Learning Dexterous Robotic Manipulation

Riley Durbin
Dept. of Computer Science of The University of Alabama
Tuscaloosa, Alabama
radurbin@crimson.ua.edu

***Abstract*—In this project, I attempted to teach a robotic arm to successfully approach and grasp a screwdriver. This was accomplished using the stable-baslines3 library by training a model using Proximal Policy Optimization (PPO) to optimize the robot's movement policy. Through the experimentation of different reward functions to incentive the robot's actions, the robot was able to develop a strong policy to consistently pick up the object.**

I. Introduction

Robotic manipulation aims to enable robots to moving objects, perform tasks, and assist humans in a real environment. By teaching robots to do these tasks, there have been many advancements in industries like healthcare and manufacturing. This emphasizes the importance of robotic manipulation, specifically with reinforcement learning (RL), which enables robotic systems to learn how to perform these tasks intelligently instead of following explicit instructions. By learning from their environment and forming an optimal policy, RL algorithms can adapt to changing environments and develop different behaviors for each situation. This combination of reinforcement learning and robotic manipulation makes the field of machine learning very promising for the advancement of intelligent robotic systems.

This specific procedure involved the use of a simulation environment to safely and inexpensively test, train, and evaluate the robot. By simulating realistic physics and movement, the robot was able to accurately depict what a real robot would do in a similar environment with a similar situation. Through the understanding of the simulation environment, the development of the primary reward function, and the use of Proximal Policy Optimization (PPO), the robot was taught to successfully navigate through the simulation environment to approach and pick up an object.

The project was ultimately successful in the goal of teaching the robot to reach for and grasp the object. Through a series of iterations, a final reward function and RL algorithm were settled upon that resulted in an 78% success rate for the robot. This emphasizes the importance of careful reward design when training the model. This also emphasizes the potential of reinforcement learning and robotic manipulation for performing real-world tasks to assist humans.

II. Method

The approach for this project involved the use of a simulated environment to replicate a Sawyer robotic arm approaching a table and picking up an object. This simulated environment was created using Pybullet and had a variety of objects to be picked up by the robot. The specific object used in this project was a screwdriver. An image of this simulation environment is shown in Figure 1. The approach in writing the reward function was to design an incentive so that the robotic arm would first move above the screwdriver and then slowly lower itself down towards it until it can grasp it. This was done by first rewarding the robot if it moved within a certain distance threshold and then rewarding it again if it was within a closer distance threshold while being aligned in the x and y-axis. Finally, a final large reward was given to the robot if it was aligned in the x, y, and z-axes, in which it could grasp the object. These three reward thresholds stacked on top of each other, and the "reward zones" can be visualized in Figure 2. This design served to guide the robotic arm towards the screwdriver from above it.

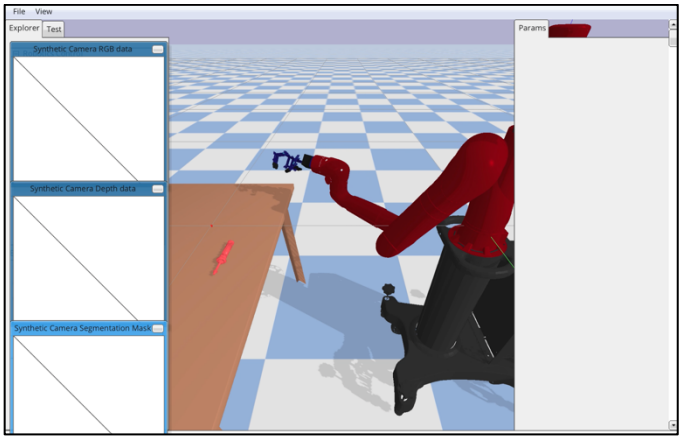

Fig. 1. The simulation environment used to simulate the Sawyer robotic arm picking up a screwdriver.

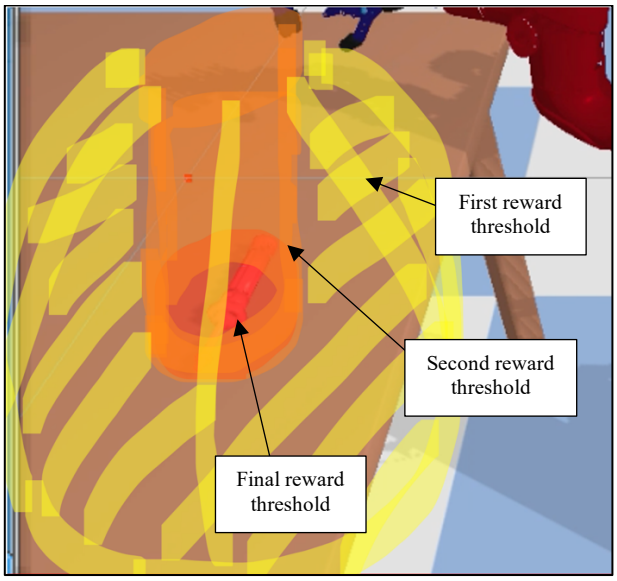

Fig. 2. The reward function visualized in the simulation environment.

The reinforcement learning (RL) algorithm used in this project is Proximal Policy Optimization (PPO). PPO is a model-free RL algorithm that focuses on policy optimization. Specifically, it is an actor-critic algorithm, which means that it combines elements of both value-based and policy-based methods. For PPO, the "actor" learns a policy for each state, exploring the environment to find the best actions for each state. The "critic" provides feedback on the quality of these actions for each state by comparing the predicted rewards to the actual rewards received. The policy is iteratively updated to optimize and maximize the rewards. In this case, the robot receives the most rewards when it approaches and picks up the screwdriver. This specific version of PPO comes from the stable-baselines3 library, so clipping was used to avoid too large of an update each iteration. PPO is a popular RL algorithm, so it was used for this project due to its ease of use, stable policy updates, and quality performance.

The reward function used in this project aims to guide the robot to first move above the object and then to lower down towards it, as described previously. This reward function can be illustrated as a piecewise function. Letting R be the reward function, d represent the distance from the robotic hand to the screwdriver, and x, y, z represent boolean values for if the robotic hand is aligned with the object in each respective axis. The piecewise function for the reward function is:

$$R = \begin{cases} 0 & d \geq 0.3 \\ 10 * \left(1 - \left(\frac{d}{0.3}\right)\right) & 0.1 \leq d < 0.3 \\ \left[10 * \left(1 - \left(\frac{d}{0.3}\right)\right)\right] + \left[100 * \left(1 - \left(\frac{d}{0.1}\right)\right)\right] & d < 0.1 \text{ and } x = 1 \text{ and } y = 1 \\ \left[10 * \left(1 - \left(\frac{d}{0.3}\right)\right)\right] + \left[100 * \left(1 - \left(\frac{d}{0.1}\right)\right)\right] + 500 & x = 1 \text{ and } y = 1 \text{ and } z = 1 \end{cases}$$

As illustrated by this piecewise function, the robot is rewarded for moving closer to the object, and it is rewarded more for being directly above it (aligned in the x and y axis). Finally, the robot is given a large final reward for being directly on the object, in which it picks it up. This reward function was successful in teaching the robot to approach and pick up the screwdriver.

III. Results

By using the reward function previously mentioned and training the model using PPO for 20,480 timesteps, the model was able to achieve a success rate of 78%. This is an excellent success rate, as it indicates that most of the time, the robot was able to successfully navigate to and pick up the screwdriver. This success rate is based off of 100 tests. This serves as a quantitative measurement for the overall performance of the robotic arm. By testing different reward functions, I was able to use the success rate of each reward function to determine which function was optimal. Figure 3 depicts a successful grasp trial, and Figure 4 depicts a failed grasp trial.

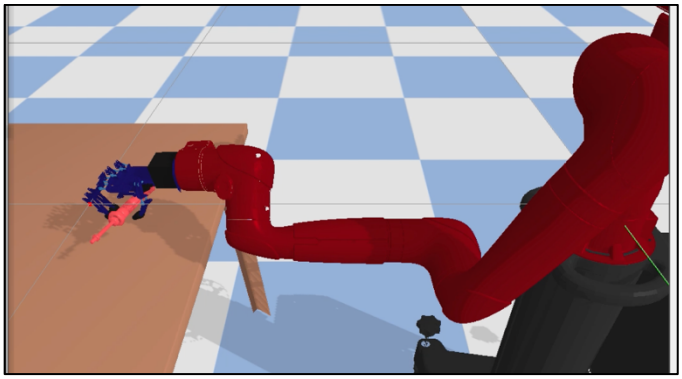

Fig. 3. A successful grasp trial.

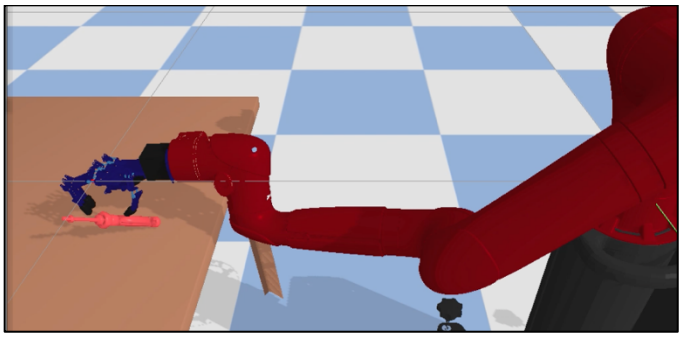

Fig. 4. A failed grasp trial.

Figure 3 indicates a successful grasp trial. In this case, the robotic arm moved to directly above the screwdriver and then lowered down to pick it up. In many of the trials, the robotic hand rested with its fingers around the screwdriver for a period before finally grasping it. It seems like this is a result of the robot not immediately recognizing that it is in the position to grab the object, so that could be improved through future development. Despite this period of waiting, the robot still picks up the screwdriver in almost every trial that I observed. Figure 4 represents a rare failed grasp trial. In this trial specifically, the hand approached the screwdriver like normal, but one of the fingers bumped the screwdrivers, turning it slightly. As a result, the hand attempted to grasp it, but the hand grabbed one side of the screwdriver at an angle, leading to the screwdriver slipping from the robot's grasp. This rare case could be fixed through an improved reward function that measures the distance between each individual finger and the screwdriver, instead of the distance between the whole hand. This could ensure that the screwdriver is centered to avoid bumps. This specific case may be unique for this object, as some other objects are not as thin

and may not turn in the same way that the screwdriver did in this trial. Another failed trail that I observed involved the robotic arm approaching the screwdriver and resting with its hand around it, but it never moved its fingers to grasp it. I think that this could be the result of a few possible issues. The reward function may not be correctly indicating to the robot that it is in a position to grasp it. There also could be an issue with the robotic fingers being stuck on the table. Further research and development could alleviate these issues. In all of the observed trials, the robotic arm successfully approached the screwdriver in the manner designed by the reward function, even though it failed to grasp the object in some trials. This leads to the conclusion that this project was successful.

IV. Discussion of Results

There were a variety of factors that led to the decision to use PPO as the reinforcement learning algorithm. Each RL algorithm, like A2C, DDPG, DQN, SAC, and PPO, has their own advantages and disadvantages that had to be considered. One notable advantage of PPO is the ability to remain stable and simple to implement despite complex behaviors and interactions with the environment. The project required that the robotic arm had to adapt to various situations, as the object was placed on random parts of the table. It also had to interact with the table physically. PPO's ease of implementation and training stability contributed to the decision to select it over other RL algorithms. One disadvantage of PPO is the data inefficiency and potential for unstable updates. The data inefficiency stems from the fact that each policy needs to generate an entire new set of data for each update. The unstable updates can come from the step size. If the step size is too large, the policy will worsen, and if the step size is too small, the learning process will be slow. Despite these limitations, PPO proved to be more successful than the other RL algorithms that I tried, as it trained the model efficiently and successfully. This is evident through the high success rate of the robot.

The designed reward function consists of four key parameters. The distance parameter measures the distance from the robotic hand to the screwdriver. It is represented by the variable d in the piecewise equation previously discussed. The other three key parameters are boolean values that indicate alignment in the x, y, and z-axes. These are represented in the piecewise equation by x, y, and z, respectfully. These parameters were selected by first designing the outline for the reward function. The goal was to first move the robotic arm towards the object. This was accomplished by rewarding the robot if the distance d was within a distance threshold, which was set to 0.3. Then, the robot needed to move its arm above the object. This was accomplished by adding more to the reward if the distance d was within a distance threshold 0.1 *and* the hand was aligned with the object in the x and y-axes. Finally, the robot needed to lower its hand and grasp the object. To accomplish this, a final large reward was added if the hand was aligned in the x, y, and z-axes, essentially ensuring that it could grab the object. By stacking each of these rewards, the robot was taught to first move towards the screwdriver, then align itself above it, then lower down towards it and grasp it. This reward function was refined and changed to achieve the required results.

Conclusion

In conclusion, this project demonstrated the effective use of machine learning with reinforcement learning to teach a robot to move in a simulated environment to approach and grasp an object. By specifically using the PPO reinforcement learning algorithm, the robot was successfully taught to pick up the screwdriver. The robot was able to perform this task with a 78% success rate, which further emphasizes the effectiveness of PPO as an RL algorithm and the reward function's design. This reward function incentivized the robot to move toward the object, and the PPO algorithm facilitated stable policy updates to optimize the policy. Some grasp trials failed, which highlights the specific areas where this project could be improved in the future. There were also multiple RL algorithms and reward functions tested, but they were changed or switched because they were not producing ideal results. All in all, this project illustrates the potential for reinforcement learning's use with robotic and object manipulation.

References

[1] R. S. Sutton and A. Barto, *Reinforcement learning : an introduction*. Cambridge, Ma ; Lodon: The Mit Press, 2018.

[2] "Manuscript Templates for Conference Proceedings," *@IEEEorg*, 2020. https://www.ieee.org/conferences/publishing/templates.html

[3] "PPO — Stable Baselines3 1.4.1a3 documentation," *stable-baselines3.readthedocs.io*. https://stable-baselines3.readthedocs.io/en/master/modules/ppo.html