column vector of the Jacobian relating to the RTU is simply

$$^{\mathbf{B}}\mathbf{J}_{\mathbf{RTU}} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{2.31}$$

By transforming (2.31) into the tool frame and appending the manipulator Jacobian, the system Jacobian is

$$^{\mathbf{T}}\mathbf{J} = \begin{bmatrix} \begin{bmatrix} ^{\mathbf{B}}_{\mathbf{T}}\mathbf{R} & \mathbf{0_{3x3}} \\ \mathbf{0_{3x3}} & ^{\mathbf{B}}_{\mathbf{T}}\mathbf{R} \end{bmatrix} {^{\mathbf{B}}\mathbf{J}_{\mathbf{RTU}}} & {^{\mathbf{T}}\mathbf{J}_{\mathbf{manip}}} \end{bmatrix}. \tag{2.32}$$

2.3.5 Recursive Newton-Euler Dynamics

To incorporate the manipulator dynamics into the control algorithm of (2.18), we utilized the recursive Newton-Euler method proposed by Luh, Walker, and Paul[19]. This inverse dynamics method uses manipulator parameters (inertia, mass, link lengths, etc.) to compute the necessary joint torques for a given set of joint positions, velocities, and accelerations,

$$\mathrm{RNE}(\ddot{\mathbf{q}}, \dot{\mathbf{q}}, \mathbf{q}) = \boldsymbol{\tau}. \tag{2.33}$$

Computing the torque is a two stage process. First, the velocities and accelerations of the center of mass are computed for each link. The motion is propagated iteratively outward from $i = 1 \rightarrow 7$ using (2.34-36),

$$\begin{aligned}
\boldsymbol{\omega}_\mathbf{i} &= {}^\mathbf{i}\mathbf{R}^\mathbf{T}\left[\boldsymbol{\omega}_\mathbf{i-1} + \mathbf{z_0}\dot{\boldsymbol{\theta}}_\mathbf{i}\right] && (2.34)\\
\dot{\boldsymbol{\omega}}_\mathbf{i} &= {}^\mathbf{i}\mathbf{R}^\mathbf{T}\left[\dot{\boldsymbol{\omega}}_\mathbf{i-1} + \mathbf{z_0}\ddot{\boldsymbol{\theta}}_\mathbf{i} + \boldsymbol{\omega}_\mathbf{i-1} \times (\mathbf{z_0}\dot{\boldsymbol{\theta}}_\mathbf{i})\right] && (2.35)\\
\dot{\mathbf{v}}_\mathbf{i} &= {}^\mathbf{i}\mathbf{R}^\mathbf{T}\dot{\mathbf{v}}_\mathbf{i-1} + \dot{\boldsymbol{\omega}}_\mathbf{i} \times \mathbf{p}_\mathbf{i} + \boldsymbol{\omega}_\mathbf{i} \times (\boldsymbol{\omega}_\mathbf{i} \times \mathbf{p}_\mathbf{i}). && (2.36)
\end{aligned}$$

The base angular velocity ($\boldsymbol{\omega}_0$) and acceleration ($\dot{\boldsymbol{\omega}}_0$) are zero, and the base linear acceleration ($\dot{\mathbf{v}}_\mathbf{0}$) is set equal to upward gravitational acceleration ($\begin{bmatrix} 0 & 0 & 9.81 \end{bmatrix}^T$).

The second stage computes the forces and torques acting at the center of mass of each link due to its linear and angular accelerations and propagates them throughout the manipulator to determine the resulting joint torques. Here, inward iterations (2.37-42) are performed from $i = 7 \rightarrow 1$,

$$\begin{aligned}
\dot{\mathbf{v}}_\mathbf{ci} &= \mathbf{v}_\mathbf{i} + \boldsymbol{\omega}_\mathbf{i} \times \mathbf{r}_\mathbf{i} + \boldsymbol{\omega}_\mathbf{i} \times (\boldsymbol{\omega}_\mathbf{i} \times \mathbf{r}_\mathbf{i}) && (2.37)\\
\mathbf{F}_\mathbf{i} &= m_i\dot{\mathbf{v}}_\mathbf{ci} && (2.38)\\
\mathbf{N}_\mathbf{i} &= \mathbf{I}_\mathbf{ci}\dot{\boldsymbol{\omega}}_\mathbf{i} + \boldsymbol{\omega}_\mathbf{i} \times (\mathbf{I}_\mathbf{ci}\boldsymbol{\omega}_\mathbf{i}) && (2.39)\\
\mathbf{f}_\mathbf{i} &= {}^\mathbf{i+1}\mathbf{R}\mathbf{f}_\mathbf{i+1} + \mathbf{F}_\mathbf{i} && (2.40)\\
\mathbf{n}_\mathbf{i} &= {}^\mathbf{i+1}\mathbf{R}\mathbf{n}_\mathbf{i+1} + \mathbf{p}_\mathbf{i} \times \mathbf{f}_\mathbf{i} + \mathbf{N}_\mathbf{i} + \mathbf{r}_\mathbf{i} \times \mathbf{F}_\mathbf{i} && (2.41)\\
\boldsymbol{\tau}_i &= \mathbf{n}_\mathbf{i}^\mathbf{T}({}^\mathbf{i}\mathbf{R}^\mathbf{T}\mathbf{z_0}). && (2.42)
\end{aligned}$$

The torques computed using the Newton-Euler method account for the left hand side of (2.17).

2.3.6 Nullspace

Redundancy, in manipulation, is defined by having more degrees of freedom available than are being specified. In our system, each WAM+RTU has eight available axes. However, the Cartesian forces and moments necessary to control the object's impedance sum to only six. Therefore, two redundant degrees of freedom allow for an infinity of physical configurations. To control the redundant degrees of freedom without affecting the manipulator tool frame, the null torques ($\boldsymbol{\tau}_n$) must not generate forces at the tip. Joint torques relate to tip forces through the Jacobian transpose,

$$\mathbf{J}^{\mathbf{T}}\mathbf{F} = \boldsymbol{\tau}. \tag{2.43}$$

Rearranging (2.43) and applying the redundancy torque criterion, we find that the null torques reside in the nullspace of the Jacobian's inverse transpose,

$$\mathbf{J}^{-\mathbf{T}}\boldsymbol{\tau}_{\mathbf{n}} = \mathbf{0}. \tag{2.44}$$

To generate these torques, we begin by finding functions for the desired nullspace variables,

$$\Phi_1 \;=\; {}^{s}x(q) - q_1 \tag{2.45}$$

$$\Phi_2 \;=\; q_6. \tag{2.46}$$

The two redundant control variables were chosen as (1) the distance between the tool frame and the RTU along the x axis of the station frame (to keep the RTU's separated) and (2) the orientation of the fifth WAM joint (to maintain a desired 'elbow' angle). The Jacobian is then computed for each equation to find the relationship between these kinematic functions and their controlling joint torques,

$$J_{\Phi_1} = \frac{\partial \Phi_1}{\partial \mathbf{q}} = \frac{\partial^s x}{\partial q} - \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.47}$$

$$J_{\Phi_2} = \frac{\partial \Phi_2}{\partial \mathbf{q}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \tag{2.48}$$

The computations for both equations are simple, noting that the first term in (2.45) is the first row of the manipulator Jacobian with respect to the base frame. The joint torques can then be computed using proportional control and the Jacobian transpose,

$$\tau_{\Phi_i} = J_{\Phi_i}^T [K_{null,i}(\Phi_{des} - \Phi)]. \tag{2.49}$$

Nullspace damping is unnecessary because it is supplied by the inherent friction in the system. To ensure that the redundancy control torques do not affect the object forces, we project them into the nullspace (Appendix B.3),

$$\boldsymbol{\tau}_{\mathbf{n}} = (\mathbf{I} - \mathbf{J^T}(\mathbf{JJ^T})^{-1}\mathbf{J})\boldsymbol{\tau}_{\boldsymbol{\Phi}}. \tag{2.50}$$

2.3.7 Frame Considerations

Throughout the many computations comprising the control law, it is necessary to ensure consistency of coordinate frames. The calculations must reflect the frames in which the values are defined. For example, the impedance forces are derived from the object position relative to the station frame: so the impedance forces are also in this frame. However, the grasp matrix is computed using $\mathbf{p_i}$, which is relative to the object frame. To utilize equation (2.14) for computing the manipulator command forces, the robot forces must be converted to the object frame using the appropriate transform (${}^{\mathbf{O}}_{\mathbf{S}}\mathbf{T}$). Then, to generate the joint torques, the manipulator forces must be converted into the same coordinate frame as the Jacobian (Base or Tool). Without frame consistency, the control law equation is invalid. Nuances, such as this one, must be accounted for when the control law is realized on a particular workcell. The specific implementation methods, equations, and parameters will be presented in Chapters 3 and 4 where the workcell software and hardware are detailed.

Chapter 3

Hardware and Software

When applying (2.18) to a physical system, many cooperating hardware and software components are involved. The dual arm workcell at the University of New Mexico has two 7-DOF Barrett Whole Arm Manipulators (WAM) with three-fingered Barrett Hands. Each is mounted atop a Robot Transport Unit (RTU) for increased mobility. As seen in Figure 3.1, several communication protocols are used between the components. MathWorks software is used to unite these components and perform rapid controller prototyping, computer simulations, and hardware-in-the-loop experiments.

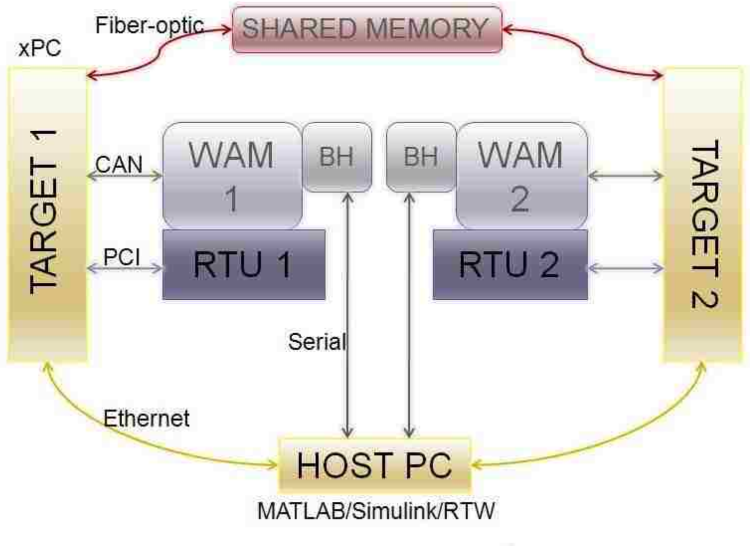


Figure 3.1: Workcell Diagram

3.1 Whole Arm Manipulators

The WAMs provide an ideal platform for controller development in that all necessary kinematic and dynamic properties are available. These robots are differentially cable driven; placing most of the motors at the robot base allows for light weight linkages and minimizes joint friction. The robots were also designed to be backdriveable: an external force on any link can cause joint motion. Each manipulator has seven degrees of freedom, is modeled after the human arm, and is commanded by a set of motor currents. This particular combination of manipulator characteristics makes the WAM ideal for interacting with the environment through force control algorithms without the need for explicit force/torque sensors. Rather than alter the desired trajectory to accommodate environmental forces, the WAM's design complies with these disturbances naturally. Rudimentary, open-source, C-based controllers are available for these robots to demonstrate example applications, but their open architecture design encourages novel, state-of-the-art controller implementation.

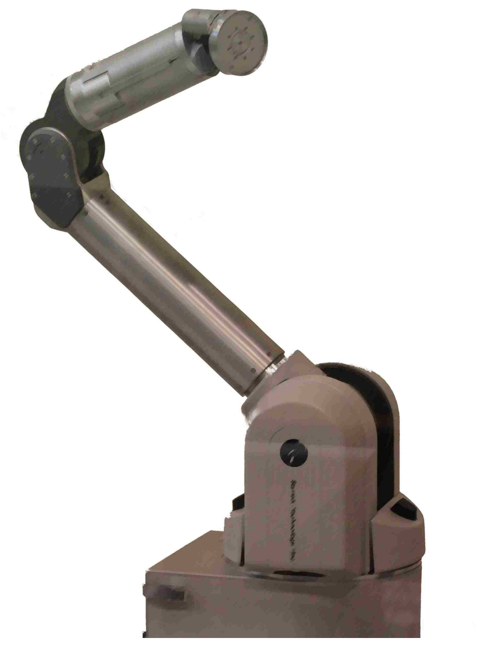


Figure 3.2: Whole Arm Manipulator by Barrett Technologies

The robots communicate with the target computers through a Controller Area Network (CAN) bus. Though this method of communication is well documented due to its extensive use in the automotive industry, the specific WAM communication protocols are proprietary. Data packets are sent over the bus with an 11 bit identifier to determine the message type, origin, and destination, and the data is packed with different bit patterns for each type of information: property, position, or torque.

Each robot has a Barrett three-fingered hand mounted for prehensile grasping. The hands are also differentially cable driven, and can be controlled by a teach pendant or directly by computer. Though the hands can be used to supply additional impedance properties, simple rigid grasps are assumed for the duration of this work.

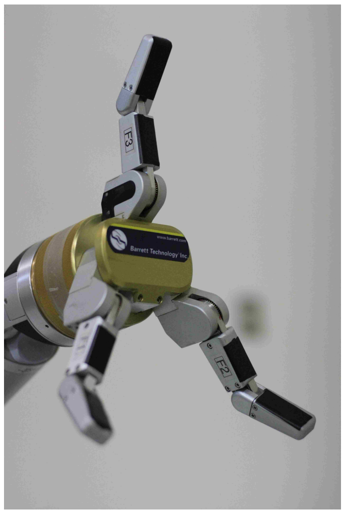


Figure 3.3: Barrett Hand

3.1.1 Kinematics

Although the dynamic parameters of the WAMs (Appendix C.1) come from CAD models rather than being physically measured through model identification, they proved capable of computing accurate gravitational torques and will continue to be used for all necessary modeling. To calculate the forward kinematics of the WAM,

i	a_i (m)	α_i (rad)	d_i (m)	θ_i (rad)
1	0	$-\frac{\pi}{2}$	0	θ_1
2	0	$\frac{\pi}{2}$	0	θ_2
3	0.045	$-\frac{\pi}{2}$	0.55	θ_3
4	-0.045	$\frac{\pi}{2}$	0	θ_4
5	0	$-\frac{\pi}{2}$	0.3	θ_5
6	0	$\frac{\pi}{2}$	0	θ_6
7	0	0	0.0609	θ_7
T	0	0	0.10	0

Table 3.1: Table of DH Link Parameters

the link parameters of Table 3.1, extracted from the frame assignments in Figure 3.4,

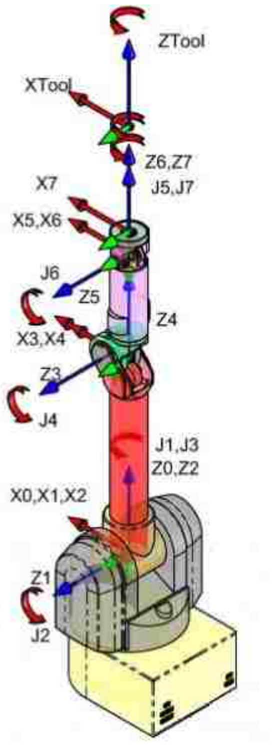


Figure 3.4: Frame Assignments for 7-dof WAM[1]

are used in equation (2.20) and combined to generate a transformation relating the base frame and tool frame,

$$ {}^{\mathbf{1}}_{\mathbf{T}}\mathbf{T} = {}^{\mathbf{1}}_{\mathbf{2}}\mathbf{T}{}^{\mathbf{2}}_{\mathbf{3}}\mathbf{T}\ldots{}^{\mathbf{7}}_{\mathbf{T}}\mathbf{T}. \tag{3.1} $$

3.1.2 Quantization

Quantization can hinder measurements of joint velocity. The motor positions are sensed by optical encoders, which inherently quantize. In calculating the Newton-Euler dynamics, it is necessary to compute the joint velocities of the WAM. However, these values can only be obtained using the quantized motor position values. Through discrete differentiation, the motor velocity values have a resolution of $\frac{\Delta\Theta_M}{\Delta t}$. Each encoder reads 4096 counts per revolution (0.00153 radians per count). Assuming a 500Hz sampling frequency, the velocity resolution becomes 0.767 radians per second. To avoid a noisy velocity signal, we implemented a discrete time Kalman filter to predict the position and velocity from the current and previous position measurements. The model we used was formulated in [20] and can be seen in Appendix B.2.

3.1.3 Force Measurement

Throughout this thesis, it is assumed that the system model is exact and forces generated by the manipulator's end-effector are equal to those commanded. However, this is never the case: nonlinearities in the motor, quantization of the signals, and unmodeled friction can cause errors. Therefore, we designed a preliminary experiment to compare the commanded and realized wrenches.

We ran four tests to compare force data from a wrist-mounted JR3 force/torque sensor and the WAM's own joint torque measurements. The first two runs were performed using a PD joint controller. The torques from the controller were saved and converted to forces using the pseudoinverse of the transpose Jacobian,

$$\mathbf{F} = (\mathbf{J}^T)^+\boldsymbol{\tau}. \tag{3.2}$$

The second two runs were made using a Cartesian space controller. In this case, forces were directly computed by the controller; no additional conversions were needed for comparison. Figure 3.5 shows data from a Cartesian control experiment, although no distinct differences could be found between the two methods. The WAM forces all contained biases in the force signals. However, once these were removed, the WAM and JR3 forces were quite similar - rarely differing by more than 5N. Signal noise only contributes to 0.5N of error in steady-state. Unmodeled friction is the likely cause of most under-actuation.

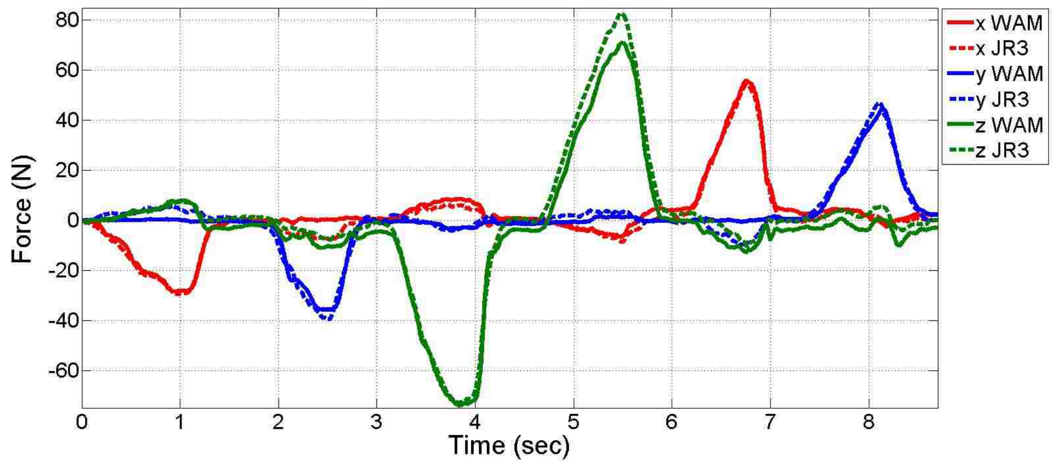


Figure 3.5: Comparison of WAM and JR3 Force Readings

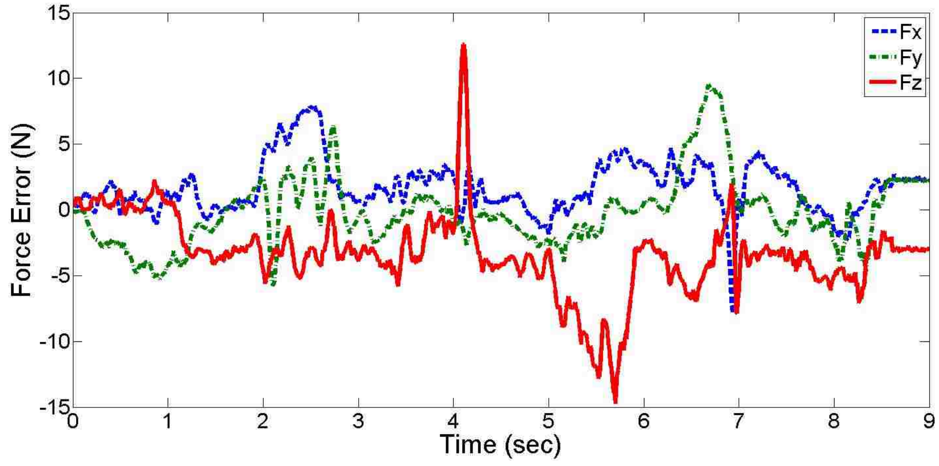


Figure 3.6: Force Reading Errors

Additional errors may be caused by error in the WAM's initial position. The WAM arms do not have a stored home position for the motors. Every time that the manipulator is reset, the encoder positions are also reset. Each joint has a physical marking to represent the zero position - this must be manually aligned; in many cases, this cannot be accurately achieved. With the Barrett Hands mounted to the wrist, the initial position changes. Minor errors in a single joint position may have substantial effects on the Cartesian errors. An error of 0.01 radians of a single joint at home position can result in up to 4mm of error in tool frame positioning. Although a manipulator will be oblivious its own error, cooperating robots may have an 8mm difference in the calculated object position; and, as the manipulator extends, these errors continue to increase. Assuming a mid-range stiffness gain of 1000N/m, this error could apply over 10N of force to a system that is perfectly tracking the trajectory. Throughout the experiments, we tried to consistently reset the manipulator to the same initial position. But initialization error always had some effect on the controller performance.

3.2 Robot Transport Unit

The RTU consists of a carriage, 4.3276 meters of usable low-friction track, and a 7 HP motor. The additional axis increases the manipulability of the workcell by both adding an extra degree of freedom to the system and extending the usable workspace of the manipulator. For simplicity, the station frame in all of our experiments lies directly between the two RTU home positions and at the same height as the manipulator base frames. Each manipulator's base frame is set back 0.209 meters (70203 counts) from the front of the RTU. Therefore, the transformations of the left and right base frames relative to the station frame are

$$ {}^{\mathbf{S}}_{\mathbf{LB}}\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & d_L - 2.3728 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3} $$

$$ {}^{\mathbf{S}}_{\mathbf{RB}}\mathbf{T} = \begin{bmatrix} -1 & 0 & 0 & 2.3728 - d_R \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.4} $$

d_L and d_R are the left and right RTU positions relative to each home position, respectively.

The carriage motion is controlled by a Galil 1816 DSP-based motion control board. This controller has the ability to drive the system in several different modes, all based on a high stiffness PID control loop around motor position. For our purposes, Position Tracking control mode is used. This mode accelerates at a constant rate to attain a maximum velocity while tracking continuously updated position setpoints. With high velocity and acceleration parameters (100,000 counts/sec and 256,000 counts/sec^2, respectively), we are able to accurately position the RTU.

The RTU and WAM are fundamentally different devices. The RTU is a rigid, accurate, position-based device, while the WAM is a compliant, torque-based device. Though the control algorithm computes forces and torques necessary for the system to correctly manipulate the object, the RTU forces must be converted into positions to be utilized. Therefore, we specify an apparent stiffness for the RTU and compute the position displacement necessary to achieve it,

$$ x = x_{des} - \frac{F_{RTU}}{K_{RTU}}. \tag{3.5} $$

3.3 Computers

The Host PC is a standard Dell Precision T3400 with a 2.83GHz, Intel Core2 Quad processor and 4GB of RAM. However, the Target PCs are custom built with a 2.53GHz, Intel Core2 Duo processor and 2GB of RAM and include PCI boards for the various communications (Softing CAN bus, Galil Motion Controller, GE Fanuc Reflective Memory, Intel Ethernet Chip Set). Each PCI board, except for the Galil, was chosen because of the support MATLAB provided. Although the communication between targets was available through Ethernet and CANbus networks, the desire for real time performance necessitated increased speed. For this reason, we chose to implement a reflective memory ring. This ring synchronizes multiple memory boards, through fiber optic cables, with nanosecond update rates.

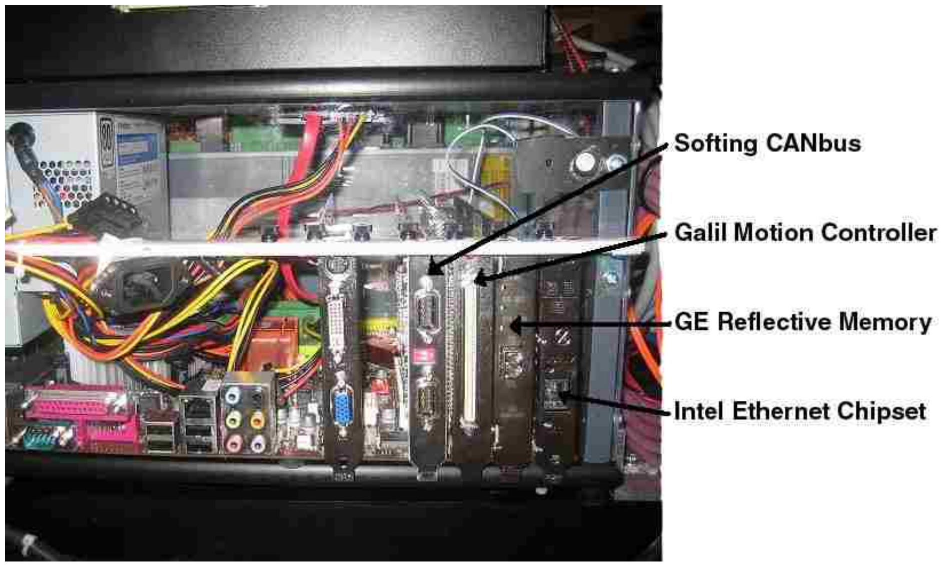


Figure 3.7: Target PC Hardware

3.4 MATLAB/Simulink/RTW/xPC

MathWorks software applications were used to design controllers, compile code and control the entire system in real time. These tools are widely used in industry and academia for control and simulation. The process begins in Simulink, a block-diagram-based environment running on the Host PC. Here, mathematical equations representing hardware models and controllers are encoded in blocks and arranged in a desired system formation. This approach to dynamic system modeling is an intuitive method that closely resembles the techniques taught in basic controls courses. The modularity of block diagrams encourages collaboration between researchers and accelerates the design process. Similar to a coded function, a given subsystem block can be stored in a library, where it can be quickly accessed for use in any future diagram. Once a system model is generated, Real Time Workshop (RTW) is used to build a program that runs in real time. This is a two step process: first the Simulink model generates corresponding C code, then the code is compiled into an executable module. For our applications, the executable module runs on a real time kernel, xPC Target, on the Target PC (one for each WAM) and communicates directly with the hardware components.

In the following chapter, computer simulations and hardware experiments are performed. We are able to construct controllers, simulate and actuate hardware, and record and analyze data all with the same multifaceted MathWorks software.

Chapter 4

Application

To apply the control law from Chapter 2 using the hardware and software described in Chapter 3, the following simulations and experiments were designed. The intent of these tests was to demonstrate the compliance and versatility of the proposed control system and compare it to a similar time-invariant impedance controller. We begin by simulating the control algorithm with simplified computer models. Then we designed two tasks for the dual-arm workcell. The first requires insertion of a commonly-held pipe into a corner whose position and orientation are uncertain; the object needs compliance to securely align with the recess. The second task requires transporting an object between two waypoints while encountering an obstacle; without damaging the object or obstacle, the controller must adjust for the unexpected forces and continue to track its trajectory. During the motions, the interaction forces were limited by the variable impedance to ensure safe handling while still completing the desired insertion/transport task.

4.1 Simulation

We first performed a series of computer simulations. In Simulink, two and three dimensional simplifications of the cooperative impedance controller were assembled without an RTU model. Free-space trajectories were followed using constant impedance parameters to ensure the control system was properly created, the manipulators maintained stability, and the computation time was sufficiently low.

4.1.1 Two Dimensional Control

A two dimensional simulation was performed in the MATLAB Simulink environment first. This undertaking used two, planar, revolute-revolute manipulators to control an imaginary object 0.2 meters in length. For ease, the object pose was calculated from the two end-effector positions,

$$\mathbf{X} = \begin{bmatrix} \frac{x_R+x_L}{2} \\ \frac{y_R+y_L}{2} \\ \mathrm{atan2}(y_R - y_L, x_R - x_L) \end{bmatrix}. \tag{4.1}$$

The robots began 1.0 meter apart at the base and had links 0.5 meters in length. The vectors, $\mathbf{p_i}$, for computing the grasp matrix were [-.1 0] and [.1 0] for the left and right manipulators respectively.

Because of the object's limited reachable workspace, we specified the manipulators to impart only a force on the object, no moment. The grasp matrix then reduces to (4.2); the torques in F_{cmd} are omitted.

$$\mathbf{W} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ -p_{1,y} & p_{1,x} & -p_{2,y} & p_{2,x} \end{bmatrix}. \tag{4.2}$$

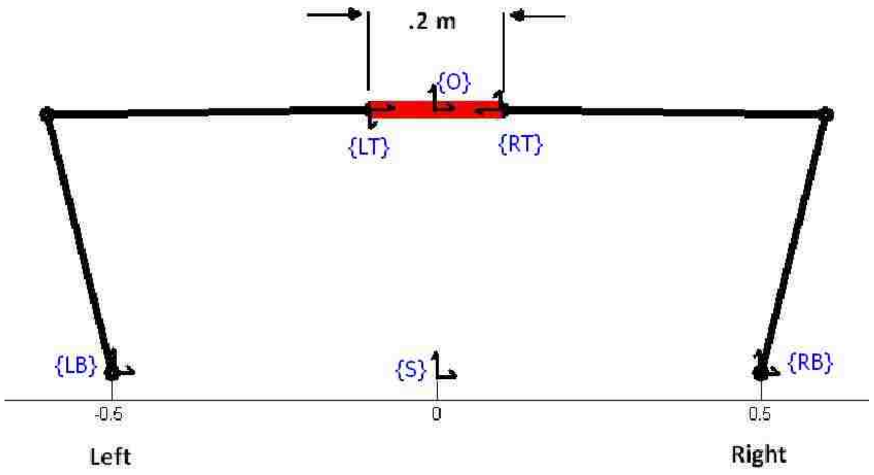


Figure 4.1: Schematic of 2-D Workcell

We assumed a rigid object, implementing a stiff virtual spring between the two manipulator tool frames to maintain a constant separation distance and relative orientation. The virtual spring force can also be used to measure the internal forces on the object. Other object properties such as mass, inertia, and shape were omitted.

During the command force computation, (2.15), we selected the desired inertia matrix to equal the actual and omitted the object dynamics. Therefore, the control law reduced to

$$\mathbf{F_{cmd}} = -\mathbf{W}^{+}(\mathbf{K_v}\dot{\mathbf{E}} + \mathbf{K_p}\mathbf{E}). \tag{4.3}$$

Though elementary, it is a useful introductory model of dual arm manipulation.

In tracking the simple position trajectory of Figure 4.2 with a high constant stiffness (2 N/mm), the tracking error remains less than 2.0 millimeters. As seen in Figure 4.4 below, instability can arise in certain orientations due to singularities in the Euler angles at $\frac{\pi}{2}$. Later experiments use the angular error equations presented in Section 2.3.2 to avoid this occurrence.

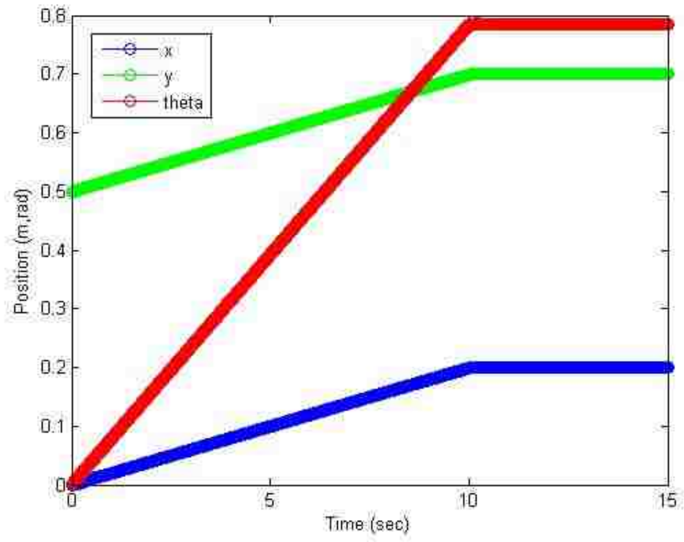


Figure 4.2: 2-D Simulation Trajectory

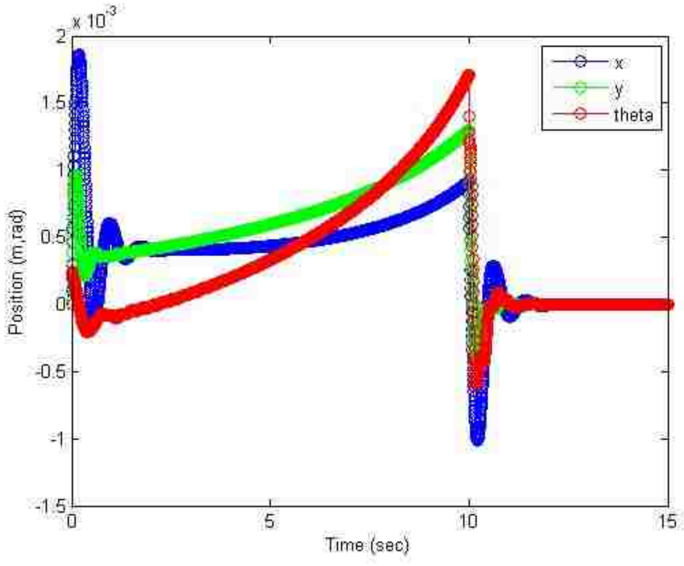


Figure 4.3: 2-D Simulation Tracking Error

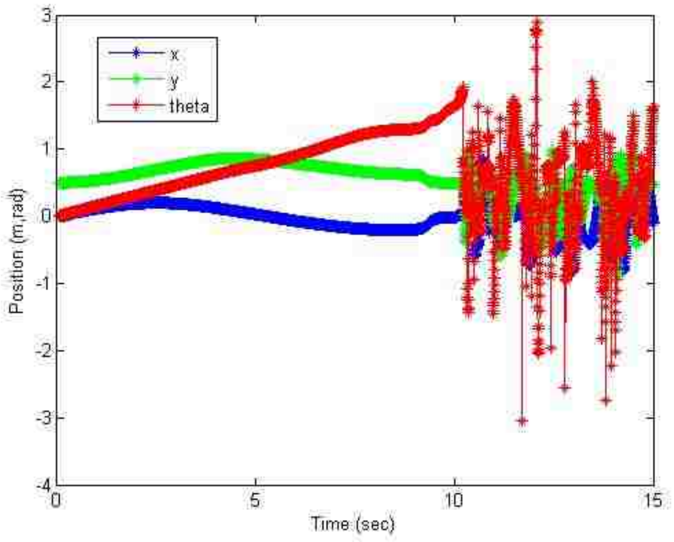


Figure 4.4: Object Position with Instability

4.1.2 Three Dimensional Control

In three dimensional controller simulations, the control equations more closely followed those presented in Chapter 2. This simulation modeled torque driven, 7-DOF manipulators representing frictionless WAMs. Because the control law and simulation used the same manipulator model, the two arm system could be completely controlled. The object used in this simulation has a 1.0 meter length, again without inertia properties, and the base frames of each manipulator were 0.5 meters from the station frame.

Unlike the two dimensional representation, where the kinematics can be determined through simple geometry, the three dimensional setup must use transform-based kinematic chains to determine the current object position. However, this method computes the object position using only one of the two available chains - either ${}^{\mathbf{S}}_{\mathbf{O}}\mathbf{T} = {}^{\mathbf{S}}_{\mathbf{LB}}\mathbf{T}\,{}^{\mathbf{LB}}_{\mathbf{LT}}\mathbf{T}\,{}^{\mathbf{LT}}_{\mathbf{O}}\mathbf{T}$ or ${}^{\mathbf{S}}_{\mathbf{O}}\mathbf{T} = {}^{\mathbf{S}}_{\mathbf{RB}}\mathbf{T}\,{}^{\mathbf{RB}}_{\mathbf{RT}}\mathbf{T}\,{}^{\mathbf{RT}}_{\mathbf{O}}\mathbf{T}$. The object's actual position and position error are dependent on only one manipulator. In simulations, this causes a feedforward effect if no direct interactions between the two robots exist. To enforce the rigid object criterion and eliminate the feedforward effect, we again employ a virtual object stiffness to apply a corrective force between the two grasp positions. Since we know the desired pose of the tool frames relative to the object, and we can compute the actual pose of the object, we can easily compute the relative error between the feedforward manipulator and its desired position. In realizing this model on actual robots, the object would transfer forces from one hand to the other and no force feedback would be necessary.

The trajectory for the three dimensional simulation proceeds from the initial position to final position by altering five of the six Cartesian parameters. The constant desired stiffness for this run was 1 N/mm for translation and 5 N-m/rad for orientation. The damping for translation was 30 N-s/m. Addition of the orientation damping caused the computation time of the simulations to increase exponentially and often lead to instability in the system. Therefore it was set to zero. Although the differences between the desired position in Figure 4.5 and the actual position in Figure 4.6 may appear significant, it must be noted that an impedance controller, like the one here, is less focused on positional accuracy and more on interaction forces. The oscillating error values may be due to several factors, including omission of orientation damping and poor manipulability (discussed in Section 4.2). When the experiment is extended to contact an environment, more analysis will be performed on the origins of controller error.

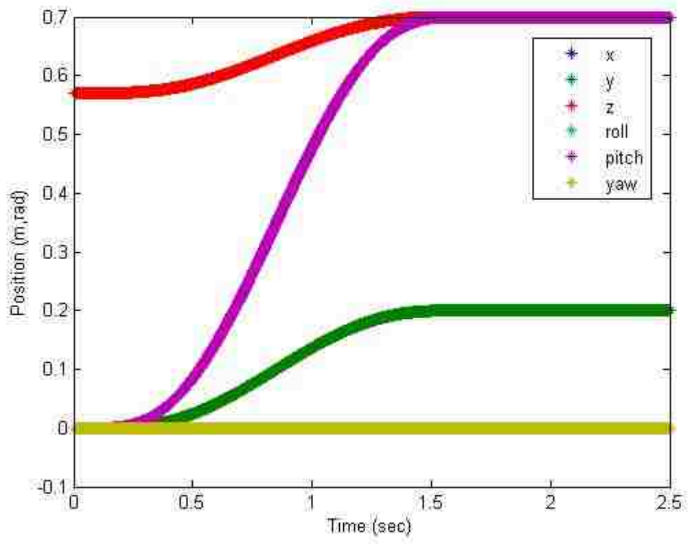


Figure 4.5: 3-D Simulation Trajectory

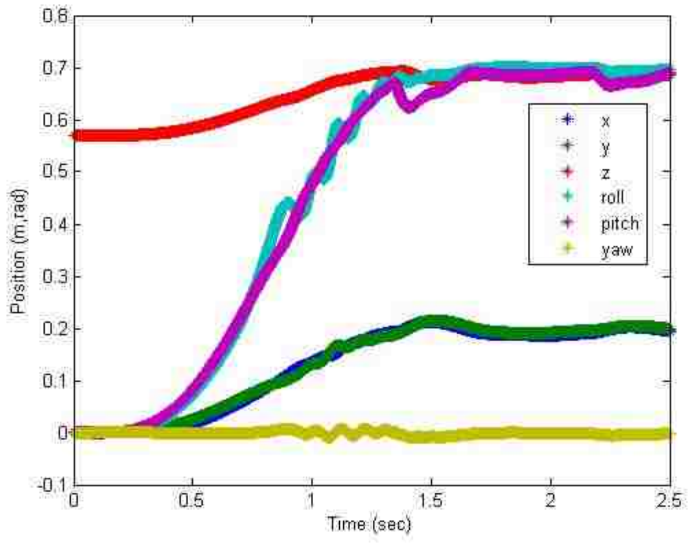


Figure 4.6: 3-D Simulation, Actual Path

4.2 Realization

4.2.1 Experiment 1

The first experiment performed on the actual hardware demonstrates the controller's compliant insertion capabilities. Manipulators attempt to insert an object into a corner while accounting for modeling errors both in the object and environment.

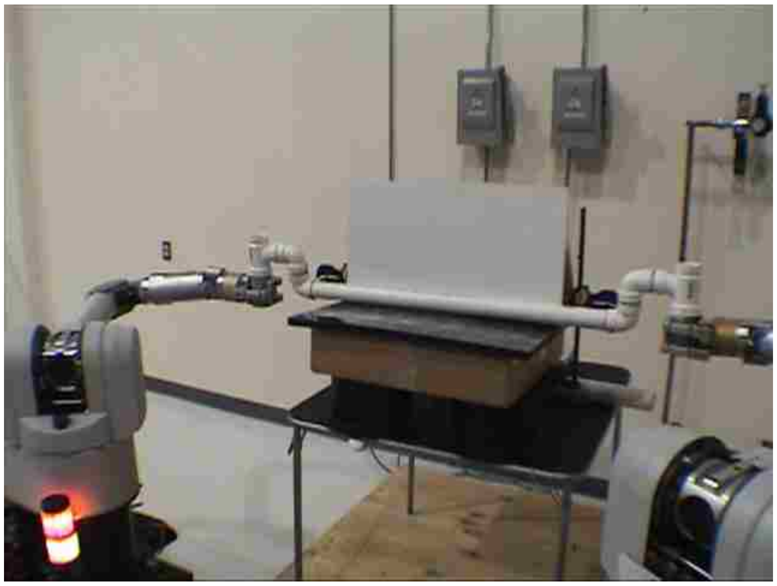


Figure 4.7: Completion of Insertion Task

Setup

The object in this experiment was formed from Schedule 40 PVC pipe. Several pieces of stock piping were combined to create a 1.78 kilogram object that lie directly between the two Barrett Hands. The object was symmetrically grasped to provide a large contact surface for the insertion without concern for manipulator interference. This simple system configuration closely resembles that of the computer simulations with the addition of the RTU and the object's inertia. The object frame and grasp locations can be seen in Figure 4.8.

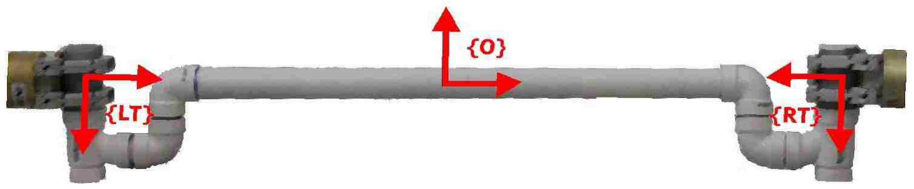


Figure 4.8: Experimental Object 1

The tool frames of the left and right manipulators are 0.605 meters on either side of the object frame. The transformation matrices relating the grasp frame to the object frame are

$$ {}^{\mathbf{LT}}_{\mathbf{O}}\mathbf{T} = \begin{bmatrix} 0 & 0 & -1 & -0.605 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.4} $$

$$ {}^{\mathbf{RT}}_{\mathbf{O}}\mathbf{T} = \begin{bmatrix} 0 & 0 & -1 & 0.605 \\ 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.5} $$

The inertia parameters of the object were estimated by assuming a single pipe with an additional point mass at both ends. The mass moments of inertia with respect to the y and z axes are the same because ${}^{O}x$ lies along the pipe's axis of symmetry. Thus, the inertia can be calculated from the equations,

$$ I_{xx} = \frac{1}{2}m(ID^2 + OD^2) \tag{4.6} $$

$$ I_{yy} = I_{zz} = \frac{1}{12}m_{ct}L_{ct}^2 + 2m_{pt}(\frac{L}{2})^2. \tag{4.7} $$

ID and OD are the inner and outer diameters of the pipe. The subscripts *ct* and *pt* correspond with the equivalent object's center pipe and point masses, respectively. L is the length of the object along the pipe's axis of symmetry. Table 4.1 lists the estimated object properties.

Parameter	**Value**
m	1.72 kg
I_{xx}	0.0013492 kg-m^2
I_{yy}	0.52308 kg-m^2
I_{zz}	0.52308 kg-m^2

Table 4.1: Estimated Object Parameters

The object motion for this experiment proceeded from the object grasp position of [0 0 0.6] to [0 0.8 0] using Cartesian straight line path and a rest-to-rest quintic velocity profile (Figure 4.9).

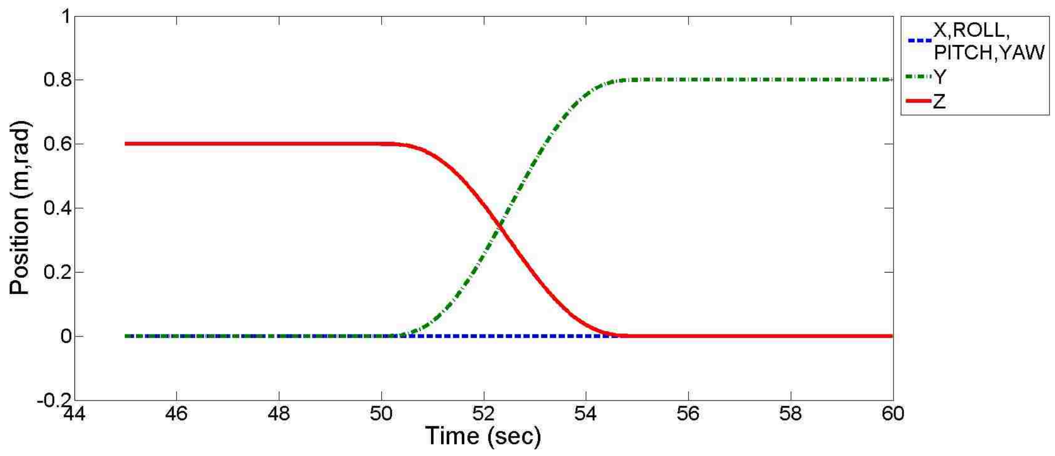


Figure 4.9: Object Trajectory for Experiment 1

The maximum force, $\mathbf{F_{max}}$, was set to [20 30 30 5 5 5], the rise constants, $\boldsymbol{\xi}$, were [0.02 0.03 0.03 0.25 0.25 0.25] and the damping coefficients, $\mathbf{K_v}$, were [10 30 30 0.3 0.3 0.3]. By specifying low stiffness forces, the object will be more compliant as it contacts the apparatus. These values result in the stiffness function seen in Figure 4.10.

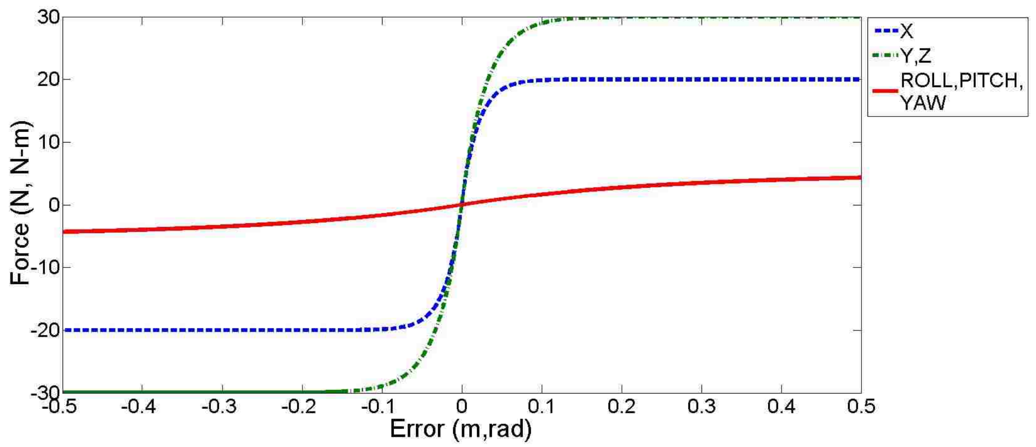


Figure 4.10: Force-Error Plot for Insertion

Results

To demonstrate the versatility of the cooperative control algorithm, it was implemented as a distributed system: each manipulator obtained its own estimate of the actual object position and second manipulator position. However, depending on the relative position and orientation of the tool frames post-grasp, large errors in the estimated actual and desired object positions can cause additional impedance forces to be applied that should not exist. To demonstrate this error and its effects on the assumed position trajectory, we plotted the position of the object as computed by each manipulator (Figure 4.11). This plot also takes into account the differences in initial WAM position noted in Section 3.1.3. It is easy to recognize the large and varying differences between each assumed position even though the actual position of the object lies approximately between the two. This means that larger forces are exerted on the object than necessary, however the majority reside in the system nullspace: internal object forces. To remain consistent throughout the remainder of

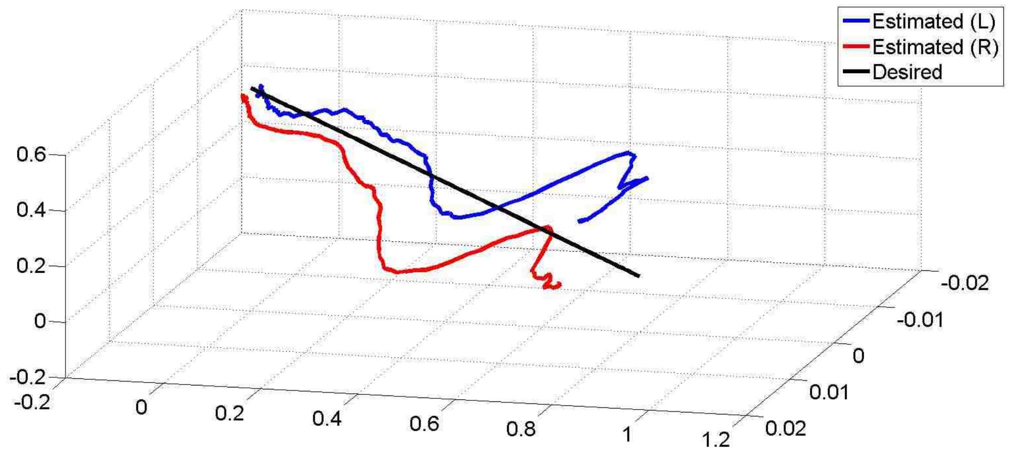


Figure 4.11: Experiment 1 Object Position

the analysis, only readings from the left robotic system will be used.

Despite the aforementioned errors, the object was successfully inserted into the corner without applying excessive forces to the environment. The total forces applied to the object due to the impedance controller can be seen plotted in Figure 4.12. The noise in the command forces are largely due to the remaining noise in the velocity signal (damping, centripetal, and Coriolis effects). Although the z forces on the object surpass the desired 30N limit, much of this surplus purely compensates for the object's mass (16.87N) and never becomes apparent to the environment.

Figure 4.13 depicts the object forces due to the time-varying stiffness. The variable gain in the proposed force-tracking impedance controller greatly limits the overall forces and clearly tracks the desired stiffness forces. Though only the stiffness parameter is adjusted in the current control scheme, any/all of the impedance gains can be controlled in the same manner. It was assumed that stiffness would be the largest contributor to the object forces. When comparing Figures 4.12 and 4.13, clearly this is a valid assumption.

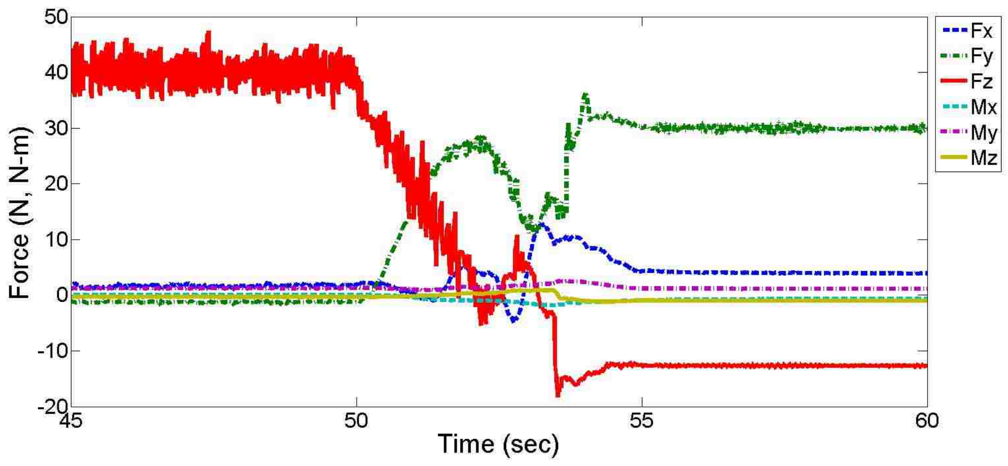


Figure 4.12: Total Impedance Force for Insertion

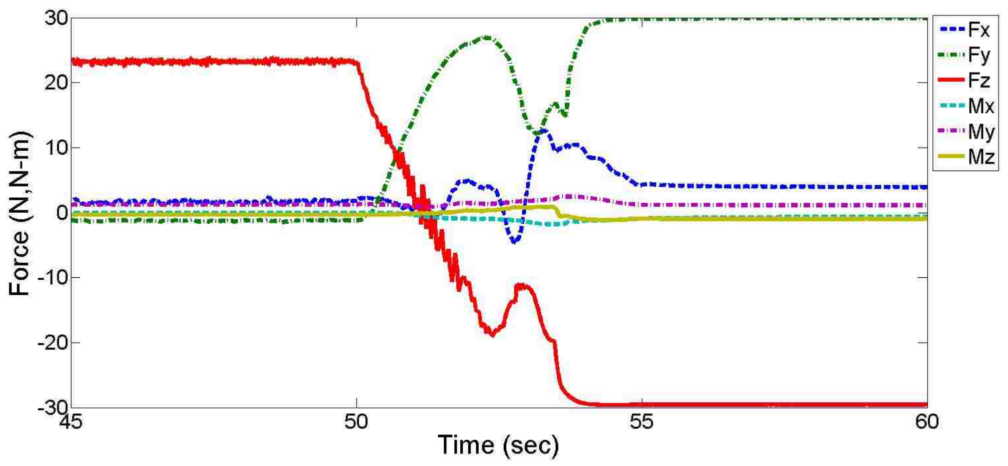


Figure 4.13: Experiment 1 Forces Due to Variable Stiffness

While testing a conventional time-invariant impedance controller, the stiffness gain remains constant - which creates large forces when all other experimental parameters remain the same. The stiffness parameter was set to the value of $\mathbf{K_p}$ at the origin, [1000 1000 1000 50 50 50]. Implementation of this controller for the same experiment with the same desired trajectory resulted in apparatus compliance rather than object compliance: the object sat snugly in the groove only after displacing the corner to a new location. Therefore, to directly compare the stiffness forces of a time-varying and a constant controller, we assume the apparatus can sustain high forces and calculate them from the errors of the compliant run. The corresponding stiffness forces are depicted in Figure 4.14.

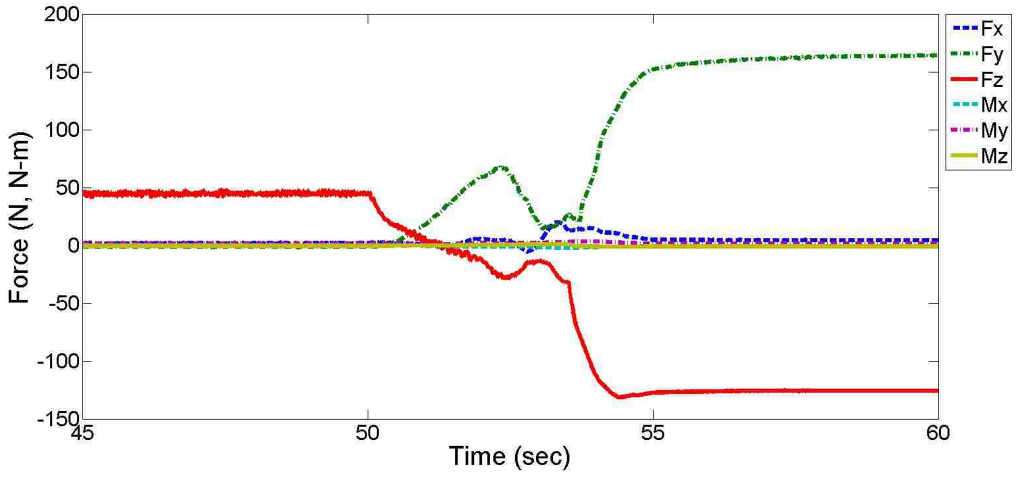


Figure 4.14: Experiment 1 Forces Due to Constant Stiffness

Finally, a comparison of the desired and realized RTU forces was performed (Figure 4.15). The realized forces were computed using (3.5) with a desired RTU stiffness of 1 N/mm.

Though the RTU tracks the overall desired forces throughout most of the experiment, its response is noticeably slower. The goal of this experiment, to limit the interaction forces while completing an uncertain insertion task, was successfully completed. Overall, the system performed well to realize desired forces. However, improvements could be made in the areas of system modeling and object position consensus.

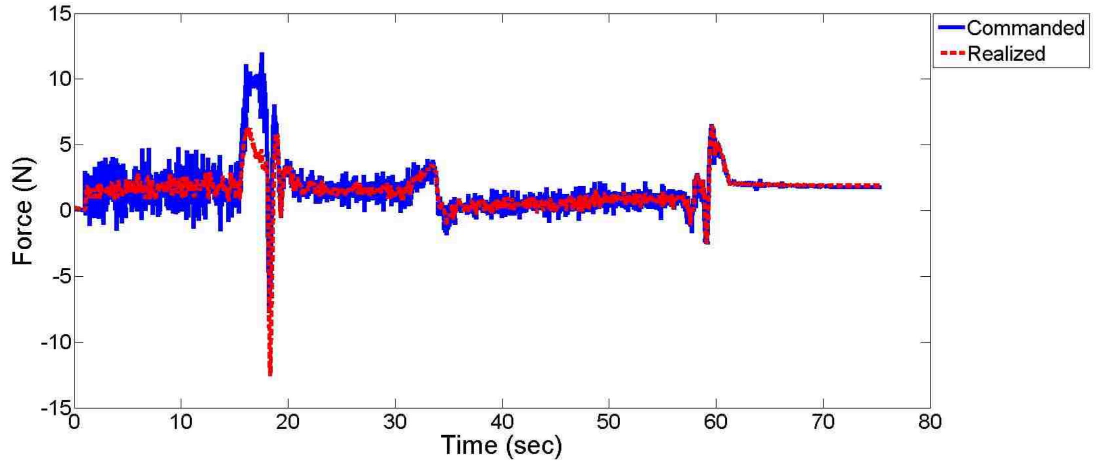


Figure 4.15: Commanded and Realized RTU Forces

4.2.2 Experiment 2

The second experiment was chosen to demonstrate the safe handling capabilities of the time-varying, cooperative impedance controller during transport tasks. This test manipulates an object along a desired free-space trajectory while encountering an obstacle. The manipulators must soften the impact with the impediment, maneuver around it, and continue along the path.

Setup

The common object in this experiment was a 1.24 meter PVC pipe with a mass of 1.28 kilograms and inner and outer diameters of 0.0520 meters and 0.0637 meters, respectively. The object was grasped, as shown in Figure 4.17, to facilitate a large object motion while keeping the arms moderately extended and protruding the pipe for contact.

The object frame and right manipulator grasp frame lie at the pipe center while

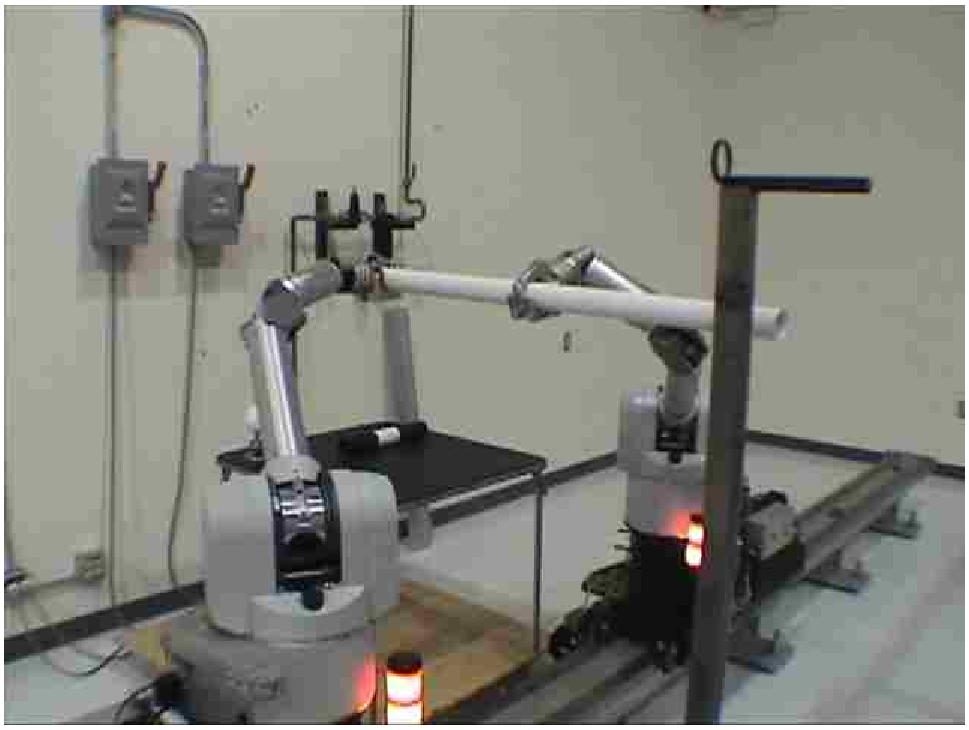


Figure 4.16: Completion of Transport Task

the left manipulator grasp frame is located 0.5 meters down the pipe in the positive ${}^{O}y$ direction. Therefore, the transformation matrices relating the object frame to the left and right grasp locations can be calculated as,

$$ {}^{\mathbf{LT}}_{\mathbf{O}}\mathbf{T} = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0.5 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.8} $$

$$ {}^{\mathbf{RT}}_{\mathbf{O}}\mathbf{T} = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.9} $$

However, this grasp configuration is not conducive to impedance control of the

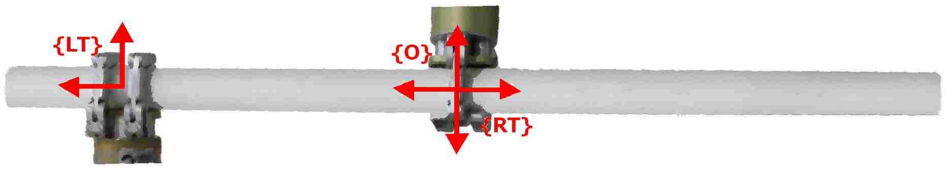


Figure 4.17: Experimental Object 2

WAM as formulated. The WAM's seventh joint is directly driven by the seventh motor, which greatly limits its ability to smoothly exert large torques. With a grasp location 0.5 meters from the object frame, small rotations of the left manipulator's seventh joint causes large motions in the object. To maintain a stiff positional impedance, this joint must employ high stiffness gains; but the the motor's limitations lead to instability of the controller under these conditions. Modifications to the grasp matrix were performed that cause the manipulator to relay zero moments to the object's x axis. Recalling the grasp matrix formulation in Chapter 2, the grasp moments relate to object moments via the 3x3 identity matrix in the lower right of each 6x6 partition. To limit the effects of the manipulator's seventh joint, we simply replace the "1" in the (1,1) entry of that identity matrix with "0". The resulting grasp matrix becomes,

$$\mathbf{W} = \begin{bmatrix} \mathbf{I_{3x3}} & \mathbf{0_{3x3}} & \mathbf{I_{3x3}} & \mathbf{0_{3x3}} & \cdots \\ \mathbf{P_1} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & \mathbf{P_2} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & \cdots \end{bmatrix}. \tag{4.10}$$

To compute **I** of object two, we again utilize the inertia equations for a pipe. Because the pipe's axis of symmetry lies along ${}^{O}y$, the mass moments of inertia for the x and z are the same,

$$\mathbf{I} = \begin{bmatrix} 0.16401 & 0 & 0 \\ 0 & 0.00433 & 0 \\ 0 & 0 & 0.16401 \end{bmatrix}. \tag{4.11}$$

For this experiment, the object has a rest-to-rest velocity profile while following a simple Cartesian line - extending 1.0 meter in the ${}^{S}x$ direction. All other Cartesian degrees of freedom are held constant. With the main task of tracking a trajectory, the translational stiffness and damping parameters for this task were set high,

$$\mathbf{F_{max}} = \begin{bmatrix} 50 & 50 & 50 & 30 & 20 & 5 \end{bmatrix} \tag{4.12}$$

$$\boldsymbol{\xi} = \begin{bmatrix} 0.05 & 0.05 & 0.05 & 1.0 & 0.667 & 0.167 \end{bmatrix} \tag{4.13}$$

$$\mathbf{K_v} = \begin{bmatrix} 10 & 30 & 30 & 0.3 & 0.3 & 0.3 \end{bmatrix}. \tag{4.14}$$

These values impose a translational and rotational stiffnesses of 1 N/mm and 30 N - m/rad when $\mathbf{E} << 1$.

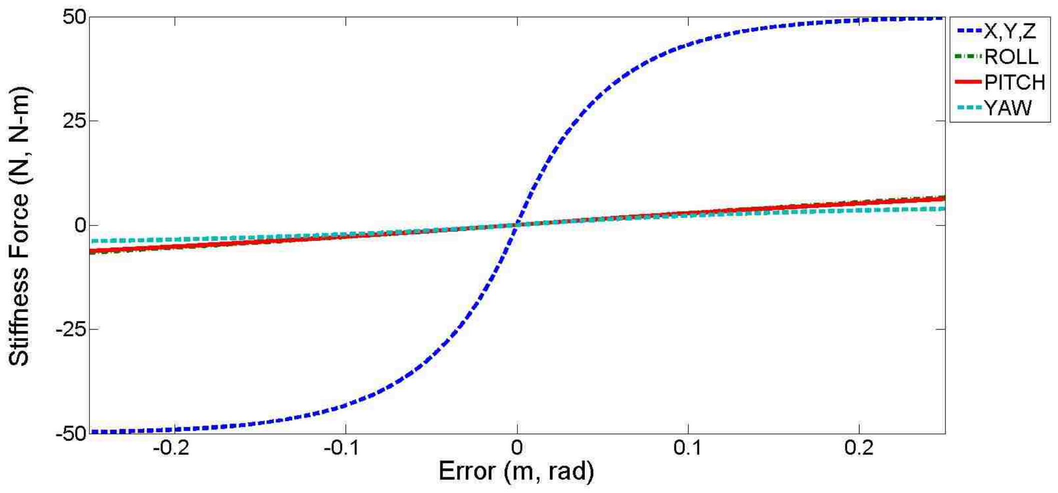


Figure 4.18: Force-Error Plot for Transport

Results

The motion trajectory began at approximately sixty-two and a half seconds. As the object traversed the desired trajectory, it encountered a stiff vertical bar that generated forces along the x axis and moments about the z axis of the station frame. At approximately the sixty-fifth second, disturbances in the x position and yaw rotation denote the impact.

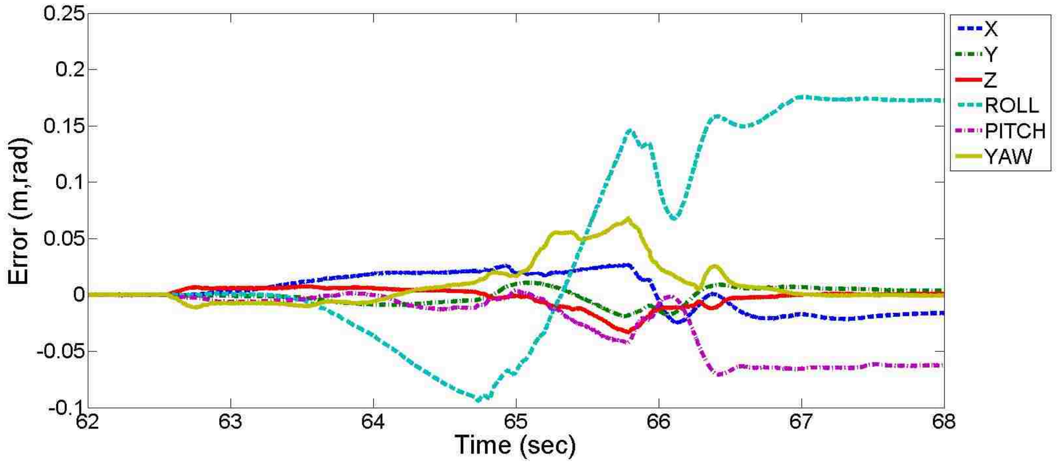


Figure 4.19: Trajectory Tracking Errors for Experiment 2

Interactions with the obstacle cause the object position errors to increase. These increased errors decrease the variable stiffness as the Force-Error relationship of Figure 4.18 is maintained. Figure 4.20 clearly shows the maximum stiffness value of 1 N/m for all three translation directions. This value can be calculated by (2.28) for when the position error is close to zero. At the instant the position errors are at a maximum, the stiffness is at a minimum and the impact forces are reduced. Once the obstacle has been overcome, the stiffness force recovers the position errors and the variable stiffness increases again.

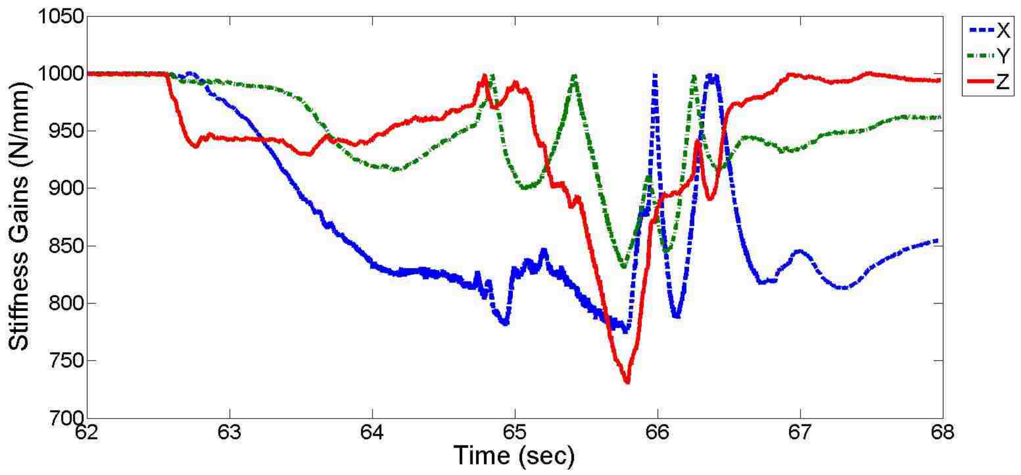


Figure 4.20: Translational Stiffness Gains During Experiment 2

Although an ideal experiment would demonstrate compliance only due to impact forces, reality shows that motions also appear in the other axes. These were not completely unforeseen; depending on the configuration and system modeling errors, the impact effects can vary.

The manipulability of a robotic arm is measured by the condition of its Jacobian and demonstrates the ability to realize tool motions. Found from the eigenvalues of the following matrix,

$$\mathbf{C_J} = \sqrt{\mathbf{J}\mathbf{J}^{\mathbf{T}}}, \tag{4.15}$$

the condition varies with the joint positions. If a robot's configuration leads to low manipulability in desired directions, trajectory tracking errors — including secondary motions — often appear.

In comparing the plots of error and manipulability, several correlations can be detected. As the manipulability of the x and z axes degrade, their errors begin to increase. Additionally, as the manipulability improves for y rotations, so too does the error. Although the yaw follows a similar trend, the collision effects provide a

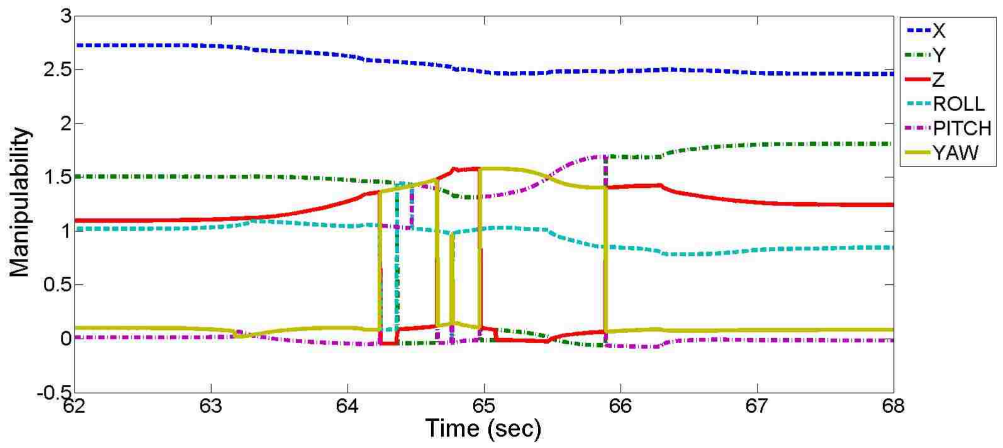


Figure 4.21: Condition of the Jacobian for Experiment 2

more significant contribution to this motion. Certainly, not all of the errors can be accounted for in this manner. Physical joint limits, for example, are not a factor in the manipulability measure. Fortunately, the manipulator's joints remained far from their extremes throughout all experimental runs.

Another cause for error in trajectory tracking is unmodeled dynamics. These propagate throughout the motion to form large residual errors that may not be recovered - even in steady state. After the desired trajectory comes to rest, friction, damping, and inertia forces resist motion and counteract the stiffness force. To demonstrate the effects of unmodeled dynamics, trials were performed with and without the feed-forward object model torques. The errors before and after each run are found in Table 4.2. Comparing the initial and final object errors, we can observe the effects of object modeling errors. Note that the change is significantly less in the roll and yaw rotations and z translation when the object's inertia is modeled. This is expected because the object model accounts for the gravity force (z axis) and Coriolis and centripetal effects- which have the greatest effect on the largest rotational motions (roll and yaw). Also, the large final error in the roll is likely due to torque restrictions in the modified grasp matrix (4.10).

WITHOUT MODELED OBJECT PARAMETERS						
	x	**y**	**z**	**roll**	**pitch**	**yaw**
Before	-0.02184	-0.007836	-0.02816	0.02772	-0.06643	-0.04758
After	-0.03767	-0.003674	-0.04586	0.15750	-0.12730	-0.07401
% Change	72.5	52.9	62.9	468.2	91.6	55.5
WITH MODELED OBJECT PARAMETERS						
	x	**y**	**z**	**roll**	**pitch**	**yaw**
Before	-0.02459	-0.008765	-0.05151	0.04569	-0.04194	-0.04635
After	-0.03922	-0.004538	-0.05065	0.21760	-0.10570	-0.04651
% Change	59.5	48.2	1.7	179.3	152.0	0.3

Table 4.2: Steady State Errors in Meters and Radians

The computer simulations and hardware experiments of this paper were designed with a specific intent: to demonstrate the versatile, compliant, and cooperative abilities of this controller. By using backdrivable torque-based manipulators with known dynamic models, we eliminated the need for force/torque sensors in our force control scheme. Clearly, the objectives were met; each cooperative task was completed successfully and the total interaction forces were bounded - even under large tracking errors.

Chapter 5

Conclusions

The goal of this research was to improve upon the available control methods and design a simple cooperative impedance controller that is versatile enough to perform multiple tasks without variation of the control structure and safe enough to interact with humans and fragile objects alike in unstructured environments. The fundamental aspect of an impedance controller is its ability to couple motion and force. Therefore, the impedance parameters are the simplest and most direct to adjust. Previous work on cooperative impedance control and force tracking impedance control dedicate hardware and software to enact forces with position-based manipulators that are inherently non-compliant. These schemes adjust the position trajectories and damping parameters to achieve a beneficial interaction; but they avoid using continuous, time-varying stiffnesses.

5.1 Contribution

This paper derives and implements a cooperative, compliant, force-tracking impedance controller on its ideal platform: a torque-based manipulator. The impedance parameters are varied with respect to the contact forces - which are obtained from position errors instead of explicit force/torque sensors. The experimental tasks performed demonstrate the ease of implementation, versatility, and compliant abilities of the controller. Even in the event of large position errors, the object's interaction forces are limited through the continuously-adjusted stiffness. Both the insertion task and transport task were successfully completed without damage to the workcell. The same was not true for the experiments run with constant stiffness controllers. Epitomizing the need for the time-varying controller, both experiments run with conventional controllers damaged equipment; the insertion task altered the corner location rather than complying with it and the transport task dislodged a manipulator cable and frayed a hand cable.

5.2 Future Research

Although the proposed controller completed the compliance tasks and demonstrated the advantages of time-varying impedance, improvements can still be made. Errors in the current control system arise from three main sources: (1) friction, (2) manipulability, and (3) consensus. Therefore, the major directions of research foreseen developing from this paper are model identification of the WAM and RTU, manipulability control, and the incorporation of computer vision.

The friction torques in each joint reduce the object's apparent forces. The desired interaction forces can not truly be realized because the controller lacks friction compensation (Figure 3.5). However, straightforward model identification can be performed to obtain a usable friction model in future WAM experiments. Because the joint torques are readily observable in the WAMs, an experiment could be devised that moves each joint through its range multiple times at different constant velocities. The simulated WAM model can then be subjected to the same experiment.

The difference between the actual and simulation torques during a constant velocity is the result of friction. By plotting these friction torques against the joint velocities, the coulomb and viscous coefficients become evident. A simple model relates friction forces to the joint velocities,

$$F_f = f_C \text{sign}(\dot{q}) + f_V \dot{q}, \tag{5.1}$$

where f_C and f_V are the Coulomb and viscous friction coefficients, respectively, and $\dot{q}$ is joint velocity. Concurrent improvements to the Kalman filter must also be performed to further reduce the velocity noise, as oscillatory Coulomb effects could be troublesome.

As seen in Section 4.2.2, poor system manipulability can also cause controller error, leading to undesirable secondary object motions. One possible approach to improve the robot's ability to realize motions is through the nullspace. Rather than controlling spacial parameters heuristically - that may or may not benefit the results, the nullspace could be used to maximize the manipulability of the object. Either through feedback control or path planning algorithms, a designated manipulability value could be tracked as an auxiliary robot task.

Finally, all manipulators must be in agreement as to the object's position to accurately control its impedance. Figure 4.11 best depicts the lack of consensus in the distributed control system. A straightforward method to unite the system is computer vision. Cameras in the workcell dedicated to determining the object's pose can relay data to a single control system. This unified system will then compute the torques for both manipulators. The Simulink controller construction in this paper facilitates sensor integration without major changes. Simply use the vision system, instead of transformation matrices, for the kinematic solution to greatly reduce the errors in object estimation and apply more accurate object forces.

A common goal in robotics research is to understand and apply methods for safe human-robot interactions. This aspect of automation is critical as more and more companies create robotic aides for use at home and work. The time-varying impedance controller presented here is one method that insures safe, compliant interactions and could be applied in such a dynamic environment.

Appendices

Appendix A

Implementation Notes

A.1 Target/Hardware Communications

The lowest level of implementation involves communication between the hardware and software. In the robot level diagram (Figure A.1), the *Target* block accepts the command torques and forces from the controller and returns the joint positions and velocities. Many subtasks are necessary to provide this simple representation. The joint torques/forces are first divided between the WAM and the RTU.

A.1.1 WAM - Softing CAN Bus

The manipulator requires the input torques to be zero while it is in *idle* mode. To ensure this, the *mode* block checks the state of the manipulator. If the arm is *idle*, *mode* sends “0” and multi-port switch sets the torques to zero; otherwise, the arm is *active*, *mode* sends “2”, and the controller’s torques are sent to the robot. Each motor is interfaced with a micro-controller known as a puck. These pucks retain all the necessary information about the motors including encoder position and motor

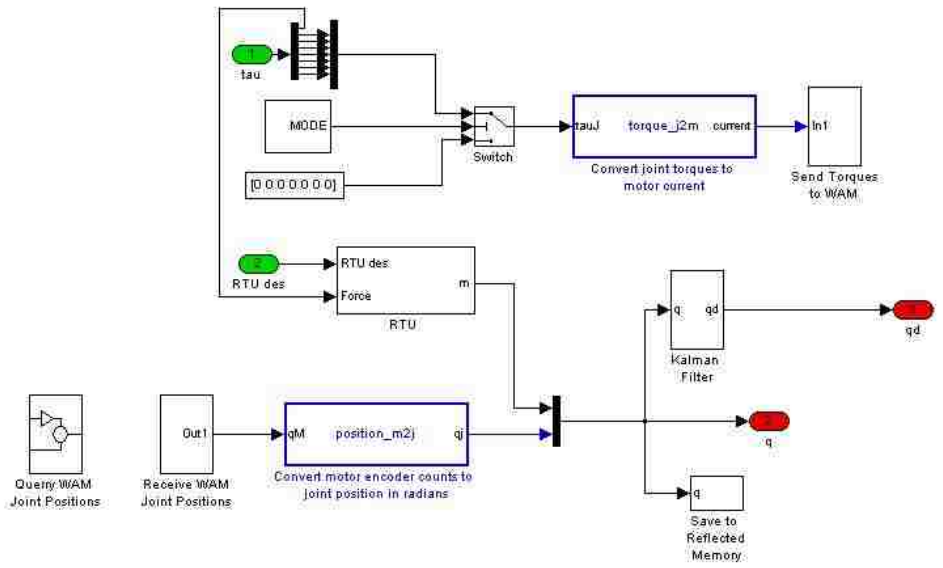

tau
MODE
[0 0 0 0 0 0 0]
Switch
tauJ
torque_j2m
current
Convert joint torques to motor current
In1
Send Torques to WAM
RTU des
Force
m
RTU
Kalman Filter
q
qd
Query WAM Joint Positions
Out1
Receive WAM Joint Positions
qM
position_m2j
qj
Convert motor encoder counts to joint position in radians
Save to Reflected Memory

Figure A.1: Block Diagram of Software/Hardware Interface

current. Because the motors drive the joints using a differential, the seven joint torques do not directly correspond to the seven motor currents. The joint torques are converted to motor torques using the drive ratios of Table A.1,

$$\boldsymbol{\tau}_M = \mathbf{D}^T \boldsymbol{\tau}_J \tag{A.1}$$

$$\mathbf{D} = \begin{bmatrix} -\frac{1}{N_1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2N_2} & -\frac{1}{2N_2} & 0 & 0 & 0 & 0 \\ 0 & -\frac{n_3}{2N_2} & -\frac{n_3}{2N_2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{N_4} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2N_5} & \frac{1}{2N_5} & 0 \\ 0 & 0 & 0 & 0 & -\frac{n_6}{2N_5} & \frac{n_6}{2N_5} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{N_7} \end{bmatrix}. \tag{A.2}$$

Then the ratios between motor torque and motor current (ν_i) and motor current and puck torque ($\tau_{p,i} = 1024$ is equivalent to 1.01A) are used to convert the i^{th} motor

Parameter	**Value**
N_1	42.0
N_2	28.25
N_3	28.25
n_3	1.68
N_4	-18.0
N_5	9.7
N_6	9.7
n_6	1.0
N_7	14.93

Table A.1: Drive Ratios Relating the WAM Motors and Joints

torques into puck torques,

$$\tau_{p,i} = \frac{1024}{\nu_i}\tau_{m,i}. \tag{A.3}$$

Finally, they are packed into data packets for the upper (pucks 5-7) and lower (pucks 1-4) arm. As was mentioned earlier, the WAM's CAN protocols are proprietary. CAN bus communications are supported by the MathWorks software and include initialization, bit (un)packing, send, and receive blocks ready for use with the Softing PCI board.

Pucks also sense and store the current encoder positions. These values are received over the CAN bus and unpacked. To make them usable in the control algorithm, the motor transition ratios are used for conversion to joint positions.

$$\boldsymbol{\theta}_J = \mathbf{D}\boldsymbol{\theta}_M \tag{A.4}$$

A.1.2 RTU - Galil 1816 PCI Motion Control Board

The RTU motor position is controlled by a Galil 1816 position control board. The communications with the Galil board are performed in using rudimentary, two letter commands. For interaction between the board and the Target PC, a driver was

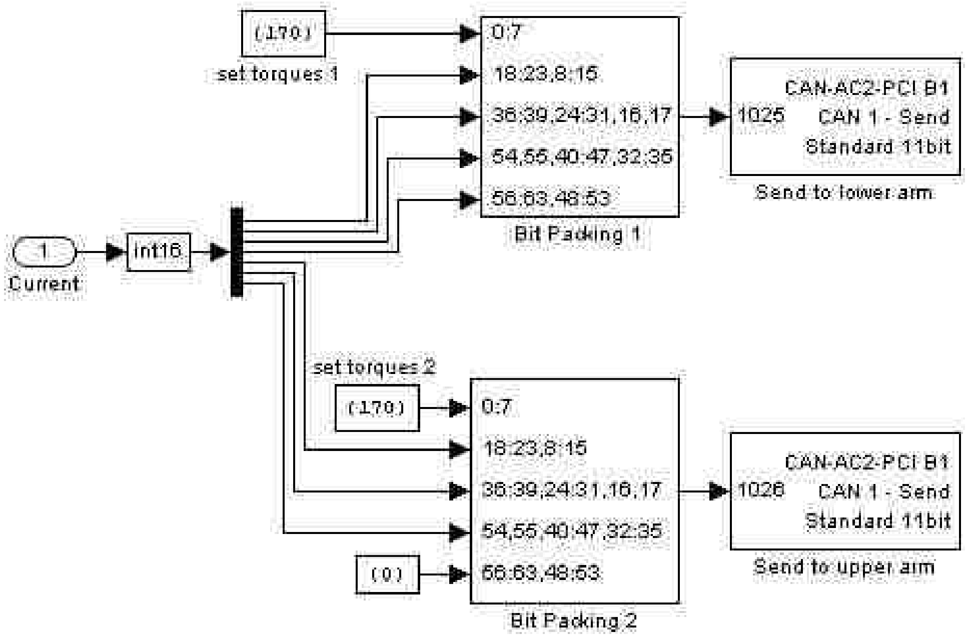


Figure A.2: Bit Packing Block

generated using a MATLAB S-function block. This block, when added to a system, initializes, sends commands to, and receives responses from the control board. The block has four inputs and four outputs. The first determines the block mode: initialize, send command, etc. The second input to the driver block accepts commands stored as int8 data types. The number of characters in the command, stored as an int32, is received by the third input port. And the final input determines the command index. Every time a new command is sent, the index number must be increased for the Galil driver to recognize it. Similarly the four output ports denote the block mode, response, response length, and response index. With this driver formulation, the block can be used in Simulink to execute commands either for a single instance or continuously.

During the controller implementation, the Galil commands are sent every time step. The following set of blocks receives the RTU position generated by the control algorithm and sends the corresponding commands the motion control board.

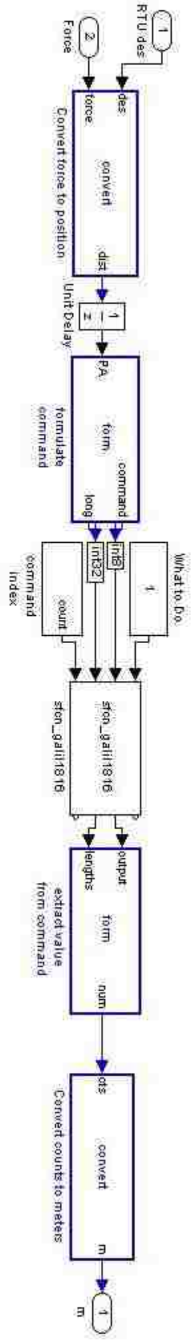
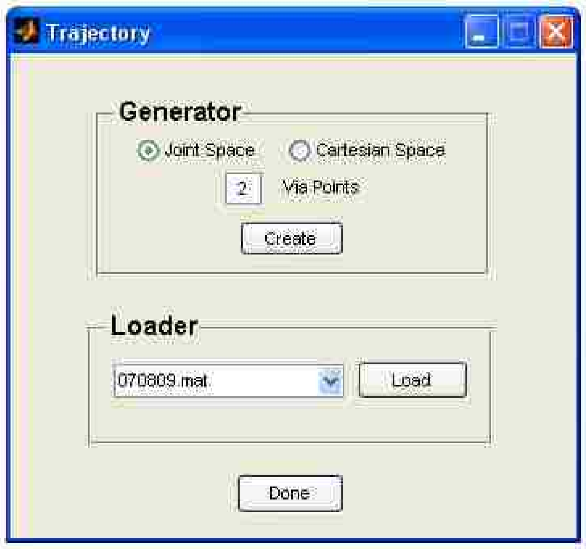


Figure A.3: RTU Interface Blocks

The calculations of (3.5) transform the controller force into a position for use with the motion control board. These are performed in the *convert force to position* block and sent to the *formulate command* block. This block determines the command input length; then the block generates two commands to be sent in a single instance: TPX to request the current encoder position and PA to set the current RTU position. The command and length are sent through the driver block (the other two inputs are optional). The driver's response is then processed in the *extract value from command* block. Here, the block determines if both commands were accepted (two semicolons in the response) and extracts the current position value. Finally, the encoder position is converted to meters using the empirical relationship: 1.00 meters = 167,954 counts.

A.2 Trajectory Generation

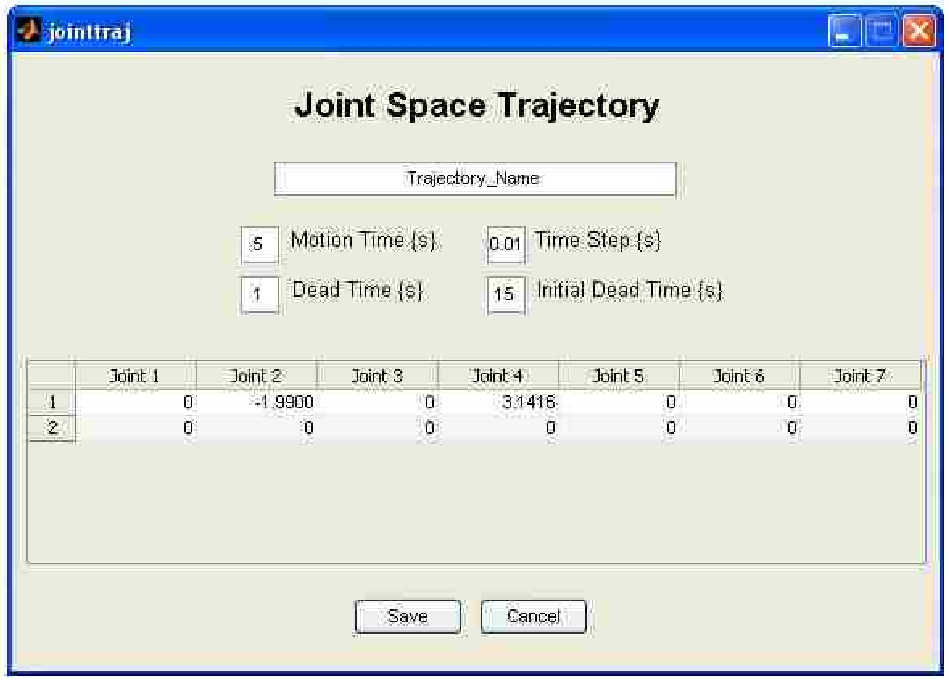
While compiling the experimental tasks for this thesis, many different trajectory files were generated and tested. To ease the trajectory formulation, we designed a Graphical User Interface (GUI) for creating joint-space and Cartesian-space paths with multiple waypoints. Upon starting the GUI, the first window lets the user choose whether to generate a trajectory or load a trajectory. If a new trajectory is to be created, the user specifies the type of trajectory and the number of waypoints. The next window, provides a table for data entry, including pauses in the trajectory, motion time, time step, and waypoints. These values are used to create a trajectory structure, *Rin* for Cartesian paths and *Pin* and *Vin* for joint paths, each with fields *time* and *signals*. The *signals* field has two subfields: *dimensions* and *values*. The dimensions vary by trajectory type: joint-space are nx7 and Cartesian space are 4x4xn. The structure is set up for easy incorporation in Simulink's *From Workspace* blocks.



[✓ Joint Space] Cartesian Space

Via Points: 2

070809.mat



Joint Space Trajectory

Trajectory_Name

Motion Time (s)	5
Time Step (s)	0.01
Dead Time (s)	1
Initial Dead Time (s)	15

	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6	Joint 7
1	0	-1.9900	0	3.1416	0	0	0
2	0	0	0	0	0	0	0

Figure A.4: Trajectory Generator GUI

The motion between each set of waypoints is calculated with a quintic rest-to-rest motion profile using the following equations:

$$P(t) = a_5t^5 + a_4t^4 + a_3t^3 + a_2t^2 + a_1t + a_0 \tag{A.5}$$

$$V(t) = 5a_4t^5 + 4a_4t^3 + 3a_3t^2 + 2a_2t + a_1 \tag{A.6}$$

$$a_0 = P_0 \tag{A.7}$$

$$a_1 = V_0 \tag{A.8}$$

$$a_2 = \frac{a_1}{2} \tag{A.9}$$

$$a_3 = \frac{20P_2 - 20P_1 - (8V_1 + 12V_0)t_f - (3a_1 - a_2)t_f^2}{2t_f^3} \tag{A.10}$$

$$a_4 = \frac{30P_0 - 30P_1 + (14V_1 + 16V_0)t_f + (3a_1 - 2a_2)t_f^2}{2t_f^4} \tag{A.11}$$

$$a_5 = \frac{12(P_1 - P_0) - 6(V_1 + V_0)t_f - (a_1 - a_2)t_f^2}{2t_f^5}. \tag{A.12}$$

For joint trajectories, the subtrajectories between each pair of waypoints are simply concatenated. However with Cartesian trajectories, the velocity trajectories (V(t)) are omitted and a transformation matrix is computed for each time step,

$$\mathbf{R}(\mathbf{t}) = \left[\begin{array}{cc} \cos(\alpha(t))\cos(\beta(t))) & \cos(\alpha(t))\sin(\beta(t))\sin(\gamma(t)) - \sin(\alpha(t))\cos(\gamma(t)) \\ \sin(\alpha(t))\cos(\beta(t)) & \sin(\alpha(t))\sin(\beta(t))\sin(\gamma(t)) + \cos(\alpha(t))\cos(\gamma(t)) \\ -\sin(\beta(t)) & \cos(\beta(t))\sin(\gamma(t)) \\ 0 & 0 \end{array}\right.$$

$$\left.\begin{array}{cc} \cos(\alpha(t))\sin(\beta(t))\cos(\gamma(t)) + \sin(\alpha(t))\sin(\gamma(t)) & x(t) \\ \sin(\alpha(t))\sin(\beta(t))\cos(\gamma(t)) - \cos(\alpha(t))\sin(\gamma(t)) & y(t) \\ \cos(\beta(t))\cos(\gamma(t)) & z(t) \\ 0 & 1 \end{array}\right], \tag{A.13}$$

where α, β, and γ are the desired yaw, pitch, and roll.

A.3 Initialization

Before the cooperative impedance controller can begin, the RTU must be homed and moved into its initial position (0.5 meters from {S}). To communicate with the control board without the full controller running, we created a teach pendant GUI that allows commands to be sent for a single instance. Every time the 'Set/Get' button is pressed, the command is sent and the response is retrieved and both are stored in the 'History' list.

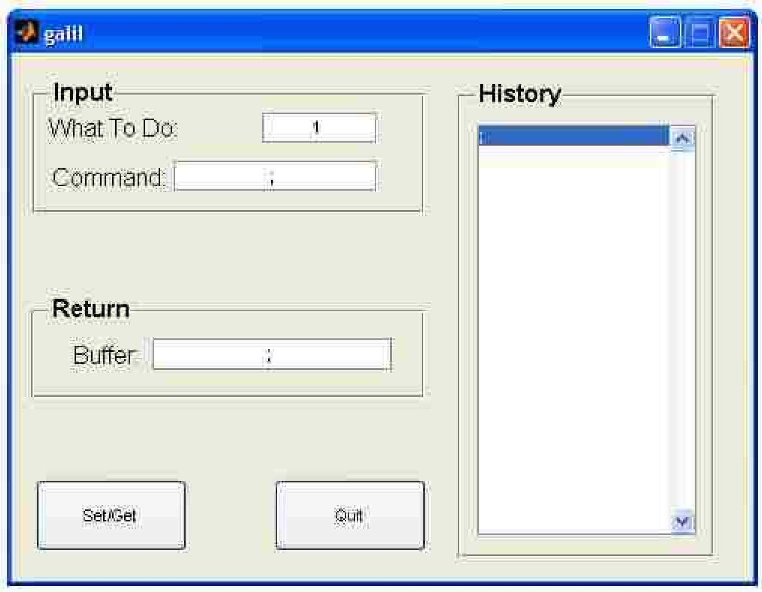


Figure A.5: GUI Teach Pendant for Galil Board

Each time the power is reset, the RTUs must be homed and set to their initial positions. To do this, the RTU speeds are set low (*HM 2000;*). Then the homing function is started (*HM;BG;*). Once the RTU has reset its encoder to the zero at its home position, the speed is returned to its desired value (*SP 100000;*), the position tracking mode is entered (*PT 1;*), and the RTU is moved to the initial position (*PA 314500;*).

Unlike the RTUs that remain still when uncontrolled, the manipulators fall. The manipulators cannot begin in a position grasping the object; there must first be a trajectory from the initial home position to the grasp position. Then a secondary trajectory, with the object, commands motion its final position. To transition between these two manipulation states, we utilize the reflective memory communications and the MATLAB Stateflow toolbox. The reflected memory allows a trigger signal to be sent to both WAM systems within the same integration time step. This signal triggers a specially designed *triggered ramp* block. When enabled, a Stateflow chart records the trigger time. By subtracting this from the current time and dividing by the update rate, an enabled ramp, with slope $\frac{1}{dt}$, is created. The initial ramp value is set to "1" by the bias block and only allowed to reach a maximum value equal to the number of steps in the secondary trajectory. This ramp signal is fed into a selector block that returns the indexed point of the secondary trajectory. Effectively, this method allows for the system to begin a secondary trajectory using a signal to trigger the motion to commence.

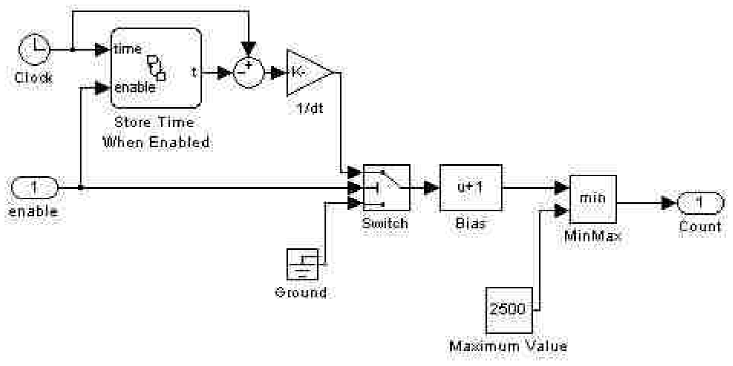


Figure A.6: Diagram of 'Triggered Ramp' Block

Each grasp-state trajectory begins and ends in the same place (for each manipulator). Even though the manipulators are acting independently during this motion, the trajectory refers to an object position to avoid changing the controller during this separate state. The grasp-state path is a Cartesian line between two points with a rest-to-rest velocity profile. These two object points correspond to the WAM starting in home position and ending ready to grasp an object at [0 0 0.6].

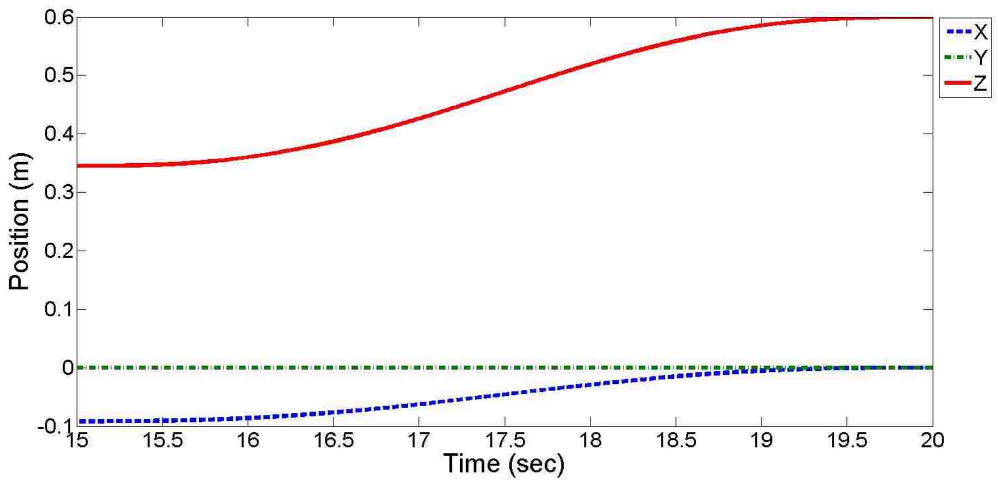


Figure A.7: Left WAM Position Trajectory During Grasp State

Appendix B

Derivations

B.1 Orientation Error

Using the notation from (2.21), we can evaluate (2.23) for $\mathbf{R_\Delta}$.

$$\mathbf{R_\Delta} = \begin{bmatrix} n_{xd} & o_{xd} & a_{xd} \\ n_{yd} & o_{yd} & a_{yd} \\ n_{zd} & o_{yd} & a_{yd} \end{bmatrix} \begin{bmatrix} n_{xa} & o_{xa} & a_{xa} \\ n_{ya} & o_{ya} & a_{ya} \\ n_{za} & o_{ya} & a_{ya} \end{bmatrix}^T \tag{B.1}$$

$$= \begin{bmatrix} n_{xd}n_{xa} + o_{xd}o_{xa} + a_{xd}a_{xa} & n_{xd}n_{ya} + o_{xd}o_{ya} + a_{xd}a_{ya} & n_{xd}n_{za} + o_{xd}o_{za} + a_{xd}a_{za} \\ n_{yd}n_{xa} + o_{yd}o_{xa} + a_{yd}a_{xa} & n_{yd}n_{ya} + o_{yd}o_{ya} + a_{yd}a_{ya} & n_{yd}n_{za} + o_{yd}o_{za} + a_{yd}a_{za} \\ n_{zd}n_{xa} + o_{zd}o_{xa} + a_{zd}a_{xa} & n_{zd}n_{ya} + o_{zd}o_{ya} + a_{zd}a_{ya} & n_{zd}n_{za} + o_{zd}o_{za} + a_{zd}a_{za} \end{bmatrix} \tag{B.2}$$

From [17], we know that the differential rotation matrix is represented by

$$\mathbf{R}_{\boldsymbol{\Delta}} = \begin{bmatrix} 0 & -\partial_z & \partial_y \\ \partial_z & 0 & -\partial_x \\ -\partial_y & \partial_x & 0 \end{bmatrix} \tag{B.3}$$

By equating the off diagonal terms of (B.2) and (B.3) we find two separate equations for the differential orientations ($\boldsymbol{\partial} = \begin{bmatrix} \partial_x \\ \partial_y \\ \partial_z \end{bmatrix}$).

$$\boldsymbol{\partial} = \begin{bmatrix} n_{zd}n_{ya} + o_{zd}o_{ya} + a_{zd}a_{ya} \\ n_{xd}n_{za} + o_{xd}o_{za} + a_{xd}a_{za} \\ n_{yd}n_{xa} + o_{yd}o_{xa} + a_{yd}a_{xa} \end{bmatrix} \tag{B.4}$$

$$\boldsymbol{\partial} = \begin{bmatrix} -n_{yd}n_{za} - o_{yd}o_{za} - a_{yd}a_{za} \\ -n_{zd}n_{xa} - o_{zd}o_{xa} - a_{zd}a_{xa} \\ -n_{xd}n_{ya} - o_{xd}o_{ya} - a_{xd}a_{ya} \end{bmatrix} \tag{B.5}$$

Adding (B.4) and (B.5) and grouping like terms, we find

$$2\boldsymbol{\partial} = \begin{bmatrix} n_{zd}n_{ya} - n_{yd}n_{za} + o_{zd}o_{ya} - o_{yd}o_{za} + a_{zd}a_{ya} - a_{yd}a_{za} \\ n_{xd}n_{za} - n_{zd}n_{xa} + o_{xd}o_{za} - o_{zd}o_{xa} + a_{xd}a_{za} - a_{zd}a_{xa} \\ n_{yd}n_{xa} - n_{xd}n_{ya} + o_{yd}o_{xa} - o_{xd}o_{ya} + a_{yd}a_{xa} - a_{xd}a_{ya} \end{bmatrix} \tag{B.6}$$

$$\begin{aligned} \boldsymbol{\partial} &= \frac{1}{2}\left[\begin{bmatrix} n_{zd}n_{ya} - n_{yd}n_{za} \\ n_{xd}n_{za} - n_{zd}n_{xa} \\ n_{yd}n_{xa} - n_{xd}n_{ya} \end{bmatrix} + \begin{bmatrix} o_{zd}o_{ya} - o_{yd}o_{za} \\ o_{xd}o_{za} - o_{zd}o_{xa} \\ o_{yd}o_{xa} - o_{xd}o_{ya} \end{bmatrix} \right. \\ &\quad \left. + \begin{bmatrix} a_{zd}a_{ya} - a_{yd}a_{za} \\ a_{xd}a_{za} - a_{zd}a_{xa} \\ a_{yd}a_{xa} - a_{xd}a_{ya} \end{bmatrix}\right] \qquad \text{(B.7)} \\ &= \frac{1}{2}(\mathbf{n_a} \times \mathbf{n_d} + \mathbf{o_a} \times \mathbf{o_d} + \mathbf{a_a} \times \mathbf{a_d}) \qquad \text{(B.8)} \end{aligned}$$

B.2 Discrete Kalman Filter

In this implementation of a Kalman filter, the encoder quantization with covariance $\frac{1}{3}\Delta\theta_m$ is considered the measurement noise (ν) and the motor acceleration is driven by white noise (w) with empirically chosen covariance (Q).

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w \tag{B.9}$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \nu \tag{B.10}$$

The discrete Kalman iteration equation is as follows.

$$P_{k+1|k} = A_d P_{k|k} A_d^T + B_d \tag{B.11}$$

$$k_d = P_{k|k-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix} (\begin{bmatrix} 1 & 0 \end{bmatrix} P_{k|k-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \frac{1}{3}\Delta\theta)^{-1} \tag{B.12}$$

$$P_{k|k} = (I_{2x2} - k_d \begin{bmatrix} 1 & 0 \end{bmatrix}) P_{k|k-1} (I_{2x2} - k_d \begin{bmatrix} 1 & 0 \end{bmatrix})^T + k_d(\frac{1}{3}\Delta\theta) k_d^T \tag{B.13}$$

where,

$$A_d = I_{2x2} + \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \Delta t \tag{B.14}$$

$$B_d = \Delta t Q \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix}. \tag{B.15}$$

B.3 Nullspace Projection

To ensure that the redundancy control torques reside in the nullspace of the Jacobian's inverse transpose, we use a mapping function. Note that $\mathbf{J^{-T}}$ is actually the pseudoinverse of the Jacobian transpose because the matrix is non-square. We want a projection operator, $\mathbf{P}$, such that $\mathbf{P}\boldsymbol{\tau}_\phi \in \mathbf{N}(\mathbf{J^{-T}})$ i.e.

$$\mathbf{J^{-T}P}\boldsymbol{\tau}_\phi = \mathbf{0}. \tag{B.16}$$

We can show that the projection matrix,

$$\mathbf{P} = (\mathbf{I} - \mathbf{J^T}(\mathbf{JJ^T})^{-1}\mathbf{J}), \tag{B.17}$$

guarantees this. Substituting () into () we find

$$\begin{aligned}
\mathbf{J^{-T}}(\mathbf{I} - \mathbf{J^T}(\mathbf{JJ^T})^{-1}\mathbf{J})\boldsymbol{\tau}_\phi &= (\mathbf{JJ^T})^{-1}\mathbf{J}(\mathbf{I} - \mathbf{J^T}(\mathbf{JJ^T})^{-1}\mathbf{J})\boldsymbol{\tau}_\phi && \text{(B.18)}\\
&= [(\mathbf{JJ^T})^{-1}\mathbf{J} - (\mathbf{JJ^T})^{-1}\mathbf{JJ^T}(\mathbf{JJ^T})^{-1}\mathbf{J}]\boldsymbol{\tau}_\phi && \text{(B.19)}\\
&= [(\mathbf{JJ^T})^{-1}\mathbf{J} - (\mathbf{JJ^T})^{-1}\mathbf{J}]\boldsymbol{\tau}_\phi && \text{(B.20)}\\
&= \mathbf{0}\boldsymbol{\tau}_\phi && \text{(B.21)}\\
&= \mathbf{0}. && \text{(B.22)}
\end{aligned}$$

Therefore, P projects the redundant control torques into the nullspace.

B.4 Joint Accelerations

To fully model the manipulator dynamics, the joint accelerations must be computed. Rather than calculating the joint accelerations through direct differentiation and amplifying the noise from the position signal, it is possible to use the following equation,

$$\ddot{\mathbf{q}} = \mathbf{J^{-1}}\dot{\mathbf{J}}\dot{\mathbf{q}}. \tag{B.23}$$

To compute the Jacobian derivative, we use the equation

$$\dot{\mathbf{J}} = \dot{\mathbf{q}}^{\mathbf{T}}\mathbf{H}. \tag{B.24}$$

where $\mathbf{H}$ is the Hessian matrix. We modified the method proposed in [21] for revolute joint variables to compute the numerical Hessian,

$$\mathbf{H} = \frac{\partial^2 \mathbf{X}}{\partial \boldsymbol{\theta}_\mathbf{i} \partial \boldsymbol{\theta}_\mathbf{j}} = \begin{bmatrix} \mathbf{a_i} \times (\mathbf{a_j} \times (\mathbf{p_e} - \mathbf{p_j})) \\ \mathbf{a_i} \times \mathbf{a_j} \end{bmatrix}. \tag{B.25}$$

If we also note that the Jacobian for revolute joint variables can be written as

$$\mathbf{J} = \frac{\partial \mathbf{X}}{\partial \boldsymbol{\theta}} = \begin{bmatrix} \mathbf{a_i} \times (\mathbf{p_e} - \mathbf{p_i}) \\ \mathbf{a_i} \end{bmatrix}, \tag{B.26}$$

we can then write the Hessian as a function of a previously computed Jacobian,

$$\mathbf{H} = \begin{bmatrix} \mathbf{J}(i, 4:6) \times \mathbf{J}(j, 1:3) \\ \mathbf{J}(i, 4:6) \times \mathbf{J}(j, 4:6) \end{bmatrix}. \tag{B.27}$$

It is then straightforward to compute the Jacobian derivative $(\dot{\mathbf{J}}(\mathbf{J}, \dot{\mathbf{q}}))$.

Appendix C

Manipulator Specifications

C.1 Inertial Data

Barrett Technologies supplied the inertial data of the WAM. The information is based on computer modeling of the components and includes link mass (m_i), center of mass location with respect to the link frame ($\mathbf{p_i}$), and the inertia tensor at the center of mass aligned with the link frame ($\mathbf{L_i}$).

Frame 1

$$
\begin{aligned}
m_1 &= 8.3936\text{kg} \\
\mathbf{p_{m,1}} &= \begin{bmatrix} 0.3506 \\ 132.6795 \\ 0.6286 \end{bmatrix} \text{mm} \\
\mathbf{L_1} &= \begin{bmatrix} 95157.4294 & 246.1404 & -95.0183 \\ 246.1404 & 92032.3524 & -962.6725 \\ -95.0183 & -962.6725 & 59290.5997 \end{bmatrix} \text{kg} - \text{mm}^2
\end{aligned}
$$

Frame 2

$$
\begin{aligned}
m_2 &= 4.8487\text{kg} \\
\mathbf{p_{m,2}} &= \begin{bmatrix} -0.2230 \\ -21.3924 \\ 13.3754 \end{bmatrix} \text{mm} \\
\mathbf{L_2} &= \begin{bmatrix} 29326.8098 & -43.3994 & -129.2942 \\ -43.3994 & 20781.5826 & 1348.6924 \\ -129.2942 & 1348.6924 & 22807.3271 \end{bmatrix} \text{kg} - \text{mm}^2
\end{aligned}
$$

Frame 3

$$\begin{aligned}
m_3 &= 1.7251\text{kg} \\
\mathbf{p_{m,3}} &= \begin{bmatrix} -38.7565 \\ 217.9078 \\ 0.0252 \end{bmatrix} \text{mm} \\
\mathbf{L_3} &= \begin{bmatrix} 56662.2970 & -2321.6892 & 8.2125 \\ -2321.6892 & 3158.0509 & -16.6307 \\ 8.2125 & -16.6307 & 56806.6024 \end{bmatrix} \text{kg} - \text{mm}^2
\end{aligned}$$

Frame 4

$$\begin{aligned}
m_4 &= 2.1727\text{kg} \\
\mathbf{p_{m,4}} &= \begin{bmatrix} 5.5341 \\ 0.0682 \\ 119.2769 \end{bmatrix} \text{mm} \\
\mathbf{L_4} &= \begin{bmatrix} 10.6749 & 0.0450 & -1.3556 \\ 0.0450 & 10.5866 & -0.1100 \\ -1.3556 & -0.1100 & 2.8204 \end{bmatrix} \text{kg} - \text{mm}^2
\end{aligned}$$

Frame 5

$$\begin{aligned} m_5 &= 0.3566\text{kg} \\ \mathbf{p_{m,5}} &= \begin{bmatrix} 0.0548 \\ 28.8629 \\ 1.4849 \end{bmatrix} \text{mm} \\ \mathbf{L_5} &= \begin{bmatrix} 0.3711 & -0.0001 & -0.0000 \\ -0.0001 & 0.1943 & -0.0161 \\ -0.0000 & -0.0161 & 0.3821 \end{bmatrix} \text{kg} - \text{mm}^2 \end{aligned}$$

Frame 6

$$\begin{aligned} m_6 &= 0.4092\text{kg} \\ \mathbf{p_{m,6}} &= \begin{bmatrix} -0.0592 \\ -16.8612 \\ 24.1905 \end{bmatrix} \text{mm} \\ \mathbf{L_6} &= \begin{bmatrix} 0.5489 & 0.0002 & -0.0001 \\ 0.0002 & 0.2385 & -0.0443 \\ -0.0001 & -0.0443 & 0.4513 \end{bmatrix} \text{kg} - \text{mm}^2 \end{aligned}$$

Frame 7

$$\begin{aligned}
m_7 &= 0.0755\text{kg} \\
\mathbf{p_{m,7}} &= \begin{bmatrix} 0.1484 \\ 0.0725 \\ -3.3579 \end{bmatrix} \text{mm} \\
\mathbf{L_7} &= \begin{bmatrix} 0.0391 & 0.0002 & 0.0000 \\ 0.0002 & 0.0388 & 0.0000 \\ 0.0000 & 0.0000 & 0.0761 \end{bmatrix} \text{kg} - \text{mm}^2
\end{aligned}$$

Barrett Hand

$$\begin{aligned}
m_{BH} &= 1.20\text{kg} \\
\mathbf{p_{BH}} &= \begin{bmatrix} 0.0000 \\ 0.0000 \\ -0.0320 \end{bmatrix} \text{mm}
\end{aligned}$$

C.2 Joint Limits

Reiterated many times earlier, position controlled devices, like the RTUs, are inherently unsafe. Because of their high accuracy and high bandwidth, the RTUs can severely damage anything in the workcell in the blink of an eye. To avoid such a catastrophe, we generated an additional control block that limits the RTU motion to the track between its home position and the other RTU. A virtual spring, with a stiffness of 3 N/mm, is added to the controller to repel either of these boundaries.

When the RTU is within 0.2977 meters (50,000 cts) to said obstacle, the virtual spring becomes engaged and forces the sled in the other direction.

Joint	**Positive Joint Limit**	**Negative Joint Limit**
	rad	*rad*
1	2.6	-2.6
2	2.0	-2.0
3	2.8	-2.8
4	3.1	-0.9
5	1.3	-4.8
6	1.6	-1.6
7	2.2	-2.2

Table C.1: WAM Joint Limits

Similarly, the physical joint limits of the WAMs must be avoided. Threshold joint limits are set to activate rotational springs at each joint - 0.03m radians from the actual limits (Table A.2).

References

[1] Wam arm: User's guide., 2006.

[2] J. Kenneth Salisbury. Active stiffness control of a manipulator in cartesian coordinates. In *Decision and Control including the Symposium on Adaptive Processes, 1980 19th IEEE Conference on*, volume 19, pages 95–100, December 1980.

[3] N. Hogan. Impedance control - an approach to manipulation. i - theory. ii - implementation. iii - applications. *ASME Transactions Journal of Dynamic Systems and Measurement Control B*, 107:1–24, March 1985.

[4] Homayoun Seraji and Richard Colbaugh. Force tracking in impedence control. *I. J. Robotic Res.*

[5] Toru Tsumugiwa, Ryuichi Yokogawa, and Kei Hara. Variable impedance control based on estimation of human arm stiffness for human-robot cooperative calligraphic task. In *ICRA*, pages 644–650, 2002.

[6] Seul Jung, T.C. Hsia, and R.G. Bonitz. Force tracking impedance control of robot manipulators under unknown environment. *Control Systems Technology, IEEE Transactions on*, 12(3):474 – 483, May 2004.

[7] S.A. Schneider and Jr. Cannon, R.H. Object impedance control for cooperative manipulation: theory and experimental results. *Robotics and Automation, IEEE Transactions on*, 8(3):383–394, June 1992.

[8] Ali Moosavian, , S. Ali, A. Moosavian, and Evangelos Papadopoulos. Multiple impedance control for object manipulation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 461–466, 1998.

[9] Haipeng Xie, I.J. Bryson, F. Shadpey, and R.V. Patel. A robust control scheme for dual-arm redundant manipulators: experimental results. In *Robotics and*

Automation, 1999. Proceedings. 1999 IEEE International Conference on, volume 4, pages 2507–2512 vol.4, 1999.

[10] F. Caccavale, P. Chiacchio, A. Marino, and L. Villani. Six-dof impedance control of dual-arm cooperative manipulators. *Mechatronics, IEEE/ASME Transactions on*, 13(5):576–586, October 2008.

[11] R.G. Bonitz and T.C. Hsia. Robust internal-force based impedance control for coordinating manipulators-theory and experiments. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 1, pages 622–628 vol.1, April 1996.

[12] H. Seraji. An on-line approach to coordinated mobility and manipulation. pages 28 –35 vol.1, May 1993.

[13] K. Ward and R. Arkin. Reactive control of a mobile manipulator using pseudo-joint damping, 1994.

[14] O. Khatib, K. Yokoi, K.-S. Chang, D. Ruspini, R. Holmberg, and A Casal. Coordination and decentralized cooperation of multiple mobile manipulators., 1996.

[15] Yoshio Yamamoto and Xiaoping Yun. Coordinating locomotion and manipulation of a mobile manipulator. *IEEE Transactions on Automatic Control*, 39:1326–1332, 1987.

[16] D. Tsetserukou, R. Tadakuma, H. Kajimoto, N. Kawakami, and S. Tachi. Intelligent variable joint impedance control and development of a new whole-sensitive anthropomorphic robot arm. pages 338 –343, June 2007.

[17] Werner F. Schindler and Ciro Natale. *Interaction Control of Robot Manipulators: Six Degrees-of-Freedom Tasks.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[18] Richard P. Paul, Burce Shimano, and Gordon E. Mayer. Differential kinematic control equations for simple manipulators. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(6):456–460, June 1981.

[19] J. Y. S. Luh, M. W. Walker, and R. P. C. Paul. On-line computational scheme for mechanical manipulators. *J. DYN. SYS. MEAS. & CONTR.*, 102(2):69–76, 1980.

[20] Armando Bellini and Stefano Bifaretti. A digital filter for speed noise reduction in drives using an electromagnetic resolver. *Math. Comput. Simul.*, 71(4):476–486, 2006.

[21] A Hourtash. The kinematic hessian and higher derivatives. In *CIRA 2005 Proceedings*, International Symposium on Computational Intelligence in Robotics and Automation series, pages 169–174. IEEE, June 2005.