Coordinated, Multi-Arm Manipulation with Soft Robots

Dustan Paul Kraus
Brigham Young University

Coordinated, Multi-Arm Manipulation with Soft Robots

Dustan Paul Kraus

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mechanical Engineering

Brigham Young University

ABSTRACT

Coordinated, Multi-Arm Manipulation with Soft Robots

Dustan Paul Kraus
Department of Mechanical Engineering, BYU
Master of Science

Soft lightweight robots provide an inherently safe solution to using robots in unmodeled environments by maintaining safety without increasing cost through expensive sensors. Unfortunately, many practical problems still need to be addressed before soft robots can become useful in real world tasks. Unlike traditional robots, soft robot geometry is not constant but can change with deflation and reinflation. Small errors in a robot's kinematic model can result in large errors in pose estimation of the end effector. This error, coupled with the inherent compliance of soft robots and the difficulty of soft robot joint angle sensing, makes it very challenging to accurately control the end effector of a soft robot in task space. However, this inherent compliance means that soft robots lend themselves nicely to coordinated multi-arm manipulation tasks, as deviations in end effector pose do not result in large force buildup in the arms or in the object being manipulated. Coordinated, multi-arm manipulation with soft robots is the focus of this thesis.

We first developed two tools enabling multi-arm manipulation with soft robots: (1) a hybrid servoing control scheme for task space control of soft robot arms, and (2) a general base placement optimization for the robot arms in a multi-arm manipulation task. Using these tools, we then developed and implemented a simple multi-arm control scheme.

The hybrid servoing control scheme combines inverse kinematics, joint angle control, and task space servoing in order to reduce end effector pose error. We implemented this control scheme on two soft robots and demonstrated its effectiveness in task space control.

Having developed a task space controller for soft robots, we then approached the problem of multi-arm manipulation. The placement of each arm for a multi-arm task is non-trivial. We developed an evolutionary optimization that finds the optimal arm base location for any number of user-defined arms in a user-defined task or workspace. We demonstrated the utility of this optimization in simulation, and then used it to determine the arm base locations for two arms in two real world coordinated multi-arm manipulation tasks.

Finally, we developed a simple multi-arm control scheme for soft robots and demonstrated its effectiveness using one soft robot arm, and one rigid robot with low-impedance torque control. We placed each arm base in the pose determined by the base placement optimization, and then used the hybrid servoing controller in our multi-arm control scheme to manipulate an object through two desired trajectories.

Keywords: Soft Robots, Multi-Arm Manipulation, Base Placement Optimization, Pneumatic Actuation, Model Predictive Control, Soft Robot Control

ACKNOWLEDGMENTS

I want to thank my wife Ellen for her constant encouragement. Throughout this work, there were many times when an algorithm didn't work, or the robots I was using broke. These were very discouraging times, and Ellen was always there to lift my spirits and encourage me to keep pushing forward. I also want to thank her for taking care of our daughter while I spent long hours on campus working on my thesis.

I am also indebted to my parents for supporting me through my undergraduate degree and always encouraging me to pursue excellence in education. Their work ethic taught me that with hard work anything is possible.

Without funding from NASA through a NASA Early Career Faculty grant NNX14A051G and a NASA Small Business Innovation Research Phase II grant, this work would not have been possible. I am very grateful for this funding.

I am especially grateful to the BYU Upholstery Shop. When our robots broke, they stitched them back together despite them being atypical, challenging jobs.

I would like to thank my labmates for fostering an open atmosphere where we could all collaborate on challenging problems. Their input and advice helped me find the solutions to many challenging problems throughout the course of this thesis. I would also like to express my gratitude to the members of my graduate committee, Randy Beard and Mark Colton, for being willing to oversee my work and for being excellent teachers.

Finally I would like to thank my advisor, Marc Killpack. Prior to meeting Marc, I had no intention of pursuing a graduate degree. Classes with him were what helped me find my passion in engineering, and his encouragement and mentoring throughout my undergraduate degree were what inspired me to pursue higher education. Throughout this thesis work, Marc has been a pleasure to work with in every way. He helped me overcome challenging problems and find a good direction for my work. He has also been extremely supportive of maintaining a good work-life balance. I hope to someday be as big an influence in somebody's life as Marc has been in mine.

TABLE OF CONTENTS

LIST OF TABLES . . . . **vi**

LIST OF FIGURES . . . . **vii**

Chapter 1 Introduction **1**
1.1 Need for Soft Robots 1
1.2 Problem Motivation and Description 2
1.3 Thesis Overview 3
1.4 Specific Contributions 4

Chapter 2 Hardware and Sensors Description **5**
2.1 Measurement Systems 5
2.1.1 Motion Capture 5
2.1.2 HTC Vive 5
2.2 Robots 7
2.2.1 King Louie 8
2.2.2 Kaa 10
2.2.3 Baxter 10

Chapter 3 Soft Robot Task Space Control **12**
3.1 Sensor Correspondence Optimization 12
3.1.1 Previous Work 13
3.1.2 Methods 13
3.1.3 Experimental Results 17
3.2 Hybrid Servoing Algorithm 21
3.2.1 Previous Work 21
3.2.2 Methods 23
3.2.3 Experimental Results 26

Chapter 4 Multi-Arm Base Placement Optimization **31**
4.1 Previous Work 31
4.2 Methods 34
4.2.1 Forward Kinematics Binning 34
4.2.2 Objective Function 38
4.2.3 Evolutionary Optimization 41
4.3 Simulation Results 45
4.4 Conclusion 60

Chapter 5 Coordinated Multi-Arm Manipulation **61**
5.1 Previous Work 61
5.2 Methods 62
5.3 Experimental Results 63

5.3.1 Trajectory 1 65
5.3.2 Trajectory 2 76
5.3.3 Results Discussion 81

Chapter 6 Conclusion 83
6.1 Future Work 83
6.2 Contributions 84

REFERENCES 86

LIST OF TABLES

4.1 These are the forward kinematic discretization parameters we used. 35
4.2 The parameters for the experiments run in simulation to validate our base placement optimization are in this table. 45

5.1 This table contains a list of the multi-arm manipulation experiments we performed. . . 65

LIST OF FIGURES

2.1 Two of the MOCAP cameras used for tracking are shown here (Kestrel on the right and Raptor on the left). 6
2.2 IR reflective markers are used to make MOCAP frames. 6
2.3 One of the two Vive Lighthouse base stations is shown here. 6
2.4 Vive trackers are used for pose estimation. 6
2.5 The coordinate frames of the Vive trackers attached to Kaa are transferred over ROS using the vive_ros package. 7
2.6 The Vive trackers are attached to Kaa as shown here and are used for estimating joint angles and end effector pose. 8
2.7 This is the inflatable humanoid robot named King Louie used in this work. 9
2.8 The soft robots used in this work are composed of pneumatic structural links and antagonistic bladders that exert joint torque (τ). 9
2.9 This block diagram represents the control state of King Louie from prior work in our research group. 10
2.10 This is a Baxter robot designed and manufactured by Rethink Robotics. In this figure, Baxter is mounted on a mobile base, though the base was only used as a stationary stand in this work. 11
2.11 This block diagram represents the Baxter control scheme used throughout this work. . 11

3.1 Two pose sensors, **a** (MOCAP) and **b** (Vive), could be attached to a soft robot end effector to determine the end effector frame (**d**) pose. 14
3.2 This demonstrates two pose sensors (**a** and **b**) attached to the same rigid object which is moved to different poses at the timesteps shown. **A** and **B** are the respective sensor reference frames. 15
3.3 The frames used in the sensor correspondence optimization for King Louie are shown here. 19
3.4 The Cartesian end effector position reported by the Vive and MOCAP (Cortex is the MOCAP data collection software) is shown here. 21
3.5 The XYZ Euler angles reported by the Vive and MOCAP (Cortex is the MOCAP data collection software) is shown here. 22
3.6 This is the control block diagram for our hybrid task space servoing controller. We run the joint angle control at 300 Hz and the servoing control at 20 Hz. 26
3.7 The response of King Louie's end effector to a desired position (x_d, y_d, z_d) with inverse kinematics and a joint angle controller (x, y, z) and using Algorithm 3 (x_s, y_s, z_s) is shown here. 28
3.8 King Louie servoing to a position set by a Vive tracker is shown above. 28
3.9 The Cartesian response of Kaa's end effector to a desired position (x_d, y_d, z_d) is shown using Algorithm 3. 29
3.10 The orientation response of Kaa's end effector to a desired orientation $(a_{x_d}, a_{y_d}, a_{z_d})$ is shown using Algorithm 3. 29
3.11 The Cartesian response of Kaa's end effector to a second desired position (x_d, y_d, z_d) is shown using Algorithm 3. 30

3.12 The orientation response of Kaa's end effector to a second desired orientation $(a_{x_d}, a_{y_d}, a_{z_d})$ is shown using Algorithm 3. 30

4.1 We discretize the pose of each manipulator's end effector using forward kinematics and the structure shown here. 35
4.2 We disassembled Kaa to weigh each joint and link for mass and center of mass calculations. 36
4.3 One of Kaa's links with a joint on either side is shown above. 37
4.4 The homogeneous transformations used throughout the objective function for a two arm manipulation task are shown here. 40
4.5 The desired orientation bin is in the center of the cube. If it isn't reached, we also search the surrounding bins where the numbers represent each bin's proximity to the desired orientation bin. 42
4.6 The mean and max scores of the designs in each generation as the optimization progresses for experiment 1 are shown here. 44
4.7 The base configurations for Kaa and Baxter found by our optimization are shown here for experiment 1 along with the end effector poses from forward kinematics on the joint angles from the path planner. 47
4.8 The base configurations for Kaa and Baxter found by our optimization are shown here for experiment 2 along with the end effector poses from forward kinematics on the joint angles from the path planner. 47
4.9 The base configurations for Kaa and Baxter found by our optimization are shown here for experiment 3 along with the end effector poses from forward kinematics on the joint angles from the path planner. 48
4.10 The Cartesian end effector position of Baxter using the base configurations found by our optimization and forward kinematics on the joint angles found by the path planner for experiment 1 is shown here. 48
4.11 The Cartesian end effector position of Baxter using the base configurations found by our optimization and forward kinematics on the joint angles found by the path planner for experiment 2 is shown here. 49
4.12 The Cartesian end effector position of Baxter using the base configurations found by our optimization and forward kinematics on the joint angles found by the path planner for experiment 3 is shown here. 49
4.13 The Cartesian end effector position of Kaa using the base configurations found by our optimization and forward kinematics on the joint angles found by the path planner for experiment 1 is shown here. 50
4.14 The Cartesian end effector position of Kaa using the base configurations found by our optimization and forward kinematics on the joint angles found by the path planner for experiment 2 is shown here. 51
4.15 The Cartesian end effector position of Kaa using the base configurations found by our optimization and forward kinematics on the joint angles found by the path planner for experiment 3 is shown here. 51
4.16 The orientation error of both Kaa and Baxter's end effectors along the desired path for experiment 1 is shown above. 52

4.17 The orientation error of both Kaa and Baxter's end effectors along the desired path for experiment 2 is shown above. 52
4.18 The orientation error of both Kaa and Baxter's end effectors along the desired path for experiment 3 is shown above. 53
4.19 The Cartesian end effector position of Kaa using the base configurations found by our optimization perturbed by a small amount and forward kinematics on the joint angles found by the path planner for experiment 1 is shown here. 54
4.20 The Cartesian end effector position of Baxter using the base configurations found by our optimization perturbed by a small amount and forward kinematics on the joint angles found by the path planner for experiment 1 is shown here. 54
4.21 The orientation error of both Kaa and Baxter's end effectors along the desired path for the perturbed results of experiment 1 is shown above. 55
4.22 This shows 4 Kaa arms with the base configurations found by our optimization as well as the end effector poses from forward kinematics on the joint angles from the path planner for experiment 4. 55
4.23 The base configurations for Kaa and Baxter found by our optimization are shown here for experiment 5 along with the end effector poses from forward kinematics on the joint angles from the path planner. 56
4.24 The Cartesian end effector position of Baxter using the base configurations found by our optimization and forward kinematics on the joint angles found by the path planner for experiment 5 is shown here. 57
4.25 The Cartesian end effector position of Kaa using the base configurations found by our optimization and forward kinematics on the joint angles found by the path planner for experiment 5 is shown above. 57
4.26 This shows the orientation error of both Kaa and Baxter's end effectors along the desired path for experiment 5. 58
4.27 The base configurations for Kaa and Baxter found by our optimization are shown here for experiment 6 along with the end effector poses from forward kinematics on the joint angles from the path planner. 59
4.28 The orientation error of both Kaa and Baxter's end effectors along the desired path for experiment 6 is shown above. 59

5.1 This shows the multi-arm setup we used for our experiments. 64
5.2 The base configurations for Kaa and Baxter found by our optimization are shown here for trajectory 1 along with the end effector poses from forward kinematics on the joint angles from the path planner. 65
5.3 The board's Cartesian position (x_a, y_a, z_a) relative to the desired position (x_d, y_d, z_d) is shown here for a 30 second trajectory 1. 66
5.4 This shows the Cartesian error of the board throughout a 30 second trajectory 1. 67
5.5 The board's orientation $(a_{x_a}, a_{y_a}, a_{z_a})$ relative to the desired orientation $(a_{x_d}, a_{y_d}, a_{z_d})$ is shown above for a 30 second trajectory 1. 67
5.6 This shows the board's orientation error throughout a 30 second trajectory 1. 68
5.7 Kaa's valves misbehaved resulting in undesired board motion. 69
5.8 This shows the board's median, 90th percentile, and 10th percentile Cartesian positions relative to the desired position (x_d, y_d, z_d) for 10 trials of a 30 second trajectory 1. . . . 69

5.9 The median, 90th percentile, and 10th percentile Cartesian error of the board throughout 10 trials of a 30 second trajectory 1 is shown above. 70
5.10 This shows the board's median, 90th percentile, and 10th percentile axis-angle orientations relative to the desired orientation $(a_{x_d}, a_{y_d}, a_{z_d})$ for 10 trials of a 30 second trajectory 1. 71
5.11 The board's median, 90th percentile, and 10th percentile orientation error throughout 10 trials of a 30 second trajectory 1 is shown here. 71
5.12 This shows the board's Cartesian position (x_a, y_a, z_a) relative to the desired position (x_d, y_d, z_d) for a 10 second trajectory 1. 72
5.13 The Cartesian error of the board throughout a 10 second trajectory 1 is shown above. . 72
5.14 This shows the board's orientation $(a_{x_a}, a_{y_a}, a_{z_a})$ relative to the desired orientation $(a_{x_d}, a_{y_d}, a_{z_d})$ for a 10 second trajectory 1. 73
5.15 The board's orientation error throughout a 10 second trajectory 1 is shown here. 73
5.16 This shows the board's median, 90th percentile, and 10th percentile Cartesian positions relative to the desired position (x_d, y_d, z_d) for 10 trials of a 10 second trajectory 1. . . . 74
5.17 The median, 90th percentile, and 10th percentile Cartesian error of the board throughout 10 trials of a 10 second trajectory 1 is shown above. 75
5.18 This shows the board's median, 90th percentile, and 10th percentile axis-angle orientations relative to the desired orientation $(a_{x_d}, a_{y_d}, a_{z_d})$ for 10 trials of a 10 second trajectory 1. 75
5.19 The board's median, 90th percentile, and 10th percentile orientation error throughout 10 trials of a 10 second trajectory 1 is shown here. 76
5.20 This shows Kaa and Baxter with the base configurations found by our optimization as well as the end effector poses from forward kinematics on the joint angles from the path planner for trajectory 2. 77
5.21 The board's Cartesian position (x_a, y_a, z_a) relative to the desired position (x_d, y_d, z_d) for a 30 second trajectory 2 is shown here. 77
5.22 This shows the Cartesian error of the board throughout a 30 second trajectory 2. 78
5.23 The board's orientation $(a_{x_a}, a_{y_a}, a_{z_a})$ relative to the desired orientation $(a_{x_d}, a_{y_d}, a_{z_d})$ for a 30 second trajectory 2 is shown above. 79
5.24 This shows the board's orientation error throughout a 30 second trajectory 2. 79
5.25 The board's Cartesian position (x_a, y_a, z_a) relative to the desired position (x_d, y_d, z_d) for a 10 second trajectory 2 is shown here. 80
5.26 This shows the Cartesian error of the board throughout a 10 second trajectory 2. 80
5.27 The board's orientation $(a_{x_a}, a_{y_a}, a_{z_a})$ relative to the desired orientation $(a_{x_d}, a_{y_d}, a_{z_d})$ for a 10 second trajectory 2 is shown above. 81
5.28 This shows the board's orientation error throughout a 10 second trajectory 2. 82

CHAPTER 1. INTRODUCTION

1.1 Need for Soft Robots

Every year, the number of robots manufactured increases. In fact, robots play a role in the production of most things we use on a day-to-day basis; however, despite this trend, we have yet to see robots play a more personal role in our daily lives (especially in terms of robotic manipulation). Traditional, rigid robots are very effective in factories and in structured settings where the environment is known or carefully constrained. However, they fall short in situations where a model of the environment is incomplete and contact is unavoidable or even desirable. Rigid robots, though highly precise and capable, have a relatively high inertia and are heavily geared. This severely limits how quickly and safely they can move while operating in close proximity to humans to avoid unexpected collisions and high impact forces. The ability to work in an unpredictable or unmodeled environment safely is becoming increasingly important for robots as we try to deploy them in our homes, schools, hospitals, and any area with humans or fragile equipment.

There are several approaches to this challenge, including equipping robots with more sensors and changing platforms to be inherently safe. Soft, lightweight robots provide an inherently safe solution (rather than safe only by sensors and software) to using robots in unmodeled environments by maintaining safety without increasing cost through expensive sensors. Besides incidental contact with the environment, soft robots can be used to make intentional contact in ways that rigid robots often cannot, such as using a hammer or other impact tool. While such impacts would damage a rigid robot's gears and joints, the compliance of soft robots allows them to absorb these impacts and distribute the force. Other advantages of soft robots are related to the ease of transport of such systems. An inflatable robot, such as the platforms used in this work, can be deflated to be transported as a small, lightweight package that can then be reinflated at its destination with almost no assembly required. This is very convenient in applications such as space exploration or search and rescue where volume and weight restrictions are stringent.

1.2 Problem Motivation and Description

Unfortunately, many practical problems still need to be addressed before soft robots can become useful in the applications mentioned above. Unlike traditional robots, soft robot geometry is not constant but can change with deflation and reinflation. Small errors in a robot's kinematic model can result in large errors in pose estimation of the end effector. This error, coupled with the difficulty of joint angle sensing (due to the lack of joint encoders), makes it very challenging to accurately control the end effector of a soft robot in task space. The ability to accurately control a robot's end effector is of paramount importance for it to have any real world use. This is one of the focuses of this thesis.

Furthermore, helping a disabled person into a wheelchair, lifting a survivor to safety in a disaster scenario, working alongside an astronaut in space, or other similar tasks are difficult, if not impossible, with only one soft, lightweight robot arm. Though soft robots have a relatively high payload to weight ratio, they still have a low payload when compared to many rigid robots. Use of multiple arms compensates for this making these types of tasks possible and providing redundancy for many safety critical missions.

Multi-arm manipulation does not have a specific agreed-upon definition. According to the literature, it could be many fingers on a hand manipulating a small object, or many arms manipulating a large object. In fact, both of these scenarios could use the same control principles. Multi-arm manipulation can be categorized by un-coordinated and coordinated tasks. Un-coordinated manipulation tasks are those in which the arms are performing tasks that do not require interaction (e.g., one arm is moving parts while a second arm is performing an unrelated assembly task). Jobs which require two or more robotic arms to physically interact with the same object are classified as coordinated manipulation tasks ([1]).

One difficulty in coordinated, multi-arm manipulation with traditional robots is that small deviations in end effector position or orientation from any of the arms while holding a rigid object can result in large stresses on both the object and on the internal parts of the arm. Because soft robots are inherently compliant, deviations in object position result in significantly lower buildup of forces; thus, they lend themselves nicely to tasks involving several arms. Even tasks with one rigid arm and multiple soft arms become simpler as the whole system is forgiving of end effector deviations due to compliance of the soft arms. Coordinated multi-arm manipulation with soft robots is the second focus of this thesis.

1.3 Thesis Overview

The overarching goal of this thesis is to enable coordinated multi-arm manipulation with soft robots. There are many types of soft robots, but in this case we mean fabric based inflatable robots where both the structure and actuation are pneumatic.

To achieve this goal, we first needed to approach several sub-problems. First, the soft robots used in this work do not have encoders at the joints. Rather, we used relative orientations between pose sensors placed on the robot links for joint angle estimation. If multiple pose sensors were to be used for sensor fusion, or if a comparison was to be made between two pose sensors, we needed to know the homogeneous transformation between the two sensors. For this we developed an algorithm to relate two unrelated pose measurement systems (see Section 3.1). We used this algorithm to compare the accuracy of one pose sensing system to another, as well as align the sensor frames with the robot frames. This algorithm is general and can relate any pose measurement systems.

Next, we developed a task space end effector controller for a single soft robot arm. Prior to this work, joint angle estimation and control methods had been developed. However, due to the soft robot geometry which is impossible to measure accurately and changes with deflation and reinflation, joint angle control alone is not enough to accurately control the end effector position of the robot. We developed a hybrid control scheme that utilizes the joint angle estimation method described in [2] for a hybrid servoing control method wrapped around the joint angle controller described in [3]. This hybrid control scheme enables accurate end effector control of soft robot manipulators (see Section 3.2).

Having developed a task space controller for soft robots, we then approached the problem of multi-arm manipulation. One benefit of soft robot manipulators is that they are relatively lightweight and can easily be relocated to a position enabling better performance on a desired task or in a desired workspace. The placement of each arm for a multi-arm task is non-trivial. In Chapter 4, we developed an evolutionary optimization that finds the optimal arm location for any number of user-defined arms in a user-defined task or workspace. We demonstrated the utility of this optimization in simulation, and then used it to determine the arm locations for two arms (a Baxter arm and a Kaa arm) in real world tasks.

We then developed a multi-arm control scheme in Chapter 5 that is only feasible with soft robots (and at most, one rigid robot). We used the optimization in Chapter 4 to determine the optimal arm mounting locations, and then controlled the arms to complete real world tasks.

1.4 Specific Contributions

The following is a list of major contributions described within this thesis:

- Designed a sensor correspondence optimization for finding the transforms between two unrelated pose measurement systems (Section 3.1)
- Developed the first accurate task space controller for large-scale soft serial link manipulators (Section 3.2)
- Formulated a metric for scoring the ability of multiple robot arms to perform coordinated manipulation tasks with an object (Chapter 4)
- Developed a general evolutionary optimization that optimizes the base location of any number of robot arms for a user defined task or workspace (Chapter 4)
- Designed a multi-arm control scheme for soft robots and performed real world coordinated multi-arm tasks (Chapter 5)

CHAPTER 2. HARDWARE AND SENSORS DESCRIPTION

This chapter contains background information on the robots and measurement systems used throughout this thesis. Familiarity with these systems enables a better understanding of the presented work.

2.1 Measurement Systems

Other than the pressure sensors built into the soft robots (the robots are shown in Figures 2.6 and 2.7), the sensors used in this work were a motion capture (MOCAP) system and an HTC Vive virtual reality system. We used both of these systems for joint angle estimation by attaching trackers (IR reflective markers for MOCAP, as shown in Figure 2.2, and Vive Trackers, shown in Figure 2.4) to the robot links and then using relative orientations between different trackers to estimate joint angles (as discussed in [2]). We also used these sensors for end effector pose estimation and validation against ground truth.

2.1.1 Motion Capture

The motion capture system used in this work consisted of 6 Kestrel cameras and 2 Raptor cameras from Motion Analysis mounted on an aluminum frame around the robot workspace (see Figure 2.1). These cameras were used to track clusters of MOCAP IR reflective markers (see Figure 2.2) that defined coordinate frames. This setup is capable of sub-millimeter tracking accuracy when well calibrated.

2.1.2 HTC Vive

The HTC Vive virtual reality system includes two Lighthouse base stations (one of which is shown mounted to the aluminum frame in Figure 2.3). These were placed at opposite ends of the

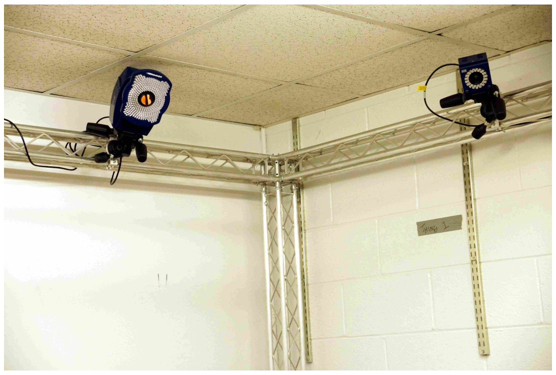


Figure 2.1: Two of the MOCAP cameras used for tracking are shown here (Kestrel on the right and Raptor on the left).

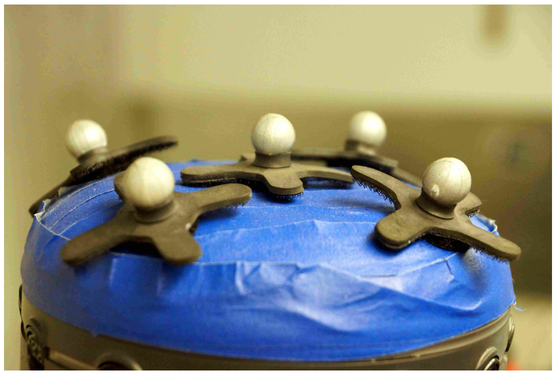


Figure 2.2: IR reflective markers are used to make MOCAP frames.

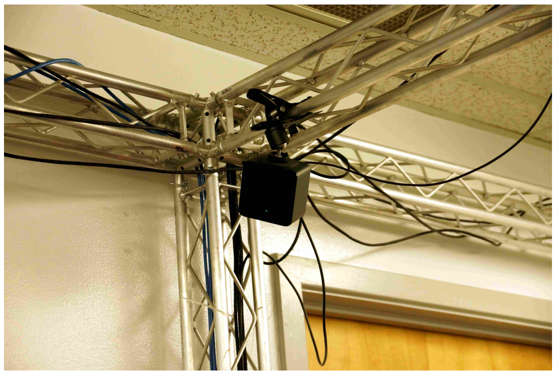


Figure 2.3: One of the two Vive Lighthouse base stations is shown here.

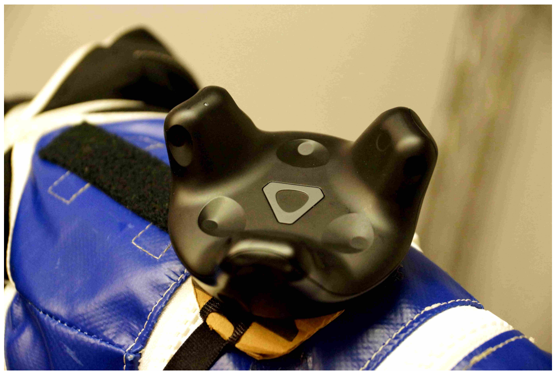


Figure 2.4: Vive trackers are used for pose estimation.

workspace and used for tracking with timed infrared pulses at 60 pulses per second. Vive Trackers (shown in Figure 2.4) attached to our robots sense the infrared pulses and combine this with data from their on-board IMUs to localize and estimate pose with millimeter precision.

The Vive and MOCAP are comparable in sensing performance; however, the Vive is self-calibrating and very easy to set up and take down. Whereas MOCAP requires that a calibration routine be performed regularly to achieve good performance, and it is not mobile. The Vive is also an order of magnitude less expensive than MOCAP. In order to use data from the Vive, we communicated with it over ROS (Robot Operating System) using the software package vive_ros [4]. ROS is a way to perform inter-process communication. For example, the vive_ros package makes the homogeneous transform of the coordinate frame of each Vive tracker available to other

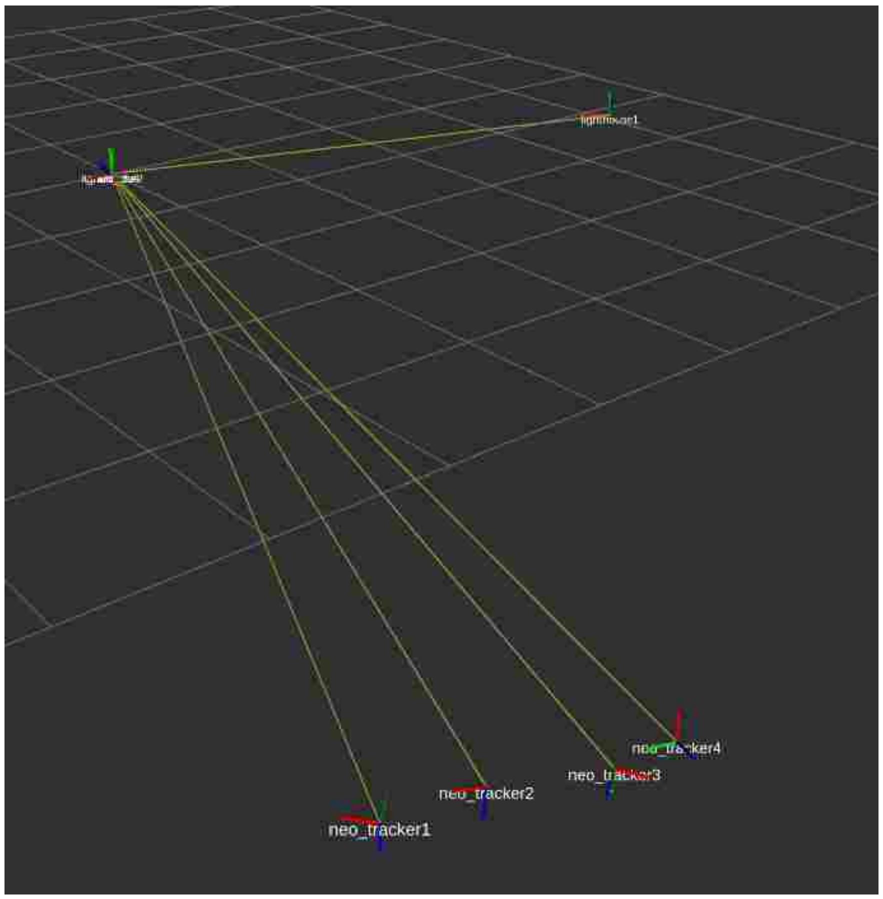


Figure 2.5: The coordinate frames of the Vive trackers attached to Kaa are transferred over ROS using the vive_ros package.

processes over ROS, which an estimation node accesses in order to perform joint angle estimation. Figure 2.5 shows a ROS visualization of the transforms to multiple trackers attached to Kaa (one of the inflatable, pneumatically actuated robots used in this research) as shown in Figure 2.6.

2.2 Robots

This section describes the three robots used in this research. Two of these robots are soft, inflatable, fabric, pneumatically actuated serial manipulators, while the third robot is a more traditional robot with compliance in the joints due to series elastic actuators and joint level impedance control. Both soft robots were developed and manufactured by Pneubotics, an affiliate of Otherlab.

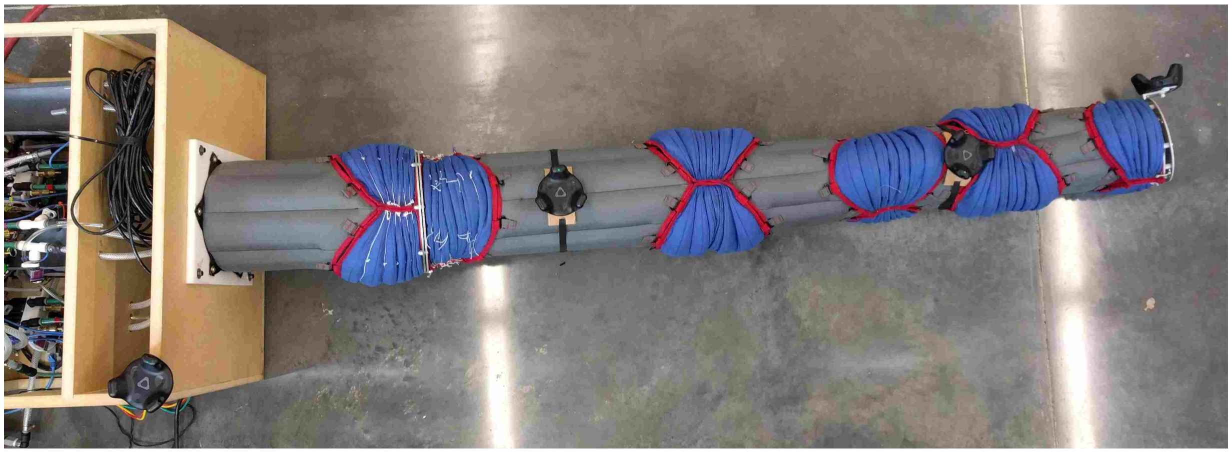


Figure 2.6: The Vive trackers are attached to Kaa as shown here and are used for estimating joint angles and end effector pose.

2.2.1 King Louie

King Louie (see Figure 2.7) is a soft robot with 10 Degrees of Freedom (DoF), four in each arm, one at the hip, and one at the abdomen; though, for this work, we only used one of his arms. The arm is composed of ballistic-grade nylon fabric, and has one structural bladder inflated to three PSIG that runs the length of the arm to provide structure. The joints are composed of antagonistic, pneumatic actuation bladders which are pressurized to vary joint stiffness and apply joint torque as shown in Figure 2.8.

King Louie was purchased from Pneubotics with a low level current controller for the valves that allowed us to control the pressure of the actuation bladders. This low level controller provided access to the actual bladder pressures over ROS, enabling us to wrap a PID pressure controller around the low level valve control. Prior to this thesis, a significant amount of work with King Louie had already been done, building upon these lower level controllers. Best et al. designed and implemented a model predictive joint angle controller on King Louie in [5]. This controller uses a linearized robot model to run a real time optimization determining which inputs will result in the desired outputs. This method treats each DoF as an individual optimization problem without considering the other DoF. Another student built upon this work by exploring a different linearization method that worked efficiently for high DoF problems while still taking into account the interaction forces between robot links [3]. This is called a disturbance torque joint angle controller. We wrapped this joint angle controller around the pressure controller to enable

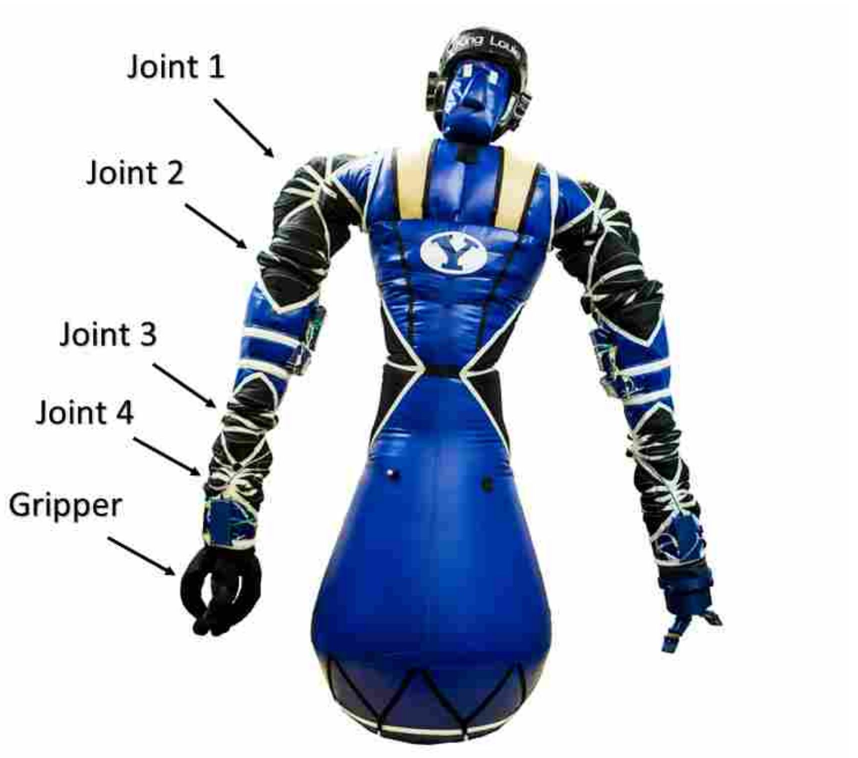


Figure 2.7: This is the inflatable humanoid robot named King Louie used in this work.

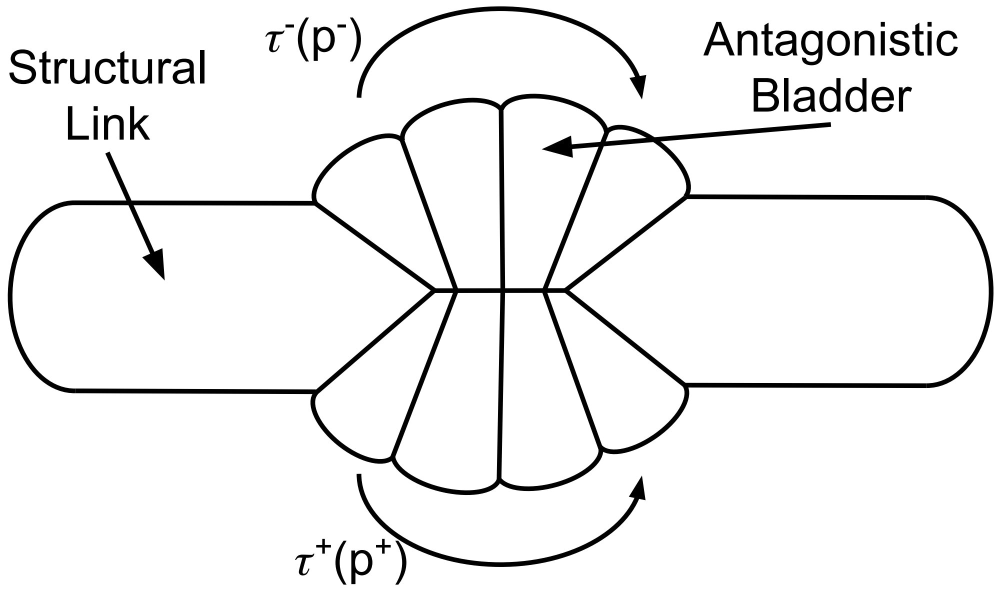


Figure 2.8: The soft robots used in this work are composed of pneumatic structural links and antagonistic bladders that exert joint torque (τ).

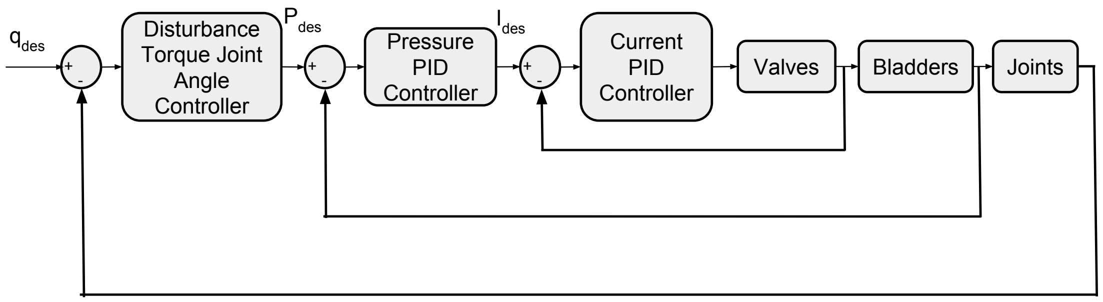


Figure 2.9: This block diagram represents the control state of King Louie from prior work in our research group.

joint angle tracking. This control architecture is shown in the block diagram in Figure 2.9. The research described in this thesis builds upon this architecture.

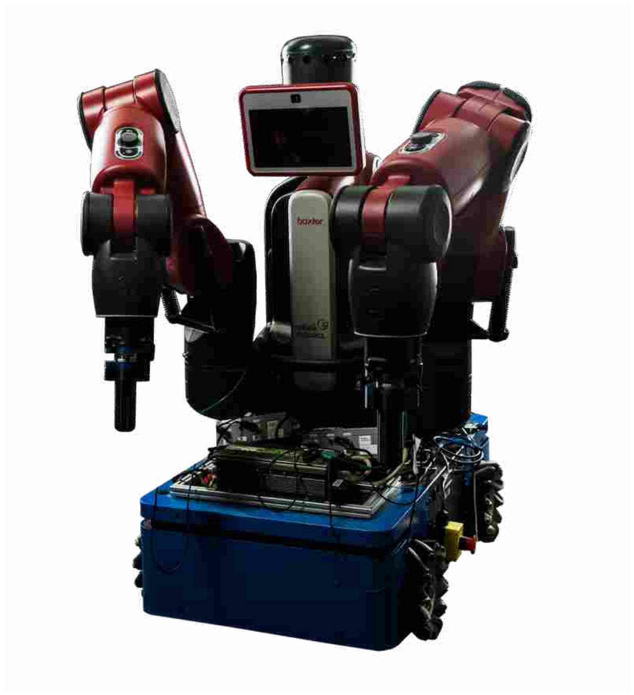
2.2.2 Kaa

Kaa is a two meter long, six DoF inflatable, pneumatically actuated serial link manipulator designed by another student in the lab [6] (see Figure 2.6). Kaa's joints are also actuated by antagonistic bladders (which have been slightly redesigned resulting in higher torque output than King Louie's joints). The same control block diagram shown in Figure 2.9 applies to Kaa. For the work in this thesis, we adapted the disturbance torque joint angle controller in [3], which was originally written for King Louie, to work for Kaa.

2.2.3 Baxter

Baxter is a rigid robot with seven DoF for each arm (see Figure 2.10). Although this is a rigid robot, the joints have a passive compliance due to series elastic actuators (a torsional spring between the actuator and the link load) at each joint, along with active compliance through low-impedance torque control.

Because this is a rigid robot, with accurate joint and torque sensors and an unchanging, known geometry, we relied on inverse kinematics to get desired joint angles (q_{goal}) for a desired end effector position. These goal joint angles were then sent to a joint angle controller, which is part of the native Baxter software package, which sends desired joint angles (q_{des}) to a low



,
;

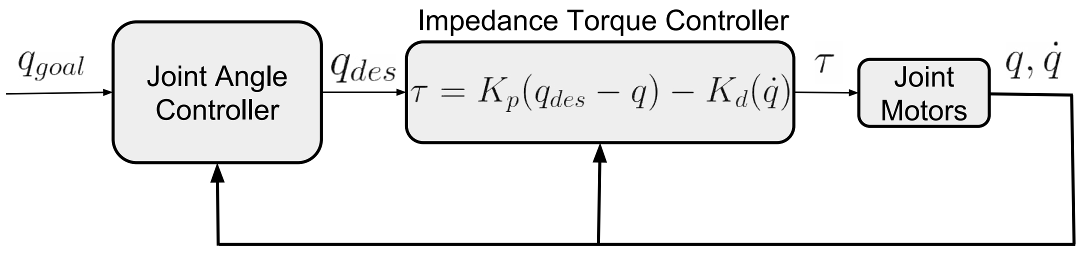


Figure 2.11: This block diagram represents the Baxter control scheme used throughout this work.

impedance torque controller. This increases the compliance of the joints further by adding active compliance. The joint angle and joint angle velocity measurements for feedback are provided by the native Baxter sensor interface. This control scheme is shown in Figure 2.11 below.

CHAPTER 3. SOFT ROBOT TASK SPACE CONTROL

In order for soft robots to be useful in real world tasks, they need to be controllable in Cartesian task space. For example, if multiple soft robots are to pick up and move a stretcher, they need to be able to control the position of their end effectors. Our control scheme for multi-arm manipulation discussed in Chapter 5 relies on each manipulator end effector being able to track a trajectory in task space. This chapter is devoted to that improving end effector trajectory tracking for soft robots and demonstrates, to our knowledge, the first implementation of a task space controller on a large-scale, soft, serial link manipulator.

Because soft robots do not have traditional encoders for joint angle sensing, we relied on relative orientations between pose sensors placed on robot links for joint angle sensing. We used these same pose sensors for end effector feedback in a task space control algorithm. Originally we used a motion capture system; however, we wanted to use the Vive for feedback instead due to its portability (use of the motion capture system restricted the robot use to within the motion capture volume). Before we could use the Vive though, we wanted to compare its performance to the motion capture system. This required finding the homogeneous transformation between the two measurement systems. We designed a novel sensor correspondence optimization (discussed in Section 3.1) for this task. Then using the Vive for feedback, we developed a task space control scheme for soft robots discussed in Section 3.2.

3.1 Sensor Correspondence Optimization

It is often desirable to use multiple pose sensors on a soft robot. For example, if one sensor had a low signal to noise ratio, while a different sensor had a high signal to noise ratio but exhibited hysteresis, these two sensors' signals could be fused together to result in better pose estimation than either of them individually. Another example is to use one pose sensor as a ground truth measurement during testing and evaluation of a new sensor. In each of these cases, it is necessary that the location of the desired pose measurement be the same, and since the sensors cannot be placed at exactly the same spot or in the same orientation due to the nature of soft robots, the homogeneous transformation between the two sensors must be found. A novel sensor correspondence optimization was designed for this task. We expect that this could be useful in many soft robot measurement systems since precise computer-aided design (CAD) models and sensor mounting points are not available for most soft robots.

3.1.1 Previous Work

Others have done similar optimizations; however, these optimizations used calibration objects like a printed square [7], a checkerboard [8], or single bright spots [9]. Holtz et al. developed a technique to find the extrinsic calibration of depth sensors without specific calibration objects [10]. All of these previous techniques are specific to cameras/depth sensors. Our optimization is unique in that it is general to any pose sensor (measuring orientation and/or position) and requires no special calibration objects. Rather, it uses optimization over changes in pose (transformations for the same sensors between different time steps) to find the transform between the two measurement systems.

3.1.2 Methods

In order to use two unrelated, pose measurement systems on a single system, or to compare their measurements, it is necessary to know the homogeneous transform between the two sensors. In this formulation we have two sensing systems with coordinate frames **a** and **b** attached to the same rigid object (i.e. the homogeneous transform between the sensors is constant). The sensors associated with **a** and **b** measure pose relative to reference frames **A** and **B** respectively. We are trying to find the pose of frame **d** relative to reference frame **A** (see Figure 3.2). For example, **d** could be the end effector frame on a soft robot, while **a** and **b** are Vive and MOCAP sensor frames placed near the end effector frame but not exactly on it as shown in Figure 3.1.

The subscripts on **a**, **b**, and **d** in Figure 3.2 represent different time steps. In this formulation we assume the homogeneous transform between sensor **a** and frame **d** is known (user defined). In reality, this transform is likely measured after mounting sensor **a**. There may be error in this;

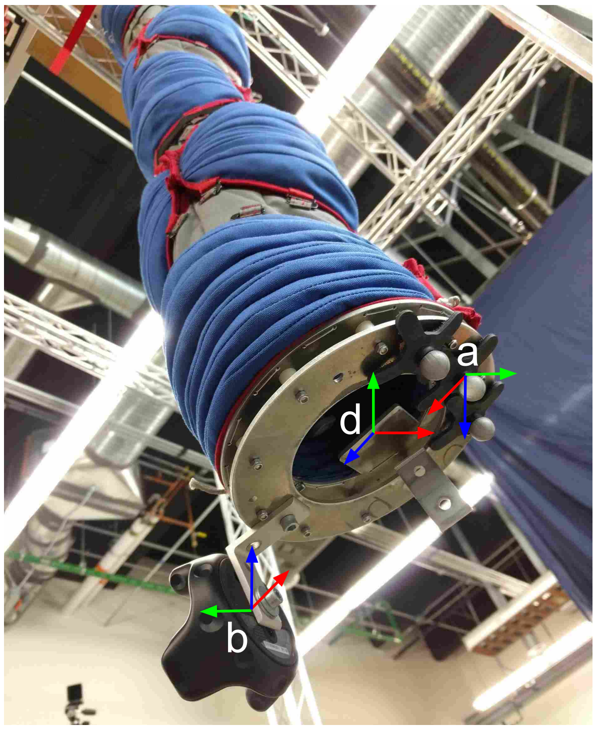


Figure 3.1: Two pose sensors, **a** (MOCAP) and **b** (Vive), could be attached to a soft robot end effector to determine the end effector frame (**d**) pose.

however, if another sensor is to be used (**b** in this case) for pose measurement of **d**, the transform between sensors **a** and **b** must be found. This is due to the fact that if a user measurement is made to find the homogeneous transform between sensor **b** and frame **d**, it will result in a different location for frame **d** than previously measured from frame **a** due to user error. This is why the homogeneous transform between sensors **a** and **b** must be known in order to use both sensors for pose estimation of the same frame **d** or for comparison of the two sensing systems.

Using each sensor (at a single time step), we can find the pose of frame **d** relative to and in terms of frame **A**, as shown in Eq. 3.1 for sensor **a** and Eq. 3.2 for sensor **b**, where T_i^j represents the homogeneous transformation that represents frame **i** in terms of and relative to frame **j**. Eq. 3.3 shows how we relate reference frame **B** to reference frame **A**.

$$T_d^A = T_a^A T_d^a \tag{3.1}$$

$$T_d^A = T_B^A T_b^B T_a^b T_d^a \tag{3.2}$$

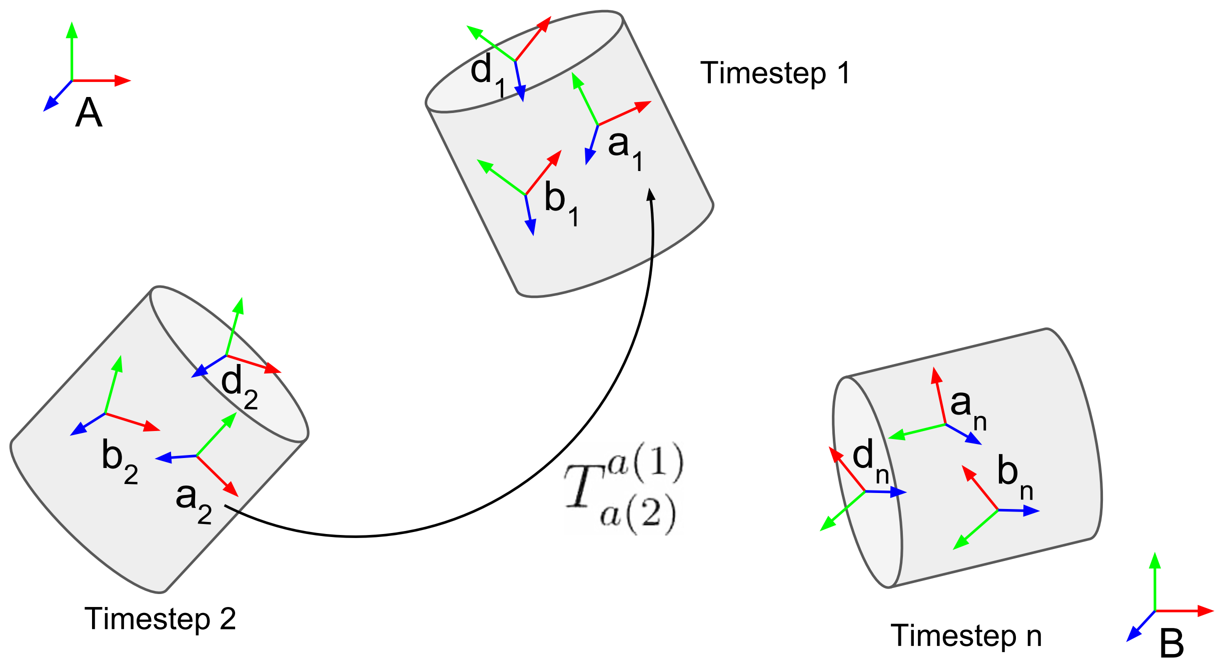


Figure 3.2: This demonstrates two pose sensors (**a** and **b**) attached to the same rigid object which is moved to different poses at the timesteps shown. **A** and **B** are the respective sensor reference frames.

$$T_B^A = T_a^A T_b^a T_B^b \tag{3.3}$$

In Eqs. 3.2 and 3.3, the unknown is T_a^b (where T_b^a is the inverse of T_a^b). In order to find T_a^b, we first attach both sensors to the same rigid object. We then proceed to move the object around while taking data simultaneously with both sensors as the object pose changes. It is important that the object pose changes a measurable amount between time steps, because the optimization relies on a change in object position and orientation which, if too small, can lead to poor numerical conditioning for the optimization (the finite difference derivatives become erroneous). So, the user must either move the object manually, and then record data points, or if moving the object continuously, they must sample slowly.

We then define a delta-transform as the homogeneous transform of a sensor from one time step to another (for example $T_{a(i)}^{a(i-1)}$ is the delta-transform for sensor **a** representing the homogeneous transform expressing the pose of sensor **a** at time step **i** in the frame of sensor **a** at time step $\mathbf{i-1}$). After collecting data, we deduce that because both sensors are attached to the same rigid object, the delta-transform for each sensor must be the same, just expressed in different frames. This is expressed mathematically in Eq. 3.4.

$$T_{a(i)}^{a(i-1)} = T_{b(i-1)}^{a(i-1)} T_{b(i)}^{b(i-1)} T_{a(i)}^{b(i)} \tag{3.4}$$

Because both sensors are attached to the same rigid object, $T_{b(i-1)}^{a(i-1)} = T_{b(i)}^{a(i)} = T_b^a$. This simplifies Eq. 3.4 into Eq. 3.5 which is the basis for the objective functions used in our optimization finding T_a^b.

$$T_{a(i)}^{a(i-1)} = T_b^a T_{b(i)}^{b(i-1)} T_a^b \tag{3.5}$$

Algorithm 1 details the procedure for running this optimization where Eqs. 3.6 and 3.7 detail the optimization objective functions used in sequential optimizations where **j** represents a row index and **k** represents a column index.

$$\begin{aligned} &\underset{\phi,\theta,\psi}{\text{minimize}} && \sum_{i=2}^{n-1}\sum_{j=1}^{3}\sum_{k=1}^{3}(R_{a(i)}^{a(i-1)} - R_b^a R_{b(i)}^{b(i-1)} R_a^b)_{j,k}^2 \\ &\text{subject to} && -\pi \leq \phi,\theta,\psi < \pi \\ &\text{where} && R_a^b \text{ is parameterized by xyz} \\ &&& \text{Euler angles } \phi, \theta, \text{ and, } \psi \end{aligned} \tag{3.6}$$

$$\begin{aligned} &\underset{x,y,z}{\text{minimize}} && \sum_{i=2}^{n}\sum_{j=1}^{3}(T_{a(i)}^{a(i-1)} - T_b^a T_{b(i)}^{b(i-1)} T_a^b)_{j,4}^2 \\ &\text{where} && T_a^b \text{ is structured as follows:} \\ &&& \left[\begin{array}{ccc|c} & & & x \\ & R_a^b & & y \\ & & & z \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \end{aligned} \tag{3.7}$$

It should be noted that this optimization is general and can be used to relate any two pose measurement systems. For example this same optimization has been used in our lab to relate MO-

Algorithm 1 General Sensor Correspondence Optimization

```
1: procedure SETUP
2:     Rigidly attach sensors a and b to the same rigid object.
3: procedure DATA COLLECTION
4:     for $i = 1$ to num. of data points desired (n) do
5:         Simultaneously collect $T^{A}_{a(i)}$ and $T^{B}_{b(i)}$ (homogeneous transforms at timestep $i$)
6:         Rotate and translate the object (this must be a non-trivial movement i.e. collect data at
           a slow rate if moving continuously). See Figure 3.2 for an example of this.
7: procedure CALCULATE DELTA-TRANSFORMS
8:     for $i = 2$ to n do
9:         $T^{a(i-1)}_{a(i)} = T^{a(i-1)}_{A} T^{A}_{a(i)}$
10:        $T^{b(i-1)}_{b(i)} = T^{b(i-1)}_{B} T^{B}_{b(i)}$
11:        $R^{a(i-1)}_{a(i)}$ = the upper left 3x3 elements of $T^{a(i-1)}_{a(i)}$
12:        $R^{b(i-1)}_{b(i)}$ = the upper left 3x3 elements of $T^{b(i-1)}_{b(i)}$
13: procedure FIND $R^{b}_{a}$
14:     Use the objective in Eq. 3.6 and an optimization library to compute $R^{b}_{a}$. We used a gradient-
        based Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm ( [11, 12]) and ran the optimiza-
        tion 50 times with random initial conditions to reduce the likelihood of finding a local mini-
        mum.
15: procedure FIND $T^{b}_{a}$
16:     Use the objective function in Eq. 3.7, $R^{b}_{a}$, and an optimization library to compute $T^{b}_{a}$ (we
        used the same optimizer and technique as procedure 13).
```

CAP and IMU measurement systems as well as MOCAP and Vive measurement systems (which is covered in Section 3.1.3).

3.1.3 Experimental Results

We tested Algorithm 1 using Vive trackers and MOCAP marker sets on King Louie. In the initial setup of King Louie, the Denavit Hartenberg (DH) frames were defined using a fixed transformation from MOCAP marker sets. However, as previously discussed, the Vive is much more portable, can work in an outdoor environment, and does not require consistent recalibration. Because of this, it was the desired sensor for pose feedback (for end effector task space control as well as joint angle estimation), but this introduced a challenge. It was impossible to place the Vive trackers at exactly the same location and orientation as the DH frames. The end effector frame (frame **d** in Figure 3.2) used for control was measured relative to and in terms of King Louie's base DH frame. So in order to obtain an accurate end effector position from the Vive trackers for comparing one sensing system against the other, the transformation between the Vive tracker frames and the MOCAP frames was needed.

From MOCAP, there were IR markers that defined King Louie's base frame ($\mathbf{m_b}$), as well as his end effector frame ($\mathbf{m_e}$). Using these marker sets, the end effector location was determined relative to King Louie's base frame as shown in Eq. 3.8 (where $T_{m_e}^{m_b}$ depicts the homogeneous transformation expressing the end effector frame in the base frame using the MOCAP marker sets, and $\mathbf{m_w}$ is the MOCAP world frame).

$$T_{m_e}^{m_b} = T_{m_w}^{m_b} T_{m_e}^{m_w} \tag{3.8}$$

In order to get end effector feedback from the Vive trackers, we placed one Vive tracker ($\mathbf{v_e}$) on the end effector (near the MOCAP end effector marker set), and the other ($\mathbf{v_b}$) on King Louie's base (near the MOCAP base marker set). Using these Vive trackers, the end effector location (the one originally defined by a fixed offset from a MOCAP marker set) relative to King Louie's base frame was determined using only the Vive as shown in Eq. 3.9 (where $\mathbf{v_w}$ is the Vive world frame). These frames are shown on King Louie in Figure 3.3.

$$T_{m_e}^{m_b} = T_{v_b}^{m_b} T_{v_w}^{v_b} T_{v_e}^{v_w} T_{m_e}^{v_e} \tag{3.9}$$

These measurement systems (the Vive and MOCAP) had no common frames, so we used the general optimization described in Section 3.1.2 to find the needed homogeneous transformations. In this case, $T_{v_w}^{v_b}$ and $T_{v_e}^{v_w}$ were known and we needed $T_{m_e}^{v_e}$ and $T_{v_b}^{m_b}$ in order to find $T_{m_e}^{m_b}$ using the Vive. In order to find the unknown transformations ($T_{m_e}^{v_e}$ and $T_{v_b}^{m_b}$), data was collected and run through an optimization as shown in Algorithm 2, which is an application of Algorithm 1 to this specific case. The following two optimization objective functions, Eqs. 3.10 and 3.11, which again are specific applications of Eqs. 3.6 and 3.7 are used and referenced in Algorithm 2 where **j** and **k** are row and column indices respectively.

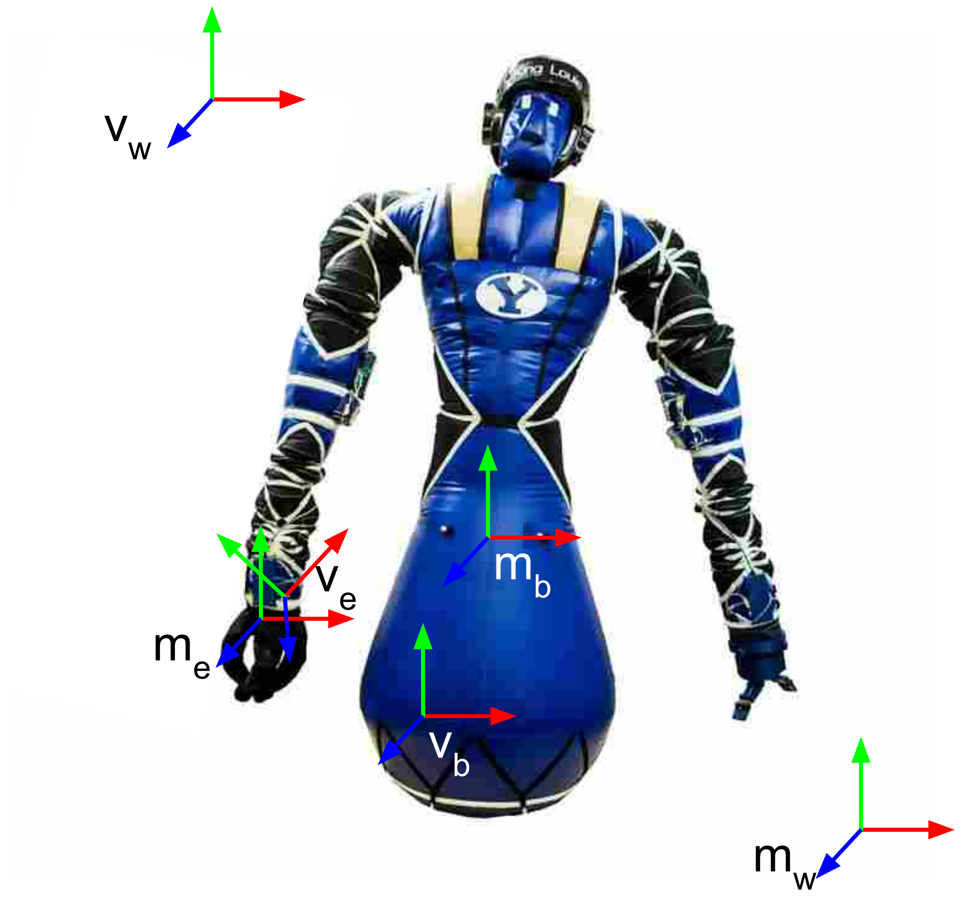


Figure 3.3: The frames used in the sensor correspondence optimization for King Louie are shown here.

$$\begin{aligned}
&\underset{\phi,\theta,\psi}{\text{minimize}} && \sum_{i=2}^{n-1}\sum_{j=1}^{3}\sum_{k=1}^{3}\left(R_{v_{e(i)}}^{v_{e(i-1)}} - R_{m_e}^{v_e} R_{m_{e(i)}}^{m_{e(i-1)}} R_{v_e}^{m_e}\right)_{j,k}^{2} \\
&\text{subject to} && -\pi \leq \phi,\theta,\psi < \pi \qquad (3.10) \\
&\text{where} && R_{m_e}^{v_e} \text{ is parameterized by xyz} \\
& && \text{Euler angles } \phi,\theta,\ \text{and}, \psi
\end{aligned}$$

Algorithm 2 Sensor Correspondence Optimization Application

```
1: procedure SETUP
2:     Rigidly attach a Vive tracker to the same link as a MOCAP DH frame (the end effector in
       this case).
3: procedure DATA COLLECTION
4:     for $i = 1$ to num. of data points desired (n) do
5:         Simultaneously collect $T_{v_{e(i)}}^{v_w}$ and $T_{m_{e(i)}}^{m_w}$ (transforms at timestep $i$)
6:         Rotate and translate the robot link
7: procedure CALCULATE DELTA TRANSFORMATIONS
8:     for $i = 2$ to n do
9:         $T_{v_{e(i)}}^{v_{e(i-1)}} = T_{v_w}^{v_{e(i-1)}} T_{v_{e(i)}}^{v_w}$
10:        $T_{m_{e(i)}}^{m_{e(i-1)}} = T_{m_w}^{m_{e(i-1)}} T_{m_{e(i)}}^{m_w}$
11: procedure FIND $R_{v_e}^{m_e}$
12:     Use the objective function in Eq. 3.10 and the optimization library of your choice to com-
        pute $R_{v_e}^{m_e}$ (we used a gradient based BFGS algorithm)
13: procedure FIND $T_{v_e}^{m_e}$
14:     Use the objective function in Eq. 3.11, $R_{v_e}^{m_e}$, and the optimization library of your choice to
        compute $T_{v_e}^{m_e}$ (again, we used a gradient based BFGS algorithm)
15: procedure REPEAT TO COMPUTE $T_{m_b}^{v_b}$
16:     Repeat steps 1-19 but using the MOCAP markers and Vive trackers on the base rather than
        the end effector
```

$$\begin{aligned}
&\underset{x,y,z}{\text{minimize}} \quad \sum_{i=2}^{n}\sum_{j=1}^{3} \left(T_{v_{e(i)}}^{v_{e(i-1)}} - T_{m_e}^{v_e} T_{m_{e(i)}}^{m_{e(i-1)}} T_{v_e}^{m_e}\right)_{j,4}^{2} \\
&\text{where} \quad T_{m_e}^{v_e} \text{is structured as follows:} \\
&\left[\begin{array}{ccc|c} & & & x \\ & R_{m_e}^{v_e} & & y \\ & & & z \\ \hline 0 & 0 & 0 & 1 \end{array}\right]
\end{aligned} \tag{3.11}$$

We used the sensor correspondence optimization in Algorithm 2 to find the transformations between the Vive trackers and MOCAP marker sets on King Louie ($T_{m_e}^{v_e}$ and $T_{v_b}^{m_b}$). We then collected data while moving the arm around to compare the end effector position reported by MOCAP in Eq. 3.8 and the Vive in Eq. 3.9. As shown in figure 3.4 and 3.5, the end effector position and

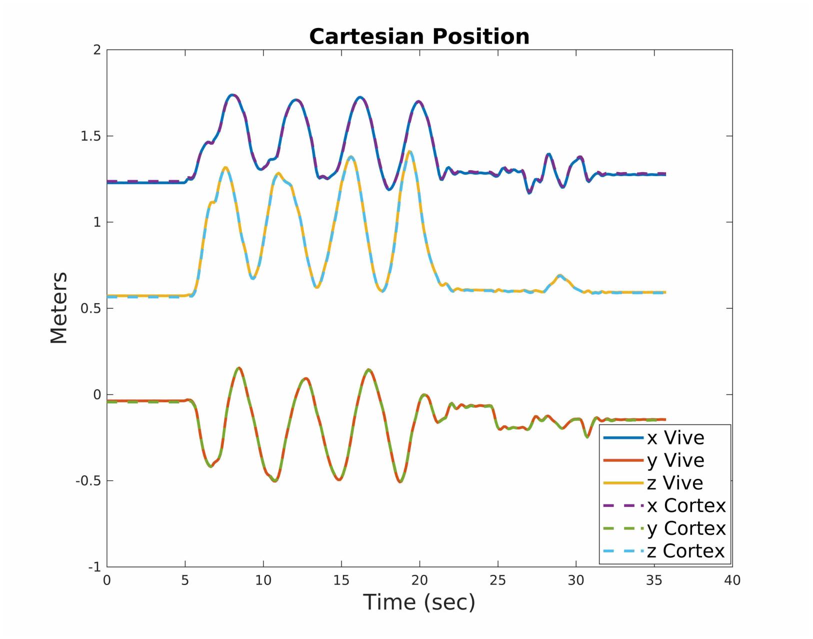


Figure 3.4: The Cartesian end effector position reported by the Vive and MOCAP (Cortex is the MOCAP data collection software) is shown here.

orientation reported by the Vive and MOCAP systems were nearly identical. The average Cartesian error between these two measurement systems was 0.82 cm and the average Euler angle error between these two measurement systems was 0.56 deg. We expect that some of this error may be due to the fact that King Louie's end effector is not truly rigid, so the homogeneous transform between the MOCAP and Vive frames was not truly constant.

3.2 Hybrid Servoing Algorithm

3.2.1 Previous Work

There has recently been significant work in soft-robot control as outlined in the survey paper by Thuruthel et al. [13]. Some of the most relevant literature is that on task-space control. Largilliere et al. used finite element analysis to solve for inverse kinematics (IK) to successfully track a trajectory for a small deformable robot made of silicone [14]. Others showed promising results in task-space control of continuum arms. Kapadia et al. demonstrated PD control for end effector control of the OctArm (a robotic manipulator with 3 continuum sections) using teleoperation; however, their experiments all took place on a horizontal surface, and they only controlled

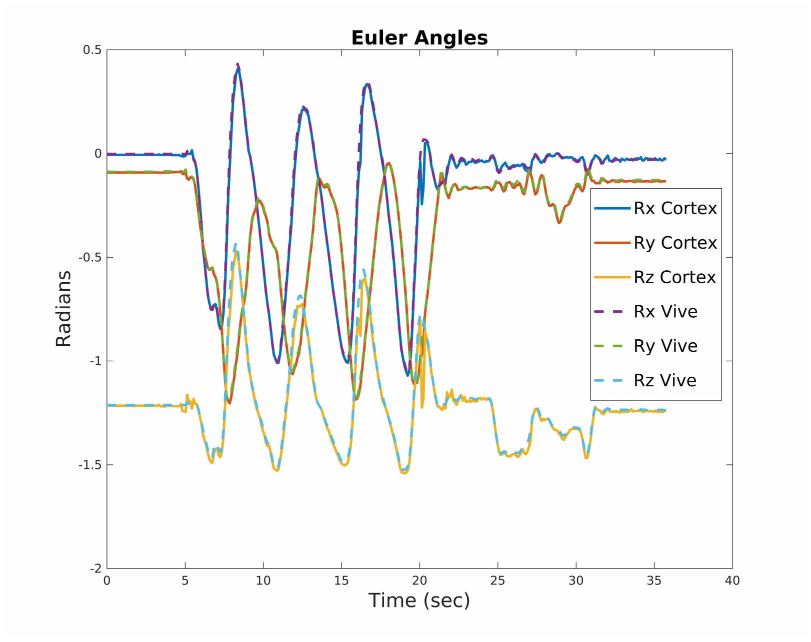


Figure 3.5: The XYZ Euler angles reported by the Vive and MOCAP (Cortex is the MOCAP data collection software) is shown here.

the robot in the Cartesian plane (2 DoF) [15]. Bajo et al. demonstrated configuration space control of a continuum manipulator with three links, but the manipulator was significantly smaller scale than the platforms used in this thesis, and they did not control in task space [16]. Yip et al. were able to control a 2 DoF planar continuum manipulator (again much smaller in scale than the robots in this thesis) to task space positions [17]. Kang et al. performed model-free task-space control on their continuum robot arm using an adaptive Kalman filter [18]. Melingui et al. used a distal supervised learning scheme and an adaptive neural network control strategy for task-space control on their continuum arm [19]; none of this control was in 6 DoF task space. Wang et al. used visual servoing for end-effector position control of a soft robot [20, 21]. Their work differs from this thesis in that their feedback and position control were both in pixel space (with no real task space goals), while our experiments are all done directly in task space. Furthermore, the soft robotic platforms with which we performed our experiments (see Fig. 2.7 and 2.6) are much larger in scale (serial manipulators over 1 m long) than the platforms in [20, 21] with more significant gravitational loads.

Others have used task-space servoing to reduce the error in flexible robotic catheters [22] [23], but again, these are on a smaller scale (both in terms of end-effector speed and manipulator length) than our robotic platform. Also, the servoing algorithms used are different from ours, as we used a hybrid controller combining model predictive control (MPC), inverse kinematics, and servoing. This hybrid control is useful as it combines the speed of the joint angle model predictive control with the task space loop closure of the servoing control, which has a slower response than the joint angle control, but reduces task space error. Thuruthel et al. specifically mentioned that the work in hybrid control approaches, such as doing traditional joint angle control and then servoing as presented in this thesis, is an unexplored field of research [13].

There have also been recent successful control of soft, inflatable robots, such as the platforms used in this thesis, for both configuration and stiffness. Best et al. implemented joint angle MPC for a serial, inflatable arm that treated each joint as an individual inverted pendulum. This enabled the use of MPC by reducing the control space in the MPC to be tractable to high degrees of freedom (DoF) and allowing for faster optimization speeds [24]. Terry et al. extended this work by finding a tractable method to take into account the dynamics of all the links for model predictive control while still keeping the control of each joint decoupled [3]. Gillespie et al. implemented a simultaneous joint angle and stiffness controller on a single DoF inflatable robot [25]. The work presented in this thesis extends this success to accurate end effector position control for these kinds of robots.

3.2.2 Methods

Given a desired pose in task space, inverse kinematics can be used along with a kinematic model (using a traditional pin-joint rigid-body model) of the manipulator to determine the joint angles that should result in the desired end effector pose. However, even with accurate joint angle control, this still results in some task space pose error at the end effector due to imperfections in the kinematic model. The kinematic model can change frequently in soft robots due to deflation and reinflation, as well as the difficulty in constructing an accurate kinematic model from measurements to begin with. A common technique used in robotics to eliminate end effector error is visual servoing. This is typically done in pixel space; however, for this work, we servo directly in task space, as task space accuracy was our objective.

Algorithm 3 describes the hybrid servoing algorithm we developed. This algorithm is represented in block diagram form in Figure 3.6. First, we use inverse kinematics and a joint angle controller to get close to a desired pose (if the robot has at least six DoF) or position (if the robot has less than six DoF). We used the inverse kinematics software package TRAC-IK, which is a fast, robust inverse kinematics library [26]. Once the joint angles are within an error tolerance of the commanded joint angles (the check steady state block in Figure 3.6), we enter the servoing part of our algorithm to eliminate the remaining end effector pose error. The actual end-effector pose is measured and delta joint angles are calculated by converting the task space pose error into small joint angle changes which are added to the previous joint angle command. These new joint angles are then sent to the joint-angle controller. This servoing approach plays the role of a task-space integrator because delta joint angles are continuously being added to the commanded joint angles when there is task-space error. It is possible to servo the whole distance to the desired position; however, we found that a quicker response is achievable by using a kinematic model to derive commanded joint angles using inverse kinematics, and commanding these joint angles prior to servoing. This speed increase is due to the fact that the joint space control response is quicker than the task space servoing response.

We adapted the Jacobian Transpose Method, which was first introduced in [28] and [29], for our servoing controller as shown in Eqs. 3.12 - 3.16. We chose to use the Jacobian transpose method for its numerical stability even in the presence of singularities of the Jacobian. This is used in lines 12 and 13 of Algorithm 3. Eq. 3.12 shows the relationship between forces/moments at the end effector expressed in the base frame and joint torques via the geometric Jacobian. In Eq. 3.13, we set F equal to a virtual spring and damper force pulling the actual end effector pose towards the desired end effector pose. K_p and K_d are 6×6 diagonal gain matrices where the upper 3×3 diagonal elements are the gains on the virtual, linear spring forces and the lower 3×3 diagonal elements are the gains on the virtual, torsional spring elements. Δx is the difference between the desired and actual end effector pose (represented as a column vector where the first three elements are the Cartesian x, y, and z positions while the final three elements are the axis-angle x, y, and z rotations) and $\Delta \dot{x}$ is the time derivative of Δx. Eq. 3.14 states that the joint torques will cause the joint velocities in the desired direction. Substituting Eqs. 3.14 and 3.13 into Eq. 3.12 results in Eq. 3.15. We added K_q as an $n \times n$ diagonal gain matrix (where n is equal to the number of DoF the manipulator has) that controls the amount that each joint will move with each servoing time step. This gives us control over the amount each individual joint moves. For example, actuation of

Algorithm 3 Vive Servoing

```
1: procedure CHOOSE REACHABLE DESIRED POSITION ($x_{des}$)
2: procedure CALCULATE THE JOINT ANGLES ($q_{des}$) THAT MAP TO $x_{des}$ USING TRAC-IK [26]
3:     Store $q_{des}$ as $q_{cmd}$
4:     Send $q_{cmd}$ to a joint space controller
5:     Wait for the joint angles to be within a desired error tolerance of $q_{cmd}$
6:     Collect actual end effector position $x_{act}$ from Vive
7:     Calculate $\Delta x = x_{des} - x_{act}$
8:     Calculate $\Delta\dot{x} = \frac{\Delta x[t_k] - \Delta x[t_{k-1}]}{\Delta t}$ where $t_k$ and $t_{k-1}$ represent the current and previous time steps
       and $\Delta t$ is the time difference between steps (the inverse of the control rate)
9: procedure SERVO TO ELIMINATE REMAINING TASK SPACE ERROR $\Delta x$
10:    for $\Delta x <$ error tolerance do
11:        Calculate the Jacobian ($J$) using sympybotics (a python robotics library) [27]
12:        Calculate a small joint angle movement in the correct direction ($\Delta q$) using a numerical
           inverse kinematics approximation (the Jacobian Transpose Method)
           $\Delta q = K_q[(J)^T K_p \Delta x - (J)^T K_d \Delta\dot{x}]$
           where $K_p$ and $K_d$ are diagonal 6x6 gain matrices that determines the aggressiveness of $\Delta q$
           and $K_q$ is a nxn diagonal matrix that determines the weight on each joint where n is the number
           of joints
13:        Calculate a new joint angle command to send to the joint space controller $q_{cmd}[t_{k+1}] =$
           $q_{cmd}[t_k] + \Delta q$ where $t_{k+1}$ and $t_k$ represent the next and current time steps
14:        Send $q_{cmd}$ to a joint space controller
15:        Collect actual end effector position $x_{act}$ from Vive
16:        Calculate $\Delta x = x_{des} - x_{act}$
17:        Calculate $\Delta\dot{x} = \frac{\Delta x[t_k] - \Delta x[t_{k-1}]}{\Delta t}$ where $t_k$ and $t_{k-1}$
```

the more proximal joints results in more end effector movement than the more distal joints. More refined movement is often desirable towards the end of a movement, so we could use K_q to move the distal joints more than the proximal joints resulting in smaller end effector movements. We absorb the time step t into the system gains in order to convert $\dot{q}$ into Δq which is the small amount each joint moves in each time step of the servoing algorithm, as shown in Eq. 3.16, where $[t_{k+1}]$ and $[t_k]$ represent time steps $k+1$ and k respectively. The rate at which the servoing controller runs plays a big part in the system's task space response. If it is run too quickly, the joint angles grow too quickly resulting in an unstable system. If it is run too slowly, the task space response is too slow. We run the servoing controller at 20 Hz to achieve good performance.

$$\tau = J^T F \tag{3.12}$$

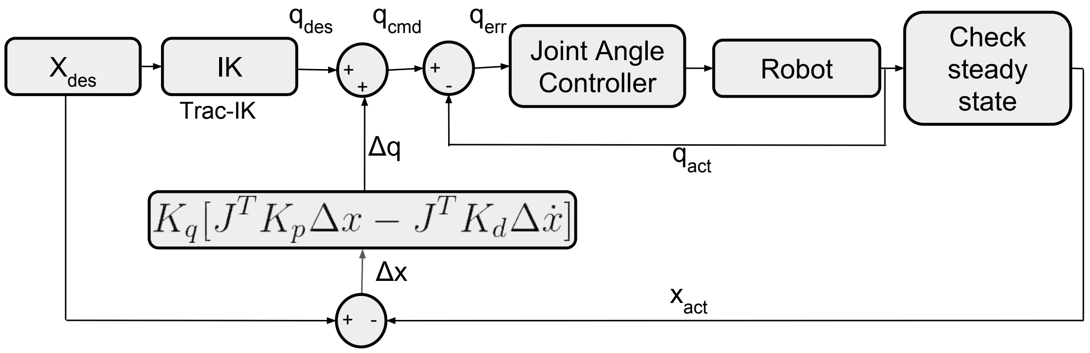


Figure 3.6: This is the control block diagram for our hybrid task space servoing controller. We run the joint angle control at 300 Hz and the servoing control at 20 Hz.

$$F = \begin{bmatrix} F_x \\ F_y \\ F_z \\ M_x \\ M_y \\ M_z \end{bmatrix} = K_p \Delta x - K_d \Delta \dot{x} \tag{3.13}$$

$$\tau \propto \dot{q} \tag{3.14}$$

$$\dot{q} = K_q J^T (K_p \Delta x - K_d \Delta \dot{x}) \approx \Delta q \tag{3.15}$$

$$q_{cmd}[t_{k+1}] = q_{cmd}[t_k] + \dot{q}_{cmd}[t_k]\Delta t = q_{cmd}[t_k] + \Delta q \tag{3.16}$$

3.2.3 Experimental Results

In order to validate our servoing algorithm on real hardware, we implemented Algorithm 3 on King Louie and Kaa demonstrating that high task space accuracy is achievable with large-scale soft robots.

King Louie

We chose several reachable positions in King Louie's workspace, and implemented Algorithm 3. The time response for one of the positions is shown in Figure 3.7, where x_d, y_d, and z_d represent the desired Cartesian position; x, y, and z represent the Cartesian response using only a joint angle controller; and x_s, y_s, and z_s represent the Cartesian response using the hybrid servoing algorithm described. Because King Louie only has four DoF, he is not controllable in six DoF task space (pose and orientation). So, we chose to only control the Cartesian position of his end effector. To do this, we slightly modified the algorithm by changing the gain matrices K_p and K_d to be 3×3, and only using the upper three rows of the Jacobian (making it 3×4).

Using this servoing method (with the Vive for feedback), we were able to drive the Cartesian error between the desired and actual end effector position to 2.2 mm. The overshoot for the example shown was 19.4%, 16.5%, and 2.5% for x, y, and z respectively. The x, y, and z 95% settling times were 4.90, 10.71, and 9.37 seconds respectively. The overshoot is largely affected by the accuracy of the kinematic model and is hardly affected by tuning the servoing controller since the initial response is the joint-angle controller response where initial joint angles are calculated using IK. The settling time was reduced by tuning K_p and K_d until the system response in task space was close to critically damped. The steady state Cartesian error using only joint-angle control (no servoing) with joint angles obtained from IK using our kinematic model was 6.7 cm.

Results for other positions were comparable as shown in https://youtu.be/V73oJvkMZ70 where we control to multiple positions set by a Vive tracker held by the user (see Figure 3.8).

Kaa

We also implemented Algorithm 3 on Kaa. Because Kaa has six DoF, we were able to select several six dimensional poses in his workspace for end effector control testing without modifying the algorithm. The time response for two of these poses is shown in Figures 3.9 and 3.10 for the first desired pose and Figures 3.11 and 3.12 for the second desired pose. As shown, we were able to servo to within 3cm while driving the orientation error to less than 2°.

This amount of error is greater than the error on King Louie for two reasons. First, in this case, we are servoing in six DoF rather than three, and the orientation and position components of

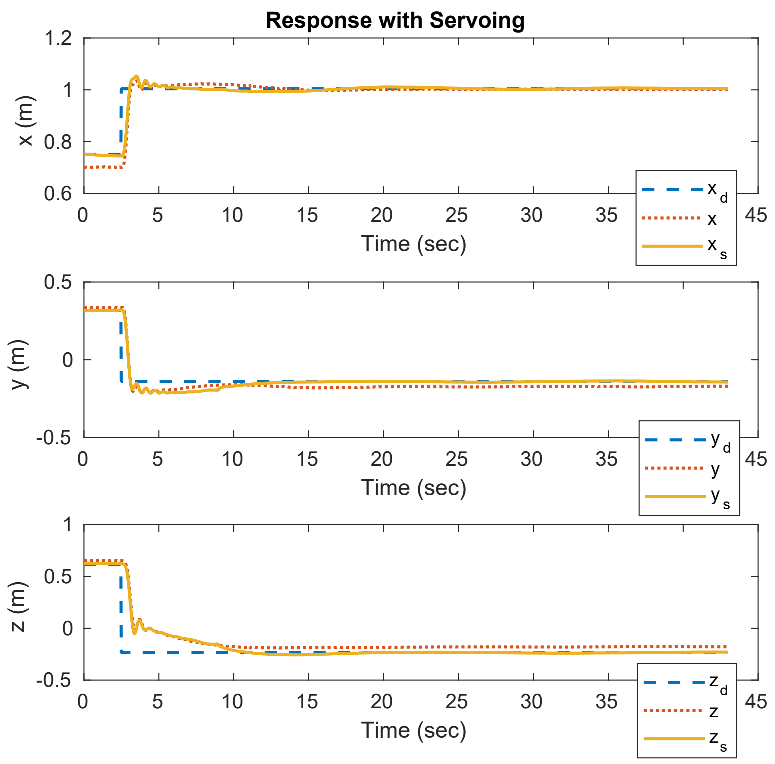


Figure 3.7: The response of King Louie's end effector to a desired position (x_d, y_d, z_d) with inverse kinematics and a joint angle controller (x, y, z) and using Algorithm 3 (x_s, y_s, z_s) is shown here.

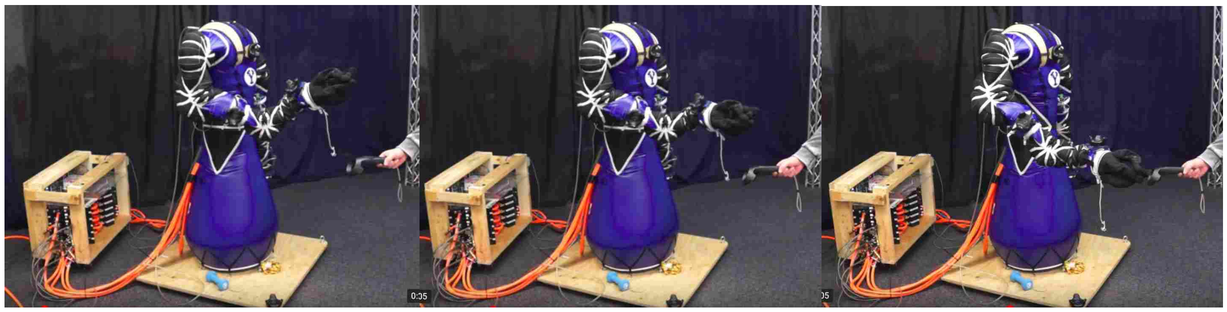


Figure 3.8: King Louie servoing to a position set by a Vive tracker is shown above.

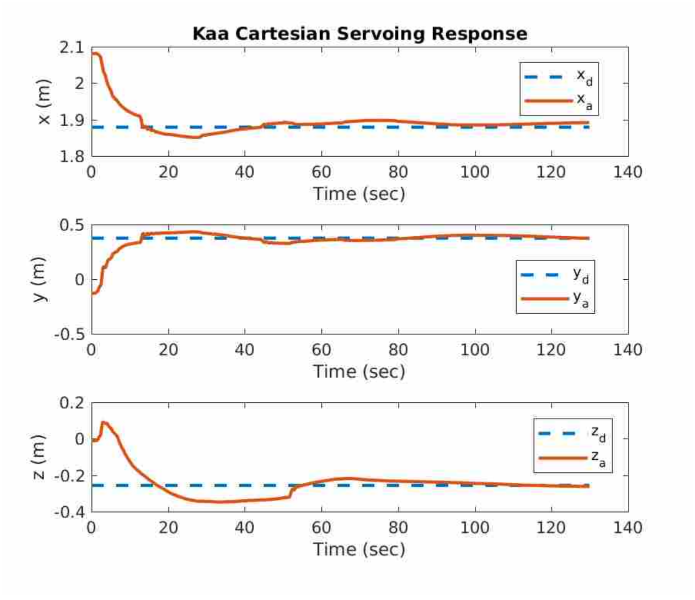


Figure 3.9: The Cartesian response of Kaa's end effector to a desired position (x_d, y_d, z_d) is shown using Algorithm 3.

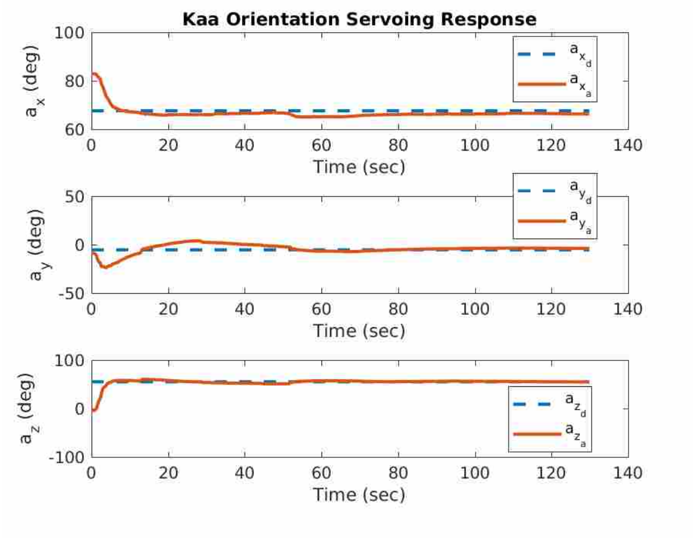


Figure 3.10: The orientation response of Kaa's end effector to a desired orientation $(a_{x_d}, a_{y_d}, a_{z_d})$ is shown using Algorithm 3.

the servoing fight each other. Second, Kaa is almost twice the length as King Louie. Nonetheless, the amount of error we were able to achieve is accurate enough to perform real world tasks as demonstrated in Chapter 5.

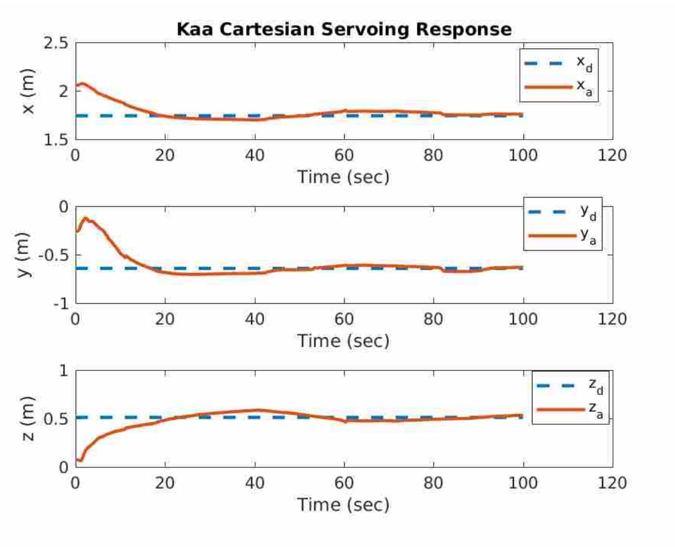


Figure 3.11: The Cartesian response of Kaa's end effector to a second desired position (x_d, y_d, z_d) is shown using Algorithm 3.

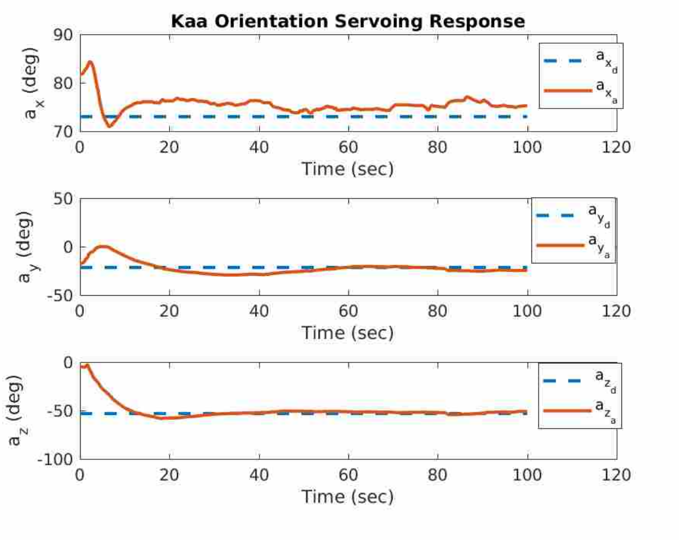


Figure 3.12: The orientation response of Kaa's end effector to a second desired orientation $(a_{x_d}, a_{y_d}, a_{z_d})$ is shown using Algorithm 3.

CHAPTER 4. MULTI-ARM BASE PLACEMENT OPTIMIZATION

The soft robots used in this work have a relatively high payload to weight ratio, though each robot's payload is relatively low when compared with traditional geared robots. This low payload can be compensated for by using multiple robot arms to manipulate an object. One of the benefits of soft robots, like the ones used in this work, is their low weight. Because of this low weight, multiple soft robots can be easily moved to a desired workspace enabling multi-arm manipulation tasks (or several individual tasks until there is a need for collaboration). However, this raises the challenge of determining where each robot should be placed in order to perform desired tasks.

In this chapter, we discuss an optimization we developed to determine the optimal mounting position (or mobile base position) of each robot in a multi-arm manipulation scenario. In this optimization, the user defines the desired path or workspace of the object to be manipulated, the end effector grasping locations on the object, and the object mass. The optimization returns the optimal pose of the base frame of each robot relative to a world frame.

We first discuss previous work on robot base placement optimization in section 4.1. In section 4.2 we discuss a forward kinematics discretization that the optimization relies on followed by the optimization objective function and the evolutionary algorithm used to find a good solution. Finally, we demonstrate our algorithm's utility in simulation in section 4.3.

4.1 Previous Work

Positioning of robotic manipulators in an industrial environment is a problem faced each time a new manipulator is purchased. Many systems have been designed to automate this positioning. The SMAR system computes the best Cartesian location of the manipulator base to maximize a set of objectives on the end-effector under distance and joint limit restrictions [30]; however, this is only a three dimensional optimization problem as they only consider planar Cartesian position and rotation about the vertical axis. In [31], they solve a similar problem by splitting the large multi-variable optimization problem into several single variable optimization problems using distributed solving. In [32] and [33] they define the envelope of a manipulator's workspace by a series of closed-form equations of surface patches, and use a cost function that places the target points within that envelope. Both of these methods are formulated for a single manipulator with specific points, and do not take into consideration the manipulability of the manipulator at those points. They only ensure that the target points are within the manipulator's workspace envelope.

In [34], they optimize the location of a work piece relative to a robot arm. They discuss an optimization which minimizes the power consumption and maximizes the stiffness of the robotic manipulator. Vosniakos and Matsas optimize the pose of a robot relative to the milling piece for a milling robot by running two genetic algorithms where the first maximizes manipulability along the milling path, while the second uses torque adequacy to search for the optimal initial position of the milling tool [35]. This is done for a six DoF robot, but the last 3 joints are held constant, so it is essentially a 3 DoF robot. Furthermore this only optimizes over the first point in the milling trajectory, i.e. finds the optimum initial position of the end-effector resulting in maximum manipulability and minimum joint torques rather than considering the whole trajectory.

Hassan et al. discuss the problem of finding a single base placement for autonomous industrial robots (AIR) required to cover a small object [36] in an operation like sand blasting. They discretize the possible base locations here in a two dimensional grid on the ground, and find the best location. They extended their work in [37] enabling several base placements for AIR to cover large objects while having constraints on space between robots. Again this is a planar base placement optimization as the AIRs they use are wheeled robots.

Others have used similar approaches to optimize the placement of welding robots. Dharmawan et al. used expanded Lagrangian homotopy to optimize the base placement of a mobile welding robot [38]. Dharwanan also discusses base placement of a mobile welding robot using an inverse kinematics approach in [39]. Both of these optimizations are for a single robot in three dimensions. Two other joint space trajectory generation tools also capable of local three DoF base placement optimization are [40] and [41].

Olaru et al. uses the iterative pseudo-inverse Jacobian matrix method combining a sigmoid bipolar hyperbolic tangent neural network with time delay and recurrent links to optimize an industrial robot's base position relative to two objective functions maximizing the precision while minimizing the movement time [42].

Another application where robotic manipulator base placement is important is robotic surgery. In [43] they discuss an optimization determining the best port placement and arm configuration using a da Vinci Surgical System for an endoscopic cardiac surgery. In this case, they use a global conditioning index to avoid singularities as their main optimization criteria. In [44], they discuss maximizing the dexterous workspace of a Raven IV surgical robot system which consist of two, mirror image pairs of surgical robotic arms. They consider the orientation of each arm, but not the position. This was for specific port locations. They discuss the position optimization as future work.

Mitsi et al. use an evolutionary optimization to find the optimum base placement and joint configurations of a robot that maximizes the manipulability throughout the trajectory [45]. Their objective function takes into account deviations between actual and desired end-effector poses as well as the manipulability measure. The design variables they solve for are the DH parameters of the robot base as well as the joint angles corresponding to each desired pose. So, for one desired tool pose, there are 10 design variables (six joint angles, and four parameters describing the base location). This constrains the base to a DH parameterization and is not completely free to move in all six DoF. Mitsi et al. use a genetic algorithm initially, and then feed this result into a gradient based opitimization for refinement. This does not scale well as an optimization for one desired end effector pose solved in 5 seconds while 9 desired end effector poses took 32 minutes to solve (since the optimization needed to find 58 design variables). This time would increase significantly more with more arms than just one (which is not discussed) or as the number of the path points increases.

Sotiropoulos et al. discuss where to dock an unmanned underwater vehicle (UUV) with a six DoF manipulator [46]. They maximize an area manipulability where the area is defined by a square on a flat docking surface, and the area manipulability is the minimum manipulability of the manipulabilities at each corner. They mount the UUV in the position that maximizes the area manipulability while minimizing the distance between the UUV's current location. This is a four DoF optimization (Cartesian space and rotation about the vertical axis) and is formulated for 1 arm.

Another base placement optimization application is where to place painting robots on a rail system as discussed in [47] and [48]. They discretize the possible base positions, then employ a two-step algorithm to first find the feasible base positions, then evaluate the feasible base positions relative to an end-effector velocity criteria to select the best one. Their algorithm is focused on maintaining a stable end-effector velocity which is important in painting. It is very specific to the painting application and restricts the feasible positions to a cone above each trajectory point. Also, in this case the base can only translate.

None of these previous methods address optimizing the base placement of multiple robots for a cooperative manipulation task. Our algorithm is unique in this way. Additionally, our optimization is general enough to optimize the base placement of any number of robot arms in up to six DoF per arm.

4.2 Methods

4.2.1 Forward Kinematics Binning

Much of the previous work in section 4.1 utilizes inverse kinematics calculations during the optimization. These calculations (which can be computationally expensive) are made during each iteration of the various optimizations. Instead, we compute and store a discretized forward kinematics workspace for each robot arm once, and use this throughout the optimization. This discretization is discussed in this section and summarized in Algorithm 4.

We first define an estimated maximum and minimum x, y, and z position of the end effector relative to each arm's base frame. Then, we select position (of the end effector) and orientation (axis-angle representation of the end effector) bin sizes (d_x, d_y, d_z, d_{ax}, d_{ay}, and d_{az}). We also select the step size to move each joint of the manipulator through (Δq). This discretization is represented in Figure 4.1 below. Each position bin (which is shown on the left in Figure 4.1) contains all of the axis-angle orientation bins with a maximum and minimum of π and $-\pi$. Table 4.1 contains the discretization parameters used for our experiments (in simulation in Section 4.3 and with hardware in Section 5.3) where all angles are listed in degrees while the Cartesian parameters are in meters.

We then step (by Δq) between the joint limits of each joint, and use forward kinematics to compute the homogeneous transformation from the end effector to the robot base frame (T_e^a).

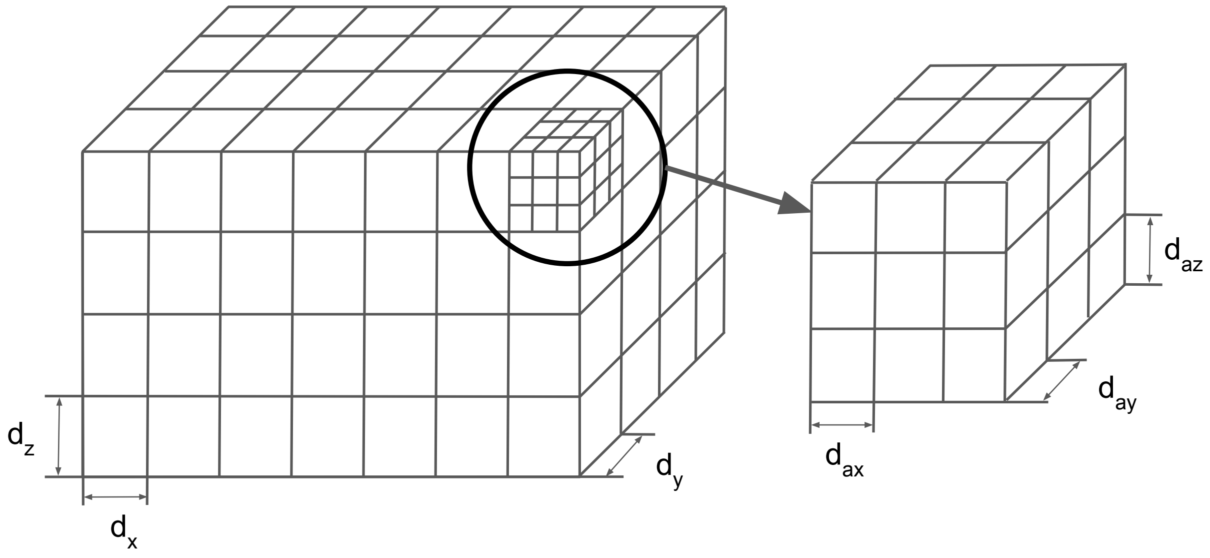


Figure 4.1: We discretize the pose of each manipulator's end effector using forward kinematics and the structure shown here.

Table 4.1: These are the forward kinematic discretization parameters we used.

Robot	x_{max}	y_{max}	z_{max}	x_{min}	y_{min}	z_{min}	d_x	d_y	d_z	d_{ax}	d_{ay}	d_{az}	Δq
Kaa	2.0	1.5	1.4	-0.4	-1.5	-1.4	0.1	0.1	0.1	30	30	30	10
Baxter	1.1	1.1	1.3	-0.8	-1.1	-0.7	0.1	0.1	0.1	30	30	30	10

This transform is converted to its x, y, and z components and the axis-angle representation of its orientation where the angles are all wrapped between $-\pi$ and π. This wrapping is very important for the discretization as it ensures that only one discretized location per end effector orientation exists. We then find the position bin closest to the end effector position, and within this position bin, find the orientation bin closest to the end effector orientation. To find the closest bin, we used the bisection method (as discussed in [49]) on a sorted list of the bins as we found this method to be the most computationally efficient. It is worth noting that because of the natural compliance of soft robot arms, a forward kinematics discretization is acceptable. This is because using a forward kinematics discretization only ensures that the robot reached within the discretization bin sizes of the desired poses. In applications where exact precision is required, it may be worth the time it takes to run the computations (inverse kinematics) during the optimization and avoid discretization.

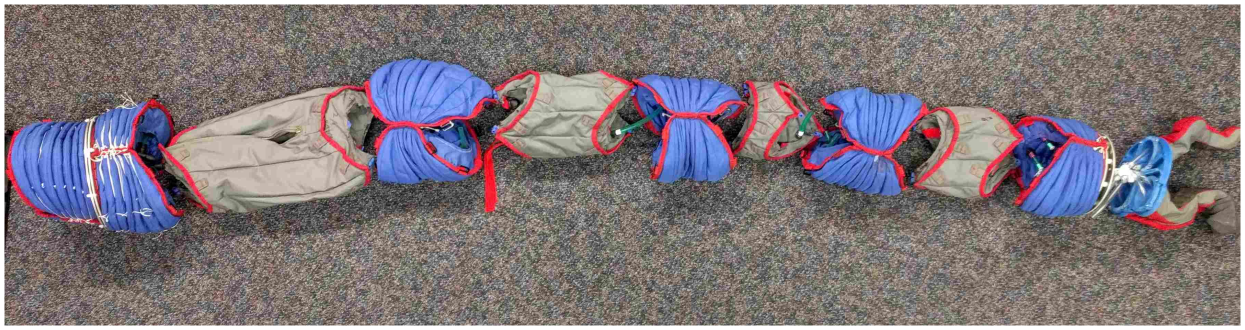


Figure 4.2: We disassembled Kaa to weigh each joint and link for mass and center of mass calculations.

Once we have found the correct discretized location for the end effector pose at a certain joint angle configuration, we compute the Yoshikawa manipulability ([50]) of the arm at the arm's current configuration. The Yoshikawa manipulability is a scalar measure of how far a manipulator is from a singular configuration. Singularities are manipulator configurations where small task space velocities may result in large joint space velocities, and the robot in general loses a degree of freedom in task space. The Yoshikawa manipulability measure is always greater than or equal to zero; the further from zero the measure is, the more capable of uniform motion in all directions the manipulator is. Eq. 4.1 shows how we compute the Yoshikawa manipulability (w) where J is the geometric Jacobian of the manipulator's end effector expressed in the base frame as a function of q which is the current joint configuration.

$$w = \sqrt{\det(J(q)J^T(q)} \tag{4.1}$$

After computing the manipulability of the manipulator, we compute the maximum, static, vertical load bearing capacity at the arm's current configuration. In order to do this, we needed to know the maximum joint torques (τ_{max}) as well as a model of the arm's weight distribution to calculate the gravity compensation torques at each configuration. The two robots used for this work were Kaa and Baxter (as they were the only robots capable of six DoF task space control). We already had an accurate model of a Baxter arm from the robot manufacturer. To create a model of Kaa, we disassembled the robot as shown in Figure 4.2 below and weighed each individual link and joint.

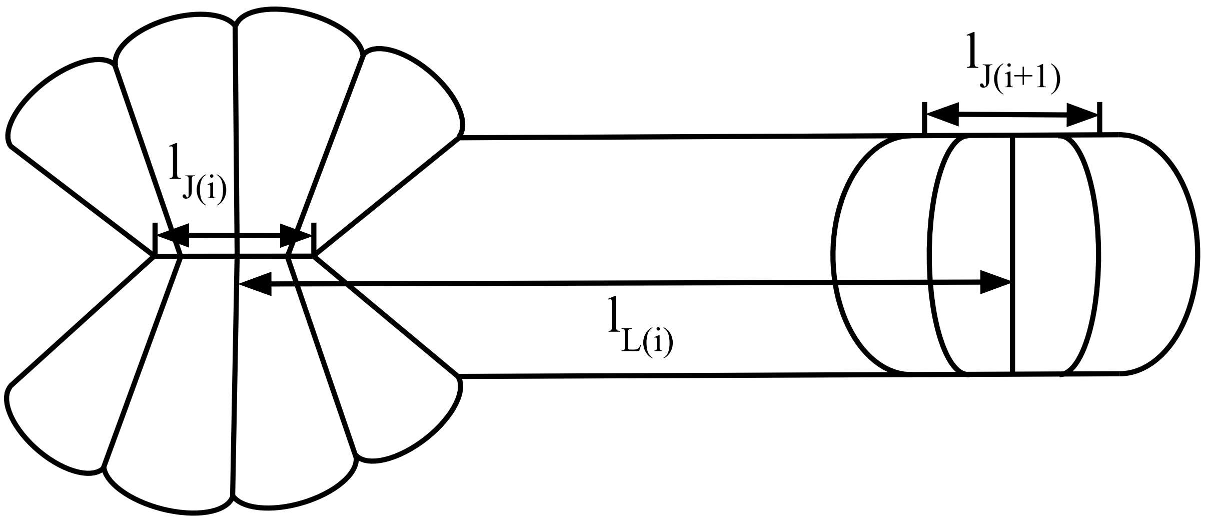


Figure 4.3: One of Kaa's links with a joint on either side is shown above.

We also accounted for the tubing that delivers air to the links and actuators by multiplying the mass per unit length of the tubing by the length of tubing running through each link. We treated each link as a uniform hollow cylinder with the center of mass at the center and included half the mass of the joints on either side of each link in the center of mass calculation. The center of mass calculation for one segment is shown below in Figure 4.3 and Eqs. 4.2 and 4.3 for reference. $l_{L(i)}$ is the length of link i, while $l_{J(i)}$ and $l_{J(i+1)}$ are the lengths of joint i and joint i+1; $m_{J(i)}$, $m_{L(i)}$, $m_{J(i+1)}$, and m_{hose} are the masses of joint i, link i, joint i+1, and the hoses. The mass of the hoses is computed by multiplying the linear density of the hose (ρ) by the number of hoses in the link (n) and the length of the link.

$$p_{com(i)} = \frac{\frac{m_{J(i)}}{2}\frac{l_{J(i)}}{4} + (m_{L(i)} + m_{hose})\frac{l_{L(i)}}{2} + \frac{m_{J(i+1)}}{2}(l_{L(i)} - \frac{l_{J(i+1)}}{4})}{\frac{m_{J(i)}}{2} + m_{L(i)} + m_{hose} + \frac{m_{J(i+1)}}{2}} \tag{4.2}$$

$$m_{hose} = n\rho w_{L(i)} \tag{4.3}$$

In order to calculate the maximum static vertical load bearing capacity at the arm's current configuration we first calculate the gravity torques (τ_g) required to hold the arm static at the current configuration (using a traditional robot model with rigid links and rotary joints). There are many robotics libraries that will do this given a model of the manipulator. We used a library from

Pneubotics (the company that made our soft robots). Next, we find the joint torques (τ_1) required to lift one newton (F_1), by using Eq. 4.4.

$$\tau_1 = J^T F_1 \tag{4.4}$$

Finally, we compute the maximum static vertical load bearing capacity by first subtracting the absolute value of the gravity torques (we use the absolute value in case there are negative gravity torques) from the max torques and dividing by the torques required to support one newton, and then taking the minimum of the resulting vector as shown in Eq. 4.5. The symbol $\oslash$ denotes element-wise division (Hadamard division).

$$F_{max} = \min[(\tau_{max} - \left|\tau_g\right|) \oslash |\tau_1|] \tag{4.5}$$

As a simple example, assume a six DoF robot has $\tau_{max} = [70, 70, 50, 50, 30, 30]Nm$ and in a certain configuration, the gravity torques are $\tau_g = [-40, 60, -5, 10, -5, 1]Nm$. We compute the joint torques required to lift one newton using Eq. 4.4 as $\tau_1 = [4, 1, -0.5, 1, -2, 1]Nm$. We then use Eq. 4.5 to find $F_{max} = \min([7.5, 10, 90, 40, 12.5, 29]) = 7.5N$.

We then store the maximum load bearing capacity F_{max} and the Yoshikawa manipulability w in the computed discretized bin and repeat this process stepping between the joint limits by Δq. This is all summarized in Algorithm 4 below.

4.2.2 Objective Function

In this section, we discuss the objective function we maximize to find the optimal base locations for each arm in a multi-arm manipulation task.

The user defines the desired object path or workspace as a list of homogeneous transforms from the object frame to the world frame (T_o^w). To define a workspace using homogeneous transforms, we select a discretized grid that spans the workspace and define homogeneous transforms at each point on the grid. The user also defines the object mass (m), and the grasping location for each arm relative to the object frame $T_{e_i}^o$. Finally, the user also defines which degrees of freedom the optimization can manipulate for the base of each robot arm.

Algorithm 4 Manipulator Forward Kinematics Discretization

```
 1: for Each robot manipulator do
 2:     procedure DISCRETIZATION BIN CREATION
 3:         Determine the extremes of the manipulator workspace relative to the base frame ($x_{min}$, $y_{min}$, $z_{min}$, $x_{max}$, $y_{max}$, and $z_{max}$)
 4:         Choose a Cartesian discretization size ($d_x$, $d_y$, and $d_z$)
 5:         Choose an axis-angle discretization size ($d_{ax}$, $d_{ay}$, and $d_{az}$)
 6:         Create a forward kinematics object where each Cartesian bin contains the orientation bins as shown in Figure 4.1. We used a python dictionary where the keys were the center of each bin.
 7:     procedure DISCRETIZATION BIN POPULATION
 8:         Select a joint angle step size to move each joint through ($\Delta q$)
 9:         Determine the joint limits of the manipulator ($q_{min}$ and $q_{max}$)
10:         for $q = q_{min}$ to $q_{max}$ by $\Delta q$ (nested for loop m levels deep where m is the number of joints) do
11:             Calculate the forward kinematics given the joint angles to get the current pose of the end effector $T_e^b$
12:             Find the discretization bin closest to the end effector pose ($x$, $y$, $z$, $ax$, $ay$, $az$)
13:             Calculate the geometric Jacobian ($J$) of the manipulator's end effector in the base frame.
14:             Calculate the Yoshikawa manipulability measure $w = \sqrt{\det(J(q)J^T(q)}$
15:             Calculate the maximum static vertical payload $F_{max} = \min[(\tau_{max} - |\tau_g|) \oslash \tau_1]$ where $\tau_{max}$ is a vector of the joint torque limits, $\tau_g$ is a vector of the gravity compensation torques, and $\tau_1$ is the torque required to lift 1 Newton. $\oslash$ is the symbol for element-wise division.
16:             if the current discretization bin has not been reached or $w_{current} < w_{previous}$ then
17:                 Store $w$ and $F_{max}$ in the current discretization bin fk[$x$][$y$][$z$][$ax$][$ay$][$az$] = [$w$, $F_{max}$] where fk is the forward kinematics discretization object
18:             else This bin has already been reached and has a lower manipulability, so don't store the current one.
```

The design variables of the optimization are the homogeneous transformations between the world frame and the base frames of each manipulator ($T_w^{a_i}$). These transformations are parameterized by XZX Euler angles and an XYZ translation. The following objective function is evaluated for each design (each population of the final version of the genetic algorithm that we used had 3000 designs).

For each desired object location (T_o^w), we compute the end effector position of each arm relative to each arm's base using Eq. 4.6. Figure 4.4 shows these transformations for a two arm manipulation task. For a given design from the optimization, we check each desired end effector

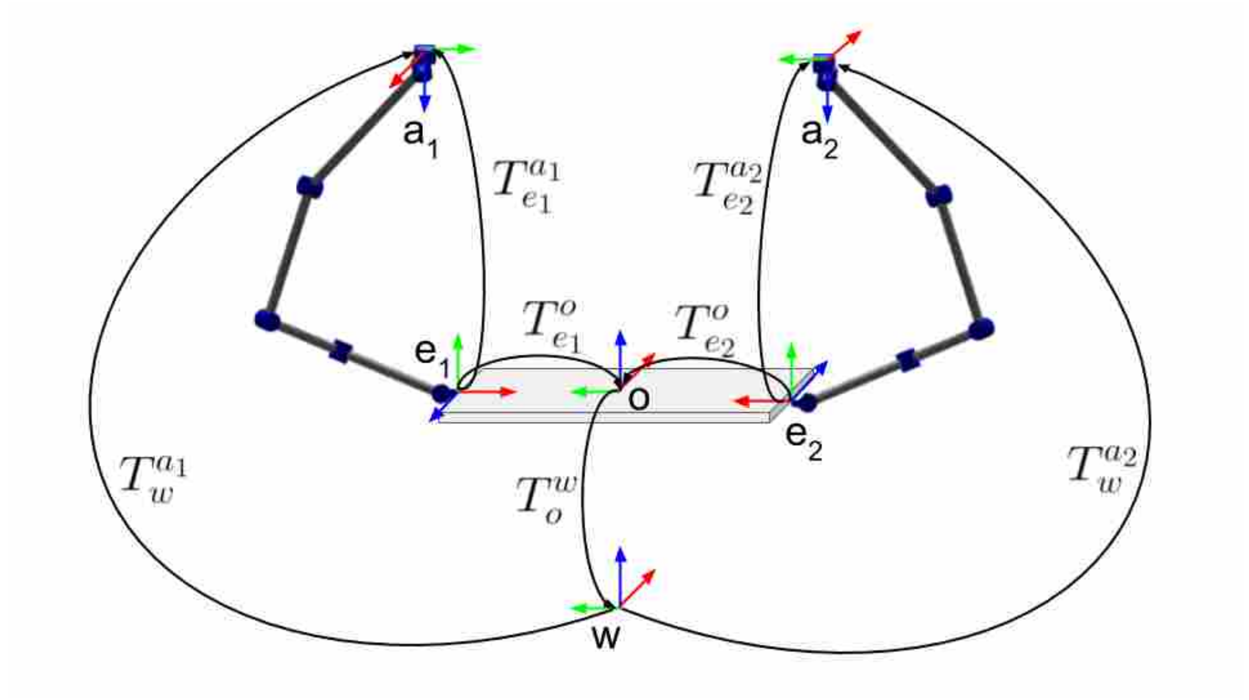


Figure 4.4: The homogeneous transformations used throughout the objective function for a two arm manipulation task are shown here.

location for each arm in order to calculate a total score for the given design ($T_w^{a_i}$ for $i = 1$ to the number of arms).

$$T_{e_i}^{a_i} = T_w^{a_i} T_o^w T_{e_i}^o \tag{4.6}$$

If the transform between the end effector and the base of the arm ($T_{e_i}^{a_i}$) is not within the arm's reachable workspace, we add a severe penalty to the score. Otherwise, we split this homogeneous transform into its axis-angle rotation components and its translation components and find the corresponding discretized forward kinematics bin (x, y, z, ax, ay, az) for the end effector using the bisection method mentioned in Section 4.2.1. If the discretized kinematics bin was reached in the forward kinematics binning, and the weight of the object divided by the number of arms in the optimization is less than the maximum vertical load bearing capacity of the arm in the desired pose, we add the scaled manipulability in the desired bin (fk[x][y][z][ax][ay][az][w]) to the score. We assume that each arm shares the load equally throughout the task. For each arm, we scale the manipulabilities so that each arm in the optimization has the same maximum manipulability. We do this so that each arm is weighted equally in the optimization (assuming that all arms in the optimization are needed to complete the task). If the weight of the object divided by the number of arms is greater than the maximum vertical load bearing capacity of the arm in the desired pose, we add a severe penalty to the score. This penalty was selected to be at least an order of magnitude bigger than the greatest manipulability of the arms used in the optimization so that unreachable positions or positions that are reachable but unable to support the load with the arm base's current configuration are heavily penalized.

If the desired discretized kinematics bin has not been reached, we check the orientation bins immediately surrounding the desired bin. This is because a small amount of orientation error is acceptable for a multi-arm manipulation problem with soft robots due to their natural compliance. We first check the six bins closest to the desired orientation bin (represented by ”1” in Figure 4.5). If any of these bins have been reached and can support the object weight divided by the number of arms, we add the mean of the manipulabilites of the reached bins multiplied by $\frac{3}{4}$ to the score and move the object to the next desired transform. We multiply by $\frac{3}{4}$ to penalize the fact that this wasn't the desired bin, but it is still close enough to increase the score. If none of these six bins have been reached, we repeat this process for the next 12 bins (represented by ”2” in Figure 4.5), but we multiply the mean of these manipulabilities by $\frac{1}{2}$ because they are farther away than the ”1” bins. If none of these 12 bins have been reached, we repeat this process for the final eight bins (represented by ”3” in Figure 4.5), but we multiply the mean of these manipulabilities by $\frac{1}{4}$ because they are farther away than the ”2” bins. Finally, if none of these bins were reached, we add a severe penalty to the score.

This objective function is summarized in Algorithm 5 below. This objective function computes the score for one design, i.e. the homogeneous transformations from the world frame to each manipulator's base frame ($T_w^{a_i}$) where $i = 1$ to n (the number of manipulators in the optimization).

4.2.3 Evolutionary Optimization

The goal of this optimization was to find the optimal base placement for each robotic manipulator in a multi-arm task. We do this by maximizing the objective function described in Section 4.2.2. We initially tried this using a gradient based optimization approach; however, the objective function is comprised of the manipulabilities stored at discretized locations. This means

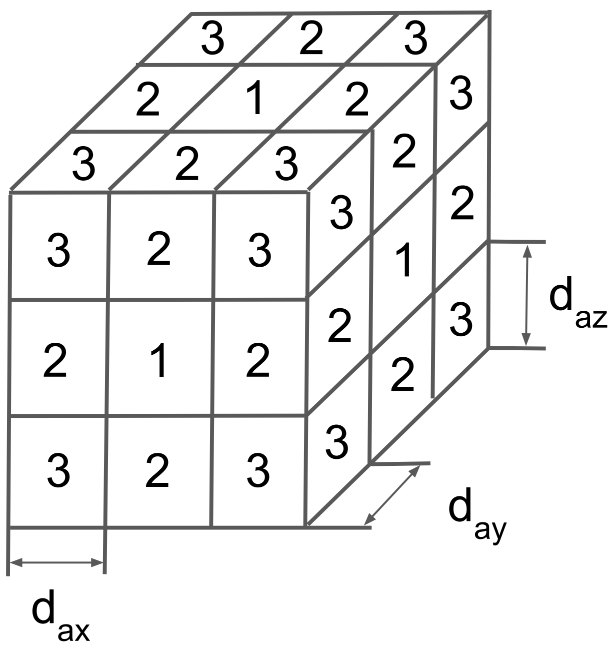


Figure 4.5: The desired orientation bin is in the center of the cube. If it isn't reached, we also search the surrounding bins where the numbers represent each bin's proximity to the desired orientation bin.

that for small changes in base placement, the gradient of the objective function is usually zero. So each time we tried to run the optimization using a gradient-based approach, the optimization would return the base positions that were used to seed the optimization. Because of this, we decided to use a genetic algorithm. Use of a genetic algorithm does not guarantee the global optimum; however, a good solution is found as shown in Figure 4.6 (which shows the optimization score convergence as the generations progress). The maximum scores are above zero, but not far above zero compared to how low below zero the penalties can cause the score to be.

We adapted the genetic algorithm described in [6] for this work. In this genetic algorithm, we first generate an initial population by randomly sampling design variables within the design constraints. The design variables in this work are the position and orientation of the base of each arm (x, y, z in translation and an XZX Euler angle representation of the orientation). Any of these design variables can be fixed as constants if the base of the robot has limited DoF (for example, a wheeled robot would only have three DoF, two in translation and one in rotation). The initial population is randomly paired and used to create a child design in a process called crossover. For

Algorithm 5 Multi-Arm Optimization Objective Function

```
1: Score = 0
2: Penalty = -100
3: for each object location $T_o^w$ do
4:     for each manipulator i = 1 to n do
5:         Calculate the desired end effector pose relative to the manipulator base $T_{e_i}^{a_i} = T_w^{a_i} T_o^w T_{e_i}^o$
6:         if $T_{e_i}^{a_i}$ is outside the manipulator's Cartesian workspace then
7:             Score = Score + penalty
8:         else
9:             Find the discretization bin closest to the desired end effector pose ($x$, $y$, $z$, $ax$, $ay$, $az$)
10:            if fk[$x$][$y$][$z$][$ax$][$ay$][$az$] was reached in the forward kinematics binning then
11:                Find ΔScore by using Procedure 21
12:            else if any of the closest six bins around fk[$x$][$y$][$z$][$ax$][$ay$][$az$] were reached (the "1" bins in Figure 4.5) then
13:                ΔScore = $\frac{3}{4}$ $\times$mean of the reached bins' manipulabilities using Procedure 21
14:            else if any of the next closest 12 bins around fk[$x$][$y$][$z$][$ax$][$ay$][$az$] were reached (the "2" bins in Figure 4.5) then
15:                ΔScore = $\frac{1}{2}$ $\times$mean of the reached bins' manipulabilities using Procedure 21
16:            else if any of the next closest eight bins around fk[$x$][$y$][$z$][$ax$][$ay$][$az$] were reached (the "3" bins in Figure 4.5) then
17:                ΔScore = $\frac{1}{4}$ $\times$mean of the reached bins' manipulabilities using Procedure 21
18:            else
19:                ΔScore = penalty
20:            Score = Score + ΔScore
21: procedure GET MANIPULABILITY IF ARM CAN SUPPORT LOAD
22:     if fk[$x$][$y$][$z$][$ax$][$ay$][$az$][$F_{max}$] $> \frac{\text{object mass}}{n}$ then
23:         Return manipulability (fk[$x$][$y$][$z$][$ax$][$ay$][$az$][$w$])
24:     else
25:         Return penalty
```

this work, crossover was done by randomly assigning each design variable in the child to be equal to one of the parents, or the average of the two parents (with a probability of $\frac{1}{3}$ for each of these options). During crossover, the child design also has a probability of mutation ($\frac{1}{3}$ for this work). In order to introduce more diversity and avoid local minima, the mutation consists of randomly selecting new values for the design variables within the design constraints. After crossover and mutation, the parent and child designs are evaluated using the objective function in Section 4.2.2. The highest scoring design is then passed onto the next generation. We also pass a perturbed copy of the highest scoring design on to the next generation to aid in local refinement. This perturbed

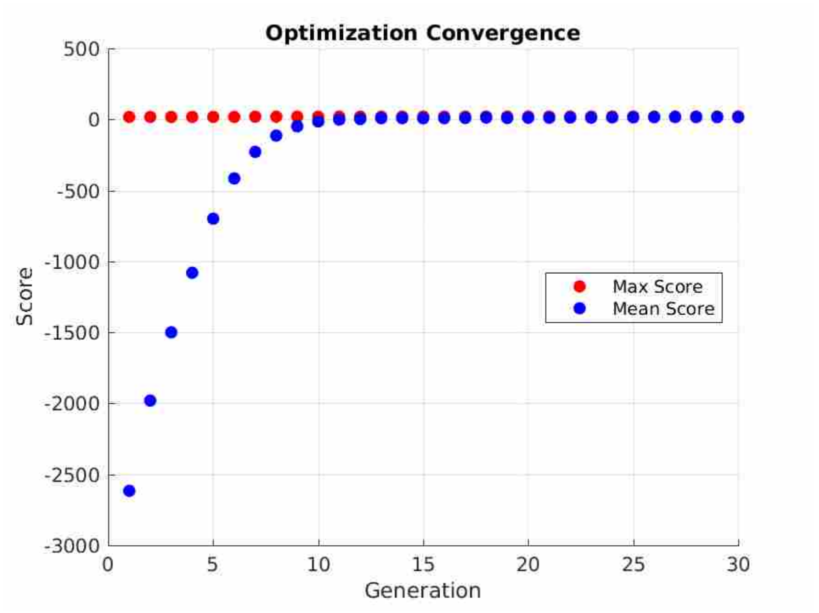


Figure 4.6: The mean and max scores of the designs in each generation as the optimization progresses for experiment 1 are shown here.

copy is created by adding a perturbation amount for each variable. This perturbation amount is randomly selected from a uniform distribution between user defined perturbation bounds. These bounds linearly decrease to half their original amount throughout the optimization in order to get refinement at a finer resolution as the optimization progresses.

The time the optimization takes to run is dependent on how many arms are in the optimization and how many homogeneous transforms define the object path or workspace. Before the optimization starts, it takes about 4.5 minutes to load the binned forward kinematics object for each arm into RAM. This could be stored in RAM to avoid having to load it each time in a situation where base relocation would occur often. It took 2.1 minutes to run the optimization with 19 homogeneous transforms defining the object path and two arms with a total of seven design variables (four design variables for Kaa and three for Baxter). It took 53.7 minutes to run the optimization with 19 homogeneous transforms defining the object path and four arms with a total of 16 design variables (four design variables each for four Kaa arms). It took 61.2 minutes to run the optimization with 1331 homogeneous transforms defining the object workspace and two arms with a total of seven design variables (four design variables for Kaa and three for Baxter). We believe

Table 4.2: The parameters for the experiments run in simulation to validate our base placement optimization are in this table.

Experiment	Path Description	Population Size	Perturbation Sizes	Arms Used[*]
1	0.5 z and 0.4 x	3000	10°, 0.1m	1 Kaa, 1 Baxter
2	0.5 z and 0.4 x	30	10°, 0.1m	1 Kaa, 1 Baxter
3	0.5 z and 0.4 x	3000	60°, 0.5m	1 Kaa, 1 Baxter
4	0.5 z and 0.4 x	3000	10°, 0.1m	4 Kaa
5	0.5 x and rotate −90° z	3000	10°, 0.1m	1 Kaa, 1 Baxter
6	General Workspace	3000	10°, 0.1m	1 Kaa, 1 Baxter

that these times could be reduced with code optimization (including parallelization which fits well with this optimization architecture), and porting the code from python to C.

4.3 Simulation Results

In order to validate this optimization in simulation, we selected several object paths, object sizes, and optimization parameters over which to test. These are summarized in Table 4.2 below. Under the ”Path Description” column, 0.5 z and 0.4 x indicates a movement of 0.5 meters in the z direction and 0.4 meters in the x direction. The ”Population Size” and ”Perturbation Sizes” columns show the optimization parameters used in the genetic algorithm. The perturbation sizes shown are in degrees for the orientation design variables and in meters for the Cartesian design variables.

For each of these experiments, we used the optimization to select the robot base locations, and the joint space path planner described in [40] to find smooth joint paths to follow the desired paths given the base locations. We then discuss how well each arm is able to follow the desired path given the base locations. We do not use any physics simulations with interaction between the robots as a part of this, but rather use kinematic calculations for each arm to check pose error. For all experiments, the data is reported relative to and in terms of the world frame.

Of the robots we had at our disposal, only two (Baxter and Kaa) have six DoF or more and are therefore capable of six DoF end effector pose control. Thus, we use a Kaa arm and a Baxter arm to verify the optimization since our ultimate goal was to implement multi-arm control in hardware. Because Baxter is on a mobile base that is at a fixed height, we optimized the x,y

[*]See Section 2.2 for a description of these platforms.

position and the z orientation of Baxter's base. Another student recently designed a vertical mount for Kaa, so we optimized the x,y,z position and the z orientation of Kaa's base.

Experiments 1, 2, and 3

The first three experiments were used to find the best optimization parameters. For each of these experiments, the desired object path consisted of moving the object up 0.5m in z and forward 0.4m in x while keeping the orientation of the object constant. Figure 4.7 shows the robots executing this path in simulation. For each of these first three experiments, we ran the optimization for the same length of time, and then generated the joint paths using the path planner ([40]) and the optimized base locations. From these joint angles, we compute each robot's end effector path using forward kinematics. For each experiment, we computed the end effector orientation error at each time step by finding the rotation matrix from the actual pose to the desired pose as follows: $R_{e_a}^{e_d} = R_w^{e_d} R_{e_a}^w$, where $R_w^{e_d}$ is the transpose of the rotation matrix representing the desired end effector orientation in the world frame. $R_{e_a}^w$ represents the actual end effector orientation in the world frame. We then converted $R_{e_a}^{e_d}$ into an axis-angle representation, the magnitude of which is the angle between the actual and desired orientation.

We ran experiment 1 using a population size of 3000, and perturbation sizes of 10° and 0.1m. Experiment 2 used a population size of 30 with the same perturbation sizes as experiment 1. Between experiment 1 and 2 we were able to determine whether a bigger or smaller population size was better. After running experiments 1 and 2, we determined that a larger population size was necessary, so we ran experiment 3 using a population size of 3000 but increased the perturbation sizes to 60° and 0.5m in order to determine whether bigger or smaller perturbations performed better.

The final robot base configurations and end effector poses for each experiment are shown in Figures 4.7, 4.8, and 4.9. For all three experiments, Baxter's base position was placed fairly similarly. Experiments 1 and 3 resulted in fairly similar Kaa positions; however, the Kaa base pose for experiment 2 was quite different, and this resulted in Kaa being unable to follow the desired end effector trajectory as shown in Figure 4.8.

As shown in Figures 4.10, 4.11, and 4.12, Baxter was able to track the Cartesian path well for all three experiments.

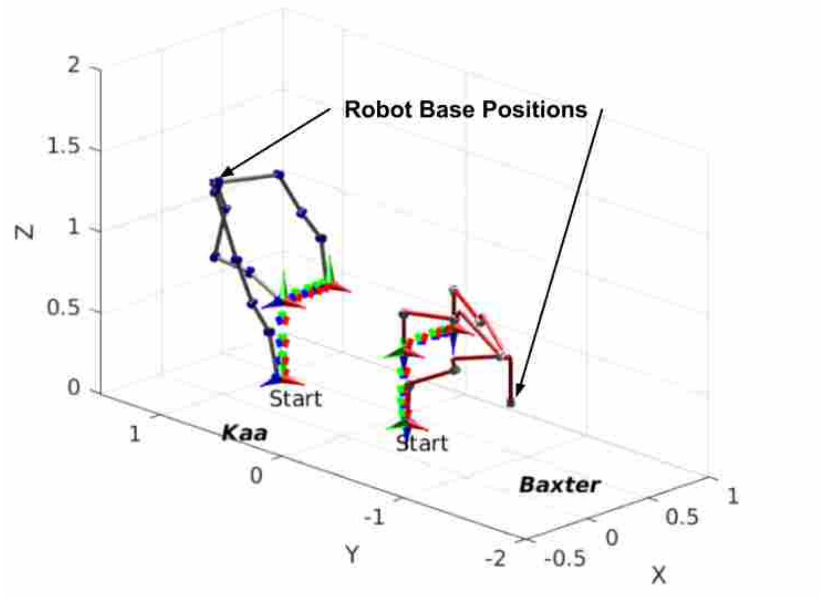


Figure 4.7: The base configurations for Kaa and Baxter found by our optimization are shown here for experiment 1 along with the end effector poses from forward kinematics on the joint angles from the path planner.

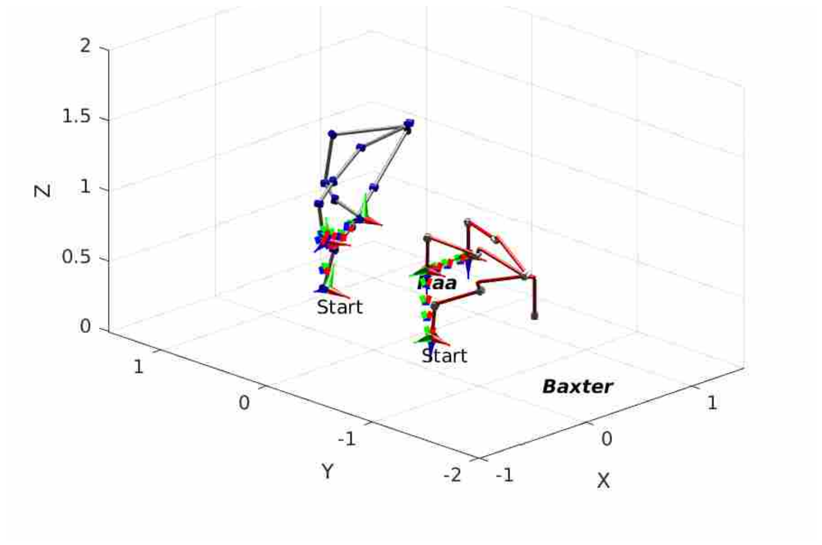


Figure 4.8: The base configurations for Kaa and Baxter found by our optimization are shown here for experiment 2 along with the end effector poses from forward kinematics on the joint angles from the path planner.

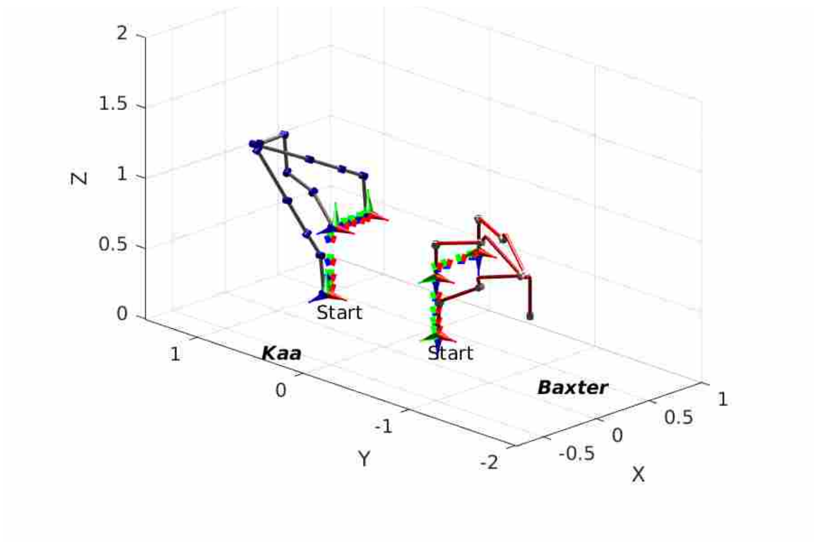


Figure 4.9: The base configurations for Kaa and Baxter found by our optimization are shown here for experiment 3 along with the end effector poses from forward kinematics on the joint angles from the path planner.

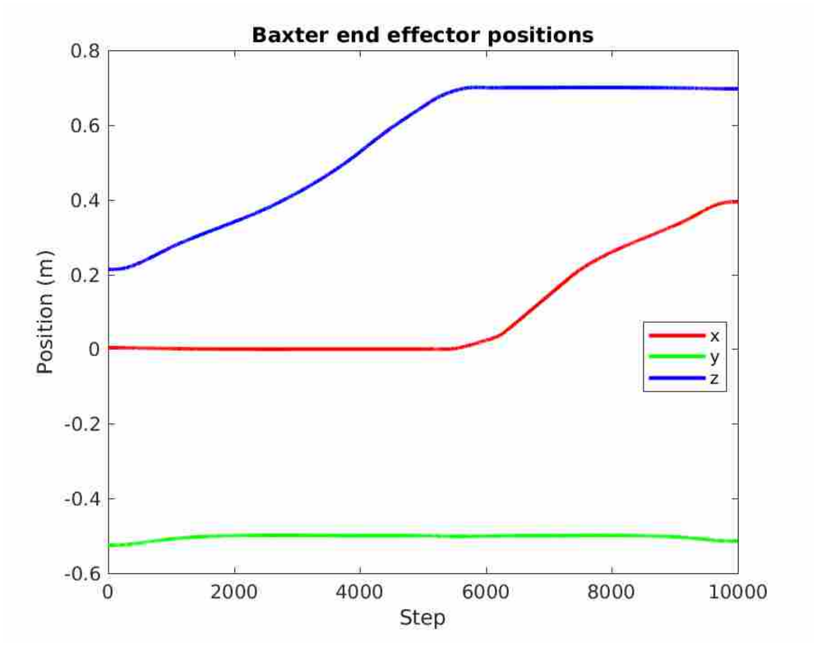


Figure 4.10: The Cartesian end effector position of Baxter using the base configurations found by our optimization and forward kinematics on the joint angles found by the path planner for experiment 1 is shown here.

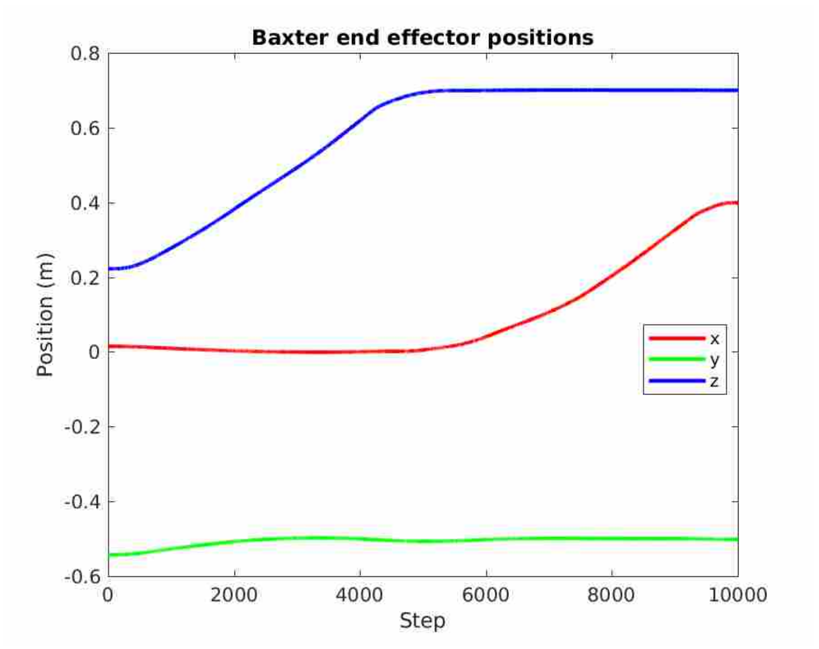


Figure 4.11: The Cartesian end effector position of Baxter using the base configurations found by our optimization and forward kinematics on the joint angles found by the path planner for experiment 2 is shown here.

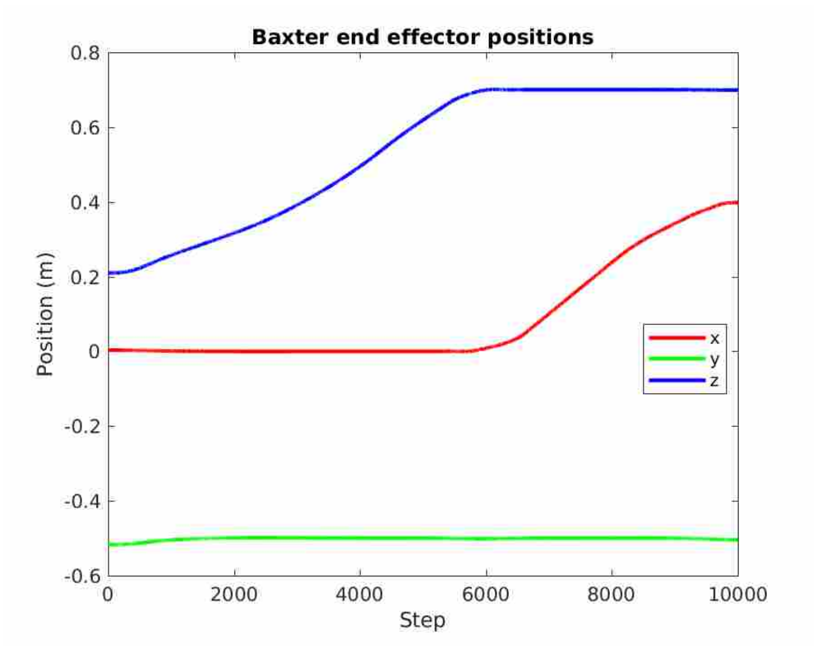


Figure 4.12: The Cartesian end effector position of Baxter using the base configurations found by our optimization and forward kinematics on the joint angles found by the path planner for experiment 3 is shown here.

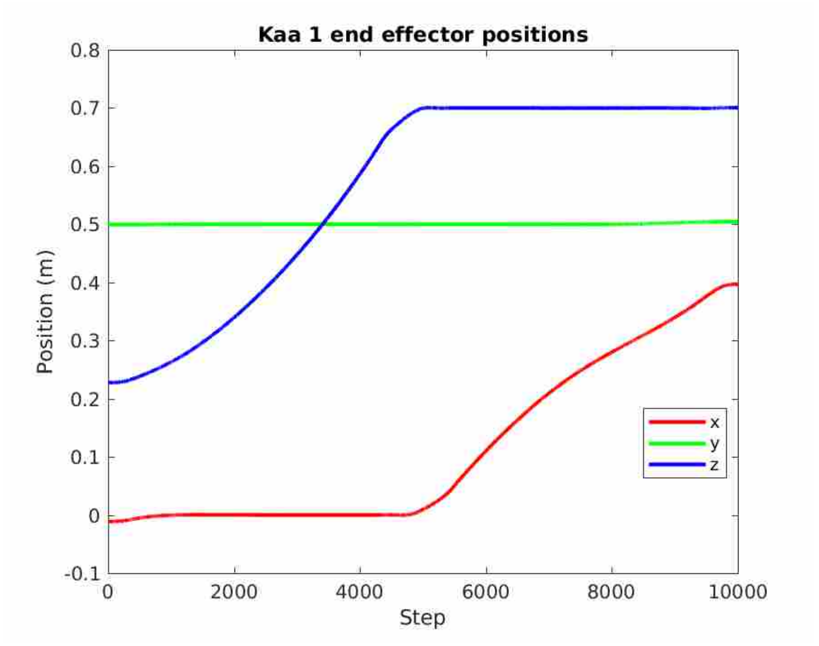


Figure 4.13: The Cartesian end effector position of Kaa using the base configurations found by our optimization and forward kinematics on the joint angles found by the path planner for experiment 1 is shown here.

Figures 4.13, 4.14, and 4.15 show Kaa's Cartesian end effector path for all three experiments. Kaa did not track the path well in experiment 2. While in experiment 3, Kaa was not able to fully reach the full 0.4m in the x direction.

Figures 4.16, 4.17, and 4.18 show the orientation error for Kaa and Baxter along the path for each experiment. Kaa performed very poorly in experiment 2 with upwards of 25° of error. Overall, experiment 1 showed the least orientation error.

Overall, experiment 1 proved the most successful in terms of ability to track the desired Cartesian path with as little end effector orientation error as possible. We believe that this is because a larger population size is able to span the large search space more effectively than a small population size as more local minima are discovered with a larger population. Also, we believe that a larger perturbation size was not able to provide the resolution of local refinement necessary to converge upon a local minimum. For this reason, we selected a population size of 3000 (this was the largest size we could feasibly use with the 32 gigabytes of RAM we had available) and a perturbation size of 10° and 0.1m for the remainder of our experiments. We did try other

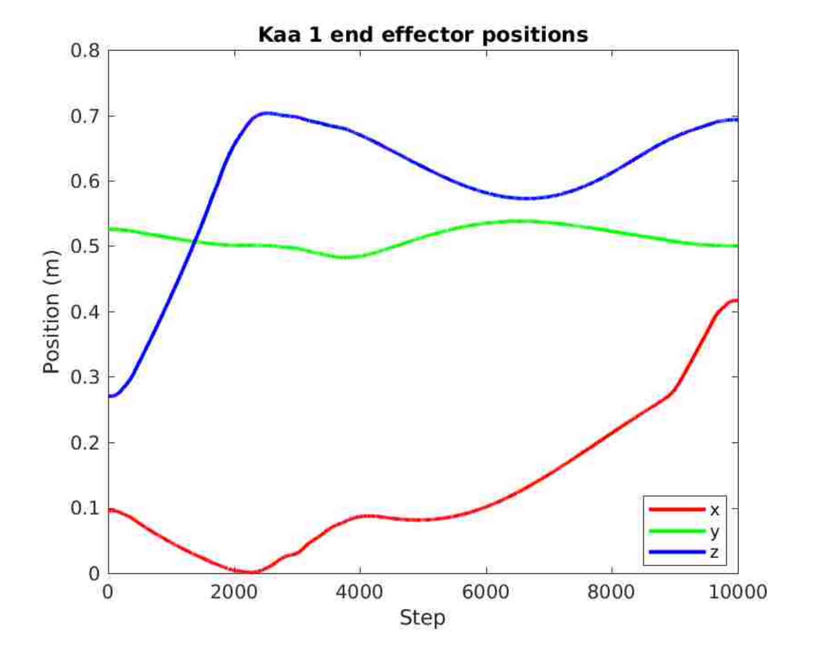


Figure 4.14: The Cartesian end effector position of Kaa using the base configurations found by our optimization and forward kinematics on the joint angles found by the path planner for experiment 2 is shown here.

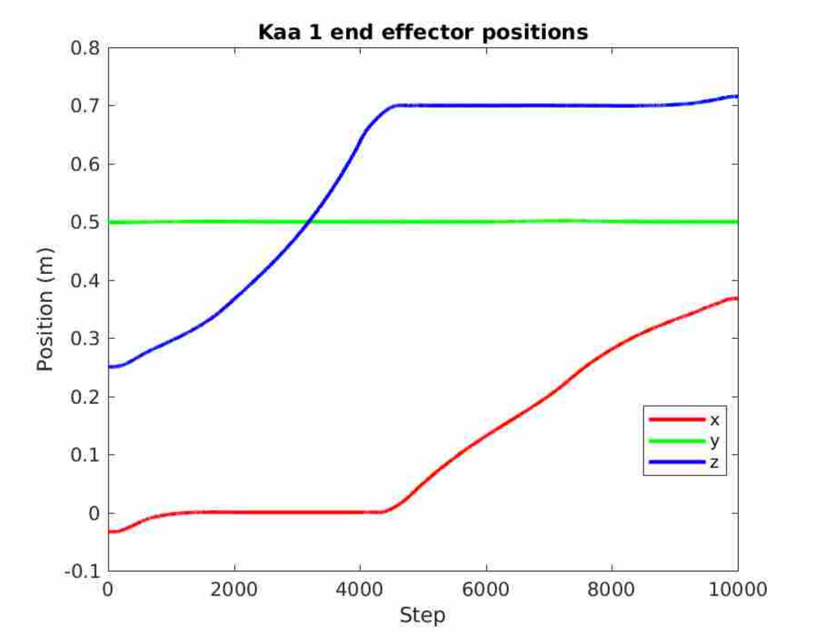


Figure 4.15: The Cartesian end effector position of Kaa using the base configurations found by our optimization and forward kinematics on the joint angles found by the path planner for experiment 3 is shown here.

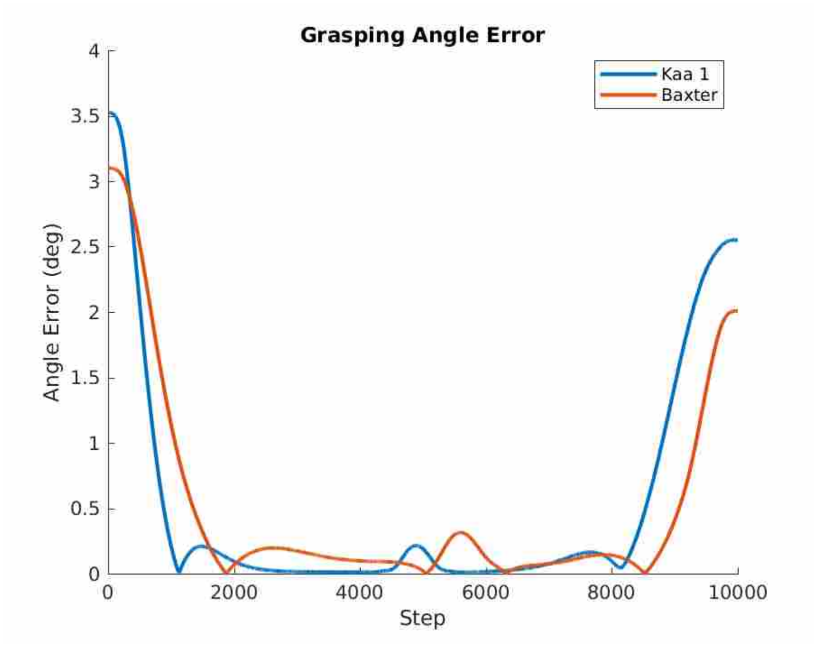


Figure 4.16: The orientation error of both Kaa and Baxter's end effectors along the desired path for experiment 1 is shown above.

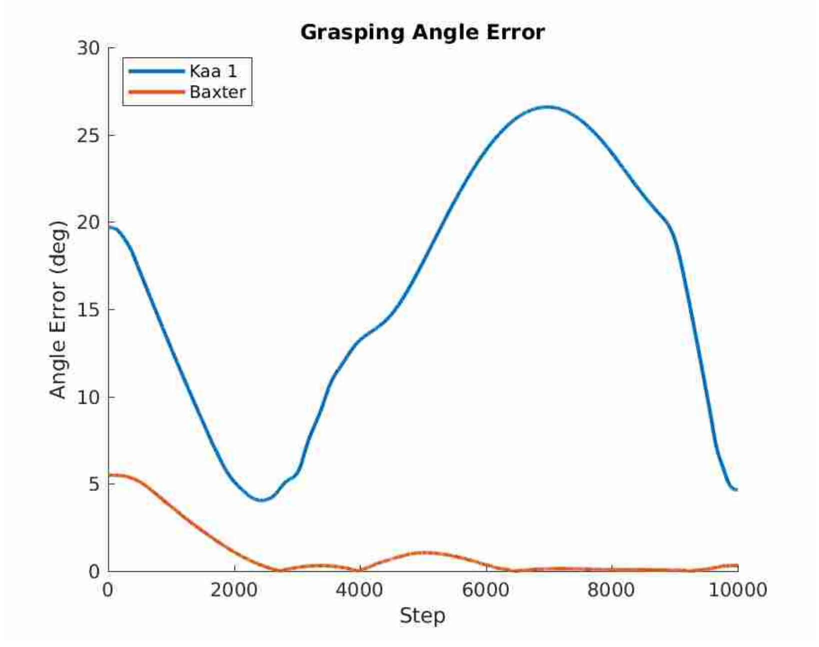


Figure 4.17: The orientation error of both Kaa and Baxter's end effectors along the desired path for experiment 2 is shown above.

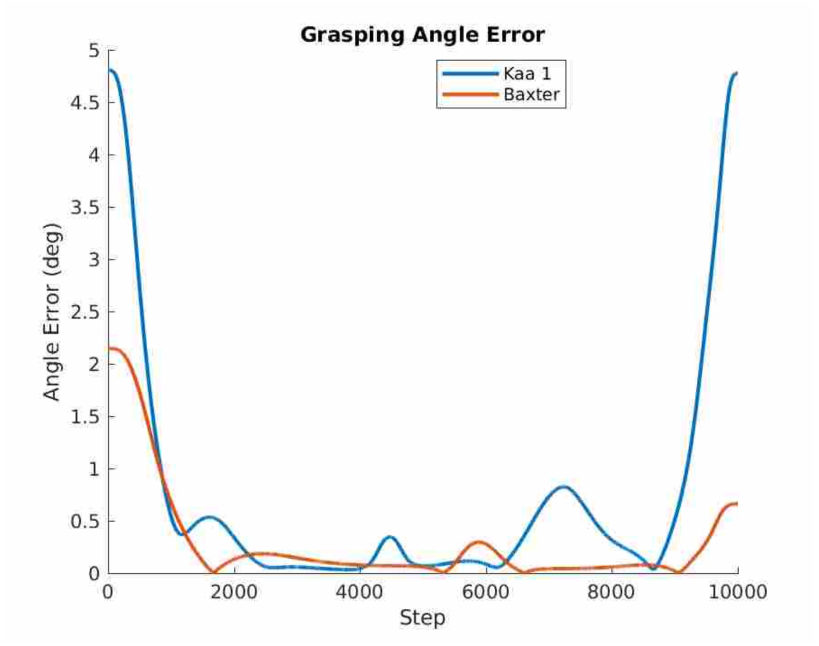


Figure 4.18: The orientation error of both Kaa and Baxter's end effectors along the desired path for experiment 3 is shown above.

population sizes and perturbation sizes than just these two, but ultimately the chosen population and perturbation sizes performed the best.

In order to demonstrate that the optimization was in fact finding good solutions, we perturbed the base positions found in experiment 1 by a small amount in random directions and then ran the path planner and generated the same plots as above. As shown in Figures 4.19 through 4.21, the arms were not able to perform the trajectory nearly as well as by using the solution found by our optimization.

Experiment 4

In order to demonstrate our optimization's applicability to more arms than two, we optimized the base placement of four Kaa arms for the same task as experiments one through three (the desired object path was up 0.5m in z and forward 0.4m in x while keeping the orientation of the object constant). The base placement and end effector path for each Kaa arm is shown in Figure 4.22.

Each robot was able to closely follow the desired path while keeping the orientation error within 6°. Because of the compliance of soft robot arms, this amount of error is acceptable as small orientation deviations do not result in force buildup in the arms or the object being manipulated.

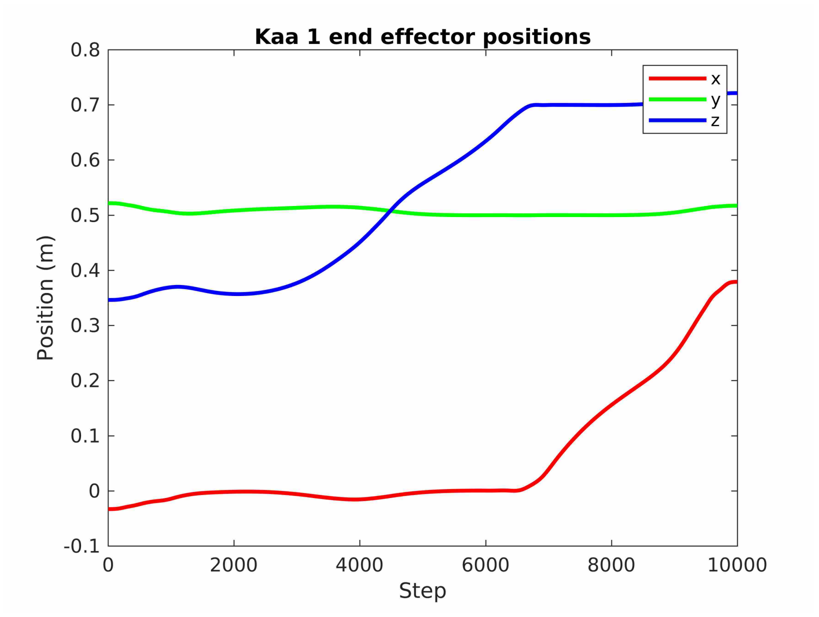


Figure 4.19: The Cartesian end effector position of Kaa using the base configurations found by our optimization perturbed by a small amount and forward kinematics on the joint angles found by the path planner for experiment 1 is shown here.

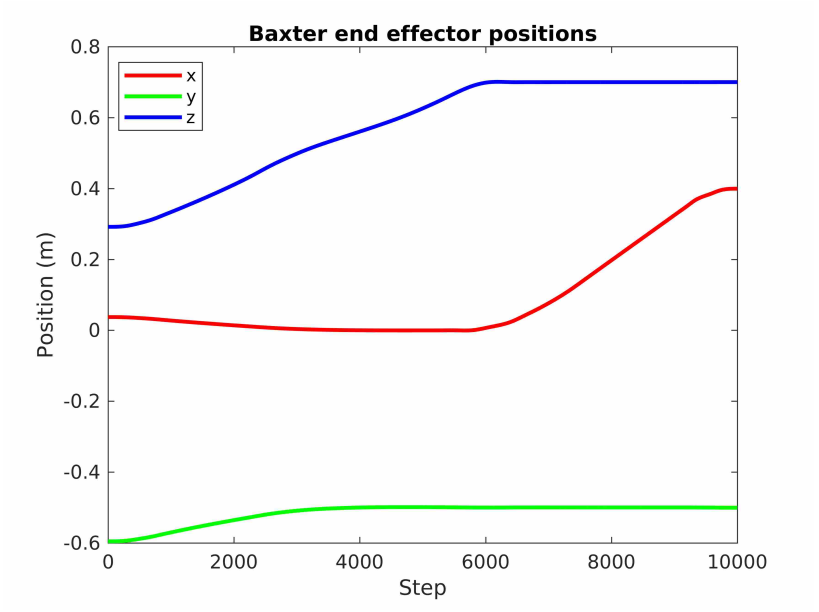


Figure 4.20: The Cartesian end effector position of Baxter using the base configurations found by our optimization perturbed by a small amount and forward kinematics on the joint angles found by the path planner for experiment 1 is shown here.

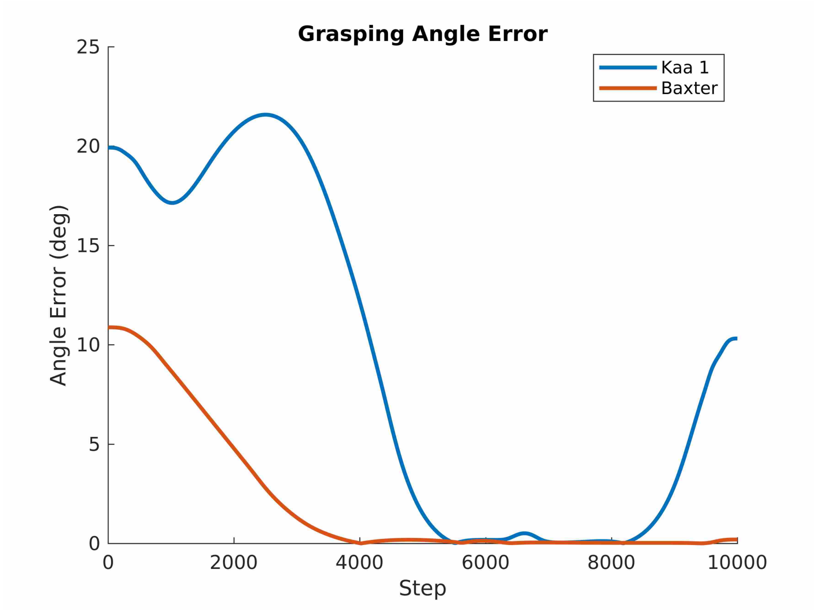


Figure 4.21: The orientation error of both Kaa and Baxter's end effectors along the desired path for the perturbed results of experiment 1 is shown above.

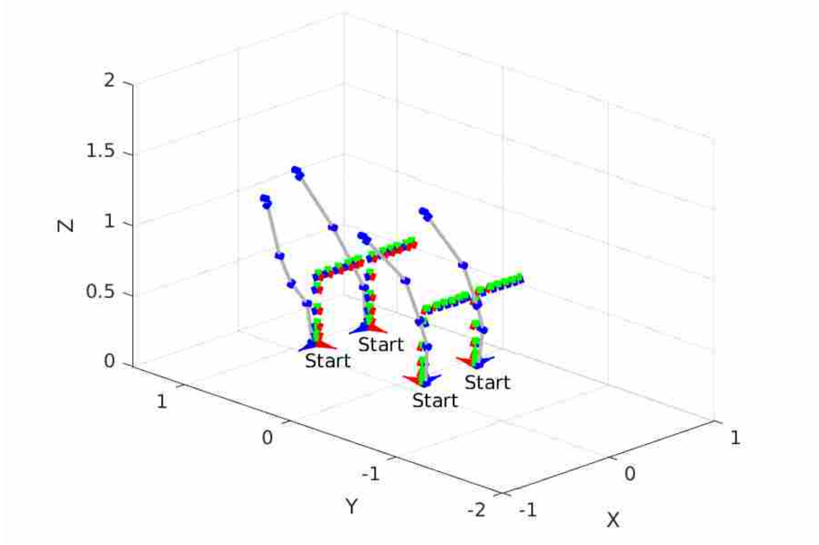


Figure 4.22: This shows 4 Kaa arms with the base configurations found by our optimization as well as the end effector poses from forward kinematics on the joint angles from the path planner for experiment 4.

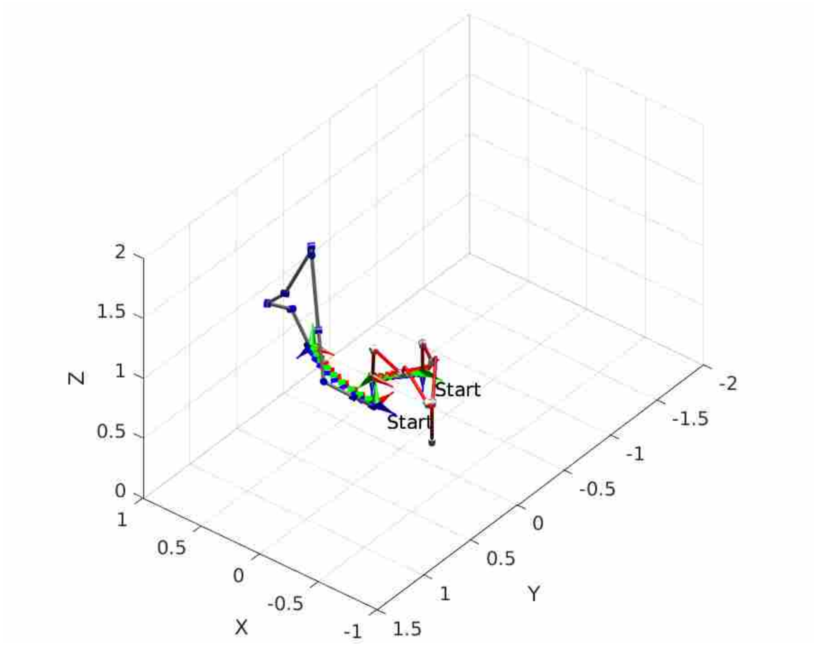


Figure 4.23: The base configurations for Kaa and Baxter found by our optimization are shown here for experiment 5 along with the end effector poses from forward kinematics on the joint angles from the path planner.

Experiment 5

All the previous experiments were run on the same path with no rotation. In order to demonstrate our optimization's ability to work with paths that include simultaneous translation and rotation, we selected an object path that moved forward 0.5m while rotating about the vertical axis $-90°$. Figure 4.23 shows the optimized base placement and end effector paths for Kaa and Baxter for this experiment.

The object length for this experiment was 0.5m. We started the object at 0m in the x direction and moved it to 0.5m while rotating the object $-90°$ about the z axis and maintaining a height in the z direction of 0.5m. This means that the starting x position for Kaa and Baxter's end effectors should both be at 0m while the ending positions should be 0.75m and 0.25m for Kaa and Baxter respectively. Similarly, the starting y positions for Kaa and Baxter's end effectors should be -0.25m and 0.25m while the final y position for both Kaa and Baxter should be at 0m. As shown in Figures 4.25 and 4.24, both robots were able to track these Cartesian trajectories.

Figure 4.26 shows the orientation error for each arm as they follow the desired paths.

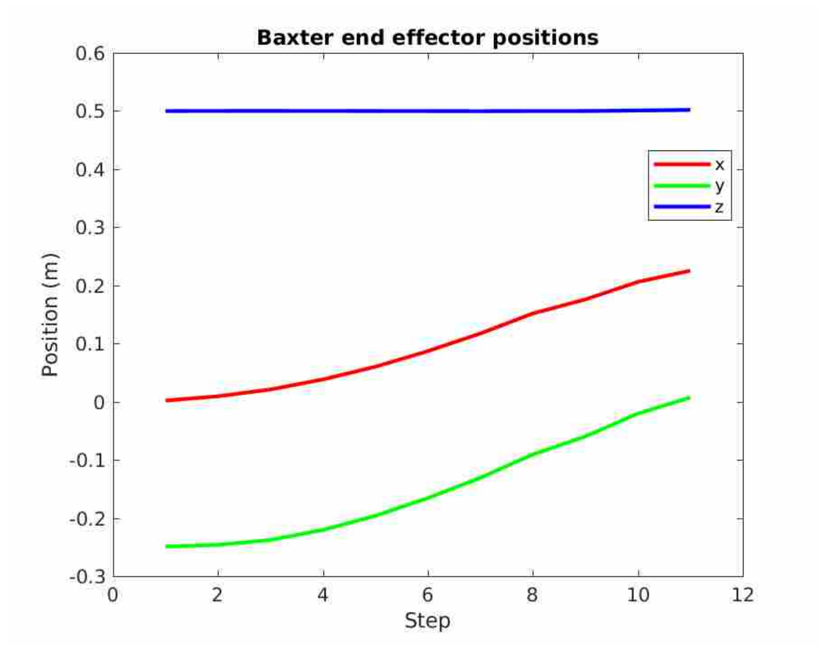


Figure 4.24: The Cartesian end effector position of Baxter using the base configurations found by our optimization and forward kinematics on the joint angles found by the path planner for experiment 5 is shown here.

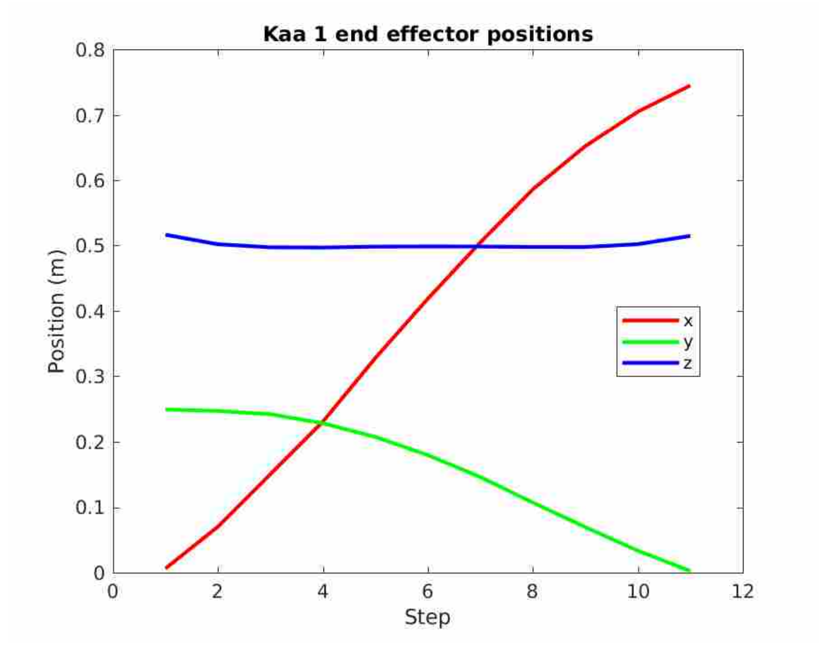


Figure 4.25: The Cartesian end effector position of Kaa using the base configurations found by our optimization and forward kinematics on the joint angles found by the path planner for experiment 5 is shown above.

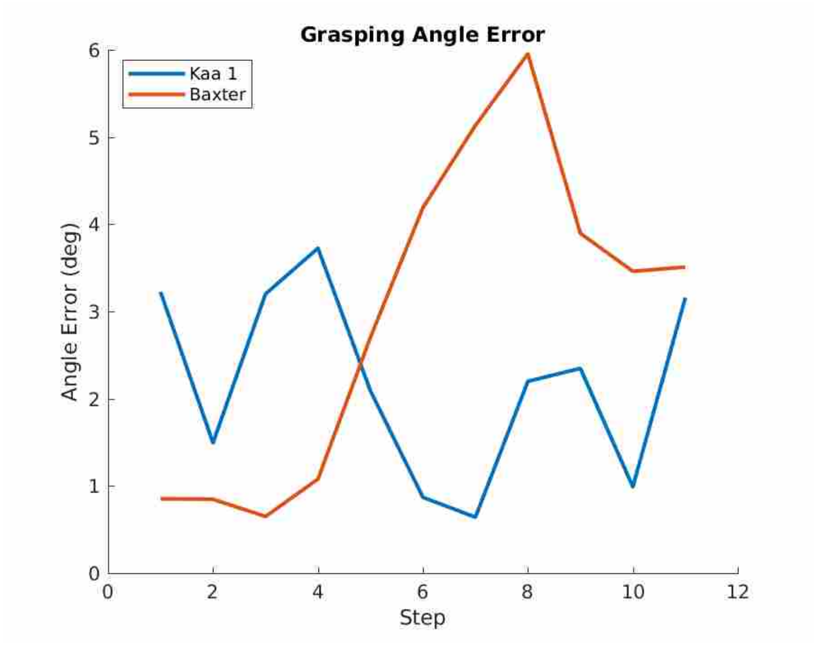


Figure 4.26: This shows the orientation error of both Kaa and Baxter's end effectors along the desired path for experiment 5.

Each manipulator was able to track the desired path with less than 6° of orientation error as shown in Figure 4.26.

Experiment 6

Finally, in order to demonstrate the optimization's applicability to a general workspace, we made a cubic workspace with 0.5m sides and optimized Kaa's and Baxter's base placements for this workspace. We then simulated four paths spanning this workspace (moving to opposite corners) using the optimized base locations. The robot configurations and orientation errors for the path where the robots performed the worst are shown in Figures 4.27 and 4.28. This is the path from the bottom left to the top right of the workspace.

For three of the four paths the robots were able to complete the path without having the orientation error go above 8.5°; however, there was one corner of the workspace that Kaa had a difficult time reaching while maintaining the desired orientation. In this corner, the error on Kaa was 18° as shown in Figure 4.28. This is because Kaa is not completely dexterous at all locations in such a large workspace. Whereas we expect the seven DoF Baxter to more easily reach all the desired orientations as was the case here. Nonetheless, the optimization was able to find a base placement where Kaa worked well in most of the workspace.

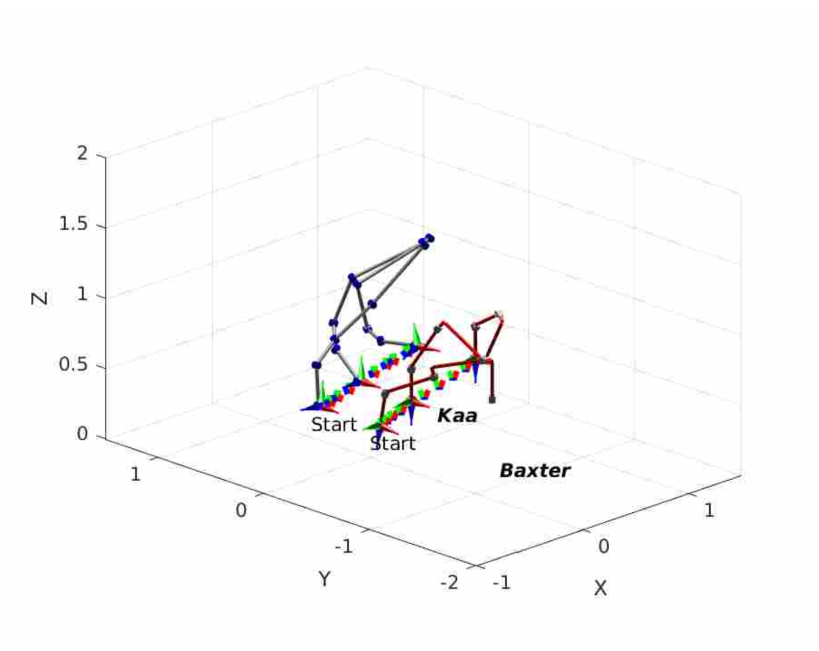


Figure 4.27: The base configurations for Kaa and Baxter found by our optimization are shown here for experiment 6 along with the end effector poses from forward kinematics on the joint angles from the path planner.

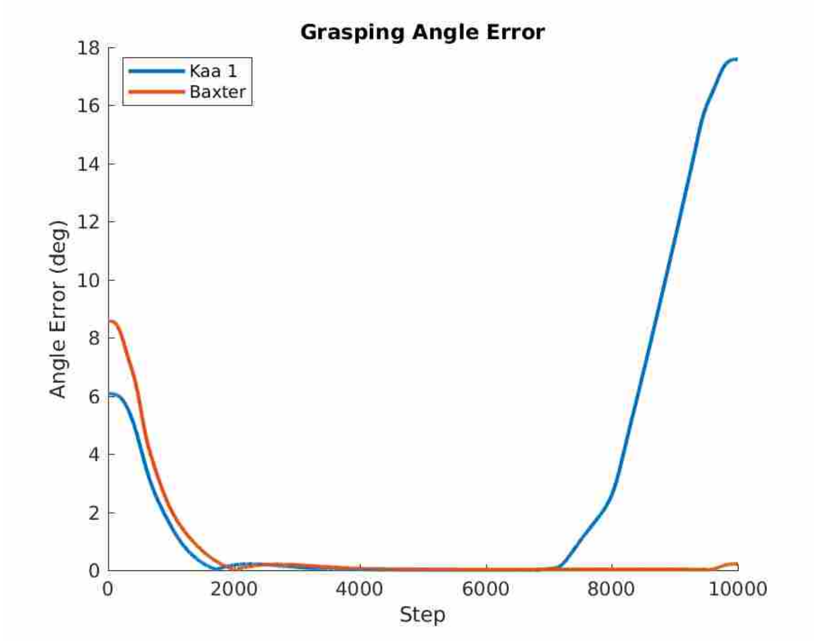


Figure 4.28: The orientation error of both Kaa and Baxter's end effectors along the desired path for experiment 6 is shown above.

4.4 Conclusion

In this chapter, we developed a general optimization that is capable of finding the optimal base locations for any number of arms in a multi-arm manipulation task. We then demonstrated this optimization in several simulated tasks. In the next chapter, we use this optimization to find the base placement for Kaa and Baxter for two real world multi-arm manipulation tasks, and then perform these tasks on the hardware after placing the robots at the optimized locations.

CHAPTER 5. COORDINATED MULTI-ARM MANIPULATION

In this chapter, we discuss our multi-arm manipulation control scheme for soft robots. Then we use the optimization described in Chapter 4 to find the optimal arm base configurations for two tasks using Baxter and Kaa and place the robots at these locations. Finally, we demonstrate our control scheme on hardware and discuss the results.

5.1 Previous Work

Small end effector pose deviations in a multi-arm manipulation task with traditional rigid robots can result in large stresses on the robots and the object being manipulated. To compensate, many researchers have proposed hybrid force/position control schemes which seek to control the position of an object being grasped by several manipulators while either keeping the forces below a certain threshold or maintaining a certain force (see [51–56]). Other researchers model the arms connected to the object as a closed kinematic chain, and use a master-slave force control strategy (see [57]) where the desired trajectory and operational force of the master arm are given in advance and the slave arm's trajectory and operational force are calculated from the closed-chain constraint equations. Other researchers have developed controllers that enforce a controlled impedance of the manipulated object, where the controller compensates for the system dynamics and directly controls the internal forces on the object using force and moment measurements at the robot's wrists (see [58–61]). A challenge with these approaches is the coordination of high bandwidth centralized controllers as any controller latency can result in dangerously high forces on an object being manipulated. Similarly, if the software or hardware malfunctions and the force control stops working, traditional robots could exert a dangerous amount of force on the object being manipulated, themselves, humans, or other delicate equipment nearby. Although the occurrence of such an event is unlikely, the risk if it does occur is very high. Additionally, transporting traditional heavy robots to many locations where multi-arm manipulation may be useful, such as search and rescue sites, may be expensive or impossible depending on available transportation infrastructure.

An alternative and novel approach is to mitigate buildup of high forces by using a robot with flexible links and passive compliance in the joints. Because soft robots are inherently compliant, deviations in object position result in significantly lower buildup of forces; thus, they lend themselves nicely to tasks involving several arms. Even tasks with one rigid arm and multiple soft arms become simpler as the whole system is forgiving of end effector deviations due to compliance of the soft arms. Therefore, one of the major concerns with successful implementation of coordinated, multi-arm manipulation is eliminated.

Additionally, because these robots are soft and inherently compliant, they are inherently safer around humans and delicate equipment even if something malfunctions. They are also much lighter and smaller than traditional robots, and therefore much easier to transport to a disaster scenario for example. On the other hand, compliant links and joints introduce new challenges (addressed below) into the control paradigm, and currently do not perform as well (with regards to metrics like rise time, repeatability, accuracy, and overshoot) as state-of-the-art torque-controlled robots like Robonaut 2.

Researchers have begun to explore the design and control of soft-bodied robots composed of compliant materials (see [62] for recent developments in the field of soft robotics). C. Laschi et al. have done work with a multi-arm soft robot to mimic crawling behavior of an octopus (see [63, 64]). Others have done research into soft robot design for different types of locomotion (see [65], [66]). However, to our knowledge, no research on multi-arm manipulation with soft robots has been done. This chapter is focused on soft robot, coordinated multi-arm manipulation tasks.

5.2 Methods

Because of the inherent compliance of soft robots, we were able to implement a simple control scheme without accounting for the complex force interaction between the robots. Given a desired object trajectory $T_o^w[t]$ and homogeneous transformation between each robot end effector and the object frame T_e^o (i.e. the grasping locations), we can compute the desired trajectory of each robot end effector as shown in Eq. 5.1. T_w^a is the homogeneous transform between the world frame and the arm base frame, and $T_e^a[t]$ is the trajectory of the robot's end effector relative to the robot base frame.

$$T_e^a[t] = T_w^a T_o^w[t] T_e^o \tag{5.1}$$

At each time step (t), we then simultaneously send the desired transform ($T_e^a[t]$) in the trajectory to each robot. The robot motions will not be perfectly coordinated, but because of the soft robot compliance, this is acceptable. This control scheme is not feasible for high precision applications; however, most real world tasks where soft robots would be useful (such as removing rubble or lifting a person in a stretcher to safety) do not require high levels of precision. This control scheme is simple and does not require high bandwidth centralized force control.

5.3 Experimental Results

In order to validate the control scheme in Section 5.2, we used Baxter and Kaa to manipulate a one meter long rigid board through two trajectories. During the multi-arm simulation experiments described in Chapter 4, we assumed that Kaa had joint angle limits of $\pm 90°$ as Kaa was not functional at the time. Once Kaa was functional, we found the the actual joint limits to be $\pm[35, 45, 57, 75, 80, 60]°$. With these new joint limits, Kaa was not able to perform the same tasks as in simulation. We modified the tasks to be achievable, and found new optimal base positions for these tasks. The multi-arm control scheme is closed loop with regard to the end effector of each manipulator, but open loop with respect to the object. For each trajectory, we used the optimization from Chapter 4 to determine the base location of each robot for the multi-arm task. For Kaa, we used the hybrid, servoing controller described in Chapter 3 to follow each trajectory with desired joint angles at each step determined from the path planner in [40]. Because Baxter is a rigid robot with known kinematics and robust joint angle tracking, we used the joint angle controller that is part of the Baxter Software Development Kit to track the desired joint angles (again using the path planner in [40] to find the joint angles that should result in the desired trajectory).

Our experimental procedure for each trajectory was as follows:

1. Compute the optimal base pose for Baxter and Kaa for the given trajectory using the optimization in Chapter 4 and place the robot bases in these positions.

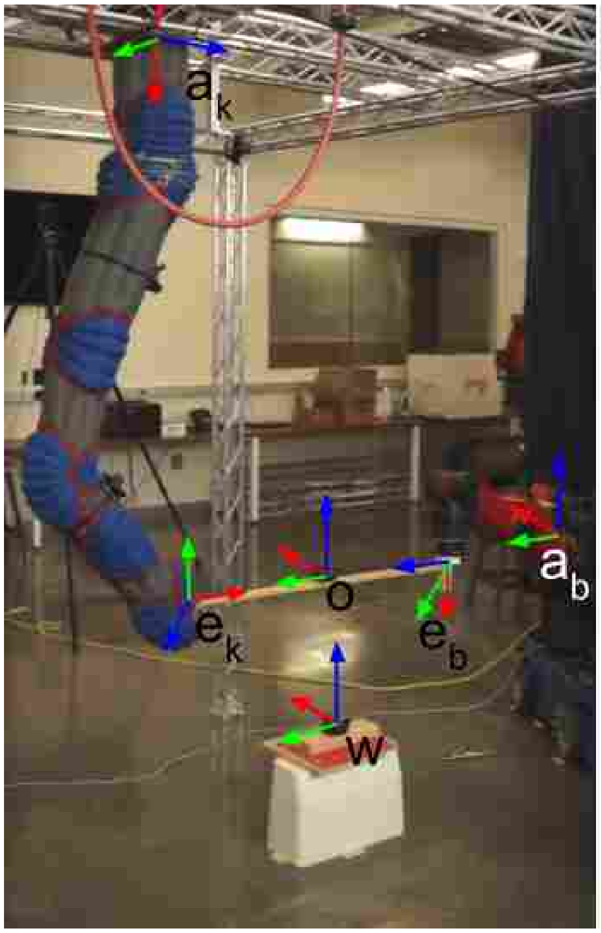


Figure 5.1: This shows the multi-arm setup we used for our experiments.

2. Compute the desired joint angle trajectories for Baxter and Kaa using the path planner in [40]. This takes less than 30 seconds to compute.
3. Command each robot to the initial position in the trajectory.
4. Rigidly connect the board to each robot's end effector.
5. Command the remaining poses in the trajectory while recording the board pose.

This experimental setup is shown below in Figure 5.1. The Vive tracker on the board was used to record the pose of the board throughout the trajectory. It was not used for control of the object or robot trajectories.

The experiments we ran are summarized below in Table 5.1 where the "Trajectory Description" column describes the desired object motion.

Table 5.1: This table contains a list of the multi-arm manipulation experiments we performed.

Experiment	Trajectory Description	Trajectory Duration	Number of Trials
1	0.1m in z and 0.4m in x	30 sec.	10
2	0.1m in z and 0.4m in x	10 sec.	10
3	0.4m in x and 0.4m in y	30 sec.	1
4	0.4m in x and 0.4m in y	10 sec.	1

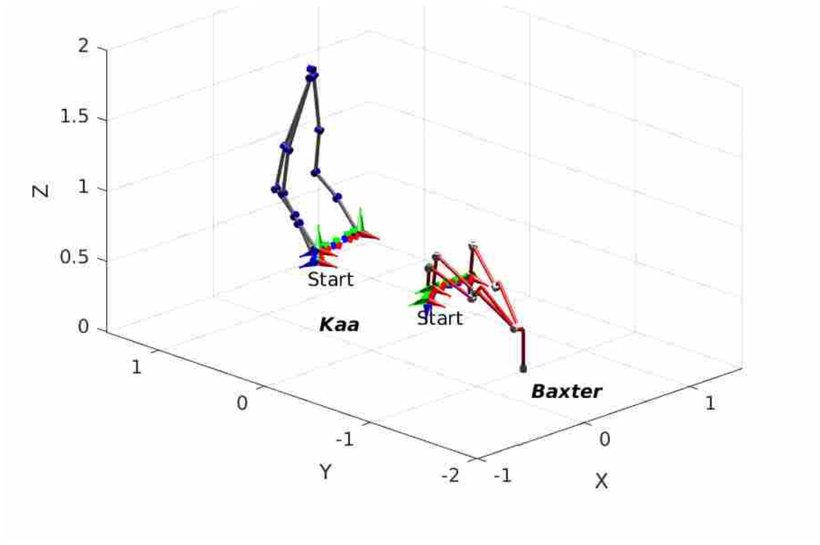


Figure 5.2: The base configurations for Kaa and Baxter found by our optimization are shown here for trajectory 1 along with the end effector poses from forward kinematics on the joint angles from the path planner.

5.3.1 Trajectory 1

Trajectory 1 consisted of lifting the board up 0.1m in z and forward 0.4m in x (relative to and in terms of the world frame w) while keeping it parallel to the ground. A real world example of a task like this is lifting a stretcher up and into an emergency response vehicle in a disaster scenario. We collected data on 20 trials of this trajectory (10 trials where the trajectory had a duration of 30 seconds and 10 trials of the same trajectory with a duration of 10 seconds). The optimized base poses and end effector trajectories are shown below in Figure 5.2.

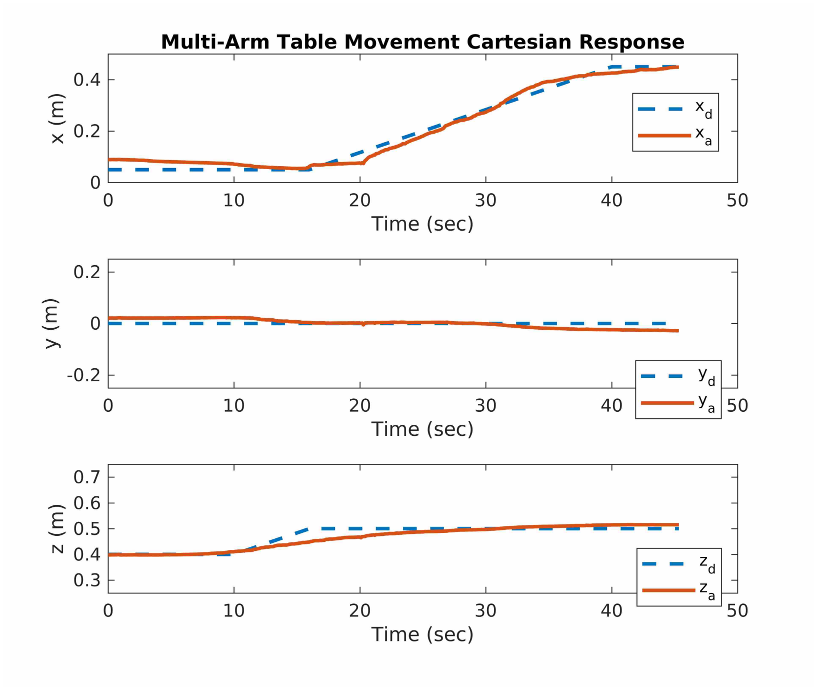


Figure 5.3: The board's Cartesian position (x_a, y_a, z_a) relative to the desired position (x_d, y_d, z_d) is shown here for a 30 second trajectory 1.

30 Second Trajectory 1

This is experiment 1 in Table 5.1. The Cartesian trajectory of the center of the board relative to the desired trajectory is shown below in Figure 5.3 for the best run. Figure 5.4 shows the Cartesian error of the board throughout the trajectory for this same run. The board is able to track the trajectory well with a maximum error less than 6cm throughout the trajectory.

The axis-angle orientation trajectory of the board relative to the desired trajectory is shown below in Figure 5.5 for the best run. Figure 5.6 shows the orientation error of the board throughout the trajectory for this same run. For each experiment, we computed the board's orientation error at each time step by finding the rotation matrix from the actual board pose to the desired board pose as follows: $R_{o_a}^{o_d} = R_w^{o_d} R_{o_a}^{w}$, where $R_w^{o_d}$ is the transpose of the rotation matrix representing the desired object (board) orientation in the world frame. $R_{o_a}^{w}$ represents the actual object orientation in the world frame. We then converted $R_{o_a}^{o_d}$ into an axis-angle representation, the magnitude of which is the angle between the actual and desired orientation (i.e. the orientation error).

Again, the orientation goal of this trajectory was to stay horizontal (lined up with the world frame). The table stays within 6.5° degrees of this goal throughout the trajectory.

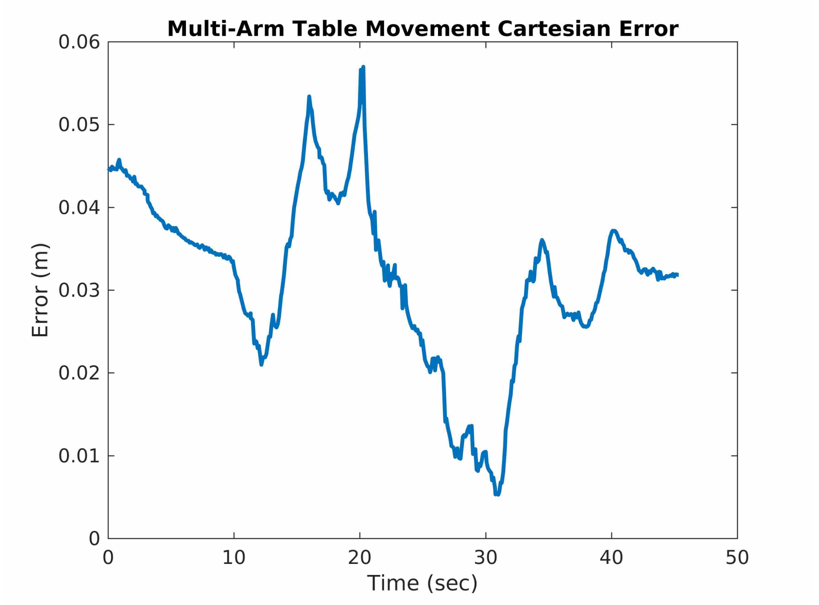


Figure 5.4: This shows the Cartesian error of the board throughout a 30 second trajectory 1.

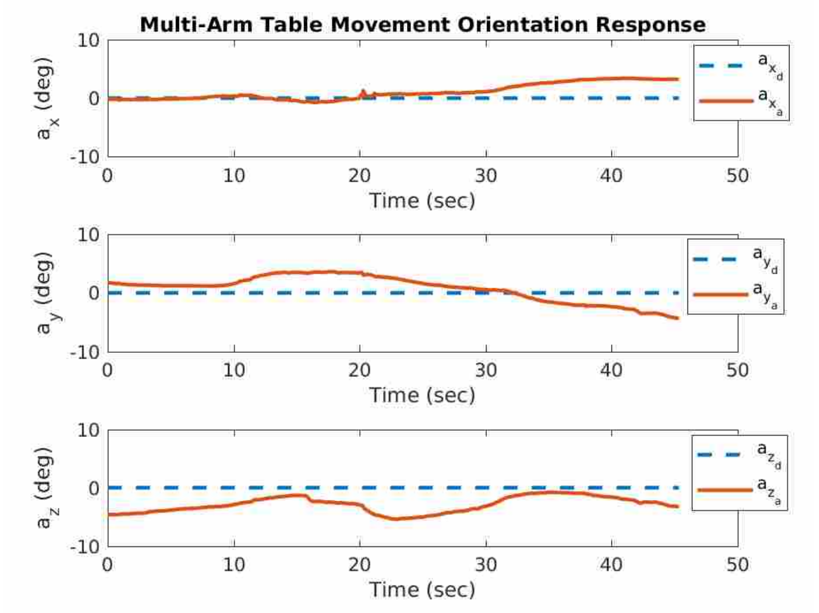


Figure 5.5: The board's orientation $(a_{x_a}, a_{y_a}, a_{z_a})$ relative to the desired orientation $(a_{x_d}, a_{y_d}, a_{z_d})$ is shown above for a 30 second trajectory 1.

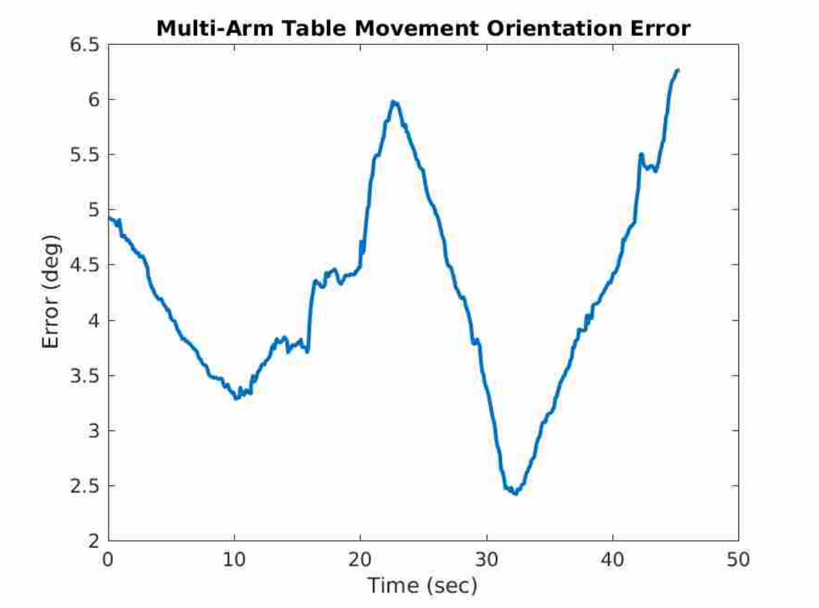


Figure 5.6: This shows the board's orientation error throughout a 30 second trajectory 1.

One concern was that because Baxter is less compliant than Kaa, Baxter would control the position of the board with Kaa dragging behind like a disturbance; however, when we examine the orientation of the board, we see that it was always negative about the z axis. This indicates that the side of the board supported by Kaa was slightly leading the side of the board supported by Baxter rather than dragging behind. As shown in https://youtu.be/aNnwKq28lo4, Kaa actually starts the movement slightly before Baxter and is in fact an active participant in the board control. To further illustrate this point, we had a few trials where Kaa's valves malfunctioned. Figure 5.7 shows one of these situations. As shown, Kaa clearly moves the board away from the desired trajectory (the board was supposed to be horizontal), illustrating that unless both Kaa and Baxter are performing their desired trajectories, we are not able to achieve desired object motion.

We performed 10 trials of this trajectory in order to show the repeatability of this response. For these 10 trials, we computed, the median, 90th percentile, and 10th percentile statistics at each time step. These statistics on the Cartesian trajectory of the center of the board relative to the desired trajectory are shown below in Figure 5.8. Figure 5.9 shows these statistics on the Cartesian error of the board throughout the trajectory.

The maximum error for all of these tests was bounded by 5cm and 12cm (10th and 90th percentiles) with a maximum median error of 7.5cm.

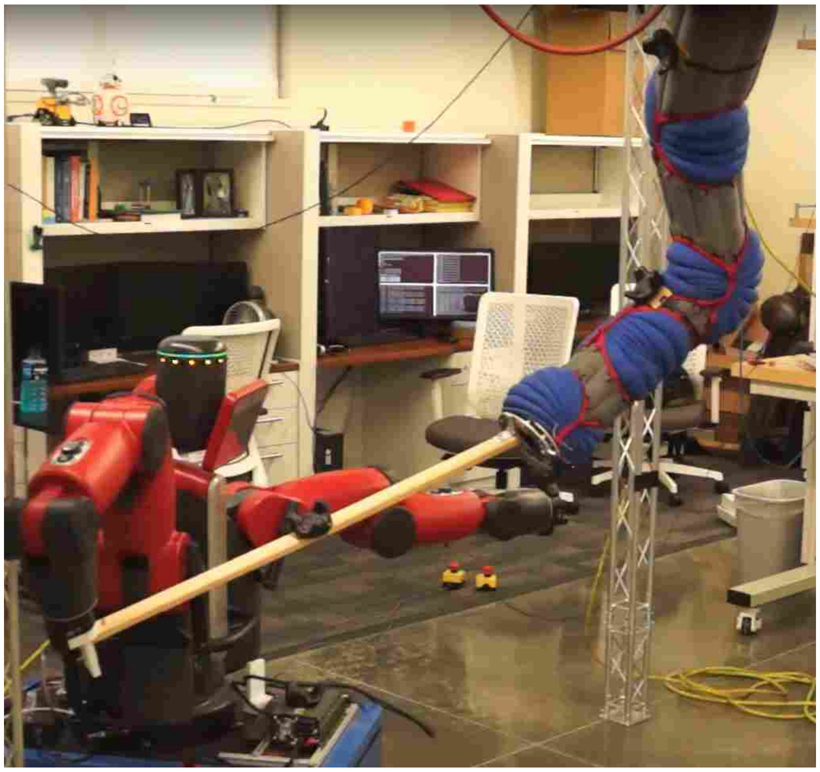


Figure 5.7: Kaa's valves misbehaved resulting in undesired board motion.

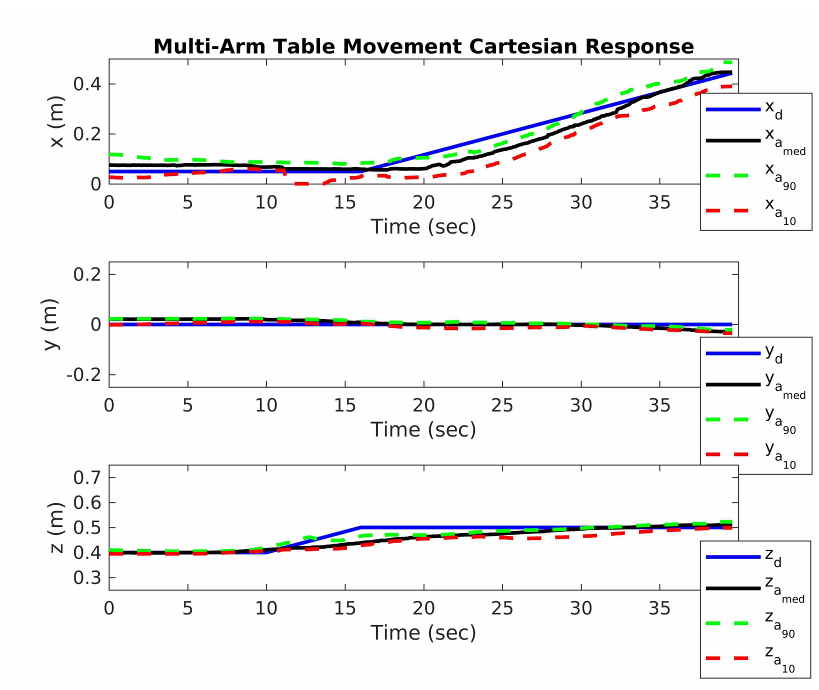


Figure 5.8: This shows the board's median, 90th percentile, and 10th percentile Cartesian positions relative to the desired position (x_d, y_d, z_d) for 10 trials of a 30 second trajectory 1.

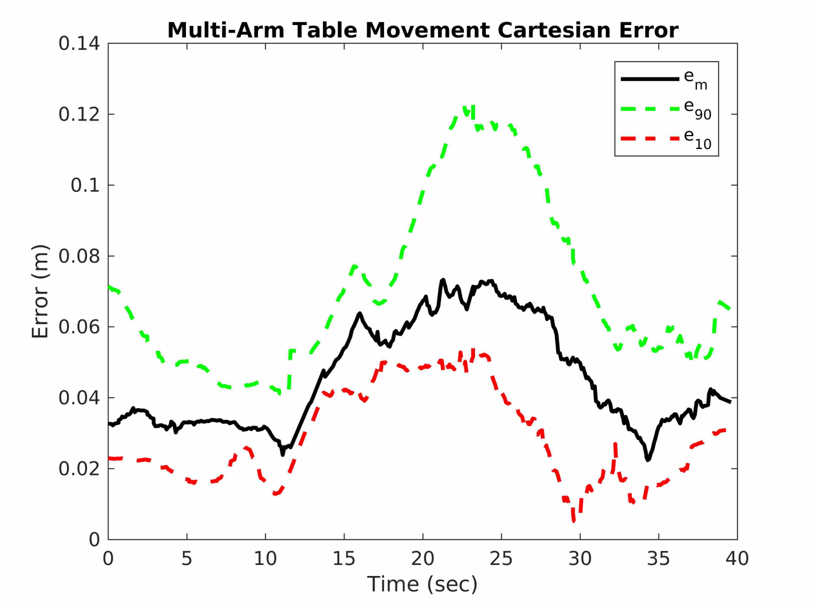


Figure 5.9: The median, 90th percentile, and 10th percentile Cartesian error of the board throughout 10 trials of a 30 second trajectory 1 is shown above.

The median, 90th percentile, and 10th percentile axis-angle orientation trajectories of the board relative to the desired trajectory are shown below in Figure 5.10. Figure 5.11 shows these same statistics for the orientation error of the board throughout the trajectory.

Throughout these 10 trials, the board was able to stay within the maximum 90th and 10th percentile bounds of 10.3° and 3° with a maximum median orientation error of less than 6°.

10 Second Trajectory 1

Next we performed the same trajectory but with a duration of 10 seconds rather than 30. This is experiment 2 in Table 5.1. The Cartesian trajectory of the center of the board relative to the desired trajectory for the best run is shown below in Figure 5.12. Figure 5.13 shows the Cartesian error of the board throughout the trajectory for the same run. The maximum Cartesian error of the board throughout the trajectory was 7cm.

The orientation trajectory of the board relative to the desired trajectory is shown below in Figure 5.14. Figure 5.15 shows the orientation error of the board throughout the trajectory. The orientation error stayed below 10° throughout the trajectory.

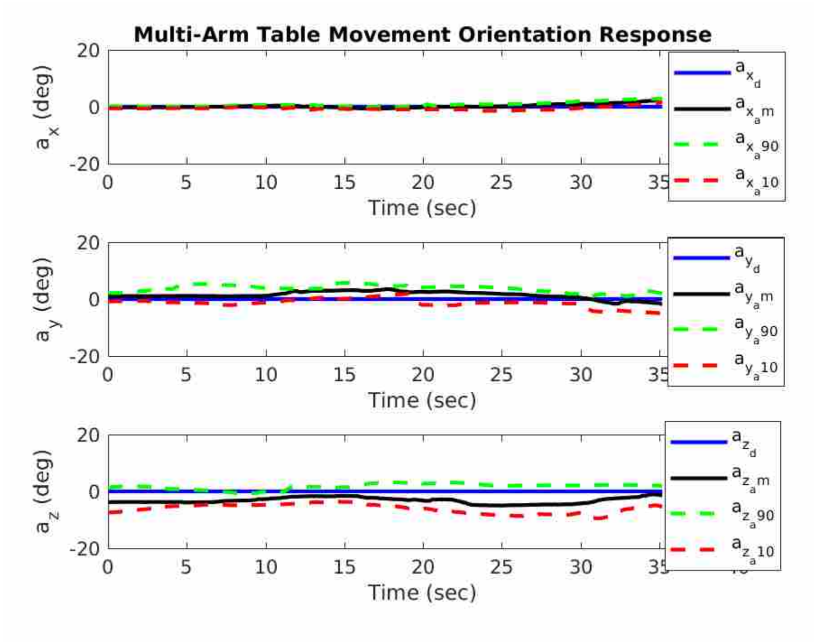


Figure 5.10: This shows the board's median, 90th percentile, and 10th percentile axis-angle orientations relative to the desired orientation $(a_{x_d}, a_{y_d}, a_{z_d})$ for 10 trials of a 30 second trajectory 1.

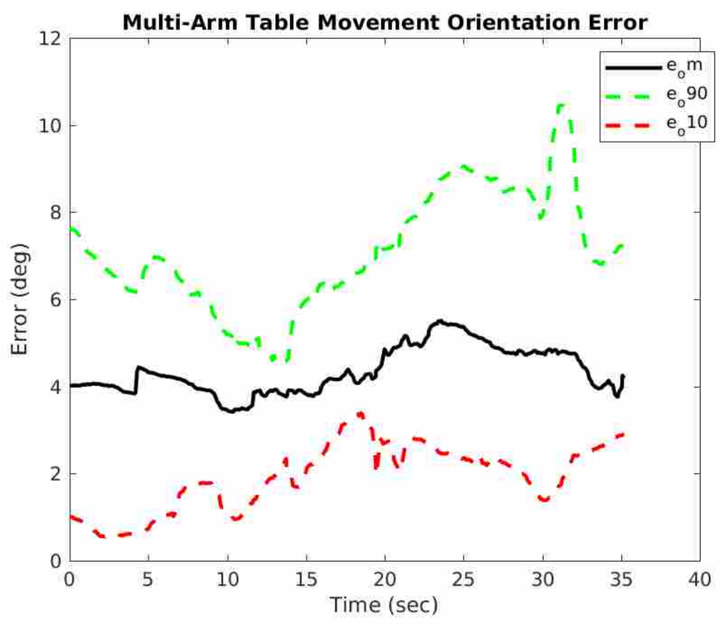


Figure 5.11: The board's median, 90th percentile, and 10th percentile orientation error throughout 10 trials of a 30 second trajectory 1 is shown here.

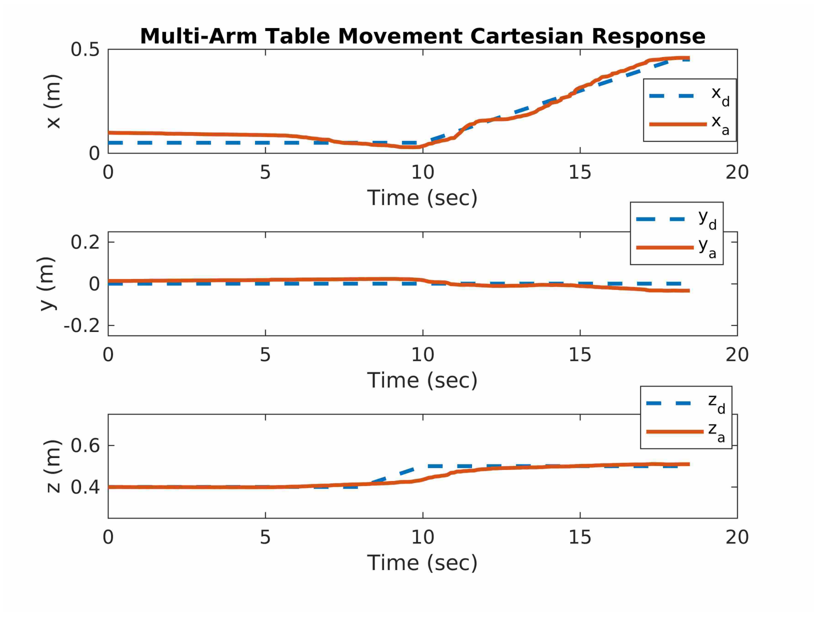


Figure 5.12: This shows the board's Cartesian position (x_a, y_a, z_a) relative to the desired position (x_d, y_d, z_d) for a 10 second trajectory 1.

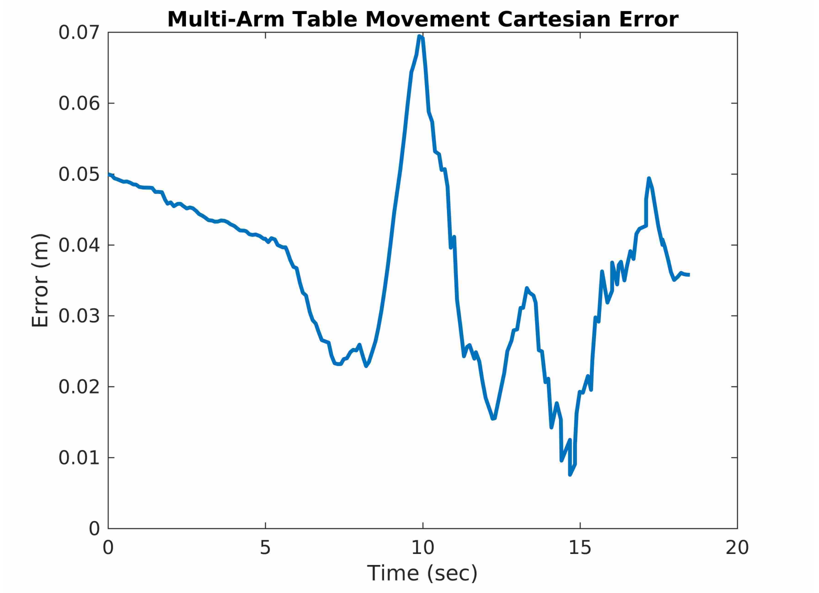


Figure 5.13: The Cartesian error of the board throughout a 10 second trajectory 1 is shown above.

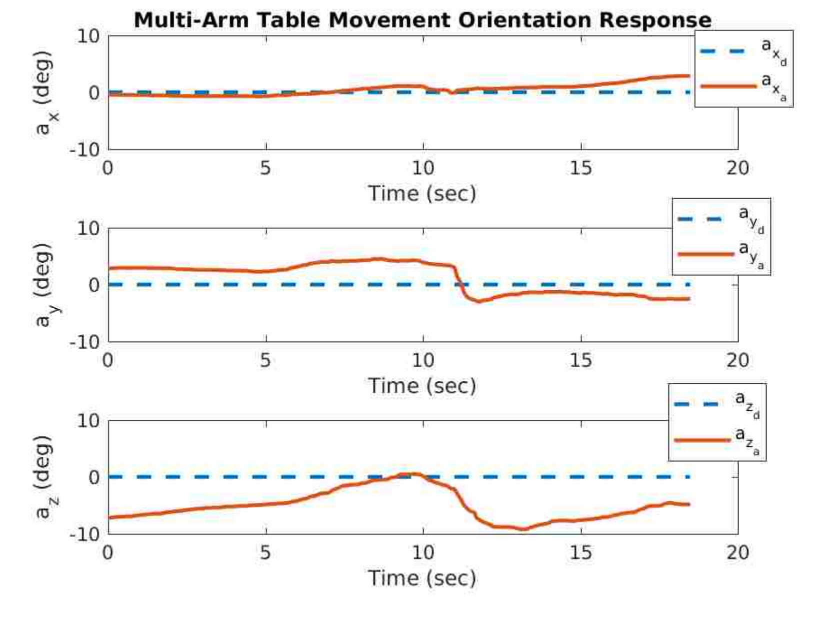


Figure 5.14: This shows the board's orientation $(a_{x_a}, a_{y_a}, a_{z_a})$ relative to the desired orientation $(a_{x_d}, a_{y_d}, a_{z_d})$ for a 10 second trajectory 1.

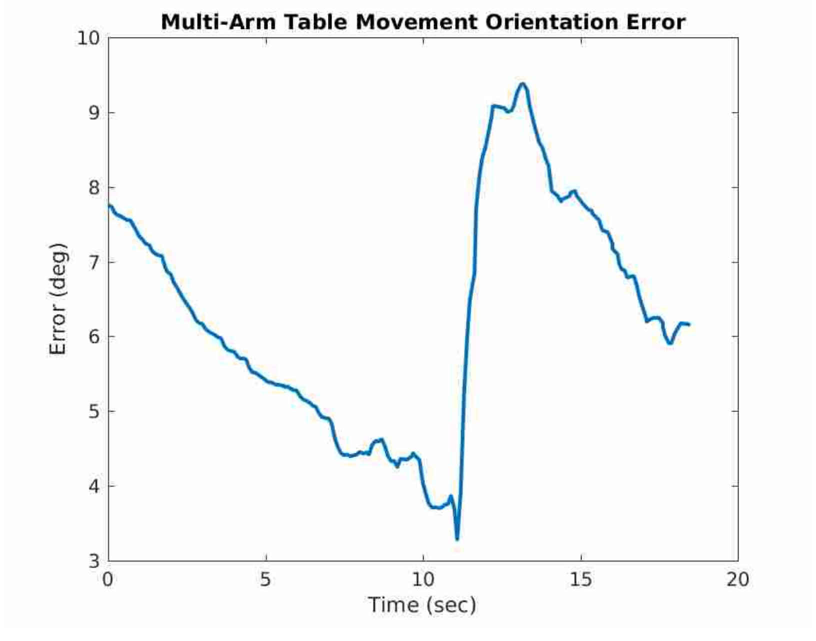


Figure 5.15: The board's orientation error throughout a 10 second trajectory 1 is shown here.

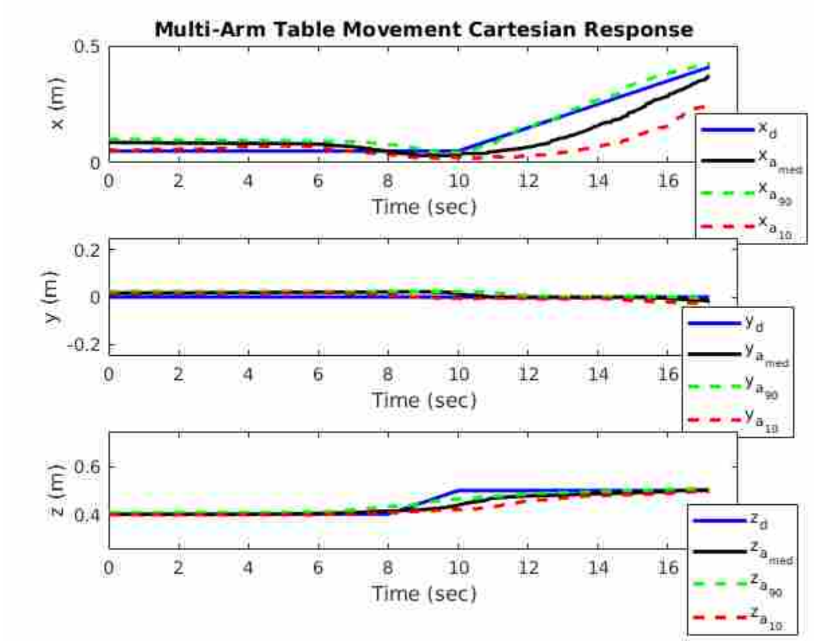


Figure 5.16: This shows the board's median, 90th percentile, and 10th percentile Cartesian positions relative to the desired position (x_d, y_d, z_d) for 10 trials of a 10 second trajectory 1.

We also performed 10 trials of this trajectory in order to show the repeatability of this response. For these 10 trials, we again computed, the median, 90th percentile, and 10th percentile statistics at each time step. These statistics on the Cartesian trajectory of the center of the board relative to the desired trajectory are shown below in Figure 5.16. Figure 5.17 shows these statistics on the Cartesian error of the board throughout the trajectory.

The maximum error for all of these tests was bounded by 4cm and 19.5cm (maximum of the 10th and 90th percentiles) with a maximum median error of 10cm. These errors are larger than the 30 second duration version of this trajectory because the task space controller is able to track the slower trajectory much better. This is due to the dynamic response of the task space controllers, and is not a fundamental limitation. Future work in this area includes developing controllers that are able to track a trajectory with less error.

The median, 90th percentile, and 10th percentile axis-angle orientation trajectories of the board relative to the desired trajectory are shown below in Figure 5.18. Figure 5.19 shows the median, 90th percentile, and 10th percentile for the orientation error of the board throughout the trajectory.

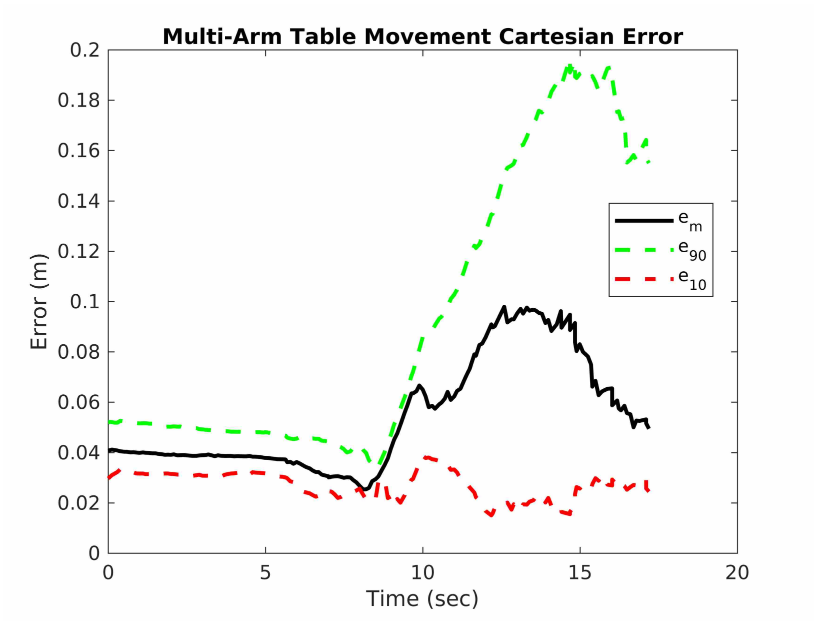


Figure 5.17: The median, 90th percentile, and 10th percentile Cartesian error of the board throughout 10 trials of a 10 second trajectory 1 is shown above.

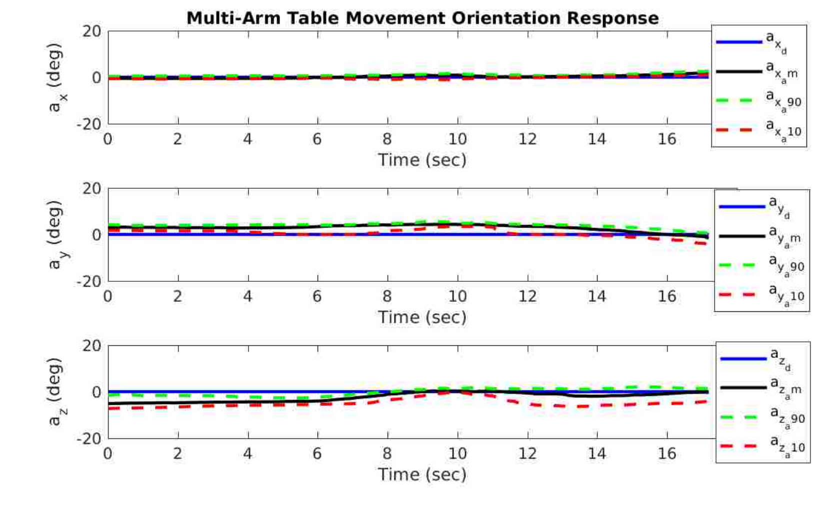


Figure 5.18: This shows the board's median, 90th percentile, and 10th percentile axis-angle orientations relative to the desired orientation $(a_{x_d}, a_{y_d}, a_{z_d})$ for 10 trials of a 10 second trajectory 1.

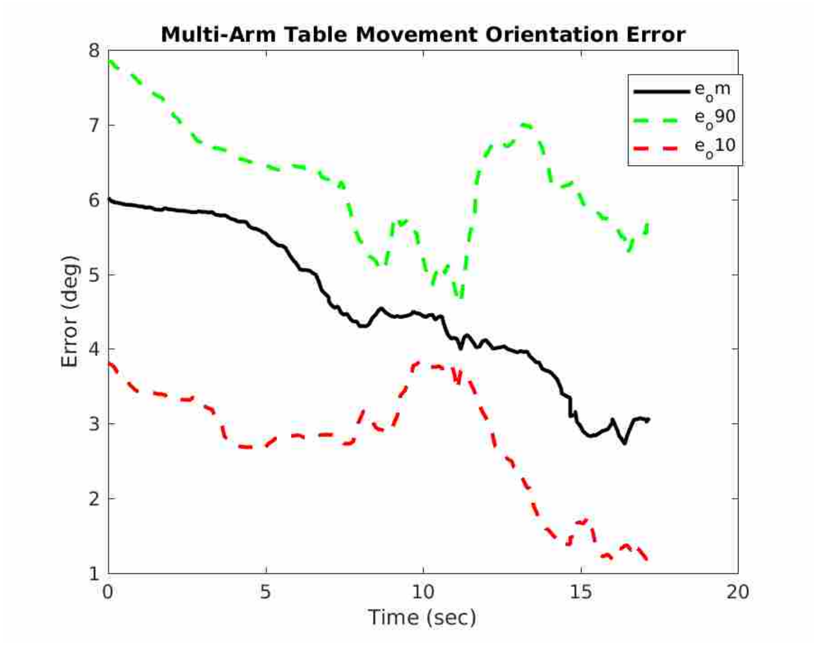


Figure 5.19: The board's median, 90th percentile, and 10th percentile orientation error throughout 10 trials of a 10 second trajectory 1 is shown here.

Throughout these 10 trials, the board was able to stay within the maximum 90th and 10th percentile bounds of 10.3° and 3° with a maximum median orientation error of less than 6°.

5.3.2 Trajectory 2

We performed 20 trials of trajectory 1 in order to show the repeatability of following a trajectory in a multi-arm task; however, this does not show how well this control method generalizes to a different trajectory. In order to show this, we chose another board trajectory to track, and again ran it at two durations (10 seconds and 30 seconds). Trajectory 2 consisted of moving the board forward 0.4m in x and left 0.4m in y while keeping it parallel to the ground and at a constant height. The optimized base poses and end effector trajectories for this experiment are shown below in Figure 5.20.

30 Second Trajectory 2

This is experiment 3 in Table 5.1. The Cartesian trajectory of the center of the board relative to the desired trajectory is shown below in Figure 5.21. Figure 5.22 shows the Cartesian error of

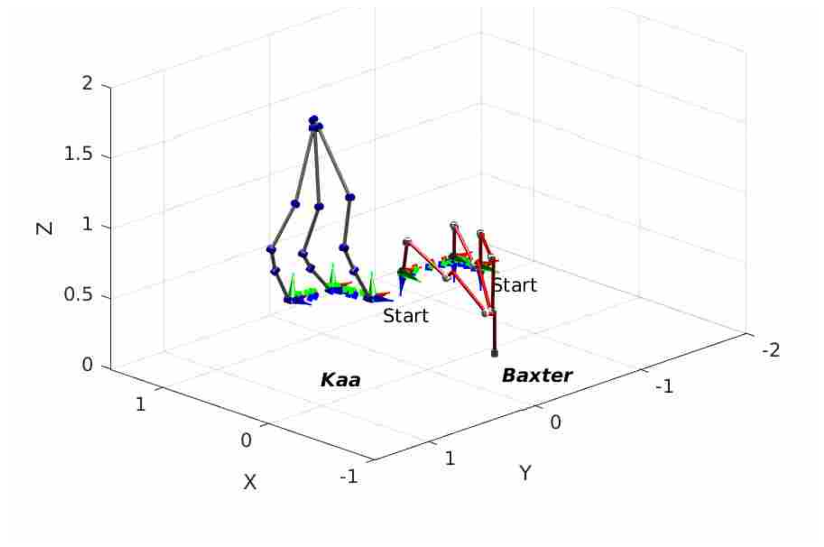


Figure 5.20: This shows Kaa and Baxter with the base configurations found by our optimization as well as the end effector poses from forward kinematics on the joint angles from the path planner for trajectory 2.

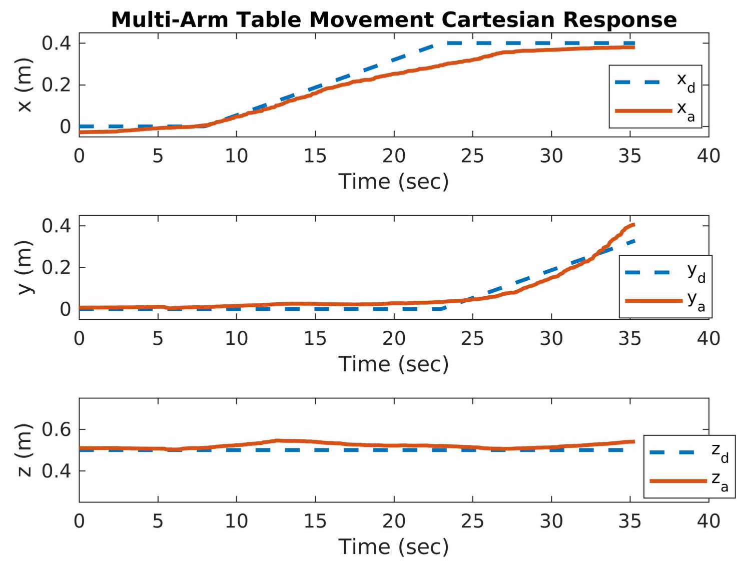


Figure 5.21: The board's Cartesian position (x_a, y_a, z_a) relative to the desired position (x_d, y_d, z_d) for a 30 second trajectory 2 is shown here.

the board throughout the trajectory. The maximum Cartesian error throughout the trajectory is less than 12cm.

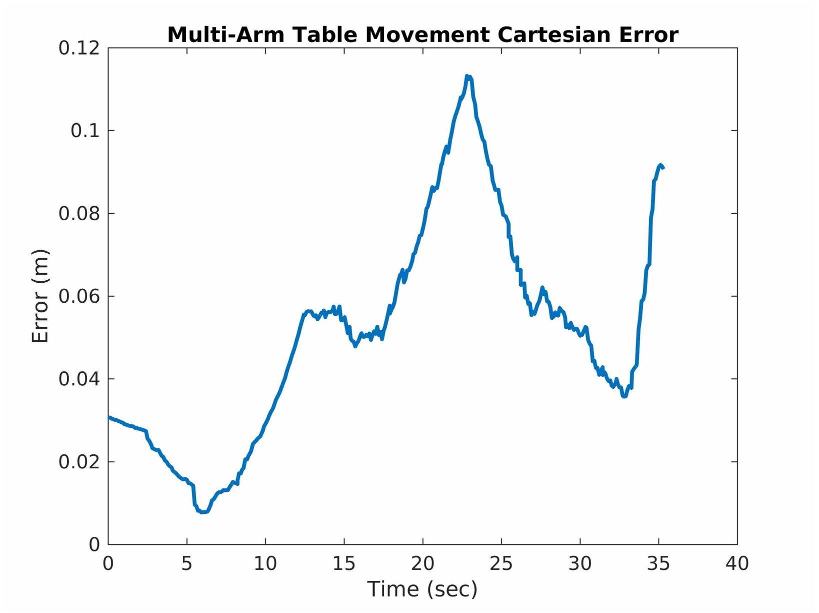


Figure 5.22: This shows the Cartesian error of the board throughout a 30 second trajectory 2.

The orientation trajectory of the board relative to the desired trajectory is shown below in Figure 5.23. Figure 5.24 shows the orientation error of the board throughout the trajectory. The goal was to keep the board horizontal throughout the trajectory. In this regard the system responds worse than the first trajectory with a max error of just under 14°. This is because this is a longer trajectory (.8m vs. .5m in Cartesian space) and the arms have a harder time tracking the extremes of the trajectory as they are less dexterous in these areas.

10 Second Trajectory 2

Next we ran this same trajectory but for a duration of 10 seconds rather than 30. This is experiment 4 in Table 5.1. The Cartesian trajectory of the center of the board relative to the desired trajectory is shown below in Figure 5.25. Figure 5.26 shows the Cartesian error of the board throughout the trajectory. The maximum Cartesian error throughout the trajectory is less than 14cm.

The orientation trajectory of the board relative to the desired trajectory is shown below in Figure 5.27. Figure 5.28 shows the orientation error of the board throughout the trajectory. Again, there is relatively high orientation error (around 15° at the beginning and end of the trajectory as

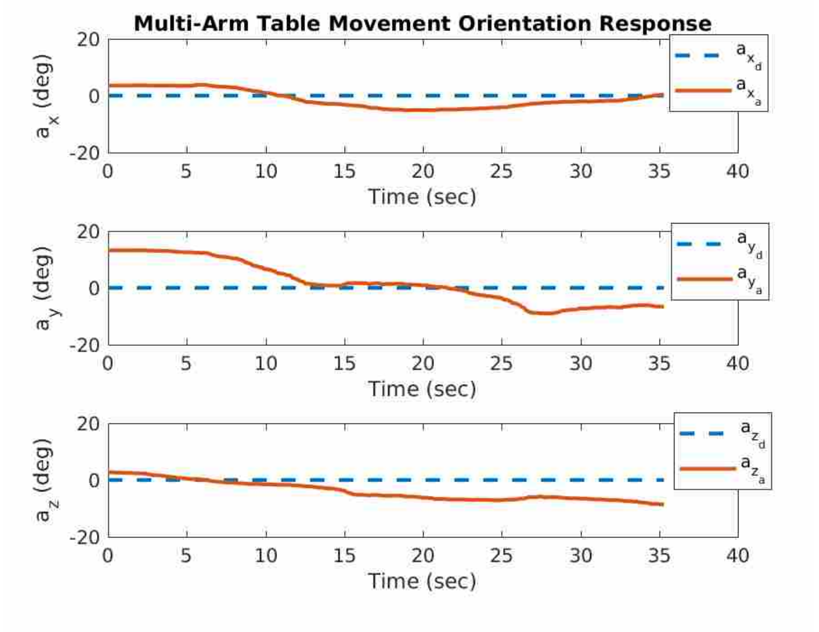


Figure 5.23: The board's orientation $(a_{x_a}, a_{y_a}, a_{z_a})$ relative to the desired orientation $(a_{x_d}, a_{y_d}, a_{z_d})$ for a 30 second trajectory 2 is shown above.

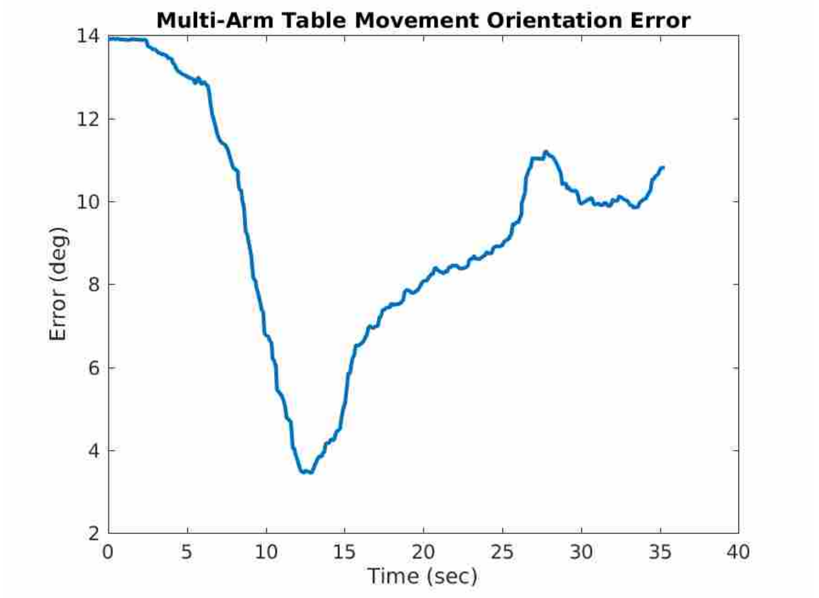


Figure 5.24: This shows the board's orientation error throughout a 30 second trajectory 2.

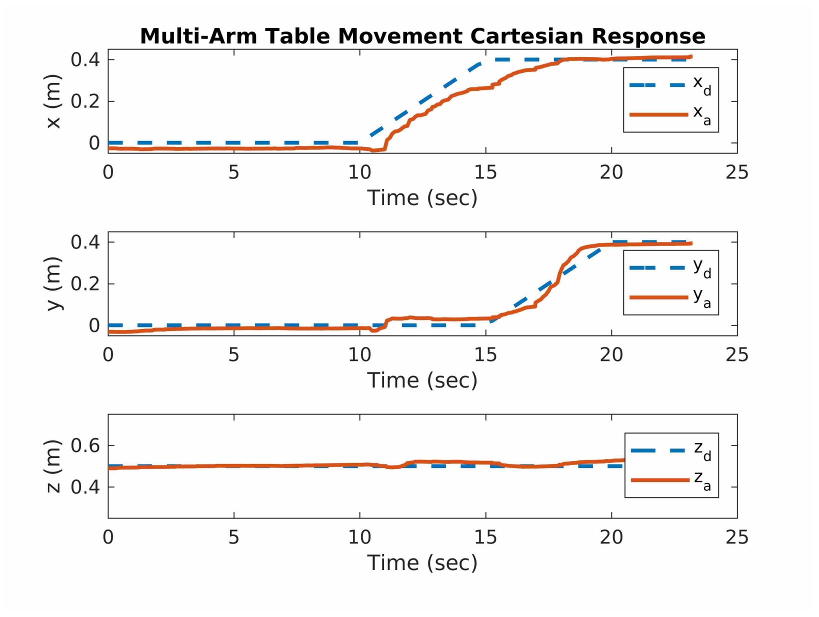


Figure 5.25: The board's Cartesian position (x_a, y_a, z_a) relative to the desired position (x_d, y_d, z_d) for a 10 second trajectory 2 is shown here.

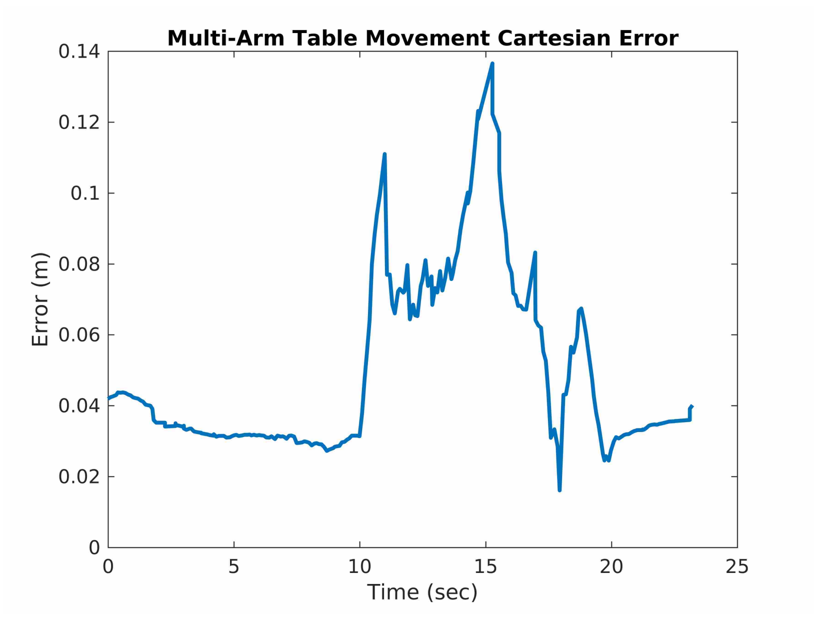


Figure 5.26: This shows the Cartesian error of the board throughout a 10 second trajectory 2.

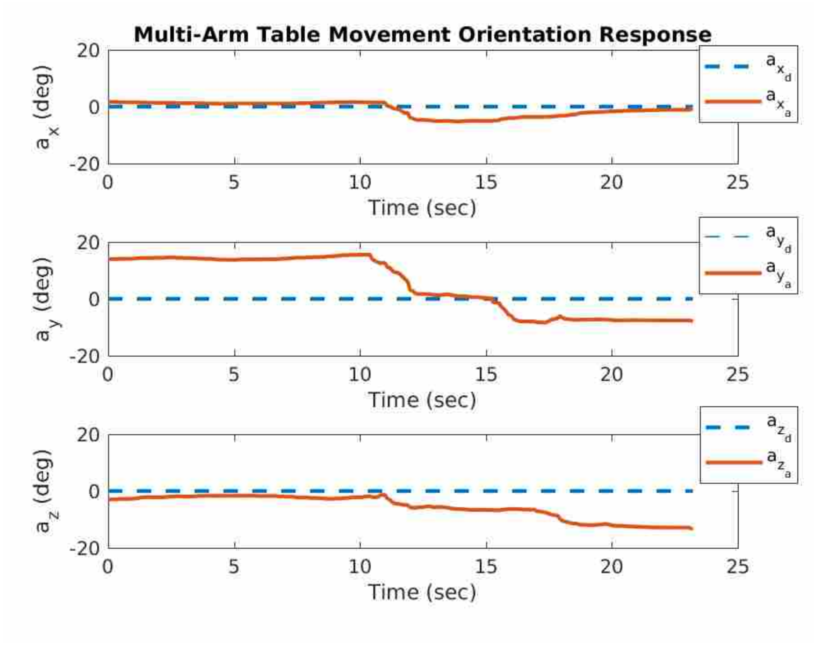


Figure 5.27: The board's orientation $(a_{x_a}, a_{y_a}, a_{z_a})$ relative to the desired orientation $(a_{x_d}, a_{y_d}, a_{z_d})$ for a 10 second trajectory 2 is shown above.

these are at a less dexterous region of the robot's workspace (because the trajectory was longer in Cartesian space).

5.3.3 Results Discussion

In this chapter, we demonstrated a simple coordinated multi-arm manipulation control scheme using Kaa and Baxter. We tested this control scheme on two trajectories with two durations each (30 seconds and 10 seconds). For the first trajectory, we ran 10 trials on each trajectory duration to demonstrate the repeatability of the control scheme. We then tested this control scheme on the second trajectory showing that the control scheme success is not trajectory specific.

This control scheme does not result in high task space precision; however, the precision is high enough to enable soft robots to complete simple everyday tasks adequately.

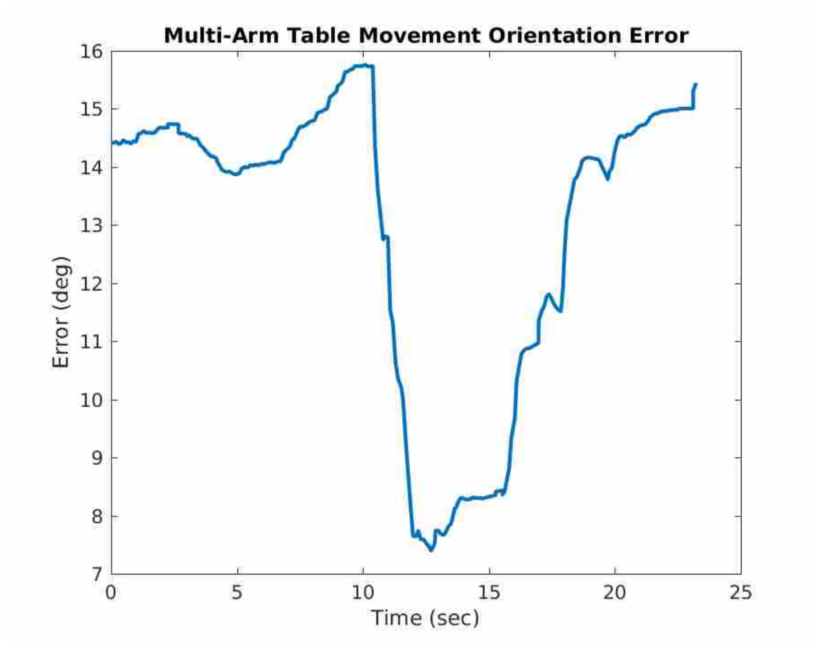


Figure 5.28: This shows the board's orientation error throughout a 10 second trajectory 2.

CHAPTER 6. CONCLUSION

In this thesis, we demonstrated the first coordinated, multi-arm manipulation tasks with soft robots. The techniques we developed to achieve this were successful; however, there are multiple areas where future work could extend this research. In this chapter, we first discuss work we believe is a natural extension of this research. We then conclude this thesis by discussing the specific contributions made.

6.1 Future Work

The multi-arm base placement optimization we developed finds the optimal base pose of each arm in a coordinated, multi-arm manipulation task. This optimization relies on the user providing object grasping poses for each arm as homogeneous transforms from the object frame to each manipulator's end effector frame. We found that if poor grasping poses are chosen, the manipulator is not able to track the desired trajectory as well as when good grasping poses are chosen. Thus a natural extension of this optimization is to optimize the grasping pose of each manipulator simultaneously with the base pose. Additionally, in its current formulation, this is a single objective optimization. Another extension of this work would be to reformulate the optimization as a multi-objective optimization with other metrics like maximizing the force output, or ability to maintain a constant object velocity.

Another area of future work is with the task space control of soft robots. The hybrid servoing controller developed in this work is able to reduce the end effector pose error; however, the response is not particularly fast. This becomes particularly detrimental in trajectory tracking. We have seen significant improvement in joint space control of soft robots by using model predictive controllers. Model predictive control is also fairly robust to modeling error. So, we believe a possible way to improve the task space response of soft robots is to implement a task space model predictive controller.

6.2 Contributions

The main goal of this thesis was to enable coordinated multi-arm manipuation with soft robots. We needed to solve several smaller research problems in order to achieve this overarching goal. We first presented an optimization to find the homogeneous transformations between unrelated pose sensors and demonstrated its effectiveness. This is especially important in soft robotics as traditional joint angle sensing with encoders at the joints is not possible. We instead used relative orientations between pose sensors placed on the links. If multiple pose sensors are going to be used (for sensor fusion or sensor validation), the homogeneous transformations between them must be known. This was the purpose of our sensor correspondence optimization. Next we presented a task space control method for soft robots. Because the kinematic model of soft robots varies with time, joint angle control alone is not enough to achieve end effector task space control. Coordinated multi-arm manipulation with soft robots required fairly accurate task space end effector control of the soft robots to be used in the multi-arm task. We developed a six DoF task space hybrid servoing controller for large-scale, soft, pneumatically actuated robots and demonstrated this control method on two soft robotic platforms (King Louie and Kaa).

Another important consideration in a coordinated multi-arm manipulation task is where to place the base of each robot involved in the task. For this purpose, we developed a general multi-arm base placement optimization. This optimization is general enough to allow for the base placement of any number of arms with one to six design variables each. We demonstrated this optimization's effectiveness in simulation, and then used it to optimize the base placement of Baxter and Kaa for two coordinated multi-arm manipulation tasks.

Finally, we demonstrated a simple coordinated multi-arm manipulation control scheme. This control scheme capitalizes on the natural compliance of soft robots, as there is no need for high bandwidth, centralized force control (which is necessary for multi-arm manipulation tasks with rigid robots). Rather we used individual task space controllers on each robot arm and sent coordinated trajectories to each arm. We demonstrated this control scheme's effectiveness by selecting two desired object trajectories, placing the bases of Kaa and Baxter at the optimized base locations for each trajectory, and then tracking the trajectories with coordinated multi-arm control. This demonstrates that because of the compliance of soft robot arms, simple control schemes can be very successful in multi-arm tasks while maintaining an inherent safety of delicate equipment and humans working around the robots.

The tools developed in this thesis are a framework which enable coordinated multi-arm manipulation tasks with soft robots.

REFERENCES

[1] Smith, C., Karayiannidis, Y., Nalpantidis, L., Gratal, X., Qi, P., Dimarogonas, D. V., and Kragic, D., 2012. "Dual arm manipulation-A survey." *Robotics and Autonomous Systems,* **60**(10), pp. 1340–1353. 2

[2] Hyatt, P., Kraus, D., Sherrod, V., Rupert, L., Day, N., and Killpack, M., 2018. "Configuration estimation for accurate position control of large-scale soft robots." *IEEE/ASME Transactions on Mechatronics,* **Accepted**. 3, 5

[3] Terry, J. S., Rupert, L., and Killpack, M. D., 2017. "Comparison of Linearized Dynamic Robot Manipulator Models for Model Predictive Control." In *IEEE Humanoids Conference*. 3, 8, 10, 23

[4] RoboSavvy, 2018. *$vive_ros$*: Ros package for publishing htc vive device locations. 6

[5] Best, C. M., Gillespie, M. T., Hyatt, P., Rupert, L., Sherrod, V., and Killpack, M. D., 2016. "A new soft robot control method: Using model predictive control for a pneumatically actuated humanoid." *IEEE Robotics & Automation Magazine,* **23**(3), pp. 75–84. 8

[6] Bodily, D. M., Allen, T. F., and Killpack, M. D., 2017. "Multi-objective design optimization of a soft, pneumatic robot." In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, IEEE, pp. 1864–1871. 10, 42

[7] Kato, H., and Billinghurst, M., 1999. "Marker tracking and hmd calibration for a video-based augmented reality conferencing system." In *Augmented Reality, 1999.(IWAR'99) Proceedings. 2nd IEEE and ACM International Workshop on*, IEEE, pp. 85–94. 13

[8] Zhang, Z., 2000. "A flexible new technique for camera calibration." *IEEE Transactions on pattern analysis and machine intelligence,* **22**(11), pp. 1330–1334. 13

[9] Svoboda, T., Martinec, D., and Pajdla, T., 2005. "A convenient multicamera self-calibration for virtual environments." *Presence: Teleoperators & virtual environments,* **14**(4), pp. 407–422. 13

[10] Holtz, J., and Biswas, J., 2017. "Automatic extrinsic calibration of depth sensors with ambiguous environments and restricted motion." In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, IEEE, pp. 2235–2240. 13

[11] Jones, E., Oliphant, T., Peterson, P., et al., 2001–. SciPy: Open source scientific tools for Python. 17

[12] Nocedal, J., and Wright, S. J., 2006. *Numerical Optimization.* Springer-Verlag New York. 17

[13] George Thuruthel, T., Ansari, Y., Falotico, E., and Laschi, C., 2018. "Control strategies for soft robotic manipulators: A survey." *Soft robotics*. 21, 23

[14] Largilliere, F., Verona, V., Coevoet, E., Sanz-Lopez, M., Dequidt, J., and Duriez, C., 2015. "Real-time control of soft-robots using asynchronous finite element modeling." In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, IEEE, pp. 2550–2555. 21

[15] Kapadia, A. D., Fry, K. E., and Walker, I. D., 2014. "Empirical investigation of closed-loop control of extensible continuum manipulators." In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, IEEE, pp. 329–335. 22

[16] Bajo, A., Goldman, R. E., and Simaan, N., 2011. "Configuration and joint feedback for enhanced performance of multi-segment continuum robots." In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, IEEE, pp. 2905–2912. 22

[17] Yip, M. C., and Camarillo, D. B., 2014. "Model-less feedback control of continuum manipulators in constrained environments." *IEEE Transactions on Robotics,* **30**(4), pp. 880–889. 22

[18] Li, M., Kang, R., Branson, D. T., and Dai, J. S., 2018. "Model-free control for continuum robots based on an adaptive kalman filter." *IEEE/ASME Transactions on Mechatronics,* **23**(1), Feb, pp. 286–297. 22

[19] Melingui, A., Lakhal, O., Daachi, B., Mbede, J. B., and Merzouki, R., 2015. "Adaptive neural network control of a compact bionic handling arm." *IEEE/ASME Transactions on Mechatronics,* **20**(6), Dec, pp. 2862–2875. 22

[20] Wang, H., Yang, B., Liu, Y., Chen, W., Liang, X., and Pfeifer, R., 2016. "Visual Servoing of Soft Robot Manipulator in Constrained Environments with an Adaptive Controller." *IEEE/ASME Transactions on Mechatronics*, pp. 1–1. 22

[21] Wang, H., Chen, W., Yu, X., Deng, T., Wang, X., and Pfeifer, R., 2013. "Visual servo control of cable-driven soft robotic manipulator." In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, IEEE, pp. 57–62. 22

[22] Penning, R. S., Jung, J., Ferrier, N. J., and Zinn, M. R., 2012. "An evaluation of closed-loop control options for continuum manipulators." In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, IEEE, pp. 5392–5397. 22

[23] Camarillo, D. B., Carlson, C. R., and Salisbury, J. K., 2009. "Task-space control of continuum manipulators with coupled tendon drive." In *Experimental robotics*, Springer, pp. 271–280. 22

[24] Best, C. M., Gillespie, M. T., Hyatt, P., Rupert, L., Sherrod, V., and Killpack, M. D., 2015. "Model predictive control for pneumatically actuated soft robots." *Robotics and Automation Magazine. IEEE.* 23

[25] Gillespie, M. T., Best, C. M., and Killpack, M. D., 2016. "Simultaneous position and stiffness control for an inflatable soft robot." In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, IEEE, pp. 1095–1101. 23

[26] Beeson, P., and Ames, B., 2015. "TRAC-IK: An open-source library for improved solving of generic inverse kinematics." *IEEE-RAS International Conference on Humanoid Robots,* **2015-Decem**, pp. 928–935. 24, 25

[27] Sousa, C. a. D., 2015. "Dynamic model identification of robot manipulators: Solving the physical feasibility problem." PhD thesis, Universidade De Coimbra. 25

[28] Wolovich, W. A., and Elliott, H., 1984. "A computational technique for inverse kinematics." In *Decision and Control, 1984. The 23rd IEEE Conference on*, Vol. 23, IEEE, pp. 1359–1363. 24

[29] Balestrino, A., De Maria, G., and Sciavicco, L., 1984. "Robust control of robotic manipulators." *IFAC Proceedings Volumes,* **17**(2), pp. 2435–2440. 24

[30] Zeghloul, S., Blanchard, B., and Ayrault, M., 1997. "Smar: A robot modeling and simulation system." *Robotica,* **15**(1), pp. 63–73. 31

[31] Ouezdou, F. B., and Régnier, S., 1997. "General method for kinematic synthesis of manipulators with task specifications." *Robotica,* **15**(6), pp. 653–661. 31

[32] Abdel-Malek, K., 2000. "On the placement of serial manipulators." In *ASME DETC, Sept. 2000*. 32

[33] Yang, J. J., Yu, W., Kim, J., and Abdel-Malek, K., 2009. "On the placement of open-loop robotic manipulators for reachability." *Mechanism and Machine Theory,* **44**(4), pp. 671–684. 32

[34] Lopes, A. M., and Pires, E. S., 2011. "Optimization of the workpiece location in a machining robotic cell." *International Journal of Advanced Robotic Systems,* **8**(6), p. 73. 32

[35] Vosniakos, G.-C., and Matsas, E., 2010. "Improving feasibility of robotic milling through robot placement optimisation." *Robotics and Computer-Integrated Manufacturing,* **26**(5), pp. 517–525. 32

[36] Hassan, M., Liu, D., Paul, G., and Huang, S., 2015. "An approach to base placement for effective collaboration of multiple autonomous industrial robots." In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, IEEE, pp. 3286–3291. 32

[37] Hassan, M., Liu, D., and Paul, G., 2018. "Collaboration of multiple autonomous industrial robots through optimal base placements." *Journal of Intelligent & Robotic Systems,* **90**(1-2), pp. 113–132. 32

[38] Dharmawan, A. G., Foong, S., and Soh, G. S., 2016. "Simultaneous optimal robot base placement and motion planning using expanded lagrangian homotopy." In *ASME 2016 Dynamic Systems and Control Conference*, American Society of Mechanical Engineers, pp. V002T24A010–V002T24A010. 32

[39] Dharmawan, A. G., Sedore, B. W. C., Soh, G. S., Foong, S., and Otto, K., 2015. "Robot base placement and kinematic evaluation of 6r serial manipulators to achieve collision-free

welding of large intersecting cylindrical pipes." In *ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, American Society of Mechanical Engineers, pp. V05CT08A010–V05CT08A010. 32

[40] Bodily, D. M., Allen, T. F., and Killpack, M. D., 2017. "Motion planning for mobile robots using inverse kinematics branching." In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, IEEE, pp. 5043–5050. 32, 45, 46, 63, 64

[41] Schulman, J., Duan, Y., Ho, J., Lee, A., Awwal, I., Bradlow, H., Pan, J., Patil, S., Goldberg, K., and Abbeel, P., 2014. "Motion planning with sequential convex optimization and convex collision checking." *The International Journal of Robotics Research,* **33**(9), pp. 1251–1270. 32

[42] Olaru, A., Olaru, S., Mihai, N., and Ciupitu, L., 2017. "Optimization of the robot's position base point by using the proper algorithm and iterative pseudo inverse jacobian neural network matrix method.." *Applied Mechanics & Materials,* **859**. 33

[43] Trejos, A. L., and Patel, R. V., 2005. "Port placement for endoscopic cardiac surgery based on robot dexterity optimization." In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, IEEE, pp. 912–917. 33

[44] Li, Z., Glozman, D., Milutinovic, D., and Rosen, J., 2011. "Maximizing dexterous workspace and optimal port placement of a multi-arm surgical robot." In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, IEEE, pp. 3394–3399. 33

[45] Mitsi, S., Bouzakis, K.-D., Sagris, D., and Mansour, G., 2008. "Determination of optimum robot base location considering discrete end-effector positions by means of hybrid genetic algorithm." *Robotics and Computer-Integrated Manufacturing,* **24**(1), pp. 50–59. 33

[46] Sotiropoulos, P., Aspragathos, N., and Andritsos, F., 2011. "Optimum docking of an unmanned underwater vehicle for high dexterity manipulation." *IAENG International Journal of Computer Science,* **38**(1), pp. 48–56. 33

[47] Ren, S., Xie, Y., Yang, X., Xu, J., Wang, G., and Chen, K., 2017. "A method for optimizing the base position of mobile painting manipulators." *IEEE Transactions on Automation Science and Engineering,* **14**(1), pp. 370–375. 34

[48] Yu, Q., Wang, G., Ren, T., Hua, X., and Chen, K., 2017. "Starting base position optimization in integrated task planning for mobile manipulators painting large workpieces." In *ASME 2017 International Mechanical Engineering Congress and Exposition*, American Society of Mechanical Engineers, pp. V002T02A076–V002T02A076. 34

[49] Stoer, J., and Bulirsch, R., 2013. *Introduction to numerical analysis.*, Vol. 12 Springer Science & Business Media. 35

[50] Yoshikawa, T., 1985. "Manipulability of robotic mechanisms." *The international journal of Robotics Research,* **4**(2), pp. 3–9. 36

[51] Sakaino, S., Sato, T., and Ohnishi, K., 2011. "Precise Position/Force Hybrid Control With Modal Mass Decoupling and Bilateral Communication Between Different Structures." *IEEE Transactions on Industrial Informatics,* **7**(2), may, pp. 266–276. 61

[52] Alberts, T., and Soloway, D., 1988. "Force control of a multi-arm robot system." In *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, IEEE Comput. Soc. Press, pp. 1490–1496. 61

[53] Hayati, S., 1986. "Hybrid position/Force control of multi-arm cooperating robots." In *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, Vol. 3, Institute of Electrical and Electronics Engineers, pp. 82–89. 61

[54] Yoshikawa, T., and Zheng, X.-Z., 1993. "Coordinated Dynamic Hybrid Position/Force Control for Multiple Robot Manipulators Handling One Constrained Object." *The International Journal of Robotics Research,* **12**(3), jun, pp. 219–230. 61

[55] Uchiyama, M., and Dauchez, P., 1988. "A symmetric hybrid position/force control scheme for the coordination of two robots." In *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, IEEE Comput. Soc. Press, pp. 350–356. 61

[56] Derventzis, C., and Davison, E., 1992. "Robust motion/force control of cooperative multi-arm systems." In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, IEEE, pp. 2230–2237. 61

[57] Yan, L., Mu, Z., Xu, W., and Yang, B., 2016. "Coordinated compliance control of dual-arm robot for payload manipulation: Master-slave and shared force control." In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, IEEE, pp. 2697–2702. 61

[58] Schneider, S. A., and Cannon, R. H., 1992. "Object impedance control for cooperative manipulation: Theory and experimental results." *IEEE Transactions on Robotics and Automation,* **8**(3), pp. 383–394. 61

[59] Caccavale, F., Chiacchio, P., Marino, A., and Villani, L., 2008. "Six-dof impedance control of dual-arm cooperative manipulators." *IEEE/ASME Transactions On Mechatronics,* **13**(5), pp. 576–586. 61

[60] Erhart, S., Sieber, D., and Hirche, S., 2013. "An impedance-based control architecture for multi-robot cooperative dual-arm mobile manipulation." In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, IEEE, pp. 315–322. 61

[61] Ren, Y., Liu, Y., Jin, M., and Liu, H., 2016. "Biomimetic object impedance control for dual-arm cooperative 7-dof manipulators." *Robotics and Autonomous Systems,* **75**, pp. 273–287. 61

[62] Rus, D., and Tolley, M. T., 2015. "Design, fabrication and control of soft robots." *Nature,* **521**(7553), pp. 467–475. 62

[63] Li, T., Nakajima, K., Calisti, M., Laschi, C., and Pfeifer, R., 2012. "Octopus-inspired sensorimotor control of a multi-arm soft robot." In *Mechatronics and Automation (ICMA), 2012 International Conference on*, IEEE, pp. 948–955. 62

[64] Cianchetti, M., Calisti, M., Margheri, L., Kuba, M., and Laschi, C., 2015. “Bioinspired locomotion and grasping in water: the soft eight-arm octopus robot.” *Bioinspiration & biomimetics,* **10**(3), p. 035003. 62

[65] Shepherd, R. F., Ilievski, F., Choi, W., Morin, S. A., Stokes, A. A., Mazzeo, A. D., Chen, X., Wang, M., and Whitesides, G. M., 2011. “Multigait soft robot.” *Proceedings of the National Academy of Sciences,* **108**(51), pp. 20400–20403. 62

[66] Seok, S., Onal, C. D., Cho, K.-J., Wood, R. J., Rus, D., and Kim, S., 2013. “Meshworm: a peristaltic soft robot with antagonistic nickel titanium coil actuators.” *IEEE/ASME Transactions on mechatronics,* **18**(5), pp. 1485–1497. 62