Real Robot Challenge Stage 1

Team: **Three Wolves**

Abstract

Deep reinforcement learning has shown its advantages in real-time decision-making based on the state of the agent. In this stage, we solved the task of using a real robot to manipulate the cube to a given trajectory. The task is broken down into different procedures and we propose a hierarchical structure, the high-level deep reinforcement learning model selects appropriate contact positions and the low-level control module performs the position control under the corresponding trajectory. Our framework reduces the disadvantage of low sample efficiency of deep reinforcement learning and lacking adaptability of traditional robot control methods. Our algorithm is trained in simulation and migrated to reality without fine-tuning. The experimental results show the effectiveness of our method both simulation and reality.

1 Introduction

Our method relies on two points:

(1) Manually set contact points is impossible to following different goal trajectory while grasping object stably. Such, we introduce the reinforcement learning to generate appreciate contact points by observing the object state and next goal position.

(2) Classic robot control method performs stably and accurately in the task of joint position control and trajectory following.

Therefore, we use a hierarchical control framework that utilize reinforcement learning as high-level planner and classic control methods as low-level controller. We broke down the task of grasping and moving the cube in given trajectory into three primitives: selecting three suitable contact points, moving the tip to the contact points of the cube, and finally lifting the cube to the given trajectory.

2 Method

In this report, we complete the task with a reinforcement learning method and PD position controller. Our framework is shown in Fig .1. Reinforcement learning selects appropriate contact points, and the control algorithm completes the

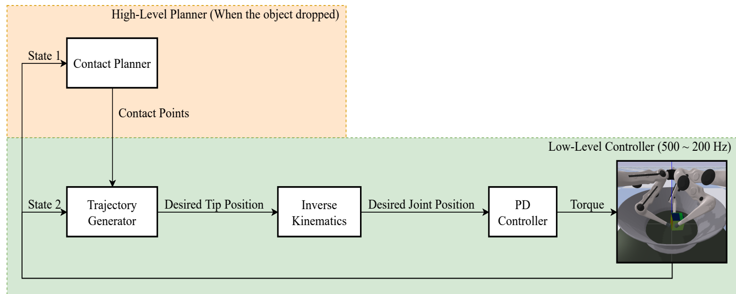
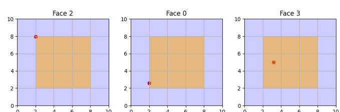
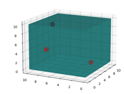


Figure 1: The overall framework of our approach



(a) 2D diagram



(b) 3D sketch

Figure 2: 2D and 3D sketch of the selected contact position, the orange area in the first diagram means the available contact area to select

movement according to the generated trajectory. Considering the safe contact points, we set the available contact area in 60% of one cube face from the center, as shown in Fig. 2, and corresponding expected contact positions are generated by the reinforcement learning and executed by a low-level PD controller.

The framework that we use is SAC[4]. In the environment of move the cube on a given trajectory, the robot need to move the target object to the desired position fast and stably. When a reasonable contact point is selected, the subsequent control algorithm is more likely to reach the target point stably. On the contrary, if an unreasonable contact point is selected, it is difficult for the low-level controller algorithm to move the block to the target point, so we use the distance between the block and the trajectory as a reward to help the agent learn to choose the right contact point.

$$r = 0.001 \exp(-300 \left\| p_{goal} - p_{cube} \right\|^2) \tag{1}$$

We generate the corresponding trajectory according to the current fingertip position and the target point. In order to keep the movement stable, we adopted fifth-order polynomials[8].

The robot is asked to avoid a discontinuous jump in acceleration at both $t = 0$ and $t = T$. Our solution limits the terminal position, velocity, and acceleration, but adding these constraints to the problem formulation requires the addition of design freedoms in the polynomial, yielding a quintic polynomial of time, $s(t) = a_0 + \cdots + a_5 t^5$. We can use the six terminal position, velocity, and acceleration constraints to solve uniquely for $s(0) = \dot{s}(0) = \ddot{s}(0) = \dot{s}(T) = \ddot{s}(T) = 0$ and $s(T) = n$, which yields a smooth motion. In the final step, the PD controller and inverse dynamics are used to keep the corresponding fingertips to the desired trajectory.

3 Result and Discussion

We test the performance of our method by training in a simulation environment and running multiple experiments in different situations in the simulation and real systems. In the results of the experiments, our method is compatible with both stability and adaptability as shown in Fig. 3.

At the same time, due to the addition of fifth-order polynomials, the movement of the square is relatively smooth and no jitters. At the same time, high-level reinforcement learning can select contact points suitable for the current situation according to different positions and states, which enables the model to have stronger adaptability. Moreover, this hierarchical control method reduces the need for training samples and training time in the training process and alleviates the sim-real problems. Our method demonstrates the potential of reinforcement learning and control methods in robotic tasks to a certain extent.

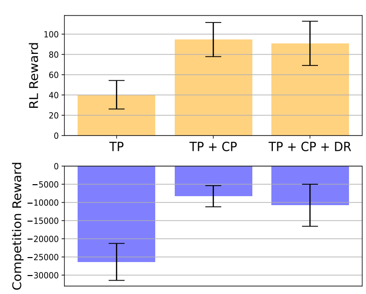


Figure 3: Rewards comparison of different policies. From left to right: Trajectory Planning(TP) only, TP + Contact Planning(CP), TP+CP+Domain Randomization(DR). Top bar-graph shows the reward for RL training, and bottom one shows the reward of the contest

The next step of the research needs to consider the challenges that the algorithm faces small. The algorithm needs to maintain the awareness of environmental changes, such as how to avoid collisions with the dice that have been arranged and complete high-level planning. We plan to introduce world models[3, 5, 6] to increase the ability of the agent to control the environment. When there is a big difference between simulation and reality, we will also add the sim-real algorithm[9, 1, 2, 7].

Video Attachments Playlist: https://youtu.be/Jr176xsn9wg

References

[1] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[2] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979. IEEE, 2019.

[3] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[4] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018.

[5] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

[6] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

[7] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12627–12637, 2019.

[8] Kevin M Lynch and Frank C Park. *Modern robotics.* Cambridge University Press, 2017.

[9] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018.