Deep RL Arm Manipulation Project

Christoph Doerr

Abstract—In this study, a DQN agent and defined reward functions are used to teach a robotic arm to carry out two objectives. First, touching a given object with every part of the robot, with an accuracy of at least 90 %. Second, touching a given object with the gripper base of the robot arm, with at least 80 % [1].

Index Terms—Robot, IEEEtran, Udacity, LaTeX, Reinforcement Learning, DQN, ROS.

1 Reward Functions

For this study, the output of a Deep Q-Network(DQN) is mapped to the control of each joint of a robotic arm. The robotic arm has 3 non-static joints, those can be controlled with velocity or position control. The position control approach is the favored one to complete this project. For both project objectives, reward functions were defined to train the robotic arm to meet the project goals. For both objectives, the reward functions were the same, except for changes in the collision check.

2 Objective 1

For the first objective the following reward functions were chosen:

1) If any part of the robot touches the object, the reward is REWARD_WIN
2) If any part of the robot touches the ground, the reward is REWARD_LOSS * 2
3) If the robot moves more than 4 mm the reward is $avgGoalDelta = (avgGoalDelta * alpha) + (distDelta * (1.0f - alpha))$. With alpha as a constant between 0 and 1 and $distDelta$ as the difference between the distance to the object at timestep t and $t + 1$. This smoothes the robot movement towards the object
4) If the robot moves less than 4 mm the reward is REWARD_LOSS / (200)

3 Objective 2

For the second objective the following reward functions were chosen:

1) If any part of the robot touches the object with the gripper, the reward is REWARD_WIN
2) If any part of the robot touches the ground, the reward is REWARD_LOSS * 2
3) If the robot moves more than 4 mm the reward is avgGoalDelta
4) If the robot moves less than 4 mm the reward is REWARD_LOSS / (200)

3.1 Hyperparamters

The following list shows the hyperparameters, which were tuned:

- **REWARD_WIN, REWARD_LOSS** were chosen to be 1000 and -100 respectively
- **INPUT_WIDTH, INPUT_HEIGHT** were reduced to 64x64 to reduce computation cost
- **NUM_ACTIONS** were set to 6, because of the 3 robot joints
- **OPTIMIZER** was choosen the RMSprop optimizer
- **LEARNING_RATE** was set to 0.01, after testing 0.3 and 0.5 it gave the best results
- **LSTM_SIZE** was set to 256

4 Results

With further tuning of the hyperparameters, the results got closer to the project objectives. The following figures show the results for both study approaches of touching the object with the arm and the gripper.

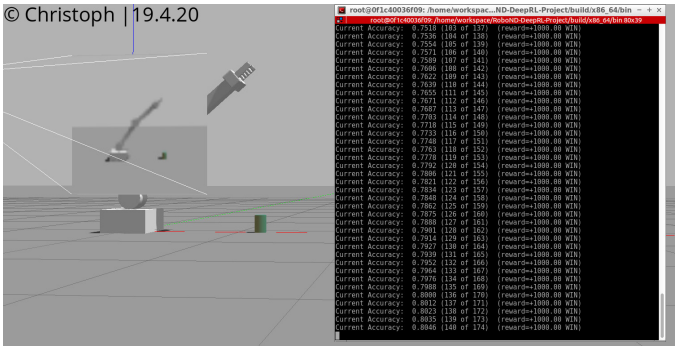

Fig. 1. Robot Arm touches the object with every part

The tuning of the REWARD_LOSS, when the robot touches the ground took some time, because the robot arm always touched the object and the ground at the same time, so the REWARD_HISTORY was always negative, although the movements of the robot got in the right direction. After more than one hundred tries without further progress, the REWARD_LOSS for touching the ground got increased by times two. This helped the DQN to make the right assumptions about its movements towards the object. Sometimes

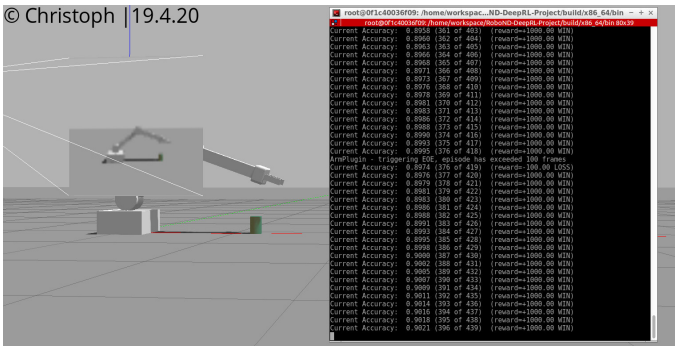

Fig. 2. Robot arm touches the object with the gripper

the arm seemed to stuck in a specific position. Therefore, I introduced the penalty for not moving 4 mm between two different time steps.

4.1 Conclusion / Further Work

The accuracy of the DQN could be improved by changing the control method to velocity control or torque control. Additionally, the hyperparameters could be redefined to achieve a better result. Furthermore, contact sensors at different positions of the arm could detect the collision between the robot arm and the object. With closer contact to the gripper, the penalty reward could be dynamically applied. As a further step, teaching to robot to grab the object would be interesting.

REFERENCES

[1] Udacity, *Udacity's Robotics Softwareengineer Nanodegree, Reinforcement Learning Lessons*. Udacity, 2019.