Deep RL Arm Manipulation Project

Tan Kai Xiong

Abstract—In this project, a deep reinforcement learning neural network to map the raw pixels from a camera to the robot arm control commands for object manipulation. This end-to-end approach means that with minimum training data from human. A reward functions is required to feedback the neural network. A 3-DoF robotics arm is simulated in Gazebo. Implementation of gazebo plugin for the arm, operating via a C++ API for the popular PyTorch library for deep learning frameworks.

Index Terms—Robot, IEEEtran, Udacity, LaTeX, reinforcement learning.

1 Introduction

THE introduction of end-to-end approach revealed a new horizon to resolve challenging robotics problems. To solve a 3-DoF robotics arm kinematic control for object manipulation problem using Deep Q-learning network.

2 Gazebo Setup

3-DoF robotics arm is one of the common arm that is deployed in the industrial applications.

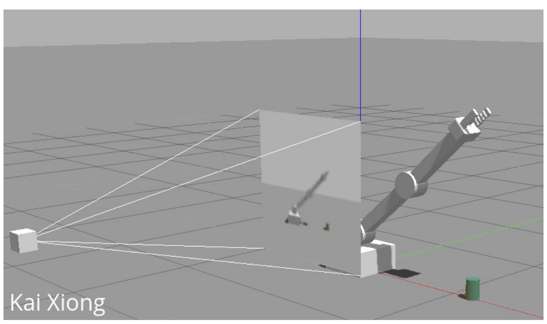

Fig. 1. Gazebo 3-DoF robotic arm.

3 Control Setup

In this project, there are two options available for the robot arm kinematic control. They are positional control and velocity control. Positional control is much more simpler to use.

TABLE 1
Discrete Action Encoding

Index	Action
0	J0 -
1	J0 +
2	J1 -
3	J1 +
4	J2 +
5	J2 -

4 Reward Function

Deep Q-Network output is usually mapped to a particular action. The rewards system is designed to train the robot arm to touch the object of interest in one attempt. Each episode is limited to a certain number of attempts, penalty will be issued if maximum length of the episode is reached without achieving the objective. Interim reward or penalty will be issued while the robot arm is moving based on the distance from the object of interest. If the robot arm touch the object of interest, it will issue a win reward and episode is ended.

Event	Rewards
Ground Collision	-1.0
Object Collision	-1.0
End of Episode	-1.0
Success	1.0
Interim Rewards	5.0 x d
Interim Distance Smoothing Factor	0.22

TABLE 2
Rewards Function

5 Hyper-parameters

The final hyper-parameters configuration is shown in table 3.

TABLE 3
Training Hyper-parameters

Parameter	Value	Description
Channel	3	Input Image Channels (RGB)
Batch size	32	Input data batch size
Replay memory	10000	Size of Memory buffer for experience replay
Optimizer	Adam	DQN Network Loss Optimizer
Learning rate	0.005	Network Learning Rate
LSTM	false	Flag to enable Recurrent Network (LSTM)
Delta	0.096	Joint angle increment per step in rad (5.5 deg)
EPS decay	400	Epsilon delay steps

5.1 Input Dimensions

The default input width and height parameters are 512x512. It is reduced to 128x128, as the default size was excessively large and did not provide a significant improvement. With this changes, it able to reduce the GPU memory used in training.

5.2 Optimizer

The Adam optimizer is widely used in vast number of different domains and the general robustness to un-tuned parameters.

5.3 Learning Rate

The initial guess for the learning rate is 0.01, a lower learning rate will prevent the network to reach plateau.

6 Results

6.1 Task 1

As for task 1, the robot arm need to the touch the object with at least a 90% accuracy for a minimum of 100 runs. After 130 iterations, it manage to achieve more than 90% accuracy. To watch the youtube video demonstration task 1.

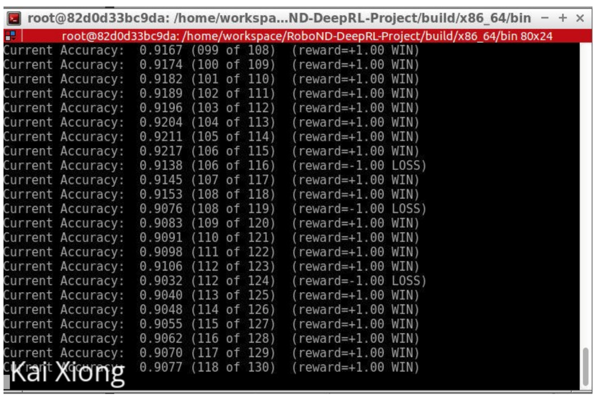

Fig. 2. Snapshot log of task 1 during execution.

6.2 Task 2

As for task 2, the robot arm's gripper base need to touch the object with at least a 80% accuracy for a minimum of 100 runs. After 500 iterations, it manage to achieve more than 80%. To watch the youtube video demonstration task 2.

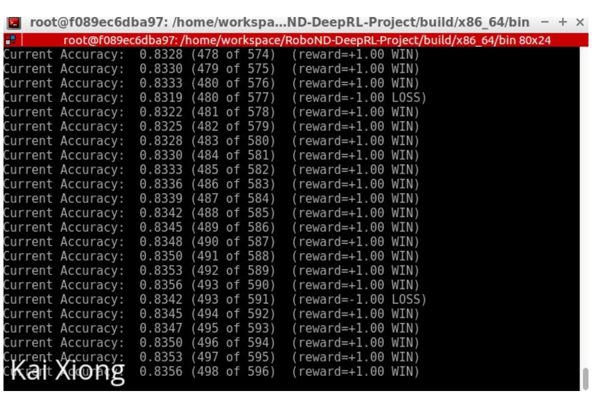

Fig. 3. Snapshot log of task 2 during execution..

7 Conclusion / Future work

The DQN agent is capable of achieve compelling performance on two of the tasks as demonstrated above. The movement generated by the DQN network is not smooth. If the object of interest is not located in the image, the decision of the DQN network at that instant will become non-deterministic and may result in unstable oscillation. To avoid this scenario, the objects need to spawn in the locations where camera visibility and reachable are both guaranteed.