

**COOPERATIVE MANIPULATION STRATEGIES FOR MULTI-ROBOT  
COLLABORATION**

A Dissertation  
Presented to  
The Academic Faculty

By

Victor Oluwatobi Aladele

In Partial Fulfillment  
of the Requirements for the Degree  
Doctor of Philosophy in the  
College of Engineering  
Department of Electrical and Computer Engineering

Georgia Institute of Technology

December 2022

© Victor Oluwatobi Aladele 2022

# COOPERATIVE MANIPULATION STRATEGIES FOR MULTI-ROBOT COLLABORATION

Thesis committee:

Dr. Seth Hutchinson (Advisor)  
School of Interactive Computing  
*Georgia Institute of Technology*

Dr. Charles Kemp  
Department of Biomedical Engineering  
*Georgia Institute of Technology*

Dr. Magnus Egerstedt  
Department of Electrical Engineering and  
Computer Science  
*University of California Irvine*

Dr. Danica Kragic Jensfelt  
School of Electrical Engineering and  
Computer Science  
*Royal Institute of Technology (KTH)*

Dr. Ye Zhao  
Department of Mechanical Engineering  
*Georgia Institute of Technology*

Date approved: August 16<sup>th</sup>, 2022

The true sign of intelligence is not knowledge but imagination.

*Albert Einstein*

For my dad, Sunday Aladele

## ACKNOWLEDGMENTS

I would like to thank my advisor, Professor Seth Hutchinson, for his guidance and mentorship in getting this far in my PhD. I would like to thank Professor Danica Kragic for hosting me in her lab in Sweden, where I got the opportunity to broaden my knowledge on reinforcement learning and learn how RL can be applied to cooperative manipulation.

Special thanks to Christian Pek, Hang Yin, Alberta Longhini and Alfredo Reichlin, for their collaboration while I was a visiting doctoral student at KTH, Sweden. Chris helped to create some of the figures used in this thesis, including figures Figure 4.1, Figure 5.1 and Figure 6.1. I also cannot forget to thank my esteemed lab mates who have been supportive in every way possible; Sergio Aguilera, Andrew Messing, Bruce Wingo, Kilanga Monga (Dan), Muhammad Murtaza, thank you!

I would also like to thank members of my committee for taking time out of their busy schedule to review my thesis and provide invaluable feedback. Finally, I would like to also thank friends and family who have supported me in every little and big way.

## TABLE OF CONTENTS

<b>Acknowledgments</b> . . . . .	v
<b>List of Tables</b> . . . . .	ix
<b>List of Figures</b> . . . . .	x
<b>List of Acronyms</b> . . . . .	xiii
<b>Summary</b> . . . . .	xiv
<b>Chapter 1: Introduction and Background on Cooperative Manipulation</b> . . . . .	1
1.1 Cooperative Manipulation . . . . .	1
1.1.1 Manipulator Dynamic Model . . . . .	4
1.1.2 External Torque estimation . . . . .	6
1.1.3 Cooperative Manipulation . . . . .	7
1.1.4 Multi-arm Cooperative Manipulation Model . . . . .	8
<b>Chapter 2: Methodology for Internal Stress Loading Approach</b> . . . . .	10
2.1 Collision Reaction Through Internal Stress Loading . . . . .	10
2.1.1 Problem Statement . . . . .	10
2.1.2 Internal Stress Loading . . . . .	10
2.1.3 Control Derivation . . . . .	12

2.1.4	Impedance-based tuning factor . . . . .	16
<b>Chapter 3: Experiments and Results for Internal Stress Loading Approach . . .</b>		<b>21</b>
3.1	Evaluation Approach . . . . .	21
3.1.1	Simulation Experiments . . . . .	24
3.1.2	Hardware Experiments . . . . .	28
<b>Chapter 4: Introduction and Background for Residual Reinforcement Learning</b>		<b>47</b>
4.1	Deep Reinforcement Learning . . . . .	47
4.2	Residual Reinforcement Learning . . . . .	48
<b>Chapter 5: Methodology for Residual Reinforcement Learning Approach . . . .</b>		<b>51</b>
5.1	Residual Reinforcement Learning . . . . .	51
5.1.1	Problem Statement . . . . .	51
5.1.2	Compensating for Errors in Cooperative Manipulation . . . . .	52
5.1.3	Learning a Residual . . . . .	54
5.1.4	Neural Network Architecture and Reward Function . . . . .	55
<b>Chapter 6: Experiments and Results for Residual Reinforcement Learning Ap- proach . . . . .</b>		<b>57</b>
6.1	Evaluation approach . . . . .	57
6.1.1	Gaussian Noise Environment . . . . .	60
6.1.2	Human-teleoperated Data Replay Environment . . . . .	62
<b>Chapter 7: Discussion . . . . .</b>		<b>68</b>
7.1	Collision Reaction via Internal Loading . . . . .	68

7.2	Compensating for Cooperative Model Errors via Residual RL . . . . .	70
7.3	Comparison between the Internal Stress Loading approach and the Residual RL approach . . . . .	71
<b>Chapter 8: Conclusion and Future Work . . . . .</b>		<b>73</b>
<b>Appendices . . . . .</b>		<b>76</b>
	Appendix A: Experimental Equipment . . . . .	77
	Appendix B: Semi-parametric Approaches to Learning Inverse Dynamics Control	84
	Appendix C: An Adaptive Cooperative Manipulation Control Framework for Multi-Agent Disturbance Rejection . . . . .	88
<b>References . . . . .</b>		<b>91</b>
<b>Vita . . . . .</b>		<b>103</b>

## LIST OF TABLES

6.1	Table showing list of parameters and their values . . . . .	58
6.2	Mean object pose error norm - Gaussian Noise (A lower value implies better performance) . . . . .	62
6.3	Mean object pose error norm - Human-teleoperated Data (A lower value implies better performance) . . . . .	66

## LIST OF FIGURES

1.1	Two kuka iiwa14 arms cooperatively lifting an object . . . . .	5
2.1	Flowchart showing the arm control hierarchy . . . . .	16
2.2	Plot of $\beta(j)$ vs $\gamma_j$ with decay constant $\lambda = 0.1$ [36] . . . . .	19
3.1	Two kuka iiwa7 arms cooperatively lifting a 0.8m rod in simulation [36] . . . . .	25
3.2	Plot of COM pose error for zero compensation, partial compensation and full compensation [36] . . . . .	26
3.3	Plot of compensation-factor ( $\beta$ ), normalized-COM-squared error and the normalized-squared-full-compensation-force/torque for the compensating arm ( $arm-2$ ). Decay constant $\lambda = 0.05$ [36] . . . . .	27
3.4	Schematic of hardware experiment setup . . . . .	30
3.5	Schematic of eight 3D coordinate frames in our system setup. . . . .	32
3.6	Plot of COM pose error for zero compensation, partial compensation and full compensation; hardware experiments . . . . .	37
3.7	Plot of compensation-factor ( $\beta(j)$ ), normalized-COM-squared error and the normalized-squared-full-compensation-force/torque for the compensating arm ( $arm-2$ ). . . . .	38
3.8	A human, introducing disturbance along the links of $arm-2$ . . . . .	40
3.9	Plot of COM pose error for zero compensation, partial compensation and full compensation; hardware experiments. In this case, the disturbance is introduced along the links of the arm. . . . .	42

3.10	Plot showing the end effector wrenches (in <i>Newtons</i> and $N/m$ ) that are produced by applying disturbances along the links of <i>arm-2</i> . . . . .	43
3.11	Plot of compensation-factor ( $\beta(j)$ ), normalized-COM-squared error and the normalized-squared-full-compensation-force/torque for the compensating arm ( <i>arm-2</i> ). In this case, the disturbance is introduced along the links of the arm. . . . .	45
4.1	RL model applied to the protagonist, in a dual-arm cooperative setup. See subsection 5.1.1 for the definition of the <i>protagonist</i> and <i>antagonist</i> . . . . .	49
5.1	Error in cooperative manipulation [51] . . . . .	53
5.2	Neural network model for Gaussian noise experiment . . . . .	56
5.3	Neural network model for human-teleop experiment . . . . .	56
6.1	Two manipulators that cooperatively lift a 0.8m rod from an initial pose to a desired goal pose. The protagonist manipulator tries to compensate for errors of the antagonist manipulator [51]. . . . .	58
6.2	Policy evaluation for the standard controller and the residual learner in the presence of Gaussian noise. . . . .	61
6.3	Policy evaluation on trajectory 1 (Traj-1) for both the standard controller and the residual learner. . . . .	64
6.4	Policy evaluation on trajectory 5 (Traj-5) for both the standard controller and the residual learner. . . . .	65
6.5	Plot of the end effector 3D positions for the test trajectories. Trajectories start at the bottom left and end at the top right. . . . .	67
A.1	Figure showing the error output when the lock error behavior happens . . . .	82
A.2	Figure showing configuration of robot when the lock error behavior happens	83
B.1	Golem Krang [96] . . . . .	86

C.1 Cooperative manipulation test bench system with two robot manipulators  
[105] . . . . . 89

## **LIST OF ACRONYMS**

**COM** Center of Mass

**DART** Dynamic Animation and Robotics Toolkit

**DOF** Degrees of Freedom

**FRI** Fast Research Interface

**ROS** Robot Operating System

## SUMMARY

The demand for robots that are capable of performing complex tasks has soared in recent years; research interests in multi-robot systems have also risen. Despite significant advances in single-arm manipulation, the need for cooperative manipulation cannot be overlooked. This thesis presents strategies that enable multiple robots to perform collaborative tasks. More specifically, this thesis will address cooperative manipulation under the possibility of external disturbance. I propose two approaches: an impedance-based approach and a residual-reinforcement-learning-based approach.

With respect to the impedance-based approach, I will focus on the concept of applied internal stress, on the jointly manipulated object, and how internal stress can be leveraged as a compensation mechanism for disturbance on a cooperative manipulation setup. I will present an impedance-based strategy that is used to determine how much compensation should be applied to the system.

Additionally, I will present a decentralized approach to cooperative manipulation that is based on a residual reinforcement learning scheme. Residual reinforcement learning involves the superposition of a learned policy with a standard controller, in such a way that the input to the learned policy is informed by the output of the standard controller, or vice versa. I will show how a robot can compensate for unexpected partner behaviors without communicating with its partner(s).

Finally, experimental demonstrations both in simulation and on the physical system will be presented. Although this work describes a multi-robot scenario, the experimental validation will be on a two-robot system. The physical system also includes mobile platforms that extend the workspace of the manipulators shown in simulation. The contribution of this thesis can be summarized as follows:

1. A collision reaction strategy based on internal stress loading.
2. An impedance-based collision reaction strategy.

3. Hardware demonstration of the above contributions.
4. A cooperative manipulation approach via residual reinforcement learning.

## CHAPTER 1

### INTRODUCTION AND BACKGROUND ON COOPERATIVE MANIPULATION

In this chapter, we will introduce the background of cooperative manipulation and discuss other related works in this domain. we will present the key equations, including the grasp matrix formulation, that make up the foundation of cooperative manipulation.

#### 1.1 Cooperative Manipulation

Cooperative manipulation offers many advantages over single-arm manipulation. However, this comes at a cost of added complexity, both in modeling and control of multi-arm systems. Cooperative manipulation has gained significant attention in the last couple decades. With increasing interest in making robots perform more complex tasks, it has become obvious that certain tasks cannot be performed by single-arm robots, either due to payload limitations or the geometry of the object to be manipulated. Therefore, there has been an increased research effort in developing the theoretical background for multi-arm systems [1, 2, 3, 4, 5]. Several areas have benefited and could continue to benefit from the applications of cooperative manipulation, including: space missions, medical surgery, eldercare and industrial manufacturing [6, 7, 8].

When multiple robots collaborate to manipulate a common object, the load is distributed among the robots, based on various properties; these properties include, but are not limited to, the load carrying capacity of each robot. The work done in [3] presents a unique perspective on load distribution for the purpose of avoiding internal wrenches. Several works have studied load distribution strategies that avoid internal forces on the manipulated object. [1] claims that there is a unique *non-squeezing* load distribution that avoids internal loading of the object, but the authors in [3] argue otherwise. According to [3], a disregard for the kinematic constraint leads to the result obtained by [1]. The authors

of [3] also show that heterogeneous load distribution does not necessarily result in internal loading of the object.

Often in robotic manipulation, collision is inevitable, especially in unstructured environments. Hence, [9] proposed a method for safe collision reaction to external forces along the links of a seven degree-of-freedom (7DOF) robot arm. The method proposed in [9] projects the reaction torques into the nullspace of the primary task, which in this case, involves tracking of an end effector trajectory. This method exploits the redundancy of the robot arm with respect to the task definition.

Internal stress is often avoided or controlled while performing cooperative manipulation. In this thesis, we will take the reverse approach by using the tensile strength of a manipulated object to an advantage. We present a novel method of reacting to collisions along the links of a robot arm by transmitting the disturbance torques through the manipulated object and thereby sharing the effect of this disturbance with the other robots in the cooperative manipulation system. One advantage of this collision reaction strategy over the proposed method in [9] is that in [9], beyond certain thresholds, the task can no longer be accomplished. However, instead of compromising the task, our method proposes *transferring* the external disturbance to the rest of the cooperative manipulation system in such a way that the primary task is not affected. Moreover, our proposed strategy is expected to work with manipulators with less than seven degrees-of-freedom by virtue of the extra degrees-of-freedom provided by multiple manipulators.

Similarly to the work presented in this thesis, [10] compensates for external forces on the grasped object by modifying the contact forces. However, their approach was not designed to handle external disturbances on the robot itself. Whereas, our approach is designed to compensate for collisions along the links of the robot. The authors of [10] presented a unique cooperative manipulation strategy that is built on separating the motion control from the contact force control. The motion control torques are projected into the nullspace of the constraint Jacobian. The constraint Jacobian by definition is related to the

grasp matrix. Therefore, [10] chose the constraint Jacobian as the nullspace projection of the grasp matrix. i.e  $J_c = (I - G^+G)$ . Where  $J_c$  is the constraint jacobian,  $G$  is the grasp matrix and  $G^+$  is the Moore-Penrose inverse of the grasp matrix. According to [11] the nullspace of the grasp matrix generates internal forces on the object. Hence, [10] was able to ensure only internal forces were allowed in the constraint space.

Another contribution of [10] is the task space impedance control which ensured the object responded to external disturbances with a defined impedance behavior. Impedance control dates back to works by [12, 13], but these works have focused on single-arm manipulator systems. Other works that have introduced impedance control schemes into the framework of cooperative manipulation include [14, 15]. There are several arguments for incorporating impedance control in the framework of multi-arm systems. If the manipulated object is expected to interact with the environment, impedance control must be applied to properly define the interaction forces between the object and the environment, just like it is the case in single-arm manipulation of an object. Another application of impedance control in cooperative manipulation is ensuring a robust *leader-follower* structure that is inherent in a cooperative system [16, 17].

Considering the fact that conventional manipulators are designed to have multiple controllable joints, multi-arms systems are inherently redundant. And this redundancy can be exploited to solve multiple objectives in parallel with the task completion. Much research has been focused on determining optimal load distribution strategies based on several objective functions, some of which include manipulability [18], energy consumption and joint torque minimization [19].

In this thesis, we will discuss disturbance compensation via internal stress loading as a redundancy resolution scheme. Although, unlike the many other works that have used the nullspace of the grasp matrix as a projector to resolve redundancy, we have used a simple method of compensation. This compensation implicitly ensures that for a zero motion command, the net-wrench applied to a manipulation system in the presence of external

disturbance, lies in the nullspace of the grasp matrix.

In [20], the authors present a framework for coordinating multiple robots for cooperative manipulation tasks, in which vision is used for defining relative pose, while maintaining formation. According to [20], the ability to maintain a prescribed formation allows the robots to ascertain a pickup pose, and to "flow" the formation to a desired position. Similarly, the authors in [21] used potential field controllers in hierarchical composition form to create a formation that will trap the object, and then transport the object to the desired destination. No doubt, certain grasp points are bound to yield better task performance. Hence, careful choices must be made to determine appropriate grasp points for each agent involved in the manipulation process. Several contributions have already been made within the realm of single-arm pick and place tasks [22, 23]. In this work, deep learning frameworks were applied to plan robust grasps using synthetic point clouds data. Although there have been a few studies that develop schemes for planning grasps, not as many of these have been done within the domain of cooperative manipulation.

When more than one manipulator jointly grasp an object, a system that can be viewed as a closed chain mechanism is formed [2]. [24] proposed a virtual linkage model to describe the closed chain mechanism. A different approach presents the dynamics of the cooperative system as a solution of resultant forces acting on the object in the object frame [19]. This approach makes the inverse dynamics problem decoupled at the joint level. Hence, we can compute the joint torques required to obtain a desired end effector wrench for each manipulator.

### 1.1.1 Manipulator Dynamic Model

The mathematical dynamic model of a single arm with  $N$  joints is given by,

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau \quad (1.1)$$

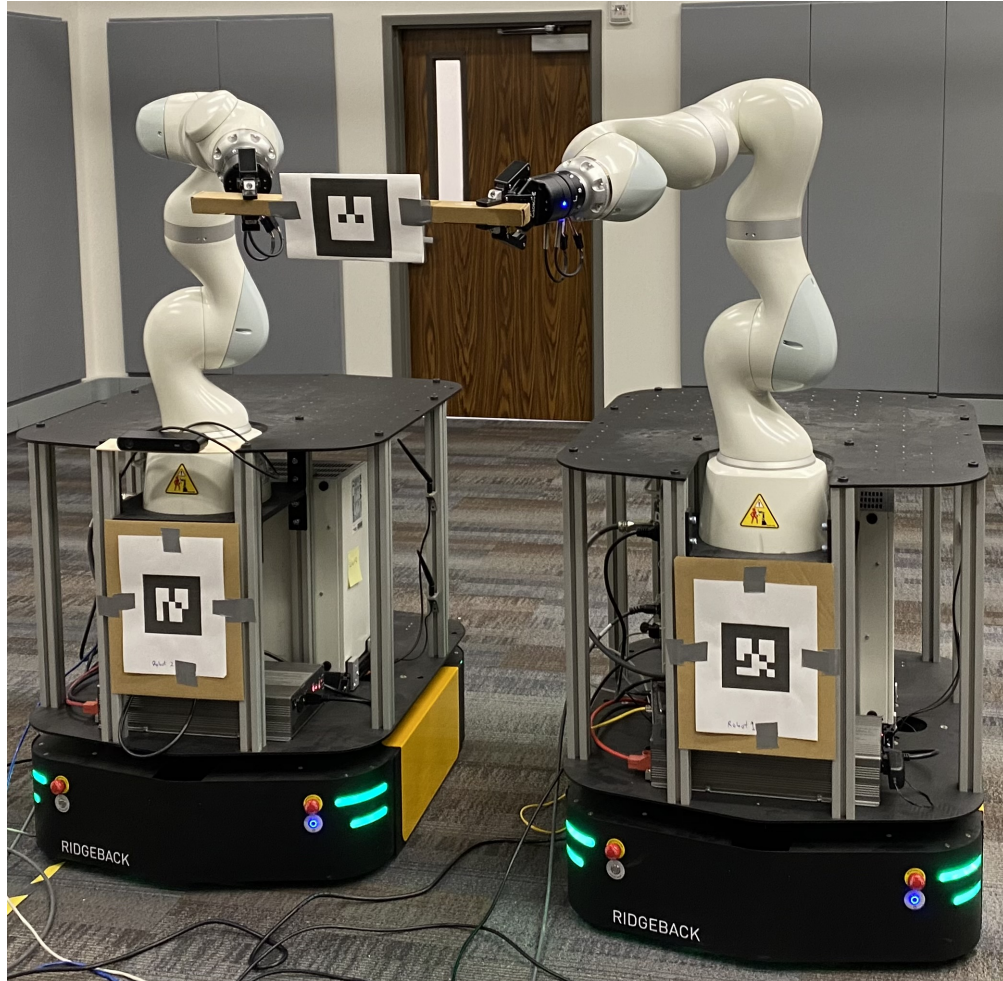


Figure 1.1: Two kuka iiwa14 arms cooperatively lifting an object

where  $M \in \mathfrak{R}^{N \times N}$  is the symmetric and positive-definite inertia matrix,  $C \in \mathfrak{R}^N$  is the coriolis/centripetal matrix,  $g$  the gravity vector,  $q \in \mathfrak{R}^N$  represents the joint positions and  $\tau \in \mathfrak{R}^N$  is the applied joint torques [25]. In the presence of external disturbances, (Equation 1.1) is modified to incorporate the external torques

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau + \tau_{ext} := \tau_m \quad (1.2)$$

where  $\tau_{ext}$  is the external joint torques that are induced by the disturbance on the robot, and  $\tau_m$  is the total joint torques resulting from the external joint torques plus the applied joint torques  $\tau$ .

### 1.1.2 External Torque estimation

In this section, we introduce the sensorless collision torque estimation that has been derived in [26, 27]. This method uses a momentum observer to estimate the collision torque between the links along the body of a 7DOF arm and the environment.

The authors of [28] described how to estimate the external torque  $\tau_{ext}$  due to collision via a momentum observer. The output of the observer  $r(t)$ , known as the residual vector, gives the estimate of the external torque.

$$r(t) = K_I \left[ p(t) - \int_0^t (\tau + C^T(q, \dot{q}) - g(q) + r) ds \right] \quad (1.3)$$

where  $r \in \mathfrak{R}^N$ , with  $r(0) = 0$ .  $K_I > 0$  is an appropriately chosen diagonal observer gain matrix. From (Equation 1.3), the dynamics of  $r$  can be defined as

$$\dot{r} = -K_I(r + \tau_{ext}) \quad (1.4)$$

The details of the derivation of (Equation 1.3) and (Equation 1.4) can be found in [9, 28]. This method for estimating the external joint torques is quite attractive as it only re-

quires proprioceptive sensors. For the purpose of simplicity, we consider torque estimation for collision on a single link at a time (see subsection 3.1.2).

### 1.1.3 Cooperative Manipulation

The forces acting on a grasped object by two manipulators are combinations of motion inducing forces/wrench, internal forces/wrench and possibly, external disturbances. As expected, the motion inducing forces on the manipulated object cause the movement of the object along a given trajectory. A frame is placed on the object. The origin of this frame, which is often located at the object's center of mass (COM) coordinate, is selected as the point to track the desired trajectory. We call this frame, the object frame, or the COM frame. The COM frame has an orientation that is expressed with an Euler representation. In simulation the orientation is given by the urdf, while on the hardware, the orientation is specified by the orientation of the AR tag that is placed on the object.

In a cooperative manipulative task involving  $K$  manipulators grasping a rigid object, with the assumptions of a rigid grasp by the end-effectors, the relationship between the applied end-effector wrenches and the resulting wrench at the origin of the COM frame is given by:

$$F_o^d = GF^d \quad (1.5)$$

where  $F_o^d = (f_o^{dT}, t_o^{dT})^T \in \mathfrak{R}^6$  is the target wrench [29] that needs to be applied to the COM of the object to track the desired trajectory  $x_o^d \in SE(3)$ .  $F^d = (F_1^{dT}, \dots, F_i^{dT}, \dots, F_K^{dT})^T \in \mathfrak{R}^{6K}$  is the vector of end-effector wrenches, with  $F_i^d = (f_i^{dT}, t_i^{dT})^T$  and force-moment  $f_i^d, t_i^d \in \mathfrak{R}^3$  for  $i \in 1, \dots, K$ .  $G \in \mathfrak{R}^{6 \times 6K}$  is the grasp matrix defined by the contact points  $l_i \in \mathfrak{R}^3$  of each end-effector, with respect to the object frame [29]. When the grasp frame of the end effectors have the same orientation as the COM frame, the grasp matrix  $G$  can

be computed as:

$$G = \begin{bmatrix} I_3 & 0_3 & \dots & I_3 & 0_3 & \dots & I_3 & 0_3 \\ S(l_1) & I_3 & \dots & S(l_i) & I_3 & \dots & S(l_K) & I_3 \end{bmatrix} \quad (1.6)$$

where  $S(l_i) \in \mathfrak{R}^{3 \times 3}$  is the skew-symmetric matrix

$$S(l_i) = \begin{bmatrix} 0 & -l_z & l_y \\ l_z & 0 & -l_x \\ -l_y & l_x & 0 \end{bmatrix}$$

Given the desired wrench of the object  $F_o^d$ , to calculate the required wrench at the grasp locations, (Equation 1.5) is inverted to obtain the generalized inverse  $G^+$  of the grasp matrix  $G$ . This leads to the inverted equation

$$F^d = G^+ F_o^d \quad (1.7)$$

#### 1.1.4 Multi-arm Cooperative Manipulation Model

We begin by discussing the dynamics of the object being manipulated. When describing the dynamics of a rigid object, we select a point of interest on the object. The point of interest is typically chosen as the center of mass of the object. Therefore, the operational space dynamics is given by [30]

$$M_o(x_o)\ddot{x}_o + C_o(x_o, \dot{x}_o)\dot{x}_o + g_o = F_o \quad (1.8)$$

where  $M_o$  is the inertia mass matrix of the object,  $C_o$  is the coriolis matrix,  $g_o$  is the gravity vector and  $x_o$  is the pose of the object's COM frame.

For  $K$  manipulators in contact with a grasped object, the following equation describes the relationship between the joint torques and the wrench  $F_i$  (w.r.t the manipulator base

frame), applied at the end effector of the  $i$ -th manipulator [30].

$$M_i(q_i)\ddot{q}_i + C_i(q_i, \dot{q}_i)\dot{q}_i + g_i(q_i) = \tau_i + J_i^T F_i \quad (1.9)$$

where  $J_i$  and  $\tau_i \in \mathfrak{R}^N$  are the manipulator jacobian and torques, respectively, of the  $i$ -th manipulator. With collision on the links of the  $i$ -th manipulator, (Equation 1.9) can be modified to yield

$$M_i(q_i)\ddot{q}_i + C_i(q_i, \dot{q}_i)\dot{q}_i + g_i(q_i) = \tau_i + J_i^T F_i + \tau_i^{ext} \quad (1.10)$$

where  $\tau_i^{ext} = J_i^T F_i^{ext}$  is the estimated torque due to the collision on the links of the robot. This torque can be estimated using the method discussed in section subsection 1.1.2. Combining the torques needed to induce motion on the grasped object with equation (Equation 1.10), we obtain

$$M_i(q_i)\ddot{q}_i + C_i(q_i, \dot{q}_i)\dot{q}_i + g_i(q_i) = \tau_i + J_i^T F_i + J_i^T F_i^{ext}. \quad (1.11)$$

The next two chapters of this thesis will present the formulation of the internal stress loading approach and the experiments that were conducted to evaluate the proposed approach. The following three chapters (chapter 4, chapter 5, chapter 6) will cover the residual reinforcement learning approach to compensating for disturbance. The last two chapters of this thesis will discuss some more details of the experiment process and will offer insights on how to extend this work in the future.

## CHAPTER 2

### METHODOLOGY FOR INTERNAL STRESS LOADING APPROACH

In this chapter, we will provide details of the proposed internal stress loading approach. I will present the mathematical formulations, with pictorial representations of our framework, for easy understanding. We will also reference certain design choices and our motivations for making such choices.

#### 2.1 Collision Reaction Through Internal Stress Loading

In this section, we will present the aforementioned internal stress loading strategy. We will enumerate some of the assumptions that need to hold for this framework to succeed. We will present the impedance-based structure that is used to regulate the amount of internal stress applied on the object.

##### 2.1.1 Problem Statement

When collision happens along one or more of the links of manipulators involved in a cooperative manipulation task, the motion of the object might be significantly impacted due to improper wrenches at the end effectors. To address this problem, we have developed an internal loading strategy to compensate for such disturbances. This compensation strategy ensures that the impact of the collision on the object's desired trajectory is minimized or even eliminated.

##### 2.1.2 Internal Stress Loading

The cooperative system, which we described in [31], is comprised of two manipulators (i.e.  $K = 2$ ), and we employ a similar setup in this thesis.

Out of the infinitely many solutions to equation (Equation 1.7), we select an appropriate end effector wrench, to be applied by the compensating arm, to compensate for the external wrench on the colliding arm. The aim is to preserve the tracking of the desired trajectory of the object's COM. The combination of collision wrenches and compensation wrenches results in internal stress on the object. Therefore, the proposed compensation scheme requires that the following assumptions hold true [29]:

- Internal force loading of the object is permissible (i.e the load can withstand squeezing/bending forces).
- The manipulated object is rigidly grasped by both end effectors.
- Both arms share information about the estimated collision torques on either arm.

For the rest of this thesis, we define the *colliding-arm* as *arm-1* and the *compensating arm* as *arm-2*. The proposed compensation scheme selects the appropriate end-effector wrench for *arm-2* to counter the effect of the external disturbance due to collision on arm-1.

When a collision occurs on arm-1, a wrench,  $h_1^{ext} = (f_1^{ext}, m_1^{ext})^T$  expressed in the object frame, can be felt at the end effector of arm-1. To compensate for the force component  $f_1^{ext}$ , *arm-2* applies a force  $f_2^c$  defined by

$$f_2^c = -f_1^{ext} \quad (2.1)$$

The torque component ( $m_1^{ext}$ ) can be compensated for by applying a compensating moment  $m_2^c$  with *arm-2* [31]

$$m_2^c = -m_1^{ext} \quad (2.2)$$

However, we still need to compensate for an additional effect of  $f_1^{ext}$  on the manipulation system, which is an induced torque about the COM [29]. Therefore, to fully compensate for the force component of the disturbance  $f_1^{ext}$ , it is not enough to simply negate

the force  $f_1^{ext}$  and apply it as a compensating force from *arm-2*, as shown in equation (Equation 2.1). The induced torque due to  $f_1^{ext}$

$$m_1^{ind} = S(\rho_1)f_1^{ext} \quad (2.3)$$

must be compensated for by modifying equation (Equation 2.2) to include this induced torque  $m_1^{ind}$ , where  $\rho_1 \in \mathbb{R}^3$  is the displacement from the origin of the end effector frame of *arm-1* to the origin of the COM frame [29].

Lastly, there is a torque induced due to the compensation force  $f_2^c$  in (Equation 2.1) which also needs to be accounted for. This induced torque can be calculated similarly to (Equation 2.3)

$$m_2^{ind} = S(\rho_2)f_2^c := -S(\rho_2)f_1^{ext} \quad (2.4)$$

where  $\rho_2 \in \mathbb{R}^3$  is the displacement from the origin of the end effector frame of *arm-2* to the origin of the COM frame. Therefore, a total torque compensation required from *arm-2* is given by

$$m_2^c = -(m_1^{ext} + m_1^{ind} + m_2^{ind}) \quad (2.5)$$

This procedure works exactly the same way if the collision is on *arm-2* instead of *arm-1*. Likewise, when both arms experience collision, the compensation would be performed by both arms, with one arm compensating for collision on the other arm and vice versa. This could result in more or less stress on the object, depending on the direction of the resulting end-effector wrenches due to the collisions [31]. It is important to note that the colliding arm does not compensate for collision on itself. Instead it is the other arms that are not involved in the collision that contribute to the compensating wrenches.

### 2.1.3 Control Derivation

The *Computed-Torque* method [32] is a well known control technique for nonlinear systems. This control technique renders the system linear through a feedback linearization

scheme. The dynamics of the cooperative manipulation system is nonlinear. Therefore, we derive the control law for the manipulated object, as well as, the robots performing the manipulation using the computed-torque control method. The total wrench to be applied to both end effectors is calculated as

$$h^d = G^+ h_o^d + \mathcal{H} + \alpha \mathcal{N} \quad (2.6)$$

where  $\mathcal{H} \in \mathbb{R}^{12}$  and  $\mathcal{N} \in \mathbb{R}^{12}$  are vectors of collision wrenches and compensation wrenches, respectively, for both end effectors;  $G$  is grasp matrix and  $h_o^d$  is the wrench that must be applied at the object's COM frame in order to track the desired object trajectory.

$$\mathcal{H} = \begin{bmatrix} h_1^{ext} \\ h_2^{ext} \end{bmatrix}, \quad \mathcal{N} = \begin{bmatrix} f_1^c \\ m_1^c \\ f_2^c \\ m_2^c \end{bmatrix} \quad (2.7)$$

$\alpha \in [0, 1]$  is a tuning factor that defines how much compensation is applied by the *compensating arm*;  $h_1^{ext} = (f_1^{ext}, m_1^{ext})^\top$  and  $h_2^{ext} = (f_2^{ext}, m_2^{ext})^\top$ . It is important to note that when  $K > 2$ , the sizes of the vectors in (Equation 2.7) are extended in proportion to the number of additional manipulators.

As it has been shown so far in this section (section 2.1), every component of the compensating term acts as a cancelling term for its collision counterpart. Therefore, we argue that the combination of the compensation wrenches and the collision wrenches, does in fact, result in wrenches that lie in the nullspace of the grasp matrix. Again, it is critical to reiterate that, the compensation wrench alone does not lie in the nullspace of the grasp matrix. It is the combination of the compensation wrench and the collision wrench that lies in the nullspace of the grasp matrix. Hence, when  $\alpha = 1$  and full compensation wrenches

are generated by the *compensating arm*

$$G(\mathcal{H} + \alpha\mathcal{N}) = 0 \tag{2.8}$$

When  $\alpha = 0$ , no compensation is performed. There are many reasons why this tuning factor might be useful. Joint-torque limitations on the *compensating arm*, in this case arm-2, would mean that the required compensation torques to fully compensate for collision on arm-1 cannot be generated by arm-2 in some cases. However, although arm-2 may not have the capacity to generate the joint torques needed to fully compensate for the disturbance on arm-1, arm-2 could still perform partial compensation. Hence, we can use the tuning factor to control how much compensation joint torque is demanded from arm-2.

Finally, the assumption that internal load stressing of the object is permissible holds true only up to a certain extent. Beyond a certain stress threshold, depending on the material properties of the object, the object could be permanently deformed or completely damaged. Therefore, the tuning factor serves as a parameter ( $\alpha \in [0, 1]$ ) to regulate the amount of applied compensation wrench, consequently limiting the internal stress due to the compensating wrench. In most cases, the object might be able to withstand some internal stress, but not the total internal stress required to fully compensate for the collision. For instance, a glass tray that is co-manipulated by two manipulators might break when full compensation is applied, but a plastic tray might withstand the same level of internal stress without being significantly deformed. Therefore, the tuning factor can be used as a constraint variable to regulate the amount of internal stress applied on the object as a result of compensating wrenches. Hence, having knowledge of the stress limit of the grasped object could help reduce the effect of the collision on the desired trajectory, while not exceeding the stress limit of the object [31].

The desired COM wrench is calculated using the computed-torque control method [32,

33].

$$h_o^d = M_o \ddot{x}_o^r + C_o + g_o \quad (2.9)$$

where,

$$\ddot{x}_o^r = \ddot{x}_o^d - K_v(\dot{x}_o - \dot{x}_o^d) - K_p(x_o - x_o^d) \quad (2.10)$$

is the commanded COM reference acceleration,  $K_p$  and  $K_v$  are the stiffness and damping matrices, respectively. Note that  $x_o \in \mathfrak{R}^6$  represents the pose of the object. We have chosen an Euler representation for the orientation of the object (i.e.  $x_o = [x, y, z, \theta, \phi, \psi]$ ). Assuming that the desired trajectory  $x_o^d, \dot{x}_o^d, \ddot{x}_o^d$  has been given, we substitute equation (Equation 2.9) into (Equation 2.6) to obtain the control law

$$h^d = G^+ (M_o \ddot{x}_o^r + C_o + g_o) + \mathcal{H} + \alpha \mathcal{N} \quad (2.11)$$

where  $h^d = (h_1^{dT}, h_2^{dT})^T$ .  $h_1^d$  is passed on to the low-level controller for arm-1 and  $h_2^d$  is passed on to the low-level controller for arm-2. Each low-level controller then uses the force-based control scheme [25, 34] in the joint space to calculate the joint torques required to obtain the corresponding end effector wrench.

According to [35], the inertia properties of the end effector rely on the dynamically consistent matrix

$$\Lambda(q) = (JM(q)J^T)^{-1} \quad (2.12)$$

resulting in manipulator control law:

$$\tau = J^T F + C(q, \dot{q}) + g(q) \quad (2.13)$$

where  $F = \Lambda(q)h^d$  is the end-effector wrench to be transformed to joint torques.

However, we observed in our simulation experiments that the presence of the inertia matrix  $M(q)$  caused aggressive behaviors of the manipulator. One possible reason for this occurrence is that model uncertainties such as joint damping and joint friction were

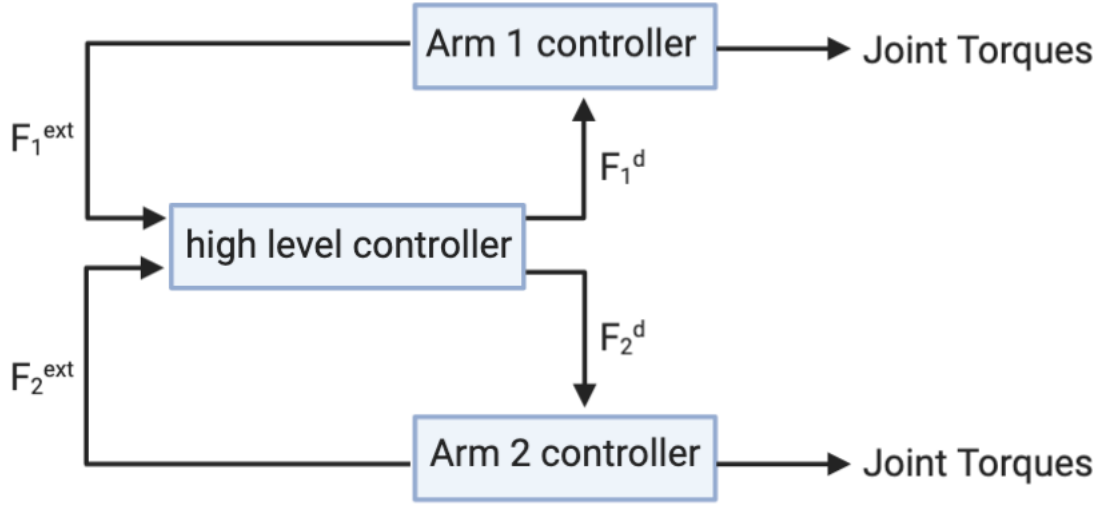


Figure 2.1: Flowchart showing the arm control hierarchy

added to the simulation model. These model uncertainties were likely to exacerbate the aggressive motion of the arm when the mass matrix was applied. Therefore, since the desired trajectory we used in our simulation experiment had low frequency accelerations and therefore a quasi-static motion, we set  $\Lambda(q) = I$  in our experiments.

The resulting control law is therefore given by

$$\tau_i = J_i^T h_i^d + C_i(q, \dot{q}) + g_i(q) \quad (2.14)$$

where  $\tau_i \in \mathbb{R}^N$  represents the joint torques for the  $i$ -th manipulator.

#### 2.1.4 Impedance-based tuning factor

In this section, we describe the work that was presented in [36], that extends the previous work of [31] by developing a unique approach to determining  $\alpha$ . Cooperative manipulation in our defined setting involves contacts and internal stress on the cooperatively manipulated object. When there is any form of contact or stress in manipulation, an impedance scheme is often preferred. Impedance control renders the behavior of the system to that

of a spring-mass damper system, thereby controlling the force-motion relationship of the system. Impedance control ensures a safer contact behavior, and in our case, a safer application of internal stress on the object.

We start by defining an impedance model on the motion of the grasped object. More specifically, we utilize a compliance model where we shape the stiffness [37] and damping of the system, but preserve the natural inertia of the system. We define the task-space compliance term  $\gamma$  as follows:

$$\gamma = -K_v^\gamma(\dot{x}_o - \dot{x}_o^d) - K_p^\gamma(x_o - x_o^d) \quad (2.15)$$

where  $K_p^\gamma$  and  $K_v^\gamma$  are the compliance gains, not to be confused with the gains  $K_p$  and  $K_v$  introduced in earlier sections. Although, both sets of gains could be equal, they do not necessarily have to be.

First, we redefine  $\alpha$  to be a diagonal matrix with dimension  $6K \times 6K$ , where  $K$  is the number of arms involved in the manipulation process.

$$\alpha = \begin{bmatrix} \beta_1 & 0_{6 \times 6} & \cdots & 0_{6 \times 6} \\ 0_{6 \times 6} & \beta_2 & \cdots & 0_{6 \times 6} \\ \vdots & \vdots & \ddots & \vdots \\ 0_{6 \times 6} & 0_{6 \times 6} & \cdots & \beta_k \end{bmatrix} \quad (2.16)$$

where  $\beta_i \in \mathbb{R}^{6 \times 6}$  is a diagonal matrix. Therefore, when  $K = 2$ ,  $\alpha$  is a  $12 \times 12$  diagonal matrix.

$$\alpha = \begin{bmatrix} \beta_1 & 0 \\ 0 & \beta_2 \end{bmatrix} \quad (2.17)$$

$$\beta_i = \begin{bmatrix} \beta_i(1) & 0 & \cdots & 0 \\ 0 & \beta_i(2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \beta_i(6) \end{bmatrix} \quad (2.18)$$

Each diagonal entry of  $\beta_i$ , i.e  $\beta_i(j)$ , defines the scalar tuning factor for the  $j$ -th component of the compensating wrench. For instance, the first diagonal entry  $\beta_2(1)$ , is the x-axis tuning factor for the force component of the compensating wrench, for robot 2.  $\beta_1(6)$ , is the z-axis tuning factor for the moment component of the compensating wrench, for robot 1. In our implementation, both diagonal entries ( $\beta_1$  and  $\beta_2$ ) have been chosen to be equal, although this is not required.

Previously, when  $\alpha$  was just a constant scalar value,  $\alpha = 0$  corresponded to no compensation and  $\alpha = 1$  to full compensation. We still want the diagonal entries of the *tuning matrix*  $\alpha$  to be between 0 and 1. At the same time, we want the values of these entries to be derived from the compliance model we have defined in equation (Equation 2.15). Therefore, we propose the formula:

$$\beta(j) = e^{-\lambda|\gamma_j(x_o, \dot{x}_o)|} \quad (2.19)$$

where  $\lambda \geq 0$  is the decay constant that provides us with a degree of freedom to indirectly tune the compliance gains. This is a parametric approach to tuning the *compensation gain matrix*, and it offers an insight into how the system is expected to behave under the compensation scheme presented in [31]. The plot in figure Figure 2.2 shows the *compensation gain* ( $\beta(j)$ ) as a function of the object's COM compliance ( $\gamma$ ). The decay constant  $\lambda$  determines the slope of  $\beta(j)$  at every point  $\gamma_j$ . It is important to note that regardless of the values of  $\lambda$  or  $\gamma_j$ , the value of  $\beta(j)$  is always within the range  $(0, 1]$ . We also note that,  $\lambda$  does not necessarily have to be a constant factor. For example,  $\lambda$  could be defined as a diagonal matrix, and we could tune the decay constant  $\lambda_j$  for each component of the compensation

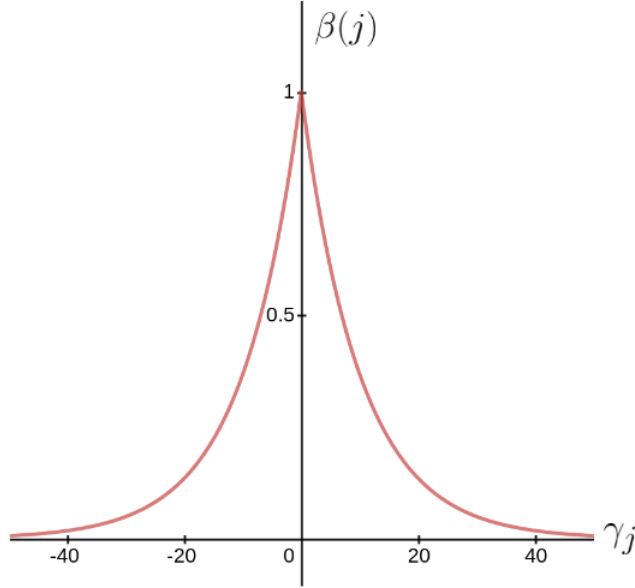


Figure 2.2: Plot of  $\beta(j)$  vs  $\gamma_j$  with decay constant  $\lambda = 0.1$  [36]

wrench. Thus,  $\lambda$  in Equation 2.19 would be replaced by  $\lambda_j$ . This would allow us to shape the compliance model of this compensation framework for each axis of the compensation wrench. However, for the purpose of this work, we have restricted  $\lambda$  to be a scalar constant.

When the collision wrench on arm-1 is high, the COM trajectory error will also be high, and by the definition of equation (Equation 2.15),  $\gamma$  will have an elevated value. Since high values of  $\gamma_j$  yield low  $\beta_2(j)$  values and vice versa for low values of  $\gamma_j$ , when high wrenches are required for compensation, the diagonal entries for the *tuning matrix* become low. This ensures that high wrenches are not applied to the object by arm-2, thereby preventing extensive internal stress on the object. The converse is true; when the required compensation wrenches are low, it is likely safe to apply the full compensation wrenches. Therefore, when  $\gamma_j$  goes to zero,  $\beta(j)$  goes to 1, and we are able to fully compensate for the collision on arm-1.

Another interesting and useful provision this kind of formulation offers is that the decay constant  $\lambda$  would fit very well in a learning scheme. In such learning scheme,  $\lambda$  could be derived based on the material properties of the grasped object. For example, objects

that can withstand higher internal stress would have lower decay constants, as this will allow a higher *compensation gain matrix*, thereby permitting higher compensation wrenches. Whereas, objects that can only withstand low internal stress would have higher decay constants, which would limit the amount of compensation wrenches applied to the object, thereby keeping the internal stress on the object within appropriate levels.

In summary, in this subsection (subsection 2.1.4), we have presented an exponential attenuation method as a means to regulate the amount of compensation wrench applied to the system. It might seem counter-intuitive to reduce the amount of compensation when the trajectory error of the object increases, as we have discussed so far in this section. However, the premise of this work is to establish a trade-off between compensating for external disturbance and restricting the amount of internal stress that is generated by the applied compensation wrench. Therefore, when large trajectory errors are detected, as expected from a large collision force, the compensation is only partial, to limit the amount of internal stress that is derived from the coupling of the compensation and collision forces.

At this point, it is important to note that the control of the internal stress on the object is only implicit. The total internal stress on the object is not actually calculated. However, it is shown in this section that our internal stress compensating scheme results in internal stress on the object. Hence, we can only control the amount of internal stress that we induce with our compensation scheme, and not the internal stress that could be caused by other factors. Hence, in this thesis, we make no claim that the total internal stress acting on the object comes from our compensation scheme.

**CHAPTER 3**  
**EXPERIMENTS AND RESULTS FOR INTERNAL STRESS LOADING**  
**APPROACH**

To validate our proposed internal stress loading approaches, we conducted several experiments both in simulation and on hardware systems. In this chapter, we present the experimental setup and the results we obtained. We discuss some of the challenges we encountered and the motivation for certain design choices. Conducting experiments both in simulation and on the hardware allowed us to understand some of the limitations of our approaches.

### **3.1 Evaluation Approach**

Works on multi-robot cooperative manipulation have evaluated their proposed cooperative strategies in various ways. The authors of [10] calculated the norm of the measured contact force with external disturbances applied to the load. This measured contact force is compared with the expected contact force that is calculated based on the applied external force on the object. A visual evaluation of the COM tracking is also used to evaluate the performance of the proposed dual-arm manipulation scheme. The authors of [14] evaluated their proposed impedance control scheme by calculating the internal stress on the object via the recorded end-effector force signals. Their proposed scheme aims to keep the internal stress of the object below a desired threshold. Similarly, [15] tried to control both the internal and the external force on the object by manually tuning the impedance gains of the controller. They evaluated the performance of their approach by comparing the desired internal force profile with the actual internal force obtained through extrapolating from the measured end effector force signals. Stability analysis was also done to evaluate the proposed approach. The authors showed that for the selected positive definite impedance gains, the system is

asymptotically stable. Likewise, [38] tried to regulate the internal force on the object using a distributed impedance control scheme. The COM tracking, as well as the internal loading, was used to evaluate the performance of their proposed scheme.

The authors of [39] addressed the question of robot posture by using a decoupled manipulation and posture control, while achieving optimal responsiveness at the effector. This is particularly useful, as the posture of the robot could impact how the internal force on the object is regulated. The authors evaluated their work by observing the behavior of the robots in the presence of obstacles. The robots are able to modify their posture to avoid the obstacles, while maintaining the desired object pose.

The authors of [29] and [3] designed their approaches from the stand-point of load distribution. As with the other works, the applied wrenches [29] and internal wrenches [3] are evaluated, but in these cases, with respect to load distribution factors. Both works evaluated what set of load distribution avoids internal stress loading of the object.

The evaluation strategies discussed above indicate that evaluating the COM tracking error and the internal forces on the object is crucial to determining a proper cooperative manipulation strategy. Therefore, using an AR tag tracking scheme, we monitored the COM pose of the object to evaluate the effect of our proposed compensation scheme. An effective compensation scheme will indicate a lower COM tracking error. Our evaluation strategy does not include a quantitative analysis of the internal forces on the system because our compensation approach does not focus on a quantitative measure of the internal stress on the object. This work only shows that internal stress can be used to compensate for disturbance on the system. Hence, the evaluation is solely on the COM pose tracking error.

In Equation 2.19, we presented the tuning factor  $\beta$  for the compensating robot. Hence, we will also show that as the pose tracking error increases,  $\beta$  decreases, and vice versa.

In this section, we discuss in detail the experimental validation we performed to demonstrate the effectiveness of the internal-stress-loading compensation strategy. We present the results comprising mainly object trajectory tracking error plots. These plots reveal how our

internal-stress-loading compensation strategy can mitigate the effect of disturbance on the system.

The following is a summary list of experiments and criteria used to evaluate the proposed approach.

1. Simulation experiments in which two arms are tasked with maintaining the pose of an object in the presence of disturbance.

(a) The disturbance is induced by:

i. commanding an external end-effector wrench in addition to the *grasp-matrix-computed*<sup>1</sup> wrench.

(b) Three compensation modes were applied:

i. Zero - No compensation is applied by the compensating arm.

ii. Partial - Compensating arm tries to partially compensate.

iii. Full - Compensating arm tries to fully compensate.

(c) The evaluation criteria for each of the compensation modes include:

i. COM frame pose error

2. Hardware experiments in which two arms, each mounted on a mobile base, are tasked with maintaining the pose of an object in the presence of disturbance.

(a) Two sets of experiments are conducted. These experiments are defined by the disturbance induced by:

i. commanding an external end-effector wrench in addition to the *grasp-matrix-computed*<sup>1</sup> wrench.

ii. having a human push against one of the links of the robot arm.

(b) Four compensation modes were applied for each set of experiment:

---

<sup>1</sup>*grasp-matrix-computed* refers to the wrench directly obtained by multiplying the inverse of the grasp matrix with the desired object frame wrench as shown in Equation 1.7.

- i. Zero - No compensation is applied by the compensating arm.
- ii. Partial - Compensating arm tries to partially compensate.
- iii. Full - Compensating arm tries to fully compensate.
- iv. *Baseline - No disturbance is applied. Hence, no compensation necessary.*

(c) The evaluation criteria for each of the compensation modes include:

- i. COM frame pose error
- ii. Measured external disturbances from the human contact is also compared against the COM frame pose tracking result.

### 3.1.1 Simulation Experiments

The experiment we used to evaluate the proposed compensation scheme was done in simulation using the physics-engine simulator *GAZEBO* [40]. In this experiment, we used two Kuka iiwa7 manipulators [41] to grasp and lift a 1 *kg* rod. The dimension of the rod was  $(0.8 \times 0.04 \times 0.04)m$  and both manipulators grasped the object along the longest axis as shown in figure Figure 3.1. *GAZEBO* provides us with the location of the COM of the object.

The cooperative task we defined for the simulation experiment involves moving the COM frame to a set-point in the world frame. The desired trajectory is communicated to a high-level controller, which then computes the required end effector wrenches for both manipulators using Equation 2.11. Both manipulators have low-level controllers that use the Equation 2.14 to compute the joint torques required to achieve the desired end effector wrench.

To evaluate our approach, we defined the desired COM frame pose (w.r.t the inertial frame)

$$x_o^d = \left[ 0.0, 0.4, 0.3, 0.0, 0.0, 0.0 \right]^T \quad (3.1)$$

The desired COM velocities and accelerations were set to zero. The collision wrench  $h_1^{ext}$

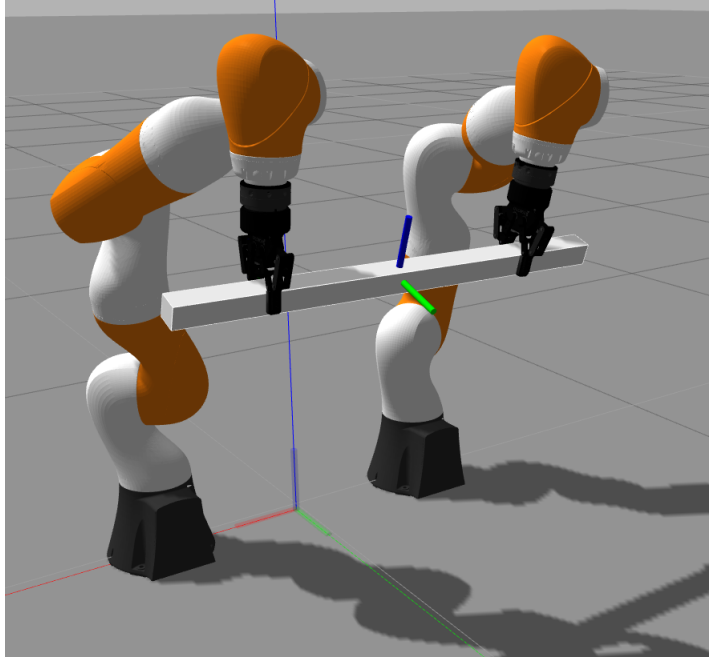


Figure 3.1: Two kuka iiwa7 arms cooperatively lifting a 0.8m rod in simulation [36]

on *arm-1* was injected as an applied wrench to the end effector in *GAZEBO*.

$$h_1^{ext} = \begin{bmatrix} 0.0, & 10.0, & 50.0, & 0.0, & 0.0, & 0.0 \end{bmatrix}^T \quad (3.2)$$

The plots for  $\beta$  shown in Figure 3.3 were generated with  $\lambda = 0.05$  (See Equation 2.19). The experiment was conducted for three different scenarios. The first scenario is shown in the red plot, in which case zero compensation was applied. In the second scenario, partial compensation, using the proposed impedance-based tuning factor, was applied. The last scenario, shown in blue, involved full compensation. These plots show the squared-error of the COM frame pose. External disturbance  $h_1^{ext}$  on the end effector of *arm-1* was introduced at time  $t = 10s$  and removed at time  $t = 30s$ .

In Figure 3.2, other than the  $x$  and  $z$  orientation errors, it can be observed that when partial compensation is applied, the COM pose error is less than that of the case where no compensation is applied, but often larger than that of the full compensation. A closer look at the plots for the  $x$  orientation show that although the error seems lowest for the zero

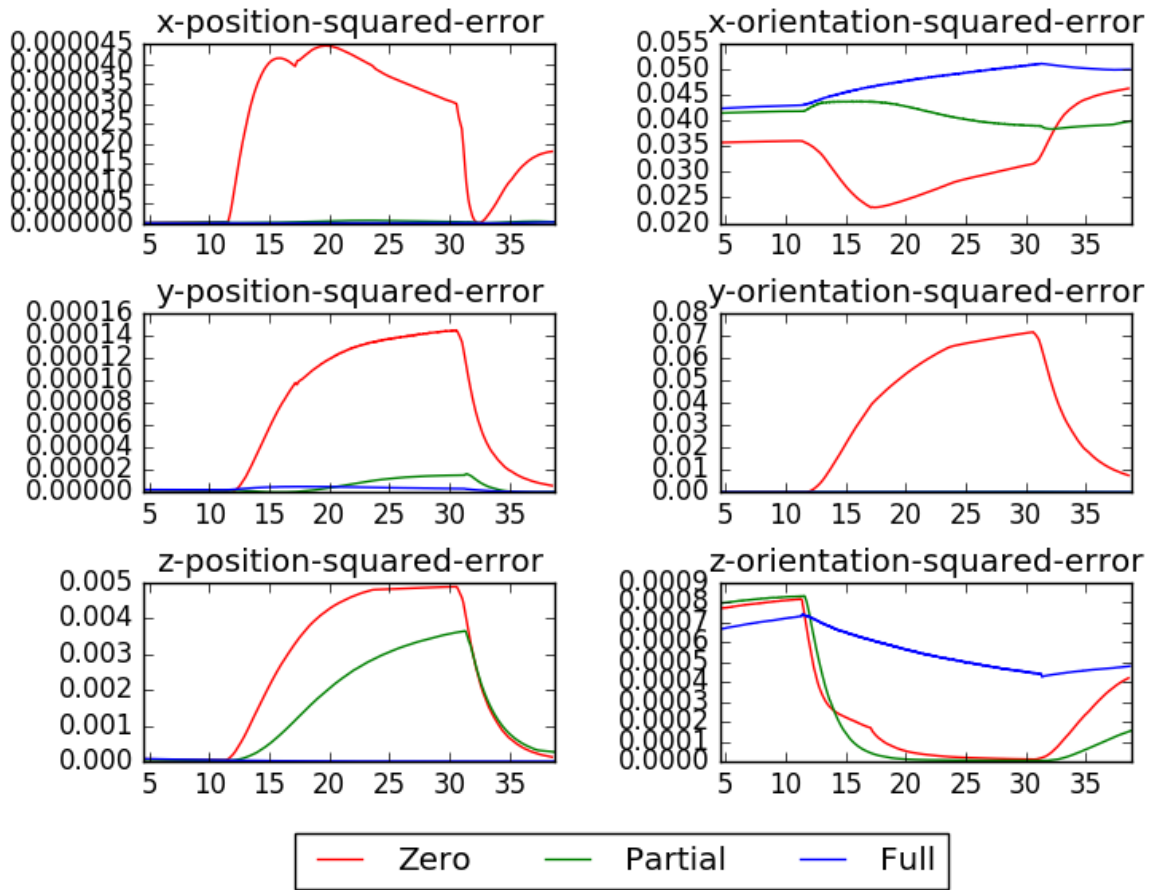


Figure 3.2: Plot of COM pose error for zero compensation, partial compensation and full compensation [36]

compensation framework, the error value changes more drastically during collision, when compared to the other two frameworks (partial and full). It might seem counter-intuitive, since the error drops to a significantly lower value when collision occurs with the zero compensation framework. However, the implication of this is that the COM pose actually changes more significantly during zero compensation, compared to the other two schemes. In summary, partial compensation ensures that the COM pose is not as impacted by the collision on *arm-1* compared to when no compensation is applied, and at the same time, ensures that extremely high wrenches are more regulated compared to the full compensation scheme.

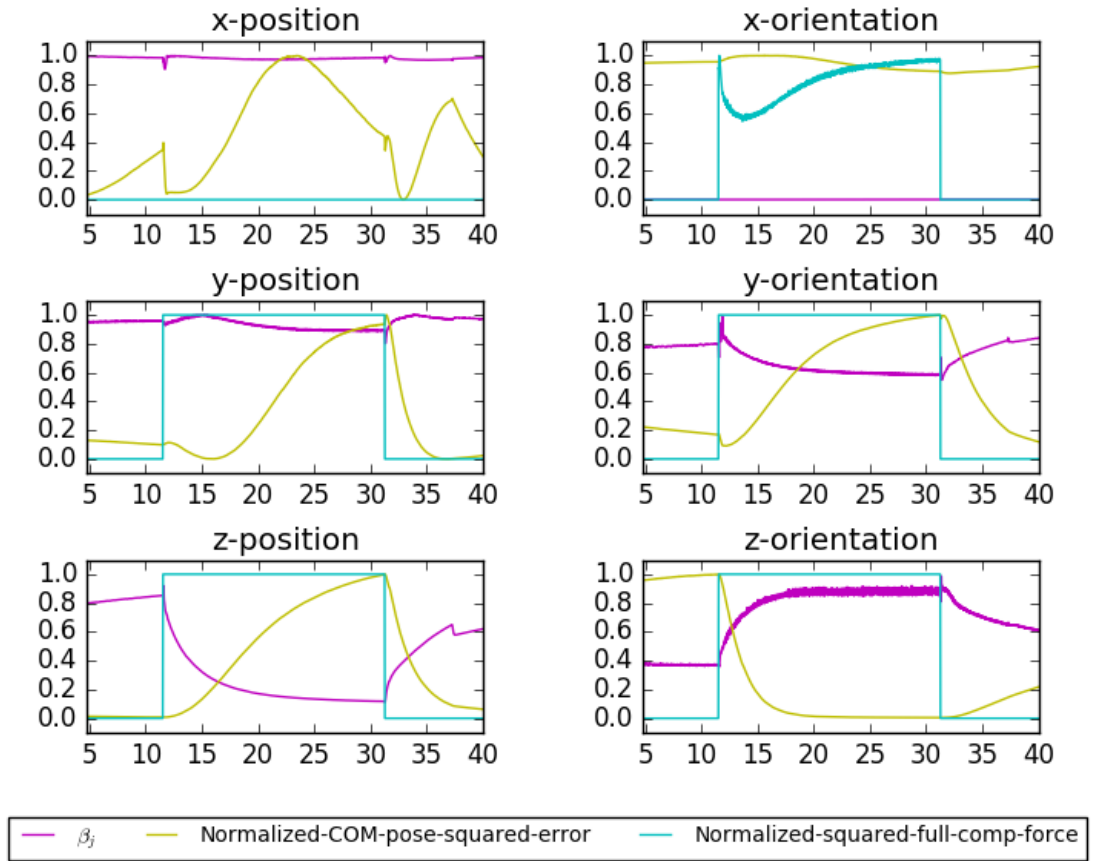


Figure 3.3: Plot of compensation-factor ( $\beta$ ), normalized-COM-squared error and the normalized-squared-full-compensation-force/torque for the compensating arm (*arm-2*). Decay constant  $\lambda = 0.05$  [36]

Finally, in figure Figure 3.3, we show how the tuning factors  $\beta(j)$  change as the COM pose error, as well as how the computed compensation wrench for *arm-2* evolves when collision is applied to the system. The plots for the compensation wrench and the COM pose error have been normalized to be between 0 and 1, since  $\beta(j) \in [0, 1]$ . Note that the compensation forces/torques shown in figure Figure 3.3 have not been scaled by the corresponding  $\beta(j)$ . Thus, the compensation forces/torques labeled as *normalized-squared-com-force* are the normalized forces/torques that would be applied when full compensation is being done. As expected,  $\beta(j)$  increases as the COM pose error decreases and vice versa. Going back to our assumption that when the COM pose error decreases, it is assumed that the required compensation force/torque decreases and as such  $\beta(j)$  should increase. Contrary to that assumption, along certain axes, this is not the case. For instance, the z-orientation in figure Figure 3.3 actually shows the COM pose error to be decreasing as the compensation torque is increasing. However, the COM pose errors for the z orientation shown in figure Figure 3.2 are low, so even though  $\beta(j)$  is close to 1 during the collision period, resulting partial compensation wrench is still extremely low.

### 3.1.2 Hardware Experiments

To further demonstrate the efficacy of our compensation scheme, we conducted hardware experiments. Our experiments involved two Kuka iiwa14 manipulators as shown in Figure 1.1. As stated earlier, we adopted an *eye-to-hand* camera tracking setup [42]. We placed AR tags on both robots and on the object to be co-manipulated. The object was a carton weighing about 0.5 kg and had the dimensions:  $52cm \times 2.5cm \times 4.2cm$ . These dimensions are quite similar to the object we used in the simulation experiments. We chose this material because the crux of our work involves inducing internal stress on the manipulated object. Thus, it behooved us to use an object that could withstand some internal stress without significant deformation. Each arm was retrofitted with a two-fingered *Robotiq* gripper, with enough contact friction to keep the object in grasp during the motion of the arms.

## Hardware Setup

The schematic shown in Figure 3.4, describes the experiment setup and how the flow of information was designed. As shown in the control hierarchy in Figure 2.1, the *high-level controller* computes the end-effector wrench each robot should apply and then broadcasts this information to the *low-level controller* for each robot. The low-level controller for each robot then maps the desired end effector wrench to desired joint torques. The dashed arrows indicate wireless connection, while the solid arrows indicate wired connection. The black solid arrows indicate ethernet cables, while the green solid arrow which runs from the camera to the main computer represents a USB cable connection. The red-colored text under the arrows indicates ROS topics published and subscribed by *Publisher* and *Subscriber*, while the green text indicates data sent over a wired connection.

We used two Dell Alienware laptops, which are also shown in the schematic. The setup required two computers because each manipulator requires an ethernet connection for communication, but each personal computer only has just one ethernet port. We considered using a *splitter* that allows for *one-to-many* ethernet connection, which would have circumvented the need for two computers. However, the high communication speed requirement for the manipulators could not be met with the *splitter* approach. The grippers, although not shown in the schematic, were also commanded via ROS.

It is worth noting that the *low-level controller*, as well as the *Ridgeback* mobile base (see Figure 3.4), send information back to the *high-level controller*. For instance, each *low-level controller* sends the current end-effector pose – with respect to the base frame – of the manipulator to the *high-level controller*, which uses this information to create the grasp matrix in Equation 1.6. These poses first need to be transformed to the object (AR tag) frame. Also, the mobile platforms publish odometry ROS topics that provide the *high-level controller* with velocity data for the base.

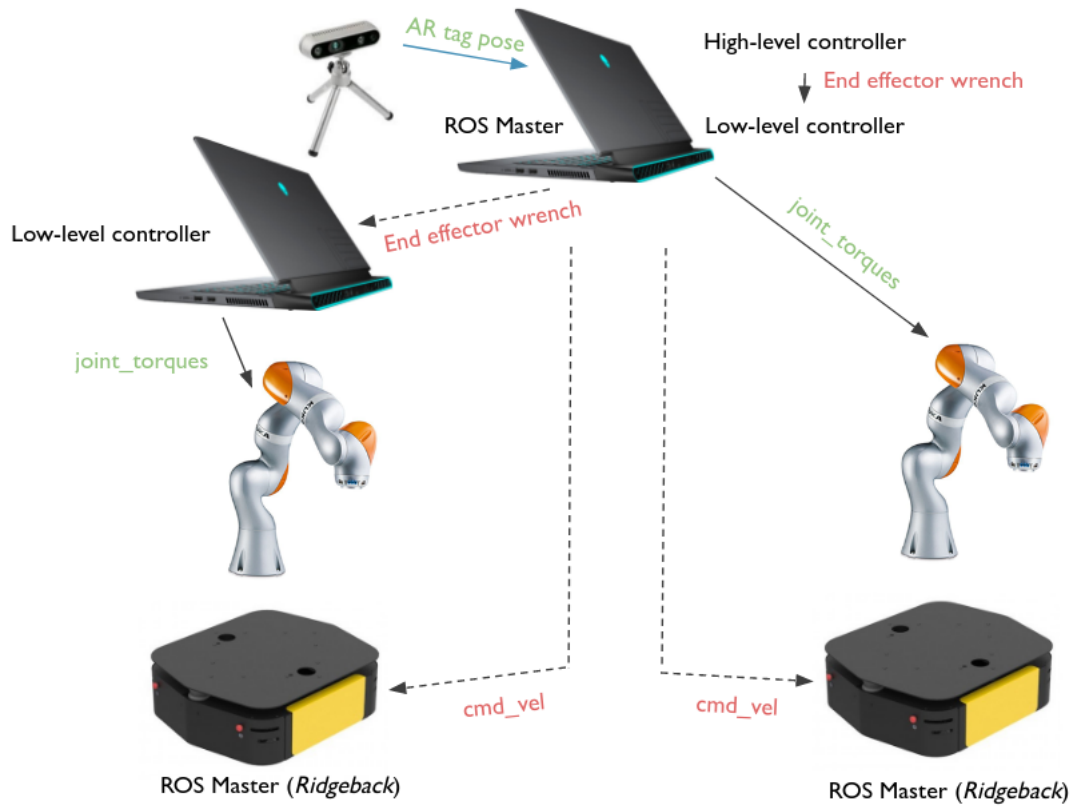


Figure 3.4: Schematic of hardware experiment setup

### *Vision Tracking*

In simulation, we had access to the poses of both end effectors and the COM of the object, which are critical information for the cooperative manipulation process. However, to obtain similar information for hardware experiments, we adopted a vision based system, using an Intel RealSense D455 camera [43] and fiducial markers (AR tags) [44] placed at appropriate locations on both the robots and the object (see Figure 1.1). The AR tags placed on the object were used to provide pose information for the robot end effectors. While the AR tag placed on the object provided information about the pose of the object's COM. Given the size of the AR tags and the lighting conditions in the lab, the range of the camera-tag detection was about four meters. Hence, the workspace was limited to this range.

The camera frame served as the global (or inertial) frame for the experiment. Thus, the desired trajectory of the object was specified in the camera frame and the poses of the object and both robots were tracked also in the camera frame. However, when commanding the desired end-effector wrench for both arms, we ensured that the computed end-effector wrench was transformed to the manipulator base frame for the corresponding arm. Similarly, the computed velocity commands for the mobile platforms were transformed to the corresponding mobile base frame.

### *Vehicle-Arm Coordination*

The two robot bases were coordinated in a *synchronous* approach, such that one robot base (*base-A*) was assigned a desired trajectory and the other robot base (*base-B*), was assigned the same trajectory, but with an offset along the y-axis; this offset was added not just for collision avoidance between the robots, but also to respect the kinematic constraint imposed by the object being grasped. At the start of the experiment, before any motion, the relative planar pose between the object and the base is established. Once the experiment has commenced, the desired pose of the object in the horizontal plane (x-y plane of the camera frame, see Figure 3.5) is offset to the desired pose of the *base-A*. In other words, if

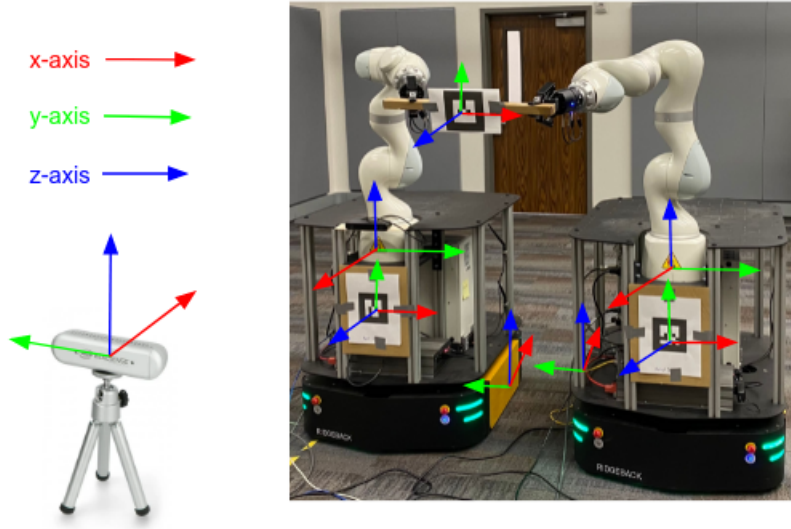


Figure 3.5: Schematic of eight 3D coordinate frames in our system setup.

the relative planar position between the *base-A* and the object at the start of the experiment was  $x_{OA} \in \mathbb{R}^2$ , then the desired x-y position for *base-A* in the plane is simply:

$$x_A^d = x_o^d + x_{OA} \quad (3.3)$$

where  $x_d^A \in \mathbb{R}^2$  is the desired planar position (x, y) assigned to the *base-A* controller. Similarly, the desired position for the *base-B* is

$$x_B^d = x_A^d + x_{AB} \quad (3.4)$$

where  $x_A$  is the current position of *base-A*, and  $x_{AB} \in \mathbb{R}^2$  is the relative position between the *base-A* and *base-B*.

It is important to note at this point that the *Ridgeback* mobile platforms we used in our experiments were omnidirectional, and therefore there were no nonholonomic constraints on their motion [45]. Thus, we are able to control the x, y and yaw degrees of freedom. However, since the field-of view of the camera is not particularly wide, yaw movements

for the object have been assigned to the arms. The base provides the increased workspace by moving linearly along the plane, while the arms co-manipulate the object in the other Degrees of Freedom (DOF). Thus, the linear component of the computed wrench that is applied at the object's COM (see Equation 2.9) is zeroed out for the x and y dimensions. In other words, the arms do not try to track the desired object's linear trajectory in the x-y plane; that role has been assigned to the mobile platforms.

The *Synchronous* approach we have adopted shares some similarity with a *leader-follower* approach, which has been applied by [46, 47]. However, our approach has not been applied in a scheme where the arm leverages internal stress on the object to compensate for disturbance on the cooperative system, as we have done here.

The purpose of the mobile platforms, on which the manipulators are mounted, is to increase the workspace of the robot arms. Therefore, we assigned a desired sinusoidal trajectory for the object in the x-y plane. For these axes, the arms do not exert forces on the object. Instead, the mobile platforms are charged with the responsibility of tracking this sinusoidal trajectory, using the prescribed synchronous approach. Visually speaking, the sinusoidal trajectory causes the entire cooperative setup to move in a circle. The equations below defines the trajectory:

$$\mathcal{X}_o^d(t) = \mathcal{O}_x + A\cos(2\pi ft - \pi) \quad (3.5a)$$

$$\mathcal{Y}_o^d(t) = \mathcal{O}_y + A\sin(2\pi ft - \pi) \quad (3.5b)$$

where  $\mathcal{X}_o^d, \mathcal{Y}_o^d$  is the desired position of the object in the plane, as a function of time;  $\mathcal{O}_x, \mathcal{O}_y$  is the origin of the circumscribed circle;  $A$  is the amplitude of the circle and  $f$  is the frequency with which the object is expected to move about the circle. In our experiments, we defined

$$A = 0.3m, \quad f = 1/20 \text{ Hz} \quad (3.6)$$

The frequency value implies that it takes 20 seconds for the object to make its complete revolution about the circle. These values were chosen arbitrarily. However, we considered the limited lab space and camera range while deciding the value of the amplitude. We also considered how both the amplitude and frequency values impacted the velocity of the robots; we wanted to achieve less aggressive motions. Hence, given the specific amplitude value, we chose a frequency that would result in an acceptable velocity of the system.

It is worth noting that from Equation 3.3,  $x_o^d$  is equivalent to the stacked vector of  $[\mathcal{X}_o^d, \mathcal{Y}_o^d]$ . Furthermore, the desired velocities  $\dot{x}_o^d = [\dot{\mathcal{X}}_o^d, \dot{\mathcal{Y}}_o^d]$  is defined by the derivative of Equation 3.5:

$$\dot{\mathcal{X}}_o^d(t) = -2\pi f A \sin(2\pi f t - \pi) \quad (3.7a)$$

$$\dot{\mathcal{Y}}_o^d(t) = 2\pi f A \cos(2\pi f t - \pi) \quad (3.7b)$$

Although, it is possible to obtain the velocity of each mobile base by taking the timed difference of pose of the attached AR tag, we chose to use the odometry information published by the odometer sensors onboard the mobile robots. We considered this a more accurate estimate of the velocities because they are less prone to the errors of pose tracking by cameras.

As we have discussed in this section so far, the robot bases are controlled by twist (velocity) commands and are linearly controlled in the plane. Therefore, velocity commands comprise just velocities in the x and y direction. These commanded velocities are derived as follows.

$$\dot{x}_{mb}^c = K_{mb}^P(x_{mb}^d - x_{mb}) + K_{mb}^D(\dot{x}_{mb}^d - \dot{x}_{mb}) \quad (3.8)$$

where  $x_{mb}^d, \dot{x}_{mb}^d \in \mathfrak{R}^2$  are the desired planar position and velocity of the mobile robot (mb), respectively;  $K_{mb}^P, K_{mb}^D \in \mathfrak{R}^{2 \times 2}$  are the positive-definite diagonal stiffness and damping matrices, respectively.

For *base-A*,  $x_{mb}^d$  is equivalent to  $x_A^d$  derived in Equation 3.3. Likewise, for *base-B*,  $x_{mb}^d$  is equivalent to  $x_B^d$  derived in Equation 3.4.

We conducted two sets of hardware experiments:

1. Experiments with the disturbances introduced at the end effector.
2. Experiments with the disturbances introduced by a human, along the links of *arm-2*.

*Experiments with the disturbances introduced at the end effector*

In the first set of experiments, we introduced an end-effector wrench disturbance ( $h_2^{ext}$ ) on *arm-2*, which is mounted on *base-B*.

$$h_2^{ext} = [f_x, f_y, f_z, m_x, m_y, m_z] = [0.0, 0.5, 1.5, 0.0, 0.0, 0.0] \quad (3.9)$$

Forces and moments are in  $N$  and  $N/m$  units. The disturbance wrench was introduced by commanding an additional  $h_2^{ext}$  to the standard wrench  $h_2$ . The disturbance was introduced at the 5-second mark and removed at the 15-second mark. Figure 3.6, shows the plots of the pose error of the AR tag across the six different axis. The results in Figure 3.6 show the comparison between the three compensation modes and the impact these modes have in minimizing the object pose error, in the presence of disturbance. Since the most significant disturbance on the manipulator is along the z-axis, logically, we expect the disturbance to induce a moment about the x-axis. Therefore, we focus our attention on the orientation squared error along the x-axis, where it is clear that object pose error is negatively impacted, when the disturbance is introduced at the 5-second mark. The partial and full compensation mode appears to minimize the impact of this disturbance. The z-position axis also show a poorer object tracking performance in the zero compensation mode, although the values appear quite negligible. The oscillatory errors along the x-position and y-position axes are likely due to the motion of the base. As stated earlier, the desired motion of the bases is sinusoidal. It is likely that a lag between when the time counter starts and when

the mobile bases start moving, could induce this observed oscillatory error. While the orientation about the z-orientation axis of the object is not directly controlled by the mobile platforms, but by the cooperating arms, a momentary out-of-sync motion of the two mobile platforms could negatively impact the tracking along the z-orientation axis. This might explain the unexpected z-orientation error plots, in which the zero-compensation mode seems to perform better than other modes.

As discussed previously, the zero compensation mode simply means that no compensation was applied by the compensating arm. The full compensation mode tries to transfer all the disturbance wrench from the *disturbance-inducing* arm into internal stress on the object. Finally, the partial compensation mode uses the impedance-based tuning factor that has been defined in chapter 2, to scale the amount of compensation applied by the compensating arm. Unlike in the simulation experiments, where we set  $\lambda$  to be a constant value for all axes, we defined the values of  $\lambda$  to be 2.0 and 1.0 for the position and orientation axes, respectively. As Equation 2.19 and Figure 3.7 show, the value of  $\beta(j)$  decreases as the  $\lambda$  increases.

A baseline experiment is also conducted to reflect the effect of the disturbance on the system. In the baseline experiment, no disturbance was introduced; hence, no compensation scheme was applied here. The plots show that on average when there is no disturbance, the object pose error is less than the case where disturbance is applied, but no compensation is performed, i.e. *zero-mode*.

The gain matrices for computing the control signal in Equation 2.10 had the following diagonal entries:

$$K_p = [0.0, 0.0, 60.0, 4.0, 16.0, 4.0] \quad (3.10a)$$

$$K_v = [0.0, 0.0, 45.0, 3.0, 12.0, 3.0] \quad (3.10b)$$

The gains for the  $x$  and  $y$  position axes were set to zero because control for both degrees-

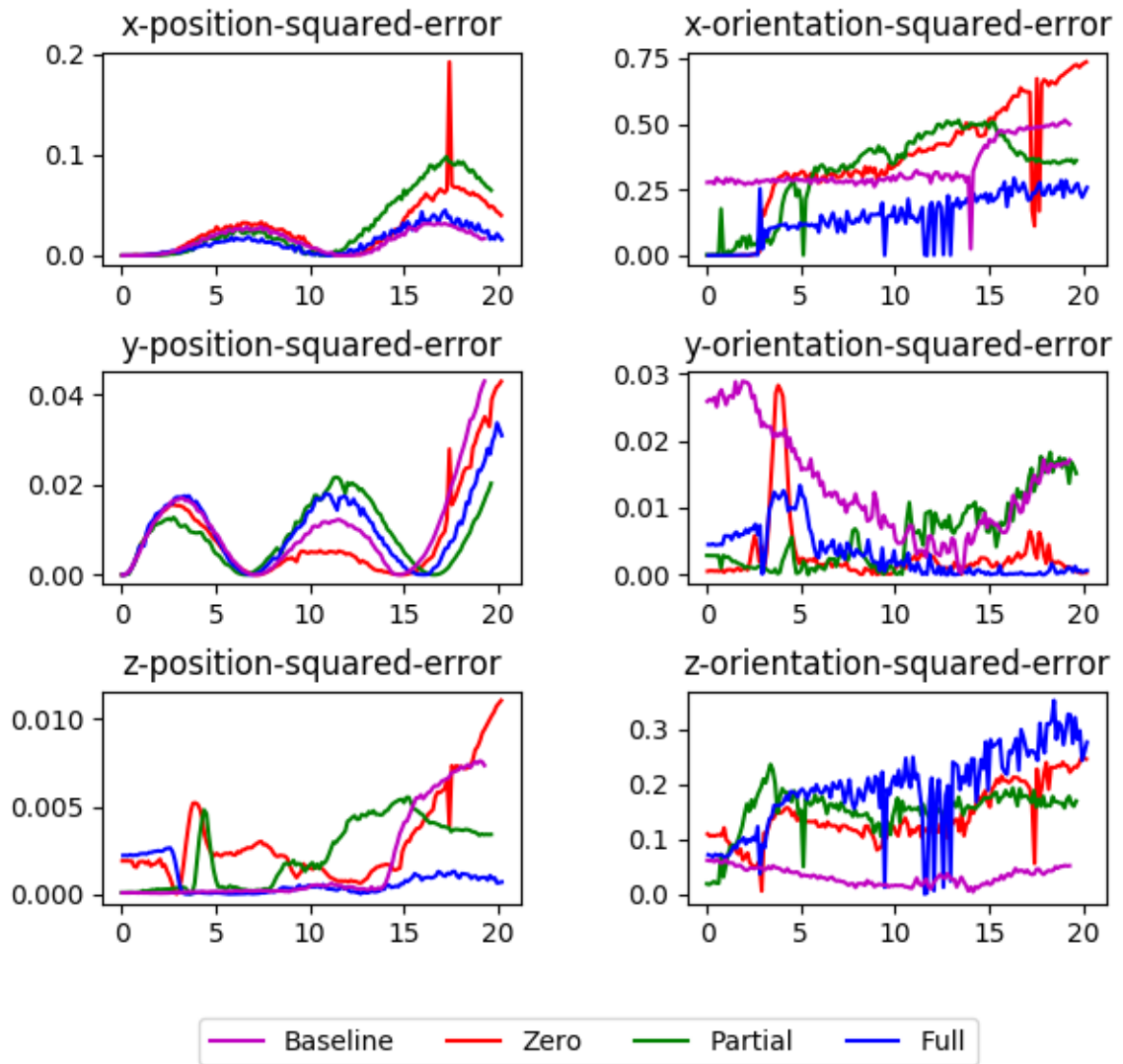


Figure 3.6: Plot of COM pose error for zero compensation, partial compensation and full compensation; hardware experiments

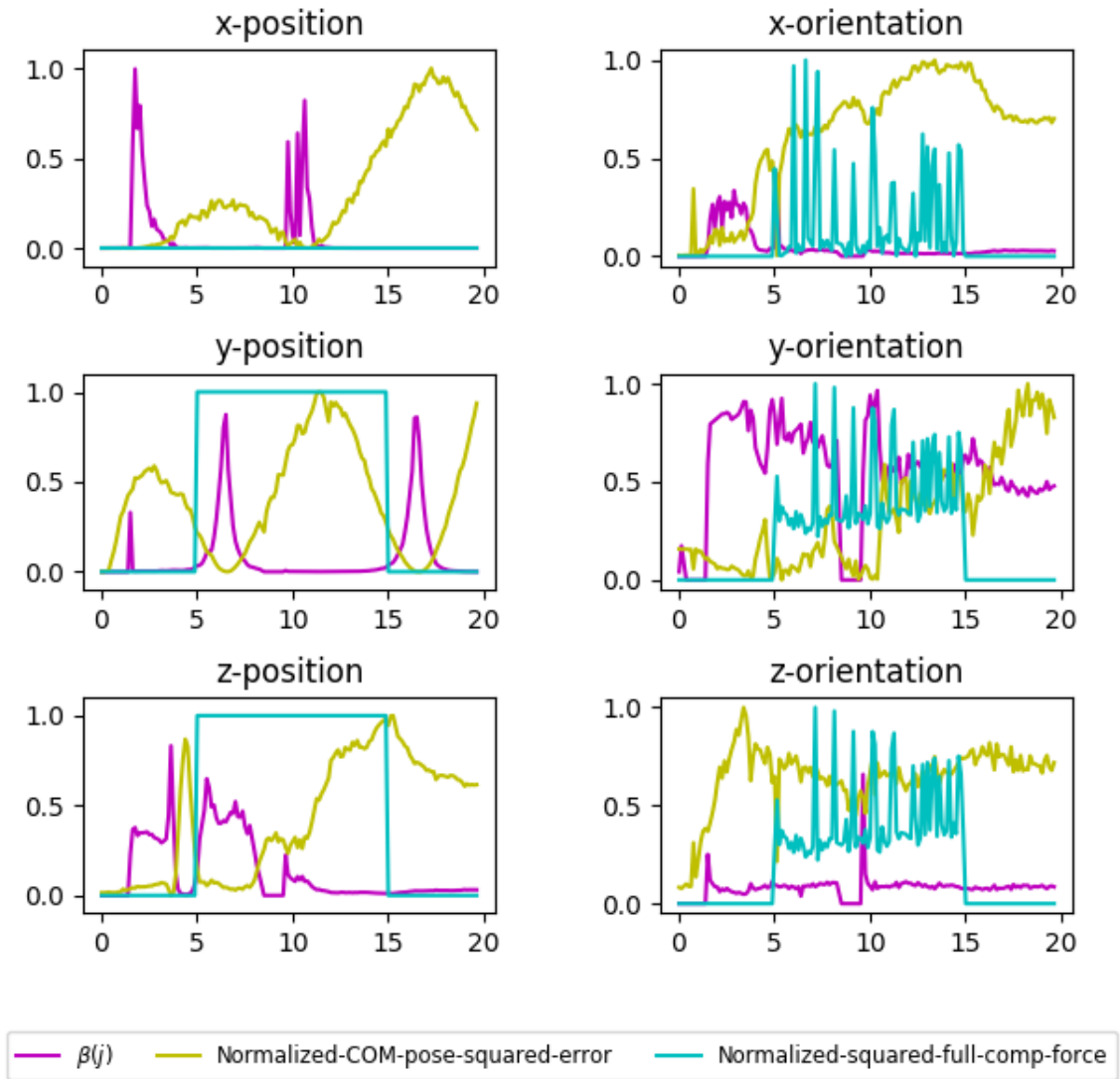


Figure 3.7: Plot of compensation-factor ( $\beta(j)$ ), normalized-COM-squared error and the normalized-squared-full-compensation-force/torque for the compensating arm (*arm-2*).

of-freedom have been completely assigned to the mobile platforms. However, the compensation is performed across all six degrees-of-freedom. Hence, the  $\gamma$  gains ( $K_p^\gamma, K_v^\gamma$ ) for the partial compensation mode, as defined in Equation 2.15, were specified as

$$K_p^\gamma = [30.0, 30.0, 30.0, 6.0, 6.0, 6.0] \quad (3.11a)$$

$$K_v^\gamma = [8.0, 8.0, 8.0, 2.0, 2.0, 2.0] \quad (3.11b)$$

*Experiments with the disturbances introduced by a human, along the links of arm-2*

In the second set of experiments, instead of directly introducing the disturbance as an end effector wrench on *arm-2*, we applied the disturbance along the links of the robot by having a human push against one or more of the links of the robot. *Kuka* robots, like the one we used in our experiments, have the functionality that provides an estimate of the external joint torques. These external joint torques can be mapped to from the joint space to the end effector space, via the Jacobian. Mathematically, this is achieved as [32]:

$$h_2^{ext} = J_2^{-\top} \tau_2^{ext} \quad (3.12)$$

where  $J_2^{-\top}$  is the Moore-Penrose inverse of the Jacobian transpose of *arm-2*, and  $\tau_2^{ext}$  is the external joint torques induced by the human during contact with the robot. Figure 3.8 shows how the human applies contact to the arm.

Apart from the fact that the disturbance is applied by a human contact with the arm, one distinction between this set of experiments and the first set of experiments discussed in subsection 3.1.2 is that in the first set of experiments, the pose of the object is used to compute end effector wrench for each arm, whereas in the second set of experiments, each arm simply uses a position controller to maintain the end effector pose. Hence, this implies that the compensation (i.e. the attempt at maintaining the desired pose of the object) is not

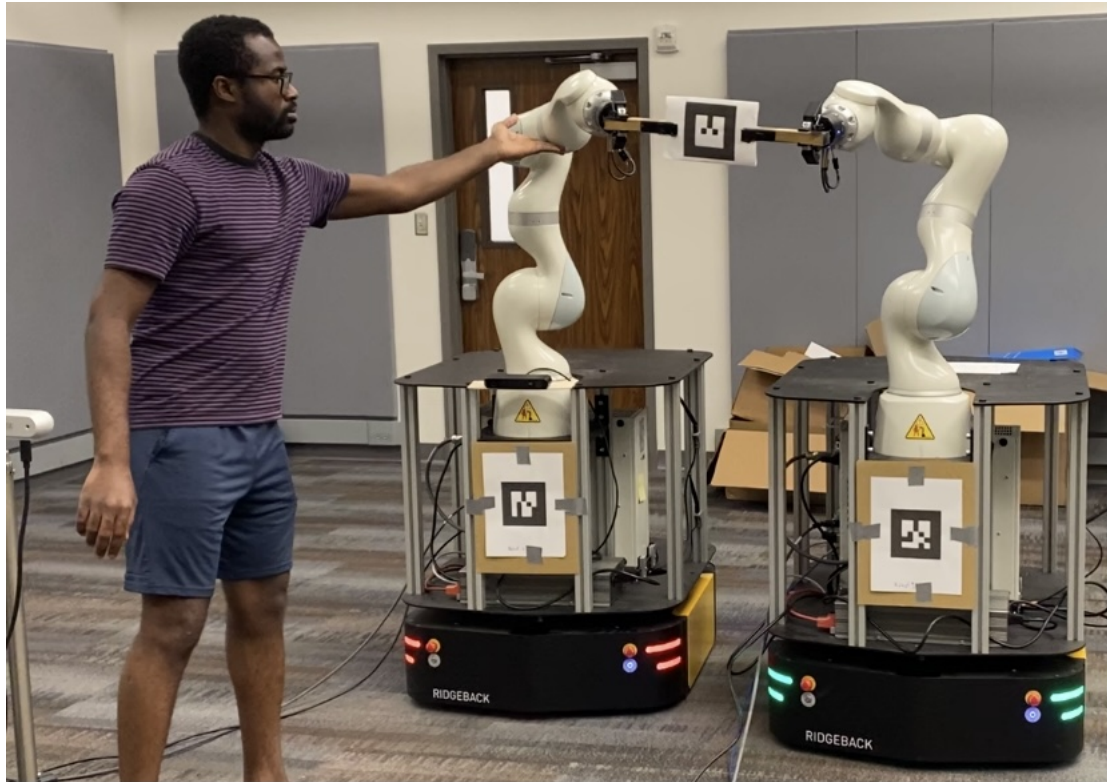


Figure 3.8: A human, introducing disturbance along the links of *arm-2*

only performed by the internal stress loading scheme, which we have presented so far, but the compensation is also attained by the feedback control by which the position controller tries to keep the end effectors at a fixed pose. This does not undermine the importance of our internal stress compensation scheme, as the results in Figure 3.9 show that having the internal stress compensation scheme in conjunction with the position controller minimizes the impact on the desired trajectory (or pose) of the object. For instance, despite the higher (on average) external end-effector wrench for the *partial* and *full* compensation modes, when compared to the *zero* compensation mode (see Figure 3.10), the object pose error appears to be the highest in the *zero* compensation mode. This implies that our internal stress compensation strategy is able to reduce the impact of the human disturbance on the desired pose of the object. Just like we did in the previous experiment section (see subsection 3.1.2), we conducted a baseline experiment without any disturbance to reveal the usefulness of the proposed compensation modes in the presence of disturbance. The  $\gamma$  gains ( $K_p^\gamma, K_v^\gamma$ ) for the partial compensation mode, as defined in Equation 2.15, were specified as

$$K_p^\gamma = [25.0, 25.0, 60.0, 1.0, 1.0, 1.0] \quad (3.13a)$$

$$K_v^\gamma = [14.0, 14.0, 33.7, 0.6, 0.6, 0.6] \quad (3.13b)$$

We observed that the external joint torques measured by onboard sensors of the arm were non-zero, even when no contact or disturbance was applied on the arm. We also observed that these external joint torque values changed when the configuration of the arm was changed. Finally, we observed that when all the joints were set to zero and the arm was in a configuration where all the gravity terms were cancelled (i.e. the arm was in a straight up configuration), the measured external joint torques were zero. This led us to the conclusion that these estimated values also combined certain gravity terms. However, there was no one-to-one mapping between gravity terms and the measured external joint

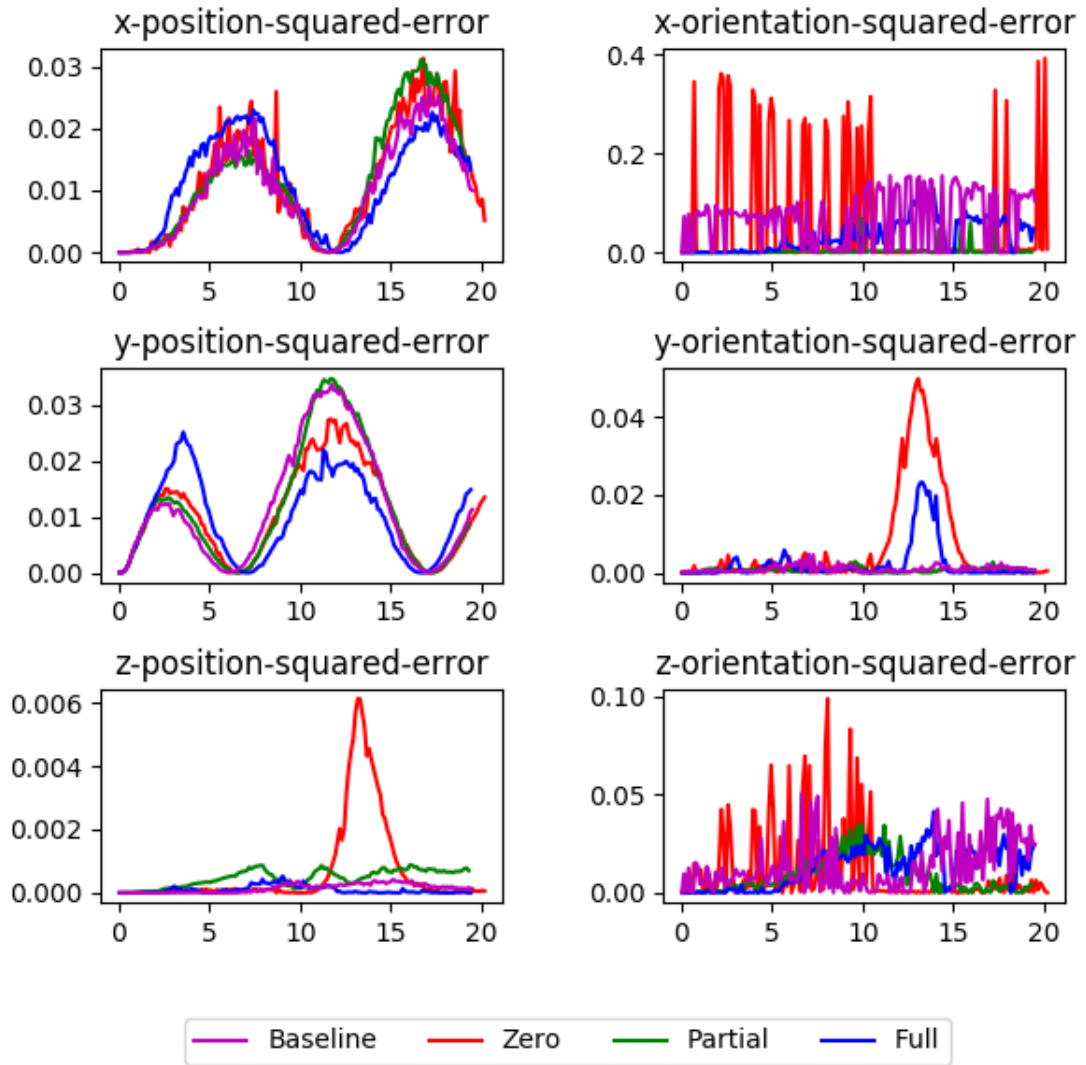


Figure 3.9: Plot of COM pose error for zero compensation, partial compensation and full compensation; hardware experiments. In this case, the disturbance is introduced along the links of the arm.

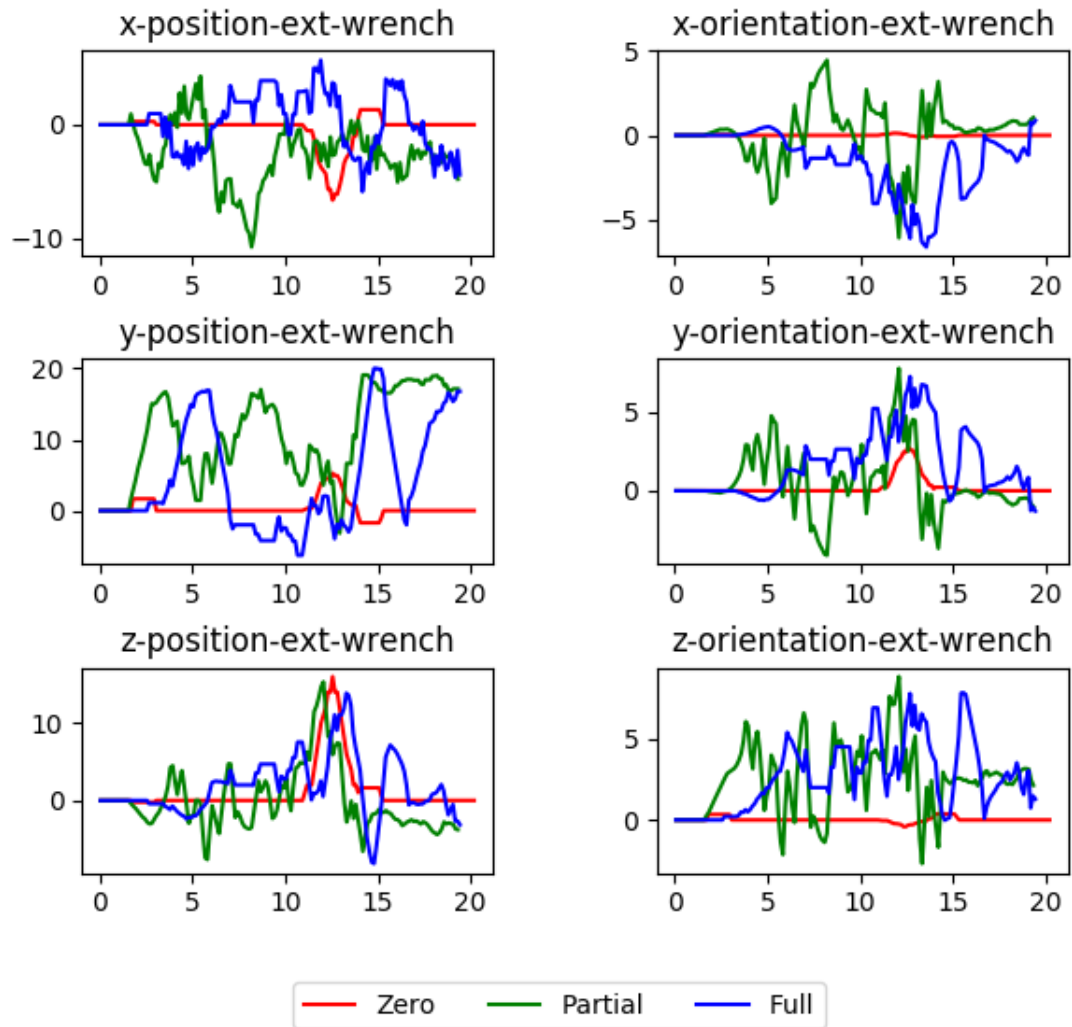


Figure 3.10: Plot showing the end effector wrenches (in *Newtons* and *N/m*) that are produced by applying disturbances along the links of *arm-2*.

torques. Hence, we had to devise a means to extract the external joint torques that were caused by contact on the arm. To achieve this, we make the assumption that the arm's configuration should not be changing much during the experiment. Therefore, at the start of the experiment, we obtain the measurement of the external joint torques for the current arm configuration and use this measurement as a baseline throughout that particular experiment. In other words, once the experiment commences, a baseline is established; any external joint torques beyond this baseline is considered to be the external joint torques induced by contact on the arm. We also established some threshold value ( $7.0N/m$ ) by which the new measured external joint torques must exceed the baseline, for it to be considered as contact-induced. This threshold is necessary because as the arm moves, there are some slight changes to the measured external joint torques that are not caused by contact on the arm. This threshold value was empirically chosen.

Additionally, the measured external joint torques can be quite noisy; since the compensating arm (*arm-1*) applies a compensation wrench that is based on the external Cartesian wrench on *arm-2*, which is derived from the noisy measurements of the external joint torques, we applied a low-pass filter to remove the noise on the external Cartesian wrench. This reduces the aggressive shaking of the compensating arm, which we observed prior to applying the low-pass filter.

In conclusion, our results indicate that internal stress loading can be used as a compensation strategy for disturbance on one of the cooperating partners. The results of our hardware experiments reflect certain details, not originally captured during our simulation experiments. One major limitation of our internal stress loading approach is the reliance on accurate pose estimates. In simulation, we had accurate values of both the poses of the end effectors, as well as the pose of the object's COM frame. On the hardware, we placed AR tags on the object and the robot, and used a vision tracking scheme to obtain these poses. We found these poses to be not as accurate as the ones in simulation. Hence, when the compensation forces are computed using the relative pose between the grasp point and

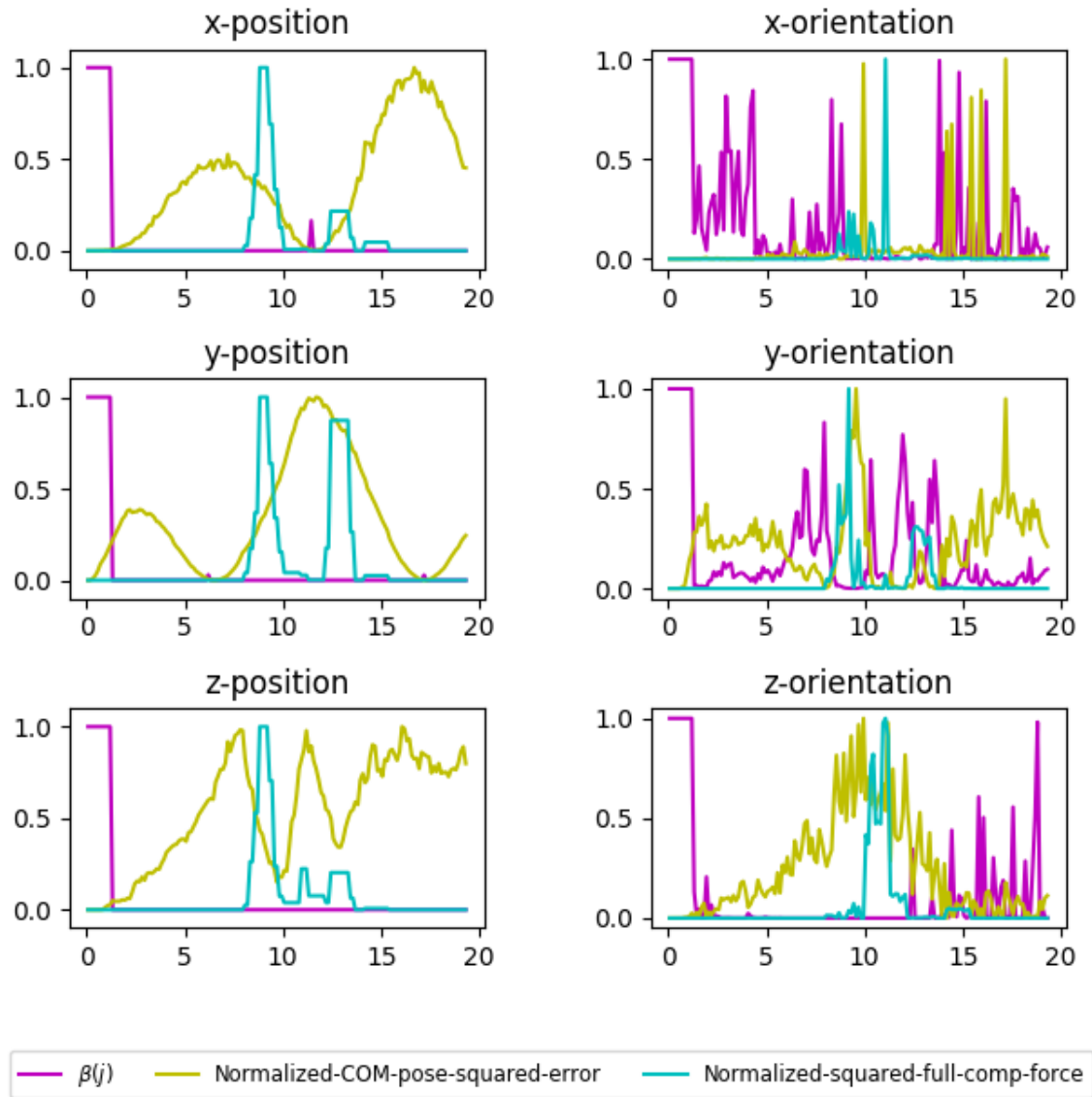


Figure 3.11: Plot of compensation-factor ( $\beta(j)$ ), normalized-COM-squared error and the normalized-squared-full-compensation-force/torque for the compensating arm (*arm-2*). In this case, the disturbance is introduced along the links of the arm.

the COM frame, as shown in Equation 2.4 and Equation 2.5, a breakdown of the compensation framework can occur when these pose estimates are inaccurate. This could lead to compensation wrenches  $\mathcal{N}$  that no longer satisfy Equation 2.8. When that happens, the compensation wrenches no longer compensate, but instead amplify the disturbance that is already being introduced by the colliding arm. Therefore, we recommend further studies to make this approach more robust to inaccurate pose estimation.

## CHAPTER 4

### INTRODUCTION AND BACKGROUND FOR RESIDUAL REINFORCEMENT LEARNING

In this chapter, we will present the framework of residual reinforcement learning. We will start by reviewing some of the works on deep reinforcement learning (DRL) and how these works have been applied to robotics. Then, we will highlight some of the previous works of others, in which residual RL was applied to single-arm manipulation tasks.

#### 4.1 Deep Reinforcement Learning

The desired tasks for robotic systems have increasingly become complex. Part of the complexity comes from the fact that many of the tasks we would like to assign to robots are tasks typically performed by humans with complex reasoning involved. This attribute makes classical modelling very difficult and sometimes impossible. Data driven approaches, which have played a major role in the machine learning community, has offered a solution to problems that cannot be modelled. However, in robotics, data can be very expensive or impossible to obtain. Hence, reinforcement learning (RL) [48] has become very attractive within the robotics community [49, 50].

An RL problem can be formulated as a Markov Decision Process (MDP)  $M$ , defined by the tuple  $M = (\mathcal{S}, \mathcal{A}, R, T, \gamma, H)$  [48]. The continuous formulation can be simplified through time discretization, where at every time step  $t$ , an agent in a state  $s_t \in \mathcal{S}$  can take an action  $a_t \in \mathcal{A}$ . The probability of the next state  $s_{t+1}$  is specified by the transition function  $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S}_{t+1} \rightarrow [0, 1]$ . For each transition, the agent receives a reward  $r_t \in \mathfrak{R}$ . We assume the MDP to have a finite horizon  $H$  and a discount rate  $\gamma$ , but this could also be extended to an infinite horizon MDP. The agent is parametrized by a model with the parameter defined by  $\theta$ . The goal is to optimize the parameter  $\theta$  of the agent's

policy,  $\pi_\theta : S \rightarrow A$ , so as to maximize the total expected reward  $J = \mathbb{E}[\sum_{\tau=t}^H \gamma^{\tau-t} r_\tau]$  [51]. When the transition function  $T$  between states is known, the RL scheme is called *Model-based RL* [52, 53]. Otherwise, the RL scheme is called *Model-free RL* [54].

The question of safety has always been an issue when applying reinforcement learning to robotics. Typically in classical controls, Lyapunov stability analysis is performed to guarantee the stability of the system. When we cannot model the evolution of the system, and RL is adopted, safety guarantees become elusive. Moreover, safety can have broader implications beyond system stability; sometimes safety might require safe human interaction between the robot and a human, which cannot be easily modelled. Works by [55, 56] present and discuss state-of-the-art methods in safe RL. The authors of [57, 58] developed Lyapunov-based approaches to safe RL.

With the increased computational capacity, deep neural networks, with multiple hidden layers, have become viable models for learning complex systems [59, 60]. When the reinforcement learning community adopted deep neural networks, the term *Deep Reinforcement Learning (DRL)* was coined. Deep neural networks are useful when the state space or action space is too large. In robotics, often both the state and action spaces are continuous. Hence, deep neural networks have been quite attractive in this sphere.

## 4.2 Residual Reinforcement Learning

Collaborative manipulation is often approached with certain assumptions in mind, including, but not limited to, knowledge of payload mass, accurate measure of the grasp map between the object frame and robot grasp locations. Decentralized approaches such as [61, 62, 63] tried to relax the assumption of known object dynamic parameters by means of adaptive control. One common theme with these adaptive control approaches is the assumption that the unknown parameters are constant or slowly varying. Hence, when the unknown component of the cooperative process is an external disturbance introduced by one of the cooperating agents, then such adaptive control approaches become less effective

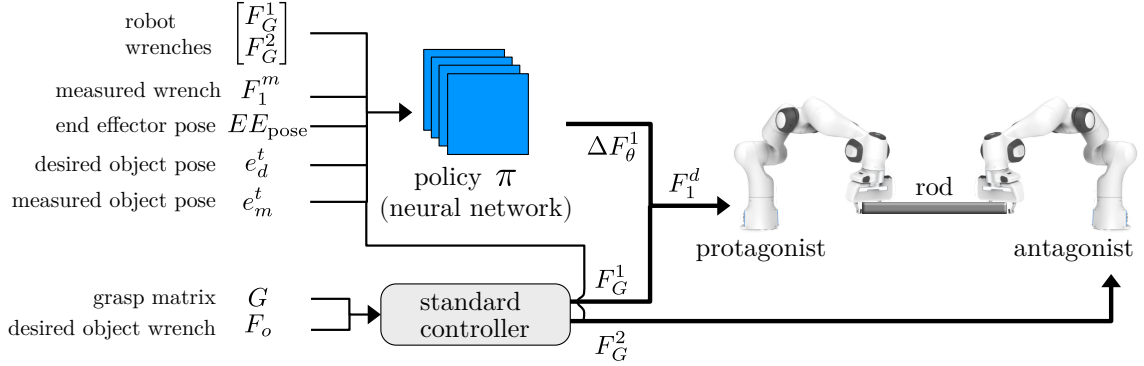


Figure 4.1: RL model applied to the protagonist, in a dual-arm cooperative setup. See subsection 5.1.1 for the definition of the *protagonist* and *antagonist*.

[51].

Therefore, in contrast to adaptive control approaches, we propose a less restrictive approach based on reinforcement learning. Our proposed approach, while having its own limitations, is able to address the scenario where one or more agents do not apply the appropriate wrenches assigned via the grasp map. This relaxes the assumption that the unknown parameter is constant, as our framework does not require strict assumptions on the evolution of the disturbance or deviation by a cooperating partner.

Reinforcement learning approaches have been developed for robotic systems to account for unknown system dynamics and model errors. However, they require significant tuning and training effort [49]. Residual reinforcement learning addresses this problem by decomposing the control effort into two parts: the nominal control part and the reinforcement learning (or residual) part [64]. The central idea of residual reinforcement learning (RL) is to learn a residual that is combined with a nominal controller to improve the overall performance of the nominal controller. How the learned policy is combined with the nominal controller depends on what form of residual RL is being applied. The authors of [65] proposed the residual policy learning (RPL) method, where the learned policy is directly superposed on the output of the nominal controller. However, [66] noted that competition between the nominal controller and the learned policy was a possibility. Hence, [66] proposed a resid-

ual feedback learning (RFL) method that, in contrast to the RPL method, superposes the learned policy on the feedback to the controller. Residual RL builds a symbiotic relationship between the nominal controller and the learned policy; the nominal controller serves as the main controller for the system, while the learned policy complements it. On the other hand, part of the input to the policy ( $F_G^1, F_G^2$  – see subsection 5.1.2 for more details) comes from the output of the standard controller, as shown in figure Figure 4.1. Hence, there is this constant feedback between the nominal controller and the RL policy.

The authors of [65, 66] focused on single-arm manipulation tasks. In [67], the authors extended the residual RL framework to a visual-based manipulation problem, however, still within the single-arm manipulation scope.

In our research, we have developed a novel application of residual reinforcement learning to cooperative manipulation [51]. We employed residual RL as a decentralized compensation strategy for disturbances in a cooperative manipulation process.

## **CHAPTER 5**

### **METHODOLOGY FOR RESIDUAL REINFORCEMENT LEARNING**

#### **APPROACH**

In this chapter, we will provide details of the proposed residual reinforcement learning approach. We will present the structure of the chosen model and algorithm used to train this model.

### **5.1 Residual Reinforcement Learning**

In this section, we will present a compensation strategy that is based on residual reinforcement learning. We will present the mathematical formulations for residual RL and how we applied these formulations to our cooperative manipulation setup.

#### 5.1.1 Problem Statement

It is often the case that an accurate model of the cooperative setup is unknown, i.e the grasp matrix can not be accurately measured. Alternatively, even with an accurate model of the grasp matrix, it is possible that one or more of the cooperating robots do not apply the appropriate wrenches, as calculated by Equation 1.7. There are many reasons why a partner robot might not apply the appropriate end-effector wrench. Imagine a scenario where two robots (robot A and robot B), with different specifications, are tasked with co-manipulating an object, but robot B is unable to produce the desired end-effector wrench because its joints have been saturated. In order for the robots to successfully complete the given task, robot A must somehow compensate for the limitations of robot B. Another scenario is one where one of the cooperating partners is a human. In such a case, it is impossible for the human to precisely follow the desired end-effector wrench trajectory expressed in Equation 1.7. This consequently leads to poorer object trajectory tracking performance or complete failure of

the cooperative objective.

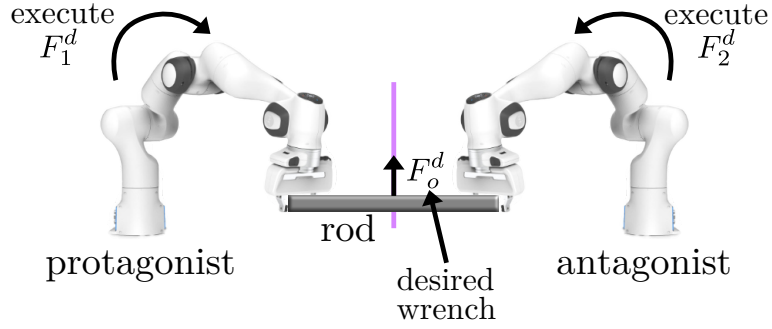
We consider the cooperative setup involving just two robots (i.e.  $K = 2$ ). Figure 5.1 shows a comparison of the ideal versus actual behavior when two robots jointly manipulate an object. In the ideal case, both robots apply the appropriate end-effector wrenches  $(F_1^d, F_2^d)$  to achieve the desired object frame wrench  $F_o^d$ . However, in a scenario with disturbance, both robots apply wrenches that are different from the desired wrenches. From the first robot's perspective, the second robot applies an end-effector wrench that deviates from the appropriate wrench  $F_2^d$  by a measure of  $\Delta_2$ . Similarly, from the second robot's perspective, the first robot applies an end effector wrench that deviates from the appropriate wrench  $F_1^d$  by a measure of  $\Delta_1$ . We describe these scenarios from each robot's perspective because it allows us to introduce the idea of a decentralized compensation scheme.

In this chapter, we will show how to learn the residuals,  $\Delta_1, \Delta_2$ , that must be added to each robot's standard controller, end-effector wrench, in order to satisfy the original cooperative manipulation objective. This is achieved in spite of deviation from the appropriate end-effector wrenches by the cooperating partners. In other words, we will show how the first robot can compensate for  $\Delta_2$  such that cooperative objective,  $F_o^d = G^1 F_T^d$ , is satisfied, and similarly for the second robot.

### 5.1.2 Compensating for Errors in Cooperative Manipulation

Consider a cooperative setup where two robots are tasked with co-manipulating an object and ensuring that the object's Center of Mass (COM) tracks a given trajectory. Also, consider that both robots have the visual sensing capability to determine the pose of each robot's grasp location, with respect to the object's COM. This information is necessary to build the grasp matrix described in subsection 1.1.3. The robots may share information, such as, desired object trajectory, object mass, etc., prior to the commencement of the task. The only communication constraint is that both robots cannot communicate with each other once the manipulation process has begun.

ideal system:



with disturbance:

$$\begin{array}{cc} \text{antagonist} & \text{protagonist} \\ \text{may execute} & \text{may execute} \\ F_2^d + \Delta_2 & F_1^d + \Delta_1 \end{array}$$

Figure 5.1: Error in cooperative manipulation [51]

Now, imagine that one of the robots is involved in a collision or has one of its joints saturated, and cannot contribute the appropriate forces needed to accomplish the task. A decentralized compensation strategy must be adopted by the other robot to account for the limitations of the robot in collision. To simplify the notation, we will define the robot without disturbance as the *Protagonist* and the robot with disturbance (or collision), as the *Antagonist*.

In an ideal scenario without disturbance, both robots will apply forces computed by each of their nominal controllers. In other words, both robots will use Equation 1.7 to arrive at  $F_G^1, F_G^2$ . This implies that the *protagonist* will apply a force of  $F_G^1$  and expect the *antagonist* to apply  $F_G^2$ , and vice-versa for the *antagonist*. However, the real scenario with disturbance on the *antagonist* results in *antagonist* applying a force that is different from its expected load contribution ( $F_G^2$ ). Therefore, according to the residual reinforcement learning framework, the final control input ( $F_1^d$ ) applied by *protagonist*, to compensate for the disturbance on the *antagonist*, is the superposition of the nominal controller output ( $F_G^1$ ) with a learned policy output ( $\Delta F_\theta^1$ ) as [51]:

$$F_1^d = F_G^1 + \Delta F_\theta^1 \quad (5.1)$$

So far, in simulation, residual RL has shown promising results in improving the object trajectory tracking performance of a dual-arm system, despite disturbance on one of the cooperating robots [51].

### 5.1.3 Learning a Residual

Figure 4.1 shows the layout of the model we trained to learn the residual for the protagonist. The input to the model is a concatenation of six vectors, each being a  $6 \times 1$  vector. The first vector is the *grasp-matrix-computed*<sup>1</sup> wrench for the protagonist ( $F_G^1$ ) and the second component is the grasp-matrix-computed wrench for the antagonist ( $F_G^2$ ). This formulation assumes that the grasp matrix is known to the protagonist. The third vector is the measured wrench ( $F_1^m$ ) at the end-effector of the protagonist. On the hardware, this would come from a force-torque sensor. In simulation this is obtained by querying the forces at the end-effector. The fourth component is the end-effector pose ( $EE_{pose}$ ) of the protagonist. Finally, the fifth and sixth components are the desired object frame pose ( $e_d^t$ ) and measured object frame pose ( $e_m^t$ ), respectively. Together, these components form the  $36 \times 1$  vector which is flattened and passed as input to a neural network model.

In residual reinforcement learning, part of the input to the model which has been labelled as *robot wrenches* in Figure 4.1, is the output from a nominal controller ( $F_G^1, F_G^2$ ) that has been computed in Equation 1.7. The model predicts a residual term that is superposed with the output of the nominal controller, as shown in Equation 5.1. We refer the reader to [68, 69, 70, 71] on how a neural network is trained and used for inference in a reinforcement learning scheme. Therefore, our model predicts the residual term  $\Delta F_\theta^1$  for the protagonist. This residual term is then added to the output of the *grasp-matrix-computed*

---

<sup>1</sup> *grasp-matrix-computed* refers to the wrench directly obtained by multiplying the inverse of the grasp matrix with the desired object frame wrench as shown in Equation 1.7.

wrench ( $F_G^1$ ) to obtain the wrench that is then applied to the system by the protagonist.

#### 5.1.4 Neural Network Architecture and Reward Function

Several applications of Deep RL to robotics [72, 73, 74] have adopted the *PPO* algorithm [75, 76] because of its inherent policy stability [77, 78]. Hence, we chose PPO as our learning algorithm, and in accordance with the PPO framework, we designed a neural network architecture with two heads; the first head (policy head) has an output layer with six nodes (representing the  $6 \times 1$  residual vector  $\Delta F_\theta^1$ ) and a hidden layer with 8 nodes; the second head (value head) has an output layer with just one node, which represents the prediction value, and a hidden layer with 4 nodes. In addition to the aforementioned hidden layers, both heads share hidden layer(s).

As the number of nodes in a neural network increases, the number of model parameters also increases; this could slow down learning. On the other hand, too few number of nodes (and/or hidden layers) might not be able to capture the complexity of the problem. The consensus in the machine learning community is to try to find the simplest model that best captures the complexity of the task. Hence, we started with just a single hidden layer for each head and a single hidden layer shared by both heads. We then gradually increased the number of nodes and observed the performance. We noticed that adding more hidden layers had no positive impact on the overall performance. Figure 5.2 and Figure 5.3 shows the schematic of these architectures.

The main objective of the cooperative process is to minimize the trajectory tracking error of the object frame. Therefore, the reward function is defined by the negative norm-squared-error of the object pose, as shown in Equation 5.2:

$$4.0 - \phi^2 \tag{5.2}$$

where  $\phi^2$  is the object pose error norm of the object pose. We biased the reward function with a positive value of 4.0 to keep the reward within low range values.

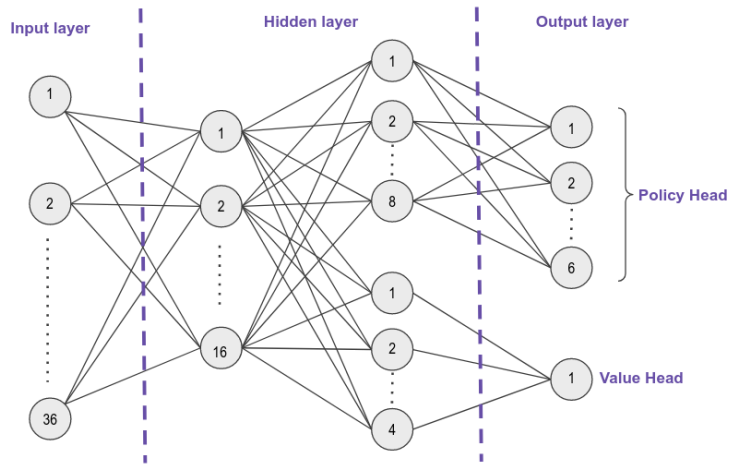


Figure 5.2: Neural network model for Gaussian noise experiment

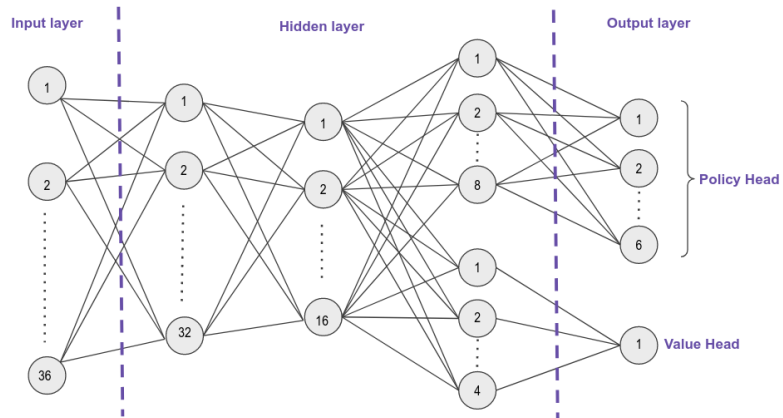


Figure 5.3: Neural network model for human-teleop experiment

## CHAPTER 6

### EXPERIMENTS AND RESULTS FOR RESIDUAL REINFORCEMENT LEARNING APPROACH

To validate our residual reinforcement learning approach, we conducted experiments in simulation. We also obtained human teleoperated trajectories on the hardware, with which we used to train a reinforcement learning model, offline.

#### 6.1 Evaluation approach

As we discussed in chapter 3, we focus our evaluation protocol on the COM tracking performance. To evaluate the efficacy of the residual RL approach, the experiment setup was divided into two components. The first experiment includes an environment where the antagonist is comprised of a nominal controller that is superposed with a Gaussian noise as a form of system disturbance. The second experiment uses data collected on a human teleoperated (real world) robot via a VR interface. The teleoperation trajectories are then deployed on the antagonist robot in simulation and used to train the protagonist. In simulation, we had access to the COM of the object and the grasp location of each robot. With this information, we were able to build the grasp matrix required by the nominal controller. We acknowledge that replicating the same procedure in the real world would likely require a camera tracking system.

We conducted the simulation experiments in the *Pybullet* simulator [79], and we used the *OpenAIgym* framework to setup the training pipeline [80]. *OpenAIgym* abstracts the training scheme, allowing the user to simply select a training algorithm and follow the training procedure for that algorithm. For instance, and as mentioned earlier, the *PPO* algorithm which we used in our experiments, requires a two-head neural network architecture.

Table 6.1: Table showing list of parameters and their values

Name	Symbol	Value
Mean object pose error norm	$\mu_o^E$	–
Multivariate Gaussian mean for training	$\mu_{train} \in \mathbb{R}^6$	$[(0.05)_{\times 6}]^T$
Multivariate Gaussian covariance for training	$\Sigma_{train} \in \mathbb{R}^{6 \times 6}$	$diag([(0.05)_{\times 6}]^T)$
Multivariate Gaussian mean for testing	$\mu_{test} \in \mathbb{R}^6$	$[(0.1)_{\times 6}]^T$
Multivariate Gaussian covariance for testing	$\Sigma_{test} \in \mathbb{R}^{6 \times 6}$	$diag([(0.1)_{\times 6}]^T)$
Learning rate for Gaussian experiment	$\lambda_{Gauss}$	$5.5 \times 10^{-6}$
Learning rate for Human-teleop experiment	$\lambda_{Hum}$	$2.5 \times 10^{-6}$
Max. number of time steps in an episode	$T$	30K
# training steps in Gaussian experiment	–	500K
# training steps in Human-teleop experiment	–	1.2M

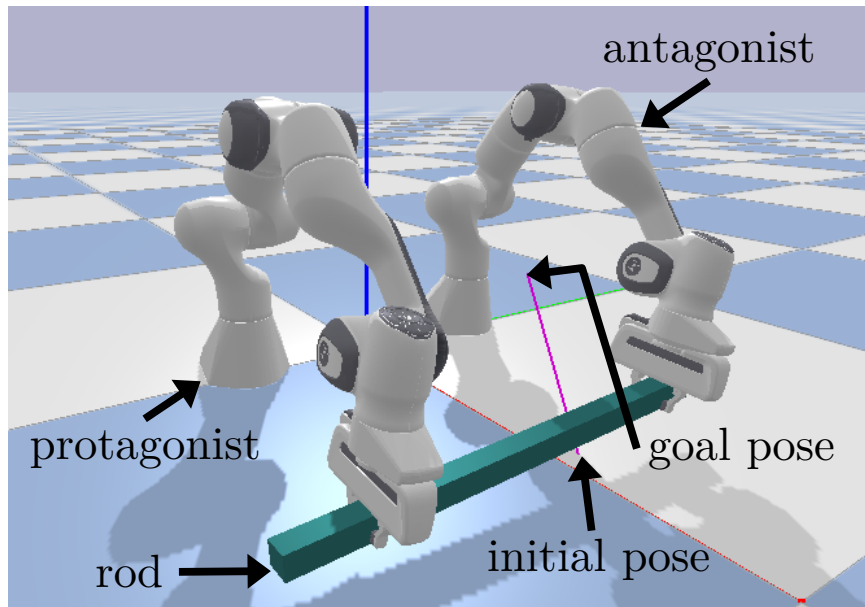


Figure 6.1: Two manipulators that cooperatively lift a 0.8m rod from an initial pose to a desired goal pose. The protagonist manipulator tries to compensate for errors of the antagonist manipulator [51].

The object goal poses for both sets of experiments were set-points. However, we created straight line interpolations between the start position and the goal position of the object’s COM, as shown in Figure 6.1. The desired orientation is left constant. Hence, the goal becomes a desired trajectory that is a function of the simulation time step.

$$x_{o_p}^d(t) = x_{o_p}^{Start} + \frac{t}{T}(x_{o_p}^{Goal} - x_{o_p}^{Start}) \quad (6.1)$$

where  $x_{o_p}^d(t) \in \mathfrak{R}^3$  is the desired position of the COM, with respect to the inertial frame at time-step  $t$ , and  $T$  is the number of time steps in one episode.

To evaluate a trained policy, we ran such policy *–in the residual RL fashion*<sup>1</sup>– for a full episode ( $T = 30,000$  time steps). Then, we computed the mean object pose error norm ( $\mu_o^E$ ) for that particular episode.

$$\mu_o^E = \frac{1}{T} \sum_{t=0}^{t=T} \|x_o(t) - x_o^d(t)\| \quad (6.2)$$

where  $x_o(t)$  is the pose of the COM frame and  $\|x_o(t) - x_o^d(t)\|$  is considered the object pose error norm at time step  $t$ .

An episode is considered successful if the object is not dropped for a total of 30,000 time steps. Hence, during training, an episode might be unsuccessful due to the object being dropped before 30,000 time steps elapse. However, this episode is still used to learn the model. During testing, only a successful episode is evaluated. We observed that after training, the resulting policy is able to keep the object in grasp to achieve a successful testing episode.

For statistical purposes we also evaluate the same policy on multiple different episodes drawn from different parameters such as, goal poses, Gaussian noise distribution seeds and human-teleoperated trajectories. The idea of simply selecting and reporting experiments

---

<sup>1</sup>*residual RL fashion* refers to training or evaluating a policy while the policy is superposed on a nominal control, as shown in equation (Equation 5.1)

from a single successful random seed was castigated by [81]. According to [81], it was better to represent the variability across seeds by presenting the performance of the trained agent across multiple seeds. The authors of [82] also used multiple seeds in their experimental validation of the *behavior suite*, which they developed. Therefore, while experimenting, we made sure that multiple policies were trained from different seeds, as shown in Table 6.2 and Table 6.3.

For the Gaussian experiment, both heads share a hidden layer with 16 nodes (see Figure 5.2). For the human-teleoperated data experiment, both heads share two hidden layers with 32 and 16 nodes each (see Figure 5.3). The network used for the human-teleop experiments has an extra hidden layer because the intuition was that human trajectories may be more complex than the Gaussian-noise-reinforced trajectories. Hence, we considered that an extra layer might better capture these complexities. The choice of the two-headed architecture is in accordance with the *PPO* training algorithm used, which requires both the policy and value prediction, hence the two-headed structure.

### 6.1.1 Gaussian Noise Environment

In this section, we evaluate the performance of a residual RL approach in minimizing the impact of Gaussian noise on the object trajectory tracking task. This Gaussian noise is introduced to the system by the Antagonist robot. More specifically, a multivariate Gaussian is superposed on the output of the standard controller for the Antagonist.

#### *Gaussian Noise Environment - Setup*

The Gaussian noise used in the training environment has a mean  $\mu_{train} \in \mathbb{R}^6$  with a value of 0.05 for each axis, and a diagonal covariance matrix  $\Sigma_{train} \in \mathbb{R}^{6 \times 6}$  with diagonal entries of 0.05. For the test phase, we used a Gaussian noise with mean  $\mu_{test} \in \mathbb{R}^6$  with a value of 0.1 for each axis, and a diagonal covariance matrix  $\Sigma_{test} \in \mathbb{R}^{6 \times 6}$  with diagonal entries of 0.1. These mean and covariance values were selected such that there was some overlap

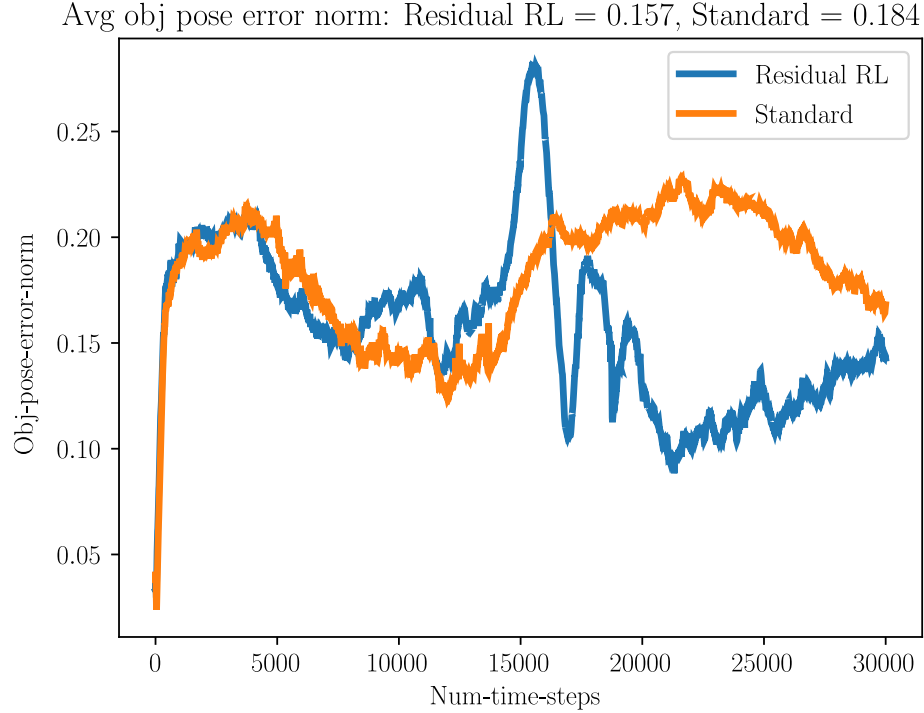


Figure 6.2: Policy evaluation for the standard controller and the residual learner in the presence of Gaussian noise.

in the training and test distribution. However, we expect that during testing, the agent will encounter noise samples not seen during training. We defined  $[0.0, 0.7, 0.02, 0.0, 0.0, 0.0]$  and  $[0.0, 0.5, 0.3, 0.0, 0.0, 0.0]$  as the object starting and goal pose, respectively. We used the learning rate,  $\lambda_{Gauss} = 5.5 \times 10^{-6}$  to tune the gradient descent in the backpropagation step. Backpropagation is used to learn the weights to minimize the training loss [70]. To establish a consistent environment when comparing both the standard approach and the residual RL approach, we specified a seed for sampling from the Gaussian distribution. We trained each policy for 500K time steps, with each time step being the *Pybullet* default length of  $1/240s$ . Finally, both robots were controlled by sending joint torque commands computed via equation (Equation 2.14).

Table 6.2: Mean object pose error norm - Gaussian Noise (A lower value implies better performance)

Policy	GN-1	GN-2	GN-3	GN-4	GN-5	Mean
<b>1</b>	<b>0.157</b>	0.362	0.182	0.209	0.157	0.213
<b>2</b>	0.239	0.156	0.217	0.224	0.273	0.222
<b>3</b>	0.178	0.244	0.186	0.251	0.261	0.224
<b>4</b>	0.249	0.298	0.214	0.255	0.251	0.253
<b>5</b>	0.272	0.233	0.260	0.265	0.168	0.240
<b>Mean</b>	0.219	0.258	0.212	0.241	0.222	—
<b>SC</b>	0.184	0.215	0.209	0.248	0.249	0.221

### *Gaussian Noise Environment - Results*

We implemented training with five different seeds, and evaluated the policy from each seed across five seeds on the Gaussian noise distribution. Table 6.2 shows the mean object pose error norm for each episode from each of these episodic evaluations. From top to bottom, different policies are trained with different seeds. From left to right, different seeds are used to generate the Gaussian noise distribution (*GN-*). *SC* is shorthand for *standard controller*. Figure 6.2 compares the instantaneous object pose error norm over the distinct test episodes where the standard approach and the residual RL approach performed the best. The residual RL approach achieved the best performance (0.157) on the test episode with policy seed 1 and GN-1. The standard controller also performs the best (0.184) on the test episode with GN-1. Finally, the results in Table 6.2 show that it is possible to find a policy that outperforms a standard controller in terms of minimizing the effect of disturbance on the cooperative manipulation system. In our case, policy 1 is the best policy, with an average mean of 0.213 across the five seeded Gaussian noises.

#### 6.1.2 Human-teleoperated Data Replay Environment

While introducing Gaussian noise as a form of disturbance on the system might reveal the efficacy of the proposed residual RL method, human-teleoperated data replay, in which the human partner certainly has no explicit adherence to the grasp matrix formulation (see

Equation 1.7), provides an even stronger argument for our proposed approach. Therefore, in this section, we evaluate the impact of residual RL on improving the object trajectory tracking of a cooperative system, in the presence of an Antagonist robot that is not controlled by the grasp matrix formulation. Instead, the Antagonist robot is controlled by replayed human-teleoperated trajectories.

#### *Human-teleoperated Replay Data - Setup*

To ensure the successful replay of hardware trajectories in simulation, we tried to replicate the hardware setup in simulation. To achieve this, without the aid of a camera tracking system for the object pose, we had to take measurements of the initial and final object pose. It is important to note, precise measurements were not necessary because minor mismatches between the hardware setup and the simulation setup could be classified under the cooperative model errors we are trying to address. Therefore, although we took measurements of the initial and final object pose, we did not use exactly the same values in simulation. Moreover, the properties of the object, such as the mass and the dimensions, were not shared between the hardware setup and simulation setup. These measurements only informed our choices in simulation.

To simplify the task of data collection, a human held the object on one end, and the robot, a Franka Emika model, grasped the object on the other end. The end grasped by the robot was teleoperated by a human using the Oculus VR interface. The commands to the robot, from the VR, were Cartesian velocities. However, the recorded trajectories were joint positions. Therefore, when replaying the recorded trajectories for the Antagonist robot in simulation, we simply used joint position control. We controlled the Protagonist robot by sending joint torque commands computed via equation (Equation 2.14).

On the hardware, we collected 25 trajectories starting from the same initial position. The goal was to bring the COM of the object to a desired pose. Finally, we split our trajectories into a training set and a test set with an 80/20 ratio, respectively. In simulation,

Avg obj pose error norm: Residual RL = 0.282, Standard = 0.364

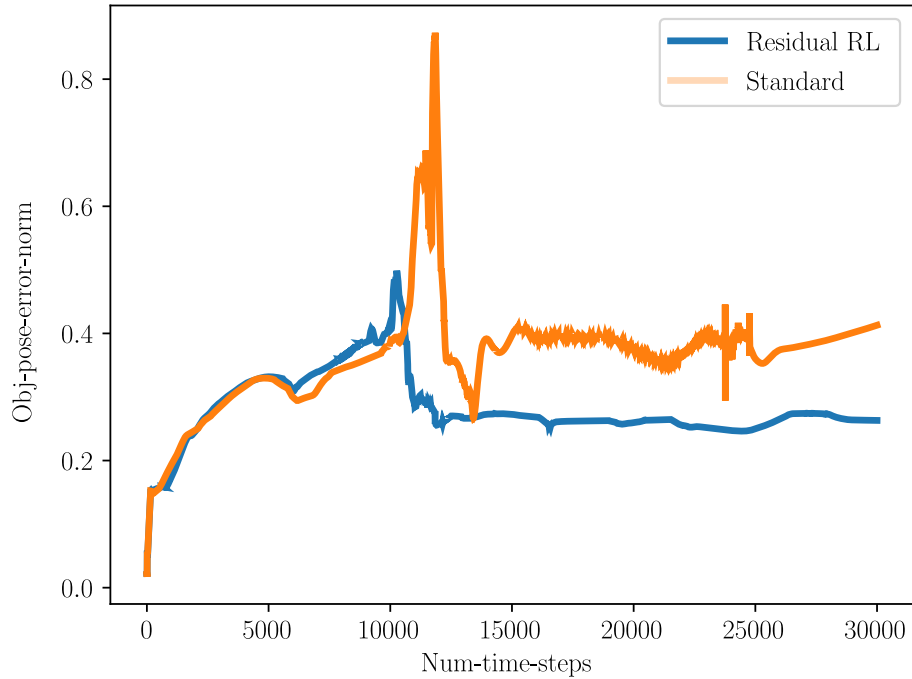


Figure 6.3: Policy evaluation on trajectory 1 (Traj-1) for both the standard controller and the residual learner.

while replaying the recorded trajectories, we defined

$$\mathbf{x}_d = [0.12, 0.41, 0.38, 0.0, 0.0, 0.0]^\top$$

as the desired object pose. We arrived at these values by computing the mean of the final end effector positions of the test trajectories, and adding the offset between the grasp point and the object’s COM. The implication of this is that although none of the test trajectories has a final pose akin to the given desired object pose defined above, using this particular desired object pose ensures that the final object goal pose is still within a reasonable region with respect to the replayed test trajectories. Finally, we trained for 1.2 million time steps with a learning rate,  $\lambda_{Hum} = 2.5 \times 10^{-6}$ .

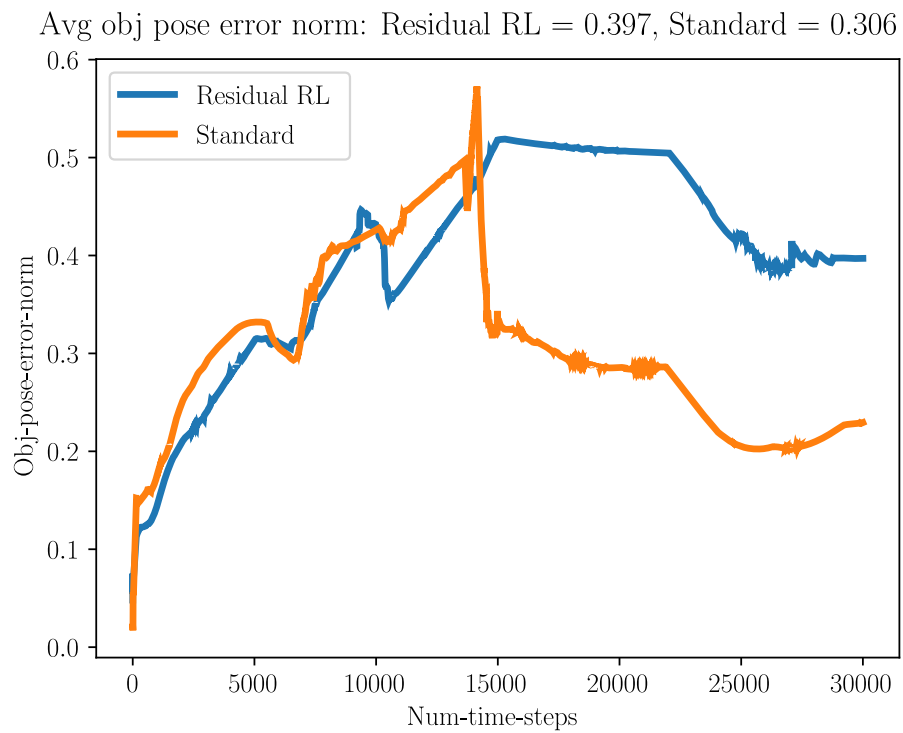


Figure 6.4: Policy evaluation on trajectory 5 (Traj-5) for both the standard controller and the residual learner.

Table 6.3: Mean object pose error norm - Human-teleoperated Data (A lower value implies better performance)

Policy	Traj-1	Traj-2	Traj-3	Traj-4	Traj-5	Mean
<b>1</b>	<b>0.282</b>	0.348	0.327	0.407	0.397	0.352
<b>2</b>	0.373	0.292	0.477	0.353	0.446	0.388
<b>3</b>	0.405	0.357	0.359	0.316	0.409	0.369
<b>4</b>	0.368	0.356	0.391	0.347	0.391	0.371
<b>5</b>	0.347	0.374	0.387	0.389	0.434	0.386
<b>Mean</b>	0.355	0.345	0.388	0.362	0.415	—
<b>SC</b>	0.364	0.425	0.382	0.343	0.306	0.364

### *Human-teleoperated Data Replay Environment - Results*

Table Table 6.3, shows the mean object pose error norm obtained by evaluating five policies from different training seeds. However, instead of having different seeded Gaussian noises superposed on the Antagonist’s nominal controller, as we did in the previous experiment, we evaluated each policy by replaying the five test trajectories on the Antagonist. The results shown in table Table 6.3 indicate that it is possible to find a policy that outperforms a standard controller with regards to minimizing the object pose error norm. In our case, policy 1 performs the best with a mean of 0.352 over the five test trajectories. It is evident that the residual learner performed the worst on trajectory 5, with a mean of 0.415. This might be due to the fact that, besides trajectory 3, trajectory 5 is one of the farthest trajectories from the mean trajectory, as the Figure Figure 6.5 shows. Unsurprisingly, the next highest mean across the policies is on trajectory 3, with a mean of 0.388. Figure 6.3 shows the plot of the object pose error norm for the test trajectory on which one of the five policies (policy 1) performed the best. The figure also shows the plot for the standard controller on the same trajectory. Figure Figure 6.4 shows the plot for the trajectory on which the standard controller performed the best (Traj-5). This figure also shows the plot for the best performing policy on the same trajectory, which is also policy 1.

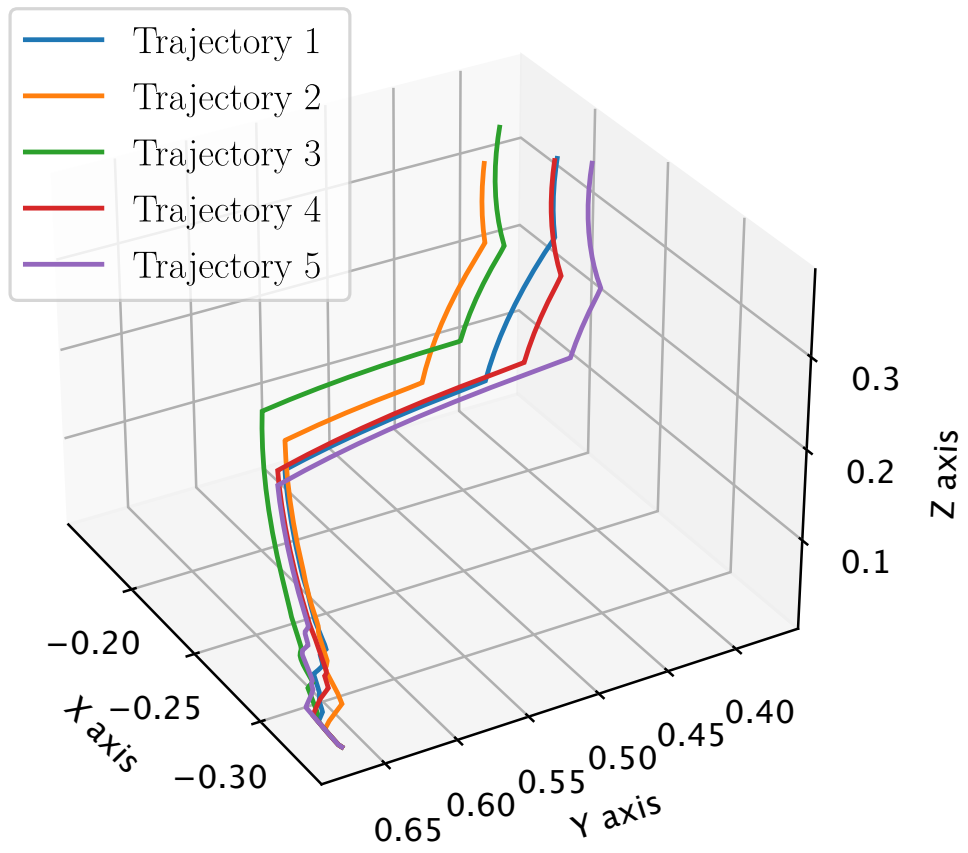


Figure 6.5: Plot of the end effector 3D positions for the test trajectories. Trajectories start at the bottom left and end at the top right.

## **CHAPTER 7**

### **DISCUSSION**

So far, we have presented two approaches to address disturbances and model errors in cooperative manipulation. In this chapter, we will discuss some of the broader impacts of the presented approaches and the limitations that surround these approaches.

#### **7.1 Collision Reaction via Internal Loading**

The formulated method implicitly introduces an extra degree of freedom into the system which is used to compensate for disturbance on the cooperative system. While there are other ways to compensate for disturbance without applying internal stress on the manipulated object, especially with manipulator systems that are already redundant with respect to the task, leveraging this extra degree of freedom could enable optimization for other system criteria such as joint limits, collision avoidance, etc.

Another benefit of this approach is that the arms could also compensate for deviation in one or both of the robot bases. In simple terms, if both robot bases were to drift away from the desired trajectories, the arms could apply the presented internal stress compensating scheme, such that the pose of the object is not impacted by the deviation of the robot bases. Obviously, a bound must be placed on how much the bases are allowed to drift, before the arms have to let go of the object, so as not to destroy the object or the robots. There are also self-induced disturbances on the system such as the dynamic response of the arms to the movement of the base, that can impact the system from tracking the desired object trajectory. Our method can act as a compensation mechanism for such disruption.

The choice of  $\alpha$  defined in this paper presents an opportunity to extend the number of manipulators involved in the cooperative manipulation process and creates the possibility of specifying unique roles for the compensating robots [36]. For example, assume we extend

the number of robots used from two to three, and these three robots (A, B, C) are involved in the joint manipulation of an object. When robot A is involved in a collision, a compensation wrench comprising of both forces and moments (i.e.  $h_A^{ext} = [f_A^{ext} \in \mathbb{R}^3, m_A^{ext} \in \mathbb{R}^3]$ ) will be required to counter the effect of the collision on robot A. Using this representation of the tuning factor, we can assign force compensation to robot B and moment compensation to robot C. The roles could also be assigned based on the load capacities of robots B and C. For example, if robot B has twice the load carrying capacity of robot C, this matrix form representation of  $\alpha$  makes it possible to specify the distribution of the compensating wrench to match the load capacity ratio between robots B and C. This approach is, in essence, a form of load distribution that can be used to satisfy several constraints. The topic of load distribution has been extensively studied by several works, including [29, 16, 83].

One drawback of this impedance-based tuning factor approach is that since the compensation wrench is now dependent on the object's COM pose error, as shown in Equation 2.15, COM pose errors that are not caused by the collision will also affect the compensation scheme. For instance, if there is a COM pose error, possibly due to inaccurate joint torque control, the values of  $\alpha$  are affected due to Equation 2.15, Equation 2.17 and Equation 2.19, regardless of whether or not a collision has occurred. Also, we initially observed large oscillations in the values of  $\beta(j)$  which were traced to the fact that *GAZEBO*'s COM pose velocity measurements had some oscillations. These oscillations are transferred to  $\beta(j)$  due to equation (Equation 2.15). Hence, we filtered the COM velocity measurements to eliminate a high frequency response of  $\beta(j)$ . Nonetheless, the proposed partial compensations scheme still holds as shown in the plots for the z-position and y-orientation, where the impact of the collision force causes the COM pose error to increase. In this case,  $\alpha$  decreases, thereby reducing the amount of applied compensation wrench.

While implementing our approach on the hardware. We found tuning the gains in (Equation 2.10) to be quite challenging; moreso with the added weight of the object. Certain gains were too low, causing the robots to gradually drop the object. Some other gains

were quite high, causing the robots to respond quite aggressively. Hence, finding the right gain values took some effort on our part. We started with low gains, and gradually increased the gains, while observing the response of the robots. We started with lower gains because we felt it was safer to do so; extremely high gains were likely to lead to aggressive motions and lead to system damage.

## 7.2 Compensating for Cooperative Model Errors via Residual RL

The results we obtained, demonstrate the potential for residual RL in the cooperative manipulation space. One major challenge we faced was ensuring rigid grasps during training. On the hardware, we used custom-made gripper tips and an object with surface properties that allowed for rigid grasps. However, in simulation, we had to specify certain constraints through *Pybullet*, to ensure that the robots maintained their grasps on the object. However, since these constraints only work up to a certain force limit, there were still several times during training, that the object was dropped. This, consequently, impacted the length of training, since an episode is reset each time the object is dropped. We also placed bounds on the action space of the policy to ensure that large wrenches that violate the grasp constraint, beyond the constraints added through *Pybullet*, were not produced by the policy.

Moreover, remember that we are only trying to learn a *residual*. This means that the output of the model should be relatively small, or at least smaller, compared to the output of the standard controller. Hence, it is possible that the bounds placed on the network output limit the improvement of the residual learner over the standard controller.

Furthermore, according to [66] there might be situations where the standard controller and the learned policy are in disagreement, leading to a diminished performance when compared to just the standard controller. Hence, like [66] described, we hope to investigate this potential problem in our future work. As stated before, in order to replay the trajectories that we collected on the hardware, in simulation, we set up the simulation environment to match that of the hardware. However, a more efficient and robust approach, which we

propose for future work, is to train an Antagonist policy via behavior cloning of the human-teleoperated trajectories on the hardware.

Finally, we acknowledge that there is still room for improvement within this framework. Hence, we postulate that better reward shaping, efficient hyper-parameter tuning and exploring more complex models (e.g. more neural network layers or using LSTMs) might improve the performance of our proposed approach. The scope of possible reward functions is inexhaustible. Hence, while we could find a better reward function, we cannot guarantee an optimal reward function. However, the reward function we used provides the advantage of being simple and easy to debug. Similarly, there are infinite neural network architectures and hyper-parameters that can achieve similar or better results than the ones we have obtained. We erred on the side of simplicity when designing our model. There is the possibility that our problem is more complex than we imagine; to obtain significant object pose tracking improvement with our residual RL model, we may need to use a temporal model such as LSTMs. LSTMs are capable of capturing temporal information [84] that could be quite relevant to the problem at hand.

### **7.3 Comparison between the Internal Stress Loading approach and the Residual RL approach**

As we have discussed so far, there are advantages, as well as limitations, to the two compensation approaches we have presented in this thesis. Here, we compare the benefits and drawbacks of both approaches.

While learning schemes, such as the one applied in the residual reinforcement approach, are useful in dealing with unknown disturbances, they are very susceptible to the training protocol used. Moreover, there are a number of hyper-parameters that significantly impact the results, even with only slight changes to their values. On the other hand, the internal stress loading approach has fewer hyper-parameters, whose effect on the system, are less pronounced.

The most significant advantage of the residual RL approach over the internal-stress-loading approach is that the former performs compensation in a decentralized manner, whereas, the latter requires the *colliding* robot to communicate the disturbance values to the *compensating* robot. Therefore, if there are communication constraints, the residual RL approach is recommended, otherwise, the internal stress loading method is recommended. Interestingly, future work could explore how to combine and leverage the benefits of both approaches.

## CHAPTER 8

### CONCLUSION AND FUTURE WORK

In this thesis, we have presented two novel approaches for compensating for disturbance in a cooperative system. The first approach is based on the idea of internal stress loading of the co-manipulated object; the second approach involves a decentralized scheme that employs residual reinforcement learning a strategy for dealing with cooperative model errors.

The presented impedance-based compensation strategy ensures that large compensation wrenches are limited, at the expense of larger deviations in the desired object COM trajectory, while smaller compensation wrenches are permitted since the resulting internal stress on the object is lower [36]. Future work will extend this framework to more than two robots, where more complex compensation role assignment can be performed. Moreover, the cooperative manipulation task used to validate our approach was designed such that motion planning of the cooperative system was not necessary. In other words, desired object trajectories or goal poses that require self-collision avoidance to be implemented, have been avoided. Therefore, future work will include more complex desired object trajectories, such that, a cooperative motion planning scheme will be required to successfully accomplish the task [85, 86].

Additionally, we have demonstrated that residual reinforcement learning can be used as a compensation strategy for errors in a cooperative manipulation setup. More specifically, we learned to compensate for inappropriate wrenches applied by the *antagonist*, without strict assumptions on the behavior of the *antagonist*. The simulation experiments show that the combination of a classical control approach with the learned residuals for unknown cooperative errors, improves the object tracking performance.

Future work will include improving the trained model, e.g., with domain knowledge about the specific task [51]. Furthermore, we propose making additional comparisons be-

tween the presented residual RL approach and other classical control approaches. This framework might also be extended to a cooperative setup involving more than two robots.

The idea of compensating for the unexpected behaviors of a cooperating partner, without communication between partners, brings us a step closer to a fully decentralized cooperative manipulation framework. Moreover, this allows for the adoption of a human partner in place of a robot partner. Hence, human-robot collaboration becomes more feasible when the robot partner does not need to communicate with its partner to account for deviations in the desired contribution of this partner. Therefore, future work should include evaluating the developed residual RL algorithm in human-robot cooperative manipulation scheme.

Another major advantage of the residual RL approach is the flexibility to optimize for additional features of the cooperative system. For instance, instead of only learning a compensating action for the *protagonist* to minimize the trajectory error of the object, we could also train a model that minimizes the effort of the human (or robot) partner. This could be easily incorporated in the reward function of the agent.

Another interesting angle worth exploring is the idea of introducing disturbance on both partners, instead of just one partner, as we have done for the *antagonist*, in this work. This would require establishing a consensus between both robots in a decentralized manner, which could be quite challenging. In other words, if both robots were to introduce disturbance into the system, then an attempt by each robot to compensate for the other's aberrant behavior, could lead to catastrophic outcomes. Therefore, one possible approach to addressing this issue could be the idea of learning a *sibling-policy*, in which all the cooperating partners are trained together and during test time, they all have to be part of the cooperative setup. The reward function for training this model will include a *sibling-prediction-error* term that provides a higher reward to the agent when the prediction of its behavior by its siblings is more accurate. The intuition here is that the robots would learn the behavior of their partners during training and also try to take actions that increases the prediction accuracy of its behavior, by its siblings. This can be considered some sort of

*reverse-adversarial* learning, or in short, *cooperative learning*. The glaring limitation of this approach is that the robots all have to be part of the cooperative system during test time. Moreover, this most likely will not work for a human-robot cooperative setup.

In closing, cooperative manipulation is extremely applicable with today's needs for robotic assistance. Therefore, as we develop cooperative manipulation strategies for multi-robot systems, it is important to consider human-robot systems. The decentralized scheme we have presented here, takes us a step further in that direction.

# **Appendices**

## APPENDIX A

### EXPERIMENTAL EQUIPMENT

The setup included two KUKA iiwa14 arms, each mounted on Clearpath's Ridgeback robots (see Figure 1.1).

#### A.1 Communication Setup

I used two personal computers to setup communication for the entire system. Both computers were synchronized via Robot Operating System (ROS) [87]. Each robot base, on startup, have their own ROS masters running. Hence, a total of three ROS masters were used to connect the entire system together. Therefore, to successfully use the ROS framework, I used a ROS multi-master scheme [88] to allow all three masters to communicate with one another. I was able to use wireless communication, via Wi-Fi connection, to send commands and receive data from each robot base.

Each manipulator was connected to a personal computer using an ethernet cable and the Fast Research Interface (FRI) [89] was used to set up communication between the each PC and the corresponding manipulator. Unfortunately, it was not possible to use Wi-Fi to communicate with the manipulators because the KUKA sunrise software, which is responsible for the low-level manipulator software, requires a communication speed of up to 200 Hz; this cannot be guaranteed with a Wi-Fi connection. Therefore, as shown in the supporting video of my experiments, there were ethernet cables that encumbered the motion of the robot bases as they moved around the workspace. Moreover, the smartpads with the emergency stop buttons, also had cables that ran from the personal computer to the robot. It is likely that when the robots wheels run over this cables, the trajectory tracking of the object pose is slightly impacted. Fortunately, the wires were able to withstand the weights of the robot bases without getting damaged. Had I had extra manpower, I would

have assigned someone to the role of holding the cables, without obstructing the camera.

## A.2 Safety Measures

I applied safeguards to the arm control by placing a threshold on the end effector wrenches commanded by the high-level controller. This prevents the low level controller from generating joint torques that can cause damage to the robot and its environment. In such a scenario, the low-level controller switches from an end effector force control, to an end effector pose-holding control. This simply means that each robot's low-level controller stops applying the end effector wrench commanded by the high-level controller and instead apply the appropriate joint torques to maintain the end effector at the current pose. Once the wrenches commanded by high-level controller returns to normal values (i.e. values below the set threshold), the end effector force control is restored, and the low-level controllers begin to apply the commanded end effector wrenches. One drawback of this is that a fast switching behavior between both modes (end effector force control and end effector pose-hold) might occur. Hence, the onus lay on us to ensure that this did not occur often. These threshold values were chosen empirically by observing the robot's behavior as the threshold values were slowly increased from low values to higher values.

Similarly, I placed bounds on the commanded mobile base velocities to ensure that the controller for the mobile base does not output unreasonably extreme values. Although, I took proper care to ensure that computed velocity commands did not exceed the thresholds I had set. This should be avoided because even though the robots' velocities are bounded by the safety threshold, the trajectory tracking performance will be impacted when the clipping by the safety guard occurs, Therefore, when I observed that the computed control velocities exceeded the set threshold, I lowered the gains of the base controller, or tuned other parameters, such as the amplitude and frequency of the desired trajectory. The velocity thresholds should only serve as a safety mechanism, and not a part of the control formulation. In addition, the synchronous scheme that was employed in the mobile base

control ensures a relative fixed pose between both bases. Therefore, a safeguard preventing both robots from colliding with each other, is inherently established. Although, this safeguard only works if both robots are tracking their assigned trajectories. If one of the robots gets stuck or suddenly loses power, this safeguard fails and it becomes likely that the robot that is still moving, crashes into the stationary robot.

Another safety measure I took, involved correcting the object's AR tag pose measured by the camera. Once in a while, at random, the values I obtained were unreasonable; values such as 300 m were outputted, which is simply not valid. One could attribute such errors to lighting conditions and other unknown factors. To address this issue, I added a check in the AR tag pose calculation subroutine; if the difference between the current pose and the previous pose – for each axis – was greater than a set threshold, I set the current pose to the previous pose. This idea was based on the notion that, being a real system, the difference between the current pose and the previous pose cannot be extremely high, after a very short amount time (matter of milliseconds). This would imply extremely high velocities, and if these unchecked values were applied in the control routine, it would introduce instability and unexpected behaviors into the system. In other words, within a short time frame, I expected the current pose of the object to be very close to the previous pose of the object. Apropos to regulating the computed pose values, I obtained the velocity of the object (see Equation 2.10), by computing the timed difference between the current pose and previous pose. I anticipated this would result in noisy values. Hence, I applied a moving average filter by creating a buffer that holds the computed velocity values from the last ten time stamps. When the buffer is full, the oldest entry is discarded and the latest entry is appended. At any point in time, the average of the buffer is used as the current velocity of the object.

Lastly, the software architecture for the *brain* of the entire cooperative setup was designed such that commands were published only when all three AR tags (i.e. AR tag on the object, *base-A* and *base-B*) were detected by the camera. This is to prevent a scenario where

one or more of the tags go out of scope of the camera, either due to distance or lighting conditions, causing the robots to receive commands that result in unexpected behaviors.

### A.3 Miscellaneous Robot Issues

While conducting my experiments, I observed a strange error (see Figure A.1), where the sunrise application of one of the robots, specifically the one shown in Figure A.2, suddenly crashes and the robot locks. The experiment is immediately terminated because both robots need to be running for the experiment to be valid. I had repeated experiences with this and searched the internet for the error message shown in Figure A.1, but the only result I got was with reference to the network communication speed. The suggestions online recommended using a *real-time Linux kernel* [90]. However, I observed that this error was only happening consistently on one robot. I also observed that this error was happening when the robot was in a specific configuration, which is the configuration of the robot marked with the red arrow in Figure A.2. This anomalous error disappeared when I changed the configuration of the robot before commencing my experiment.

To investigate this problem, I printed the joint positions for the erroneous robot to my computer terminal. It appears that this error occurs whenever the robot is in a configuration where one of the joints is at its limit. As it turns out, the last joint – written in bold in Equation A.1 – was indeed at the joint limit when the robot produced this error.

$$q = [-0.076, 0.205, -1.388, -1.380, -1.446, -1.565, \mathbf{-3.0543}] \quad (\text{A.1})$$

In conclusion, my proposed solution of changing the pose of the erroneous robot does not negate the recommendation of other researchers who have suggested that the issue might be related to the network communication speed. However, it offers a temporary solution to my readers who might be facing a similar problem. More investigation still needs to be done. Besides it is important to note that this error might be only relevant to

this particular sunrise application [91]. Colleagues in my lab have yet to report this type of error, when using DRAKE [92].

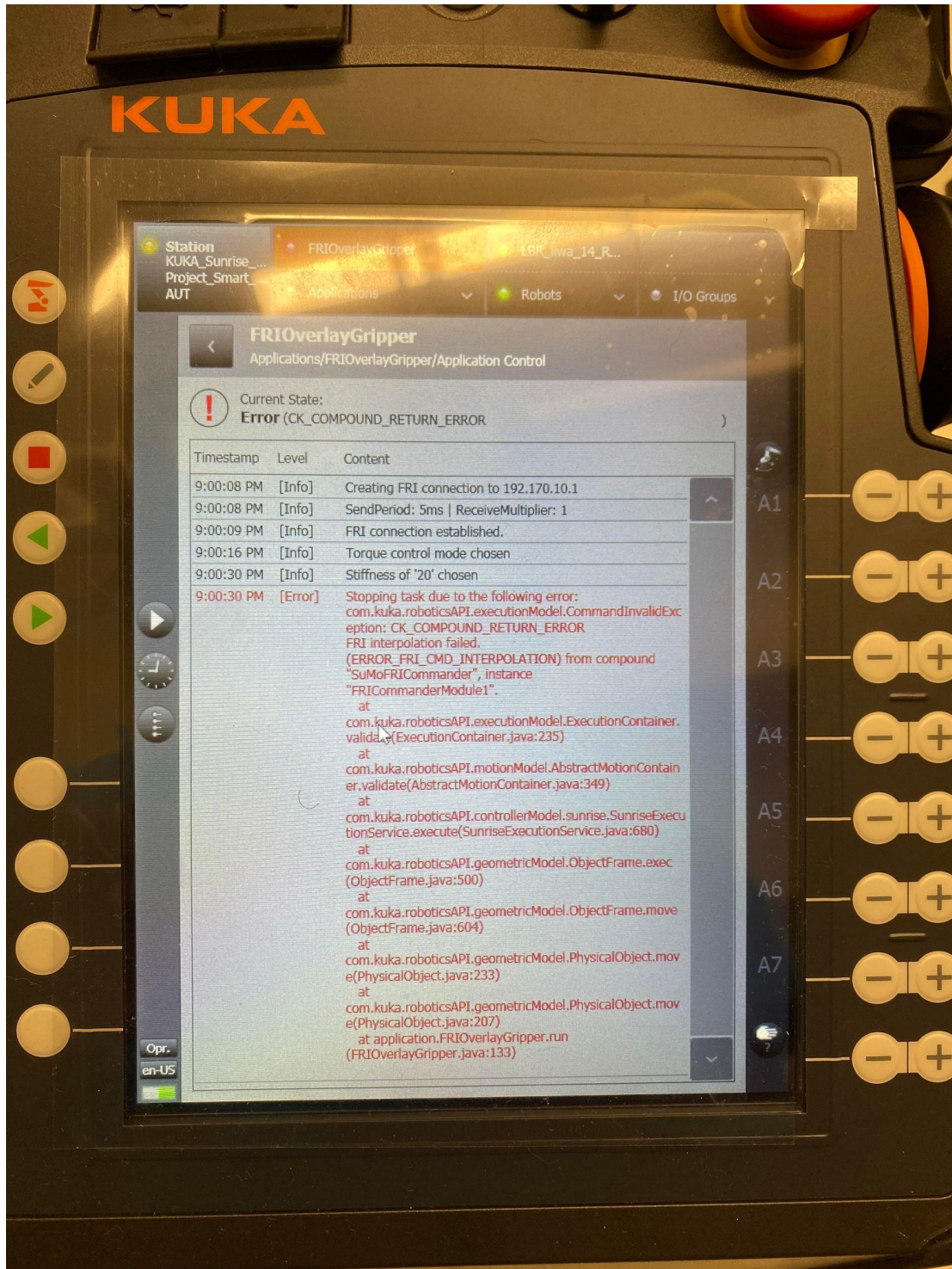


Figure A.1: Figure showing the error output when the lock error behavior happens

Robot locked in error state

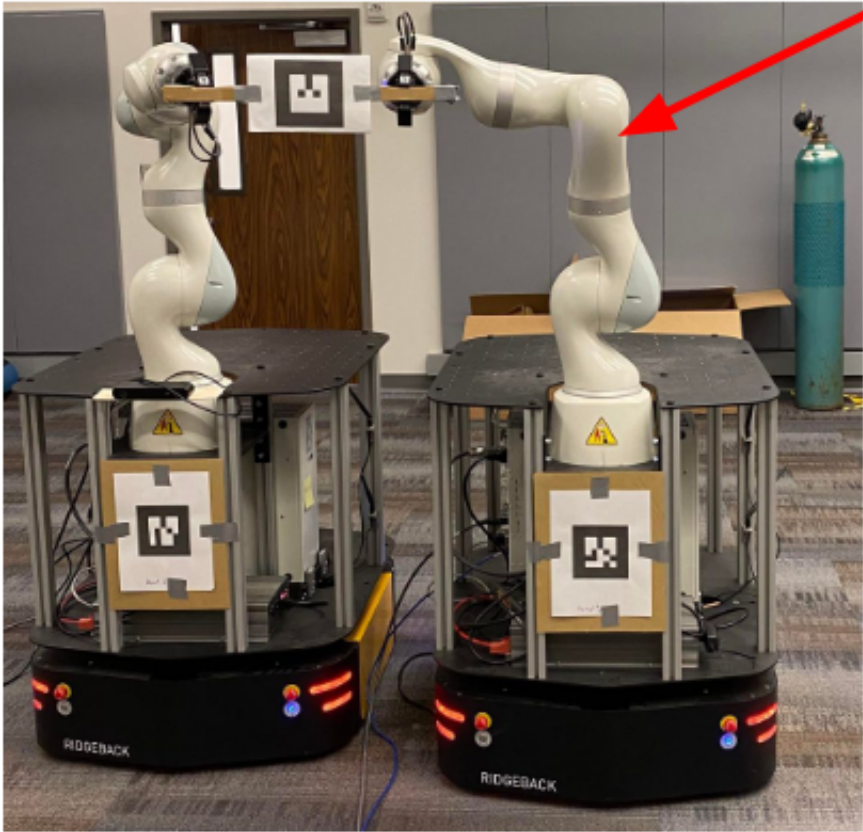


Figure A.2: Figure showing configuration of robot when the lock error behavior happens

**APPENDIX B**  
**SEMI-PARAMETRIC APPROACHES TO LEARNING INVERSE DYNAMICS**  
**CONTROL**

Earlier in my PhD, I worked on a semi-parametric approach for controlling robots. More specifically, I applied Gaussian Processes (GPs) [93, 94, 95] as a scheme for learning the inverse dynamics model of a seven DOF custom built manipulator arm. This work later led to the publication [96], which involved a more complex system known as *Krang* [97], which is shown in the Figure B.1. I worked both in simulation, using Dynamic Animation and Robotics Toolkit (DART) [98], and on one of the arms of the *Krang* robot.

The model of a robot is required to perform inverse dynamics control [99] on the robot. Therefore, given a robot with unknown dynamic parameters, it is possible to learn the model through a combination of approaches. The authors of [100] developed a data-driven approach which combined a parametric model obtained by performing linear regression on the recorded data and a non-parametric model based on Gaussian Processes. This ensemble of a parametric and a non-parametric model results in what is known as the *semi-parametric* approach.

Consider the dynamic model of a robot which includes an unstructured error term.

$$T(q, \dot{q}, \ddot{q}) = M(q)\ddot{q} + C(q, \dot{q}) + G(q) + \epsilon(q, \dot{q}, \ddot{q}) \quad (\text{B.1})$$

This error term ( $\epsilon(q, \dot{q}, \ddot{q})$ ) is caused by unmodelled dynamics, such as friction, joint flexibility, etc. [100]. According to [99], the *Rigid Body Dynamics* (RBD) model of a manipulator is known to be linear in the parameters  $\beta$  [100]. Therefore, the structured component

of Equation B.1 can be reformulated as linear function of the dynamic parameters [100].

$$T(q, \dot{q}, \ddot{q}) = \underbrace{\Phi((q, \dot{q}, \ddot{q}))\beta}_{\text{parametric}} + \underbrace{\epsilon(q, \dot{q}, \ddot{q})}_{\text{non-parametric}} \quad (\text{B.2})$$

## B.1 Gaussian Processes

A Gaussian Process model for the prediction of the joint torques in Equation B.2 can be defined by

$$T(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \quad (\text{B.3})$$

where  $\mathbf{x} = [q, \dot{q}, \ddot{q}]^\top$  is the input,  $m(\mathbf{x})$  is the mean function, and  $k(\mathbf{x}, \mathbf{x}')$  is the covariance function of the Gaussian Process [100]. When there is no prior knowledge,  $m(\mathbf{x}) = 0$ . However, the parametric model can be applied as the mean of the GP. Therefore, when  $m(\mathbf{x}) = \Phi(\mathbf{x})\beta$  the following equations are considered equivalent [100]

$$T(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \quad (\text{B.4a})$$

$$T(\mathbf{x}) = \Phi(\mathbf{x})\beta + \mathcal{GP}(\mathbf{0}, k(\mathbf{x}, \mathbf{x}')) \quad (\text{B.4b})$$

Gaussian processes provide uncertainty estimates of the model prediction, making them quite suitable for a semi-parametric approach where a part of model is comprised of a parametric component. Therefore, the final output of the model could be a weighted output of the Gaussian Process prediction. The weights could be a function of the uncertainty estimate. In other words, when the uncertainty in the Gaussian Process model prediction is low, the predicted mean can be weighted more heavily and vice-versa when the prediction uncertainty is higher. Thus, the contribution of the non-parametric component is limited by how certain the prediction is.

Finally, the data collection process involved generating excited trajectories based on

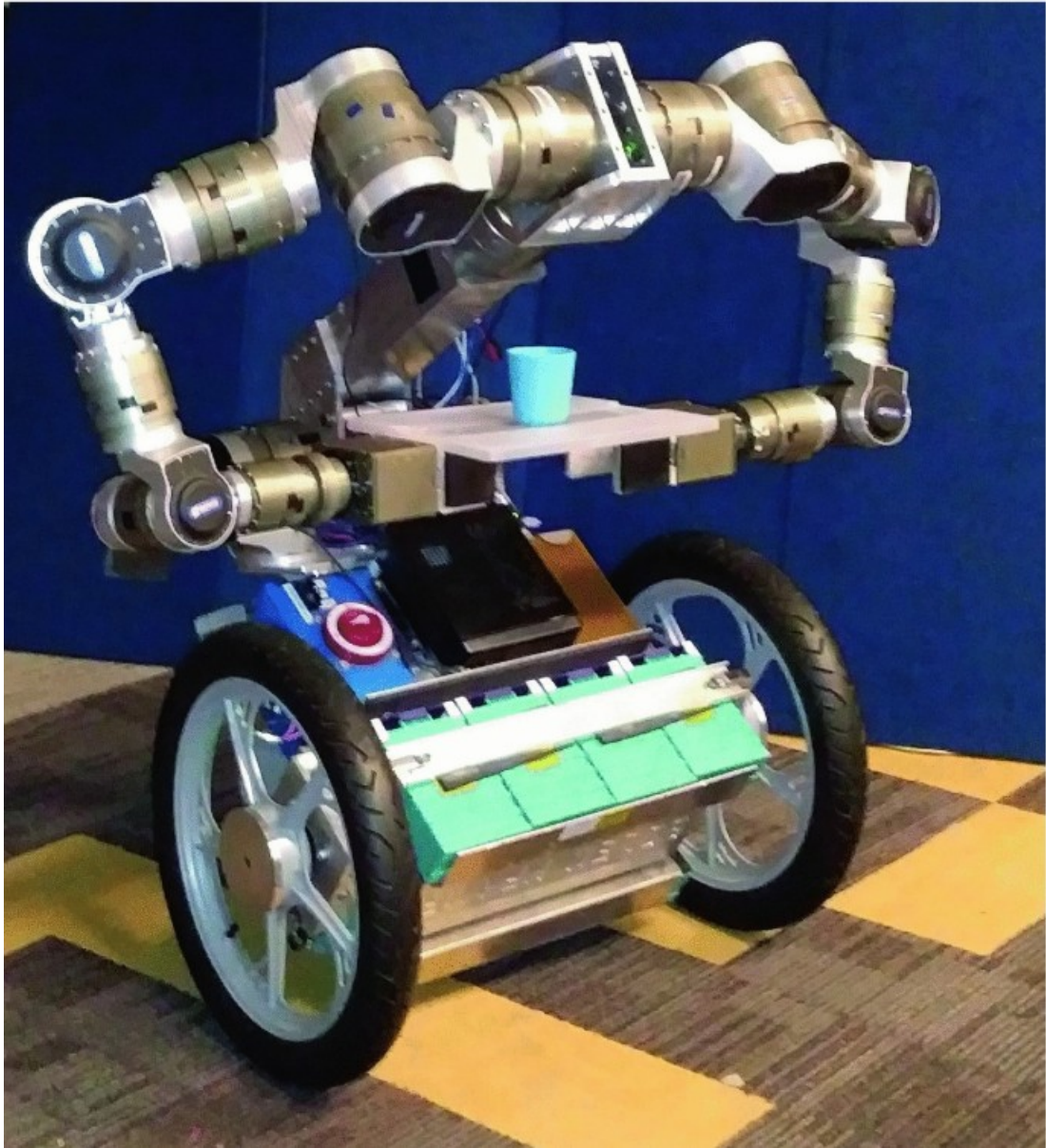


Figure B.1: Golem Krang [96]

the work by [101], to explore the state-space of the robot. Other works provide different methods of generating excited trajectories, including the works by [102, 103, 104]. In conclusion, the semi-parametric approach combines the benefits of a parametric model and a non-parametric component, which might also include prior knowledge of the system, to augment and improve the overall accuracy of the constructed model.

**APPENDIX C**

**AN ADAPTIVE COOPERATIVE MANIPULATION CONTROL FRAMEWORK  
FOR MULTI-AGENT DISTURBANCE REJECTION**

While in Sweden, I collaborated with a colleague in the division of decision and control systems to develop an adaptive control approach for compensating for disturbance in a cooperative manipulation setup. We worked primarily in simulation, using the *Simulink* framework in MATLAB. This work [105] is currently under review for publication at the IEEE Conference on Decision and Control (CDC). The rest of this chapter presents the introductory section of our paper [105], where we reviewed similar works and highlighted the contribution of our paper.

Adaptive manipulation control dates back to [106, 107, 108], and has been used to perform single arm manipulation with limited knowledge of the environment, as in [109]. However, the application of adaptive control to cooperative manipulation is still limited [110]. Several works in this field have proposed a high-level controller that distributes contributions among agents [111, 112], while others have accounted for the deviation from their assigned contribution [3, 113].

Alternatively, different *leader-follower* approaches have been proposed in [114, 115, 116] to circumvent the need for communication between agents. However, these options do not account for abnormal behaviors by the cooperating partners; this is further exacerbated when the aberrant robot is the leader, propagating the undesired behavior to the rest of the agents. Similarly, distributed formulations have been employed for cooperative manipulation, including [117], where a distributed approach without adaptive control was applied, and [118], in which an unknown payload was manipulated with an optimization of the decoupled dynamics of both subsystems. Other relevant approaches include using distributed Bayesian learning to infer the load dynamics and internal manipulation coupling

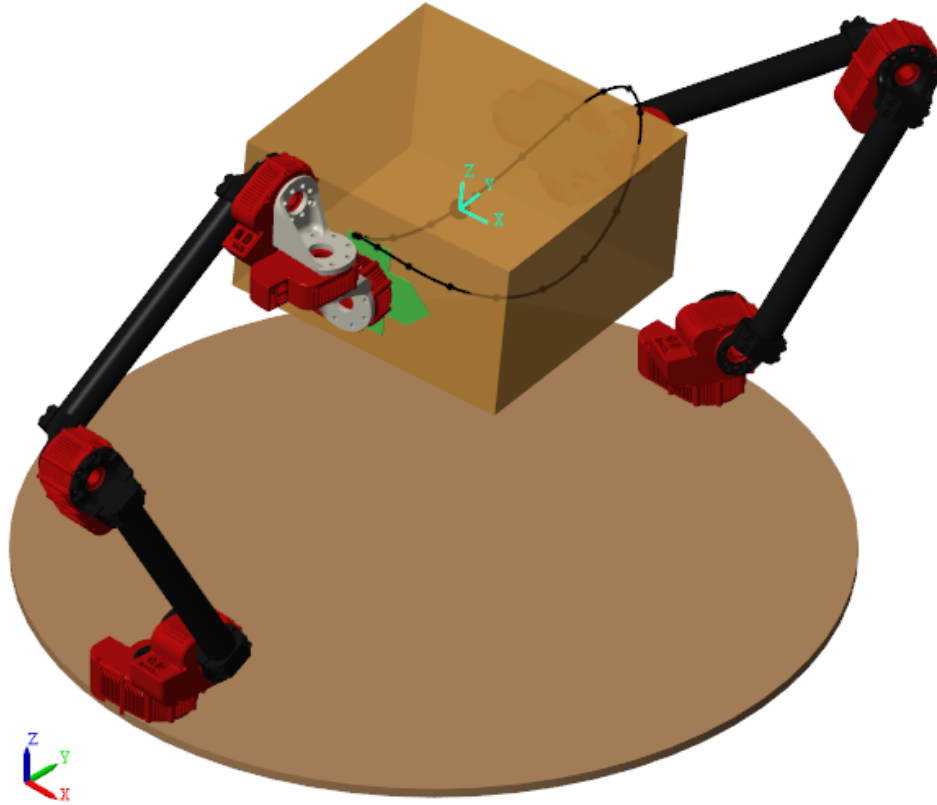


Figure C.1: Cooperative manipulation test bench system with two robot manipulators [105] [119]; employing formation control for the cooperative manipulation task, such as in [120] and [121], which led to a distributed controller that requires only local measurements, and a solution that requires a model-based add-on to compensate for the internal disturbances, respectfully; finally, using decentralized sliding mode control for cooperative load transport, as in [122];

Returning to adaptive control, we must highlight that its application on multi-agent manipulation systems is generally done either in the joint space [108, 123] or in the task space [124]; but not both. In contrast, we present a control framework that not only applies adaptive control in both the joint space and task space, but also improves the robustness of the solution by accounting for an extra coupling criterion for the manipulators (see Fig. Figure C.1). In particular, we opt for a backstepping approach [125, 126] to simplify the controller design and introduce the aforementioned coupling criterion directly into the formulation. The main contributions of this paper can then be summarized as:

- C1.** An adaptive backstepping framework that mitigates both joint- and task-space disturbances, considering the full dynamics of the load and the robot manipulators.
- C2.** The analysis of two applications of this framework: one tracking the end-effector positions in concordance with the load trajectories, and the other minimizing the joint speeds as a load-manipulator coupling criterion.

The novelties of the aforementioned contributions compared with relevant works are as follows. The work of [63] also proposed an adaptive approach for the collaborative manipulation task; but in that case, the adaptation is applied to the weight of the payload. In contrast, our framework can be distributed and only assumes prior knowledge of the inertial parameters of the object and its desired trajectory. In [62], a coupled dynamic formulation is used to obtain an adaptive controller to compensate for disturbances in the joint and task spaces. However, this approach tracks only the load pose, while we propose adding an extra coupling criterion for robustness. The approach in [127], in turn, also includes adaptation to uncertainties in the grasping setting and is formulated for any number of heterogeneous agents. However, this formulation involves calculating joint-space errors, despite their inherent limitations. Moreover, the optimization of the load-manipulator interaction is not fully explicit. Finally, in [128], the compensation for disturbances in the task and joint spaces is considered, but only for point masses with elastic grasping within a 1-D task-space and without theoretical guarantees.

## REFERENCES

- [1] I. D. Walker, R. A. Freeman, and S. I. Marcus, "Analysis of motion and internal loading of objects grasped by multiple cooperating manipulators," *The International journal of robotics research*, vol. 10, no. 4, pp. 396–409, 1991.
- [2] K.-S. Chang, R. Holmberg, and O. Khatib, "The augmented object model: Cooperative manipulation and parallel mechanism dynamics," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, IEEE, vol. 1, 2000, pp. 470–475.
- [3] S. Erhart and S. Hirche, "Internal force analysis and load distribution for cooperative multi-robot manipulation," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1238–1243, 2015.
- [4] N. Michael, J. Fink, and V. Kumar, "Cooperative manipulation and transportation with aerial robots," *Autonomous Robots*, vol. 30, no. 1, pp. 73–86, 2011.
- [5] C. K. Verginis, D. Zelazo, and D. V. Dimarogonas, "Cooperative manipulation via internal force regulation: A rigidity theory perspective," *arXiv preprint arXiv:1911.01297*, 2019.
- [6] N. Inaba and M. Oda, "Autonomous satellite capture by a space robot: World first on-orbit experiment on a japanese robot satellite ets-vii," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, IEEE, vol. 2, 2000, pp. 1169–1174.
- [7] A. Freddi, S. Longhi, A. Monteriù, and D. Ortenzi, "Redundancy analysis of cooperative dual-arm manipulators," *International Journal of Advanced Robotic Systems*, vol. 13, no. 5, p. 1 729 881 416 657 754, 2016.
- [8] H. Lee, H. Kim, and H. J. Kim, "Planning and control for collision-free cooperative aerial transportation," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 1, pp. 189–201, 2016.
- [9] A. De Luca and L. Ferrajoli, "Exploiting robot redundancy in collision detection and reaction," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2008, pp. 3299–3305.
- [10] H.-C. Lin, J. Smith, K. K. Babarahmati, N. Dehio, and M. Mistry, "A projected inverse dynamics approach for multi-arm cartesian impedance control," in *2018*

- IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 1–5.
- [11] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer, 2016.
- [12] N. Hogan, “Impedance control: An approach to manipulation,” in *1984 American control conference*, IEEE, 1984, pp. 304–313.
- [13] R. J. Anderson and M. W. Spong, “Hybrid impedance control of robotic manipulators,” *IEEE Journal on Robotics and Automation*, vol. 4, no. 5, pp. 549–556, 1988.
- [14] S. Erhart, D. Sieber, and S. Hirche, “An impedance-based control architecture for multi-robot cooperative dual-arm mobile manipulation,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2013, pp. 315–322.
- [15] D. Heck, D. Kostić, A. Denasi, and H. Nijmeijer, “Internal and external force-based impedance control for cooperative manipulation,” in *2013 European Control Conference (ECC)*, IEEE, 2013, pp. 2299–2304.
- [16] A. Tsiamis, C. K. Verginis, C. P. Bechlioulis, and K. J. Kyriakopoulos, “Cooperative manipulation exploiting only implicit communication,” in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2015, pp. 864–869.
- [17] C. Alford and S. Belyeu, “Coordinated control of two robot arms,” in *Proc. of the IEEE Int. Conf. on robotics and automation*, vol. 1, 1984, pp. 468–473.
- [18] H. Shen, Y.-J. Pan, and G. Bauer, “Manipulability-based load allocation and kinematic decoupling in cooperative manipulations,” in *2019 IEEE 28th International Symposium on Industrial Electronics (ISIE)*, IEEE, 2019, pp. 1168–1173.
- [19] Y. F. Zheng and J. Luh, “Optimal load distribution for two industrial robots handling a single object,” in *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, IEEE, 1988, pp. 344–349.
- [20] J. Spletzer, A. K. Das, R. Fierro, C. J. Taylor, V. Kumar, and J. P. Ostrowski, “Cooperative localization and control for multi-robot manipulation,” in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, IEEE, vol. 2, 2001, pp. 631–636.
- [21] P. Song and V. Kumar, “A potential field based approach to multi-robot manipulation,” in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, IEEE, vol. 2, 2002, pp. 1217–1222.

- [22] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, “Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics,” *arXiv preprint arXiv:1703.09312*, 2017.
- [23] S. Ekvall and D. Kragic, “Interactive grasp learning based on human demonstration,” in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA’04. 2004*, IEEE, vol. 4, 2004, pp. 3519–3524.
- [24] D. Williams and O. Khatib, “The virtual linkage: A model for internal forces in multi-grasp manipulation,” in *[1993] Proceedings IEEE International Conference on Robotics and Automation*, IEEE, 1993, pp. 1025–1030.
- [25] J. Nakanishi, R. Cory, M. Mistry, J. Peters, and S. Schaal, “Operational space control: A theoretical and empirical comparison,” *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 737–757, 2008.
- [26] A. De Luca and R. Mattone, “Sensorless robot collision detection and hybrid force/motion control,” in *Proceedings of the 2005 IEEE international conference on robotics and automation*, IEEE, 2005, pp. 999–1004.
- [27] A. De Luca, A. Albu-Schaffer, S. Haddadin, and G. Hirzinger, “Collision detection and safe reaction with the dlr-iii lightweight manipulator arm,” in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2006, pp. 1623–1630.
- [28] S. Haddadin, A. De Luca, and A. Albu-Schäffer, “Robot collisions: A survey on detection, isolation, and identification,” *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1292–1312, 2017.
- [29] A. Z. Bais, S. Erhart, L. Zaccarian, and S. Hirche, “Dynamic load distribution in cooperative manipulation tasks,” in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2015, pp. 2380–2385.
- [30] P. Hsu, “Coordinated control of multiple manipulator systems,” *IEEE Transactions on Robotics and Automation*, vol. 9, no. 4, pp. 400–410, 1993.
- [31] V. Aladele and S. Hutchinson, “Collision reaction through internal stress loading in cooperative manipulation,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 9102–9107.
- [32] R. Murray, Z. Li, and S. Sastry, *A mathematical introduction to robotic manipulation*. Boca Raton, FL: CRC press, 1994.
- [33] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control*. John Wiley & Sons, 2020.

- [34] K. M. Lynch and F. C. Park, *Modern Robotics*. Cambridge University Press, 2017.
- [35] O. Khatib, “Inertial properties in robotic manipulation: An object-level framework,” *The international journal of robotics research*, vol. 14, no. 1, pp. 19–36, 1995.
- [36] V. Aladele and S. Hutchinson, “Impedance-based collision reaction strategy via internal stress loading in cooperative manipulation,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2021, pp. 5837–5843.
- [37] A. Dietrich and C. Ott, “Hierarchical impedance-based tracking control of kinematically redundant robots,” *IEEE Transactions on Robotics*, vol. 36, no. 1, pp. 204–221, 2019.
- [38] J. Szewczyk and P. Bidaud, “Coordinated manipulation under distributed impedance control,” in *Experimental Robotics VI*, Springer, 2000, pp. 121–130.
- [39] O. Khatib, K. Yokoi, O. Brock, K. Chang, and A. Casal, “Robots in human environments: Basic autonomous capabilities,” *The International Journal of Robotics Research*, vol. 18, no. 7, pp. 684–696, 1999.
- [40] N. Koenig and A. Howard, “Design and use paradigms for gazebo, an open-source multi-robot simulator,” in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, IEEE, vol. 3, 2004, pp. 2149–2154.
- [41] R. Bischoff, J. Kurth, G. Schreiber, R. Koepe, A. Albu-Schäffer, A. Beyer, O. Eiberger, S. Haddadin, A. Stemmer, G. Grunwald, *et al.*, “The kuka-dlr lightweight robot arm-a new reference platform for robotics research and manufacturing,” in *ISR 2010 (41st international symposium on robotics) and ROBOTIK 2010 (6th German conference on robotics)*, VDE, 2010, pp. 1–8.
- [42] G. Flandin, F. Chaumette, and E. Marchand, “Eye-in-hand/eye-to-hand cooperation for visual servoing,” in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, IEEE, vol. 3, 2000, pp. 2741–2746.
- [43] L. Keselman, J. Iselin Woodfill, A. Grunnet-Jepsen, and A. Bhowmik, “Intel realsense stereoscopic depth cameras,” in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2017, pp. 1–10.
- [44] M. Fiala, “Artag, a fiducial marker system using digital techniques,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, IEEE, vol. 2, 2005, pp. 590–596.

- [45] R. Holmberg and O. Khatib, “Development and control of a holonomic mobile robot for mobile manipulation tasks,” *The International Journal of Robotics Research*, vol. 19, no. 11, pp. 1066–1074, 2000.
- [46] K. H. Kowdiki, R. K. Barai, and S. Bhattacharya, “Leader-follower formation control using artificial potential functions: A kinematic approach,” in *IEEE-International Conference On Advances In Engineering, Science And Management (ICAESM-2012)*, IEEE, 2012, pp. 500–505.
- [47] S. Mastellone, D. M. Stipanović, C. R. Graunke, K. A. Intlekofer, and M. W. Spong, “Formation control and collision avoidance for multi-agent non-holonomic systems: Theory and experiments,” *The International Journal of Robotics Research*, vol. 27, no. 1, pp. 107–126, 2008.
- [48] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
- [49] J. Kober, J. A. Bagnell, and J. Peters, “Reinforcement learning in robotics: A survey,” *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.
- [50] P. Kormushev, S. Calinon, and D. G. Caldwell, “Reinforcement learning in robotics: Applications and real-world challenges,” *Robotics*, vol. 2, no. 3, pp. 122–148, 2013.
- [51] V. Aladele, A. Longhini, A. Reichlin, H. Yin, C. Pek, and D. Kragic, “Compensating for errors in cooperative manipulation: A decentralized approach via residual reinforcement learning,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2022 (under review).
- [52] L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R. H. Campbell, K. Czechowski, D. Erhan, C. Finn, P. Kozakowski, S. Levine, *et al.*, “Model-based reinforcement learning for atari,” *arXiv preprint arXiv:1903.00374*, 2019.
- [53] T. M. Moerland, J. Broekens, and C. M. Jonker, “Model-based reinforcement learning: A survey,” *arXiv preprint arXiv:2006.16712*, 2020.
- [54] T. Degris, P. M. Pilarski, and R. S. Sutton, “Model-free reinforcement learning with continuous action in practice,” in *2012 American Control Conference (ACC)*, IEEE, 2012, pp. 2177–2182.
- [55] J. Garcia and F. Fernández, “A comprehensive survey on safe reinforcement learning,” *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

- [56] N. Fulton and A. Platzer, “Safe reinforcement learning via formal methods: Toward safe control through proof and learning,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.
- [57] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh, “A Lyapunov-based approach to safe reinforcement learning,” *Advances in neural information processing systems*, vol. 31, 2018.
- [58] T. J. Perkins and A. G. Barto, “Lyapunov design for safe reinforcement learning,” *Journal of Machine Learning Research*, vol. 3, no. Dec, pp. 803–832, 2002.
- [59] H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin, “Exploring strategies for training deep neural networks.,” *Journal of machine learning research*, vol. 10, no. 1, 2009.
- [60] C. Szegedy, A. Toshev, and D. Erhan, “Deep neural networks for object detection,” *Advances in neural information processing systems*, vol. 26, 2013.
- [61] Z. Wang and M. Schwager, “Multi-robot manipulation with no communication using only local measurements,” in *Proc. of the IEEE Conf. on Decision and Control*, 2015, pp. 380–385.
- [62] C. K. Verginis, M. Mastellaro, and D. V. Dimarogonas, “Robust cooperative manipulation without force/torque measurements: Control design and experiments,” *IEEE Transactions on Control Systems Technology*, vol. 28, no. 3, pp. 713–729, 2019.
- [63] P. Culbertson, J.-J. Slotine, and M. Schwager, “Decentralized adaptive control for collaborative manipulation of rigid bodies,” *IEEE Transactions on Robotics*, 2021.
- [64] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, “Residual reinforcement learning for robot control,” in *Proc. of the Int. Conf. on Robotics and Automation*, 2019, pp. 6023–6029.
- [65] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling, “Residual policy learning,” *arXiv preprint arXiv:1812.06298*, 2018.
- [66] A. Ranjbar, N. A. Vien, H. Ziesche, J. Boedecker, and G. Neumann, “Residual feedback learning for contact-rich manipulation tasks with uncertainty,” *arXiv preprint arXiv:2106.04306*, 2021.
- [67] G. Schoettler, A. Nair, J. Luo, S. Bahl, J. A. Ojea, E. Solowjow, and S. Levine, “Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 5548–5555.

- [68] L.-J. Lin, *Reinforcement learning for robots using neural networks*. Carnegie Mellon University, 1992.
- [69] S. S. Mousavi, M. Schukat, and E. Howley, “Deep reinforcement learning: An overview,” in *Proceedings of SAI Intelligent Systems Conference*, Springer, 2016, pp. 426–440.
- [70] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “Deep reinforcement learning: A brief survey,” *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.
- [71] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, J. Pineau, *et al.*, “An introduction to deep reinforcement learning,” *Foundations and Trends® in Machine Learning*, vol. 11, no. 3-4, pp. 219–354, 2018.
- [72] R. Martin-Martin, M. A. Lee, R. Gardner, S. Savarese, J. Bohg, and A. Garg, “Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2019, pp. 1010–1017.
- [73] R. Han, S. Chen, and Q. Hao, “Cooperative multi-robot navigation in dynamic environment with deep reinforcement learning,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 448–454.
- [74] K. Kamali, I. A. Bonev, and C. Desrosiers, “Real-time motion planning for robotic teleoperation using dynamic-goal deep reinforcement learning,” in *2020 17th Conference on Computer and Robot Vision (CRV)*, IEEE, 2020, pp. 182–189.
- [75] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, *Stable baselines*, <https://github.com/hill-a/stable-baselines>, 2018.
- [76] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.
- [77] G. C. Lopes, M. Ferreira, A. da Silva Simoes, and E. L. Colombini, “Intelligent control of a quadrotor with proximal policy optimization reinforcement learning,” in *2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE)*, IEEE, 2018, pp. 503–508.
- [78] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, *et al.*, “Soft actor-critic algorithms and applications,” *arXiv preprint arXiv:1812.05905*, 2018.

- [79] E. Coumans and Y. Bai, *Pybullet, a python module for physics simulation for games, robotics and machine learning*, <http://pybullet.org>, 2016–2021.
- [80] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “Openai gym,” *arXiv preprint arXiv:1606.01540*, 2016.
- [81] K. Clary, E. Tosch, J. Foley, and D. Jensen, “Let’s play again: Variability of deep reinforcement learning agents in atari environments,” *arXiv preprint arXiv:1904.06312*, 2019.
- [82] I. Osband, Y. Doron, M. Hessel, J. Aslanides, E. Sezener, A. Saraiva, K. McKinney, T. Lattimore, C. Szepesvari, S. Singh, *et al.*, “Behaviour suite for reinforcement learning,” *arXiv preprint arXiv:1908.03568*, 2019.
- [83] C. K. Verginis and D. V. Dimarogonas, “Energy-optimal cooperative manipulation via provable internal-force regulation,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 9859–9865.
- [84] P. Malhotra, L. Vig, G. Shroff, P. Agarwal, *et al.*, “Long short term memory networks for anomaly detection in time series,” in *Proceedings*, vol. 89, 2015, pp. 89–94.
- [85] A. Yamashita, T. Arai, J. Ota, and H. Asama, “Motion planning of multiple mobile robots for cooperative manipulation and transportation,” *IEEE Transactions on Robotics and Automation*, vol. 19, no. 2, pp. 223–237, 2003.
- [86] F. Sun, Y. Chen, Y. Wu, L. Li, and X. Ren, “Motion planning and cooperative manipulation for mobile robots with dual arms,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2022.
- [87] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, *et al.*, “Ros: An open-source robot operating system,” in *ICRA workshop on open source software*, Kobe, Japan, vol. 3, 2009, p. 5.
- [88] A. Tiderko, F. Hoeller, and T. Röhling, “The ros multimaster extension for simplified deployment of multi-robot systems,” in *Robot operating system (ROS)*, Springer, 2016, pp. 629–650.
- [89] G. Schreiber, A. Stemmer, and R. Bischoff, “The fast research interface for the kuka lightweight robot,” in *IEEE workshop on innovative robot control architectures for demanding (Research) applications how to modify and enhance commercial controllers (ICRA 2010)*, Citeseer, 2010, pp. 15–21.

- [90] Y.-C. Wang and K.-J. Lin, “Enhancing the real-time capability of the linux kernel,” in *Proceedings Fifth International Conference on Real-Time Computing Systems and Applications (Cat. No. 98EX236)*, IEEE, 1998, pp. 11–20.
- [91] K. Chatzilygeroudis, B. Fichera, and A. Billard, *Iiwa\_ros: A ros stack for kuka’s iiwa robots using the fast research interface*, 2019.
- [92] R. Tedrake and the Drake Development Team, *Drake: Model-based design and verification for robotics*, 2019.
- [93] C. E. Rasmussen, “Gaussian processes in machine learning,” in *Summer school on machine learning*, Springer, 2003, pp. 63–71.
- [94] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*, 3. MIT press Cambridge, MA, 2006, vol. 2.
- [95] C. E. Rasmussen and H. Nickisch, “Gaussian processes for machine learning (gpml) toolbox,” *The Journal of Machine Learning Research*, vol. 11, pp. 3011–3015, 2010.
- [96] M. Murtaza, V. Aladele, E. A. Theodorou, S. Hutchinson, and B. Boots, “Semi-parametric approaches to learning in model-based hierarchical control of complex systems,” in *Proceedings of the 2018 International Symposium on Experimental Robotics*, Springer Nature, vol. 11, 2020, p. 387.
- [97] M. Stilman, J. Olson, and W. Gloss, “Golem krang: Dynamically stable humanoid robot for mobile manipulation,” in *2010 IEEE International Conference on Robotics and Automation*, IEEE, 2010, pp. 3304–3309.
- [98] J. Lee, M. X. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. S. Srinivasa, M. Stilman, and C. K. Liu, “Dart: Dynamic animation and robotics toolkit,” *Journal of Open Source Software*, vol. 3, no. 22, p. 500, 2018.
- [99] M. W. Spong, S. Hutchinson, M. Vidyasagar, *et al.*, *Robot modeling and control*. Wiley New York, 2006, vol. 3.
- [100] D. Nguyen-Tuong and J. Peters, “Using model knowledge for learning inverse dynamics,” in *2010 IEEE international conference on robotics and automation*, IEEE, 2010, pp. 2677–2682.
- [101] J. Swevers, C. Ganseman, D. B. Tukel, J. De Schutter, and H. Van Brussel, “Optimal robot excitation and identification,” *IEEE transactions on robotics and automation*, vol. 13, no. 5, pp. 730–740, 1997.

- [102] C. Presse and M. Gautier, “New criteria of exciting trajectories for robot identification,” in *[1993] Proceedings IEEE International Conference on Robotics and Automation*, IEEE, 1993, pp. 907–912.
- [103] M. Gautier and W. Khalil, “Exciting trajectories for the identification of base inertial parameters of robots,” *The International journal of robotics research*, vol. 11, no. 4, pp. 362–375, 1992.
- [104] B. Armstrong, “On finding exciting trajectories for identification experiments involving systems with nonlinear dynamics,” *The International journal of robotics research*, vol. 8, no. 6, pp. 28–48, 1989.
- [105] V. Aladele, C. de Cos, D. Dimarogonas, and S. Hutchinson, “An adaptive cooperative manipulation control framework for multi-agent disturbance rejection,” in *IEEE Conference on Decision and Control (CDC)*, IEEE, 2022.
- [106] K. J. Åström, “Theory and applications of adaptive control—a survey,” *Automatica*, vol. 19, no. 5, pp. 471–486, 1983.
- [107] T. Hsia, “Adaptive control of robot manipulators—a review,” in *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, IEEE, vol. 3, 1986, pp. 183–189.
- [108] J.-J. E. Slotine and W. Li, “On the adaptive control of robot manipulators,” *The International Journal of Robotics Research*, vol. 6, no. 3, pp. 49–59, 1987.
- [109] C. R. de Cos, J. Á. Acosta, and A. Ollero, “Adaptive integral inverse kinematics control for lightweight compliant manipulators,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 282–289, Apr. 2020.
- [110] A. Billard and D. Kragic, “Trends and challenges in robot manipulation,” *Science*, vol. 364, no. 6446, 2019.
- [111] H.-C. Lin, J. Smith, K. K. Babarhamati, N. Dehio, and M. Mistry, “A Projected Inverse Dynamics Approach for Multi-Arm Cartesian Impedance Control,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 5421–5428.
- [112] N. Dehio, J. Smith, D. L. Wigand, G. Xin, H.-C. Lin, J. J. Steil, and M. Mistry, “Modeling and control of multi-arm and multi-leg robots: Compensating for object dynamics during grasping,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 294–301.

- [113] T. Rugthum and G. Tao, “An adaptive actuator failure compensation scheme for a cooperative manipulator system with parameter uncertainties,” in *2015 54th IEEE Conference on Decision and Control (CDC)*, 2015, pp. 6282–6287.
- [114] Z. Wang and M. Schwager, “Force-amplifying n-robot transport system (force-ants) for cooperative planar manipulation without communication,” *The Int. Journal of Robotics Research*, vol. 35, no. 13, pp. 1564–1586, 2016.
- [115] G. Chen and F. L. Lewis, “Distributed adaptive controller design for the unknown networked lagrangian systems,” in *49th IEEE Conference on Decision and Control (CDC)*, 2010, pp. 6698–6703.
- [116] M. N. Mahyuddin, G. Herrmann, and F. L. Lewis, “Distributed adaptive leader-following control for multi-agent multi-degree manipulators with finite-time guarantees,” in *52nd IEEE Conference on Decision and Control*, 2013, pp. 1496–1501.
- [117] A. Marino, G. Muscio, and F. Pierri, “Distributed cooperative object parameter estimation and manipulation without explicit communication,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2017, pp. 2110–2116.
- [118] T. Miyano, J. Romberg, and M. Egerstedt, “Distributed force/position optimization dynamics for cooperative unknown payload manipulation,” in *2020 59th IEEE Conference on Decision and Control (CDC)*, 2020, pp. 5366–5373.
- [119] P. B. G. Dohmann, A. Lederer, M. Dißmond, and S. Hirche, “Distributed bayesian online learning for cooperative manipulation,” in *2021 60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 2888–2895.
- [120] K. Sakurama, “Formation control of mechanical multi-agent systems under relative measurements and its application to robotic manipulators,” in *2021 60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 6445–6450.
- [121] H. Wu, B. Jayawardhana, H. G. De Marina, and D. Xu, “Distributed formation control of manipulators’ end-effector with internal model-based disturbance rejection,” in *2021 60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 5568–5575.
- [122] H. Farivarnejad, S. Wilson, and S. Berman, “Decentralized sliding mode control for autonomous collective transport by multi-robot systems,” in *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016, pp. 1826–1833.
- [123] C. Yang, G. Ganesh, S. Haddadin, S. Parusel, A. Albu-Schaeffer, and E. Burdet, “Human-like adaptation of force and impedance in stable and unstable interactions,” *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 918–930, 2011.

- [124] Y. Ren, S. Sosnowski, and S. Hirche, “Fully distributed cooperation for networked uncertain mobile manipulators,” *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 984–1003, 2020.
- [125] P. V. Kokotovic, “The joy of feedback: Nonlinear and adaptive,” *IEEE Control Systems Magazine*, vol. 12, no. 3, pp. 7–17, 1992.
- [126] M. Krstic, P. V. Kokotovic, and I. Kanellakopoulos, *Nonlinear and adaptive control design*. John Wiley & Sons, Inc., 1995.
- [127] Yan, Lei and Stouraitis, Theodoros and Vijayakumar, Sethu, “Decentralized Ability-Aware Adaptive Control for Multi-Robot Collaborative Manipulation,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2311–2318, 2021.
- [128] A. Smith, C. Yang, H. Ma, P. Culverhouse, A. Cangelosi, and E. Burdet, “Biomimetic joint/task space hybrid adaptive control for bimanual robotic manipulation,” in *11th IEEE International Conference on Control & Automation (ICCA)*, IEEE, 2014, pp. 1013–1018.

## VITA

Victor Aladele currently lives in Atlanta, Georgia, where he is a PhD student in the Electrical and Computer Engineering (ECE) department at the Georgia Institute of Technology. While at Georgia Tech, Victor has taken the opportunity to intern at several companies, including his most recent experience as a research intern at Google Brain. Victor also had a five-month research stint at the Royal Institute of Technology (KTH) in Stockholm Sweden, where he worked in the Robotics, Perception and Learning (RPL) lab, under Professor Danica Kragic.

Prior to attending Georgia Tech, Victor obtained his bachelor's degree in Electrical Engineering from the New Jersey Institute of Technology (NJIT), Newark, New Jersey. While at NJIT, Victor was a summer research intern at the Massachusetts Institute of Technology (MIT), where he worked with Professor Daniela Rus and Robert MacCurdy (PhD).

Victor enjoys outdoor activities such as playing tennis, hiking, etc. Victor hopes to one day start a robotics company that develops robots like smartphones, so people around the world could create and upload different robotic applications to an "app store" (just like Google or Apple app store); these application can then be downloaded from the app store, by robot users or owners. The goal is to democratize the design of robotic applications beyond just robotics experts. Victor's dream is for each household to one day own a robot that can perform multiple tasks around the home, with these robot capabilities obtained by downloading "robot apps" that have been designed by various people from around the world.