Deep RL Arm Manipulation

Ghee Chong Foo

Abstract—In this project, a robotic arm was trained through reinforcement learning, using Deep Q-Learning networking, with the objective of different part of the arm touching the object with desired accuracy.

Index Terms—Robotics, DeepRL.

1 Introduction

This project explores the feasibility of manipulating a robotic arm using Deep RL (reinforcement learning) in a gazebo environment and arm plugin (ArmPlugin.cpp) which assigns positive or negative rewards when the arm touches the object with different part of the arm, or misses it by touching the ground.

Figure 1 shows the setup for the project. The gazebo plugin, namely ArmPlugin.cpp will subscribe to camera and contact/collision node which is then pass on to the DQN agent as input. Code snipppet as below:

```
cameraSub = cameraNode->Subscribe(
"/gazebo/arm_world/camera/link/camera/image",
&gazebo::ArmPlugin::onCameraMsg, this);

collisionSub = collisionNode->Subscribe(
"/gazebo/arm_world/tube/tube_link/my_contact",
&gazebo::ArmPlugin::onCollisionMsg, this);
```

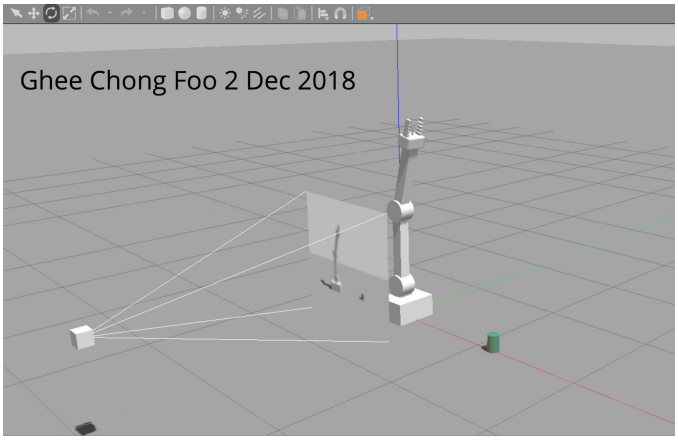

Fig. 1. Robotic Arm Initial Position

2 Reward Functions and Joint Control

Three types of rewards are defined:

1) **Win**: Reward is issued when the robotic arm achieve the objective and ending the episode with a success.
2) **Loss**: Penalty issued when robotic arm misses the objectives or hitting undesirable location. This ends the episode with a failure
3) **Interim**: Interim reward is issued while robot is moving and episode still running. This will guide the agent to move the robot in the desired direction and will be positive or negative depending on position of the arm.

2.1 Win Reward

Positive reward is issued when robot arm touches the object. In objective 1, reward is issued if any part of the robot touches the object, while in objective 2 reward is only issued when gripper touches the object. Win reward is 1000x larger than interim reward to guide the robot from trying to collect reward through interim reward, rather than moving to the right direction.

2.2 Loss Reward

Negative reward is issued as follows:

1) The robotic arm gripper touches the ground
2) The middle joint touches the ground
3) Part of robots beside the gripper touches the object (objective 2)

2.3 Interim Reward

Interim reward varies between a reward or penalty depending on the position of the arm and the last action it took. Code snippet as below:

```
REWARD_INTER = 0.2;
distDelta = lastGoalDistance - distGoal;
weightedDelta=distDelta*(1.0f-REWARD_INTER);
weightedHist = avgGoalDelta * REWARD_INTER;
goalReward = weightedDelta + weightedHist;
scaledReward=goalReward*20; //objective 1
scaledReward = goalReward; //objective 2
avgGoalDelta = goalReward;
rewardHistory = scaledReward;
lastGoalDistance = distGoal;
```

disDelta is the difference between distance to the object from last state and the current object, and this should be minimized. The formula is such that current distance has more impact on the learning than the last reward.

Parameter	Value	Default
INPUT_WIDTH	64	512
INPUT_HEIGHT	64	512
OPTIMIZER	Adam	None
LEARNING_RATE	0.02f	0.0f
REPLAY_MEMORY	10000	10000
BATCH_SIZE	64	8
USE_LSTM	true	false
LSTM_SIZE	256	32
EPS_DECAY	200	200

TABLE 1
Hyperparameters

2.4 Joint Control

Velocity control is implemented for joint control. It was found that enabling *Velocity Control* provided better result than Position Control.

3 Hyperparameters

The Hyperparameters for this project is changed as in Table 1.

- INPUT_WIDTH and INPUT_HEIGHT are reduced from 512 to 64 to provide a more reasonable image size and save memory.
- OPTIMIZER used is Adam. While another possible option is RMSProp, Adam is found to give better performance in this case.
- LEARNING_RATE is increased from 0 to 0.02 with BATCH_SIZE of 64 to speed up the learning process. USE_LSTM is turned on to enable LSTM agent with LSTM_SIZE changed to 256.
- Other values remain unchanged.

4 Results

The 2 primary objectives for the project has been met:

1) Have any part of the robot arm touch the object of interest, with at least a 90% accuracy, as in Figure 2
2) Have only the gripper base of the robot arm touch the object, with at least a 80% accuracy, as in Figure 3

The first objective was harder to achieve as at one point the accuracy only reaches 80+% and cannot move beyond the 90% objective. Only by repeatedly scaling the interim reward provided the necessary improvement.

The second objective, on the other hand, was much easier to achieve by adjusting the interim reward and loss reward as learned during objective 1.

5 Discussion and Future work

Training the DeepRL agent is a time consuming process which takes many hours of computing time. Same goes for hyperparameter tuning which might give different result for same setting, and one has to run the same setting multiple times to see the consistency, but might encounter few outlying results in particular run, i.e. one might observe a very poor accuracy in one run but a very high accuracy on another. The consistency should converge if the agent was trained for longer hours, but this assumption is not able to be proved as often Udacity workspace timed out before longer hours of result can be observed. This can be mitigated by running in a Jetson TX2 hardware.

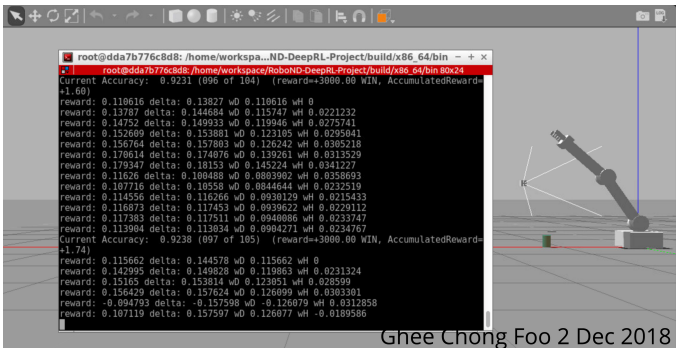

Fig. 2. Robotic Arm Touching Object with non-Gripper in objective 1

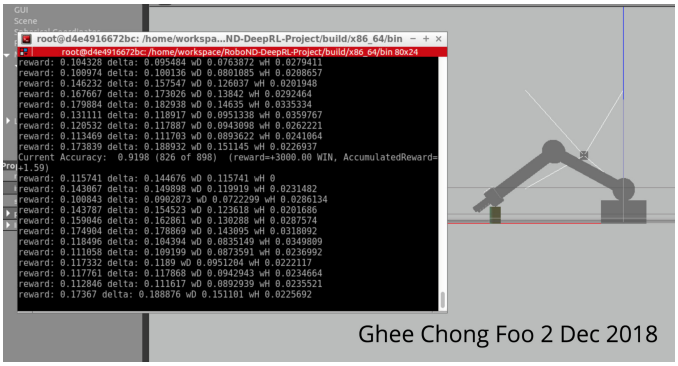

Fig. 3. Robotic Arm Touching Object with Gripper in objective 2

The use of this setup can be expanded in a factory environment whereby the robotics arm can be trained to recognize interested object and perform pick and place action. However as demonstrated in this project, a lot of training and hyperparameter tuning has to be done in order to achieve a desirable goal setting which might not be possible for the agent to learn this in real time. Instead a better result can be achieved by training the agent offline and making the agent more intelligent by providing more input, such as more camera from different angles.