PlaceSphere-v1: A novel sphere manipulation task built on Maniskill3 and solvable via PPO

Nan Xiao
Department of Electrical and Computer Engineering
A69027384

Abstract

This project aims at exploring and utilizing Maniskill [1] platform to build our own robot manipulation tasks and verify the solvability of it via RL. We built a novel manipulation tasks: PlaceSphere-v1, and solved it using Proximal Policy Optimization (PPO) algorithm within 50 million steps. We studied the environments by varying the environmental parameters including the radius of the sphere and the design of the reward function, and see how such changes can affect the performance in terms of success rate and the convergence speed.

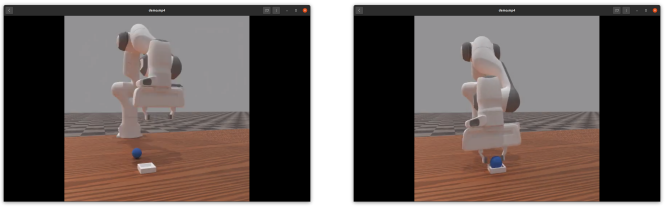
1 Description of the environments built

1.1 PlaceSphere-v1

The goal for the task is to grasp a sphere and place that on top of a bin using the Panda robot arms.

1.1.1 Reset and solution state

Figures 1(a) and 1(b) give the images corresponding to the reset state and the solution state of the task, respectively. At the reset state, the sphere and the bin are static, and the robot arm is in its default position. At the solution state, the sphere is placed on top of the bin without moving and the robot arm is not grasping anything.



((a)) Reset state ((b)) Solution state

Figure 1: Reset and solution state visualizations

1.1.2 Initializations and randomizations

The sphere's xy position is randomized on top of a table in the region [-0.1, -0.05] x [-0.1, 0.1]. It is placed at the first $\frac{1}{4}$ zone along the x-axis. The bin's xy position is randomized on top of a table in the region [0.0, 1.0] x [-0.1, 0.1]. It is placed at the last $\frac{1}{2}$ zone along the x-axis so that it doesn't collide the sphere. The robot arm is set at its default position.

1.1.3 observations and evaluations

We simply use the existing observations from the cube, including its pose (xyz position and rotation quaternion), linear, and angular velocities, to design the staged rewards and to check the success conditions. We also use the agent's tcp pose and the bin's pose to determine the goal site. We further use the gripper pose (qpos) design continuous and dense rewards for releasing the sphere from the grippers.

We evaluate the object and the goal reaching conditions to provide flags to supervise the reward generation. We evaluate whether an object is on the bin by checking if the xy-distance between the object and the bin is less than a small threshold, and checking if the z-distance is closed to the bin's half size plus the sphere radius enough. We check whether the object is static by checking whether its angular and linear velocities are approaching 0.

1.1.4 Success and failure conditions

To reach the successful state, we need to check 3 conditions: Firstly, the agent should not be grasping the sphere, indicating the sphere is placed on somewhere. Secondly, the object should be placed on the bin, that is, the xy-distance to the bin should approach 0 and the z-distance to that should approach the radius plus the half size of the shortest edge of the bottom bin block. Thirdly, the sphere should remain static on the bin. That is, its linear and angular velocity should be kept approximately 0.

2 Solvability via PPO

2.1 PlaceSphere-v1

2.1.1 Reward design

The reward computation involves several components: Firstly, we drive the gripper to reach the sphere by designing a reaching reward based on the distance between the end effector and the object. We normalized the value to the range [0, 1] by using tanh function for mapping and subtracting from 1. Secondly, we assign a grasp and distance-based place reward only to those environments where the object is grasped so that the gripper can grab the sphere and move to the bin top, and we similarly normalized the range to [0, 1]. Thirdly, we award ungrasping environments with the ungrasping and the static rewards: The ungrasping reward is characterized by the fraction between the gripper's current open width and the gripper's maximal open width, which is continuous and hence can encourage the gripper to open. The static reward involves awarding the agent using the norm of the linear and angular velocities of the object subtracting from 1, so that the robot can be encouraged to keep the sphere static.

2.1.2 PPO solution

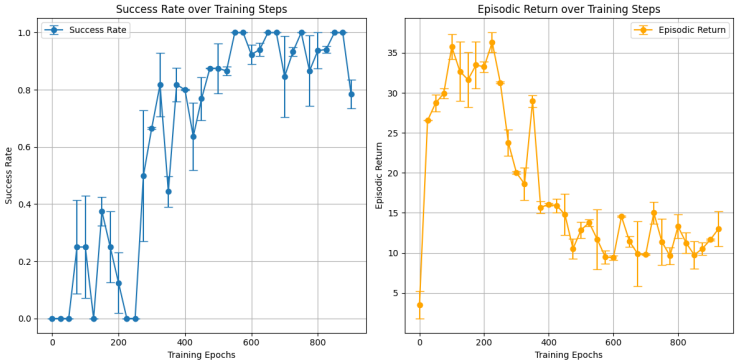


Figure 2: Success rate curve (trained over 50 million steps)

Figure 2 gives the success rate as well as the episodic return curve of solving the task PlaceSphere-v1 via PPO. We can see that the final success rate reaches around 0.8, indicating that the task is solved. The radius of the sphere is 0.020, which is the same as the inner side length of the bin. The number of steps used is 50 million, which indicates a reasonable amount of time.

We trained more steps (around 100 million steps) but turned out to find that keeping the success rate 1 is hard. There are always episodes where the task doesn't succeed even if we trained a lot more epochs. To investigate further reasons, we printed out the success conditions term-by-term during evaluation, and see which among them are satisfied under the non-one success rate trajectories and which are not. We observe that the most difficult success criteria to reach is the static final state of the sphere: The gripper can only grasp the sphere in an unstable way where the sphere is about to fall anytime it is being grabbed by the gripper. In this sense, after the gripper placed the sphere inside the bin, the sphere keeps spinning with some angular velocity, making it hard to succeed in keeping static. Hence, it seems the agent has difficulty learning keeping the sphere static in the end even after many millions of steps of training.

3 Environment design study

3.1 PlaceSphere-v1

We explored how changing the sphere radius can affect the task difficulty and the success rates. We vary the radius ranging from 0.015 to 0.030, and see how the resulting success rate curve will be.

3.1.1 Change of sphere radius

Figure 3 gives the comparisons of the success rates under different sphere radius used to solve the same task, that is, to place the sphere on top of the bin. The bin's inner square side length is 0.020. In terms of the reward design, we also used different distance functions to map the distance to the normalized [0, 1] range and convert it to the reaching reward term.

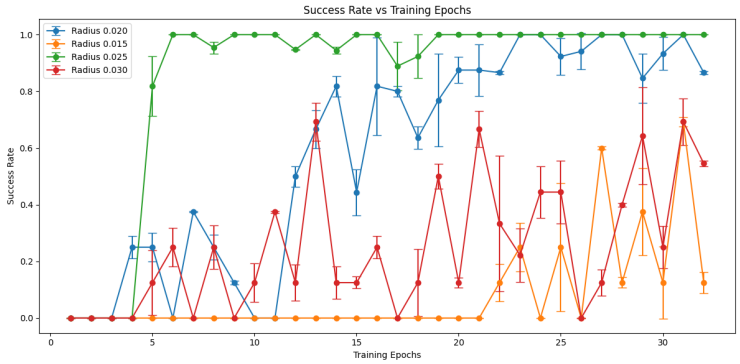
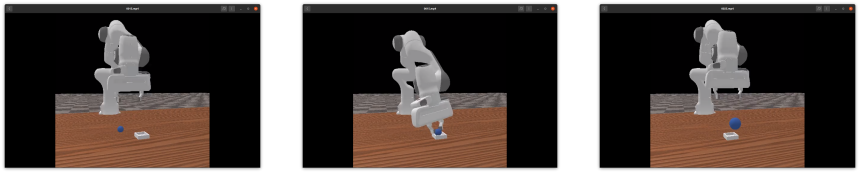


Figure 3: Curves with different sphere radius (all trained over 50 million steps)

We can see that the case with sphere radius = 0.025 reaches the optimal performance, whose value is a bit larger than the inner square radius. A potential reason is that such sphere can remain static more easily at the top of the bin without rolling out of the bin or rolling around inside the bin. It can stuck to the bin because of the comparatively larger radius. The performance under the radius to be 0.015 has the lowest overall success rate among them. Potentially it is because when the radius is too small, the sphere can roll around inside the bin more easily since the inner side length is larger, making it hard to keep the sphere static at the end state when the gripper placed that on top of the bin. The performance when the radius equals 0.030, which is much larger than the bin's inner side length, turns out to converge harder than the 0.020 and 0.025 cases, and is almost as hard to converge as the 0.015 case. A potential reason is that the sphere is easy to roll out of the bin given the large radius it has and the unstable final state (non-zero angular velocity) it may have in the end. The agent needs more epochs to learn placing the sphere statically on the bin under those scenarios.

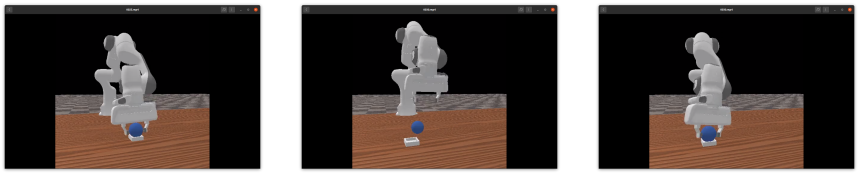
Figure 4 further visualizes the reset and end states for the environments with different radius. It shows visually how such rollings can take place due to the differences between the sphere radius and the inner bin side length.



((a)) Reset state for radius=0.015

((b)) End state for radius=0.015

((c)) Reset state for radius=0.025



((d)) End state for radius=0.025

((e)) Reset state for radius=0.030

((f)) End state for radius=0.030

Figure 4: Reset and solution state visualizations for different radius

In a word, a radius that is either too large or too small both can make the task difficult to solve. To make the task solvable, we should choose a radius that can slightly exceed the inner side length of the bin (for example, choosing 0.025, which is just a bit larger than the inner side length 0.020 and not larger than 0.030), so that the sphere can keep static in the end state without rolling either inside or out of the bin easily.

3.1.2 Change of the distance function

Figure 5 gives the success rate curves corresponding to using the Manhattan and the L1-norm distance as the reaching reward functions in the dense reward design, respectively. We can see that while the success rates under the Manhattan distance function are larger than that under the L1-norm, the difference is not significant. It seems both distance function, as long as it can penalize a large distance to the goal site by assigning a smaller reward to a larger distance position, can help the training converge with a similar speed. It also shows that the scale of the reward (that is, how large the reward gap is when the agent is taking a step closer to the goal) doesn't affect the convergence speed much. As long as there is a right gap (incremental when the agent takes a step closer to the goal, or decremental when taking a step further from the goal) between the rewards given by the agent before and after taking a step, the agent's convergance will not be affected by the scale of the reward. This also shows that we can normalize our dense rewards to make them more interpretable without affecting the convergence.

Figure 5: Curves with different distance functions (all trained over 50 million steps)

4 Conclusions

In conclusion, we built the novel PlaceSphere-v1 task where the goal is to place a sphere on top of a bin. We verified the solvability under PPO within 50 million training steps by showing the task success rates curve during training. We investigated the environment design by varying the radius of the sphere, and discovered that a radius that is slightly larger than the inner bin side length is ideal for making the task solvable. We also investigated the choice of distance function for the reach reward designs, and found that the reward scales don't affect the convergence speed for this task.

References

[1] Gu, J., Xiang, F., Li, X., Ling, Z., Liu, X., Mu, T., Tang, Y., Tao, S., Wei, X., Yao, Y., Yuan, X., Xie, P., Huang, Z., Chen, R., and Su, H. (2023). ManiSkill2: A Unified Benchmark for Generalizable Manipulation Skills. arXiv preprint arXiv:2302.04659.