Deep RL Arm Manipulation

Koki Mitsunami

Abstract—This paper describes building a DQN agent and define reward functions to teach a robotic arm to the robot arm touch the object of interest.

Index Terms—Robot, Udacity, Deep RL.

1 Reward Functions

In this project, each reward function and associated reward values are explained.

1.1 Robot Gripper Hitting the Ground

Checking if the gripper is hitting the ground or not is done with below functions. Assigned reward is always large negative value because gripper hitting the ground is critical for the task. The reward becomes better when gripper position is closer to a target. Additionally a threshold value and action joint interval are decreased from default to avoid the collision.

```
#define REWARD_LOSS -100.0f
actionJointDelta = 0.10f; // from default 0.15f
groundContact = 0.01f;    // from default 0.05f
checkGroundContact
           = ( gripBBox.min.z < groundContact );
rewardHistory
     = 10*REWARD_LOSS
        + 500*(gripBBox.min.x - propBBox.min.x);
newReward     = true;
endEpisode    = true;
```

1.2 Interim Reward based on the Distance to the Object

Interim reward is calculated based on the distance between the arm and the object. To calculate reward, smoothed moving average of the delta of the distance to the goal is used. When the average is closing to the object, then reward becomes positive. Better value is attained, closer the arm to the object.

```
#define REWARD_WIN  100.0f
#define REWARD_LOSS -100.0f
alpha = 0.9;
avgGoalDelta  = (avgGoalDelta * alpha)
              + (distDelta * (1 - alpha));
if( avgGoalDelta > 0.01 ){
        rewardHistory =
          REWARD_WIN*(1-tanh(distGoal))
          - 100*(gripBBox.min.x - propBBox.min.x);
}else{
        rewardHistory = REWARD_LOSS - 10*distGoal;
}
newReward     = true;
```

1.3 Collision between the Arm and the Object

Touching the arm with the object is purpose of the objective 1. Therefore, reward becomes large positive value. And condition is provided as below:

```
#define REWARD_WIN  100.0f
#define COLLISION_ITEM
              "tube::tube_link::tube_collision"
collisionCheck = ( collision1 == COLLISION_ITEM );
if (collisionCheck){
        rewardHistory = 100*REWARD_WIN;
        newReward  = true;
        endEpisode = true;
}
```

1.4 Collision between the Arms Gripper Base and the Object

Touching the arm's gripper base with the object is purpose of the objective 2. The reward function is provided as below:

```
#define REWARD_WIN  100.0f
#define COLLISION_ITEM
             "tube::tube_link::tube_collision"
#define COLLISION_POINT
             "arm::gripperbase::gripper_link"
#define COLLISION_POINT2
             "arm::gripper_middle::middle_collision"
collisionCheck1 = ( collision1 == COLLISION_ITEM);
collisionCheck2 = ( collision2 == COLLISION_POINT) ||
                  ( collision2 == COLLISION_POINT2);
if( collisionCheck1 ){
        if (collisionCheck2){
                rewardHistory = 100*REWARD_WIN;
                newReward  = true;
                endEpisode = true;
    }
        else{
                rewardHistory = REWARD_LOSS / 5.0;
                newReward  = true;
                endEpisode = true;
    }
}
```

1.5 Joint Control

There are two existing approaches to control the control of each joint movement for the robotic arm, which are Velocity Control or Position Control. In this project, Position Control is chosen.

2 HYPERPARAMETERS

Hyperparameters selected are same for both objectives as below:

```
#define INPUT_WIDTH   64
#define INPUT_HEIGHT  64
#define OPTIMIZER "RMSprop"
#define LEARNING_RATE 0.1f
#define REPLAY_MEMORY 10000
#define BATCH_SIZE 32
#define USE_LSTM true
#define LSTM_SIZE 32
```

INPUT_WIDTH and INPUT_HEIGHT are reduced to 64 from default value 512 because the default value is too large to calculate in reasonable time and 64 is enough to achieve required results.

OPTIMIZER is selected to ”RMSprop” because it is one of most common optimizers and adaptively adjust its learning rate using gradients.

LEARNING_RATE is selected to 0.1 heuristically.

LSTM is used to consider about past behavior. BATCH_SIZE and LSTM_SIZE are also decided experimentally by exploring results.

3 RESULTS

This section explains the results. In this project, there are two objectives:

1) Any part of the robot arm should touch the object with at least an accuracy of 90%.
2) Only the gripper base of the robot arm should touch the object with at least an accuracy of 80%.

Both objectives are achieved with settings described in previous sections.

3.1 Objective 1

As for objective 1, 96% accuracy is achieved for 102 runs, which means 98 of 102 are success. Screenshot of the result is shown below:

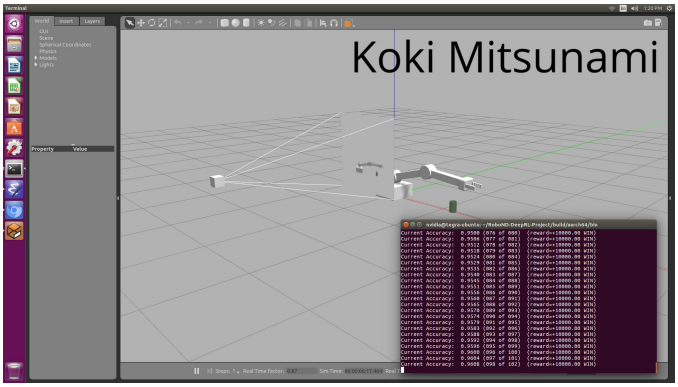

Fig. 1. Screenshot of Objective1

To achieve this result, some hyperparameter tuning is done. After iterations, convergence behavior is affected by initial random state a little. In other words, arm’s movement slightly changes for each run. However, the requirement is attained easier than the objective 2.

3.2 Objective 2

As for objective 1, 82% accuracy is achieved for 102 runs, which means 84 of 102 are success. Screenshot of the result is shown below:

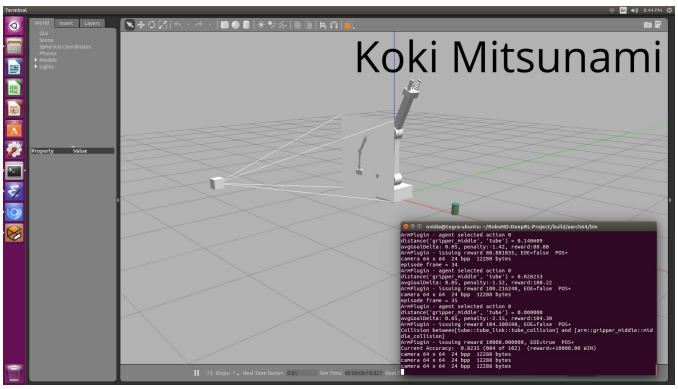

Fig. 2. Screenshot of Objective 2

To achieve this result, a lot of hyperparameter tuning is done. Arm’s behavior is very sensitive to reward functions and hyperparameters. To have only the gripper base of the robot arm touch the object, reward function is adjusted to avoid touch with middle parts of the arm and the object. And the arm’s behavior largely depends on its initial state.

4 FUTURE WORK

As described in previous section, randomness of learning behavior is issue of this DeepRL project. And iterative process of learning should be also tackled. As for the randomness, some techniques to stabilize the result should be considered. And for iterative process, it is essential to explore hyperparameter to achieve better results in a current technological circumstances. To reduce each process, using high-performance GPU is considerable.