Deep RL Arm Manipulation

Abanob Effat

Abstract—This project aims to train a 3 degree of freedom arm manipulator by using RL agent so that the arm can achieve two tasks the first is over 90% accuracy of making the any part of the arm touch the can and the second task is to make only the gripper touch the can with accuracy over 80% .

Index Terms—Robot, IEEEtran, Udacity, LaTeX, Reinforcement Learning, RL, DEEP RL, DQN, jetson.

1 Introduction

THE objective is to get the robot to achieve a certain accuracy for each objective

1) Make any part of the arm touch the can. 90% accuracy.
2) Make only the gripper part of the arm to touch the can. 80%accuracy.

The jetson jetson-utils and jetson-reinforcement libraries are used to realize this project [?]. The Gazebo is the simulation environment for the robotic arm. There is a tube object, and the robot should learn to reach to this object.
The project defines several reward functions and tunes the parameters of the libraries.

2 Background

Build QDN agent to train the arm using several reward functions for each task and tune hyperparameters. Using libraries for the RL agent and gazebo for simulating the environment.

2.1 Reinforcement Learning

In reinforcement learning (RL), the agents tries to achieve the max reward by trying the best action. The rewards should be defined for the agents. They start learning with trial and error. As they learn, they can take logical actions [?].

2.2 Q-learning

In Q-learning, the agent learns a policy which contains best actions in the different states of the environment [?]. It is a model-free algorithm and does not need to know the every detail of the environment. It aims to build a table to store rewards for each state and action pair. At every time step, the agent observes it state and choose an action. Then it updates the reward value of that state, action pair.

2.3 Deep Reinforcement Learning

Deep reinforcement learning (DRL) is a special kind of reinforcement learning that uses artificial neural networks to train the agent. It replaces the processing sensor measurements step with an end-to-end approach. The network directly receive the sensor measurements and returns the best action to perform [?].

2.4 Deep Q Network

DQN trains an NN for the Q value function. It uses the following techniques to let the network converge faster and more reliably.

- The replay pool takes the average of the behavior distribution over previous states. It is essential for smoothing the learning and handling the noise. It provides the advantage of using every step more than once in the weight update process.
- The target network represents the Q function that is the function to compare when calculating the loss. Since the Q function values are changing during the training, it is required to use another network.

3 Simulations

Using gazebo to simulate the arm by using camera and collision detection so that the Rl agent can use these for training and reward function.

3.1 Rewards

There are three rewards in general. They can be listed as follows.

- Reward for the successful end of episode
- Reward for the unsuccessful end of episode
  - Hit to the ground
  - End of episode without success
  - Touch the object with other parts than gripper (task 1)
- Reward Delta, based on distance for each episode.

The reward function is if successful +100 and if unsuccessful -100.

3.2 Hyperparameters

The hyperparameters used in both tasks are given in Table 1.

TABLE 1
Hyperparameters

Parameter	Task 1	Task2
INPUT_WIDTH	64	64
INPUT_HEIGHT	64	64
OPTIMIZER	RMSprop	RMSprop
LEARNING_RATE	0.1	0.01
REPLAY_MEMORY	10000	20000
BATCH_SIZE	256	512
USE_LSTM	true	true
LSTM_SIZE	256	256

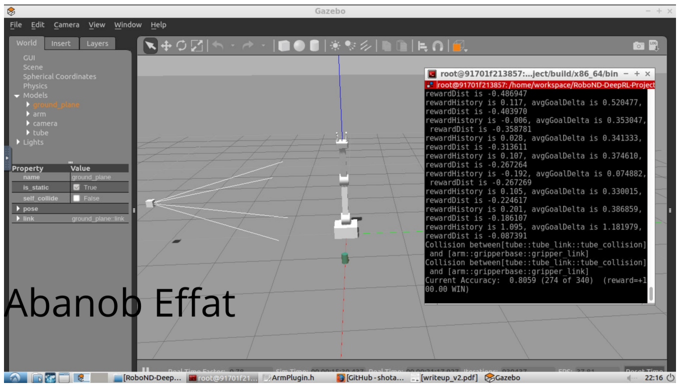
3.3 Task 1

A reward(WIN *10) is giving If any part of the arm touch the object.
A reward(WIN) is giving if a positive weighted average is derived.
A penalty(LOSS *10) is giving if any part of the robot touch the ground and the episode end.
A penalty(LOST * distance to goal) is provided if a negative weighted average is derived.
Any collision ends the episode.

3.4 Task 2

A reward(WIN *20) is giving if the gripper base of the robot touch the object. A penalty(LOSS *5) is giving if any part of the robot touch the object. A penalty(LOSS *10) is the robot touch the ground A penalty(LOST) is added for no movement if the absolute average goal is less than 0.001 for the gripper base. Any collision ends the episode.

3.5 Task 1

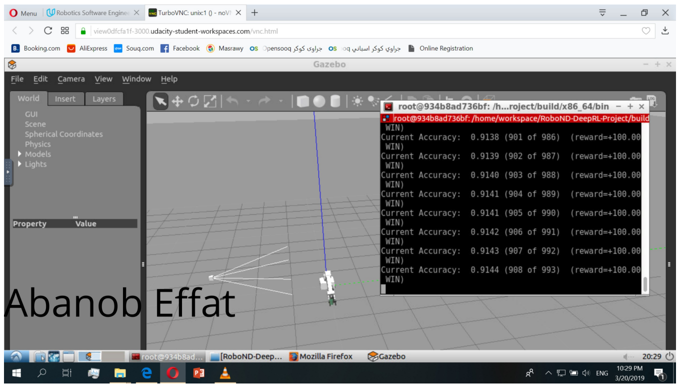


```
root@91701f213857:...ject/build/x86_64/bin
root@91701f213857: /home/workspace/RoboND-DeepRL-Project
rewardDist is -0.486947
rewardHistory is 0.117, avgGoalDelta is 0.520477,
rewardDist is -0.403970
rewardHistory is -0.006, avgGoalDelta is 0.353047,
 rewardDist is -0.358781
rewardHistory is 0.028, avgGoalDelta is 0.341333,
rewardDist is -0.313611
rewardHistory is 0.107, avgGoalDelta is 0.374610,
rewardDist is -0.267264
rewardHistory is -0.192, avgGoalDelta is 0.074882,
 rewardDist is -0.267269
rewardHistory is 0.105, avgGoalDelta is 0.330015,
rewardDist is -0.224617
rewardHistory is 0.201, avgGoalDelta is 0.386859,
rewardDist is -0.186107
rewardHistory is 1.095, avgGoalDelta is 1.181979,
rewardDist is -0.087391
Collision between[tube::tube_link::tube_collision]
 and [arm::gripperbase::gripper_link]
Collision between[tube::tube_link::tube_collision]
 and [arm::gripperbase::gripper_link]
Current Accuracy:  0.8059 (274 of 340)  (reward=+1
00.00 WIN)
```

Fig. 1. Task 1

3.6 Technical Comparison

The accuracy graph for both of the tasks are given in the Figures **??**, **??**. Red line and points show the Reward value. For Task 2, the values are divided by 10 on the graph. The Yellow line and points show 100 for win and 0 for loss. Blue line shows the accuracy as the percentage value.



```
root@934b8ad736bf: /h...roject/build/x86_64/bin
root@934b8ad736bf: /home/workspace/RoboND-DeepRL-Project/build
 WIN)
Current Accuracy:  0.9138 (901 of 986)  (reward=+100.00
 WIN)
Current Accuracy:  0.9139 (902 of 987)  (reward=+100.00
 WIN)
Current Accuracy:  0.9140 (903 of 988)  (reward=+100.00
 WIN)
Current Accuracy:  0.9141 (904 of 989)  (reward=+100.00
 WIN)
Current Accuracy:  0.9141 (905 of 990)  (reward=+100.00
 WIN)
Current Accuracy:  0.9142 (906 of 991)  (reward=+100.00
 WIN)
Current Accuracy:  0.9143 (907 of 992)  (reward=+100.00
 WIN)
Current Accuracy:  0.9144 (908 of 993)  (reward=+100.00
 WIN)
```

Fig. 2. Task 2

4 Discussion

The algorithm was able to teach the agent for the objective, and the robot arm was able to reach the object which was always in the same location. However, there were limitations on the agent's progress. The results were not good enough for a problem where the object location changes randomly.
The first task has an accuracy rate of 91.04%. 908 from 993 episode.
The second task lasted 340 episodes. The robot arm reached to target 274 time The accuracy was 80.59%. The success rate was improving with each trial. Testing for more episodes would probably result in higher accuracy, but the run was terminated to prevent out of memory error.

Besides improving the reward functions, enhancing the capacity of the neural network would help to achieve better results. Increasing the batch size, replay memory, LSTM size parameters are promising to obtain more successful outcomes.

5 Conclusion / Future work

The parameters and the rewards can be improved. Besides, the algorithm may enable the robot to grasp the object. The agent should be trained to access objects that located randomly. First, only on the x-axis, then both on the x-axis and y-axis.
Testing the algorithm on the Jetson TX2 and comparing the results with the current results is a waiting task for the future work. Upon receiving successful results, the algorithms in the project can be used on a real robot

References