Reinforcement Learning for Robotic Arm Pick-and-Place Action using PPO

Flora Huo

May 16, 2022

1 Introduction

This project aims to build a control policy that can robustly manipulate a simulated robotic arm to pick-and-place boxes with arbitrary initial and target locations

2 Background

We used Proximal Policy Optimization [1]. Proximal Policy Optimization (PPO) is an on-policy actor-critic algorithm. It consists of a policy π_θ and a value function V_ϕ, parameterized respectively by θ and ϕ. PPO optimizes the policy π_θ using policy gradient formed by advantage estimate $A^t = A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$

PPO minimizes a clipped-ratio loss over mini_batches of trajectories collected under $\pi_{\theta_{\text{old}}}$:

$$L_\pi(\theta) = -\mathbb{E}_{\pi_\theta}\left[\min\left(\rho_t(\theta)A_t, 1-\epsilon, 1+\epsilon\right)\right], \rho_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}. \tag{1}$$

$Q^{\pi_\theta}(s_t, a_t)$ is estimated using empirical return and $V^{\pi_\theta}(s_t) \approx V_\phi(s_t)$ is regressed against a target of discounted returns, V_t^{targ}, to minimize the Generalized Advantage Estimation:

$$L_V(\phi) = \mathbb{E}_{\pi_\theta}\left[\left(V_\phi(s_t) - V_t^{\text{targ}}\right)^2\right] \tag{2}$$

We simulate the pick-and-place environment using PyBullet (see Figure 1). In this environment, the robotic arm's goal is to reach to the green box (object location), pick it up, bring it to the red platform (target location) and drop it there. Throughout the training we set the box mass to be 50kg. Dimension of the box is 0.1m x 0.1m x 0.1m, and the dimension of the platform is 0.15m x 0.15m x 0.005m. The action space is a 7-dimensional vector corresponds to the joint torque. The values are

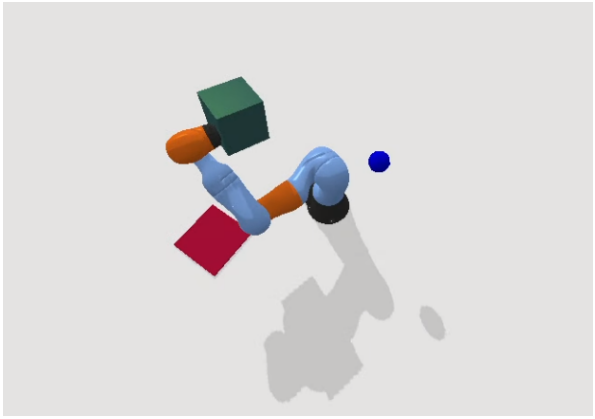

Figure 1: Training in Pybullet

Phase	Distance-term (x)	Constraint	Condition
01	> 0.05	/	once the distance-term is < 0.05, enter phase 02
02	> 0.05	constraint created	once the distance-term is < 0.05, enter phase 03
03	< 0.05	constraint removed	box dropped successfully

Table 1: Phases

clipped in between [-0.5, 0.5] to mitigate jerk. Using `createConstraint` and `removeConstraint`, we are able to create a vacuum gripper.

x in all the reward functions represents the distance-term used in training.

3 Phases

Phase 01 requires the end effector of the robot arm to reach to the box (the object). Once the distance is within 0.05m, a constraint will be created between the box and the end effector. Once the constraint id is generated, we will enter Phase 02. Phase 02 requires the end effector of the robot arm to reach to the target location. Once the distance between the end effector and the target location is within 0.05, the constraint will be removed. Phase 03 is when the constraint is removed, indicating the box has been dropped at the target location.

For more detailed information please refer to **Table 1**.

4 Training with fixed object location and fixed target location

For the training with fixed object location and fixed target location, 5 reward functions are selected in total.

The first three reward functions we compared are $R = 2^{-x}$, $R = 0.8^x$, and $R = \frac{1}{0.5+x}$. In **figure 2** and **figure 3** the max reward learning curves are compared, followed by the normalized reward learning curves. In **figure 3**, we observed that $R = 2^{-x}$ has overall a higher reward compared to the other two functions at all time steps, and once it reaches to a steady phase, it still has a relatively higher reward compared to the other functions.

In testing, the robot arm trained with the three reward functions shows little to no motivation to reach the object within 0.05m, it displayed motion to reach towards the object, but stopped before it can reach the goal to create constraint. $R = 2^{-x}$ shows the most motivation but it's still not high enough to complete the task. One explanation could be that the reward is already high enough for the robot despite it haven't reached the object within 0.05m to create the constraint. To improve this situation, further training was designed.

We compared $R = 2^{-x}$ with the following two reward functions:

$$R2 = \begin{cases} 2^{-x} & 0 \leq x \leq 0.07 \\ 0 & otherwise \end{cases}$$

$$R3 = \begin{cases} 2^{-x} & 0 \leq x \leq 0.07 \\ 0.8^x - 0.5 & otherwise \end{cases}$$

In **figure 5**, the normalized reward comparisons, although $R1$ has a relative steady learning rate, $R2$ shows higher rewards starting 3.5×10^6 time steps and increasing, while $R1$ has already entered the steady phase. In testing, the robot arm trained under $R2$ shows an significant increase of motivation to reach the object in order to create constraint.

5 Training with fixed object location and random target location

We increased the complexity of the task by making the target location random within the range of $[-0.55, -0.25]$ & $[0.25, 0.55]$ respectively for both x and y coordinates, and z remains 0.0. We selected

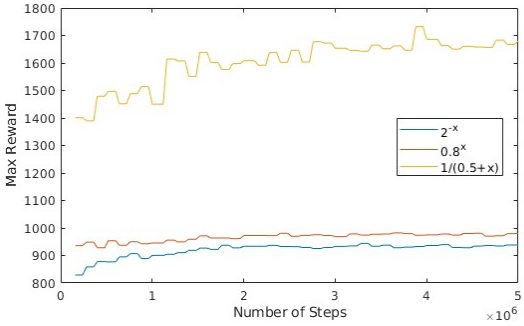

Figure 2: Fixed object and target location - Max reward comparison

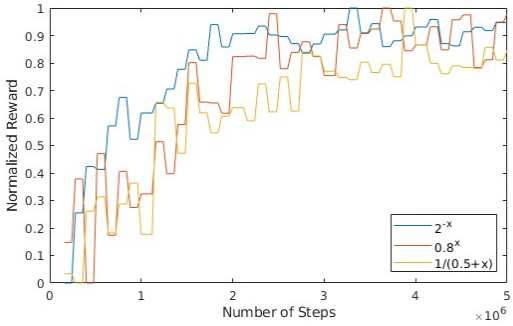

Figure 3: Fixed object and target location - Normalized reward comparison

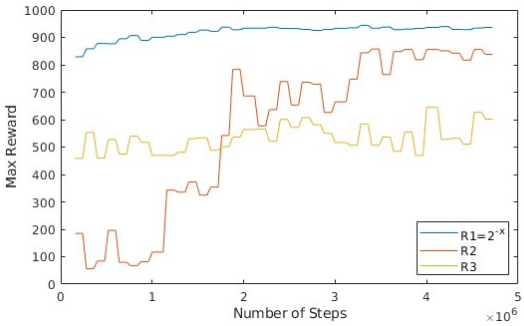

Figure 4: Fixed object and target location - Max reward comparison-2

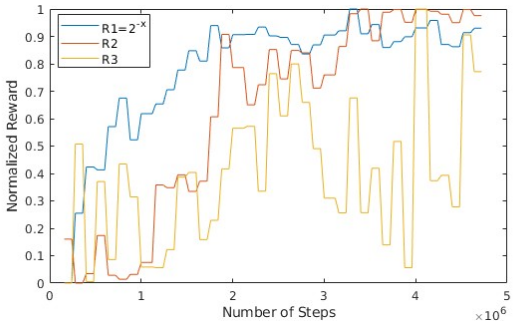

Figure 5: Fixed object and target location - Normalized reward comparison-2

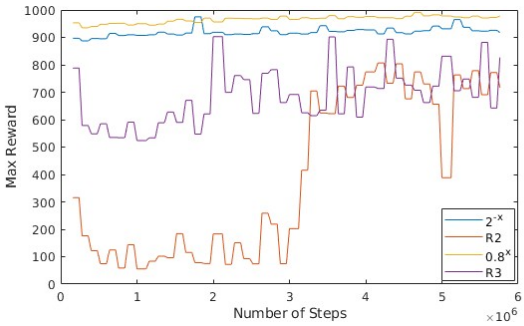

Figure 6: Fixed object location and random target location - Max reward comparison

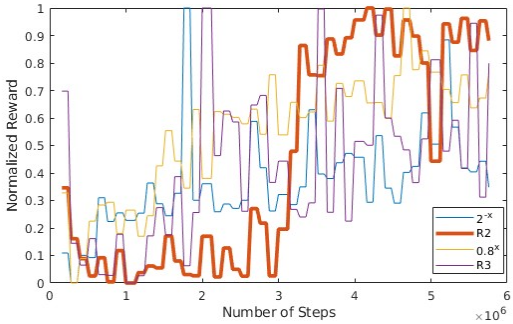

Figure 7: Fixed object location and random target location - Normalized reward comparison

four reward functions, $R1 = 2^{-x}$,

$$R2 = \begin{cases} 2^{-x} & 0 \leq x \leq 0.07 \\ 0 & otherwise \end{cases}$$

, $R4 = 0.8^x$, and

$$R3 = \begin{cases} 2^{-x} & 0 \leq x \leq 0.07 \\ 0.8^x - 0.5 & otherwise \end{cases}$$

We decided to keep $R3$ despite its learning curve in the previous training process is not ideal and appears too noisy compared to the other ones, but because of the complexity of the task, training with $R2$ could be a tougher process for the robot to learn.

In **figure 7**, $R2$ is still the most ideal reward function so far. We observed that once the robot arm has explored enough, it will quickly act in return for the highest reward, which is only within 0.07m of the distance term. In that way, the robot arm becomes robust at reaching the box and the target in a time-efficient way. For $R3$, the issue is its learning curve is too noisy, similar to the previous training process. In testing, the robot arm shows motivation to reach the target in only some of the trials. $R = 2^{-x}$ and $R = 0.8^x$ in general show a good learning curve but the performance is not as robust as $R2$.

6 Training with random object location and random target location

We changed the object(box) location to random and within the same range as the target location. The x, y coordinates of the box are randomly generated from $[-0.55, -0.25]$ & $[0.25, 0.55]$ respectively, and z is fixed at 0.03m. We selected three reward functions: $R = 2^{-x}$,

$$R2 = \begin{cases} 2^{-x} & 0 \leq x \leq 0.07 \\ 0 & otherwise \end{cases}$$

and $R = 0.8^x$. We decided to exclude $R3$ due to its noise.

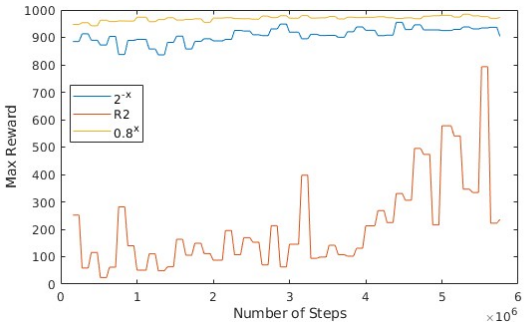

Figure 8: Random object location and random target location - Max reward comparison

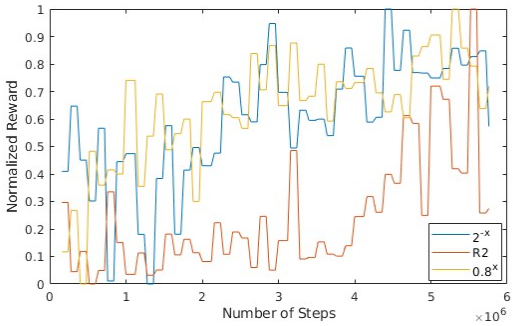

Figure 9: Random object location and random target location - Normalized reward comparison

In **figure 9**, we observed that on the contrary, $R2$ is not the most ideal reward function for this task, $R = 2^{-x}$ and $R = 0.8^x$ are performing significantly better in normalized reward comparison. In testing, the robot arm trained with $R = 2^{-x}$ and $R = 0.8^x$ shows approximately similar motivation to reach the target and robustness to reach towards a random location in each trial. The robot arm trained with $R2$ shows some motivation to reach but not in all trials. However, the motivation to complete the whole task is not as high as in previous tasks, due to the complexity of this task.

7 Limitation and Future Works

One major issue we noticed throughout all the testing process is that the robot arm is reluctant to finish the task of dropping the box to the target location. It always has a higher motivation to pick up the box, but once that's done, it shows a overall decreasing motivation to drop the box. One solution is to set two different rewards, $reward\_pick\_up$, the distance between the end effector and the box, and $reward\_drop\_off$, the distance between the end effector and the target location. While we keep track of two distance-terms throughout the training process, we set the $reward\_drop\_off$ slightly higher, for example:

$$Reward\_pick\_up = \begin{cases} 2^{-x} & 0 \leq x \leq 0.07 \\ 0 & otherwise \end{cases}$$

$$Reward\_drop\_off = \begin{cases} 2^{-0.5 \times x} & 0 \leq x \leq 0.07 \\ 0 & otherwise \end{cases}$$

$$Final\_reward = Reward\_pick\_up + Reward\_drop\_off$$

This approach is tested but the result is not ideal. The learning curves are highly noisy and show no learning most of the time steps. In testing this approach shows little to no improvement compared to the previous training when the rewards are not separated.

Another approach is to add environment reward with an additional bonus to encourage extra exploration. Extra rewards such as after the box is dropped off at the target location, an additional reward will be added to the cumulative rewards. We could also take entropy into consideration. A higher entropy indicates a higher uncertainty in the policy, causing the agent to explore more instead of always choosing certain actions.

References

[1] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.