

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, L^AT_EX, 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.

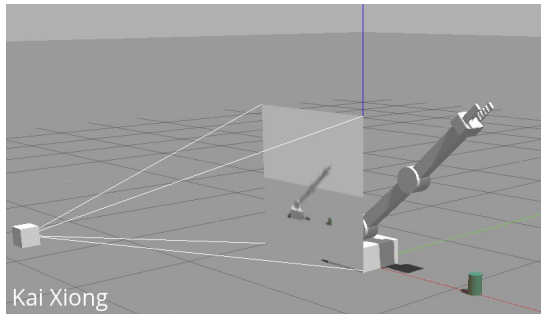


Fig. 1. Gazebo 3-DoF robotic arm.

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](#).

```

root@82d0d33bc9da: /home/workspa...ND-DeepRL-Project/build/x86_64/bin - + x
root@82d0d33bc9da: /home/workspa/RoboND-DeepRL-Project/build/x86_64/bin 80x24
Current Accuracy: 0.9167 (899 of 108) (reward=+1.00 WIN)
Current Accuracy: 0.9174 (100 of 109) (reward=+1.00 WIN)
Current Accuracy: 0.9182 (101 of 110) (reward=+1.00 WIN)
Current Accuracy: 0.9189 (102 of 111) (reward=+1.00 WIN)
Current Accuracy: 0.9196 (103 of 112) (reward=+1.00 WIN)
Current Accuracy: 0.9204 (104 of 113) (reward=+1.00 WIN)
Current Accuracy: 0.9211 (105 of 114) (reward=+1.00 WIN)
Current Accuracy: 0.9217 (106 of 115) (reward=+1.00 WIN)
Current Accuracy: 0.9138 (106 of 116) (reward=-1.00 LOSS)
Current Accuracy: 0.9145 (107 of 117) (reward=+1.00 WIN)
Current Accuracy: 0.9153 (108 of 118) (reward=+1.00 WIN)
Current Accuracy: 0.9076 (108 of 119) (reward=-1.00 LOSS)
Current Accuracy: 0.9083 (109 of 120) (reward=+1.00 WIN)
Current Accuracy: 0.9091 (110 of 121) (reward=+1.00 WIN)
Current Accuracy: 0.9098 (111 of 122) (reward=+1.00 WIN)
Current Accuracy: 0.9106 (112 of 123) (reward=+1.00 WIN)
Current Accuracy: 0.9032 (112 of 124) (reward=-1.00 LOSS)
Current Accuracy: 0.9049 (113 of 125) (reward=+1.00 WIN)
Current Accuracy: 0.9048 (114 of 126) (reward=+1.00 WIN)
Current Accuracy: 0.9055 (115 of 127) (reward=+1.00 WIN)
Current Accuracy: 0.9062 (116 of 128) (reward=+1.00 WIN)
Current Accuracy: 0.9070 (117 of 129) (reward=+1.00 WIN)
Current Accuracy: 0.9077 (118 of 130) (reward=+1.00 WIN)
Kai Xiong

```

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](#).

```

root@f089ec6dba97: /home/workspa...ND-DeepRL-Project/build/x86_64/bin - + x
root@f089ec6dba97: /home/workspa/RoboND-DeepRL-Project/build/x86_64/bin 80x24
Current Accuracy: 0.8328 (478 of 574) (reward=+1.00 WIN)
Current Accuracy: 0.8330 (479 of 575) (reward=+1.00 WIN)
Current Accuracy: 0.8333 (480 of 576) (reward=+1.00 WIN)
Current Accuracy: 0.8319 (480 of 577) (reward=-1.00 LOSS)
Current Accuracy: 0.8322 (481 of 578) (reward=+1.00 WIN)
Current Accuracy: 0.8325 (482 of 579) (reward=+1.00 WIN)
Current Accuracy: 0.8328 (483 of 580) (reward=+1.00 WIN)
Current Accuracy: 0.8330 (484 of 581) (reward=+1.00 WIN)
Current Accuracy: 0.8333 (485 of 582) (reward=+1.00 WIN)
Current Accuracy: 0.8336 (486 of 583) (reward=+1.00 WIN)
Current Accuracy: 0.8339 (487 of 584) (reward=+1.00 WIN)
Current Accuracy: 0.8342 (488 of 585) (reward=+1.00 WIN)
Current Accuracy: 0.8345 (489 of 586) (reward=+1.00 WIN)
Current Accuracy: 0.8348 (490 of 587) (reward=+1.00 WIN)
Current Accuracy: 0.8350 (491 of 588) (reward=+1.00 WIN)
Current Accuracy: 0.8353 (492 of 589) (reward=+1.00 WIN)
Current Accuracy: 0.8356 (493 of 590) (reward=+1.00 WIN)
Current Accuracy: 0.8342 (493 of 591) (reward=-1.00 LOSS)
Current Accuracy: 0.8345 (494 of 592) (reward=+1.00 WIN)
Current Accuracy: 0.8347 (495 of 593) (reward=+1.00 WIN)
Current Accuracy: 0.8350 (496 of 594) (reward=+1.00 WIN)
Current Accuracy: 0.8353 (497 of 595) (reward=+1.00 WIN)
Current Accuracy: 0.8356 (498 of 596) (reward=+1.00 WIN)
Kai Xiong

```

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.

What methods could you use to improve on the results you got?

- Implement Double Q-Learning
- Scaling rewards
- Resizing input dimensions

Would a different network configuration work better maybe?

- Yes, using a LSTM architecture. It will require a lot of training set.

Different types of control that you haven't tried yet but think could provide improvements?

- Velocity control will make the robot arm movement more unstable
- Positional control is much simpler to use and more stable