Robotics Reinforcement Learning

Ulrich Ludmann

Abstract—This report covers the Reinforcement Learning Project of the Udacity Robotics Software engineer Nanodegree Program. Reinnforcement learing is a a new way to teach the robot to do tasks efficiently. Reinforcement learning can also be considered as a cheaper way of robotic teaching because the robot is not programmed explicitly but learns to do the task on its own.

Index Terms—Robot, Reinforcement Learning, Nvidia Jetson TX2, Udacity,

1 Introduction

Programming a robot turns out to be a complicated task. A new way to program a robot is through reinforcement learning, where the robot learns to do tasks by trial and error. The robot (agent) sits in a simulated environment. It receives a reward for each action it takes. The environment rates the decisions of the agent. If the decision is considered to be good. The robot gets a positive reward. If the agent performs a wrong action it receives a negative reward. The relationship of the agents action and the reward is described as the reward function. The agents goal is to maximize it reward.

2 Reward Function

Position based control was used. The following reward function for each action was used:

```
avgGoalDelta  = (avgGoalDelta * alpha) +
 ↪  (distDelta * (1.0f - alpha));
rewardHistory = avgGoalDelta *
 ↪  REWARD_MULTIPLIER;
```

This reward function smooths the goal delta by the factor of alpha and multiplies it by the REWARD_MULTIPLIER. The distance between gripper middle and the tube is calculated. If a collision is detected and the episode ends, the following reward is applied:

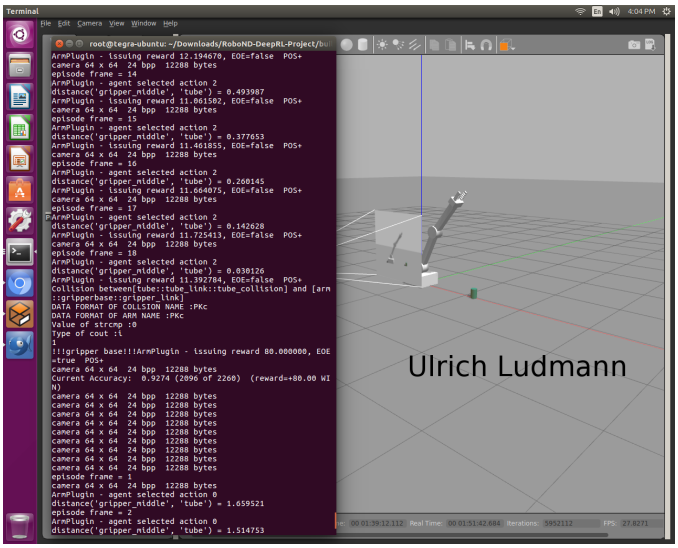
```
if (strcmp(armcollision,
 ↪  "arm::gripperbase::gripper_link") == 0)
 ↪  { // if gripper base contact
 ↪  printf("!!!gripper base!!!");
rewardHistory = 4 * REWARD_WIN;
newReward  = true;
endEpisode = true;
return;
}
else if (strcmp(armcollision,
 ↪  "arm::gripper_middle::middle_collision")
 ↪  == 0) { // if gripper middle
printf("!!!middle Collision!!!\n");
rewardHistory = 2 * REWARD_WIN;
newReward  = true;
endEpisode = true;
return;
}
else if (strcmp(armcollision,
 ↪  "arm::link2::collision2") == 0) { //
 ↪  DONT DO THIS
printf("!!!NO!!!\n");
rewardHistory += 5 * REWARD_LOSS;
newReward  = true;
endEpisode = true;
return;
}
else {
printf("!!!Else...!!!\n" );
rewardHistory += 0.5f * REWARD_LOSS;
newReward  = true;
endEpisode = true;
return;
}
```

The environment differentiates between the different collision points. If a collision with the gripper base is detected, the agent receives the highest reward. The agent also receives rewards if it touches similar points like the middle collision item of the arm. If a collision between the arm link is detected, the agent receives a high negative reward because this behavior is not wanted. The goal of the Robot is to touch the tube with its gripper base. The implemented reward function leads to a accuracy of success rate of 92.74 Percent. So there was no need to implement a second reward function to achieve 90 percent contact between any component of the arm and the tube.

3 Hyperparameters

It turned out that the reinforcement learning algorithm is really sensible to hyperparameter changes. These hyperparameters were used:

```
#define INPUT_CHANNELS 3
#define ALLOW_RANDOM true
#define DEBUG_DQN false
#define GAMMA 0.95f
#define EPS_START 0.5f
#define EPS_END 0.05f
#define EPS_DECAY 100
#define INPUT_WIDTH   64
#define INPUT_HEIGHT  64
#define OPTIMIZER "RMSprop"
#define LEARNING_RATE 0.28f
```




```
#define REPLAY_MEMORY 100
#define BATCH_SIZE 32
#define USE_LSTM false
#define LSTM_SIZE 256
#define REWARD_WIN  20.0f
#define REWARD_LOSS -30.0f
#define REWARD_MULTIPLIER 100.0f
```

The EPS-parameters are called epsilon parameters and help the robot to take random actions while optimizing the Q-Network. If a low learning rate is chosen, the agent needs to take random actions more often. I used RMSprop as a optimizer because its considered as a good optimizer for Q-Learning. I was not able to achieve better results with a LSTM.

4 Results

I was impressed by the results. I even saw the agent recovering from really bad positions. However one need to be really careful with the Hyperparameters. If they are changed with a too high magnitude the agent performance decreases rapidly and often the agent is not able to achieve its task anymore.

5 Future Work

The next step would be to activate the robots base joint and take the task to the third dimension. It would also be interesting to change the position of the tube. But i think this would require much more computation power than we have on our Jetson TX2. The performance of the robot could be improved by considering the smoothness of the movement or the force that is applied to the joints or the tube when the gripper touches the tube. This could be implemented into the reward function.