Reinforcement Learning for Targeted 3D Object Top-Down Grasping with Emphasis on Selective Shape Manipulation

Abhishek Chauhan[1], Shreeprasad Sonar[2], and Vedang Wartikar[3]

***Abstract*— The final report presents a comprehensive overview of the project titled "Reinforcement Learning for Targeted 3D Object Top-Down Grasping with Emphasis on Selective Shape Manipulation." The project addresses the challenge of developing a resilient and adaptive robotic grasping system that can recognize, target, and manipulate objects based on their distinct shapes. This report details the project's motivation, methodology, related work, experiments, and the conclusion.**

I. Introduction

In the realm of autonomous robotics, the task of precise manipulation within three-dimensional spaces has always been a formidable challenge. The demand for robotic systems capable of accurately identifying, targeting, and manipulating objects of specific geometric shapes is ever-increasing. Yet, traditional grasping algorithms often fall short in achieving the level of precision and reliability required to navigate this complex landscape effectively. This disparity between theoretical potential and practical application has been a central issue that our project tries to address. The project is motivated by the imperative need to develop a resilient and adaptive robotic grasping system [1] that bridges the gap between the theoretical possibilities of selective shape manipulation and the practical constraints of real-world environments.

The primary objective of our project is to introduce an innovative approach to tackle this fundamental challenge in robotic manipulation with the integration of reinforcement learning [2]. Our vision is to create a grasping system that not only identifies and targets objects with specific shapes but also learns to manipulate them with remarkable efficacy. The central emphasis of this project is on selective shape manipulation - the ability of a robotic system to recognize the unique geometric characteristics of objects [3] and apply tailored grasping and manipulation strategies. By prioritizing selective shape manipulation, we aim to empower robotic systems with the capacity to interact seamlessly with different range of objects with distinct geometric shapes.

To accomplish this task, we adopt an implementation-oriented approach, focusing on the development of a robust and adaptable grasping system. Our methodology revolves around the integration of state-of-the-art reinforcement learning techniques [4]. We emphasize the incorporation of advanced reinforcement learning algorithms, such as Deep Q-Networks (DQN) [5] and Deep Deterministic Policy Gradients (DDPG) [6], to train the grasping system. These cutting-edge algorithms enable the system to learn optimal grasping strategies and continuously refine its actions through iterative learning. By employing DQN and DDPG, our system becomes a dynamic, learning entity capable of adapting and improving its grasping performance over time.

A pivotal component of our methodology involves the use of a sophisticated simulation environment. We have chosen the PyBullet simulation framework, a widely recognized and trusted platform for modeling and simulating dynamic interactions in three-dimensional spaces. PyBullet offers features such as precise physics modeling, collision detection, and dynamic object interactions, making it an ideal choice for testing and evaluating our reinforcement learning-based top-down grasping system. Furthermore, PyBullet's compatibility with various robotic hardware interfaces and its extensive library of pre-defined 3D objects and environments facilitate seamless integration with our project.

To assess the effectiveness of our proposed approach, we employ a set of well-defined evaluation metrics that quantify the performance of our system. The primary metric for evaluation is the "Success Ratio," to assess the agent's ability to successfully achieve specific tasks or objectives. Moreover, the duration required to successfully grasp an object can serve as an effective evaluation metric.

II. Related Work

Reinforcement learning in robotic manipulation has been an active research area, with significant contributions from various studies [7]. Significant research endeavors in the domain of Reinforcement Learning (RL) have notably featured the deployment of deep reinforcement learning techniques, which have witnessed profound successes in training robotic systems for intricate manipulation tasks. Notably, techniques such as Deep Q-Networks (DQN) and Deep Deterministic Policy Gradients (DDPG) have emerged as frontrunners in enabling robots to learn and adapt to complex scenarios. DQN leverages deep neural networks to approximate Q-values, enabling efficient and scalable learning, while DDPG combines deep learning with deterministic policies. DQN uses a discrete action space, whereas DDPG utilizes continuous action spaces. Additionally, there are advancements in physics simulation environments like PyBullet, which provide realistic and dynamic simulation environments. These environments offer an invaluable testing ground for reinforcement learning-based grasping systems.

III. Method

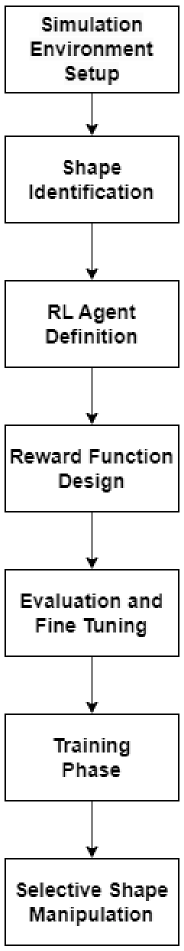

Simulation Environment Setup
Shape Identification
RL Agent Definition
Reward Function Design
Evaluation and Fine Tuning
Training Phase
Selective Shape Manipulation

Fig. 1. Different phases of the Methodology

A. Simulation Environment Setup

The first step in this approach is to establish a 3D simulation environment using the PyBullet physics engine. This environment contains objects of various shapes such as cubes and cuboids. Additionally, the robot arm and gripper models are defined to interact with these objects within the simulation. We have based our setup on one of the official example environments [8] available in the PyBullet GitHub Repository.

B. Shape Identification

Shape identification is being done by converting the RGB camera image into a NumPy array. This array is then used to train a deep learning model to identify the shape of the object. Once the model is trained, it can be used to identify the shape of an object in a new image. The object with the most similar feature vector is identified as the shape of the object in the new image.

C. RL Agent Definition

Two RL algorithms are chosen for training the agent: Deep Deterministic Policy Gradients (DDPG) for continuous action spaces and Deep Q-Networks (DQN) for discrete action spaces. The agent's observation space is designed to accommodate sensory input from RGB sensors from the overhead camera. The action space is defined to control the robot arm and gripper for grasping objects.
The State and Action for the RL algorithm are defined as follows,
State: RGB Image-based Numpy Array
Action: XYZ Offset, Vertical Angle Offset and Grasp Angle

D. Reward Function Design

The reward function is structured to incentivize the agent to grasp objects based on their shapes. It incorporates shape-specific rewards to encourage the agent to adapt its grasping behavior accordingly. If it picks up object other than the specific object, a penalty is applied to the action.

Algorithm 1 Specific_Object_Reward

```
1: A = get_base_position(specific_object)
2: B = get_base_position(other_object)
3: reward = 0
4: if specific_object not at A then
5:     reward = +1
6: else if other_object not at B then
7:     reward = -1
8: end if
9: return reward
```

E. Training Process

The training process involves training the RL agent in the simulation environment using the custom reward function and the chosen RL algorithm - DQN and DDPG with LnCnn Policy. Each training episode focuses on a specific target object with the desired shape. The RL agent learns to adapt its grasping strategy to match the desired shape.

F. Evaluation and Fine-Tuning

Once the agent is trained, its performance is evaluated by testing it on a target and other shape. The agent's parameters and the reward function are fine-tuned as needed to enhance its performance. For evaluation, we have used success ratio as a parameter to check how the model performs when tested over n grasps. Since each time the success ratio may vary due to model being trained in an RL environment, we have computed average success ratio over a few timesteps for DDPG and DQN.

G. Selective Shape Manipulation

The primary aim of this method is to show that the agent can manipulate specific objects. Based on the training algorithm above, user-defined specific shapes are rewarded with better incentives and hence prioritized for picking up by the robot arm.

IV. Experiments

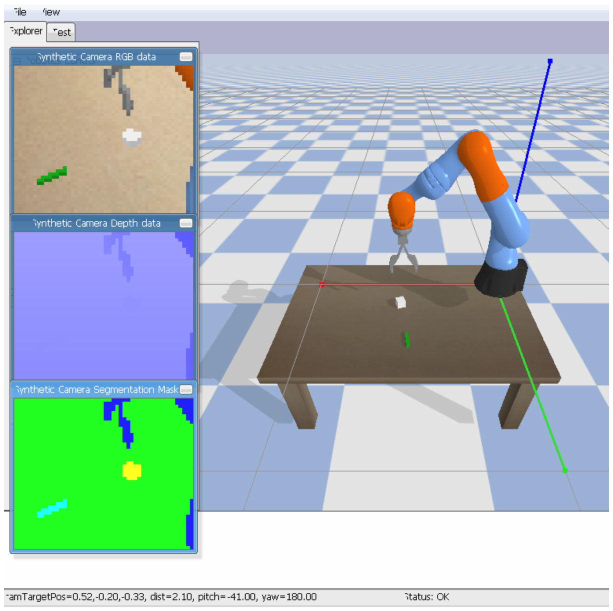
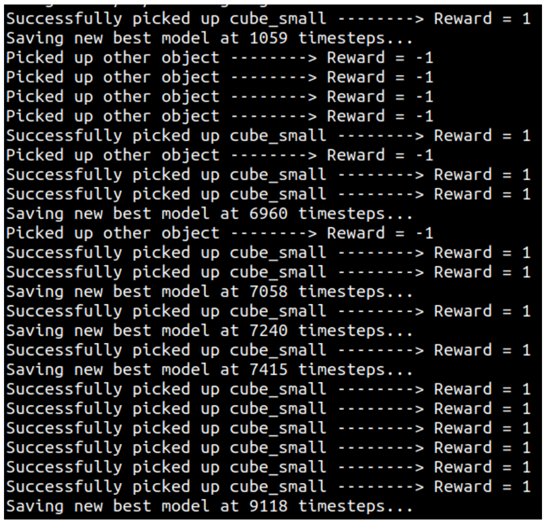

File View
Explorer Test
Synthetic Camera RGB data
Synthetic Camera Depth data
Synthetic Camera Segmentation Mask
camTargetPos=0.52,-0.20,-0.33, dist=2.10, pitch=-41.00, yaw=180.00
Status: OK

Fig. 2. Simulation Environment with Kuka Robot and Objects

In our experiments, we use PyBullet as the simulation environment (Fig. 2) to evaluate the performance of our reinforcement learning-based grasping system.



```
Successfully picked up cube_small --------> Reward = 1
Saving new best model at 1059 timesteps...
Picked up other object --------> Reward = -1
Picked up other object --------> Reward = -1
Picked up other object --------> Reward = -1
Picked up other object --------> Reward = -1
Successfully picked up cube_small --------> Reward = 1
Picked up other object --------> Reward = -1
Successfully picked up cube_small --------> Reward = 1
Successfully picked up cube_small --------> Reward = 1
Saving new best model at 6960 timesteps...
Picked up other object --------> Reward = -1
Successfully picked up cube_small --------> Reward = 1
Successfully picked up cube_small --------> Reward = 1
Saving new best model at 7058 timesteps...
Successfully picked up cube_small --------> Reward = 1
Saving new best model at 7240 timesteps...
Successfully picked up cube_small --------> Reward = 1
Saving new best model at 7415 timesteps...
Successfully picked up cube_small --------> Reward = 1
Successfully picked up cube_small --------> Reward = 1
Successfully picked up cube_small --------> Reward = 1
Successfully picked up cube_small --------> Reward = 1
Successfully picked up cube_small --------> Reward = 1
Successfully picked up cube_small --------> Reward = 1
Saving new best model at 9118 timesteps...
```

Fig. 3. Kuka Robot learning to pickup specific object

In the Fig. 3 above, DQN algorithm for 25,000 timesteps is being run. In the initial timesteps (around 1000) it can be seen that most of the times the wrong object is being picked up and that action is being rewarded as **-1**. After a few timesteps, the algorithm learns over time to pickup the correct object for which the action is rewarded as **+1**.

In our exploration of metrics and measurement methods for Reinforcement Learning (RL) algorithms, we have analyzed the key criteria used to evaluate RL approaches. [9] and therefore we make use of the following evaluation metrics:

Average Success Ratio: The success ratio calculates the correct grasps within a specified number of test episodes. Since the success ratio may vary each time a model is being tested on n episodes, we compute an average of those ratios within the same timesteps and the model. This metric provides insights into the agent's consistency in accomplishing specific tasks within the environment.

While testing the model and the grasp accuracy, we compile the metrics into a CSV file so it's easy to plot the graph. The sample metrics CSV file looks as follows,

```
algo,ts,#episodes,success
    DQN,10000,100,0.32
    DQN,10000,100,0.28
    DQN,25000,100,0.45
    DQN,25000,100,0.44
    DQN,50000,100,0.54
    DQN,50000,100,0.69
    DQN,75000,100,0.68
    DQN,75000,100,0.65
```

We tested the trained DQN model for the following timesteps - *10000*, *25000*, *50000* and *75000*. The accuracy of grasping the specific object goes on increasing as the timesteps increase. The graph (Fig. 4) below shows the average success ratio which is computed by running the DQN model for 5 test iterations over *100* episodes each. For models above *25000* and above, the training requires more than *10+* hours of training.

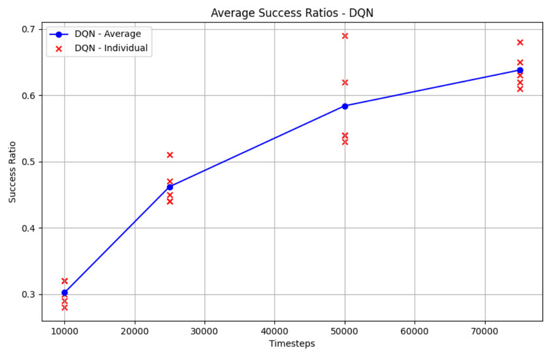


Fig. 4. DQN - Avg Success Ration VS. Timesteps

We also trained the DDPG model for *10000* and *25000* timesteps, but the test results showed less than *10%* accuracy for *10000* steps and around *22%* accuracy for *25000*. DQN's accuracy is almost *2x* times better than DDPG for our environment. DQN is better at learning using the RGB provided array and tries to go near the base position of the specific object to attempt a grasp action.

V. Conclusion

After observing the above experimental results, we concluded that DQN significantly outperforms DDPG when tasked with grasping objects. We explored a few potential reasons on why this could happen. One reason would be

DDPG often has more hyperparameters [10] to tune (different learning rates for actor and critic), and its performance can be sensitive to changes in these parameters whereas DQN is easier to implement with default parameters.

References

[1] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, and Vincent Vanhoucke. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, page 651–673. PMLR, 2018.

[2] Deirdre Quillen, Eric Jang, Ofir Nachum, Chelsea Finn, Julian Ibarz, and Sergey Levine. Deep reinforcement learning for vision-based robotic grasping: A simulated comparative evaluation of off-policy methods. In *Conference on Robotics and Automation (ICRA)*, page 6284–6291. IEEE, 2018.

[3] Siyuan Liu, Yifan Chen, and Fei Li. Object detection with deep reinforcement learning. *arXiv preprint arXiv:2208.04511*, 2022.

[4] MARWAN QAID MOHAMMED, KWEK LEE CHUNG, and CHUA SHING CHYI. Review of deep reinforcement learning-based object grasping: Techniques, open challenges, and recommendations. In *IEEE Access*, pages 178450–178481. IEEE, 2020.

[5] Dong Han, Beni Mulyana, Vladimir Stankovic, and Samuel Cheng. A survey on deep reinforcement learning algorithms for robotic manipulation. In *Advances in Intelligent Robotics Systems Based Machine Learning*. MDPI, 2023.

[6] Gang Peng, Jinhu Liao, Shangbin Guan, Jin Yang, and Xinde Li. A pushing-grasping collaborative method based on deep q-network algorithm in dual viewpoints. 2022.

[7] Yazici Baris. Branch Dueling Deep Q-Networks for Robotics Applications, 2020.

[8] Bullet Physics SDK. Bullet3. https://github.com/bulletphysics/bullet3, 2020.

[9] Laura L. Pullum. Review of metrics to measure the stability, robustness and resilience of reinforcement learning. *arXiv preprint arXiv:2203.12048*, 2022.

[10] Stable Baselines. Ddpg documentation. https://stable-baselines.readthedocs.io/en/master/modules/ddpg.html#parameters.