Soft Object Dexterous Manipulation Using Deep Reinforcement Learning

Sornsiri Promma[1], Sakmongkon Chumkamon[1], and Eiji Hayashi[1]
[1]Department of Mechanical Information Science and Technology, Kyushu Institute of Technology, 680-4 Kawazu, Iizuka, Fukuoka 820-8502, Japan

Abbe Mowshowitz[2]
[2]Department of Computer Science, The City College of New York, 160 Convent Avenue, New York, NY 10031, USA

E-mail: promma.sornsiri363@mail.kyutech.jp, m-san@mmcs.mse.kyutech.ac.jp, haya@mse.kyutech.ac.jp, amowshowitz@ccny.cuny.edu
http://www.kyutech.ac.jp/

Abstract

Manipulation of objects is one of the basic tasks that has been studied for a long time in the robotic field. Many experiments were setting the environment and using Deep Reinforcement Learning to train robot arm to grasp various objects. Still, most of those objects are solid objects, whereas nowadays, robot arms are used for grasping soft objects as well. In this study, we develop object manipulation tasks in pybullet simulation, focusing on soft objects by using Deep Reinforcement Learning (DRL) based on Soft Actor-Critic and Proximal Policy Optimization algorithm which aims to make the robot able to grasp soft objects in the exact position with proper force that does not damage them.

Keywords: Object manipulation, Soft Object, Deep Reinforcement Learning (DRL), Soft Actor-Critic, Proximal Policy Optimization

1. Introduction

Manipulation of objects is one of the basic tasks that has been studied for a long time in the robotic field. For humans, it is a simple movement that we can do immediately without thinking, whereas, it is quite challenging for the robot to grasp the objects without making them fall. To teach the robot, one of the effective ways that have been widely used nowadays is Deep Reinforcement Learning, using a robot arm to try to grasp objects repeatedly.

Many papers have succeeded in using robot arms to grasp different shapes of objects, focusing on solid ones. On the contrary, soft-body objects are rarely studied and are more challenging. The main factor when grasping solid objects mostly depended on precise position only, whereas, grasping the solid object needs to consider proper force, elasticity, and the shape both before and after touching them.

So, in this study, we develop object manipulation tasks in pybullet simulation, focusing on soft objects by using Deep Reinforcement Learning (DRL) based on Soft Actor-Critic and Proximal Policy Optimization algorithm which aims to make the robot able to grasp soft objects in the exact position with proper force that does not damage them.

2. System Overview

2.1. *Simulation platform*

The simulation platform that we chose for this research is pybullet. Pybullet is an open-source physics simulation that is widely used in the robotic field. It is easy to use and integrated with Reinforcement Learning such as Stable Baseline which is also implemented on pybullet. Moreover, not only is it able to import various formats of mesh and articulated bodies, it can import those as soft objects with high accuracy of collision as well.

2.2. *Environment Configuration*

In pybullet, we are setting the environment that will be used for training Deep Reinforcement Learning, using Open AI gym, a python API that is used for developing Reinforcement Learning agents. In this paper, for the agent, the robot arm that we are using is Xarm7 [1], a 7-axis robotic arm with a pair of grippers, implemented in pybullet, as we can see in Fig 1. Its movement will be calculated as inverse kinematics that receive the position from the DRL model. For soft object, we used mesh in VTK format from bullet3 library which is in a tube shape, so that the agent needs to grasp it in the proper position. After importing, soft object needs to be adjusted some parameters to make it has suitable properties such as not too soft or bounce too high.

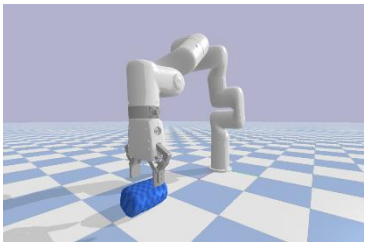

Fig. 1. Xarm7 with a pair of grippers and soft object in tube shape that is used for training in pybullet

2.3. *Deep Reinforcement Learning Model*

As described, to teach the robot to grasp soft object, we used Deep Reinforcement Learning (DRL), same as many papers that do object manipulation tasks with solid objects. DRL is not required to collect any data which is suitable for the task in the robotic field. Additionally, it can learn complex behaviors, taking action repeatedly based on observation to get the reward and evaluate them.

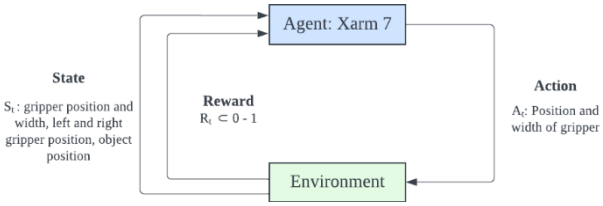

Fig. 2. Process of training Xarm7 to grasp soft object using Deep Reinforcement Learning

According to the Fig 2, in our experiment, the agent is Xarm7. The agent will take the action received from the model, which consists of the next position (x, y, z) and gripper width. After it performed, it will observe by get the value gripper position and width, left and right gripper position, plus gripper position. At the same time, it will calculate rewards as the result of the action, which detail will be described in the next section. Lastly, state and reward will be calculated together, so that model can determine the next step for the agent. This one-time process will be calculated as one step, and it will do repeatedly until the episode ends.

To make the robot arm knows its action result, we need to define a proper reward function. Shaped reward function is used in order to evaluate each episode, which is calculated from the equation below.

$$R_{total} = R_{distObjGoal} + R_{distObjGripper} + R_{gripper} \quad (1)$$

From the Eq (1), $R_{distObjGoal}$ is calculated from distance between object and goal in z-axis only. The higher it is, the more reward it gets. The maximum reward for $R_{distObjGoal}$ is when gripper grasps the object to 1 on z-axis. Next, $R_{distObjGripper}$ is calculated from distance between object and gripper in 3-axis (x, y, z) Agent will get more reward if the gripper is near the object. For $R_{gripper}$, this reward depends on the gripper state if it opens or closes in the right position. For example, when it is not in the grasping position, it will get higher reward if the gripper stays open. In contrast, if it is in a position of being able to grasp the object, it will get reward if it closes and touches the object. Finally, we normalize R_{total} to be in the range of 0 to 1.

The Deep Reinforcement Learning algorithms that we used for training the agent are originally from Stable Baseline3 [2], which are PPO and SAC. We chose these

two algorithms based on their types, which are on-policy and off-policy respectively so that we can compare the training result. Firstly, PPO, Proximal Policy Optimization algorithm is an on-policy model-free DRL algorithm that uses a policy gradient approach to optimize a policy by avoiding policy to not change too drastically from one iteration to the next which will make training more stable. To prevent this, it uses clipped objective functions to keep the difference between the current policy and the old one [3]. On-policy DRL algorithm concept such as PPO can be seen in Fig 3.

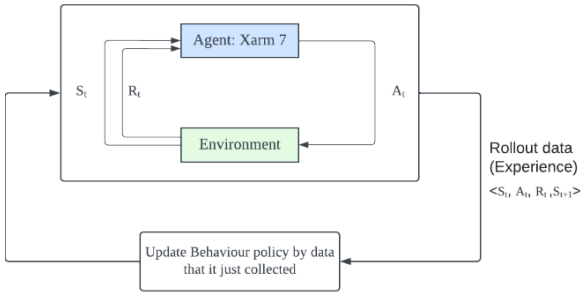

Fig. 3. Illustration of on-policy algorithm process such as PPO

Second, SAC or Soft Actor-Critic, is an off-policy model-free DRL algorithm which overall concept can be seen in Fig 4. Normally, many DRL algorithms will focus on maximizing reward function only, while SAC focuses on maximizing the entropy term as well[4]. It is the algorithm that will trade-off between explore and exploit which is suitable for the real-world task, plus, it can work on continuous action space. SAC can also reuse data that has been collected from other tasks, just adjusting some parameters and reward functions for the new task will make it able to learn faster.

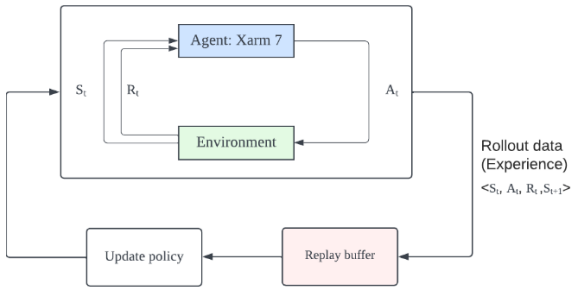

Fig. 4. Illustration of off-policy algorithm process such as SAC

3. Experiment and Result

We trained the agent 2 times using the process as mentioned in the previous section, the first time training with the PPO algorithm and the second time training with SAC algorithm. Moreover, we also presented other 2 results in the case of changing the observation space into the image and the result of using the grasping position to be the starting position.

3.1. *Training with PPO algorithm*

Firstly, agent, Xarm7, will take the action which is position and gripper width. The position will be normalized to be in the range [-1, 1] before the agent does the action. After that, it will observe the environment by using the gripper position, gripper width, left and right gripper position, plus the current object position. Only gripper width will be normalized to be in the range [0, 1] All of the observations will be used to calculate the reward using shaped reward function that has been described in above section. After that, it will update the policy before doing the next episode. Each episode contains 3,000 timesteps. Termination can be decided by 3 factors

- Reach 3,000 timesteps (reward -100)
- Object reach the goal (reward +1,000)
- Soft body crash (reward -100)

For the last factor, sometimes, soft object can crash by the gripper putting too much force on it which makes pybullet not able to calculate its physics. To test the environment and model, we use PPO as the DRL algorithm to test the result.

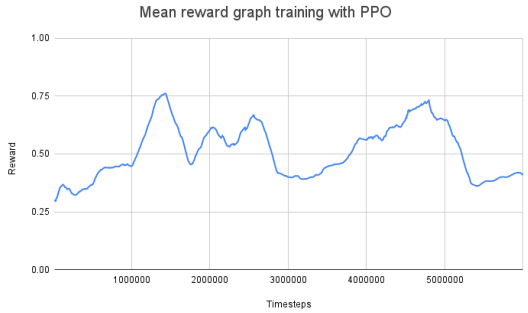

Fig. 5. Mean reward per timesteps graph, training with PPO

From Fig 5, we can see that the environment is basically working. Mean reward gradually increases while as decreases at some point such as when it reached 1.5 million timesteps or 2.5 million timesteps. After we

have tested and observed, we noticed some interesting results.

- If it is able to grasp the object, the mean reward will be around 0.7
- From the testing result, it can stay at the grasping position perfectly when it reached 2.5 million timesteps.
- Training object manipulation task for soft object is slower than training with solid object around 35%

Still, even if the agent can reach the grasping position, it does not grasp the object up to the given position. We also continue further around 2 million timesteps but the result is not better.

3.2. *Training with SAC algorithm*

Training with SAC process is different from PPO in some parts, in order to make it faster for training.

- Move the starting position closer to the object.
- Change timesteps per episode, from 3,000 timesteps to 1,000 timesteps.
- Reduce terminal reward due to fewer timesteps per episode.
- Adjust clipping reward range due to less distance of starting position.

With nearer distance, agent training with SAC can reach grasping position in 300,000 timesteps, with mean reward of around 0.35. After that, as we can see in Fig 6, the mean reward sharply reduces after training around 1.3 million timesteps. It tries to grasp and move the object to various positions. Still, sometimes it is moving too much which damage the soft object and gets much minus termination reward, leading to the gripper trying to avoid the grasping position and move to the opposite side. So, there is a possibility that SAC exploitability is not suitable for object manipulation task for soft object.

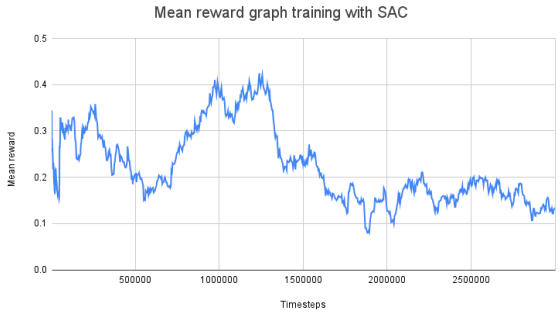

Fig. 6. Mean reward per timesteps graph, training with SAC

3.3. *Training by using image as the observation, with SAC algorithm*

Not only training with normal observation like using the current position of the gripper or object, but we also use a gym wrapper to change the observation space into the array of images. We have tested this environment with SAC algorithm and changed the policy from MlpPolicy to CnnPolicy to extract features from the image. This agent has been trained after changing some parts as mentioned in the previous section.

The result, as in Fig 7, is not different from training with the normal one. Still, one noticeable point is that using image observation for training takes 2 times longer than training with normal observation space.

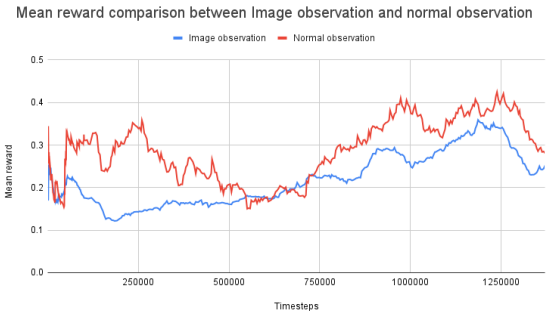

Fig. 7. Mean reward per timesteps comparison graph between image observation and normal observation

3.4. *Training with PPO algorithm by adjusting the starting position*

Lastly, to ensure that it is able to grasp the soft object by mainly considering only the gripper width, we trained the agent by changing the starting position, as we can see from Fig 8, the gripper stays at the grasping position from the beginning. We trained for 400,000 timesteps with PPO algorithm, using the same configuration with SAC.

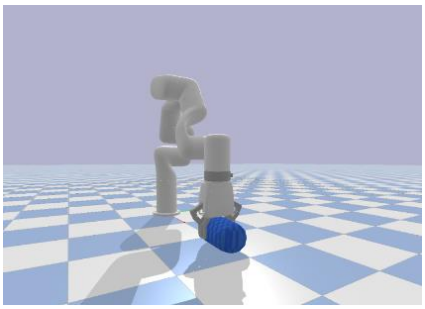

Fig. 8. New starting position

The result, according to Fig 9, is the agent can grasp the object and hold it in the air when it reached only 50,000 timesteps. It adjusts the gripper width correctly to hold the object, not too far to make the object fall and not too close to damage the soft object.

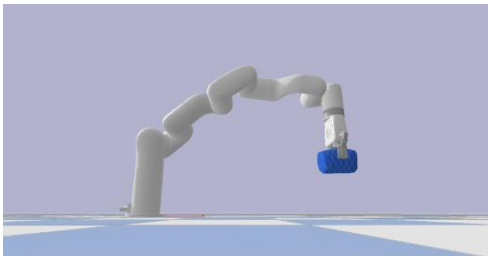

Fig. 9. Agent can grasp soft object into the air after training around 50,000 timesteps

4. Conclusion

In this research, we developed object manipulation task by focusing on soft object. The results show that the agent can hold soft object without damaging it if the gripper is in the correct position. Still, if the starting position is far from the object, it needs some period of time for training which is longer than training with solid object due to soft object physic calculation. Furthermore, using SAC algorithm for soft object manipulation task, the result is not good enough which the reason can be that its exploitability leads to action that harm soft object or the model needs to tune more hyperparameter to get the better result.

References

1. Pan, C. (2022). Gym implementation for Xarm. [online] GitHub. Available at: https://github.com/jc-bao/gym-xarm [Accessed 15 Dec. 2022].
2. stable-baselines3.readthedocs.io. (n.d.). Stable-Baselines3 Docs - Reliable Reinforcement Learning Implementations — Stable Baselines3 1.7.0a5 documentation. [online] Available at: https://stable-baselines3.readthedocs.io/en/master [Accessed 26 May. 2022].
3. huggingface.co. (n.d.). *Proximal Policy Optimization (PPO)*. [online] Available at: https://huggingface.co/blog/deep-rl-ppo [Accessed 1 Dec. 2022].
4. Yazici, B. (2021). Sample Efficient Robot Training on Pybullet Simulation with SAC Algorithm. [online] Medium. Available at: https://towardsdatascience.com/sample-efficient-robot-training-on-pybullet-simulation-with-sac-algorithm-71d5d1d4587f [Accessed 5 Dec. 2022].

Authors Introduction

Sornsiri Promma

She received bachelor degree in 2022 from Computer Engineering, King Mongkut's University of Technology Thonburi, Thailand. She is currently a Master student at Kyushu Institute of Technology and conducts research at Hayashi Laboratory.

Dr. Sakmongkon Chumkamon

Dr. Sakmongkon Chumkamon received Doctor of Engineering degree from Kyushu Institute of Technology in 2017. He was a postdoctoral researcher at Guangdong University of Technology in 2017-2019. Presently he is a postdoctoral researcher in Kyushu Institute of Technology since 2019. His research interests include factory automation robots and social robots.

Prof. Eiji Hayashi

Prof. Eiji Hayashi is a professor in the Department of Intelligent and Control Systems at Kyushu Institute of Technology. He received the Ph.D. (Dr. Eng.) degree from Waseda University in 1996. His research interests include Intelligent mechanics, Mechanical systems and Perceptual information processing. He is a member of The Institute of Electrical and Electronics Engineers (IEEE) and The Japan Society of Mechanical Engineers (JSME).

Prof. Abbe Mowshowitz

Prof. Abbe Mowshowitz received the Ph.D. degree from University of Michigan in 1967. He has been professor of computer science at the City College of New York and member of the doctoral faculty at the Graduate Center of the City University of New York since 1984. His current research interests lie in two areas are organizational and managerial issues in computing, and network science. In addition to teaching and research, He has acted as consultant on the uses and impacts of information technology (especially computer networks) to a wide range of public and private organizations in North America and Europe.