AN ABSTRACT OF THE THESIS OF

Anjali Harish Asar for the degree of Master of Science in Robotics presented on December 15, 2021.

Title: A Structured Approach for In-Hand Manipulation

Abstract approved: _____

Cindy Grimm

In-hand manipulations consist of dexterous motions that come easy to humans but still pose a challenge to robotic systems. It is difficult to control finger motions in long complicated sequences due to high DOFs and intricate contact interactions. For such complex motions, in-hand manipulations have generally been broken into a hierarchy of low and high level control. In this case, high level control sequences low level controllers to perform motion primitives. The low level motion primitives tend to be task dependent and do not always uniformly sample from the manipulation space. This narrows its scope and makes it difficult to adapt to other in-hand manipulation tasks. Another technique that has widely proven to be promising is reinforcement learning (RL) based controllers. These controllers are able to perform a complex set of motions. However, applying RL directly to complicated in-hand manipulations limits what can be generalized to other tasks.

This thesis focuses on using a set of motion primitives that are task agnostic, sampled uniformly and symmetrically from the manipulation space to provide a structured approach for in-hand manipulation.

Specifically, we design two low level controllers (an inverse kinematics controller and a reinforcement learning controller) that can perform primitive in-hand manipulations. We further assess the ability of the reinforcement learning controller to adapt to other, unseen primitive manipulations. Finally, we show that structuring complex manipulations by staging low level controllers to learn from a uniform, symmetric, task agnostic set of motion primitives can adapt better to more tasks as compared to using RL for an end-end manipulation task.

©Copyright by Anjali Harish Asar
December 15, 2021
Creative Commons License

A Structured Approach for In-Hand Manipulation

by

Anjali Harish Asar

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented December 15, 2021
Commencement June 2022

Master of Science thesis of Anjali Harish Asar presented on December 15, 2021.

APPROVED:

Major Professor, representing Robotics

Robotics Chair of the Graduate School of College of Engineering

---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Anjali Harish Asar, Author

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my advisor, Dr. Cindy Grimm without whose expertise this thesis would not be possible. A special note of gratitude towards my committee members for sitting on this panel. I would also like to thank John Morrow for being a mentor and always giving great advice. I would also like to acknowledge Nigel Swenson, Stephanie Hughes, Josh Campbell and Keegan Nave for their contributions. Last but not the least, I would like to show appreciation towards my parents and my sister for being constant pillars of support and for always believing in me.

TABLE OF CONTENTS

Page

1 Introduction 1

2 Related Works 5
2.1 Hierarchical Control 5
2.2 Reinforcement Learning in Manipulation 6

3 Methodology 8
3.1 Parameterized Motion Primitives 11
3.2 Inverse Kinematics (IK) Controller 13
3.3 Reinforcement Learning (RL) Controller 16
3.3.1 State Space 17
3.3.2 Action Space 18
3.3.3 Rewards 19
3.3.4 Policy 20
3.4 Experimental Setup 21

4 Results 23
4.1 Manipulation Paths 23
4.1.1 IK Controller 23
4.1.2 RL Controller 26
4.2 RL Controller Performance 28

5 Discussion 31
5.1 Failures of the RL controller 31
5.2 Reward function fluctuations 32
5.3 Future Work 33

6 Conclusion 34

Bibliography 35

LIST OF FIGURES

Figure		Page
1.1	Asterisk Test motion directions	3
3.1	Motion directions from Asterisk Test used as motion primitives	9
3.2	Human puppeteering the object along the 'Right' direction with 2 link 2 fingered hand	10
3.3	Palm centric coordinate system	11
3.4	Physical set up and Simulation set up	12
3.5	Rough estimate of initial contacts	13
3.6	A visualization of expected contacts and actual contacts	15
3.7	Motion primitives for test data	16
3.8	Phases in an episode: Phase 1 places the object. Phase 2 Grasps the object. Phase 3 Manipulates the object.	21
4.1	Object pose paths of the IK controller. a) Averaged human data motions, b) Averaged IK controller motions, c) IK controller straight paths. directions	24
4.2	Failures of the IK Controller. (left) Failure of the controller with straight line paths, (right) Failure of the controller with human study data	25
4.3	Object pose paths of the RL controller: a) Averaged human data motions, b) Averaged IK controller motions, c) RL controller motions direction. Each direction has a single RL trial path	25
4.4	Single trials of motion direction of RL Controller along each test direction	26
4.5	Failures of the RL Controller along training motion primitive	27
4.6	Failures of the RL Controller along testing motion primitives	28
4.7	Reinforcement learning reward over episodes. All motion primitives are trained simultaneously.	29

LIST OF FIGURES (Continued)

Figure		Page
4.8	Critic Loss over 20k episodes.	29
4.9	Actor Loss over 20k episodes	30

LIST OF TABLES

Table Page

3.1 State Space Metrics 18

Chapter 1: Introduction

The ability to manipulate objects within the hand comes easy to humans. But the dexterity of such tasks still pose a challenge for robotic systems. Controllers designed for manipulation struggle to perform simple in-hand motions like moving a pen or rotating a grasped object. As the motions get longer and more complex, these controllers tend to fail due to the complicated contact interactions and high DOFs [6], [15].

A typical approach to in-hand manipulation is using reinforcement learning. This control method has given promising results due to its ability to learn complex, end-to-end motions. At the same time, since the controller is able to learn a specific, complicated task, it limits the ability to adapt to other manipulation tasks [15], [5]. For example, a learning based controller might be able to learn the complicated task of unlocking a door very well. This task might include grasping a key, navigating it to the keyhole and manipulating the key around to unlock the door. But this end-to-end controller would not be able to translate these motions to twirl a pen.

Another common approach is to divide the tasks into a hierarchy of low level and high level control. Where, at the lower level, there are a number of primitive controllers and at the higher level there tends to be a sequencer that selects from these controllers [8], [4]. For example, for a robot hand to twirl a pen there might

be certain low level controllers designed to grasp the pen, manoeuvre the fingers in a certain way, re-grasp the pen and slide the fingers along the surface. At the high level, a controller would sequence the low level motions to generate the twirling motion of the pen.

Breaking up a task into levels of control could potentially adapt to more tasks better, as compared to an end-to-end controller- The low level controllers can be re-sequenced to perform a similar task. Still, one of the drawbacks of this approach is that for every new task, different sets of low level controllers are used, making it task specific. In the example above, the low level breakdown is still too complex to support transferring the motions to a task such as unlocking a door. The entire manipulation space is not sampled uniformly. Instead, only the task specific manipulations form the basis of the low level controllers. This results in a limited ability to be robust and generalize to motions apart from the task at hand.

We can see how it becomes necessary to have a set of motion primitives that are not task dependent and uniformly parameterize the manipulation space. More importantly, it is also necessary to deconstruct complex manipulations into extremely simple motions to support a multitude of tasks. Small tasks such as moving an object within the hand span, along straight line trajectories or including slight rotations make up fundamental motions. Such motions can make for an ideal set of motion primitives as they form a part of most manipulations.

With such primitive motions, the manipulation space is small and contact interactions are less complicated. Thus providing building blocks for complex motions and making it easier to generalize to other tasks, objects and hand designs. It is

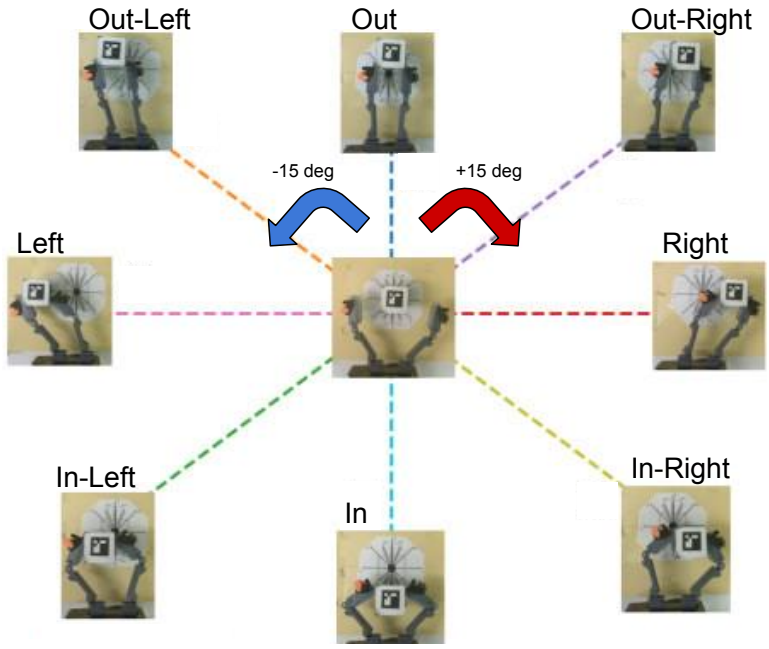

Figure 1.1: Asterisk Test motion directions

also easier to track where and why manipulations fail, giving us a better understanding of an appropriate breakdown of tasks.

The Asterisk Test [7], originally designed to benchmark manipulations, provides us with uniformly sampled and symmetric motions that are task independent. The Asterisk Test measures maneuverability along the central axis of the palm. The data samples 8 directions [out, out-right, right, in-right, in, in-left, left, out-left] with 2 twist types, as shown in Figure 1.1.

In this thesis, we use a structured framework consisting of motion primitives based off of the Asterisk Test, to build a reinforcement learning (RL) based controller. We assess the controllers ability to generalise to arbitrary motions not part of this dataset. This framework builds up two low-level controllers that can

perform primitive in-hand manipulations. We first build an inverse kinematics (IK) controller, which we then use as an expert for imitation learning to build a reinforcement learning (RL) controller.

In this scope, we refer to in-hand manipulations as pure finger motions (without allowing arm or palm movements) to reposition an acquired object while explicitly maintaining contact. For simplicity, we limit motions to a 2D plane. Although our framework allows for it to be just as easily extended to 3D.

Contribution: A structured learning framework that builds up an RL-based in-hand manipulation controller that is parameterized by desired motion directions. This learning framework is largely hand-design and object agnostic and the controller supports decomposing complex tasks into a set of generalized motion primitives.

Chapter 2: Related Works

We cover related works in two common approaches to in-hand manipulation: hierarchical control and reinforcement learning (RL) based control. In section 2.1, we cover the different ways control composition schemes have been applied, with focus on low level control. In section 2.2 we cover end-to-end RL based controllers along with their conjunction with hierarchical control.

2.1 Hierarchical Control

Control composition schemes have been applied in early works of grasping and manipulation where complex tasks are decomposed into subtasks as reviewed in [9]. Jefferson Coelho Jr et al. broadly categorized controllers designed based on composition as procedural control (object/ task-dependent control actions) and declarative control [1]. Declarative control describes a more abstract, object/ task/ environment independent control specification. In their work they design a grasp controller based on control decomposition, made up of two individual controllers, capable of solving a family of control tasks. In the scope of our work, we follow a declarative design of the controllers.

Decomposition of controllers for manipulation can further be split into a hierarchy of low and high level control. At the higher level we have control for

task planning and sequencing of manipulation phases. At the lower level, we have controllers designed for each manipulation phase [8]. Our work focuses on low level control. Previous works including [4] divide the problem into a hierarchy of sequencing and low-level primitives for in-hand manipulation. However, the low-level primitives are hand design specific. The low-level controls used here are traditional, torque based controllers that are sensitive to calibration and require hand coding. While this control works for certain manipulation, as the motions get more complex and the DOFs increase, the number of low-level traditional controllers required will also increase.

2.2 Reinforcement Learning in Manipulation

Over the decade, reinforcement learning (RL) has widely gained the acceptance of the manipulation community due to its non-reliance on analytic dynamics or kinematic models. One of the earliest demonstrations of using reinforcement learning to learn an in-hand manipulation skill was done in 2015 [12]. Recent work such as [14] uses a mix of reinforcement learning and hierarchical control (in the form of multi-step learning) to solve manipulation tasks such as object stacking. Both these works have proven the capabilities of using RL on end-end tasks. However, the multi-steps of [14] are task based. Neither of the works are directly able to extend the decomposed control functionality to other complex manipulations. While RL has been used at high level control and as a mix of low and high level control [2], little or no work has been done that applies the generalizability of RL to low-

level control in order to extend the abilities of a controller to adapt to a variety of complex tasks. Freek Stulp et al. come close and is most similar to our work wherein they combine RL with imitation learning and use dynamic motion primitives DMPs [3] to sequence robust manipulations[10]. However, these primitives are still task based and used with the sequencing, and are designed for end-end tasks.

Chapter 3: Methodology

Our goal is to build up a low-level reinforcement learning based controller using a structured framework and uniformly sampled, task agnostic motion primitives.

The motion primitives we chose are derived from the asterisk test as they are simple in-hand motions that most complex manipulations can be broken down to. In this thesis, we focus on only the 8 translations (no rotations) as our motion primitives. These motions are parameterized by the palm- In, Out, Left, Right. This is shown in Figure 3.1. However, our structured framework can be extended to include more primitive motions as well.

A single manipulation path is defined from the point when initial contact is established with the object until it has been moved as far and as straight as possible along the given direction. Releasing contact with the object during manipulation is not allowed. However, sliding contact along the object's edge is permitted.

The capability of a robot hand to manipulate an object can be limited by it kinematic ability and mechanical design. Even if the manipulation are simple straight line paths that are generally applicable to most in-hand motions. Trying to predict these limitations for a general purpose controller can be challenging. In this work, we are able to recognize these limitations with a dataset of baseline manipulations that are a part of the asterisk test. The asterisk test comes with a dataset of various robot hands manipulating an object along these primitives.

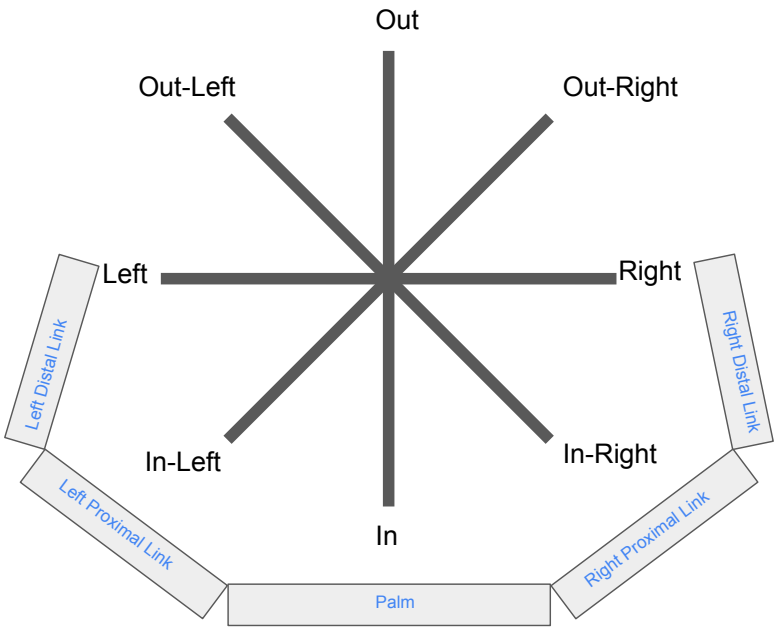

Figure 3.1: Motion directions from Asterisk Test used as motion primitives

These motions are human puppeteered and the dataset consists of the object poses along each manipulation direction. Figure 3.2 shows an example of a human puppeteering an object along a trajectory using a 2-link 2-fingered hand. We use this dataset to give desired poses to the IK solver to emulate the human trials in simulation.

The structured framework could be used along with the IK controller to build up motion sequences. Given our goal to build a general purpose controller, such a traditional controller is limited in its ability to expand to new motion primitives, new objects and new hand designs. It would require hand tuning of multiple low level controller at every stage which defeats the goal. Hence we only use the IK

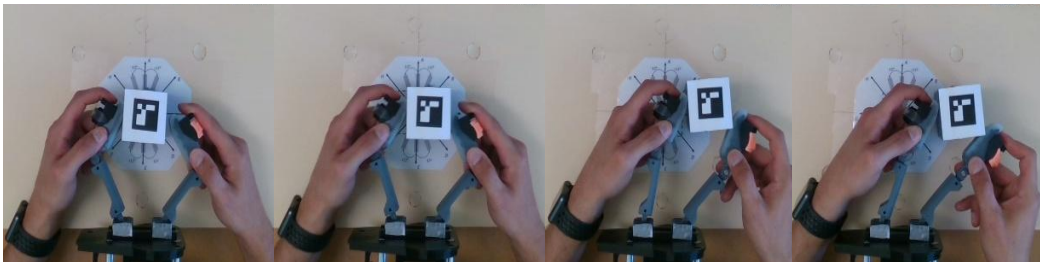

Figure 3.2: Human puppeteering the object along the 'Right' direction with 2 link 2 fingered hand

controller to bootstrap the learning of the RL controller. Being a learning based method, the RL controller is better suited to be a general purpose controller.

The RL controller is trained with the help of expert demonstrations from the IK controller to perform manipulations along the standard motion primitives described above (training dataset). To train the RL controller, the state space and rewards were carefully selected to best represent the agnostic nature of our framework.

To assess the ability of the RL controller to generalize, we test how well the controller performs along primitive directions that it has not trained on.

The breakup of the following sections is as given: Section 3.1 provides more insight on how we define our motion primitives and manipulation constraints. Section 3.2 and Section 3.3 give details on the construction of the IK controller and RL controller respectively. Section 3.4 outlines the experimental setup.

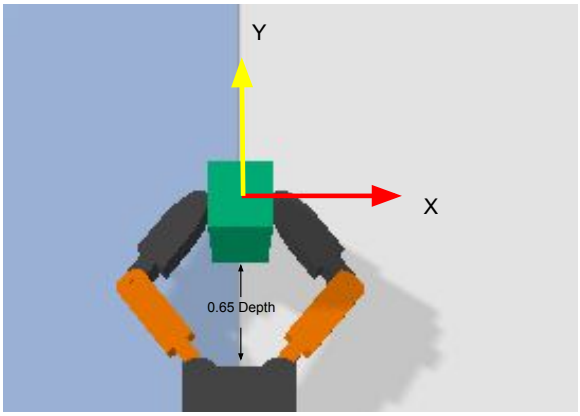

Figure 3.3: Palm centric coordinate system

3.1 Parameterized Motion Primitives

The motion primitives are defined by a palm-centric coordinate system. The origin is centred at the palm in the X-axis and 0.65 times the total finger length (from the palm) in the Y-axis. The object is placed at the origin at the start of every manipulation, as shown in Figure 3.3. Each motion primitive is described by the angle it makes with the X-axis. The object pose is represented as (x, y, θ). This allows us to sample from a large space of the hand span. Given this manipulation space, the asterisk test motions and the dataset it provides are an ideal fit for our primitives.

The object pose dataset of the asterisk test is derived from 5 trials each of 3 human subjects puppeteering the manipulations along the 8 trajectories i.e 5*3*8 trials. The human puppeteered dataset consists of object pose information along every trajectory.

Equation 3.1 describes how information is extracted from the object pose

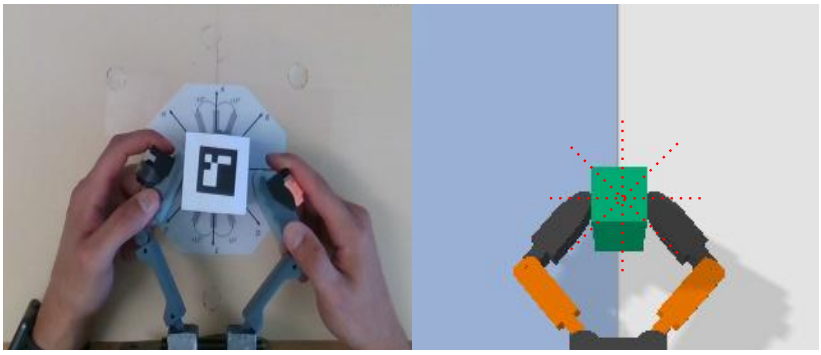

Figure 3.4: Physical set up and Simulation set up

dataset for a single manipulation episode. An episode starts once the object is placed at the origin and contact is established with all fingers of the hand. The number of object poses in a single episode may vary depending on the human trial and when contact was terminated.

$$(x, y, \theta)_n\text{: for } n \text{ in } [0, N] \tag{3.1}$$

where,

N is the total number of object poses along a manipulation direction and

n is the index of the current object pose.

The physical set up and the simulation hand and object are shown in Figure 3.4.

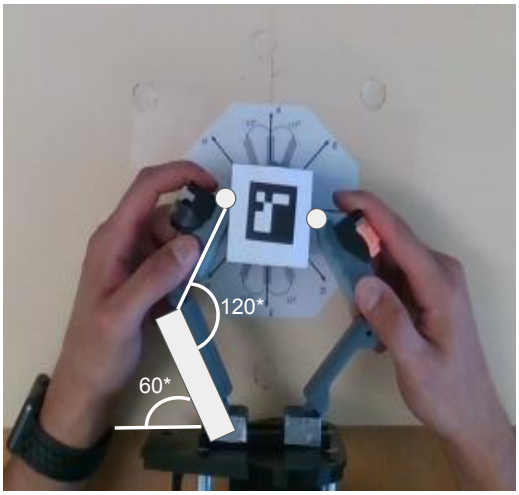

Figure 3.5: Rough estimate of initial contacts

3.2 Inverse Kinematics (IK) Controller

The aim of the IK controller is to replicate the object pose data of the asterisk test in simulation, so as to bootstrap the learning of the RL controller.

To solve the inverse kinematic equations to derive the joint angle solutions, we require contact information between the two fingers and the object. One of the key challenges of building this controller was the lack of contact information. As the dataset only provides object pose data and no contact data, it was not straight forward the estimate the contact points at each simulation step for the inverse kinematic calculations.

Joint angles at initial contact was extracted by rough estimation from the human puppeteering images as shown in Figure 3.5. Accounting for the differences in the simulated and physical hand, the joint angles were further adjusted to

establish the initial contact. We then attempt to maintain the initial contact points, while allowing for sliding.

In a simulation step n of an episode, $(x, y, \theta)_n$ are extracted from the object pose data. Using the frame of reference of the object, we calculate the expected contact points for the left and right fingers (i.e. initial contact points in object reference frame). The transformation matrix for the expected contact point is calculated as show in Equation 3.2

$$\begin{aligned} T_{cp_{exp}l} &= T_{obj_n} * T_{cp_{curr}l} \\ T_{cp_{exp}r} &= T_{obj_n} * T_{cp_{curr}r} \end{aligned} \tag{3.2}$$

where,

$T_{cp_{exp}l}$ is the transformation matrix of the expected contact between the object and left finger,

$T_{cp_{exp}r}$ is the transformation matrix of the expected contact between the object and right finger,

T_{obj_n} is the transformation matrix of the object pose extracted at simulation step n,

$T_{cp_{curr}l}$ is the transformation matrix of the current contact between the object and left finger and

$T_{cp_{curr}r}$ is the transformation matrix of the current contact between the object and right finger

Note: All values in this equation are in the object reference frame

The IK controller then finds suitable joint angles solution from its current

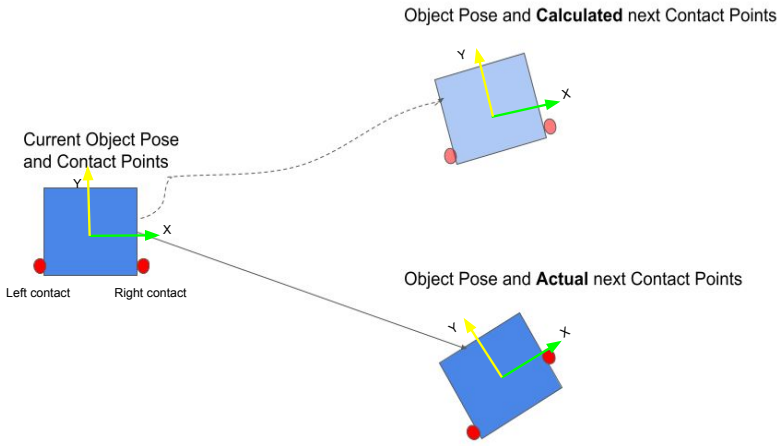

Figure 3.6: A visualization of expected contacts and actual contacts

contact points to the expected contact points. It is important to note that the the initial contact points are not assumed to always be maintained. The assumption is to *attempt* to maintain them. A descriptive visualization of this is shown in Figure 3.6. For the IK solve, we use the damped least squares method.

Along the way, if contact is lost we allow 20 simulation steps to try and re-establish the contact. We do this by moving the fingers towards the centre of the object. If contact is not established at this time, the manipulation is terminated.

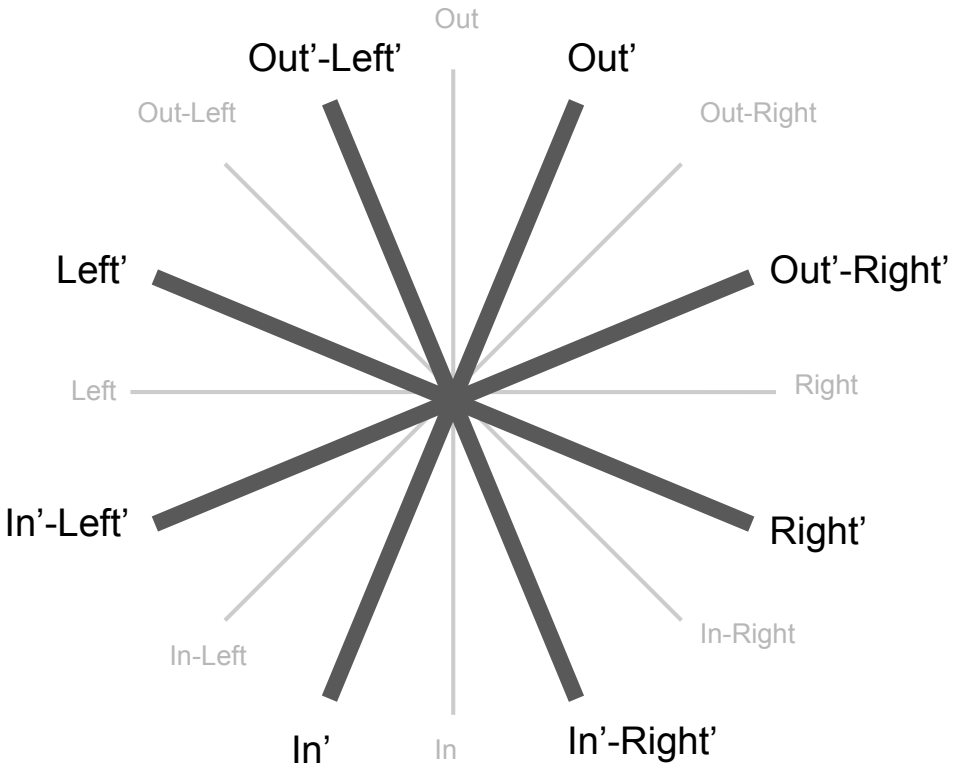

Figure 3.7: Motion primitives for test data

3.3 Reinforcement Learning (RL) Controller

The RL controller is built with the aim of an agnostic and generalizable controller in mind. Thus, for our state space, we went with an object-centric state space representation, inspired by [11] to assist with the generalization capabilities of the controller. While shaping our rewards, we kept in mind two primary objectives:

1. Manipulate an object as *far* and as *straight* as possible in a given direction.

2. Always maintain contact while allowing for sliding along the object's edges.

Keeping in sync with the IK controller, joint angles were chosen as the action space.

To assess the generalizability of the RL controller, we test its performance on new motion primitives as its test data. The test motion primitives are path interpolations between the directions of the training data and are identified as [Out', Out'-Right', Right', In'-Right', In', In'-Left', Left', Out'-Left']. The test motion primitives are shown in Figure 3.7.

The following subsections give us more specifics on the state space metrics, action space, rewards and RL policy used.

3.3.1 State Space

We choose a dense state space comprising of 6 different metric groups-

- Motion primitive direction representation
- Object pose
- Object size
- Joint-Object distance
- Joint position
- Joint angle

Metric Group	**Metric Type**	**Dimension**
Target	Manipulation Direction	[θ in radians]
Object	Object Pose	[[x, y], [θ]]
	Object Size	[l, b, h]
Joint-Object Distance	Obj-Left Proximal Distance	$[x]_{obj\_prox\_l}$
	Obj-Right Proximal Distance	$[x]_{obj\_prox\_r}$
	Obj-Left Distal Distance	$[x]_{obj\_dist\_l}$
	Obj-Right Distal Distance	$[x]_{obj\_dist\_r}$
Joint Position	Left Proximal position	$[x, y]_{l\_prox}$
	Right Proximal Joint position	$[x, y]_{r\_prox}$
	Left Distal Joint position	$[x, y]_{l\_dist}$
	Right Distal Joint position	$[x, y]_{r\_dist}$
Joint Angle	Left Proximal Angle	$[\theta]_{l\_prox}$
	Right Proximal Joint Angle	$[\theta]_{r\_prox}$
	Left Distal Joint Angle	$[\theta]_{l\_dist}$
	Right Distal Joint Angle	$[\theta]_{r\_dist}$

Table 3.1: State Space Metrics

Each individual metric falls under one of the metric groups. The state space consists of a total of 23 dimensions comprised of the following metrics:

Each metric of the state space is normalized over its minimum and maximum values.

3.3.2 Action Space

We use joint angles as our action space. Since it is a 2 link 2 fingered hand, the action space is four dimensional consisting of joint angle values for the left proximal joint, left distal joint, right proximal joint and right distal joint.

3.3.3 Rewards

The Reward is a weighted sum of a contact reward and a distance reward, shaped by our two main objectives.

1. Contact Reward: We wanted the controller to be extremely sensitive to this reward. A slight change in contact distance should indicate a strong consequence to ensure that the contact constraint is enforced. We chose a continuous reward function instead of a discrete on/off reward to encourage sliding along the object surface. For this, we choose an exponential reward.

$$Cont_{rew} = -exp(Cont_{dist}) \text{ where,}$$

$Cont_{rew}$ is the contact reward and $Cont_{dist}$ is the contact distance in metres.

$$Cont_{rew} = \begin{cases} -100, & \text{if } Cont_{rew} <= -1000 \\ \frac{Cont_{rew}}{100}, & \text{otherwise} \end{cases}$$

If the reward is less than -1000, it is capped at - 100 to avoid overflow. Else, it is divided by 100.

2. Direction Reward: This reward is measured by considering the axis of the particular direction as X-axis and it's perpendicular as Y-axis. A negative penalty is given to the 'y' direction to ensure straightness of the trajectory.

This reward is then calculated as:

$$Dist_{rew} = X_{rew} - 0.5 * |Y_{rew}| \text{ where,}$$

$Dist_{rew}$ is the distance reward,

X_{rew} is the distance in x (from the palm centre) and

$Y_{rew}$is the distance in y (from the palm centre).

Finally, the total reward is calculated as:

$$Tot_{rew} = Cont_{rew} + 100 * Dist_{rew}$$

3.3.4 Policy

For our controller, the policy we implement is the deep deterministic policy gradient from demonstrations (DDPGfD) as defined in [13].

We train our network for 20,000 episodes with a learning rate of 10^{-4} and weight decay of 10^{-4}. We use a discount factor of 0.995 along with an n-step look ahead of 5.

For the replay buffer, each timestep of an episode is stored as a single sample. It includes the current state, action, next state and reward.

The expert replay buffer consist of the samples of the IK controller. We store 5000 randomly selected episodes of the expert. We do this to ensure that sufficient samples of each motion primitive are present. The agent replay buffer has a size

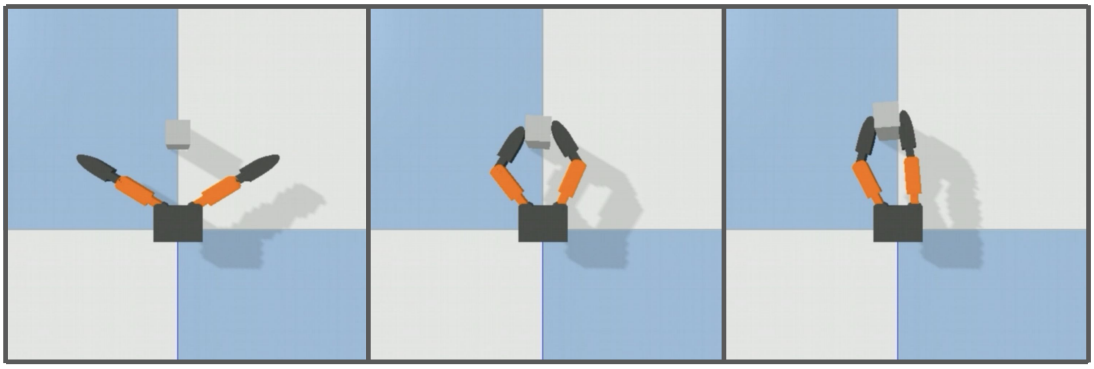

Figure 3.8: Phases in an episode: Phase 1 places the object. Phase 2 Grasps the object. Phase 3 Manipulates the object.

of 10000 episodes and stores the RL controller's trajectories. We sample from the replay buffers with a batch size of 10. We decay the sampling from the expert buffer by a factor of 0.2.

For stability, we use L2 regularization on the parameters of the actor and the critic networks.

3.4 Experimental Setup

This work was performed in simulation using PyBullet as the simulation engine. Simulations were run at 0.004 seconds per simulation step. The simulated hand and object have been modeled after the 2 fingered 2 linked hand and cube shaped object from the Asterisk Test respectively.

In order to ease the implementation, we divide an episode into phases. We

create a simulator framework that takes care of iterating through these phases, stepping the simulator and extracting relevant data at every simulation step. Thus, freeing up the user to focus purely on building phases. A phase is recognized as a slice of an episode which requires similar actions that can be grouped under a single controller. Each phases is driven by a single control mechanism. Thus, making it easy to switch out, for example, an IK control for an RL control.

For this work, we break down an episode into 3 phases as shown in Figure 3.8. In Phase 1, the fingers are extended and the object is placed at the start position. The controller used in this phase a simple PID controller that moves to a pre-defined set of joint poses. In Phase 2, we move the fingers towards the object to establish contact. Because we estimate these joint angles, this phase also uses a PID controller with pre-defined joint poses. Finally, in Phase 3, the hand attempts to manipulate the object. In this phase, either the IK or the RL controller is used depending on the type of experiment being performed.

This simulation framework deconstructs the episode so as to create a framework that is able to smoothly shift between these phases and apply the state, action and rewards suitable for a particular phase.

In the next chapter, we present results comparing the object poses along the manipulation paths between the human puppeteered data, the IK controller and the RL controller. We evaluate the performance of the RL controller by providing learning curves and plots along the test data. We also illustrate the regions of the failures of the two controllers.

Chapter 4: Results

We present results for the following: In Section 4.1 we analyze the resulting object paths for the IK and RL controller vs the human data. Section 4.2 presents the learning curves of the RL controller.

4.1 Manipulation Paths

In this section we present the results of the object paths along the manipulation directions of the two controllers.

4.1.1 IK Controller

We first present the performance of the IK controller along all motion directions. We also evaluated how the controller performs if provided with straight line trajectories along every direction instead of human puppeteering. In Figure 4.1, we compare the IK controller to the human puppeted data and also the performance over the straight line paths. Plots a, b are averaged as each motion primitive has 15 trials. Plot c has only single trials in every direction.

We find that the performance of the controller is much better in the upper horizon of the asterisk as compared to the lower horizon. The controller has significant deviation in the lower quadrant in directions In-Right, In-Left. In has not been

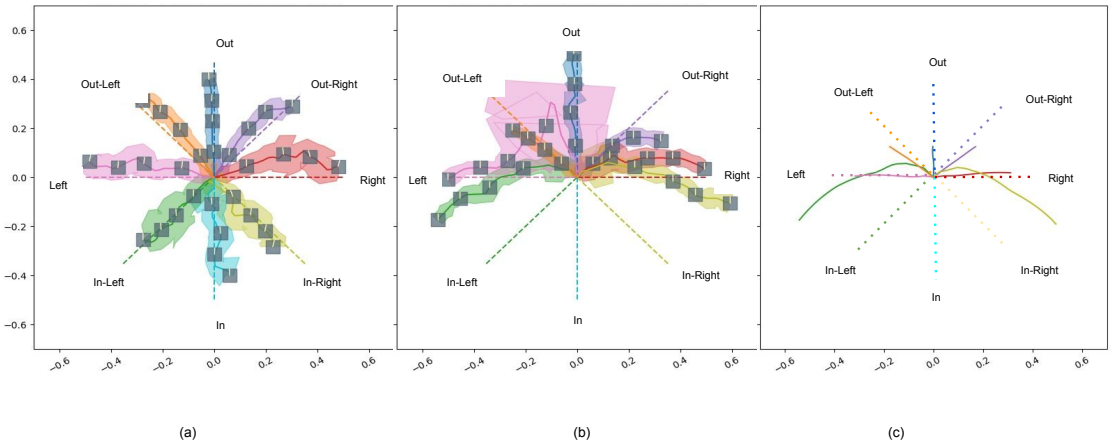

Figure 4.1: Object pose paths of the IK controller. a) Averaged human data motions, b) Averaged IK controller motions, c) IK controller straight paths. directions

plotted as the controller was unable to perform any movement in this direction.

While the straight line paths performed better in directions Left and Right, they offered no other significant improvement. It can be observed that the IK motions from the human study data travel further along in most directions as compared to the straight line motions.

Figure 4.2 shows example manipulations of direction In where the IK Controller fails. The controller is not able to move the object in this direction at all. We consider directions In-Right, In-Left as partially successful as it is able to move in the general direction even though it does a poor job of maintaining the path.

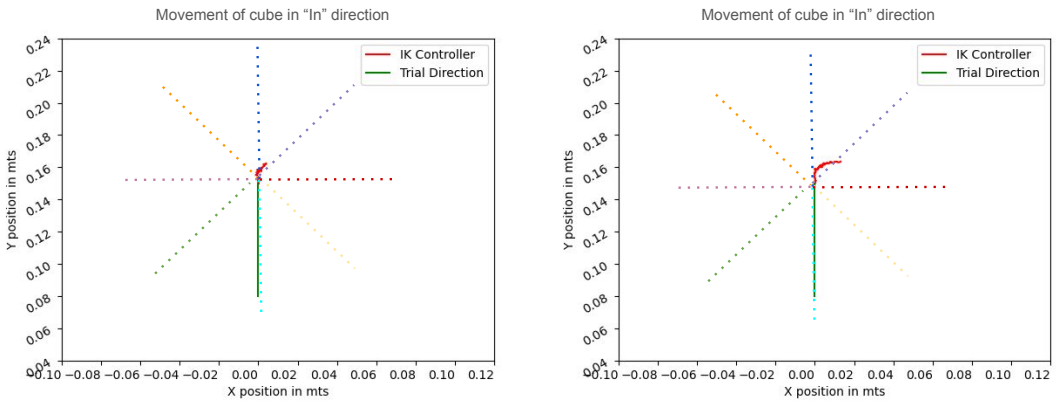

Figure 4.2: Failures of the IK Controller. (left) Failure of the controller with straight line paths, (right) Failure of the controller with human study data

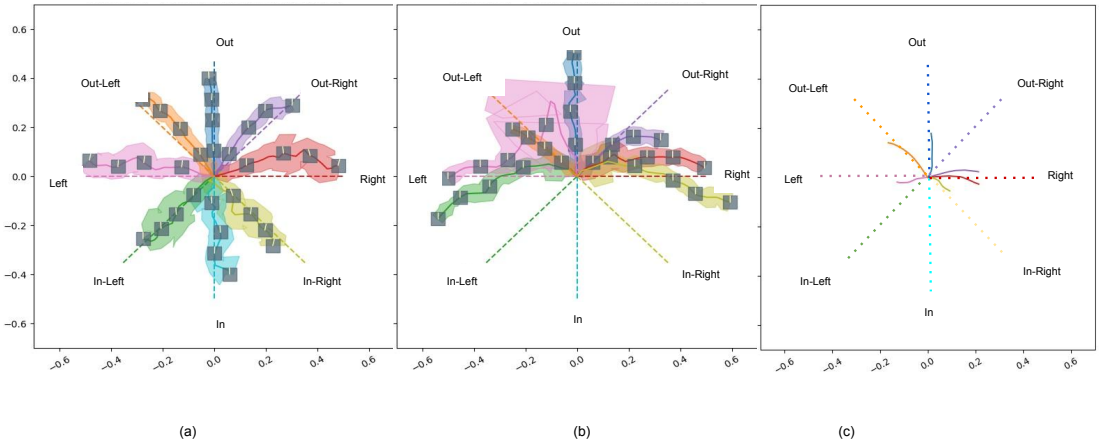

Figure 4.3: Object pose paths of the RL controller: a) Averaged human data motions, b) Averaged IK controller motions, c) RL controller motions direction. Each direction has a single RL trial path

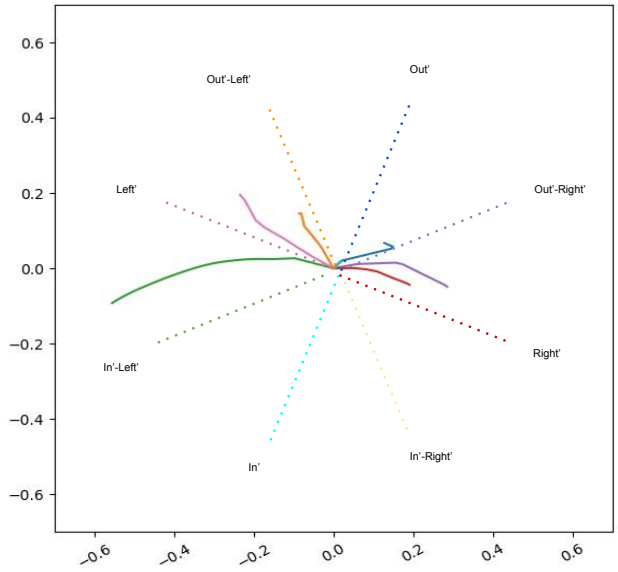

Figure 4.4: Single trials of motion direction of RL Controller along each test direction

4.1.2 RL Controller

Figure 4.3 (c) shows the object paths for the RL controller in each of the training directions. This is presented alongside the Averaged human data (a) and Averaged IK controller (b) paths for easy comparison. From all the motions that it was able to attempt, the RL controller performs well in all of the directions except the out-right one. It was not able to perform in directions In and In-Left.

We show the trajectories object pose along each of the test direction in Figure 4.4. While the test directions do not exactly follow the test paths, we can see that Out', Out'-Right', Right', In'-Left', Left', Out'-Left' roughly lie in the middle of their corresponding paths of interpolation from the motions of the train set.

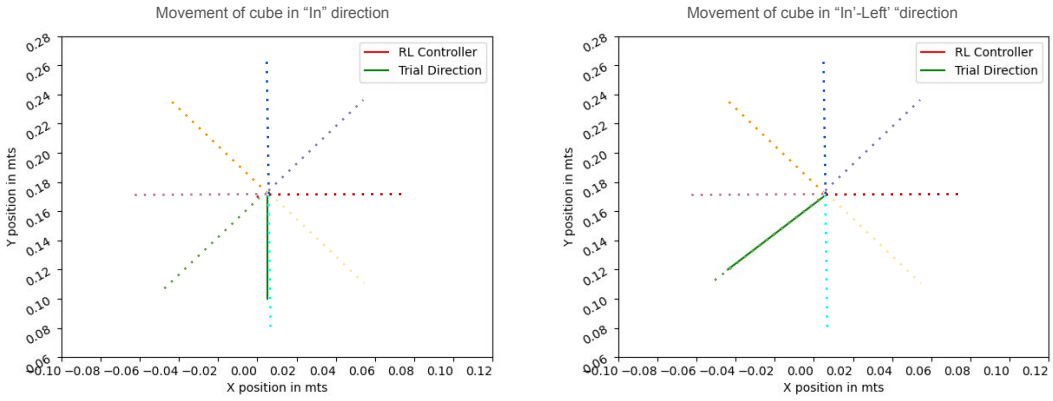

Figure 4.5: Failures of the RL Controller along training motion primitive

For example, path Out', differs heavily from the Out' direction. But this path lies in between the manipulation paths the controller performed in directions Out, Out-Right separately.

Figures 4.5 and 4.6 illustrate the failures of the RL Controller along train and test motions respectively. It is not surprising that as the IK controller (which is used for imitation learning for the RL controller) was not able to perform in direction In, neither was the RL controller. Similarly, since In'-Right', In' directions lie close to In they too fail to manipulate the object along these primitives. It was anomalous to find the controller failing in direction In-Left as well. We shed more light on what we believe to be the reasons behind the failures in the discussions chapter.

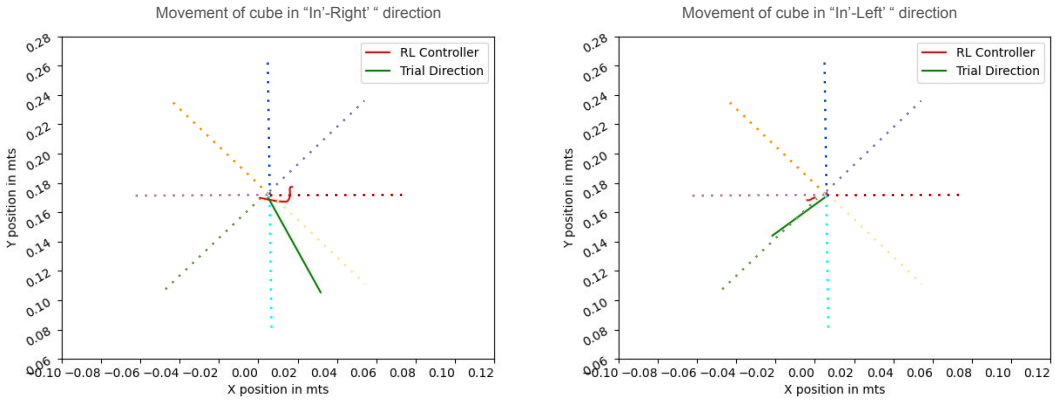

Figure 4.6: Failures of the RL Controller along testing motion primitives

4.2 RL Controller Performance

In this section we present the learning curves of the RL controller. The network was evaluated every 200 episodes. All curves are smoothened by a factor of 0.7.

Figure 4.7 represents the final reward received per episode. Although we see fluctuations in the graph, there is a general increase in the reward. The high negative values can be attributed to the design of the reward function where loss of contact and deviation in 'y' direction (with respect to the direction of motion) are heavily penalized. As the controller learns to keep contact established and deviate less in 'y' the negative value decreases.

Figure 4.8 and Figure 4.9 show the Critic and Actor losses respectively. The Critic loss is for the most part stable and doesn't explode. We observe that the actor loss reduces steadily over the last 5000 episodes.

Possible improvements to the learning performance are considered in the discussions chapter.

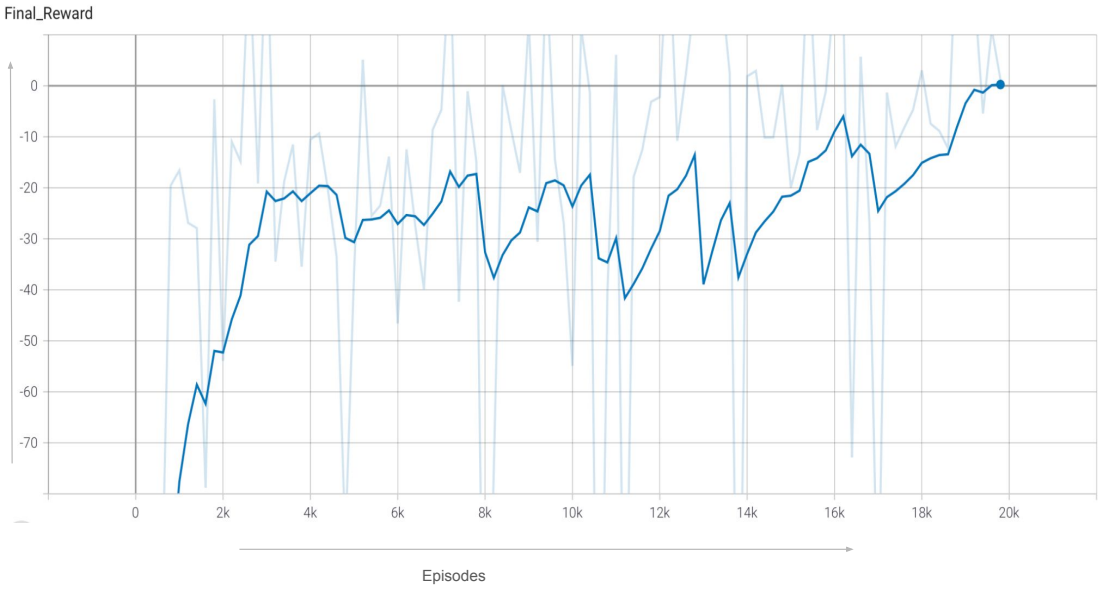

Figure 4.7: Reinforcement learning reward over episodes. All motion primitives are trained simultaneously.

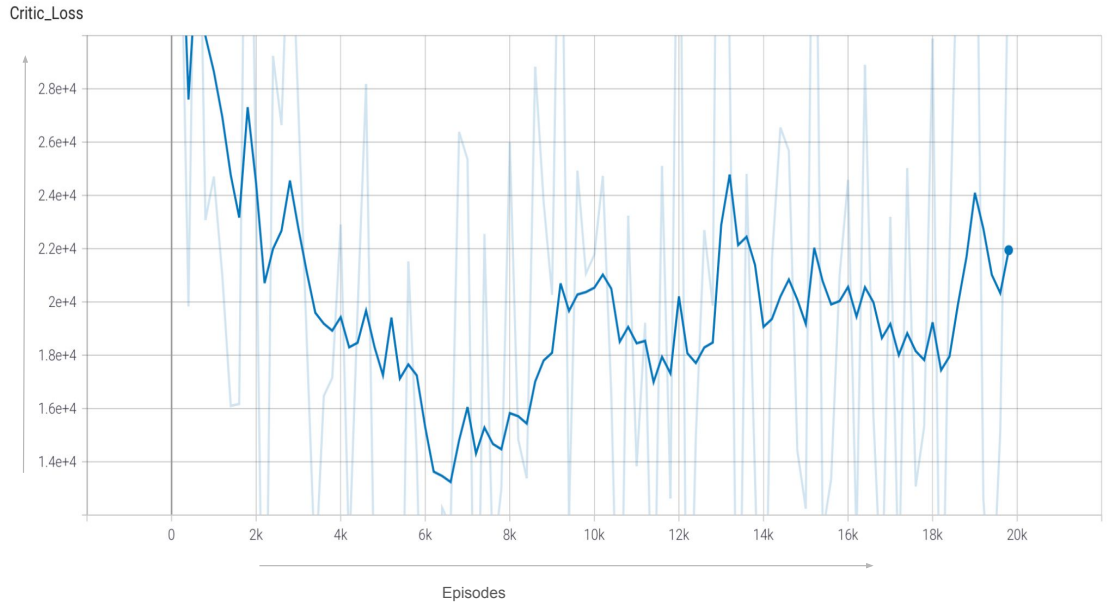

Figure 4.8: Critic Loss over 20k episodes.

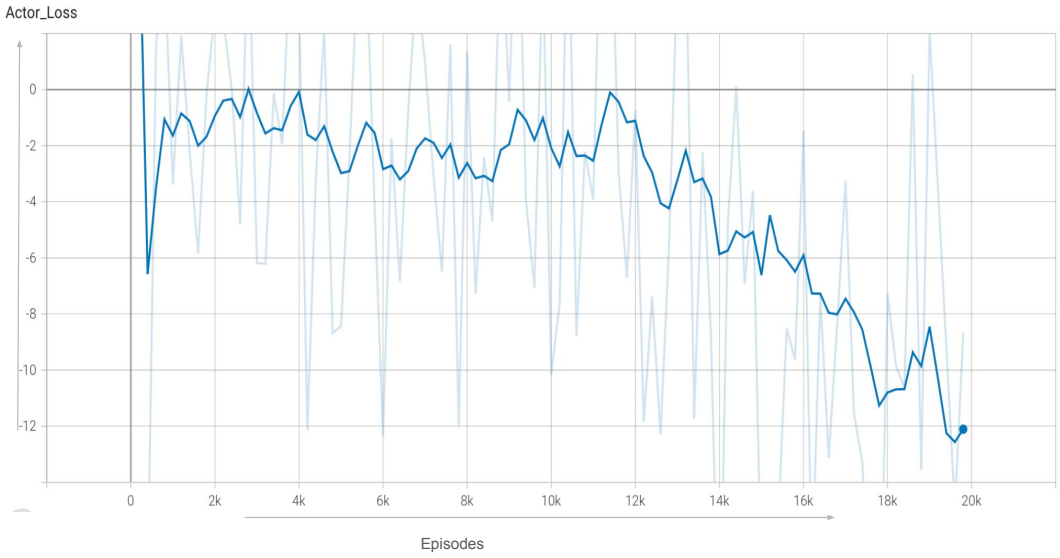

Figure 4.9: Actor Loss over 20k episodes

Chapter 5: Discussion

5.1 Failures of the RL controller

From the motions in the training data, the RL controller was able to perform well in Out, Out-Right, Right, In-Right, Left and Out-Left but not in In, In-Left. Similarly from the test data, the RL controller was not able to perform in In'-Right', In', In'-Left'. As these motions are more or less interpolated between motions in the asterisk test, it might be caused by the inability of the controllers performance in direction In.

We believe that the inability of the controller to manipulate the object along these directions could be caused by the IK controller not being able to perfectly emulate the human puppeteering in these directions due to incorrect initial contact positions. In our work, these points were rough estimates. A better representation of these contacts or a simple optimization search could result in more accurate motions as starting contact points play an important role in determining valid manipulations.

Another reason for this could also be due to the contact dynamics being different in simulation as compared to the physical hand used in the human studies. From the simulation videos, it was observed that the controller tries to move the fingers towards direction 'In' but loses contact with the object while doing so. We

believe with better suited initial contact points, the controller might be able to perform better in these directions.

For certain anomalous results such as the non-performance of the controller in In-Left, it could also be beneficial to increase the number of training episodes. It is possible that with sufficient training, the controller might be able to pick up some of the failed motions. It is possible that for some of these directions, significant variation in the y-axis is necessary to move the object. The negative penalization of the distance reward in y direction might not allow for this. With some reward shaping, this could be addressed.

5.2 Reward function fluctuations

The fluctuations observed in the reward function (Figure 4.7) might be attributed to the non-uniform nature of the reward. If the object is not manipulated far enough in a direction, it does not receive as much of a positive reward due to heavy negative penalization by the reward function. Also, this limitation can vary by hand design. A possible solution to this can be to use a different learning algorithm that takes into account "goal states", such as Hindsight Experience Replay (HER). In doing so, it would make it easier to calculate a reward by a simple measure of distance to the goal state. It is possible that longer training and some tuning of the reward functions might be beneficial in countering the fluctuations as well.

5.3 Future Work

As part of the future work, we would like to better replicate the hand in simulation and have more accurate initial contact data. We would also like to test a different learning algorithm (HER) in order to introduce goal states and stabilize our reward function. Another significant addition would be to add more directions and motions in the primitives that include rotations and also motions in 3D.

Chapter 6: Conclusion

With a structured framework that uses imitation learning and task agnostic primitive motions, we have built a learning based controller that supports decomposing complex tasks into a set of generalized motion primitives. We demonstrated the controllers ability to perform motions not part of its dataset. Due to the agnostic structure of the learning framework, it can be bootstrapped to apply to a richer set of parameterized motions. Finally, this structured approach can be applied to low level control in accordance with a high-level controller to sequence a set of motions to perform complex tasks.

Bibliography

[1] Jefferson Coelho Jr and Roderic A Grupen. Effective multifingered grasp synthesis. 1994.

[2] Tuomas Haarnoja, Vitchyr Pong, Aurick Zhou, Murtaza Dalal, Pieter Abbeel, and Sergey Levine. Composable deep reinforcement learning for robotic manipulation. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6244–6251. IEEE, 2018.

[3] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 2, pages 1398–1403. IEEE, 2002.

[4] Tingguang Li, Krishnan Srinivasan, Max Qing-Hu Meng, Wenzhen Yuan, and Jeannette Bohg. Learning hierarchical control for robust in-hand manipulation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8855–8862. IEEE, 2020.

[5] Rongrong Liu, Florent Nageotte, Philippe Zanne, Michel de Mathelin, and Birgitta Dresp-Langley. Deep reinforcement learning for the control of robotic manipulation: a focussed mini-review. *Robotics*, 10(1):22, 2021.

[6] Igor Mordatch, Zoran Popović, and Emanuel Todorov. Contact-invariant optimization for hand manipulation. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on computer animation*, pages 137–144, 2012.

[7] John Morrow, Joshua Campbell, Ravi Balasubramanian, and Cindy Grimm. Benchmarking a robot hand's ability to translate objects using two fingers. *IEEE Robotics and Automation Letters*, 7(1):588–593, 2021.

[8] Allison M Okamura, Niels Smaby, and Mark R Cutkosky. An overview of dexterous manipulation. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 255–262. IEEE, 2000.

[9] Jose L Pons, R Ceres, and Friedrich Pfeiffer. Multifingered dextrous robotics hand design and control: a review. *Robotica*, 17(6):661–674, 1999.

[10] Freek Stulp, Evangelos A Theodorou, and Stefan Schaal. Reinforcement learning with sequences of motion primitives for robust manipulation. *IEEE Transactions on robotics*, 28(6):1360–1370, 2012.

[11] Nigel Swenson, Garrett Scott, Peter Bloch, Paresh Soni, Nuha Nishat, Anjali Asar, Cindy Grimm, Xiaoli Fern, and Ravi Balasubramanian. Improving grasp classification through spatial metrics available from sensors. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6147–6153. IEEE, 2021.

[12] Herke Van Hoof, Tucker Hermans, Gerhard Neumann, and Jan Peters. Learning robot in-hand manipulation with tactile features. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 121–127. IEEE, 2015.

[13] Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

[14] Xintong Yang, Ze Ji, Jing Wu, Yu-Kun Lai, Changyun Wei, Guoliang Liu, and Rossitza Setchi. Hierarchical reinforcement learning with universal policies for multistep robotic manipulation. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[15] Shenli Yuan, Lin Shao, Connor L Yako, Alex Gruebele, and J Kenneth Salisbury. Design and control of roller grasper v2 for in-hand manipulation. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9151–9158. IEEE, 2020.