Cooperative Object Manipulation with Force Tracking on the da Vinci Research Kit

by

Radian Azhar Gondokaryono

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Robotics Engineering

by

August 2018

APPROVED:

Professor Gregory S. Fischer, Thesis Advisor

Professor Jie Fu, Thesis Committee Member

Professor Loris Fichera, Thesis Committee Member

Abstract

The da Vinci Surgical System is one of the most established robot-assisted surgery device commended for its dexterity and ergonomics in minimally invasive surgery. Conversely, it inherits disadvantages which are lack of autonomy and haptic feedback. In order to address these issues, this work proposes an industry-inspired solution to the field of force control in medical robotics. This approach contributes to shared autonomy by developing a controller for cooperative object manipulation with force tracking utilizing available manipulators and force feedback. To achieve simultaneous position and force tracking of the object, master and slave manipulators were assigned then controlled with Cartesian position control and impedance control respectively. Because impedance control requires a model-based feedforward compensation, we identified the lumped base parameters of mass, inertias, and frictions of a three degree-of-freedom double four-bar linkage mechanism with least squares and weighted least squares regression methods. Additionally, semidefinite programming was used to constrain the parameters to a feasible physical solution in standard parameter space. Robust stick-slip static friction compensation was applied where linear Viscous and Coulomb friction was inadequate in modeling the prismatic third joint. The Robot Operating System based controller was tested in RViz to check the cooperative kinematics of up to three manipulators. Additionally, simulation with the dynamic engine Gazebo verified the cooperative controller applying a constant tension force on a massless spring-damper virtual object. With adequate model feedback linearization, the cooperative impedance controller tested

on the da Vinci Research Kit yielded stable tension force tracking while simultaneously moving in Cartesian space. The maximum force tracking error was +/- 0.5 N for both a compliant and stiff manipulated object.

Acknowledgements

For my advisor, comitee members, family, and friends, thank you for all the support and motivation on this journey. I would like to give a special acknowledgement to my late father who, in his time was a great engineer/professor, showed me hard work, honesty, pride, and persistance. Also, to my significant other, which this work would not have been completed without your love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Robot-Assisted Surgery (RAS) technology has been developed since 1984 [4] and has established itself in modern robotic surgeries ranging from cardiac, colorectal, gynecologic, thoracic, and urologic surgeries [6]. With the advent of RAS, Camarillo, et. al [4] explained the advantages and disadvantages of robotic surgeries versus traditional surgery (Table 1.1) where RAS provided accuracy, diverse sensors, and stability while also inheriting drawbacks, for example, non-versatility and premature technology. In 2000, the daVinci Surgical System developed by Intuitive Surgical acquired FDA approval and quickly gained commercial success for its dexterous and ergonomic Endowrist manipulator which provided many advantages for general Minimally Invasive Surgery [7]. While this innovative device is adopted among surgeons worldwide, it lacks autonomy. It can be seen in table 1.2 that other RASs adopt supervised or shared control strategies. For example, the NeuroMate utilizes CT scans to apply trajectory planning which the surgeon must approve and execute. Another example is the ACROBOT, which uses a virtual fixture to constrain the robot motion to a region of safety during operation. Additionally, a second major disadvantage of the da Vinci Surgical System resides in the fact that this method

detaches the surgeon from natural sensory of the fingers [8]. This sense, called tactile or haptic feedback in the robotics community [9], is broadly defined as interactions between robots, humans, and real/simulated environments and their combinations. It relates to the skin's sense of force, pressure, vibrations, and/or temperature of the environment. This problem has encouraged several research initiatives addressing the application of haptic feedback with force sensors and low-inertia haptic devices to RAS.

The general hardware of the daVinci Surgical System with additional research advancements on force feedback presented an opportunity for the emergence of innovative software and control. Some examples include virtual fixtures [10], trajectory smoothing [11], and automated camera control [12]. In particular relevance to this work, Haidegger et. al [13] reviews force control strategies applied to surgical robotics to achieve a variety of tasks with shared or supervised control. Force control is an approach to achieve versatile and robust behavior in uncertain environments and in the presence of modeling errors due to contact forces [14]. Additionally, a recent development in shared autonomy included dual arm object manipulation which gained considerable traction due to the successful implementations in industrial robotics [15]. Here, the objective is to provide an object motion with a controller that uses the cooperation of two manipulators. The daVinci however, teleoperates each manipulator with an individual hand of the surgeon. This results in one standby manipulator that could be better utilized in cooperative object manipulation with one of the teleoperated manipulators.

Table 1.1: Advantages and disadvantages of human and robot capabilities [4]

	Surgeons	Robots
Advantages	Task versatility Judgement experience Hand-eye coordination Dexterity in mm-cm scale Many sensors Quickly process extensive and diverse qualitative information	Repeatability Stability and accuracy Tolerant of ionizing radiation Diverse sensors Optimized for particular environment Spatial hand-eye transformations handled with ease Manage multiple simultaneuos tasks
Disadvantages	Tremors Fatigue Imprecision Variability in skill, age, state of mind Inability to process quantitative information easily Ineffective at submillimeter scale	Expensive Cumbersome Large Inability to process qualitative information Not versatile Technology still in infancy

Table 1.2: Robotic systems with various levels of autonomy [5]

Name	Branch of Surgery	Level of Autonomy
da Vinci Surgical System	General minimally-invasive	Direct control
EndoBot	General minimally-invasive	Direct, shared control, supervised
Trauma Pos	General	Direct control
Sensei Robotic System	Cardiac	Direct control
NeuroMate	Neurosurgery	Supervised
ACROBOT	Orthopedic	Shared control
ROBODOC	Orthopedic	Supervised

1.1 Motivation

On the basis of previous technological and research advances, we summarize the motivation of this work:

- The daVinci Surgical System provided no autonomy incorporated into the algorithms where surgeons are only able to control two out of three manipulators at any given time. Shared autonomy of the remaining manipulator could improve surgical tasks and procedures.

- With the advent of force sensors in haptic feedback, a new opportunity for automated research in robot assisted surgery has ensued. Adoption of force control that has been widely used in industrial robotics would establish new medical procedures and enhancements of the daVinci Surgical System.

1.2 Goal

This research aims to develop controllers for cooperative object manipulation with two or three daVinci Surgical Manipulators. This entails applying a referenced tensile/compression force while simultaneously moving the object in Cartesian space.

1.3 Contributions

There are six main contributions of this work:

1. A kinematic and dynamic model for the Patient Side Manipulator (PSM) was developed using motion capture system, conventional DH notation, and Euler-Lagrange energy based method. In general, this model construes a double-parallelogram 5-link remote center of motion mechanism which, to the authors knowledge, is rarely discussed in literature.

2. Parameter identification of the PSM was conducted to eliminate non-linear dynamics through feed-forward control providing a basis for more advanced controllers. Identification is done on a simplified 3 degree-of-freedom (DOF) mathematical model utilizing least squares/weighted least squares regression and Semi-Definite Programming (SDP) to obtain the lumped base parameters of masses, mass locations, inertias, and frictions. SDP provides physically feasible standard parameter solutions. Additionally, identification requires Fourier series trajectory optimization to obtain an optimal trajectory minimizing the errors of the least squares solution which infers the non-excitation of unmodelled dynamics.

3. A robust stick-slip static friction compensation was applied to alleviate parameter identification errors of the prismatic joint. The compensation utilizes

the maximum stick-slip friction identified during a force ramp up test.

4. A Robot Operating System (ROS) based cooperative object manipulation with force tracking controller was developed and tested in RViz and Gazebo simulations. The RViz framework enabled the kinematics of any number of manipulators to be tested for Cartesian movement, object reorientation, and object force increase/decrease movement. The Gazebo framework facilitated the dynamic interaction of manipulator and object which verified the stability of force tracking while simultaneously moving in Cartesian space.

5. A computed torque controller and impedance controller with feedforward model-based compensation and robust friction compensation was applied to the PSM manipulator.

6. A cooperative object manipulation with tensile force tracking controller was developed and tested for two PSMs on a compliant and stiff environment.

1.4 Outline

The outline of the thesis is as follows: Chapter 2 reviews related work of force control, cooperative control, medical applications of cooperative control, and parameter identification. Chapter 3 discusses the system architecture including hardware, and software of the daVinci Research Kit. Chapter 4 presents the Kinematics and Dynamics mathematical modelling to form the standard used in this thesis. Chapter 5 discusses the dynamic identification in detail, for example, mathematical linearization, base parameter lumping, Fourier series trajectory optimization, and physical feasibility Semi-Definite Programming (SDP). In Chapter 6, a simulation environment was developed and utilized for the development of cooperative control kine-

matics. Chapter 7 elaborates on methods such as nonlinear feedback linearization, computed torque controller, and impedance control. In addition to this, the chapter also discusses controller architecture, test setup, and kinematic calibration. Finally, chapter 8 states the conclusions of this work and future areas of research.

Chapter 2

Related Work

This chapter presents prior research contributions that serve as a foundation for the following chapters. We discuss the capabilities and applications of force and haptic feedback in medical settings, the development of force control from industrial robotics to medical robotics, the development of cooperative object manipulation in research institutes, the medical applications that could potentially benefit from cooperative manipulation, as well as relevant parameter identification research used throughout this work.

2.1 Force Feedback and Haptic Sensing

The development of haptic sensing technology allowed surgeons to sense forces and motions during teleoperation of robotic-assisted surgeries. Therefore, research groups explored various methods of applying force sensors in medical robotics. Okamura et al. [16] developed a modular 2 degree-of-freedom (DOF) force sensing instrument for laparoscopic surgery. The biocompatible device is specifically designed to restore sensory information to minimally invasive surgery and therefore works for a variety of 5 mm tools. As opposed to previous work, it succeeded in mea-

suring intra-abdominal forces constraining laparoscopic surgery. Rosen et al. [17] developed an effect controlled teleoperated endoscopic grasper. The force control of the grasper enables force sensing by allowing the surgeon to discriminate between different types of soft tissues. Kim et al. [18] developed force sensing capabilities for a more tactile sensory feedback measuring normal and shear forces from the tool end effector for minimally invasive surgery. The hardware consists of four capacitive transducers calibrated and validated using the open surgical system Raven-II [19]. It proved to have adequate measurements, hysteresis, and calibration to be used for tissue damage prevention.

2.2 Force Control

The emergence of force sensing capabilities encouraged the development of force control to achieve stable interaction with uncertain environments. For example, when a robot hits an obstruction along its trajectory, it will not suddenly apply a large force on the object due to the proportional gain on the joints, but rather, will be compliant and apply a force ramp-up with large position errors in the controller.

Force control has been developed for industrial robots [14] and is categorized into indirect and direct force control. Indirect force control is robust to a particular environment without actually measuring the forces of contact. Stiffness control, impedance control and admittance control fall into this category. Direct force control is the hybrid position/force control where some Cartesian/orientation degrees of freedom are constrained to position control, and other DOFs are force controlled. Surgical tasks operate in uncertain environments where there are no quantitative data, for example, on the size or shape of the object. They also operate in dynamic environments where, for example, the volume of lungs changes in a relatively small

amount of time. For this reason, recent medical robotics research focused on indirect force control and how it applies to robotic surgical tasks [13]. For example, John Hopkins University applied admittance control on the five degree-of-freedom (DOF) NeuroMate stereotactic robot for bone drilling applications [20]. Depending on the force applied to the master tool, the robot moves in a specific direction with a certain velocity. Another example is the MIRO manipulator at DLR which uses an impedance control law that allows users to guide the robot to the desired position [21]. As a safety measure, torque sensing at the joint level provided compliance in case of a collision. The resulting overview suggests that impedance/admittance control is practical for environment manipulation in surgery.

The fundamental difference between this dual concept of impedance and admittance controllers is explained in [22]. In impedance control, the plant/manipulator interacts with the controller and environment through forces. In admittance control, the controller applies input positions to the plant and produces force outputs to the environment. This configuration causes the impedance controller to be stable for stiff environments and the admittance controller to be stable for compliant environments. It is apparent that the correct choice of force control depends on the dynamic properties pertaining to the environment.

2.3 Cooperative Object Manipulation

This work, in essence, aims to develop shared autonomy for the daVinci Surgical system, therefore, it explores various contributions in cooperative object manipulation. The earliest works were published by Schneider [23] at Stanford University. Schneider's project analyzed a complex dynamic control problem of multiple-armed robotic systems. Also, they developed an object impedance controller (OIC) which

utilizes object level impedance thus, indirectly controlling an object's internal forces. Tests were conducted to prove the stability of the controller in free motion and environmental contact.

Because of OIC stability considerations in flexible environments, Meer et al. [24] developed the condition that allows, with little modifications, stable OIC if the grasp matrix rank is higher than the degrees of freedom of the object. Additionally, they also developed a flexible impedance controller (FIC), modified from the OIC, [25] which provides intuition on how to use an admissibility matrix to achieve stability criterions using Nyquist and Bode plots. The system stability can be achieved by choosing actuator mass, object mass, desired mass, and force filter cutoff frequency.

Moosavian et al. [26] developed a different approach to address the issue of OIC instability during interaction with a flexible object. They created the multiple impedance control (MIC) for object manipulation which takes into account the impedance of multiple end-effectors as well as the impedance of the object. The MIC simulation resulted in controllable tracking errors and object inertia compensation for flexible and massive objects under high accelerations.

2.4 Medical Applications of Cooperative Object Manipulation

Exploration of cooperative object manipulation contributions in dual manipulation [15], multiple robot systems [27], and cooperative aerial robots [28], suggests that cooperative manipulation for robot-assisted surgery is an oppurtunity for novel research. Therefore, this thesis explores potential applications of robot surgery requiring the coordination of two manipulators or more.

A necessary task in minimally invasive surgery is needle suturing. The gen-

eral procedure includes the orientation of the needle, insertion into the object, re-grasping on the other side, pulling, then tying a knot [29]. Force/cooperative control can be applied during the pulling, re-grasping, and trying knot steps. The individual suture materials and general sutures tested in [30] concludes that the modulus of elasticity of suture materials ranges from 300 to 4000 MPa and the break strength ranges from 15 to 50 N.

A promising application for cooperative object manipulation control could be medical grafts, which is a living tissue for surgical transplants. For specific grafts such as vascular grafts that connect blood vessels, a study concluded that excess shear forces between the graft and native vessel result in poor long-term patency [31]. Another specific case would be in ligament reconstruction surgery [32], where grafts are pre-tensioned before suturing. Additionally, skin grafts mostly use the tie-over bolster dressing technique which requires precision in applying correct tension and pressure over a small grafting area [33].

Video recordings of actual minimally invasive surgery using the da Vinci Surgical System show that the surgeon would use the redundant manipulator to retract an obstruction resulting in a larger workspace of the teleoperated manipulators.

2.5 Parameter Identification

Force control requires feedforward model-based linearization of nonlinear robot dynamics [34]. Since the daVinci Surgical System is a proprietary product, information such as link lengths, mass, inertias, and frictions are unavailable to researchers. Therefore, it is required to identify the dynamics with a method known as parameter identification.

A thorough search of the relevant literature yielded that parameter identification

started in 1974 for a NASA project to identify the dynamics of a 6 DOF Jet Propulsion Laboratory Robot Research Project Manipulator [35]. Wisama Khalil [36] started research on parameter identification in the 90's and formulated the mathematical foundation for a complete methodology including least squares regression, base parameter identification, and trajectory optimization which functions to estimate parameters of robot manipulators mainly focusing on industrial manipulators. For the da Vinci Surgical System, parameter identification has been done for the Patient Side Manipulator (PSM), and Master Tool Manipulator (MTM) [37] and resulted in significant torque prediction errors (30-40 %). This work implements many research advancements about the aforementioned parameter identification procedure further explained in Chapter 4.

Chapter 3

System Architecture

This chapter explains the hardware and software of the da Vinci Surgical System and the da Vinci Research Kit (dVRK). The dVRK is a retired commercial da Vinci Surgical System donated to research groups by Intuitive Surgical.

3.1 da Vinci Surgical System

The da Vinci Surgical System, was formally created in 2000 when it received its Food and Drug Administration (FDA) approval [7]. This robot allows surgeons to perform accurate and precise minimally invasive surgical procedures with its dexterous Endowrist manipulators. The system also improved conventional laparoscopic surgery in ways that enable surgeons to control the camera movement as opposed to having a patient side assistant manually adjusting the camera.

The da Vinci Surgical System consists of: An ergonomically designed surgeon's master console with two 7 degree-of-freedom (DOF) master tool manipulator (MTM) arms with camera viewport; a patient cart set up joint with three 6 DOF patient side manipulators (PSM) and one 5 DOF endoscope camera manipulator (ECM); and a high-performance vision stereoscopic camera endoscope. Figure 3.1 shows the

Figure 3.1: The da Vinci Surgical System at Automated and Interventional Medicinal Laboratory

system set up at Worcester Polytechnic Institute (WPI): Automated Interventional Medicinal Laboratory(AIMLab)

3.1.1 Master Tool Manipulator

The MTMs function to teleoperate the surgeons' hand movements to PSM motion. It consists of two left and right 7 DOF manipulators corresponding to each hand of the surgeon which gives an ergonomic feel while controlling the Patient Side Manipulator (PSM). Figure 3.2 shows the joint movements of each DOF. Every joint is equipped with a brushless motor to provide gravity compensation when the surgeon releases the manipulator. The first three joints of the MTM allow it to move in Cartesian space. To give natural wrist motion, the last four joints (joints 4-7) construct a gimbal mechanism with all axes intersecting at one point where the

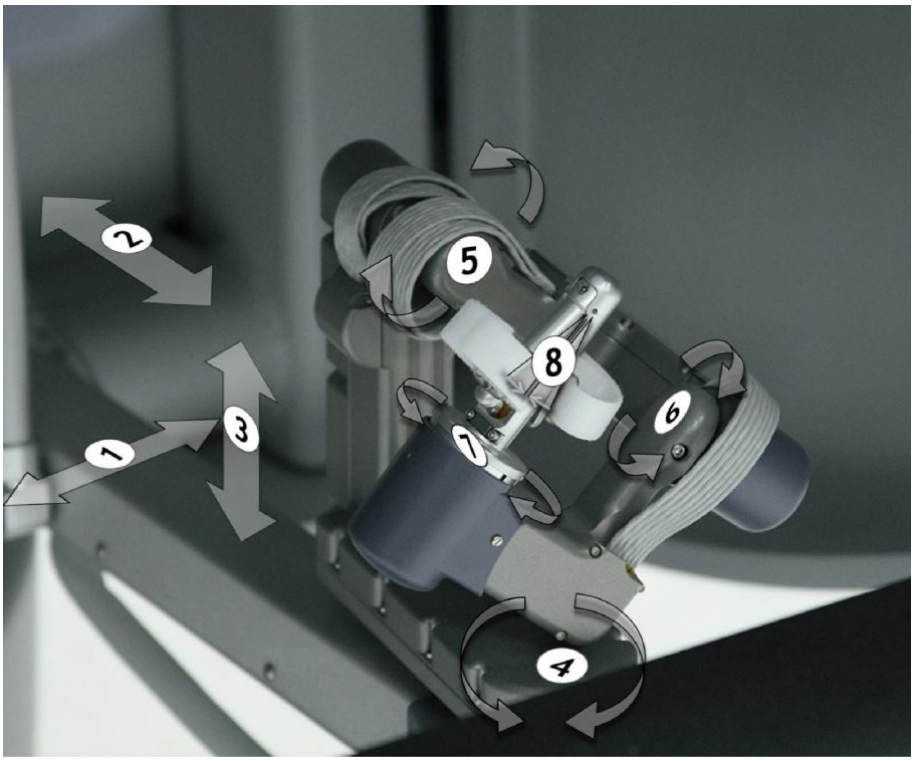


Figure 3.2: Master Tool Manipulator [3]

fingers pinch the tool. Some consider the MTM to have 8 DOF where the 8th joint is used to pinch the PSM gripper.

3.1.2 Patient Side Manipulator

The PSM is a 7 DOF manipulator actuated with Maxon brush DC motors. All motors are located near the base where gear transmissions are used to actuate the first 2 DOFs and cable transmissions are used to actuate the remaining 5 DOFs. On all joints, an absolute encoder measures position while Maxon motor Hall effect sensors measure velocity. Refer to figure 4.3 for detailed kinematics and motion of the PSM. The first joint actuates the yaw axis. The second joint is a double four-bar linkage consisting of 5 links (figure 4.3) that forms a pitch axis intersecting the yaw axis. This intersection constructs a virtual remote center of motion (RCM),

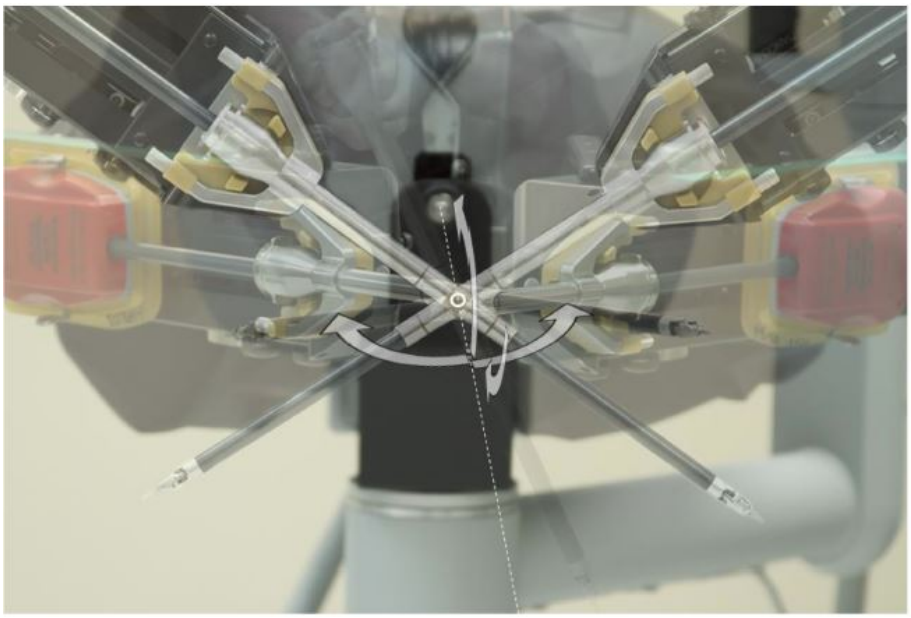


Figure 3.3: Remote center of motion mechanism of the Patient Side Manipulator. Source: dVRK Manual

Figure 3.3, which functions as the point of entry in minimally invasive surgery. The third joint prismatic joint then inserts the tool through this point. These three joints contribute to the Cartesian movement of the end-effector. The last four joints consisting of wrist roll, pitch, yaw, and gripper open and close function as the standard intersecting axis wrist designed for increased dexterity during operation.

3.2 da Vinci Research Kit

Figure 3.4 shows the da Vinci Research Kit (dVRK) setup at the Automation and Interventional Medicine Lab (AIMLab) at Worcester Polytechnic Institute (WPI) which consists of 2 PSMs, 1 ECM, 2 MTMs and the corresponding controllers. Since the da Vinci Surgical System is a proprietary product and provides limited data to researchers, hardware and firmware must provide access to low-level I/O, joint-level control, and teleoperation. Kazanzides et al. [38] provided an open-sodaurce research

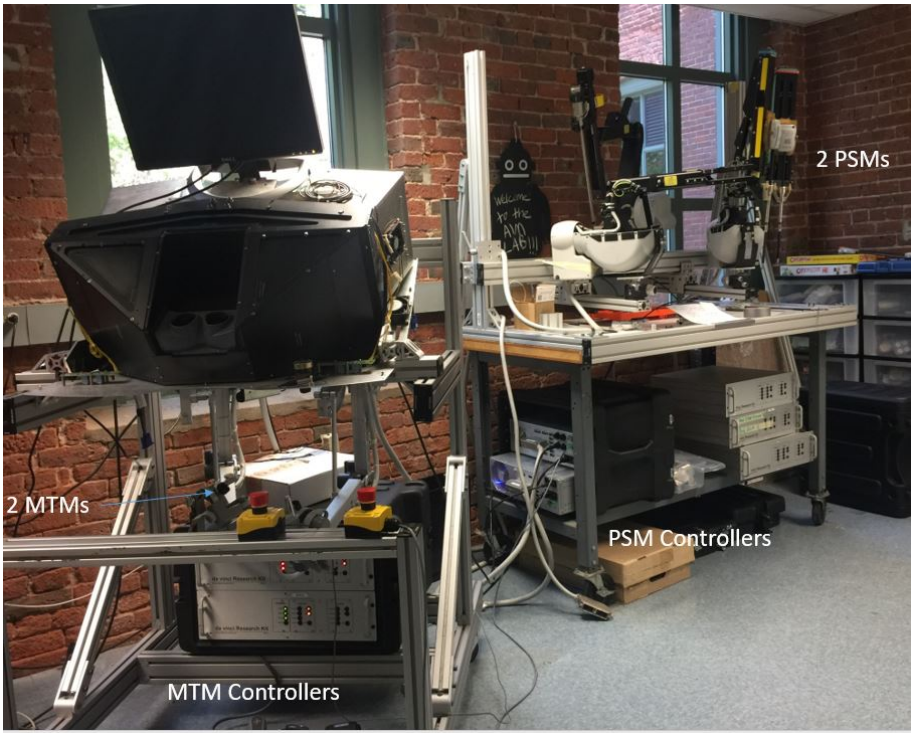


Figure 3.4: da Vinci Research Kit Hardware at Automated and Interventional Medicinal Laboratory

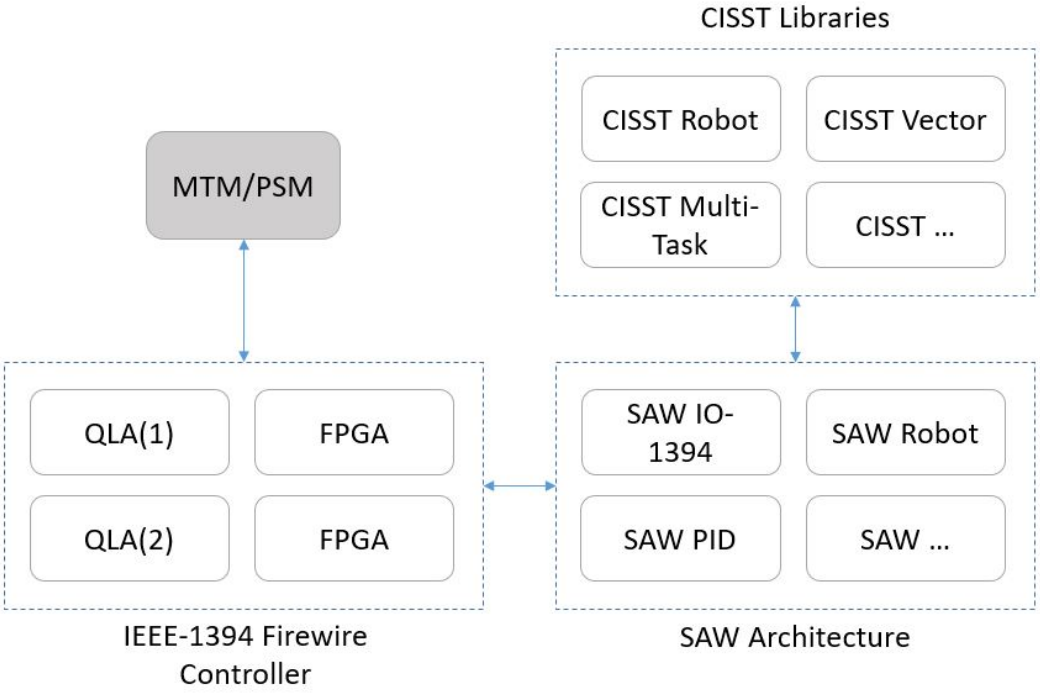


Figure 3.5: dVRK controller hardware and software architecture

platform consisting of firmware, electronics, and software to control da Vinci systems while fundamental data. This platform aims to provide researchers with an interface to implement new algorithms. Here, a centralized computation and distributed I/O system design was chosen to allow high-level control algorithms to be implemented on a computer while maintaining real-time hardware control on the I/O boards. The goal is realized with two IEEE-1394 FPGA boards and two quad-linear amplifiers packaged in an enclosure for each manipulator. The FPGA processes joint currents, joint positions, joint efforts, amplifiers temperatures, etc.

Figure 3.5 shows the architecture of the controller: the hardware on the left and the software on the right. The Research Kit Software is divided into functional layers: hardware interface (I/O), low-level control (e.g., PID), high-level control, teleoperation, and applications. With this type of structure, data is synchronized

up to 1 kHz. The research kit provides forward, and inverse kinematics, homing, and trajectory generation through the sawIntuitiveResearchKit (SAW) application that utilizes CISST libraries. SAW applications implement PID control laws which can run up to 6kHz. Additionally, gains can be set digitally through the software application.

Recently, John Hopkins University developed a Robot Operating Software (ROS) bridge interface that communicates with CISST-SAW [39]. ROS is a TCP/IP protocol messaging service that further simplifies the implementation of new controllers by creating independent thread nodes communicating with each other through messages being published and subscribed. In this work, they published satisfactory results of the real-time latency using ROS with histograms of 1KHz and 2KHz tests.

Chapter 4

Kinematic and Dynamic Models of the Patient Side Manipulator

A mathematical model of kinematics and dynamics of the Patient Side Manipulator (PSM) is used as building blocks to develop control algorithms in the later sections. Various research references are used to standardize this process.

4.1 Kinematics

The standard transformation matrix, $T \in SO^4$, is the relative position and rotation of one frame to another which can be formulated to describe the kinematics of a robot manipulator. To standardize this transformation, the conventional Denavit-Hartenberg (DH) [40] was used:

- z_i axis: direction of joint axis.
- x_i axis: parallel to common normal.
- y_i axis: right hand rule to complete z and x.

- d_i: Distance along z_{i-1} to x_i.

- a_i: z_{i-1} to current z_i.
- θ: Angle about z_{i-1}.
- α: Angle about common normal (x_i) (z_{i-1} to current z_i).

Technical documents, table 4.2, of the da Vinci Research Kit (dVRK) provides joint limits and maximum velocities of the PSM and MTM. Figure 4.2a gives a visualization of $q_1 - q_6$. The manual also provides DH parameters to represent accurate kinematics of the PSM. However, it did not describe the kinematics of the double four-bar linkage (Links 1, 2, 3, 4, and 5 in Figure 4.1). These kinematics are the basis that represent the motion of each body mass and body inertia that is used in the dynamics model. Therefore, we developed detailed kinematics for the PSM with matching output tool-tip kinematics.

4.1.1 Kinematics Measurement

A measurement technique using the optitrack motion capture is used to obtain unknown kinematics link lengths. The setup (Figure 4.1) uses six motion capture (Mocap) cameras to track a total of 18 optical markers (three markers on each link and three extra ones for the world position). Every three markers represent a rigid body transformation of position and rotation. The PSM is actuated along the full range of joint q_1 while q_2 is held constant at 0 radians, then along the full range of q_2 while q_1 is held constant at 0 radians. A least squares regression is used to identify the relative axis rotation position and orientation between any two given rigid bodies. We can then represent the axis locations in a global coordinate frame and later measure the distance between every axis along a plane parallel to the axis'. For further details about this method, refer to appendix A.1. The results of the link lengths are shown in Table 4.1 with notation in Figure 4.2a.

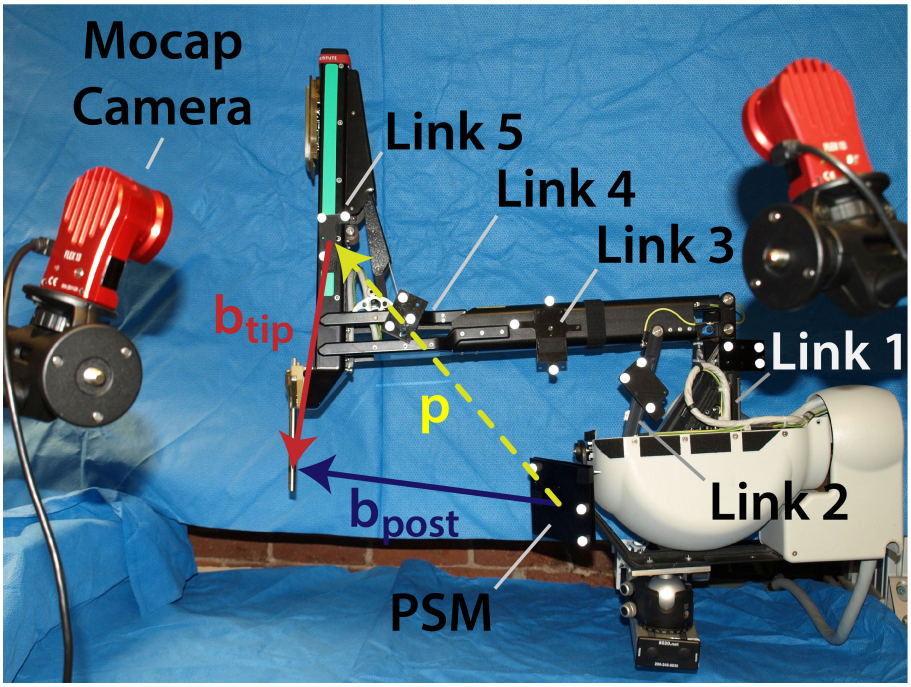


Figure 4.1: Motion tracking setup for calculating the link lengths and remote center of motion of the Patient Side Manipulator. Three Markers are placed on each link to represent a coordinate frame. The global coordinate frame is named PSM on the bottom of the picture.

Table 4.1: Kinematic Measurement Link length Results. For notations refer to figure 4.2a

	Link lengths [m]
l_1	0.150
l_2	0.150
l_{12}	0.00958
l'_2	0.1842
l_3	0.5152
l_4	0.5156

Table 4.2: PSM-MTM Joint Limits and Velocity Limits

PSM	1	2	3	4	5	6	7	8
Joint Upper Limit [rad]	1.58	0.93	0.24	3.03	3.03	3.03	3.03	
Joint Lower Limit [rad]	-1.58	-0.93	-0.24	-3.03	-3.03	-3.03	-3.03	
Max Velocity [rad/s]	2	2	0.4	6	5	5	-	
MTM								
Joint Upper Limit [rad]	1.27	1.19	0.72	1.68	3.27	0.82	7.92	0.52
Joint Lower Limit [rad]	-0.72	-0.35	-0.25	-3.60	-1.71	-0.82	-8.34	-
Max Velocity [rad/s]	1.1	1.1	1.1	2	2	2	2	-

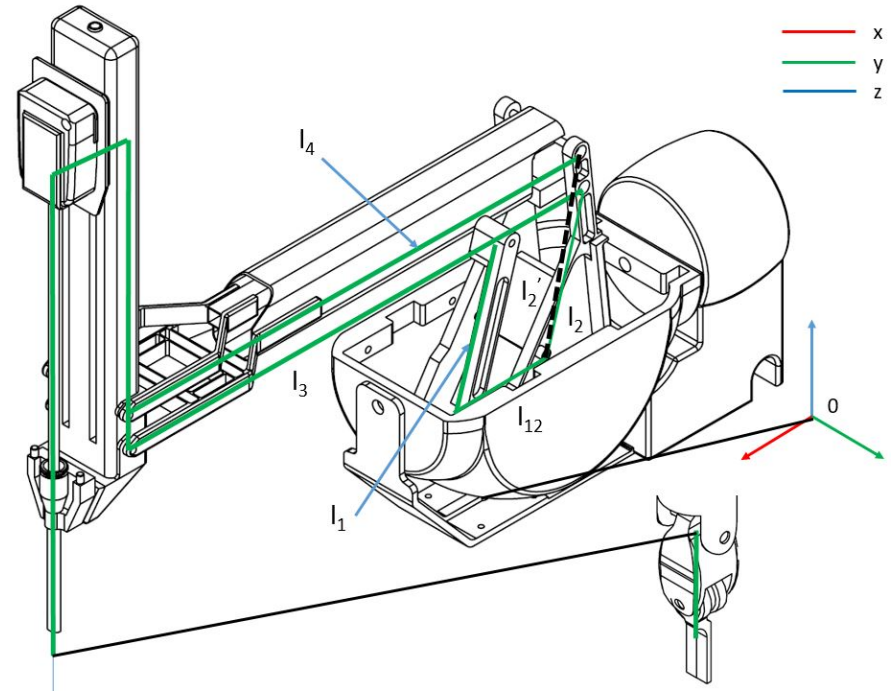
Table 4.3: Denavit-Hartenberg Parameters: Conventional

Frame	Parent Frame	a	α	d	θ
1	0	0	$-\pi/2$	0.1524	$\pi/2$
2	1	0	$-\pi/2$	0.0296	$-\pi/2 + q_1$
3	2	0.150	0	0	$-\beta + q_2$
4	3	0.516	0	0	$\beta + \pi/2 - q_2$
5	4	0.2881	0	0	$-\beta + q_2$
6	5	-0.0430	$\pi/2$	0	$-\pi/2$
7	6	0	0	q_3	$\pi/2$
8	7	0	$\pi/2$	0	$\pi/2 + q_4$
9	8	0.0091	$-\pi/2$	0	$\pi/2 + q_5$
10	9	0.0102	0	0	q_6
11	10	0	$-\pi/2$	0	$-\pi/2$
C1*	2	0	0	0	q_2
C2-int*	2	0	$-\pi/2$	0	$-\pi/2 - \beta + q_2$
C2*	C2-int	0	0	δq_3	0

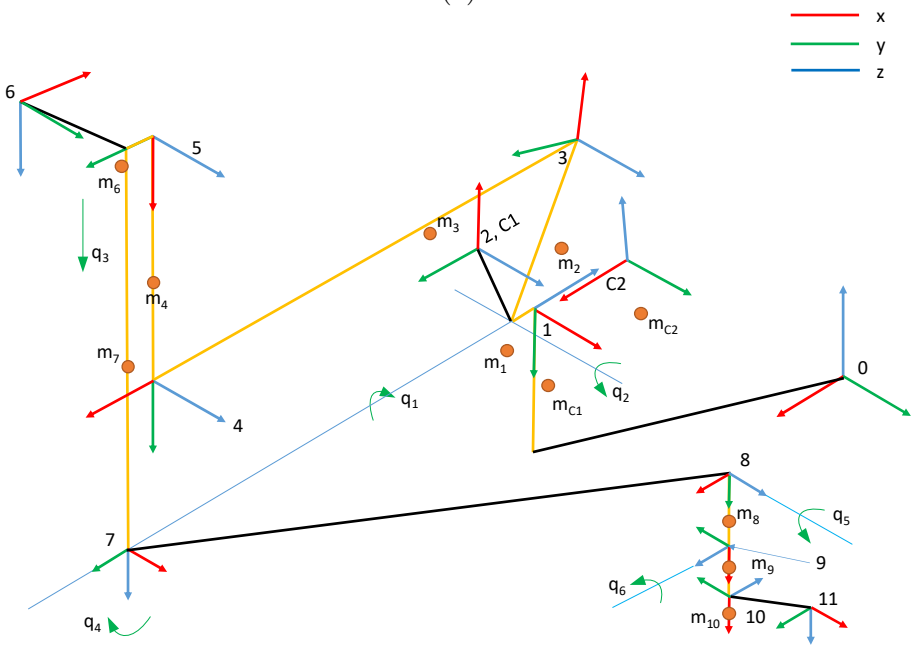
4.1.2 Patient Side Manipulator Kinematics

The resulting link lengths are used to construct a standard kinematic model with DH parameters (Table 4.3). Using these parameters, we develop a kinematic model (Figure 4.2b).

By observation of Figure 4.2, the double four-bar linkage with a total of 5 moving bodies (l_1 - l_5) was lumped to 3 moving bodies (m_2, m_3, and m_4) actuated by q_2 which simplifies the model while still accurately describing the dynamics of the robot. Counterweight m_{c1} is added and actuated by q_2. The frame C_1, with frame



(a)



(b)

Figure 4.2: (a) PSM showing the double parallelogram structure of the pitch joint. Notation $l's$ are for link length measurement results (b) Derived kinematic frames for every link of the Patient Side Manipulator. Frame 0 represents the base frame of the Manipulator and frame 11 represents the end effector frame. Estimated locations of the masses are shown. The double parallelogram structure is assumed to be a serial chain with links 2,3, and 4. m_{c1} and m_{c2} are counterweight masses that actuate according to joints q_2 and q_3 respectively where frames C1, and C2 follow DH convention.

2 parent, was added to follow DH convention. Another counterweight m_{c2} is added and actuated by δq_3. δ is a measured constant that represents a different range of motion. The additional parent-child tree is frame 2, C_2-int, then C_2. For this model, the last three joints $q_4 - q_6$ and the resulting end-effector frame coincide with the PSM kinematics in the manual. Frame 4's orientation is always horizontal to the z_0 axis of the robot.

The calculated end effector 6 x N Jacobian using equation 4.1 is used to check the correct motion of the mathematical model. N = degrees of freedom of robot. This equation is also used for dynamics derivation and in chapter 5 for force controllers.

$$J = \begin{bmatrix} J_v \\ J_o \end{bmatrix} \tag{4.1}$$

where the linear Jacobian, J_v:

$$J_v = \begin{bmatrix} \frac{d(p_1^0)}{dq_1} & \frac{d(p_2^0)}{dq_2} & \dots & \frac{d(p_N^0)}{dq_N} \end{bmatrix} \tag{4.2}$$

and the orientation Jacobian, J_o:

$$J_o = \begin{bmatrix} \frac{d(z_0^0)}{dq_1} & \frac{d(z_1^0)}{dq_2} & \dots & \frac{d(z_{N-1}^0)}{dq_N} \end{bmatrix} \tag{4.3}$$

4.2 Dynamic Model

Figure 4.2b shows the mass (m_i) and estimated center of mass (COM) locations of the dynamic model. The COMs are represented by a vector $r_i = [r_{ix}, r_{iy}, r_{iz}]$ that originates from T_i and rotates with T_{i+1} (The corresponding link joint angle). In order to simplify the model, some assumptions were developed as follows:

- Counterweight m_{c1} was lumped with m_2 as both were actuated by q_2.
- For the purpose of controllers, only the first three joints were considered in the dynamics model. The last 3 wrist masses (m were assumed to be lumped into the tool mass, frame 6, with $q_4 = q_5 = q_6 = 0$

As part of the previous assumption, there are 11 identifiable masses, $n = 11$, with three degrees-of-freedom, $N = 3$.

4.2.1 Euler-Lagrange Energy Method

Equations of motion were derived using the Euler-Lagrange energy-based method [40] which uses the derivatives of Kinetic and Potential Energy. To obtain the Kinetic Energy of the robot manipulator, the following equations were used:

$$K = \frac{1}{2}\dot{q}^T B(q)\dot{q} \tag{4.4}$$

where B is:

$$B(q) = \sum_{i=1}^{n} m_i J_v^{(m_i)T} J_v^{m_i} + J_o^{m_i T} R_{m_i} I_{m_i} R_{m_i}^T J_o^{m_i} \tag{4.5}$$

and Inertia tensor about its center of mass:

$$I_{m_i} = \begin{bmatrix} I_{i,xx} & I_{i,xy} & I_{i,xz} \\ I_{i,xy} & I_{i,yy} & I_{i,yz} \\ I_{i,xz} & I_{i,yz} & I_{i,zz} \end{bmatrix} \tag{4.6}$$

Where m_i is the mass at link i and I_i is the Inertia matrix of eq. (4.6). B describes the sum of kinetic energy of every link based off the mass m_i moving in the world frame (multiply $J_v^{m_i}$) and the energy of the rotating mass inertia I^{m_i} of mass i rotating

about the world frame (multiply by R, and $J_o^{m_i}$). $J_v^{m_i}$ is the linear Jacobian and $J_o^{m_i}$ is the orientation Jacobian at the point m_i presented in equations 4.2 and 4.2.

The Potential Energy (P) of the robot manipulator is:

$$P = -\sum_{i=1}^{n} m_i g_o^T p_{m_i} \tag{4.7}$$

where $g_o^T \in R^3$ is the vector describing the direction and magnitude of gravity acceleration and $p_{m_i} \in R^3$ is the vector describing the mass i location in world frame.

The kinetic and potential energy is used to derive the Lagrange (L) which is then used to derive the equations of motion for generalized forces (τ). In this case, the generalized forces are joint torques.

$$L(q, \dot{q}) = K(q, \dot{q}) - P \tag{4.8}$$

$$\frac{d}{dt}(\frac{\partial L}{\partial \dot{q}})^T - (\frac{\partial L}{\partial \dot{q}})^T = \tau \tag{4.9}$$

To verify the model, equation 4.9 is reformulated to the more common mass inertia(M), Coriolis (C), and gravity (G) matrices.

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = \tau_f + \tau_m \tag{4.10}$$

In this situation, the torque τ is due to friction torques τ_f and motor torques τ_m.

$$\tau = \tau_f + \tau_m \tag{4.11}$$

4.2.2 Friction Model

To completely model the dynamics, linear Viscous (F_v) and Coulomb friction (F_c) of each joint were added as [41]:

$$\tau_f = F_v q_d + F_c sign(q_d) \tag{4.12}$$

The equation states that F_v is proportional to the joint velocity q_d and F_c is a constant friction depending if either q_d is positive or negative. F_v is a n x n matrix where $F_v = diag(F_1, F_2, ..., F_n)$ (similar for F_c, F_{vo}, and F_{co}).

Friction in the positive velocity direction might not be the same as the negative velocity direction ($F_+ \neq F_-$). To model this situation, consider a Viscous and Coloumb friction offset, F_{vo} and F_{co}, which is suitable for linear regression. The resulting friction is:

$$\tau_f = F_v q_d + F_c sign(q_d) + F_{vo}(\delta_{i,0} - 1) + F_{co}(\delta_{i,0} - 1) \tag{4.13}$$

where $\delta_{qd_i,0}$ is the Kronecker Delta Function that returns a value 1 when $q_{d,i}$ is 0 and 0 otherwise. For control stability in Chapter 5, the sign function in equation (4.12) is changed to a continuous sigmoid function:

$$F_c sign(q_d) = F_c sigmf(q_d) = F_c(\frac{-2}{1 + \exp -ax} + 1) \tag{4.14}$$

The sigmoid function of F_c describes a continuous transition between positive and negative values of q_d. Parameter a, the slope of this transition, is chosen to be as small as possible which results in a stable controller for the system. The resulting friction used in this work is :

$$\tau_f = F_v q_d + F_c sigmf(q_d) + F_{vo}(\delta_{i,0} - 1) + F_{co}(\delta_{i,0} - 1) \tag{4.15}$$

where the Kroenecker Delta function is deleted because an exact zero velocity, $q_{d,i}$, is only achievable in theory. Therefore, the friction model simplified for linear regression is:

$$\tau_f = F_v q_d + F_c sigmf(q_d) + (F_{vo} + F_{co}) \tag{4.16}$$

When the friction model is used in a controller to compensate the robot friction in Chapter 7, we apply zero friction compensation when the joint velocity is below a certain threshold (deadband).

Chapter 5

Parameter Identification

For parameter identification, the identifiable parameters (δ) must be reformulated to δ_d such that it is linear to the regressor matrix Y:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) - \tau_f = Y\delta_d = \tau_m \tag{5.1}$$

where δ_d is the vector of linear lumped inertial parameters or standard parameters. Equations (5.2) and (5.3) are used to lump the nonlinear terms of mass m_i times mass location r_i in different combinations. The Matlab substitute function was used to find the combination of nonlinear terms in the left hand side of eq. (5.1) and replace with linear terms l_i and ll_i respectively.

$$l_i = \begin{bmatrix} l_{i,x} \\ l_{i,y} \\ l_{i,z} \end{bmatrix} = m_i r_i = \begin{bmatrix} m_i r_{i,x} \\ m_i r_{i,y} \\ m_i r_{i,z} \end{bmatrix} \tag{5.2}$$

$$ll_i = \begin{bmatrix} ll_{i,xx} & ll_{i,xy} & ll_{i,xz} \\ ll_{i,xy} & ll_{i,yy} & ll_{i,yz} \\ ll_{i,xz} & ll_{i,yz} & ll_{i,zz} \end{bmatrix} = m_i R_i = \begin{bmatrix} m_i r_{i,x}^2 & m_i r_{i,x} r_{i,y} & m_i r_{i,x} r_{i,z} \\ m_i r_{i,x} r_{i_y} & m_i r_{i,y}^2 & m_i r_{i,y} r_{i,z} \\ m_i r_{i,y} r_{i_z} & m_i r_{i,y} r_{i_z} & m_i r_{i,z}^2 \end{bmatrix} \tag{5.3}$$

In conclusion the vector δ_d of all standard parameters of the robot with N links is:

$$\delta_d = \begin{bmatrix} \delta_{d_1}^T & \delta_{d_2}^T & \delta_{d_2}^T & ... & \delta_{d_N}^T \end{bmatrix}^T \tag{5.4}$$

where:

$$\delta_{d_i} = \begin{bmatrix} I_{i,xx} & I_{i,xy} & I_{i,xz} & I_{i,yy} & I_{i,yz} & I_{i,zz} & l_{i,x} & l_{i,y} & l_{i,z} & ll_{i,xx} & ll_{i,xy} & ll_{i,xz} & ll_{i,yy} & ll_{i,yz} & ll_{i,zz} & m_i & f_i \end{bmatrix}^T \tag{5.5}$$

where f_i is the vector of all friction components of link i which in our case:

$$f_i = \begin{bmatrix} F_{c,i} & F_{v,i} & F_{co,i} & F_{vo,i} \end{bmatrix} \tag{5.6}$$

is the Coulomb friction, Viscous friction, Coulomb friction offset, and Viscous friction offset respectively (refer to eq. 4.16).

5.1 Base Parameters with SVD Decomposition

Not all parameters in δ_d give contribution to joint torques [42]. This leads to 3 categories of parameters:

- Independent identifiable parameters
- Identifiable parameters in linear combinations

- Unidentifiable parameters

Independent identifiable parameters are ones that have independent columns in Y. The identifiable parameters in linear combinations only contribute to joint torques in linear combinations. Therefore, the corresponding parameters can be grouped in linear combinations to preserve its contribution to joint torques and conversely. Similarly, corresponding columns in Y are deleted. Unidentifiable parameters do not contribute to the joint torques and are not in the model of equation (5.1).

From a linear algebra perspective, this is a rank deficient problem. By making the Y matrix into a full rank matrix, it is possible to have a linear least squares regression that estimates the parameters that correctly correspond to the measured output torques. These regrouped parameters which make the new Y matrix full rank is called the minimal base parameters, δ_b. A numerical method using the SVD decomposition [43] is modified to a more robust and automatic calculation of the base parameters [42]. Let Y be the (n x c) regressor matrix and δ_d be a vector of size c. n is the rows of the regressor matrix corresponding to the degree of freedom of the robot. c is the number of linear parameters in δ_d. The $rank(Y) = p < c$ where p is the dimension of the base parameters vector, δ_b. First, an SVD decomposition of Y gives:

$$Y = USV^T = U \begin{bmatrix} \sum & O \\ O & O \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix} \tag{5.7}$$

where $U \in R^{nxn}$, $V_1 \in R^{cxp}$, $V_2 \in R^{cx(c-p)}$, $\sum \in R^{pxp}$. $\sum$ is the diagonal matrix of singular values. U and V are orthagonal matrices.

Notice that the null columns of V_2^T correspond to the identifiable parameters. The remaining columns are identifiable by linear combinations. With this information, the columns of Y that correspond to absolutely identifiable, δ_i, and unidentifi-

able parameters are eliminated to generate a new regressor matrix Y' and parameter vector δ'_d. To represent this mathematically:

$$EY^T = \begin{bmatrix} Y_i^T \\ Y'^T \end{bmatrix} \tag{5.8}$$

and as for parameters:

$$E\delta_d = \begin{bmatrix} \delta_i \\ \delta'_d \end{bmatrix} \tag{5.9}$$

where E is an c x c permutation matrix. This matrix of identifiable parameters in linear combinations is derived by SVD:

$$Y' = U'S'V'^T = U' \begin{bmatrix} \sum' & O \\ O & O \end{bmatrix} \begin{bmatrix} V_1'^T \\ V_2'^T \end{bmatrix} \tag{5.10}$$

where

$$Y'V'_2 = O \tag{5.11}$$

The above equation shows that the columns of V'_2 define the linear combinations of regressor Y' that results in a null space. Using V_2, an equation is derived as:

$$Y'\delta'_d = Y'(\delta'_d + V'_2\delta'_a) \tag{5.12}$$

where $\delta'_a \in R^p$ is an arbitrary vector. The above equation proves that there are infinite solutions to the combined vector:

$$\delta'_r = (\delta'_d + V'_2\delta'_a) \tag{5.13}$$

Next, the columns of V_2' (also δ_d') can be rearranged by a permutation matrix E' such that:

$$E'^T V_2' = \begin{bmatrix} V_{21}' \\ V_{22}' \end{bmatrix} \tag{5.14}$$

$$E'^T \delta_d' = \begin{bmatrix} \delta_{d1}' \\ \delta_{d2}' \end{bmatrix} \tag{5.15}$$

$$E'^T Y'^T = \begin{bmatrix} Y_1'^T \\ Y_2'^T \end{bmatrix} \tag{5.16}$$

The purpose of this method is to acquire an E' matrix that makes V_{22}' a full rank square matrix. There exists many E' solutions. After, a solution to equation (5.13) is:

$$\delta_{r1}' = \delta_{d1}' - V_{21}' V_{22}'^{-1} \delta_{d2}' \tag{5.17}$$

where δ_{r1}' is the lumped linear combinations of parameters in δ_d'. This process is bijective by adding the codomain δ_{d2}' [44]:

$$\begin{bmatrix} \delta_{r1}' \\ \delta_{d2}' \end{bmatrix} = \begin{bmatrix} 1 & -V_{21}' V_{22}'^{-1} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \delta_{d1}' \\ \delta_{d2}' \end{bmatrix} = G\delta_A \tag{5.18}$$

Finally, the final regressor matrix Y_b is built:

$$Y_b = \begin{bmatrix} Y_i & Y_1' \end{bmatrix} \tag{5.19}$$

where Y_i are the columns of matrix Y with absolute identifiable parameters and Y_1'

are the columns of matrix Y' corresponding to the parameters δ'_{d1}. Similarly for the parameters:

$$\delta_b = \begin{bmatrix} \delta_i \\ \delta'_{r1} \end{bmatrix} \tag{5.20}$$

resulting in the final regressor equation:

$$Y_b \delta_b = \tau_m \tag{5.21}$$

Using equations (5.9), (5.16), (5.18), and (5.20) we are able to define a bijective function that maps from standard parameters to base parameters:

$$\begin{bmatrix} \delta_b \\ \delta'_{d2} \end{bmatrix} = m(\delta_d) \tag{5.22}$$

and its inverse

$$\delta_d = m^{-1}(\delta_b, \delta'_{d2}) \tag{5.23}$$

5.2 Least Squares Regression

Consider robot joint positions, velocities, accelerations, and torques measured at every time instant t_1, t_2, t_3, ..., and t_M. Therefore, equation (5.21) could be rewritten as [37]:

$$Y_B \delta_b = \begin{bmatrix} Y_b(t_1) \\ Y_b(t_2) \\ \dots \\ Y_b(t_M) \end{bmatrix} \delta_b = W_b \delta_b = \begin{bmatrix} \tau(t_1) \\ \tau(t_2) \\ \dots \\ \tau(t_M) \end{bmatrix} = \omega \tag{5.24}$$

where W_b is the resulting regressor matrix to a specific trajectory for M data points. Since numerical errors of different joint torques may affect the solution, a diagonal weighting matrix A is premultiplied to both sides of equation (5.21) where:

$$A = diag(a_1 \quad a_2 \quad ... \quad a_N) \tag{5.25}$$

and $a_i = 1/\tau_{i,max}$. Another weighting matrix P is used to normalize the different magnitudes in parameters where P is:

$$P = diag(\tfrac{1}{|W_{b,1}|} \quad \tfrac{1}{|W_{b,2}|} \quad ... \quad \tfrac{1}{|W_{b,c}|}) \tag{5.26}$$

where $|W_{b,c}|$ is the norm of column c of regressor matrix W_b. The δ_b using the normalized W_bP must be premultiplied by P to get the correct solution.

5.3 Fourier Series Optimal Trajectory

The trajectory for parameter identification must allow accurate estimation of the dynamics parameters [45] that does not excite unmodelled dynamics. By excitation of the identifiable parameters, the regressor matrix in equation (5.24) is a well-conditioned matrix. The condition number of a matrix is the largest singular value divided by the smallest singular value of a matrix. Intuitively, this is the sensitivity of the solution δ_b with respect to errors in W_b or τ. For closed-loop kinematic chain robots, a good regressor for parameter identification should have a condition number around 100 [46]. Therefore, the optimal trajectory should be a trajectory $q, \dot{q}, \ddot{q}$ that minimizes the condition number of regressor W_b:

$$minimize \quad cond(W_b(q(t))) \tag{5.27}$$

Method from [46] is applied to develop sinusoidal trajectories of the form:

$$q_i(t) = \sum_{i=1}^{l} \frac{a_l^i}{w_f l} sin(w_f l t) - \frac{b_l^i}{w_f l} cos(w_f l t) + q_{i_0} \tag{5.28}$$

where l is the number of fourier series harmonics and w is the fundamental frequency. Here l is chosen to be $l = 5$ and 6 different fundamental frequencies,$w = \begin{bmatrix} 0.1 & 0.12 & 0.13 & 0.14 & 0.16 & 0.2 \end{bmatrix}$ rad/s, were used to generate 6 different trajectories used to identify the model. Variables a_l^i, b_l^i, and q_{i_0} are variables to minimize the condition number. Here, the number of discrete data points ($M = 6000$) and s = 0, 1, 2, ... t_f/t_s, t_f, where the final time ($t_f = 30$ s) and the sampling time ($t_s = 0.005$ s), were chosen. The constraints are position and velocity limits of the PSM at every time in the trajectory. Refer to Table 4.2 for the limits.

$$q_{min} \leq q(st_s) \leq q_{max} \tag{5.29}$$

$$\dot{q}_{min} \leq q(st_s) \leq \dot{q}_{max} \tag{5.30}$$

Optimization uses Matlab fmincon "active-set" constrained optimization method. Table 5.1 gives the resulting condition numbers of the optimization results for six different trajectories with different fundamental frequencies, w. Figure 5.1 shows the position and velocity trajectory for three joints for 15 s with $w = 0.13$Hz.

5.4 Parameter Physical Feasibility Analysis

Souza, et. al [44] states that the mass inertia matrix, M, of equation (5.1) needs to be positive semidefinite [47] to be a physical realizable robot. Also, dynamic simulations would require positive principal inertias and masses. This paper introduced the

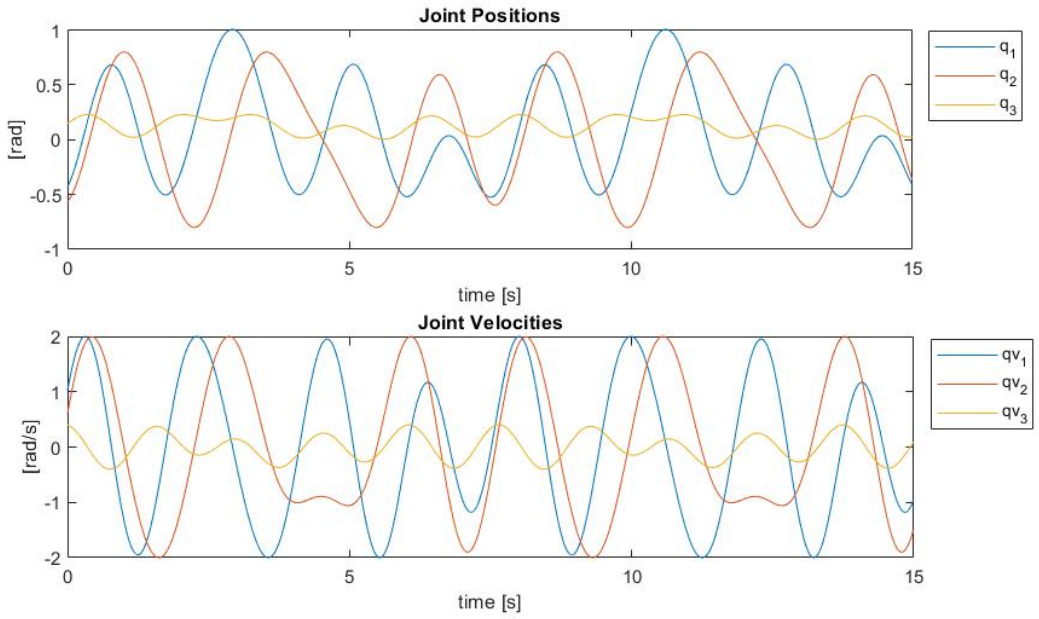


Figure 5.1: Resulting optimal trajectory for three joints when w = 0.13 rad/s and l = 5

problem and solution to the physical feasibility of parameter identification of robot manipulators. Here, constraining the positive semidefinite property of the inertia matrix I_i and the mass m_i of link i by constraints maintains the feasibility of matrix M:

$$D = (\delta \in R^n : m_i > 0, I_i > 0 | i = 1, ..., N) \tag{5.31}$$

and the same constraint in linear matrix inequality (LMI) form:

$$D_i(\delta_i) = \begin{bmatrix} I_i & O \\ O & m_i \end{bmatrix} = \begin{bmatrix} I_{i,xx} & I_{i,xy} & I_{i,xz} & 0 \\ I_{i,xy} & I_{i,yy} & I_{i,yz} & 0 \\ I_{i,xz} & I_{i,yz} & I_{i,zz} & 0 \\ 0 & 0 & 0 & m_i \end{bmatrix} \geq 0 \tag{5.32}$$

where $D_i \geq 0$ in this notation means that matrix D_i must be positive semidefinite. For a robot of N links the constraint is expanded:

$$D(\delta) = \begin{bmatrix} D_1(\delta_1) & 0 & ... & 0 \\ 0 & D_2(\delta_2) & ... & 0 \\ ... & ... & \ddots & ... \\ 0 & 0 & ... & D_N(\delta_N) \end{bmatrix}_{4Nx4N} \geq 0 \tag{5.33}$$

The inverse map from equation (5.23) is used to transform the standard parameter constraints D to the base parameter constraints D_{δ_b}.

$$D_{\delta_b}(\delta_b, \delta_{r2}) = D(m^{-}1(\delta_b, \delta_{r2})) \tag{5.34}$$

5.4.1 Semidefinite Programming (SDP) for Least Squares(LS) parameter identification

The least squares regression problem in the previous section is reformulated to a semidefinite programming optimization problem by minimizing the square of residual errors, $\|\epsilon\|^2$, where:

$$\|\epsilon\|^2 = \|\omega - W_b\delta_b\|^2 \tag{5.35}$$

The above equation is used to formulate a minimization statement:

$$\begin{aligned} &minimize_{(u,\delta_b)} && u \\ &subjectto && u \geq \|\omega - W_b\delta_b\|^2 \end{aligned} \tag{5.36}$$

By Schur complement [48], the above equation is formulated to SDP form:

$$\begin{array}{cc} minimize_{(u,\delta_b)} & u \\ subjectto & U_\omega(u,\delta_b) \geq 0 \end{array} \tag{5.37}$$

where:

$$U_\omega(u,\delta_b) = \begin{bmatrix} u & (\omega - W_b\delta_b)^T \\ (\omega - W_b\delta_b) & 1 \end{bmatrix} \tag{5.38}$$

$U_\omega(u,\delta_b)$ has sN+1 rows which might be too large for SDP formulations. Variable s is the number of data points along a trajectory. Therefore, a new formulation of the SDP problem is presented. For a base parameter regressor matrix, a QR decomposition is done:

$$W_b = QR = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_1 \\ 0 \end{bmatrix} \tag{5.39}$$

where:

$$\|Q^T\epsilon\|^2 = (Q^T\epsilon)^T(Q^T\epsilon) = \epsilon^T Q Q^T \epsilon = \|\epsilon\|^2 \tag{5.40}$$

and from eqs. (5.35), (5.39), and (5.40) the LS problem is:

$$\|\epsilon\|^2 = \|Q^T\omega - Q^T W_b\delta_b\|^2 = \| \begin{bmatrix} Q_1^2 \\ Q_2^2 \end{bmatrix} \omega - \begin{bmatrix} R_1 \\ 0 \end{bmatrix} \delta_b\|^2 \tag{5.41}$$

By defining:

$$\begin{bmatrix} p_1 \\ p_2 \end{bmatrix} = \begin{bmatrix} Q_1^T \\ Q_2^T \end{bmatrix} \omega \tag{5.42}$$

equation (5.41) is simplified:

$$\|\epsilon\|^2 = \| \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} - \begin{bmatrix} R_1 \\ 0 \end{bmatrix} \delta_b \|^2 = \|p_2\|^2 + \|p_1 - R_1\delta_b\|^2 \tag{5.43}$$

From equations (5.36), (5.41), and (5.43):

$$u - \|p_2\|^2 \geq \|p_1 - R_1\delta_b\|^2 \tag{5.44}$$

By Schur complement, eq. (5.44) is reformulated to an SDP problem of:

$$\begin{aligned} &minimize_{(u,\delta_b)} && u \\ &subjectto && U_p(u,\delta_b) \geq 0 \end{aligned} \tag{5.45}$$

where:

$$U_p(u,\delta_b) = \begin{bmatrix} u - \|p_2\|^2 & (p_1 - R_1\delta_b)^T \\ (p_1 - R_1\delta_b) & 1 \end{bmatrix} \tag{5.46}$$

The size of matrix $U_p(u,\delta_b)$ is a reasonable size as compared to U_ω which makes for an efficient SDP solution. This formulation is expanded to entail the constraints D_{δ_b} from the previous section:

$$\begin{aligned} &minimize_{(u,\delta_b),\delta_{r2}} && u \\ &subjectto && F_p(u,\delta_b,\delta_{r2}) \geq 0 \end{aligned} \tag{5.47}$$

where:

$$F_p(u,\delta_b,\delta_{r2}) = \begin{bmatrix} U_p(u,\delta_b) & 0 \\ 0 & D_{\delta_b}(\delta_b,\delta_{r2}) \end{bmatrix} \tag{5.48}$$

5.5 Results

The tests were done at a 200 Hz sampling rate for a total of 30 s following the Fourier trajectory. Data collection yielded joint positions, joint velocities, and joint efforts

data. Offline data processing in Matlab is used to filter the velocities and torques backward and forwards with the filtfilt function such that it does not produce a delay on the system. An 8th order Butterworth filter with a cutoff frequency of 3.5 Hz was used. The filtered velocities were derived to achieve acceleration data on each joint.

Table 5.1 shows the resulting condition number of the regressor matrix W_b given data from the simulated trajectory and tested trajectory of the hardware. LS Condition Num denotes the condition number for the tested trajectory for an ordinary least squares formulation. WLS Condition Num means the condition number for the tested trajectory for a weighted least squares by multiplication of A in equation (5.25). Note that the SDP does not show a condition number as it is not a regression method. All simulated and collected data is multiplied by the P weighting matrix in equation (5.26).

It is apparent the WLS data gives a larger condition number than the LS and SDP data which results in larger residual errors. The base frequency range of 0.12 and 0.16 gives the smallest condition number for all solutions and also the smallest residual errors. Choosing the LS or SDP solution for a base frequency of 0.13 gives the best results of model prediction.

The results for SDP and LS are similar which means the LS solution is physically feasible. The Cholesky function (function that would return a boolean depending on positive/negative eigenvalues) in Matlab was used to double check the feasibility of the mass inertia matrix M throughout the workspace of the robot. On a different set of tests, the SDP once returned a large difference in residual errors as compared to the LS solution. Here, the Cholesky function indicated non positive eigenvalues in the mass matrix indicating a physically non-feasible solution.

Figure 5.2a shows the difference between measured torques and computed torques

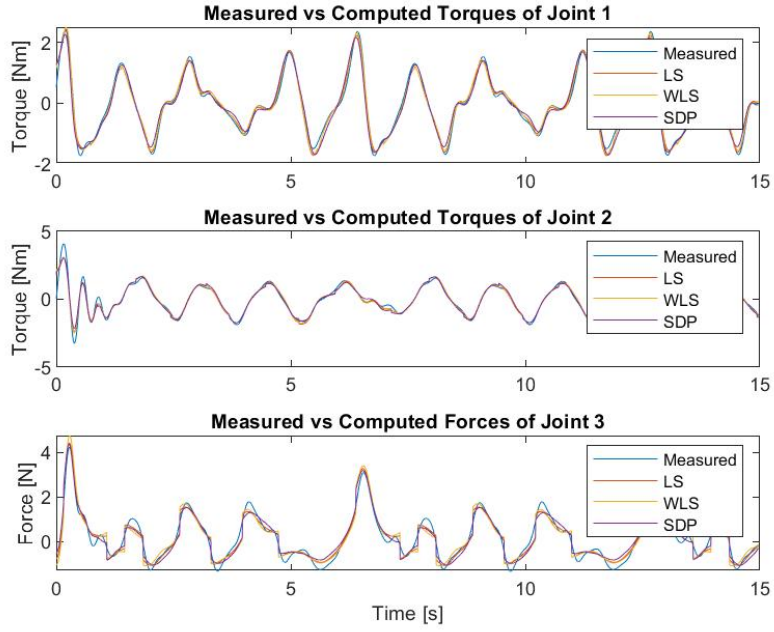
Table 5.1: Fourier Trajectory Condition Number and Residual Errors Results

Fourier Trajectories	1	2	3	4	5	6
w [Hz]	0.1	0.12	0.13	0.14	0.16	0.2
Optimization Condition Num	145.92	136.97	128.74	129.61	111.19	110.87
LS Condition Num	236.90	240.00	159.66	179.50	178.94	158.54
WLS Condition Num	1198.50	1024.80	297.98	606.24	389.43	349.75
LS Residual Errors	1440	1161.8	1034	1252.5	1181.1	1492.3
WLS Residual Errors	1917.5	1466.5	1208	2146.5	1311.4	1792.4
SDP Residual Errors	1434.0	1154.6	1387	1243.6	1169.4	1485.3

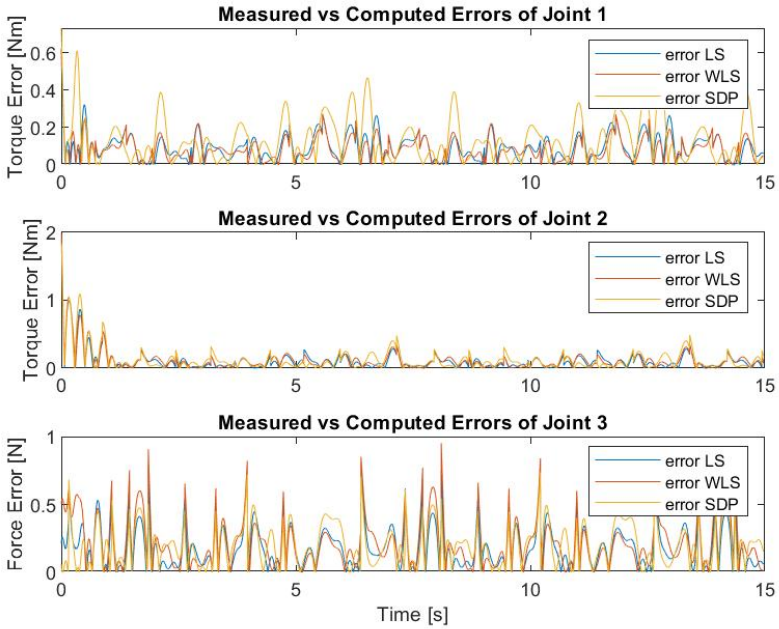
obtained from parameter identification. From Figure 5.2b, the maximum error on each joint is approximated to be 0.4 Nm, 0.25 Nm, 0.75 N (15%, 15%, and 40% error from the maximum measured torque) for joint 1, joint 2, and joints 3 respectively. Errors of these joint torque are due to unmodeled dynamics such as stick-slip friction, cable-tension in the transmission, and incorrect kinematics. Figure 5.3a shows the measured torques vs. computed torques on a test trajectory different from the identification trajectory. Results show that the model generalizes to another given trajectory and does not overfit the identification trajectory. An increase in prediction errors on all three joints is observed.

5.6 Friction Modeling and Identification for Control

40% force estimation error suggests incorrect friction modeling of the third joint. To study this effect, there are two categories of friction, dynamic or static. The Dahl model is a dynamic friction model where hysteresis is accounted for as an internal first-order variable z(t) [49]. Since it seems too complicated, we consider a simpler Stribeck friction model [50], which is a modified static Viscous and Coulomb friction model with large friction forces.

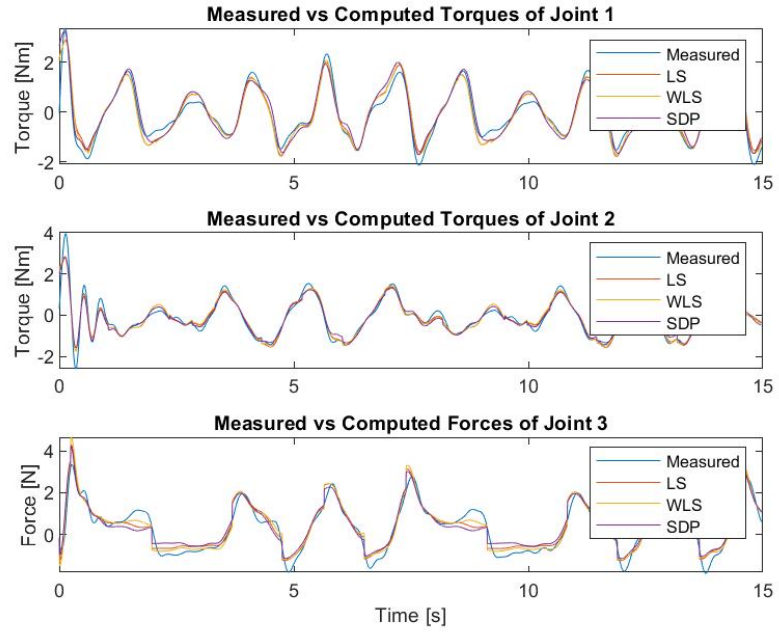


(a)

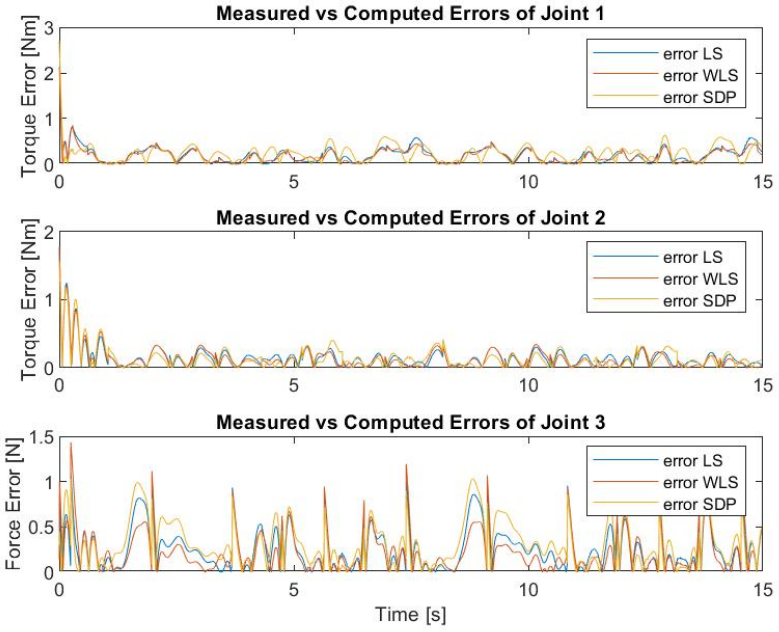


(b)

Figure 5.2: (a) Measured vs. predicted torque of the first three joints tested on the same identifying trajectory (b) Absolute errors of the computed trajectory vs. the measured trajectory



(a)



(b)

Figure 5.3: (a) Measured vs. predicted torque of the first three joints tested on a test trajectory (b) Absolute errors of the computed trajectory vs. the measured trajectory

After obtaining the base parameters δ_b from section (5.1), we compute the resulting friction torque using equation:

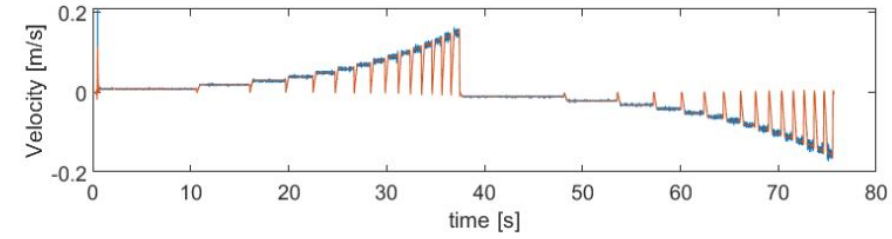
$$\tau_{f,test} = \tau_{test} - (W_b\delta_b - \tau_f) \tag{5.49}$$

Computing τ_f (eq. 3.10) separately is viable because friction parameters are absolutely identifiable. Figure 5.4 shows the constant velocity trajectory for the friction test of the third joint where $q_1 = 0$ and $q_2 = 0$. Figure 5.5 shows the friction forces vs. velocity where there is a large hysteresis. During constant velocity, the friction is also not constant which suggest a change in friction based on the joint position. Due to this, the curve fitting of the Stribeck function failed to execute.

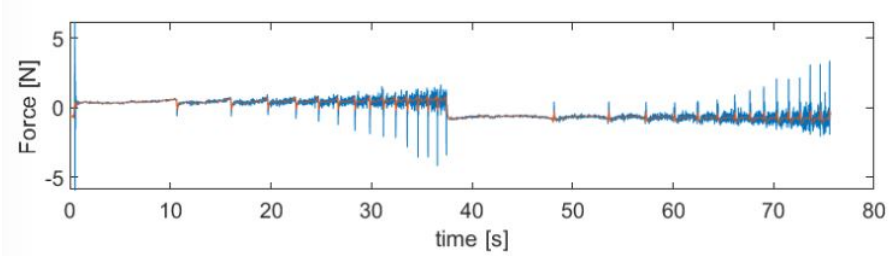
5.6.1 Robust Stick-slip Friction Compensation

Because the Stribeck friction model failed, we explore other causes of the friction error. An impedance control applied in chapter 7 resulted no motion of the prismatic joint when commanded a desired Cartesian position. This suggests that the Viscous and Coulomb friction does not adequately model static stick-slip friction when the joint is static.

Therefore, we measure the maximum stick-slip friction force to apply robust stick-slip friction compensation as in [51]. Figure 5.6 shows the force vs. velocity curve during a test of applied ramp force from 0-2 N in the positive direction and negative direction. The maximum force in which the joint begins to move is recorded and used for friction compensation in Chapter 7. Here the chosen static friction compensations is $Fs_+ = 1$ and $Fs_- = -1$. Figure 5.7 shows an example of the robust friction compensation force vs. velocity of the model.



(a)



(b)

Figure 5.4: Velocities (a) and Torques (b) of the third joint while $q_1 = 0$ and $q_2 = 0$. Data is friction torques from (eq. 5.49). The orange line indicates data processed with the Butterworth filter.

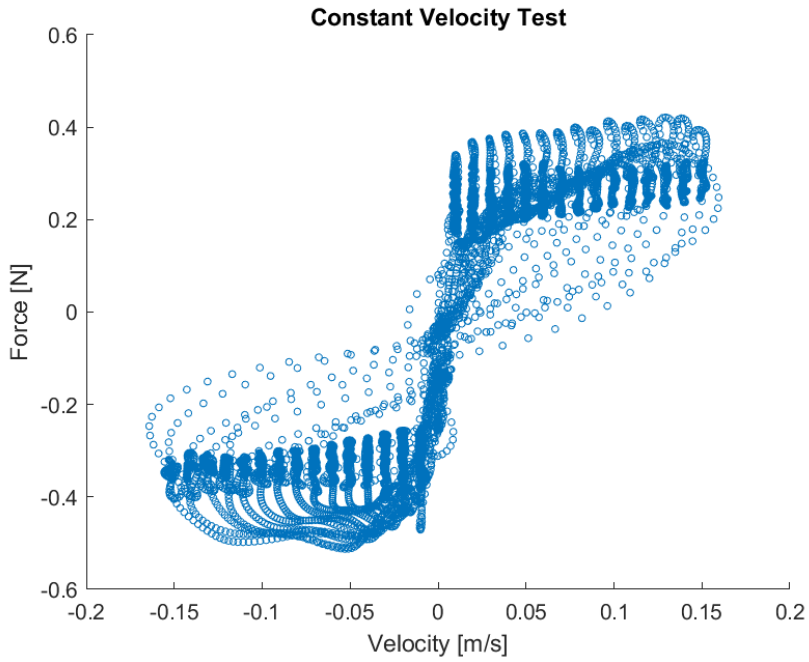


Figure 5.5: Constant Velocity Test of the third joint: Force vs Velocity

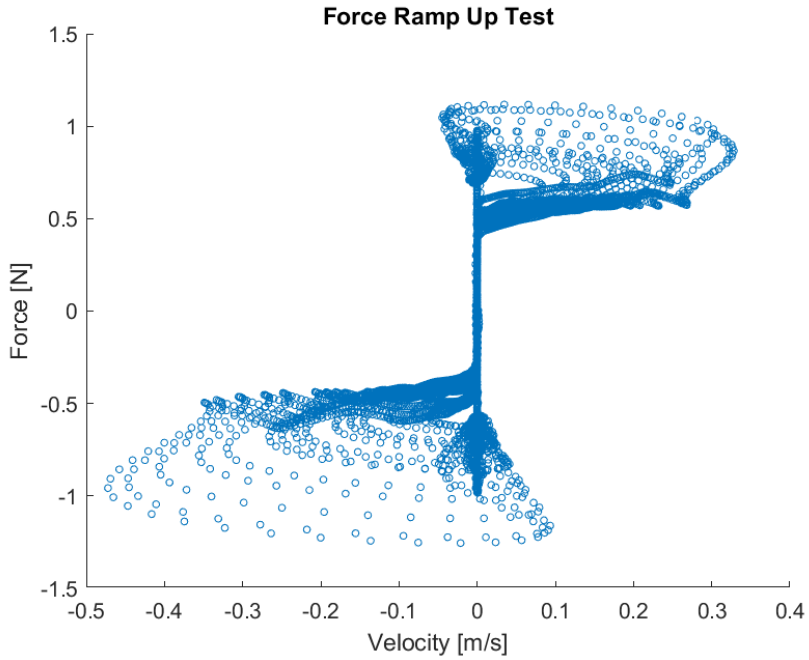


Figure 5.6: Force Ramp Up Test of the third joint: Force vs Velocity

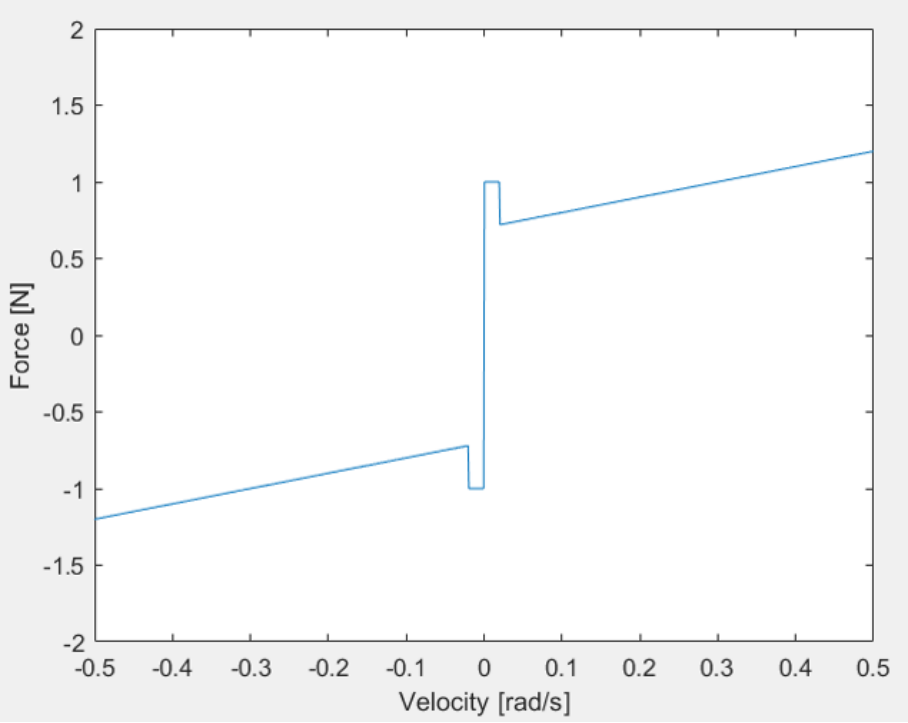


Figure 5.7: Robust stick-slip friction compensation example

Chapter 6

Cooperative Object Manipulation in ROS Simulation Environment

This chapter develops Robot Operating Software (ROS) [52] based controllers tested in simulation environments. This ROS framework streamlines controller software development because it will communicate with the ROS interface that uses CISST-SAW libraries to control the Patient Side Manipulator (PSM) [38]. ROS is a structured communication layer above a heterogeneous computation cluster that standardizes communication between threads; therefore simplifies software modification and development [53]. The controllers communicate with RViz and Gazebo simulation environments to test the cooperative kinematics, dynamics, and inverse kinematics of the PSM. A simulation environment avoids real-world nuances of the mechanical and electrical system.

Table 6.1: Remote Center of Motions for PSM Hardware and Simulations

RCM	Position [mm]			RMS Error [mm]		
	x	y	z	x	y	z
PSM hardware	-1.49	-516.11	2.19	0.62	0.62	0.31
PSM simulation	0.56	-518.60	0.93	0.85	1.91	1.08

6.1 Computer Aided Design (CAD) Modeling

A Solidworks PSM model is modified from a previous design developed at Automation and Interventional Medicine (AIM) Robotics Research Laboratory at Worcester Polytechnic Institute. Axis to axis link lengths, remote center of motion (RCM), and end-effector tool tip position are to be as accurate as possible by combining kinematic data from the dVRK technical document and the motion capture measurements.

A detailed CAD model is developed as Solidworks uses mesh volume and material density to estimate mass and inertia values. The base parameters (lumped parameters) from the parameter identification are not used for this simulation because the model requires values of individual link parameters (standard parameters).

Figure 6.1 shows the resulting model in Solidworks while Figure 6.2 shows the remote center of motion (RCM) accuracy and precision of simulation vs. hardware. Table 6.1 concludes that the simulation RCM location is only around 1-2mm different than the hardware. The root-mean-square (RMS) error shows the movement of the estimated RCM along the full workspace of joint q_1 and q_2 where zero RMS is the ideal RCM. Here the simulation shows higher RMS error values than the hardware. It is due to the vibrations in the double four-bar linkage and dynamic solutions solver in Gazebo.

Figure 6.1: Patient Side Manipulator Solidworks Model

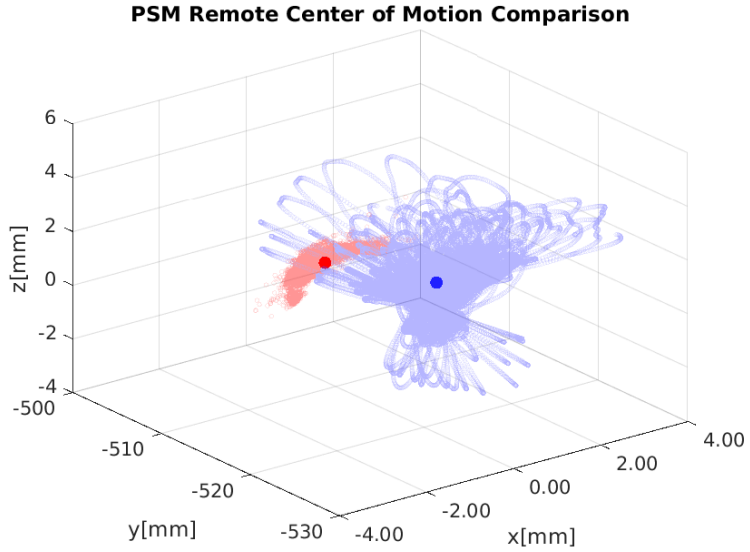


Figure 6.2: Estimated b_{post} hardware (red) and gazebo (blue) remote center of motions (RCM). Transparent markers are actual RCM points at a certain joint angle using b_{tip}.

6.1.1 Exporting to ROS Simulation Models

The Solidworks add-on, Solidworks to URDF exporter [54] developed by the ROS community, was used to assign coordinate frames of each link, export the kinematic data into a URDF file, and export the visual STL mesh files into one ROS package. URDF is an acronym for Universal Robot Description File developed by ROS as a standard to describe the kinematics of serial chain robots. It is an Extensible Markup Language (XML) which has tags that determine the relative parent-child link position, mesh coordinate frame position, joint type/position, the center of mass position, etc.

This URDF is then visualized in RViz simulation tool integrated with ROS. The simulation launches with inherent rosnodes, joint_state_publisher and robot_state_publisher, that allows control of each joint angle with a visual toolbar or rostopics that is programmed through python or C++. To simulate the cooperative environment, two (three) PSMs were simulated with different position and rotations in a given world. Figure 6.6 shows the RViz simulation.

6.2 Gazebo

Gazebo is a dynamic simulator wrapper for ROS based on the Bullet dynamic engine [55]. It allows dynamic simulations of robot manipulators communicating through ROS. The gazebo simulations used parameters from Solidworks because parameter identification of the real hardware resulted in base parameters for the simplified kinematic model. An extra step of mapping it into standard parameters plus adding the similar kinematic model of the PSM would require further verification.

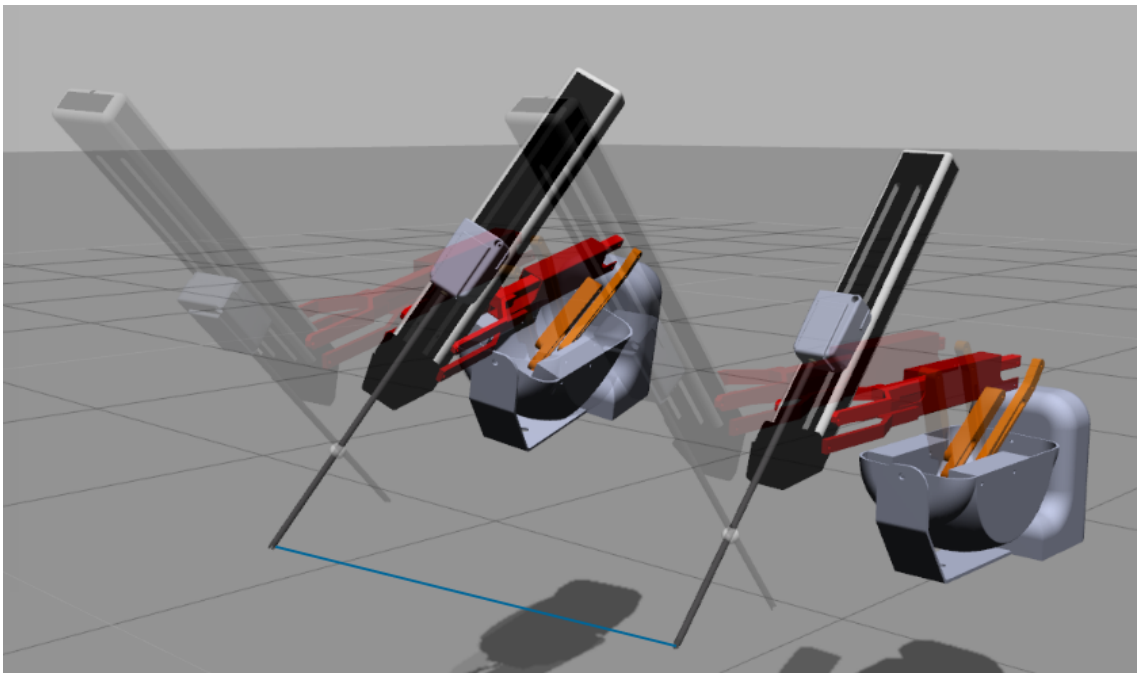


Figure 6.3: Cooperative manipulation control simulation in Gazebo with force tracking on object (blue line). Left: Slave PSM. Right: Master PSM

6.2.1 Massless Spring-Damper Object Environment

We develop a virtual object being manipulated by the PSMs by applying forces on the end effectors. These forces are generated by the difference between the length of the object vs the initial length of the object when the simulation started:

$$F_i = \frac{v_{o,i}}{\|v_{o,i}\|}[K_e(x_{e,i}(t) - x_o) + C_e(x_{e,i}(t) - x_{e,i}(t-1))] \tag{6.1}$$

where F_i is the force on link i, K_e the stiffness of the environment/object, C_e the damping of the environment/object, $v_{o,i}$ the vector representing the object and its norm $\|\|$, $x_{e,i}(t)$ the Cartesian position of link i at time t, $x_{e,i}(t-1)$ the Cartesian position of link i at time t-1, and x_o the initial position of link i at the start of the simulation. A rosnode is created that listens to the tool-tip positions, calculates the applied force (6.1), and applies the force on the two manipulators.

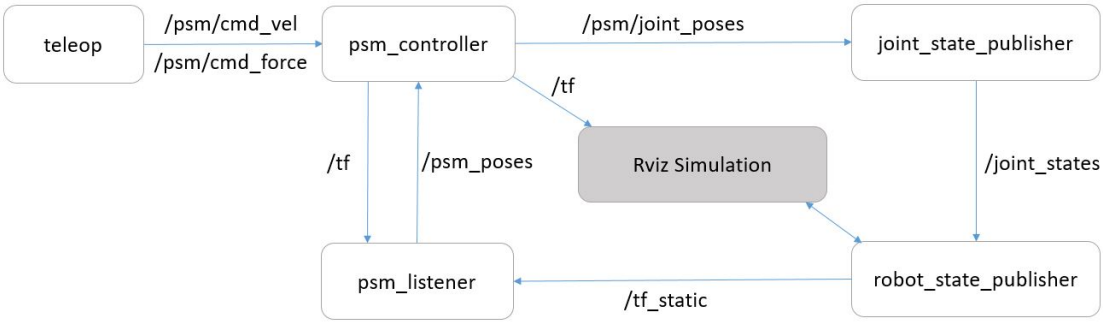


Figure 6.4: ROS controller architecture with RViz simulation. Squares represent rosnodes and arrows represent rostopics for communication.

6.3 ROS Controller Architecture

The controller was developed to work for both the RViz and Gazebo simulations. The ROS architecture of this controller interaction with RViz (Figure 6.4) and Gazebo (Figure 6.5) is shown below.

The ROS software environment consists of rosnodes depicted by squares. Each rosnode is an individual process that performs computation where each node communicates with another node using ROS TCP/IP like communication called rostopics (arrows). This type of architecture reduces code complexity by reducing exposed API to other nodes.

Node psm_controller represents the main controller which receives the Cartesian positions of the robot from the topic /psm_poses, calculates the desired Cartesian positions, computes inverse kinematics, and outputs the desired joint angles with the topic /psm/joint_poses.

The teleop node is an open source code ROS package [56] that takes keystroke data and in this work, is modified to output x,y, or z desired Cartesian increments and x, y, z Euler angles increments with topic /psm/cmd_vel. It also outputs an increment set force with the topic /psm/cmd_force. A data collection node called psm_listener receives link position and rotation data from the RViz or Gazebo sim-

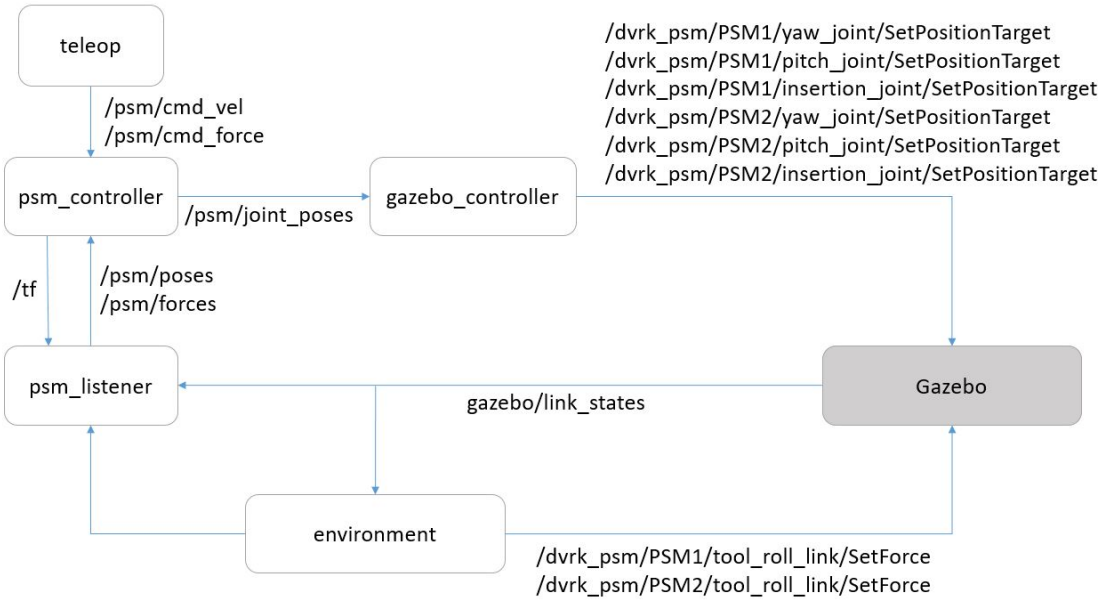


Figure 6.5: ROS controller architecture with Gazebo simulation. Squares represent rosnodes and arrows represent rostopics for communication.

ulation and publishes the data to psm_controller. In RViz, it reads link positions and rotations using the tf library. In Gazebo, the topic gazebo/link_states publishes link positions and rotations.

Other supplementary nodes that launch with the RViz simulation is the joint_state_publisher which is a GUI to change the robot joint angles via joint_states. Here this node is used as it was programmed to receive joint commands from other nodes. robot_state_publisher is a node that contains the robot kinematic data from the URDF, receives desired joint angle data from joint_state_publisher and publishes the correct position and rotation of each link of the robot to the RViz simulation.

Other nodes developed to communicate with Gazebo is the gazebo_controller node. This node is just complementary converting the rostopic /psm/joint_poses to rostopics that gazebo reads (top right of Figure 6.5).

The environment node is the virtual object for Gazebo that simulates an elastic massless spring-damper that reads link Cartesian positions via gazebo/link_states

and outputs forces to the robot links in the simulation via /dvrk_psm/PSM1/tool_roll_link/SetForce.

6.3.1 psm_controller: Cooperative controller with Force tracking

psm_controller is centralized controller that stores PSM manipulators as objects, P_i. For this environment, one PSM is delegated as a master while the rest are slaves. It receives remote center of motions (RCMs), tool-tip positions from psm_listener, and creates the vector describing the relative tool tip position to a master tool tip position, $v^w_{o,i}$, in world frame. PSM i initializes by defining $x_{e,i}$ (current cartesian), $x_{d,i}$ (desired cartesian), $R^m_{s,i}$ (rotation matrix describing the orientation to a master PSM), $p_{m,i}$ (3 x 1 vector describing the RCM position to master RCM position), and $v_{o,i}$ (3 x 1 vector describing the tool tip position to the master tool tip in the slave coordinate frame).

There are three basic commands of the controller, Cartesian move, orientation move, and force move that is set up as callbacks to listen for rostopics from the teleop node. As this is a ROS architecture, this controller can essentially read the same message from any hardware publishing to the same topic, /psm/cmd_vel or /psm/cmd_force. By defining the new desired position $x_{d,i}$, the controller then solves the inverse kinematics and publishes an output command /psm/joint_poses. It also publishes the position of the RCM, tool-tip, and $v_{o,i}$ as a TF transform to RViz to visually check if the kinematic computation is correct.

For the master, move in Cartesian is straightforward. For the slave, the movement is synchronized with the master's coordinate frame and is defined by:

$$x^i_{d,i} = (R^m_{s,i})^{-1}\Delta x + x^i_{d,i} \tag{6.2}$$

where the desired position $x_{d,i}$ is incrementally increased/decreased by the variable, Δx from rostopic /psm/cmd_vel. Note, all notation here is in manipulator frame. Additionally, the orientation move is:

$$x^i_{d,i} = (R_{eul,i} - I)(-c^i_i) \tag{6.3}$$

where I is a 3 x 3 identity matrix, and R_{eul} is an Euler rotation matrix [57] that describes the change in rotation of the object created by the change of Euler angles α, β, and γ from the /psm/cmd_vel/orientation message. c^i_i is the vector from tool tip pointing at the centroid of the manipulated object. Last, the force move is:

$$x^i_{d,i} = \Delta x \frac{c^i_i}{\|c^i_i\|} + x^i_{d,i} \tag{6.4}$$

where Δx is the increment variable from /psm/cmd_force.

6.3.2 Controller tested in RViz

Figure 6.6 shows the simulation environment for 3 PSMs in a certain position/rotation that is realistic for a cooperative manipulation task. Also, this Figure shows the motion of the PSMs to increase the internal tensile force of the object. Additionally, Figure 6.7 shows the motion of the PSMs to reorient the object.

6.4 Position Controller Force Tracking Results

Figure 6.3 illustrates the gazebo simulation for cooperative manipulation control with 2 PSMs moving 10mm in the y-direction while tracking a 5 N tensile force on the object (blue). During this test, the chosen stiffness for equation (6.1) is $K_e = 1000$ [N/m]. The controller is running at 400 Hz. Figure 6.8 shows quantitative data of

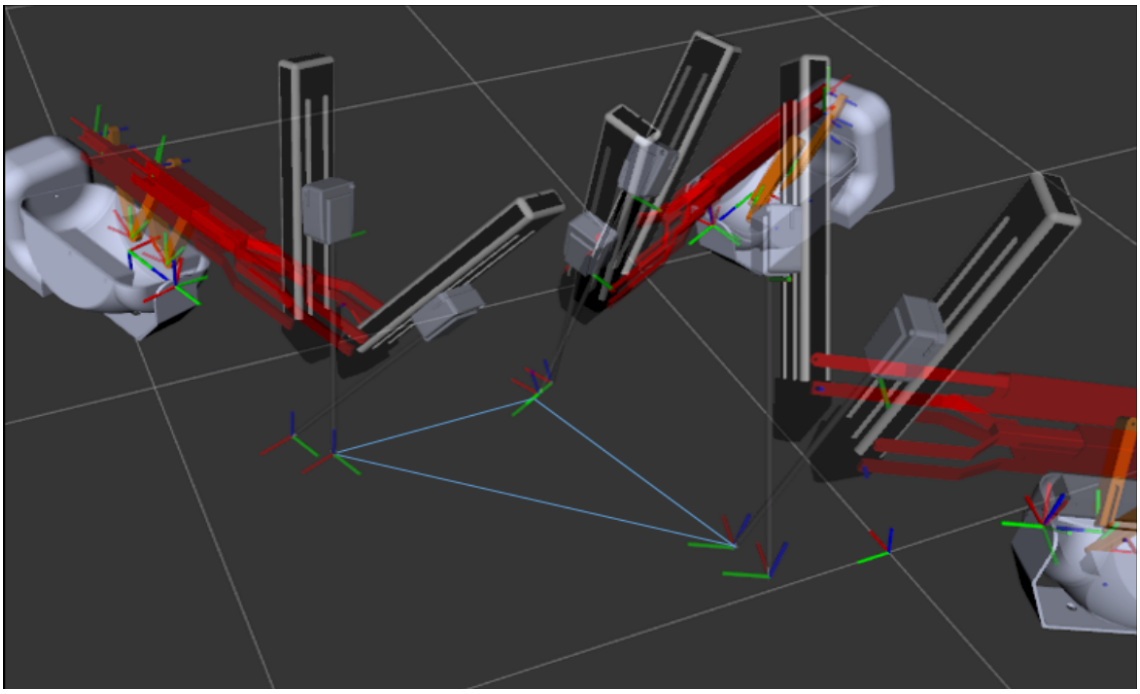


Figure 6.6: Cooperative manipulation control simulation in RViz with 3 PSMs. This shows the desired displacement during a tension force increase on the object.

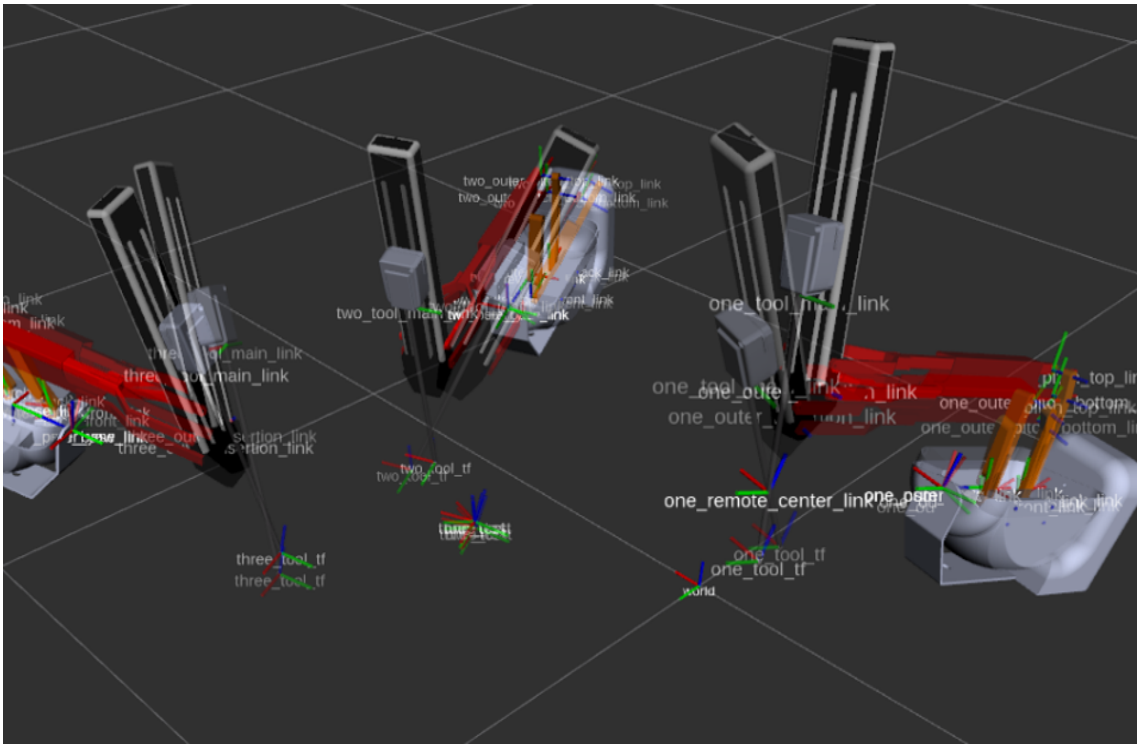


Figure 6.7: Cooperative manipulation control simulation in RViz with 3 PSMs. This shows the desired reorientation motion of the object.

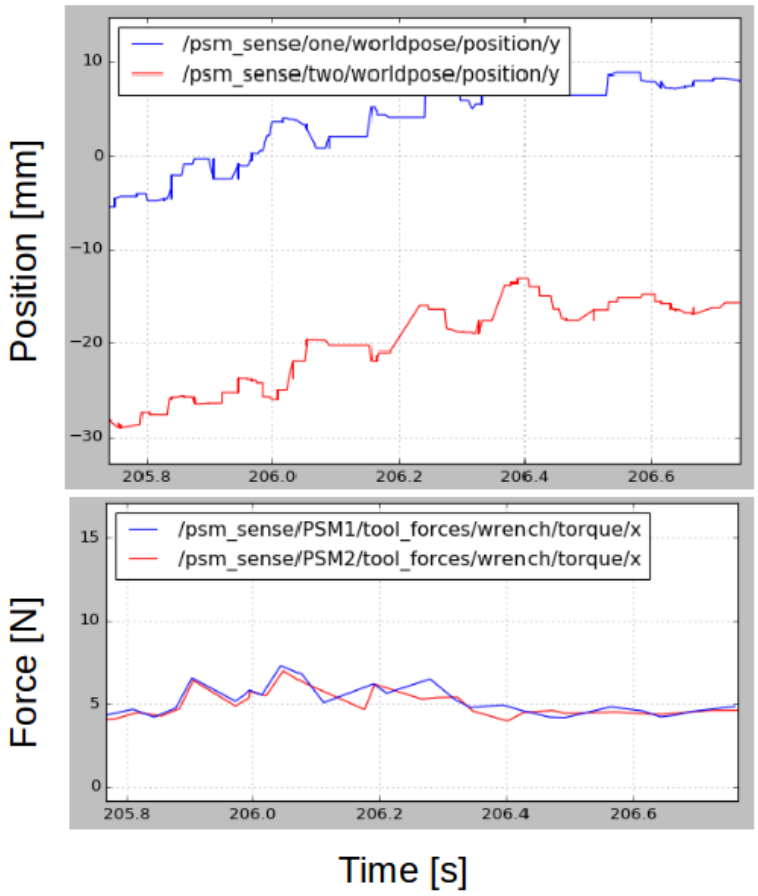


Figure 6.8: Gazebo controller tracking 5N force on object with 10mm y-direction movement

the y-position of the tooltip (top) and force on the object (bottom). Both PSMs move in the world frame y-direction in coordination and results in the satisfactory tracking of the Force.

Chapter 7

Cooperative Object Manipulation with Force Control

This chapter presents the implementation of feedback linearization of the model from chapter 5, a computed torque controller to verify the model, an impedance controller for the Patient Side Manipulator (PSM), and the position based cooperative controller with force tracking from chapter 6. Finally, the aforementioned cooperative controller is modified to use an impedance controller for the slave PSM.

7.1 Manipulator Dynamics Linearization

The manipulator dynamics in Equation (4.10) is eliminated with a feedforward model-based control input [34]:

$$\tau_m = u = \hat{M}(q)y + N(q, \dot{q}) + \hat{\tau}_f(q, \dot{q}); \tag{7.1}$$

where

$$N(q, \dot{q}) = \hat{C}(q, \dot{q})\dot{q} + \hat{g}(q) \tag{7.2}$$

$\hat{M}$ is the estimated mass inertia matrix of the robot and N is the estimated Coriolis matrix, $\hat{C}$, plus estimated gravity matrix, $\hat{G}$. N is used for simplification in controller software. The generalized coordinates q, $\dot{q}$, and $\ddot{q}$ represent joint positions, joint velocities, and joint accelerations respectively. For ideal estimates of the dynamics, this control input results in the linearized dynamics of the manipulator described as:

$$M(q)\ddot{q} = \hat{M}(q)y \tag{7.3}$$

and:

$$\ddot{q} = y \tag{7.4}$$

is the double integrator system. Choosing the desired joint acceleration, y, defines different types of force control behavior.

7.2 Computed Torque Controller

Defining y as:

$$y = K_p(q_d - q) + K_d(\dot{q_d} - \dot{q}) \tag{7.5}$$

results in the computed torque controller where output torque on each joint is based on the joint position and joint velocity error with linearized dynamics. K_p is the positive definite n x n diagonal proportional gain matrix and K_d is the positive definite n x n diagonal derivative gain matrix. Where n is the degree-of-freedom (DOF) of the manipulator.

7.3 Impedance Controller

$J^T h_e$ is added to equation (7.1) to compensate the coupled effects of the interaction force h_e. This variable is the estimated interaction to a known stiffness environment. If force measurement is available, then h_e is equal to the sensor data. The modified control input, u, is:

$$u = M(q)y + N(q, \dot{q}) + \tau_f + J^T h_e \tag{7.6}$$

The chosen desired acceleration, y, is

$$y = J_a^{-1} M_d^{-1} (M_d \ddot{x_d} + K_p(x_d - x_e) + K_d(\dot{x_d} - \dot{x_e}) - M_d \dot{J}_a \dot{q} - h_a) \tag{7.7}$$

where J is the end-effector geometric Jacobian, J_a the end-effector analytical Jacobian, M_d the diagonal desired inertia matrix, K_p the diagonal proportional gain matrix, K_d the diagonal derivative gain matrix, x_d desired position vector, x_e actual end effector position vector, and its derivatives. h_a is defined as:

$$h_a = T_A^G h_e \tag{7.8}$$

where T_A^G is the transformation matrix between the geometric Jacobian and analytical Jacobian. Substituting equations (7.6) and (7.7) into equation (4.10) results in a decoupled force interaction closed loop system with dynamics:

$$M_d(\ddot{x}_d - \ddot{x}_e) + K_d(\dot{x}_d - \dot{x}_e) + K_p(x_d - x_e) = h_a \tag{7.9}$$

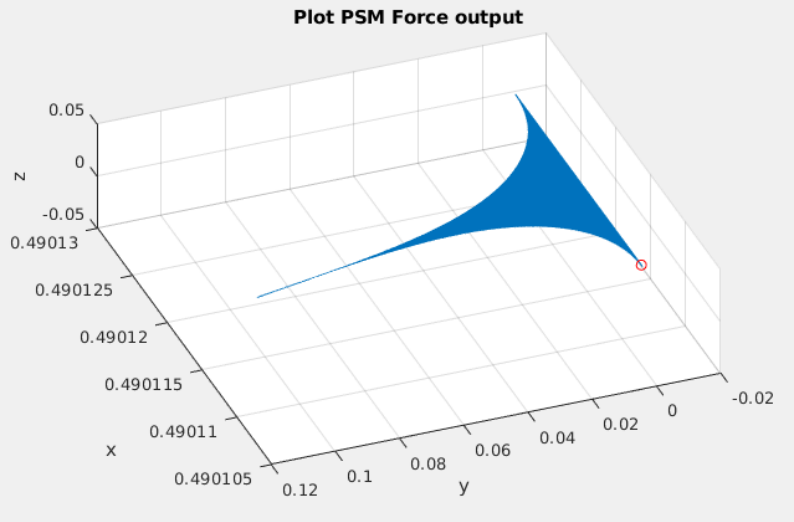


Figure 7.1: Impedance Controller on Robot with simulated compliant environment in the x-direction.

7.4 Stability Analysis

With the identified robot base parameters, the impedance controller interaction with a one DOF massless spring object in the x-direction is simulated in Matlab ODE. Figure 7.1 shows stable tracking of desired positions in y and z free motions and minuscule oscillating stability in the x-direction tracking a given force.

7.5 Force Controller Architecture

Figure 7.2 illustrates the ROS communication framework between the developed controller and dVRK hardware with sawIntuitiveResearchKit_dvrk CISST-ROS interface. PsmForceControl_node controller takes position commands from teleop and publishing outputs as joint efforts. The PsmForceControl_node is compiled in C++ with Eigen library for crucial matrix computations. It takes the joint positions, velocities, efforts, and Cartesian positions data from the CISST-SAW rosnode bridge. It also reads the force magnitude from the load cell. With this data, it is able to

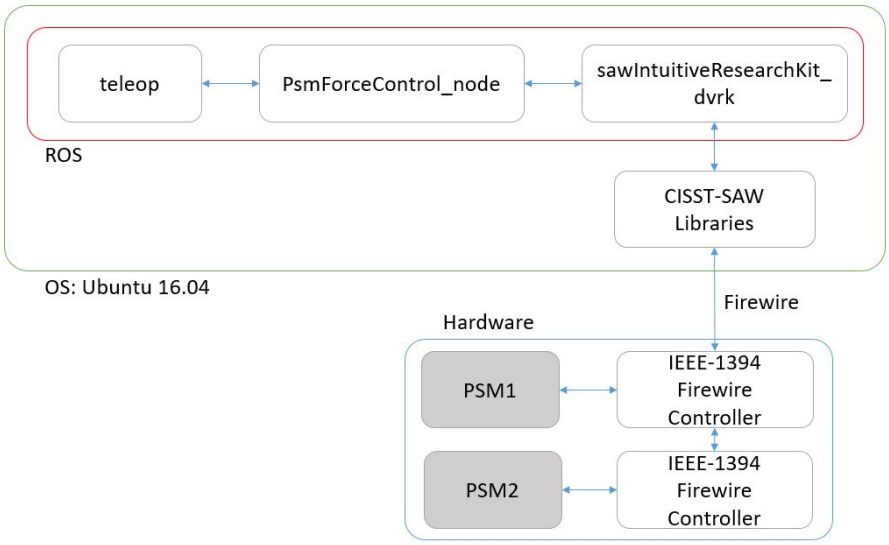


Figure 7.2: Software Architecture of ROS controllers communicating with PSM/MTM hardware

compute friction compensation $\hat{\tau}_f(q, \dot{q})$, nonlinear dynamic compensation $N(q, \dot{q})$, the mass inertia matrix, $\hat{M}(q)$, environment force h_e, and the end effector Jacobian $J(q)$ all needed for force control. We filter the joint velocities with a 10 point moving average filter. The node outputs the resulting joint torques to the robot with topic /dvrk/$psm_name/SetEffortJoint. The load cell measurement used an instrumentation amplifier on Arduino developed by Novoseltseva [58]. It used ROS rosserial_node package to convert serial messages into rostopics publishing analog-to-digital values at 500Hz.

7.5.1 Robust Stick-slip Friction compensation

We apply the algorithm for friction compensation:

```
//Friction Compensation
void PsmForceControl::CalcFr(const Eigen::VectorXd &q, const
    Eigen::VectorXd &qd)
```

```
{
  float x[3]; //Intermediate variable
  for (int i=0;i<3;i++)
    {
        if (abs(qd(i)) < deadband(i)) {
            x[i] = 0; // Set velocity to zero because of deadband
        }
        else {
            if (i==2 & v_int(i) > qd(i)) {
                x[i] = v_int(i); // Use interpolated velocity for
                    compensation
            }
            else {
                x[i] = qd(i);  // Normal friction compensation
            }
        }
    }
    // Stick-slip friction parameters
    float Fs_pos = 1;
    float Fs_neg = -1;
    // Position error on joint 3
    float x_e = joint_des(2) - joint_act(2);
    if(name == "PSM1")
    {
        Fr(0) = x[0]*9.542E-2+1.090E-1/(exp(x[0]*-4.0E2)+1.0)+2.751E-1;
        Fr(1) = x[1]*1.633E-1+1.631E-1/(exp(x[1]*-4.0E2)+1.0)+1.937E-1;
        //Robust stick-slip friction compensation
```

```
    if(abs(qd(2)) < deadband(2)) {
        if (x_e > pos_deadband) {
            Fr(2) = Fs_pos; //Positive stick-slip friction
        }
        else if (x_e < -pos_deadband) {
            Fr(2) = Fs_neg; //Negative stick-slip friction
        }
        else {
            Fr(2) = 0;     //No friction compensation because of position
                deadband
        }
    }
    else {
         Fr(2) = x[2]*7.663E-1+1.069/(exp(x[2]*-4.0E2)+1.0)-3.845E-1;
             //Viscous and Coulomb friction compensation
    }
   }
}
```

where qd(2) is the velocity of the third joint, q_e is the position error of the third joint, and Fr(2) is the calculated friction compensation of the third joint. The velocity deadband (vel_deadband) is 0.01 rad/s, 0.01 rad/s, and 0.005 m/s for each joint respectively and the position deadband (pos_deadband) is 0.0003 rad for the third joint.

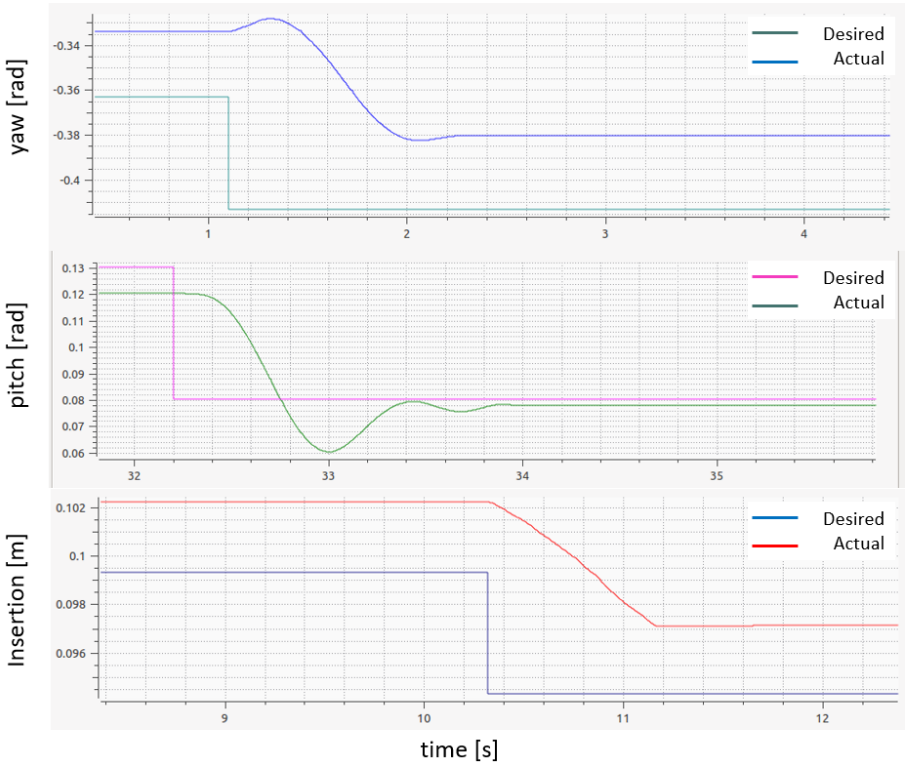


Figure 7.3: Computed torque controller joint motion with commanded increments of the yaw [rad], pitch [rad], and insertion [m] joints respectively. The two different lines of each plot indicate the desired joint position and the actual joint position.

7.6 Single Manipulator Force Controller Results

7.6.1 Computed Torque Controller

This computed torque controller is used to test whether the feedforward compensation from the estimated model (eq. 7.1) from chapter 5 is correct for the Patient Side Manipulator (PSM). Figure 7.3 shows results for computed torque controller of PSM1. Values of Kp and Kd for yaw, pitch, and insertion joint are chosen as Kp = 25, 20, and 75 and Kd = 10, 11, and 15 respectively. While moving at an incremental desired joint position of 0.05 rad (0.005 m for prismatic), the first two joints and third prismatic joints show a maximum steady-state error of 0.03 rad and 0.002 m. The settling time is around 1s which corresponds to the interpolator

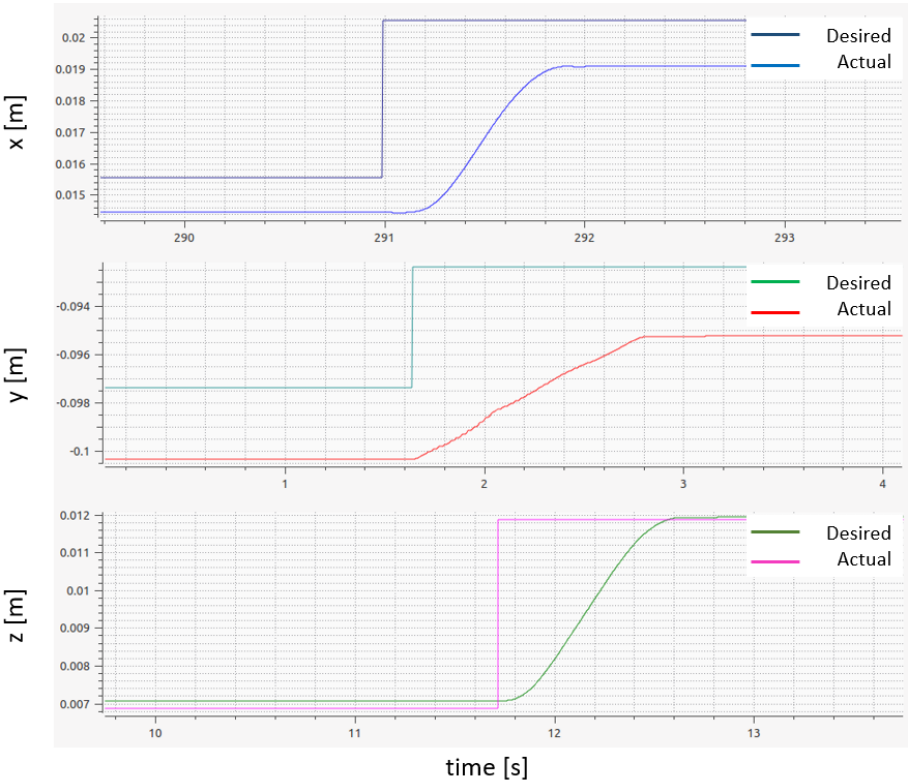


Figure 7.4: Impedance controller Cartesian motion with commanded increments of the x [m], y [m], and z [m] positions respectively. The two different lines of each plot indicate the desired joint position and the actual joint position.

reaching the final desired position in 1s. Comparing the default PID gains from the CISST-SAW libraries (200, 200, 600) versus the computed torque controller gains (25, 20, 75), suggests the feedforward model compensation results in satisfactory linearization.

7.6.2 Impedance Controller

The nonlinear feedforward compensation used for this controller is the same used for the computed torque controller. Early tests did not use robust stick-slip compensation, resulting in no motion of the third prismatic joint given an incremental Cartesian command. Because the problem was in joint space and not in Cartesian

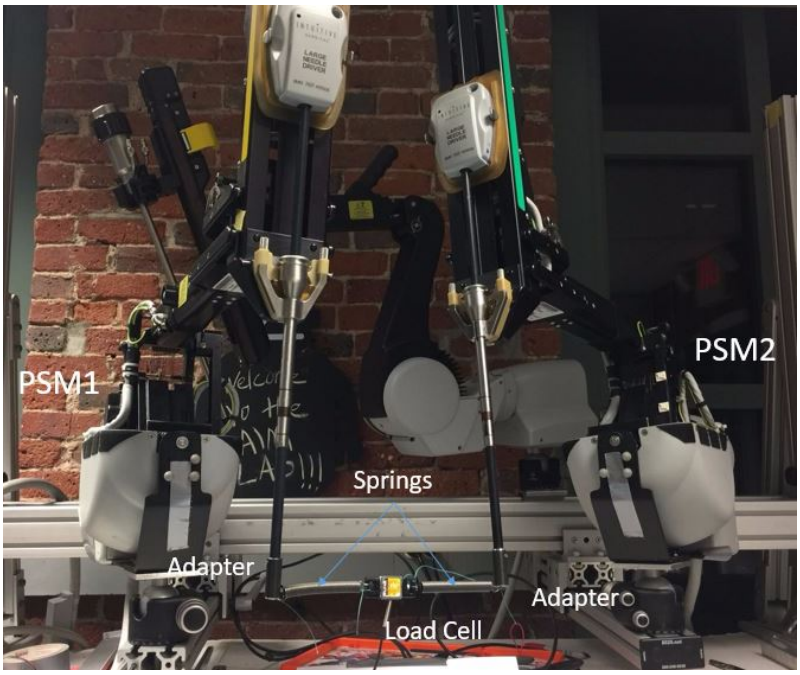
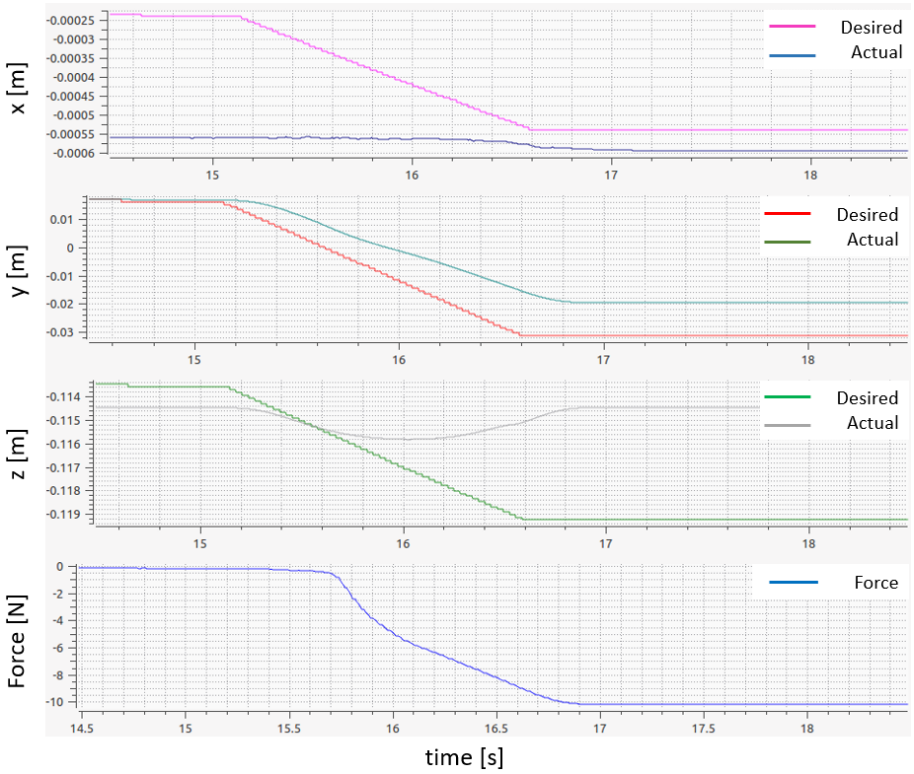


Figure 7.5: Test setup. A load cell is placed in between two springs mounted to an adapter on the end effectors of the PSM. The springs are interchangeable with strings.

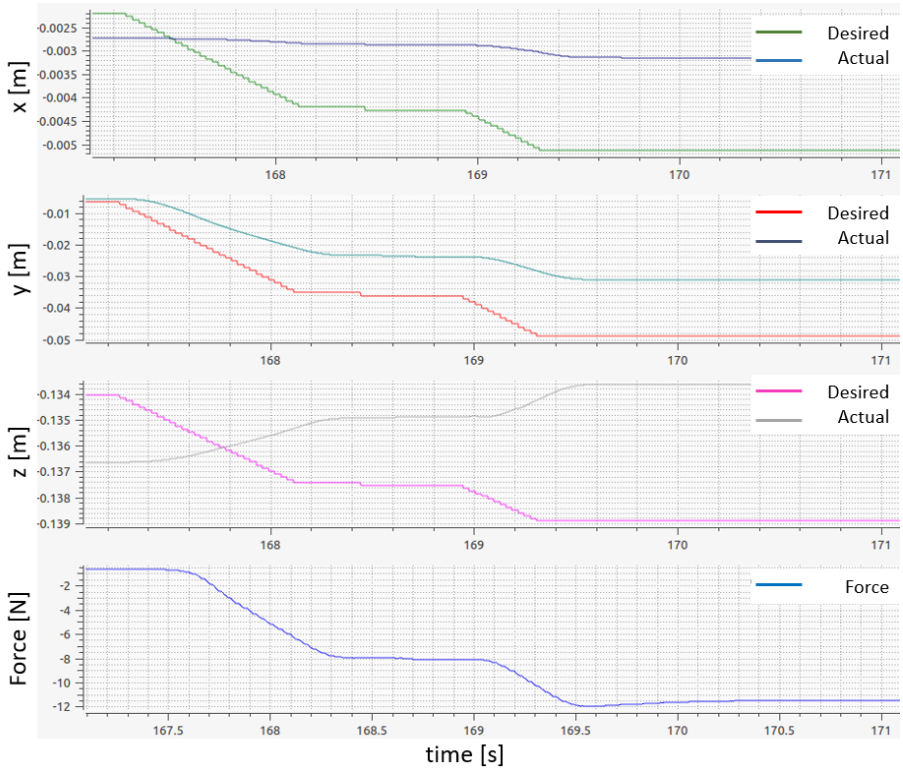
space, we anticipate errors in static friction modelling. Values for K_p andK_d corresponding to each XYZ axis respectively is chosen as $K_p = \text{diag}(15, 15, 15)$ and $K_d = \text{diag}(3, 3, 3)$. Low K_p values provide good manipulator compliance while still providing enough input to move the end-effector to the desired position. The resulting low K_p gain similar in all XYZ directions suggests the model compensation is accurate. The chosen desired inertia matrix, $M_d = \text{diag}(0.35, 0.36, 0.3)$, is the same as the estimated inertia matrix, $\hat{M}$, at joint positions $q_1 = q_2 = q_3 = 0$.

Figure 7.4 shows the end-effector motion with an impedance controller in Cartesian space. The steady state error of a 1sec 5mm incremental move in each direction is 1, 0, and 2 mm respectively. While not the best results for a robot, the trade-off between a compliant impedance control and tracking error is expected and is explained later to be satisfactory for cooperative control.

Figure 7.5 shows the setup for the PSM1 impedance controller interaction with



(a)



(b)

Figure 7.6: Impedance controller applying a force [N] to (a) Compliant spring (b) Stiff string.

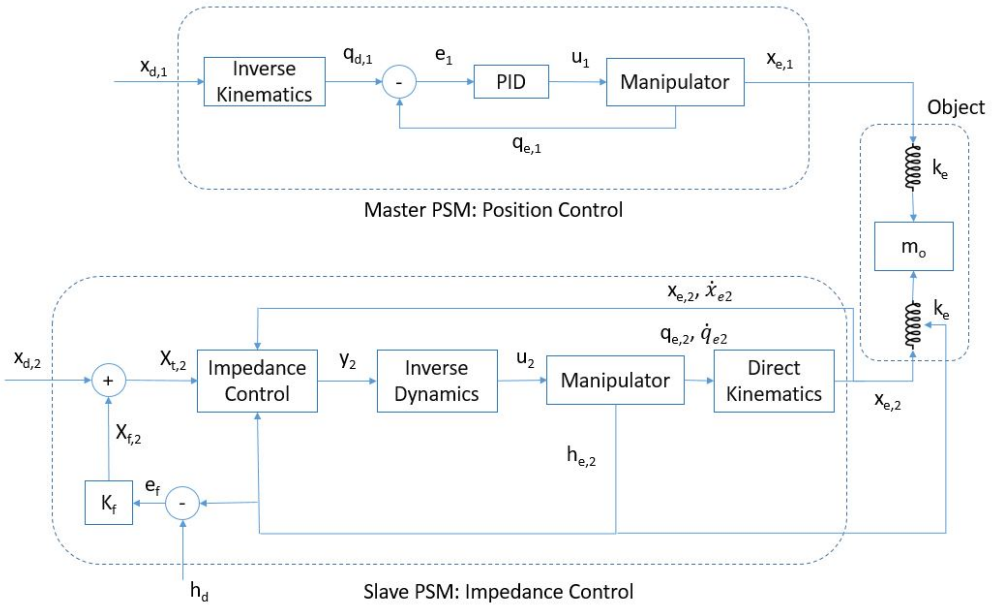


Figure 7.7: Cooperative force controller with impedance control diagram

an environment. PSM2 is held stiff with a Cartesian position controller where two springs with a stiffness of 2450 N/m (compliant environment) are attached to the end-effectors of PSM1 and PSM2. The test on a stiff environment is also conducted using a string. Figure 7.6 shows an applied tension to (a) springs and (b) strings. The first three top graphs are x, y, and z-axis desired end effector position, x_d [m], and actual end effector position, x_e [m]. The bottom graph is the force [N] on the object. The horizontal axis shows time [s] of the ROS system. The results verify that stable force interaction, in this case up to 12 N, is controllable for both compliant and stiff objects.

7.7 Cooperative Force Control

Figure 7.7 illustrates the control architecture for cooperative object manipulation. The master PSM utilizes the standard position Cartesian controller from CISST-SAW libraries and the Slave PSM utilizes the Impedance controller. The object has

Figure 7.8: Kinematic calibration of PSM1 and PSM2

a certain mass, m_o, and certain stiffness, k_e. Position control constrains the object with end-effector output, $x_{e,1} \in R^3$, while the Impedance control interacts with the object by force, $h_{e,2} \in R^3$, and end-effector position, $x_{e,2} \in R^3$. Ideally, feedback of $h_{e,2} \in R^3$ is data from the force sensor, where in this case, we use a load cell.

7.7.1 Kinematic Calibration

The Optitrack Motion capture along with least squares solution (section A.2) was used to calibrate the relative RCM position, $p_{12} \in R^3$, and the relative orientation, $T_2^1 \in SO^4$, of PSM1 and PSM2. Refer to Figures 7.8 and 7.9 for notations.

7.7.2 Cooperative Kinematics Revisited

Figure 7.9 shows the cooperative controller kinematics in world frame. In our case, O_2 is the master and O_1 is the slave where O_2 is our world frame origin. To have O_1

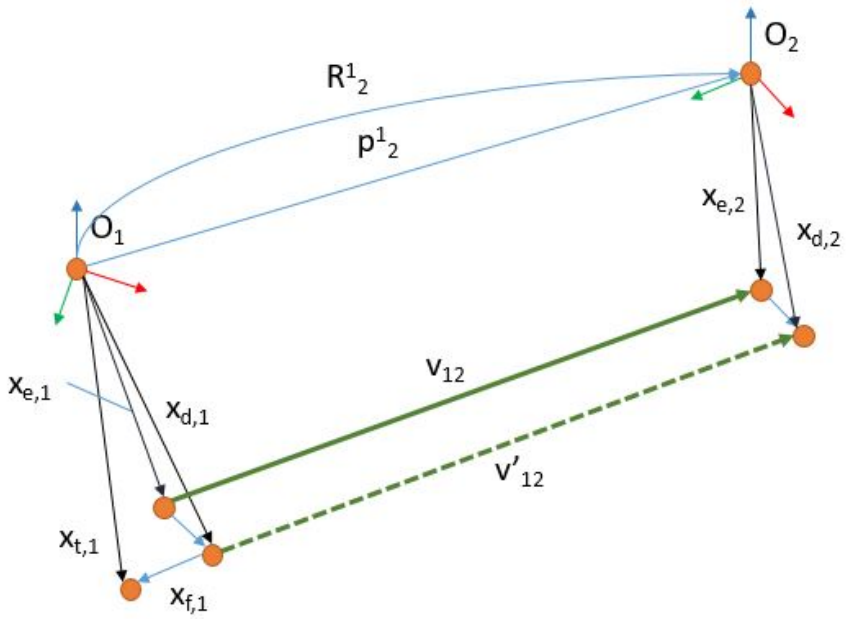


Figure 7.9: Kinematic illustration of the cooperative controller. Black arrows indicate vectors from origin, blue arrows indicate incremental vectors, and the green line is the object vector. O_1 and O_2 indicate the origin of each PSM (RCM point).

move in O_2 frame, refer to equation (6.2) for XYZ movement and equation (6.3) for object orientation movement. Note, the previous equations are in each manipulators frames and the equations presented in this section are all in world frame. The vector $v_{12} \in R^3$ is the vector from $x_{e,1}$ (PSM1 end-effector) to $x_{e,2}$ (PSM2 end-effector):

$$v_{12} = x_{e,2} - x_{e,1} \tag{7.10}$$

Therefore, vector v_{12} represents the manipulated object. In addition, we define a new variable $x_f \in R^1$. It allows the separation of the desired position based on motion $x_d \in R^3$ and the desired position based on force tracking, x_f. The total desired position, $x_t \in R^3$, is used as the reference for the impedance controller (eq. 7.7):

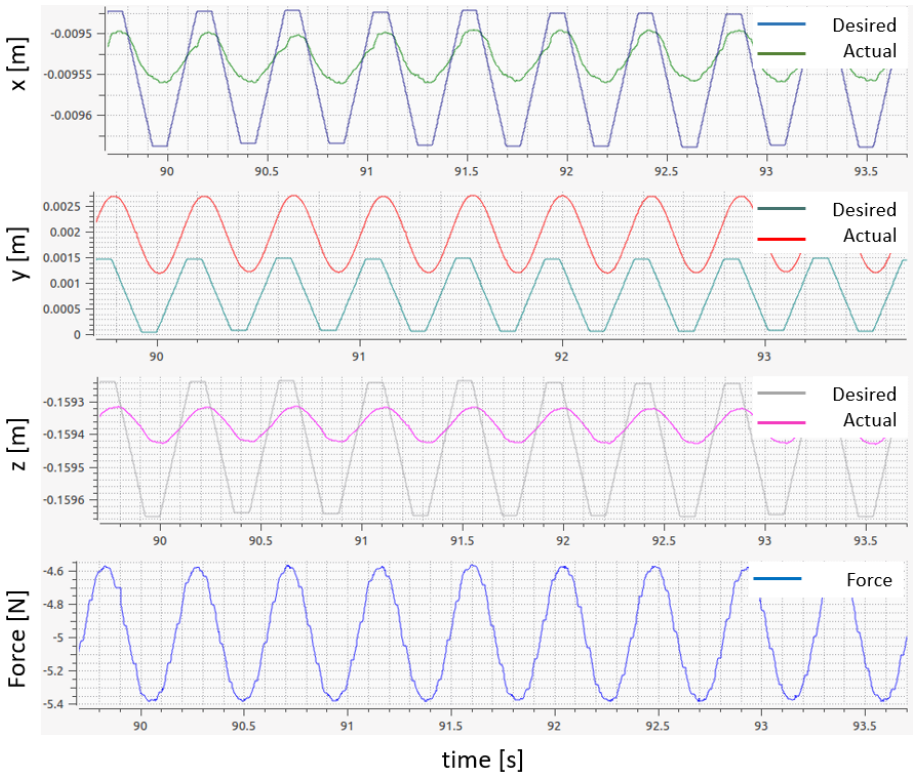
$$x_t = \frac{v_{12}}{\|v_{12}\|} x_f + x_d \tag{7.11}$$

With this reformulation, x_f is updated at time step t by a proportion gain, K_f, from the error of the actual force magnitude of the load cell, $h_e \in R^1$, and desired force, $h_d \in R^1$:

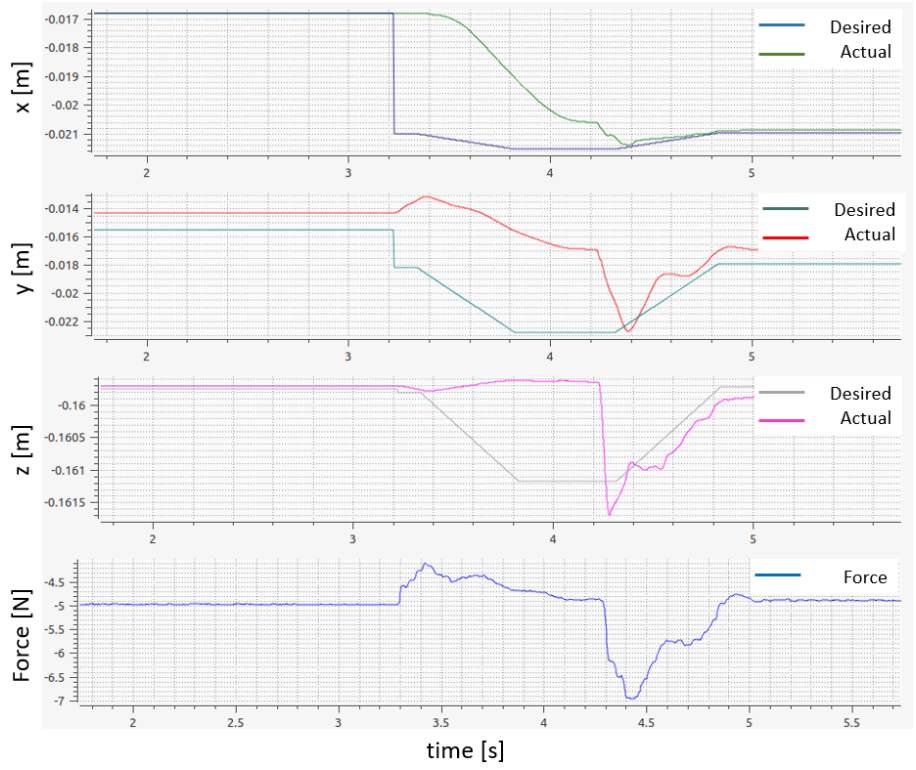
$$x_f(t) = x_f(t-1) + K_f(h_e - h_d) \tag{7.12}$$

7.7.3 Position Based Force Tracking

To test whether the kinematics are correct, both PSMs were controlled using the standard Cartesian position controller from the CISST-SAW libraries. Figure 7.5 shows this setup. PSM2 is the master and PSM1 is the slave moving to compensate the force on the load cell. We set the desired tensile force to 5 N. Figure 7.10 shows the results of this test with the first top three showing the XYZ axis desired position and actual position [m] and the bottom graph showing the force output of the load cell. The horizontal axis is the time [s] of the ROS system. Figure 7.10a shows that with a 0.2 N force deadband and a 0.0002 $x_{d,2}$ increment at 500 Hz, force tracking results in an unstable behavior of the controller. Changing the deadband to 0.4 N (Figure 7.10b) gives a stable result for tracking the force during a 5 cm move in the x-direction of the master. This result concludes that a position Cartesian force tracking is sensitive to environment stiffness and would require to adjust gains accordingly.



(a)

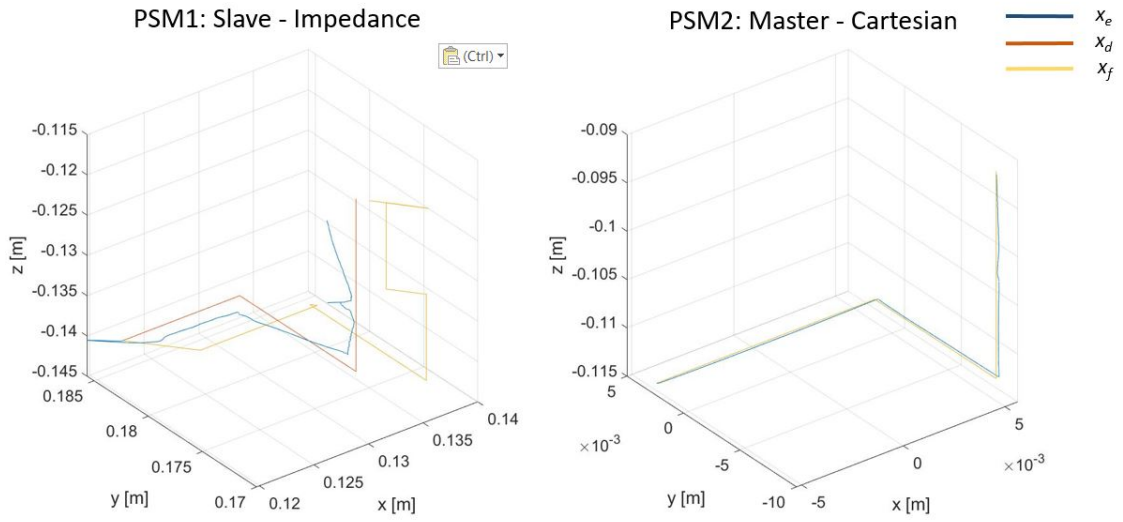


(b)

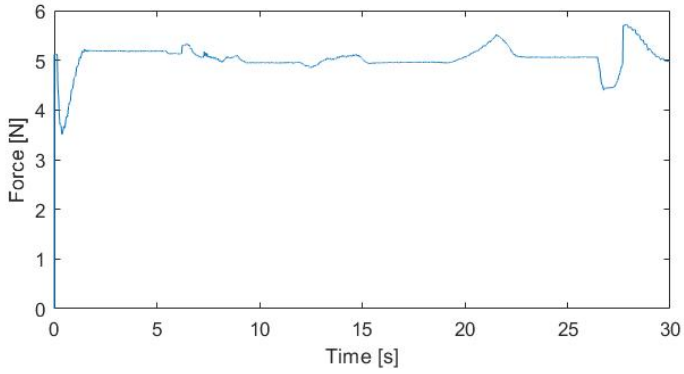
Figure 7.10: Position Cartesian force tracking results on a spring. (a) Unstable limit cycles (b) Stable with simultaneous motion

7.7.4 Cooperative Object Manipulation with Force Tracking

This test uses the same setup as in Figure 7.5. A load cell placed in between the end-effectors was used to measure the force magnitude. The relative end-effector positions were known from the kinematic calibration and forward kinematics. Figure 7.11 shows the results for cooperative impedance force tracking of PSM1 and PSM2 while simultaneously moving in synchronization with PSM2 master for a compliant spring. The blue line represents actual end-effector position, x_e, the red line represents the desired position, x_d, and the yellow line represents desired position force tracking, x_f. Because there is no damping gain, 1-2 N maximum force tracking errors occurred in the beginning to stabilize from 0 to 5 N. Once the controller tracked a 5 N force, the maximum force tracking error while simultaneously moving in Cartesian trajectory is 0.5 N. The cooperative force tracking on stiff suture string is also tested (Figure 7.12) resulting in a 0.5 N maximum force tracking error.

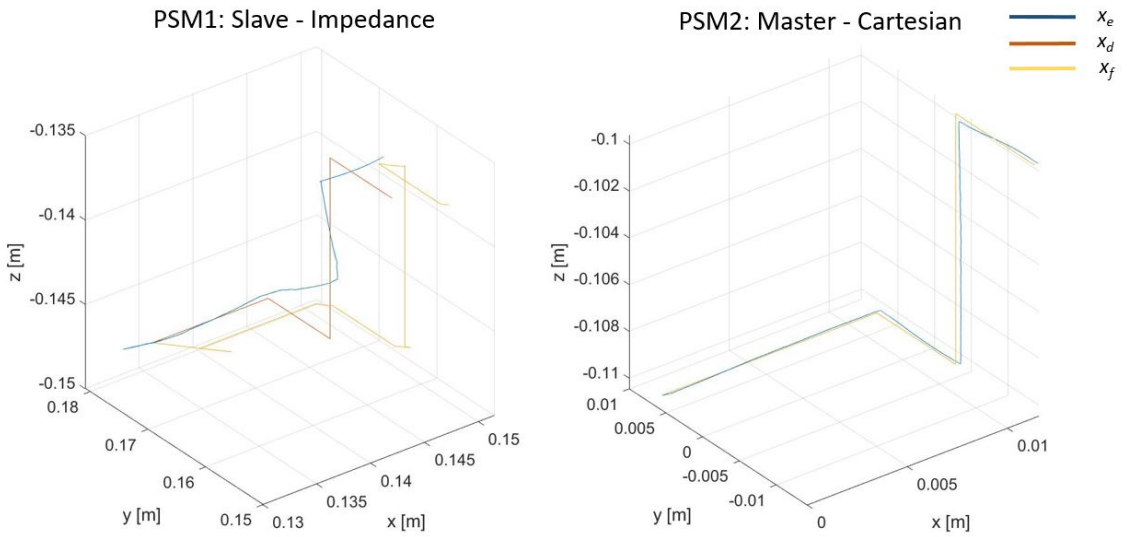


(a)

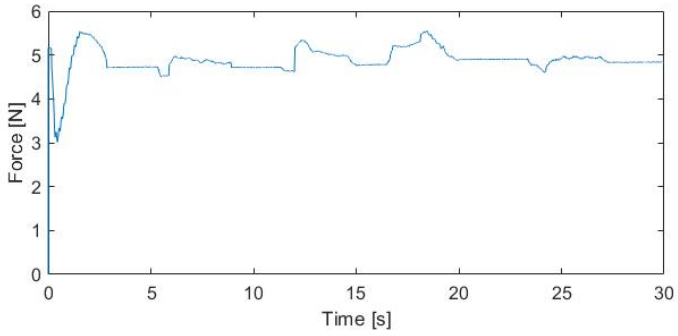


(b)

Figure 7.11: Cooperative force tracking with impedance controller with a compliant spring. (a) Motion of the Manipulators. Left: slave PSM1 Impedance, Right: master PSM2 Cartesian (b) Tensile force magnitude [N] vs Time [s]



(a)



(b)

Figure 7.12: Cooperative force tracking with impedance controller with a stiff suture string. (a) Motion of the Manipulators. Left: slave PSM1 Impedance, Right: master PSM2 Cartesian (b) Tensile force magnitude [N] vs Time [s]

Chapter 8

Conclusion and Future Work

8.1 Conclusion

This work applied kinematic link length calibration, parameter identification, controller development in simulations, and cooperative object manipulation with force tracking to a remote center of motion (RCM) double four-bar linkage Patient Side Manipulator of the da Vinci Research Kit. Parameter identification for a simplified 3 degree-of-freedom (DOF) model with least squares, weighted least squares, and semi-definite programming resulted in base parameters that predicted joint torques/-forces adequately for different trajectories. However, 15%, 15%, and 40% joint errors of measured vs. predicted torque/force ensued due to 100+ condition number and unmodeled dynamics. A computed torque controller proved that linearization with model-based dynamics (from parameter identification) and robust stick-slip friction compensation was adequate because it decreased the PD gains significantly compared to the original PD gains. In order to yield adequate tracking errors and stable interaction, this work conducted several single manipulator impedance tests with free motion, stiff environment interaction, and compliant environment interaction.

Position Cartesian cooperative controller with force tracking caused instability on a compliant environment when choosing slightly different gains. Utilizing a load cell, the cooperative object manipulation controller proved to work on a compliant and stiff environment with constant tension force tracking while simultaneously moving in Cartesian space. Nonetheless, 1-2 N maximum force tracking errors occurred in the beginning due to the lack of a damping gain. Once the controller tracked a 5 N force, the maximum force tracking error was +/- 0.5 N for both compliant and stiff environments during motion.

8.2 Discussion and Future Work

An impedance controller applied to a robot that is not designed for force control presents specific issues. Since there are no references that state the parameter identification condition number for our particular double four bar-linkage robot, we found that parallel structured robots have large condition numbers around 600-700 [59]. Serial chain robots have condition numbers as low as 4 [46]. However, adding a gravity compensation spring increased the condition number to 111. Therefore, the resulting 100-200 condition number of the PSM is contingent upon the chosen structure of the model (parallel or serial) and the chosen identified parameters (i.e., frictions).

Simplifying the 5-link double four-bar linkage structure to a 3-link mathematical model is adequate for torque prediction. However, the variability of the individual parameters restricts this work being applied to develop dynamically accurate simulations of the da Vinci. The demand for simulations has increased to provide accesibility in the medical robotics education/research community .

For a general scientific contribution, parameter identification of the Patient Side

Manipulator (PSM) should include all 7 degree-of-freedoms (DOF) with wrist dynamics. For cooperative force control, this addition in wrist dynamics would only be significant if the wrist was used for applying forces on the object.

While torque prediction was adequate for the first two joints, it was not suitable for the third prismatic joint. Constant velocity tests on the third joint results show large hysteresis and position-based friction forces. Consequently, the impedance controller without static friction compensation failed to actuate the prismatic joint as the robot moved in low velocities. While the robust static friction compensation is satisfactory for our means, a better stick-slip friction modeling using dynamic models could help alleviate these errors [49]. Additionally, modeling the residual errors/unmodeled dynamics as a stochastic disturbance and applying an adaptive controller could reduce said errors [60]. In this reference, the adaptive control was used to control an unmodeled robot. Applying adaptive control to linearize the system for impedance control would require further research.

The impedance free motion tracking errors raise concerns about the controller's efficacy in a medical setting. A marginal 2 mm error on a compliant manipulator is acceptable as long as hardware and software safety checks are implemented. Using a state machine to validate the safety of different controllers in different conditions would satisfy this controller in a medical environment [61].

A larger concern for medical applications is the stability of the controller throughout the workspace. This system with friction compensation, filter delays, desired force position, and deadbands introduces a hybrid close-looped system. Piecewise, each part of the system is proven to be stable from previous work and references [25] [51] [62]. A global stability solution would prove to validate this impedance controller research work. Similarly, the stability of the cooperative controller with position master and impedance slave with a certain DOF object has not been the-

oretically proven. Hypro toolbox [63] provides a toolbox to simulate and analyze the stability of hybrid systems using ordinary differential equations and reachable sets [64]. This toolbox could be used to prove stability.

With a particular set of gains, the cooperative object manipulation with force tracking controller in our experiments proved to be stable. Additional experiments with different objects' stiffness and DOFs, filter parameters, controller gains, and desired inertias would further validate the stability of the proposed controller. Also, it is essential to check the Jacobian matrix at different joint positions, joint velocities, and in general different instances as this is the main factor that introduces instability in the system. Also, the system should implement an integrated force sensor to show the cooperative controller stability because a lower DOF force sensor could induce energy into the closed-loop system [65].

The Jacobian matrix is not dimensionless due to the revolute-revolute-prismatic joint configuration; therefore, it should be normalized [66]. While this reference utilizes the normalized Jacobian matrix to design a robot, no reference has shown how to normalize the Jacobian for force control application as in equation (7.7).

Related work [26] explored the use of object impedance control (multiple impedance control) which uses object impedance (both end-effector impedance and object impedance). This controller proved to eliminate object inertias and results in smoother trajectories. These types of controllers should be implemented and tested on the system.

Since medical applications are a core motivation for this work, the controller should be tested on, for example, compliant gel phantoms that represent organs. The controller should also be implemented in an intraoperative minimally invasive surgery environment where the remote center of motion is located at the patients' incision.

CONCLUSION AND FUTURE WORK

The 3^{rd} da Vinci manipulator should be implemented in the cooperative object manipulation controller. The cooperation of three manipulators will apply a uniform internal force on an object, for example, a skin graft that requires low internal forces (i.e. 1-5 N) and uniform internal stresses. A particularly useful demonstration has two manipulators use the cooperative controller to reorient an object with small internal forces while the 3^{rd} manipulator is moved to various desired positions on the object previously not reachable before reorientation.

Appendix A

A.1 Least Squares Solution for Estimating the Axis of Rotation [1]

$$C = \sum_{p=1}^{P} \sum_{k=1}^{N} [(v_k^p - m^p)n]^2 \tag{A.1}$$

Minimizing the cost function of equation A.1 defines the vector representation n of the Axis of Rotation (AoR). For P markers and N frames, the vector component $v_k^p - m^p$ should be on a plane perpendicular to AoR. The cost function is first differentiated with respect to n:

$$\sum_{p=1}^{P} \sum_{k=1}^{N} (v_k^p - m^p)n(v_k^p - m^p) = 0 \tag{A.2}$$

Differentiating the cost function wrt m^p:

$$m^p n = (\frac{1}{N} \sum_{k=1}^{N} v_k^p)n = \vec{v^p} n \tag{A.3}$$

by substituting $m^p n$ from eq. (A.3) into eq. (A.2) results in:

$$\sum_{p=1}^{P} \sum_{k=1}^{N} v_k^p n - \vec{v^p} n v_k^p = 0 \tag{A.4}$$

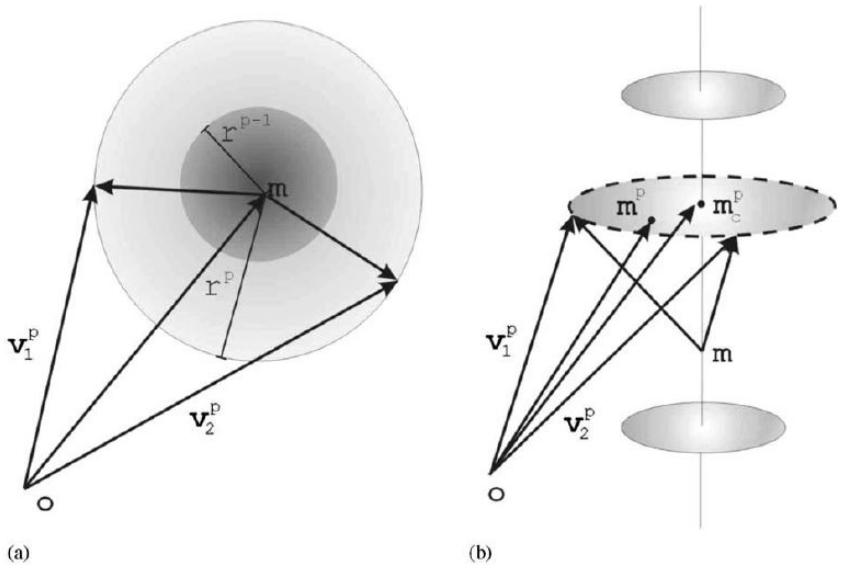


Figure A.1: (a) (b)The axis of rotation defined by point m. o is the optical camera origin where v_t^p at time instant t is a vector pointing at the circle around center of axis m^p. Source [1]

and reformulating to a least squares form:

$$\sum_{p=1}^{P}\left[(\tfrac{1}{N}\sum_{k=1}^{N} v_k^p(v_k^p)^T) - \vec{v_k^p}(\vec{v_k^p})^T\right] n = 0$$
$$An = 0 \tag{A.5}$$

With this form we can get the least squares pseudoinverse of A to get the vector that is perpendicular to axis n. To get where the axis is would need a point where vector n originates, m , this is done by minimizing the cost function:

$$C = \sum_{p=1}^{P}\sum_{k=1}^{N}[(v_k^p - m)^2 - (r^p)^2]^2 \tag{A.6}$$

$$x_{AoR} = m + \tau n \tag{A.7}$$

With a defined direction of the axis above, a point m on the axis must be

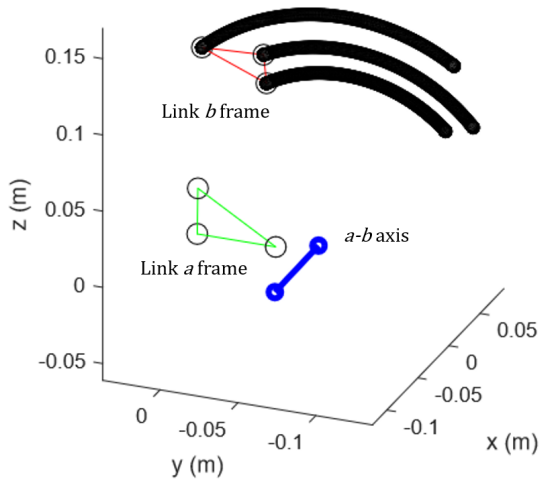


Figure A.2: Least squares solution of the resulting axis position for link length calculation. Axis a-b is found with marker data link b rotating about markers of link a.

defined to fully represent the axis. This variable m is found by again minimizing a cost function in equation (A.6).

A.2 Least Squares Solution for Estimating the Remote Center of Motion [2]

With a minimum of three optical markers defining a rigid body transformation from the optical frame as :

$$T_m^c = \begin{bmatrix} R_{tool} & \vec{p} \\ 0 & 1 \end{bmatrix} \tag{A.8}$$

where $p = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}$ and $R_{tool} \in SO^3$ rotation matrix. By sweeping across the RCM's range of motion, and taking the $R_{tool,i}$ and p_i at every time step, there exists a constant 3x1 vector (b_{tip}) from the tool frame to the RCM point and a also a

3x1 vector (b_{post}) from optical frame to the RCM point (refer to figure 4.1). The following equations formulate this procedure:

$$b_{post} = R_{tool} b_{tip} + p \tag{A.9}$$

$$\begin{bmatrix} R_{tool_1} & -I \\ R_{tool_2} & -I \\ \vdots & \vdots \\ R_{tool_N} & -I \end{bmatrix} \begin{bmatrix} b_{tip} \\ b_{post} \end{bmatrix} = - \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_N \end{bmatrix} \tag{A.10}$$

The rows of matrix $[R_{tool} - I]$ and $-p$ are expanded for the N number of data points. A pseudoinverse of equation (A.10) gives the least squares best fit for the vectors b_{tip} and b_{post}.

Bibliography

[1] Sahan S Hiniduma Udugama Gamage and Joan Lasenby. New least squares solutions for estimating the average centre of rotation and the axis of rotation. *Journal of biomechanics*, 35(1):87–93, 2002.

[2] Ziv Yaniv. Which pivot calibration? In *Medical Imaging 2015: Image-Guided Procedures, Robotic Interventions, and Modeling*, volume 9415, page 941527. International Society for Optics and Photonics, 2015.

[3] Adnan Munawar. Implementation of a surgical robot dynamical simulation and motion planning framework. 2015.

[4] David B Camarillo, Thomas M Krummel, and J Kenneth Salisbury Jr. Robotic technology in surgery: past, present, and future. *The American Journal of Surgery*, 188(4):2–15, 2004.

[5] Michael Yip and Nikhil Das. Robot autonomy for surgery. *arXiv preprint arXiv:1707.03080*, 2017.

[6] Tommaso Falcone. Introduction: robot-assisted laparoscopic surgery. *Fertility and sterility*, 102(4):909–910, 2014.

[7] Intuitive Surgical. *Intuitive Surgical Company Profile*, Accessed: 2018-07-12. `https://www.intuitivesurgical.com/company/profile.php`.

[8] Pinyo Puangmali, Kaspar Althoefer, Lakmal D Seneviratne, Declan Murphy, and Prokar Dasgupta. State-of-the-art in force and tactile sensing for minimally invasive surgery. *IEEE Sensors Journal*, 8(4):371–381, 2008.

[9] Allison M Okamura. Haptic feedback in robot-assisted minimally invasive surgery. *Current opinion in urology*, 19(1):102, 2009.

[10] Shinsuk Park, Robert D Howe, and David F Torchiana. Virtual fixtures for robotic cardiac surgery. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 1419–1420. Springer, 2001.

[11] Murat Cenk Cavusoglu, Frank Tendick, Michael Cohn, and S Shankar Sastry. A laparoscopic telesurgical workstation. *IEEE Transactions on Robotics and automation*, 15(4):728–739, 1999.

[12] Lingtao Yu, Hongwei Li, Lingyan Zhao, Sixu Ren, and Qing Gu. Automatic guidance of laparoscope based on the region of interest for robot assisted laparoscopic surgery. *Computer Assisted Surgery*, 21(sup1):17–21, 2016.

[13] Tamás Haidegger, Balázs Benyó, Levente Kovács, and Zoltán Benyó. Force sensing and force control for surgical robots. In *7th IFAC Symposium on Modeling and Control in Biomedical Systems*, volume 7, pages 413–418, 2009.

[14] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. Springer, 2016.

[15] Christian Smith, Yiannis Karayiannidis, Lazaros Nalpantidis, Xavi Gratal, Peng Qi, Dimos V Dimarogonas, and Danica Kragic. Dual arm manipulation—a survey. *Robotics and Autonomous systems*, 60(10):1340–1353, 2012.

[16] Srinivas K Prasad, Masaya Kitagawa, Gregory S Fischer, Jason Zand, Mark A Talamini, Russell H Taylor, and Allison M Okamura. A modular 2-dof force-sensing instrument for laparoscopic surgery. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 279–286. Springer, 2003.

[17] Jacob Rosen, Blake Hannaford, Mark P MacFarlane, and Mika N Sinanan. Force controlled and teleoperated endoscopic grasper for minimally invasive surgery-experimental performance evaluation. *IEEE Transactions on Biomedical Engineering*, 46(10):1212–1221, 1999.

[18] Uikyum Kim, Dong-Hyuk Lee, Woon Jong Yoon, Blake Hannaford, and Hyouk Ryeol Choi. Force sensor integrated surgical forceps for minimally invasive robotic surgery. *IEEE Transactions on Robotics*, 31(5):1214–1224, 2015.

[19] Blake Hannaford, Jacob Rosen, Diana W Friedman, Hawkeye King, Phillip Roan, Lei Cheng, Daniel Glozman, Ji Ma, Sina Nia Kosari, and Lee White. Raven-ii: an open platform for surgical robotics research. *IEEE Transactions on Biomedical Engineering*, 60(4):954–959, 2013.

[20] Renishaw plc. *neuromate® stereotactic robot*, Accessed: 2018-07-12. `http://www.renishaw.com/en/neuromate-stereotactic-robot--10712`.

[21] Ulrich Hagn, Mathias Nickl, Stefan Jörg, Georg Passig, Thomas Bahls, Alexander Nothhelfer, Franz Hacker, Luc Le-Tien, Alin Albu-Schäffer, Rainer Konietschke, et al. The dlr miro: a versatile lightweight robot for surgical applications. *Industrial Robot: An International Journal*, 35(4):324–336, 2008.

[22] Christian Ott, Ranjan Mukherjee, and Yoshihiko Nakamura. Unified impedance and admittance control. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 554–561. IEEE, 2010.

[23] Stanley A Schneider and Robert H Cannon. Object impedance control for cooperative manipulation: Theory and experimental results. *IEEE Transactions on Robotics and Automation*, 8(3):383–394, 1992.

[24] DM Weer and Stephen M Rock. Experiments in object impedance control for flexible objects. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pages 1222–1227. IEEE, 1994.

[25] David W Meer and Stephen M Rock. Coupled-system stability of flexible-object impedance control. In *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, volume 2, pages 1839–1845. IEEE, 1995.

[26] S Ali A Moosavian and Evangelos Papadopoulos. Cooperative object manipulation with contact impact using multiple impedance control. *International Journal of Control, Automation and Systems*, 8(2):314–327, 2010.

[27] Ignacio Mas and Christopher Kitts. Object manipulation using cooperative mobile multi-robot systems. In *Proceedings of the World Congress on Engineering and Computer Science*, volume 1, 2012.

[28] Nathan Michael, Jonathan Fink, and Vijay Kumar. Cooperative manipulation and transportation with aerial robots. *Autonomous Robots*, 30(1):73–86, 2011.

[29] Siddarth Sen. Surgical automation for multilateral multi-throw suturing. 2016.

[30] Steven E Naleway, William Lear, Jamie J Kruzic, and Cory B Maughan. Mechanical properties of suture materials in general and cutaneous surgery. *Journal of Biomedical Materials Research Part B: Applied Biomaterials*, 103(4):735–742, 2015.

[31] S Sarkar, HJ Salacinski, G Hamilton, and AM Seifalian. The mechanical properties of infrainguinal vascular bypass grafts: their role in influencing patency. *European journal of vascular and endovascular surgery*, 31(6):627–636, 2006.

[32] Robert T Burks and Robert Leland. Determination of graft tension before fixation in anterior cruciate ligament reconstruction. *Arthroscopy: The Journal of Arthroscopic & Related Surgery*, 4(4):260–266, 1988.

[33] Tsukasa Isago, Motohiro Nozaki, Yuji Kikuchi, Takashi Honda, and Hiroaki Nakazawa. Skin graft fixation with negative-pressure dressings. *The Journal of dermatology*, 30(9):673–678, 2003.

[34] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[35] Antal K Bejczy. Robot arm dynamics and control. 1974.

[36] Wisama Khalil and Etienne Dombre. *Modeling, identification and control of robots*. Butterworth-Heinemann, 2004.

[37] GA Fontanelli, F Ficuciello, L Villani, and B Siciliano. Modelling and identification of the da vinci research kit robotic arms. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 1464–1469. IEEE, 2017.

[38] Peter Kazanzides, Zihan Chen, Anton Deguet, Gregory S Fischer, Russell H Taylor, and Simon P DiMaio. An open-source research kit for the da vinci® surgical system. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 6434–6439. IEEE, 2014.

[39] Zihan Chen, Anton Deguet, Russell H Taylor, and Peter Kazanzides. Software architecture of the da vinci research kit. In *Robotic Computing (IRC), IEEE International Conference on*, pages 180–187. IEEE, 2017.

[40] Mark W Spong, Seth Hutchinson, Mathukumalli Vidyasagar, et al. *Robot modeling and control*, volume 3. Wiley New York, 2006.

[41] Basilio Bona and Marina Indri. Friction compensation in robotics: an overview. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pages 4360–4367. IEEE, 2005.

[42] Gianluca Antonelli, Fabrizio Caccavale, and Pasquale Chiacchio. A systematic procedure for the identification of dynamic parameters of robot manipulators. *Robotica*, 17(4):427–435, 1999.

[43] Maxime Gautier. Numerical calculation of the base inertial parameters of robots. *Journal of robotic systems*, 8(4):485–506, 1991.

[44] Cristóvão D Sousa and Rui Cortesão. Physical feasibility of robot base inertial parameter identification: A linear matrix inequality approach. *The International Journal of Robotics Research*, 33(6):931–944, 2014.

[45] Maxime Gautier and Wisama Khalil. Exciting trajectories for the identification of base inertial parameters of robots. *The International journal of robotics research*, 11(4):362–375, 1992.

[46] Jan Swevers, Chris Ganseman, Dilek Bilgin, Joris De Schutter, and Hendrik Van Brussel. Optimal robot excitation and identification. *IEEE transactions on robotics and automation*, 13(5):730–740, 1997.

[47] Carl D Meyer. *Matrix analysis and applied linear algebra*, volume 71. Siam, 2000.

[48] Fuzhen Zhang. *The Schur complement and its applications*, volume 4. Springer Science & Business Media, 2006.

[49] Jan Swevers, Farid Al-Bender, Chris G Ganseman, and Tutuko Projogo. An integrated friction model structure with improved presliding behavior for accurate friction compensation. *IEEE Transactions on automatic control*, 45(4):675–686, 2000.

[50] Sören Andersson, Anders Söderberg, and Stefan Björklund. Friction models for sliding dry, boundary and mixed lubricated contacts. *Tribology international*, 40(4):580–587, 2007.

[51] Steve C Southward, Clark J Radcliffe, and CR MacCluer. Robust nonlinear stick-slip friction compensation. *Journal of Dynamic Systems, Measurement, and Control*, 113(4):639–645, 1991.

[52] Open Source Robotics Foundation. *ROS Tutorials*, Accessed: 2018-07-12. `http://wiki.ros.org/ROS/Tutorials`.

[53] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[54] Stephen Brawner. *SolidWorks to URDF Exporter*, Accessed: 2018-07-12. `http://wiki.ros.org/sw_urdf_exporter`.

[55] Nathan P Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IROS*, volume 4, pages 2149–2154. Citeseer, 2004.

[56] Open Source Robotics Foundation. *Teleoperation Package*, Accessed: 2018-07-12. `http://wiki.ros.org/teleop_twist_keyboard`.

[57] Open Source Robotics Foundation. *Tf Tranformations Documentation*, Accessed: 2018-07-12. `http://docs.ros.org/jade/api/tf/html/python/transformations.html`.

[58] Anna Novoseltseva. Force feedback for the patient side manipulator of the davinci research kit. 2018.

[59] Nidal Farhat, Vicente Mata, Álvaro Page, and Francisco Valero. Identification of dynamic parameters of a 3-dof rps parallel manipulator. *Mechanism and Machine Theory*, 43(1):1–17, 2008.

[60] A Koivo and Ten-Huei Guo. Adaptive linear controller for robotic manipulators. *IEEE Transactions on Automatic Control*, 28(2):162–171, 1983.

[61] Fred B Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys (CSUR)*, 22(4):299–319, 1990.

[62] Milos Zefran, Francesco Bullo, and Matthew Stein. A notion of passivity for hybrid systems. In *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, volume 1, pages 768–773. IEEE, 2001.

[63] Stefan Schupp, Erika Abraham, Ibtissem Ben Makhlouf, and Stefan Kowalewski. H y p ro: A c++ library of state set representations for hybrid systems reachability analysis. In *NASA Formal Methods Symposium*, pages 288–294. Springer, 2017.

[64] Eugene Asarin, Olivier Bournez, Thao Dang, and Oded Maler. Approximate reachability analysis of piecewise-linear dynamical systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 20–31. Springer, 2000.

[65] Lawton N Verner and Allison M Okamura. Sensor/actuator asymmetries in telemanipulators: Implications of partial force feedback. In *Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2006 14th Symposium on*, pages 309–314. IEEE, 2006.

[66] Leo Stocco, SE Salcudean, and Farrokh Sassani. Matrix normalization for optimal robot design. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 2, pages 1346–1351. IEEE, 1998.