Learning to Grasp Using the Extrinsic Property of the Environment

by

Anant Sah

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2024 by the
Graduate Supervisory Committee:

Nakul Gopalan, Chair
Wenlong Zhang
Ransalu Senanayake

ARIZONA STATE UNIVERSITY

May 2024

ABSTRACT

Grasping objects in a general household setting is a dexterous task, high compliance is needed to generate a grasp that leads to grasp closure. Standard 6 Degree of Freedom (DoF) manipulators with parallel grippers are naturally incapable of showing such dexterity. This renders many objects in household settings difficult to grasp, as the manipulator cannot access readily available antipodal (planar) grasps. In such scenarios, one must either use a high DoF end effector to learn this compliance or change the initial configuration of the object to find an antipodal grasp. A pipeline that uses the extrinsic forces present in the environment to make up for this lack of compliance is proposed. The proposed method: i) Takes the point cloud input from the environment, and creates a search space with all its available poses. This search space is used to identify the best graspable position for an object with a grasp score network ii) Learn how to approach an object, and generate an appropriate set of motor primitives that converts the current ungraspable pose to a graspable pose. iii) Run a naive grasp detection network to verify the proposed methods and subsequently grasp the initially ungraspable object. By integrating these components, objects that were initially ungraspable, with a standard grasp detection model DexNet, remain no longer ungraspable.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my thesis supervisor, Dr. Nakul Gopalan, for his unwavering guidance and support throughout the entirety of my thesis. I am also deeply thankful to my thesis committee members, Dr. Ransalu Senanayak and Dr. Wenlong Zhang, for their insightful feedback and constructive criticism. Their contributions have significantly helped me in enriching the quality of this work.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

Grasping objects in a simple household setting is an inherently complex task. It requires high dexterity and often different grasp strategies to solve a single scene, where the object might not be present in an initially graspable pose. Herzog *et al.* (2014) demonstrate that it is possible to create a grasp library with the help of human kinematic teaching for diverse objects. They generate hand-crafted features for each demonstration across different objects to create grasp templates. While this method enables a system to execute multiple grasping strategies, it also necessitates the creation of new hand-crafted features for different embodiments. Additionally, this approach struggles to generalize in out-of-distribution starting positions for known objects and objects previously unknown to the system. Hence, a grasping solution that can generate multiple strategies over a generalized setting is needed.

The human hand kinematically provides for 27 DoF, but paired with the flexibility and variable stiffness provided by the musculoskeletal structure, we can generate near-infinite DoF for grasping. Standard Manipulators with parallel end effectors are incapable of replicating this compliance. For example, to grasp a credit card from a flat surface, you leverage the flexibility of your joints to hook your thumb to one end of the card. The index finger then creates a sweeping motion across the opposite edge, lifting the card over your thumb and tilting it to secure force closure. To achieve this type of motion you would either need to design a specialized end effector such as (Babin and Gosselin (2018); Andronas *et al.* (2021); Zhang *et al.* (2023)), that is capable of recreating this grasping, or use a human end effector with high degrees of freedom.

The approaches mentioned above have their own set of limitations. ((Babin and Gosselin (2018); Andronas *et al.* (2021); Zhang *et al.* (2023)) allow you to perfect a particular type of grasp, for example, scooping grasp. However, this approach standardizes the grasping procedure for the intended objects, leaving no room for generalization with a different set of objects. Consequently, it becomes difficult to transfer these strategies to a household setting where multiple grasping strategies might be necessary.

A high DoF end effector has shown the ability to replicate human dexterity to some extent. Andrychowicz *et al.* (2020) learn in-hand object manipulation of objects with a humanoid gripper. While these grippers generate much higher dexterity, they also come with larger state spaces. This increase in state space makes the learning problem more complicated. Extending their policy to learn how to approach different objects becomes near impossible due to the high dimensionality of the problem. Therefore, restricting them to only applications where the object is either right below the end effector or to in-hand manipulation tasks.

To tackle the above-stated problems, we propose a solution that uses a parallel gripper to learn how to utilize extrinsic dexterity in the form of friction, surface area, and density of different objects found in the environment. This allows us to learn a simpler policy that is cognizant of the environment around it and utilize it to make up for the lack of inherent compliance. By adopting this setup, we can learn how to approach the objects and, therefore, learn a motor policy that is more aligned with the object increasing the generalizability across objects. There has been work that shows the ability to manipulate objects in a non-prehensile manner Zhou *et al.* (2023); Dogar and Srinivasa (2012); Hogan and Rodriguez (2020). Furthermore, other studies have shown the ability to grasp objects with the help of the environment, that is pushing towards the wall to tilt the object Zhou and Held (2023), sliding the object

to the edge of the table and then grasping it from the new side Zeng *et al.* (2018); Pinto and Gupta (2017), or directly find a 6 DoF grasp approach Sundermeyer *et al.* (2021); Mousavian *et al.* (2019); Ten Pas *et al.* (2017); Mahler *et al.* (2017).

While these approaches are valid, they operate under constrained settings, and deviating from any of them would render the policy ineffective. For example, the authors in (Zhou and Held (2023); Zeng *et al.* (2018); Pinto and Gupta (2017)), learn how to grasp objects by first manipulating them. They learn a standard way to approach the objects that would not withstand the change in dynamics resulting from altered poses. (Zhou *et al.* (2023); Dogar and Srinivasa (2012); Hogan and Rodriguez (2020)) try to learn how to manipulate objects in a table setting. While they learn different contact points in varying positions, they do not establish a link between grasping and manipulation of objects in their work. (Sundermeyer *et al.* (2021); Mousavian *et al.* (2019); Ten Pas *et al.* (2017); Mahler *et al.* (2017)) provide 6 DoF grasp proposals for varying objects. While these methods are capable of providing grasp strategies beyond planar grasps, they lack the ability to find grasp proposals from objects in an initially ungraspable position.

In this work, we formulate a pipeline that combines the dexterity of nonprehensile manipulation with the rigidity of 6 DoF grasp networks to provide a way to grasp objects from ungraspable poses. Our proposed method allows us to reason about the orientation of the object and its graspability. Allowing us to change the object to a pose from which it is most likely to be grasped. To achieve this we first, create a search space based on the graspability of the object. The pose with the highest-rated antipodal points would be selected as the goal state. Second, we transform the object from the current pose to the newly obtained goal pose. We validate our method by grasping objects that were ungraspable in the beginning. This method is pose agnostic, where we find the ideal pose for an object to be present to execute planar

grasps. The object could be initially placed anywhere on the table as we learn an approach function capable of interacting with the object and changing it to the final graspable pose. We observed that our method was able to grasp 42.3% of the objects that were initially ungraspable by DexNet Mahler *et al.* (2017).

Chapter 2

RELATED WORK

Manipulation for grasping objects as an end-to-end pipeline in a household environment is a much studied problem. The various studies can be abstractly summarized into two main categories, namely Prehensile grasping and Nonprehensile manipulation. We also briefly review the model-free architectures used for such tasks.

2.1 Prehensile Grasping

A grasp in which you can hold the object between the appendages is called a prehensile grasp. In a 3D setting, they are represented as 6 DoF grasps, where the end effector is defined with 3D orientation (roll, pitch, yaw) and 3D cartesian position (x, y, z) of the gripper. For our work, we solely focus on Point cloud observations as they provide a geometric representation that can be easily extracted with the help of various architectures such as Pointnet Qi *et al.* (2017a), Pointnet ++ Qi *et al.* (2017b), and Point Transformers Zhao *et al.* (2021b).

The Pointnet representation preserves information about the local geometric features while providing an abstraction of high-dimensional data. Along with the point clouds, object models from (Calli *et al.* (2017)), the YCB dataset, are used to learn about the global structure. The approaches mentioned below either use the point cloud approach or a combination of both. (Agnew *et al.* (2021); Yan *et al.* (2018)) have reconstructed 2D images into 3D images, which are then used to generate grasp proposals. However, this procedure tends to lead to ambiguous grasp constructions. Therefore, for our purposes, we stick to point cloud data (PCD) as input.

There are broadly two ways to classify the grasp from the PCD. First is a model-

based approach, here we specifically map the PCD to an object model and then run analytical grasps. All of the computation is done in an analytical fashion where we calculate wench space matrices (Borst *et al.* (2004); Li and Sastry (1988)), force closure approximations (Zhu and Ding (2006); Liu *et al.* (2021)) or utilize the antipodal grasp rule (Shi *et al.* (2022); Cai *et al.* (2019)). While these methods are effective in producing robust grasps within the data distribution. They are unable to generalize for out-of-distribution data leading to poor performances and limited use cases.

The other method is a model-free approach, where the 6DOF grasp is directly classified based on the PCD input. The general outline for this method involves the following steps: given a PCD, sample grasps on it, train a grasp scoring network on these samples, and then post-process the grasps based on the gripper parameter. There are two ways to sample grasps, an antipodal sampler as introduced in Ten Pas and Platt (2018), or a sampling method learned as a separate model. (Liang *et al.* (2019); Mahler *et al.* (2017)) make use of the antipodal sampler. Mahler *et al.* (2017) classify the antipodal grasps based on a grasp quality convolutional neural network (gqcnn). They train a neural network model directly on the depth images. Liang *et al.* (2019) use a similar approach but instead of training on the depth images they use the Pointnet features of the point cloud. The other method learns the sampling method directly instead of using the analytical methods to generate samples. Mousavian *et al.* (2019) trained a variational autoencoder to generate such grasp samples. This grasp sampling methods allow for a more general grasp sampling, as it retains more object-centric during grasp sampling. Alliegro *et al.* (2022) use a differential point cloud sampler Lang *et al.* (2020), that is optimized with the grasp scoring model. Zhao *et al.* (2021a), learn to sample points on the point cloud and train a network that generates grasp proposals. These generated grasp proposals are further passed into a network that is used to fine-tune the generated grasps.

These models predict stable 6DoF grasps, provided that the grasps are sampled correctly. If the initial grasp samples are inaccurate, the inference of these 6DoF grasp becomes unreliable, even when the object has graspable samples in different orientations. We sample different antipodal grasps in various orientations to ensure that there are any graspable poses in the object. This ensures that if the object has only one antipodal grasp accessible, it would be placed in a pose that the manipulation can access that pose.

2.2 Non-prehensile Manipulation for Grasping

Non-prehensile manipulation is defined as the manipulation of objects without grasping them. This form of manipulation is used in reorientation, pushing, or sliding tasks on objects. It can also be used to achieve tasks such as implicit grasping of objects like using a tray to carry objects. These tasks, in general, are divided into two parts: learning the approach function, and learning appropriate motor primitives. The first part entails, how the manipulator approaches an object, Hogan and Rodriguez (2020) learn the contact modes for planar prehensile pushing tasks. This is a high-dimensional problem as each contact point could be a discrete dimension, increasing the complexity linearly. The second part is to learn torques optimized for a certain goal. Reinforcement Learning has been used extensively for this task as it allows for smooth learning of the object dynamics and a robust evaluation framework.

Pinto and Gupta (2017) make use of a shared embedding to identify pushing, poking, and grasping tasks. They directly predict torques but are unable to learn the object properties. Zeng *et al.* (2018) learns motor primitives, that is a set of torques defined by the user, using Deep Reinforcement Learning. So instead, of learning the torques directly, they learn primitives such as pushing up, down, left, and right. In this, the object starts right below the end-effector, and the reward

function is designed to encourage pushing. These two approaches are meant to be used in specialized scenarios that lead to grasping objects, they are not robust to either a change in object property or a change in object pose. Zhou and Held (2023) learn how to grasp an object using the extrinsic properties of the world. This work is closest to our work, they use a sparse reward to train a model-free Reinforcement Learning Algorithm, to push an object toward a wall and grasp it by tilting it. They learn an approach function based on a standard planar representation of the object. While this work is promising, they are unable to reason about the environment, and any change in the dynamics of the object will decrease the model's accuracy.

Finally, Zhou *et al.* (2023); Kim *et al.* (2023) learn how to manipulate objects in a nonprehensile way. They learn both the contact policy and the motor primitive, by either explicitly or implicitly dividing their policies into pre-contact and post-contact policies. Kim *et al.* (2023) optimizes two policies with different state representations. Zhou *et al.* (2023) learns a single policy that divides the object into N discrete points based on the pointcloud input, and calculates the per-point motor primitives. The set of primitives that has the maximum q value is selected.

While these methods learn contact and motor primitives, they do not aim to reason about the environment and produce poses for which grasp proposals would be possible. We combine the flexible architecture of Zhou *et al.* (2023), and incorporate reasoning about the object pose so that we can understand about the graspability of an object. This allows us to convert objects from ungraspable positions to graspable positions and finally execute planar grasp strategies to grasp the objects.

2.3 Model Free Learning

The model-free architecture allows for learning a function that maps state observations to the output actions if an avenue for exploration is provided. Various

algorithms have been used to learn policies for tasks where it isn't easy to model a transition function of the environment. Watkins and Dayan (1992); Williams (1992) forms the backbone of this approach, allowing models to learn a utility function and a policy optimized by that learned utility function.

Techniques such as Proximal Policy Optimization Schulman *et al.* (2017), Soft Actor Critic Haarnoja *et al.* (2018) and Twin-Delayed Deep Deterministic Policy Gradient Fujimoto *et al.* (2018) provide a stable and sample efficient policy to train deep neural networks as function approximations, that can be used as RL agents.

These techniques have shown promise allowing us to learn tasks mapping the proprioceptive observations and object observations directly to torques in the end effector space Vecerik *et al.* (2017); Rajeswaran *et al.* (2017). Integration of visual observations Boroushaki *et al.* (2021); Lobos-Tsunekawa *et al.* (2018); Wang *et al.* (2022) have also shown to learn a motor control policy directly. These solutions find a mapping between the observations and the actions space, provided you define the rewards properly, whether in a sparse or dense reward setting. In our case, external guidance is necessary in the form of a reasoning module over the graspability of the object. This is not implicitly present in the architecture of the traditional Model Free Pipeline and would have to be specifically modeled to guide the Motor policy to change the pose of ungraspable objects.

Chapter 3

BACKGROUND

In this chapter, we talk about the relevant background information. We start with Reinforcement Learning which forms the backbone of our nonprehensile manipulation. We further explain: MDP, in which the RL problem is modeled after and Q learning, a general set of algorithms that defines how a general update into our model works. We also include information about our environment here

3.1 Reinforcement Learning

In reinforcement learning, we generally have an agent A that is capable of interacting with the environment E. The environment has a structure with varying physics of which the agent may or may not be cognizant explicitly. The agent takes some observations from the environment that include the current state $s_t \in S$ of the agent. The agent is capable of executing actions, which allows the agent to interact with the environment. The objective of this paradigm is to learn a function π, that is capable of mapping state s_t, to an action $a_t \in A$, to maximize an objective function, in the form of expected future discounted reward. A Reward function R is defined to guide the policy. This reward decides how your trained agent is going to behave in your environment.

3.1.1 MDP

RL is modeled as a Markov Decision Process (MDP) which consists of:

- States S

- Actions A
- Reward $R(s_t, a_t, s_{t+1})$, where Reward is defined as the values that you receive when you transition from state s_t to state s'_{t+1} after executing action a_t
- Transition function $P(s_t \mid s_{t+1}, a_t)$, which is the probability of the agent being in state s_{t+1} provided its current state is S and it executes an action a_t.

The optimal policy π^* trained after maximizing the objective function on the Reward. For a finite horizon problem the Expected Return J can be modeled as :

$$U(\pi) = \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \tag{3.1}$$

where:

- $U(\pi)$ is the utility of policy π,
- t is the current time step,
- γ is the discount factor, that decides how far into the future we look into,
- $R(s_t, a_t, s_{t+1})$ is the reward received when transitioning from state s_t to state s_{t+1} after taking action a_t.

The utility function captures the cumulative reward an agent expects to receive over an infinite time horizon, with rewards discounted by γ^t to account for the uncertainty and temporal dependencies in the environment. We can also formulate this in a finite horizon setting, if this were the case we would replace the ∞ to the horizon H.

The expected utility can be formulated in the form of the Bellman Equation to account for temporal updates:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(R(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a) \right) \tag{3.2}$$

where:

- $Q(s,a)$ is the Q-value for state-action pair (s,a),
- α is the learning rate,
- $R(s,a)$ is the immediate reward received after taking action a in state s,
- γ is the discount factor,
- $\max_{a'} Q(s',a')$ represents the maximum Q-value over all possible actions a' in the next state s'.

3.1.2 Q-learning

Q - Learning: This is a subsection of RL algorithms that is used to find an optimal policy in an MDP. It learns based on the Q function which is represented by the state action pair. The Bellman optimality equation for Q-values, denoted as $Q^*(s,a)$, is given by:

$$Q^*(s,a) = \max_{\pi} \mathbb{E} \left[r + \gamma \max_{a'} Q^*(s',a') \mid s,a \right] \tag{3.3}$$

The updation of this function depends on the maximum Q values generated by the state action pair, that is the action which generates the maximum utility for a state is selected. This allows for the updation of your model based on the optimal action present as per your current utility. The shortcoming of this method is the inherent lack of exploration due to its Off Policy nature. This is remedied by introducing an epsilon greedy exploration strategy that is with the probability of the parameter Epsilon, the agent selects a random action, otherwise it selects an action that maximizes the Q value (exploitation).

Algorithm 1 Q-Learning

Initialize $Q(s, a)$ arbitrarily for all state-action pairs
Set hyperparameters: learning rate (α), discount factor (γ), exploration rate (ϵ)
for each episode **do**
    Initialize state s
    while episode not terminated **do**
        Choose action a using an exploration strategy (e.g., epsilon-greedy) based on Q-values for state s
        Take action a, observe reward r and next state s'
        Update Q-value for state-action pair:
        $Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right]$
        Update state: $s \leftarrow s'$
    end while
end for

3.2 Environment

We had several options that could have been used as our environment. Makoviychuk *et al.* (2021), the nvidia powered issac simulator that allows for efficient parallelism of resources. Rohmer *et al.* (2013), built on top of the bullet engine. This allows for a straightforward and modular and high-level control of the environment.In situations such as planning tasks, or learning high-level policy copaliasim is easier to interface. Our choice of simulation framework was Robosuite Zhu *et al.* (2020). The reasons were twofold, first, it is based on the MuJoCo physics engine Todorov *et al.* (2012). Mujoco renders realistic values of masses and inertia of objects, unlike the bullet engine that aims to maintain a certain balance between all of its shape. This makes learning about the physical properties extremely difficult in any other frame-

work. For tasks, similar to ours it is necessary to have this level of accuracy as far as the physics is concerned. Secondly, the rich documentation of Mujoco as compared to the newly introduced issac sim, makes it much easier to work with.

Robosuite provides various sets of environments along with different models of robots that replicate their real-world specifications. They also provide us with different controllers for the manipulator such as Operation Space Control (OSC). This allows us to operate the robot in the end effector space and learn a policy that aligns with our task. The modular environment also allowed for a straightforward way to modify an already-built environment with a realistic physics engine. The base environment of our work was built upon the bin-picking task. The objects used to train were a set of household objects and boxes that varied in size, such that their x-y plane was always bigger than the gripper length.

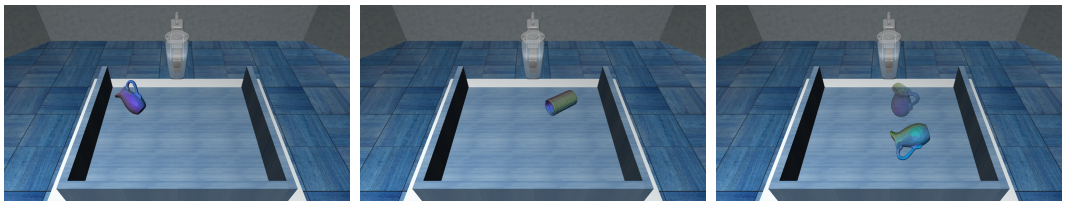

Figure 3.1: Robosuite Environment

In the Figure 5.1 we show our environment.Here we show objects that are randomly sampled from the household dataset. We allocate them arbitrary poses such that are lying flat on the surface.

Chapter 4

METHODOLOGY

There are three inherent subproblems that are required to be solved in order to grasp objects from ungraspable positions. First, we need to find a pose in which the object is going to be graspable. Secondly, we need to convert that object to this newly discovered pose. Finally, we have to generate grasp proposals to grasp the object. We solve each sub-problem independently and execute them as a sequence of temporal actions.

4.1 DECO

Alliegro *et al.* (2021) propose a point cloud completion algorithm, that they have shown to work for grasping scenarios. This makes use of two parallel encoders that are implemented as a graph convolution neural network. One encoder is used to denoise the input while the other is trained with contrastive loss to encode the global features. The loss could be defined as :

$$L_{\text{cc}}(Q, C) = L_{\text{nn}}(Q, C) + L_{\text{nn}}(C, Q) + L_{\text{mn}}(Q, C) \tag{4.1}$$

Where,

- Q is the set of contact points
- C is the set of visible contact points
- L_{nn} is the average neighbor loss
- L_{mn} is the maximal neighbor loss

With this formulation, the partial point cloud data is filled and a complete observation is produced. These observations are completed based on the ShapeNet dataset Chang *et al.* (2015). This observation allows us to run different grasp samples and find the best grasp proposal within the gripper constraints. Based on this inference, we are then able to reason about the pose that would most likely lead to stable grasps.

4.2 HACMan

This acts as our base for the nonprehensile transformation of objects, that were once in an ungraspable pose. A hybrid state representation is implemented. The state representation consists of action location α^{loc} and motion parameters a_m. The contact location is to learn the approach function. Each contact locaiton is associated with a set of torques a_m. The contact location is derived from based on the point cloud $\{x_i | i \text{ to } N\}$ where each i represents a point in the object. Therefore the contact locations are defined over a discrete state space of size N, each of these discrete state spaces will learn different motor primitives to maximize the reward. These motion policies are in the end-effector space of the robot and thus are continuous. Therefore, at each point, $N \times a_m$ actions are generated. Out of which the set of actions that provides the maximum rewards are used, creating a policy that not only knows how to create motor primitives but also where to apply those primitives.

The reward for this problem is defined as goal flow, where the mask of the goal-positioned point cloud is subtracted from the masked current object position. It is given by

$$\Delta x_i = x_i' - x_i \tag{4.2}$$

where,

- Δx_i is the goal flow.

- x'_i is the segmented goal point cloud mask
- x' is the segmented object point cloud mask

This is a 3D vector representation that maps the difference in position and orientation of the object. The negative of the goal flow is used as a reward to train the RL model and a threshold is set based on the goal flow to define task success.

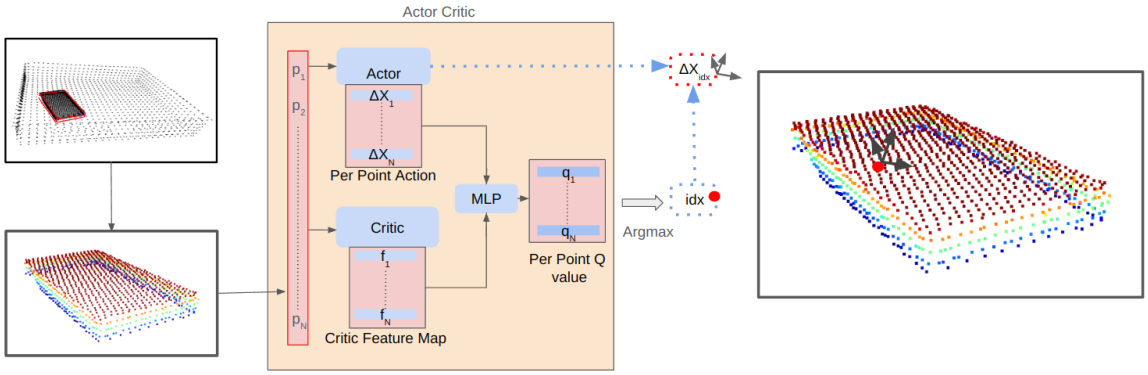

Figure 4.1: Hacman Pipeline

In Fig. 4.1 we explain the HACMan pipeline. The input point cloud data is first extracted and provided to the network. The segmented object pcd is extracted and unraveled as $p_1, p_2, ...p_N$. This acts as an input to both the actor and critic. The actor maps create per-point cartesian motor primitive, while the Critic map creates a per-point feature map. This is then passed through a Multi-Layer Perceptron (MLP), which outputs per point q values. The argmax of these q values provides us with the contact location of the pcd to access. The corresponding idx of the per point actions is executed once the gripper reaches the contact location.

The model architecture to learn this policy is based on a Q-learning-based Actor-Critic architecture. It used the Twin Delayed Deep Deterministic Policy Gradient (TD3), which consists of the actor that learns the policy and the critic that learns

the Q-value function. Here the twin part indicates the creation of two networks to learn these Q values, where the lower estimate of the two is propagated forward to avoid the overestimation bias and improve the overall stability of the algorithm. They also implement a system for Delayed Rewards as the actor-network (policy network) should be updated less frequently compared to the critic network (Q-value function network), as the Q values might deviate the policy to a poor optima, making the whole process unstable. Therefore this increases the policy as the updating to the actor (policy network) only occurs after the Q values have been optimized to a stable state.

4.3 End to End architecture

We first spawn our base environment with the help of Robosuite. This environment contains a panda robot, a table, and a bin placed on top of the table. Here a random box object is sampled and spawned in a random location in the bin. The environment provides us with depth camera information in the form of point clouds which are extracted from three cameras that triangulate the scene. Proprioceptive information is also extracted from the manipulator in the form of end effector poses. These observations allow us to create the state space representation that consists of the pointcloud of the object and the proprioception information of the manipulator.

Based on this we first run a Pointnet++ feature encoder to create segmentation masks to separate the background and foreground. This helps us to achieve X_{obj}, which is the point cloud of the object. This PCD is partially completed as all the information of the object in contact with the surface is lost. To complete this representation we make use of a DECO encoder Alliegro *et al.* (2021) as explained in section 4.1, the output of which is a completed PCD. We can then sample analytical antipodal grasp, that is points that are diametrically opposite. We condition this

sampling based on the end effector parameters, therefore leaving us with only graspable samples. We set the goal pose of the object perpendicular to the plane where that had the most constrained antipodal points.

This pose of the point cloud is now set as the goal pose. We let HacMAN Zhou *et al.* (2023) as explained in section 4.2 generate motion primitive for us that are temporally abstracted and spatially grounded. This pipeline is executed throughout ten steps. Once the policy is executed or the success condition, a new PCD is generated. This PCD is passed to a 6 DoF graspnet Mousavian *et al.* (2019). A collision-free path that goes to the contact point with the approach angle is executed. If this action leads to force closure, then we consider our task as success.

Chapter 5

EXPERIMENT AND RESULTS

5.1 Experiment

We run our experiments for 1000 different configurations where objects and their position are randomly sampled. All the objects are generated in such a way that no antipodal grasps are possible within the constraints of our gripper, that is, the box's x-y plane is greater than the width of the gripper. This eliminates any possibility for antipodal grasps, as there are no samples that lead to force closure in this configuration.

We generate a variety of different combinations for the object. We also make sure that the objects are initially ungraspable by running a standard DexNet model on our initial pose. A check for force closure in the simulator is done to verify whether the objects are graspable. We confirm that all of our generated configuration leads to initially ungraspable as DexNet is unable to generate valid grasp proposals.

5.2 Results

Here we present our results in both qualitative and quantitative. In Figure 5.1, we present objects in two configurations, the first in the start pose and the second in the goal pose. Grasp proposals are then generated on the PCD of these objects. We notice that the initial configuration does not generate any grasp. After changing the orientation of the box, we see that the same model is now capable of generating multiple grasps.

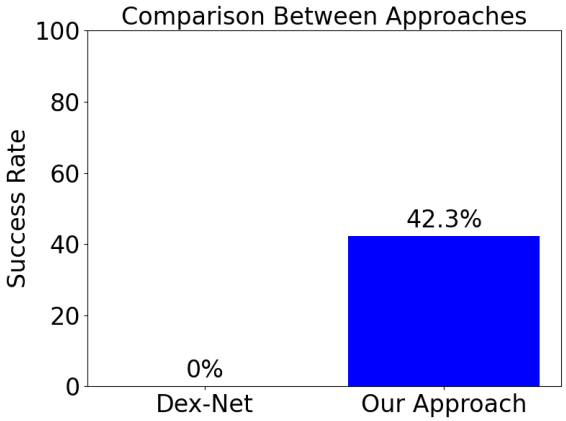

Figure 5.1: Success Rate

In our experiments that we ran on 1000 object models that were initially ungraspable, we observed that we were able to generate a policy that executed force closure on 42.3% of the objects. This accuracy is an upgrade over DexNet which was unable to find a single proper grasp proposal for this configuration.

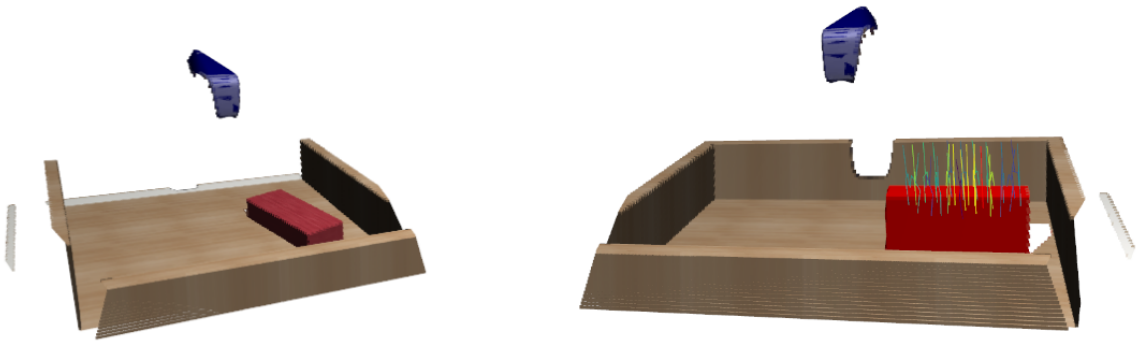

Figure 5.2: Qualitiative Results

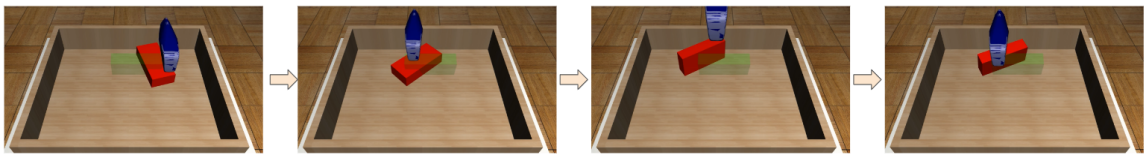

Figure 5.3: Successful Rollout

In Fig 5.2 we present a successful rollout from our experiment. This is the box experiment where the object is initially placed in an ungraspable pose. Here the object is first moved laterally in the first step. Then the object orientation is changed by tilting it from one of its edges. After that, a grasp motion is generated based on the grasp proposals.

Chapter 6

CONCLUSION AND FUTURE WORK

Based on our experiments, we can conclude that our pipeline is capable of reasoning over the graspability of objects and grasping them from initially ungraspable poses, which is beyond the ability of a naively trained grasping pipeline. Breaking down the problems to learn different instances of the task, such as pose reasoning, re-posing, and grasping allows for the abstraction and a mature policy to be executed.

We found from our experiments that while we can effectively execute our policy on simple shapes like cuboids of various dimensions, our ability to reason over the environment is limited. Our policy inherently learns about the physics of the world through its training in the environment. Therefore, it does not adapt to changes in parameters such as the object density or the position of the walls. There is no explicit reasoning over these parameters making it difficult to adapt in scenarios other than the learned setting.

Additionally, there is a lack of synergy among our pushing, reorientation, and grasping actions. Our current approach involves a fixed number of steps, changing the object pose before executing the grasping pipeline. This sequential approach leaves room for errors, where the object may initially reach a graspable pose, but eventually, change back to an ungraspable pose as we iterate through the remaining steps. Therefore, there is a need to understand the transition between the mid-level actions (pushing, re-orientating, grasping). Learning these transitions will enable us to seamlessly switch between different forms of actions and determine when to change the object pose and transition into grasping policy.

In the future, to extend our work, we would like to address the above mentioned

concerns by implementing a hierarchical structure for our policy. The hierarchical structure involves learning mid-level options, such as pushing, grasping, and flipping. Each of these mid-level options would correspond to a combination of low-level action. These actions would be learned based on the input observations of the model. Additionally, we plan to incorporate intra-options learning to better understand the execution of the mid-level policy. This would allow us to learn a synergy between all of our mid-level options. To this, we would encode a better state representation of the object than the initial point cloud observations. The objective of the new representation is to have a deeper understanding of objects with their articulation in terms of their physical property and graspability, leaving us with a more general representation for manipulating the objects.

In conclusion, while we currently focus on reasoning over simple objects, we can extend this work in more generalizable settings. A single model trained on a more apt observation space that learns the synergy between its sub-modules such as reposing, pushing, and grasping can be implemented. This enables us to expand our work to a wider variety of objects capable of reasons based on the explicitly trained physical characteristics of the object.

REFERENCES

Agnew, W., C. Xie, A. Walsman, O. Murad, Y. Wang, P. Domingos and S. Srinivasa, "Amodal 3d reconstruction for robotic manipulation via stability and connectivity", in "Conference on Robot Learning", pp. 1498–1508 (PMLR, 2021).

Alliegro, A., M. Rudorfer, F. Frattin, A. Leonardis and T. Tommasi, "End-to-end learning to grasp via sampling from object point clouds", IEEE Robotics and Automation Letters **7**, 4, 9865–9872 (2022).

Alliegro, A., D. Valsesia, G. Fracastoro, E. Magli and T. Tommasi, "Denoise and contrast for category agnostic shape completion", in "Proceedings of the IEEE/CVF conference on computer vision and pattern recognition", pp. 4629–4638 (2021).

Andronas, D., S. Xythalis, P. Karagiannis, G. Michalos and S. Makris, "Robot gripper with high speed, in-hand object manipulation capabilities", Procedia CIRP **97**, 482–486 (2021).

Andrychowicz, O. M., B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray *et al.*, "Learning dexterous in-hand manipulation", The International Journal of Robotics Research **39**, 1, 3–20 (2020).

Babin, V. and C. Gosselin, "Picking, grasping, or scooping small objects lying on flat surfaces: A design approach", The International journal of robotics research **37**, 12, 1484–1499 (2018).

Boroushaki, T., I. Perper, M. Nachin, A. Rodriguez and F. Adib, "Rfusion: Robotic grasping via rf-visual sensing and learning", in "Proceedings of the 19th ACM conference on embedded networked sensor systems", pp. 192–205 (2021).

Borst, C., M. Fischer and G. Hirzinger, "Grasp planning: How to choose a suitable task wrench space", in "IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004", vol. 1, pp. 319–325 (IEEE, 2004).

Cai, J., H. Cheng, Z. Zhang and J. Su, "Metagrasp: Data efficient grasping by affordance interpreter network", in "2019 International Conference on Robotics and Automation (ICRA)", pp. 4960–4966 (IEEE, 2019).

Calli, B., A. Singh, J. Bruce, A. Walsman, K. Konolige, S. Srinivasa, P. Abbeel and A. M. Dollar, "Yale-cmu-berkeley dataset for robotic manipulation research", The International Journal of Robotics Research **36**, 3, 261–268 (2017).

Chang, A. X., T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su *et al.*, "Shapenet: An information-rich 3d model repository", arXiv preprint arXiv:1512.03012 (2015).

Dogar, M. R. and S. S. Srinivasa, "A planning framework for non-prehensile manipulation under clutter and uncertainty", Autonomous Robots **33**, 217–236 (2012).

Fujimoto, S., H. Hoof and D. Meger, "Addressing function approximation error in actor-critic methods", in "International conference on machine learning", pp. 1587–1596 (PMLR, 2018).

Haarnoja, T., 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).

Herzog, A., P. Pastor, M. Kalakrishnan, L. Righetti, J. Bohg, T. Asfour and S. Schaal, "Learning of grasp selection based on shape-templates", Autonomous Robots **36**, 51–65 (2014).

Hogan, F. R. and A. Rodriguez, "Reactive planar non-prehensile manipulation with hybrid model predictive control", The International Journal of Robotics Research **39**, 7, 755–773 (2020).

Kim, M., J. Han, J. Kim and B. Kim, "Pre-and post-contact policy decomposition for non-prehensile manipulation with zero-shot sim-to-real transfer", in "2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)", pp. 10644–10651 (IEEE, 2023).

Lang, I., A. Manor and S. Avidan, "Samplenet: Differentiable point cloud sampling", in "Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition", pp. 7578–7588 (2020).

Li, Z. and S. S. Sastry, "Task-oriented optimal grasping by multifingered robot hands", IEEE Journal on Robotics and Automation **4**, 1, 32–44 (1988).

Liang, H., X. Ma, S. Li, M. Görner, S. Tang, B. Fang, F. Sun and J. Zhang, "Pointnetgpd: Detecting grasp configurations from point sets", in "2019 International Conference on Robotics and Automation (ICRA)", pp. 3629–3635 (IEEE, 2019).

Liu, T., Z. Liu, Z. Jiao, Y. Zhu and S.-C. Zhu, "Synthesizing diverse and physically stable grasps with arbitrary hand structures using differentiable force closure estimator", IEEE Robotics and Automation Letters **7**, 1, 470–477 (2021).

Lobos-Tsunekawa, K., F. Leiva and J. Ruiz-del Solar, "Visual navigation for biped humanoid robots using deep reinforcement learning", IEEE Robotics and Automation Letters **3**, 4, 3247–3254 (2018).

Mahler, J., 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).

Makoviychuk, V., L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning", arXiv preprint arXiv:2108.10470 (2021).

Mousavian, A., C. Eppner and D. Fox, "6-dof graspnet: Variational grasp generation for object manipulation", in "Proceedings of the IEEE/CVF international conference on computer vision", pp. 2901–2910 (2019).

Pinto, L. and A. Gupta, "Learning to push by grasping: Using multiple tasks for effective learning", in "2017 IEEE international conference on robotics and automation (ICRA)", pp. 2161–2168 (IEEE, 2017).

Qi, C. R., H. Su, K. Mo and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation", in "Proceedings of the IEEE conference on computer vision and pattern recognition", pp. 652–660 (2017a).

Qi, C. R., L. Yi, H. Su and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space", Advances in neural information processing systems **30** (2017b).

Rajeswaran, A., V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations", arXiv preprint arXiv:1709.10087 (2017).

Rohmer, E., S. P. N. Singh and M. Freese, "Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework", in "Proc. of The International Conference on Intelligent Robots and Systems (IROS)", (2013).

Schulman, J., F. Wolski, P. Dhariwal, A. Radford and O. Klimov, "Proximal policy optimization algorithms", arXiv preprint arXiv:1707.06347 (2017).

Shi, Y., Z. Tang, X. Cai, H. Zhang, D. Hu and X. Xu, "Symmetrygrasp: Symmetry-aware antipodal grasp detection from single-view rgb-d images", IEEE Robotics and Automation Letters **7**, 4, 12235–12242 (2022).

Sundermeyer, M., A. Mousavian, R. Triebel and D. Fox, "Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes", in "2021 IEEE International Conference on Robotics and Automation (ICRA)", pp. 13438–13444 (IEEE, 2021).

Ten Pas, A., M. Gualtieri, K. Saenko and R. Platt, "Grasp pose detection in point clouds", The International Journal of Robotics Research **36**, 13-14, 1455–1473 (2017).

Ten Pas, A. and R. Platt, "Using geometry to detect grasp poses in 3d point clouds", Robotics Research: Volume 1 pp. 307–324 (2018).

Todorov, E., T. Erez and Y. Tassa, "Mujoco: A physics engine for model-based control", in "2012 IEEE/RSJ International Conference on Intelligent Robots and Systems", pp. 5026–5033 (IEEE, 2012).

Vecerik, M., T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards", arXiv preprint arXiv:1707.08817 (2017).

Wang, L., Y. Xiang, W. Yang, A. Mousavian and D. Fox, "Goal-auxiliary actor-critic for 6d robotic grasping with point clouds", in "Conference on Robot Learning", pp. 70–80 (PMLR, 2022).

Watkins, C. J. and P. Dayan, "Q-learning", Machine learning **8**, 279–292 (1992).

Williams, R. J., "Simple statistical gradient-following algorithms for connectionist reinforcement learning", Machine learning **8**, 229–256 (1992).

Yan, X., J. Hsu, M. Khansari, Y. Bai, A. Pathak, A. Gupta, J. Davidson and H. Lee, "Learning 6-dof grasping interaction via deep geometry-aware 3d representations", in "2018 IEEE International Conference on Robotics and Automation (ICRA)", pp. 3766–3773 (IEEE, 2018).

Zeng, A., S. Song, S. Welker, J. Lee, A. Rodriguez and T. Funkhouser, "Learning synergies between pushing and grasping with self-supervised deep reinforcement learning", in "2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)", pp. 4238–4245 (IEEE, 2018).

Zhang, F., Z. Chen, Y. Wang, R. Bao, X. Chen, S. Fu, M. Tian and Y. Zhang, "Research on flexible end-effectors with humanoid grasp function for small spherical fruit picking", Agriculture **13**, 1, 123 (2023).

Zhao, B., H. Zhang, X. Lan, H. Wang, Z. Tian and N. Zheng, "Regnet: Region-based grasp network for end-to-end grasp detection in point clouds", in "2021 IEEE international conference on robotics and automation (ICRA)", pp. 13474–13480 (IEEE, 2021a).

Zhao, H., L. Jiang, J. Jia, P. H. Torr and V. Koltun, "Point transformer", in "Proceedings of the IEEE/CVF international conference on computer vision", pp. 16259–16268 (2021b).

Zhou, W. and D. Held, "Learning to grasp the ungraspable with emergent extrinsic dexterity", in "Conference on Robot Learning", pp. 150–160 (PMLR, 2023).

Zhou, W., B. Jiang, F. Yang, C. Paxton and D. Held, "Hacman: Learning hybrid actor-critic maps for 6d non-prehensile manipulation", in "7th Annual Conference on Robot Learning", (2023).

Zhu, X. and H. Ding, "Computation of force-closure grasps: An iterative algorithm", IEEE transactions on robotics **22**, 1, 172–179 (2006).

Zhu, Y., J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, S. Nasiriany and Y. Zhu, "robosuite: A modular simulation framework and benchmark for robot learning", in "arXiv preprint arXiv:2009.12293", (2020).