ROBOTICS ND - DeepRL - Project

Reward Functions: Explain the reward functions that you created.

As proposed in task 5 for the project and smoothed average of the mean of the distance of the goal is used to reward the arm when getting close to the object.

```
const float distDelta  = lastGoalDistance - distGoal;
const float alpha = 0.5f;
float average_delta = 0.0f;

// compute the smoothed moving average of the delta of the distance to th
e goal
average_delta = (average_delta * alpha) + (distDelta * (1.0f - alpha));
rewardHistory = average_delta * REWARD_WIN;
newReward     = true;
```

The reward parameters are the following:

- REWARD_WIN 1.0f
- REWARD_LOSS -1.0f

The REWARD_LOSS is applied when the object touches the ground, then the rewardHistory is set to REWARD_LOSS and the episode is ended. The variable checkGroundContact is defined as true when the lowest part of the gripper touch the ground.

```
const float groundContact = 0.00f;
bool checkGroundContact = false;

        if (gripBBox.min.z <= groundContact) {
            checkGroundContact = true;
        }


        if(checkGroundContact)
        {

            if(DEBUG){printf("GROUND CONTACT, EOE\n");}

            rewardHistory = REWARD_LOSS;
            newReward     = true;
            endEpisode    = true;
        }
```

On the contrary, for a REWARD_WIN is delivered when the arm collision with the object, the function to check the collision can be found below. When the condition is meet the function add the REWARD_WIN as an extra to the rewardHistory to reinforce the learning, the episode is ended to continue with the training.

```
if (strcmp(contacts->contact(i).collision1().c_str(), COLLISION_ITEM) ==
0)
        {
            rewardHistory = rewardHistory + REWARD_WIN;
            newReward  = true;
            endEpisode = true;
            return;
        }
```

The Function above, is valid to the first objective of this project. It has a variable named COLLISION_ITEM == "tube::tube_link::tube_collision", and that means in this case we are checking whenever the robot touch the object.

For the second objective we check the collision between the arm (collision2) and the gripper, defined as COLLISION_POINT "arm::gripperbase::gripper_link". The function in this case is as follows.

```
if (strcmp(contacts->contact(i).collision2().c_str(), COLLISION_POINT) ==
 0)
        {
            rewardHistory = rewardHistory + REWARD_WIN;
            newReward  = true;
            endEpisode = true;
            return;
        }
```

Hyperparameters: Specify the hyperparameters that you selected for each objective, and explain the reasoning behind the selection.

First Objective 90% accuracy for Arm Collision

The first decision was to reduce the input width and height after some memory errors arose.

The DQN python agent definition comes preload with two different optimizers, Adam(Adaptive Moment Estimation) and RMSprop. An optimizer should reduce the loss function as quickly as possible to obtain a good solution. RMSprop was chosen, with a learning rate of 0.1, while this learning rate could be too high and let to local minima or completely erroneous solutions, it behaves well for this problems.

HYPERPARAMETERS

- INPUT_WIDTH 64
- INPUT_HEIGHT 64
- OPTIMIZER "RMSprop"
- LEARNING_RATE 0.1f
- REPLAY_MEMORY 10000
- BATCH_SIZE 8
- USE_LSTM false
- LSTM_SIZE 32
- EPS_DECAY 80

Second Objective 80% accuracy for Gripper Collision

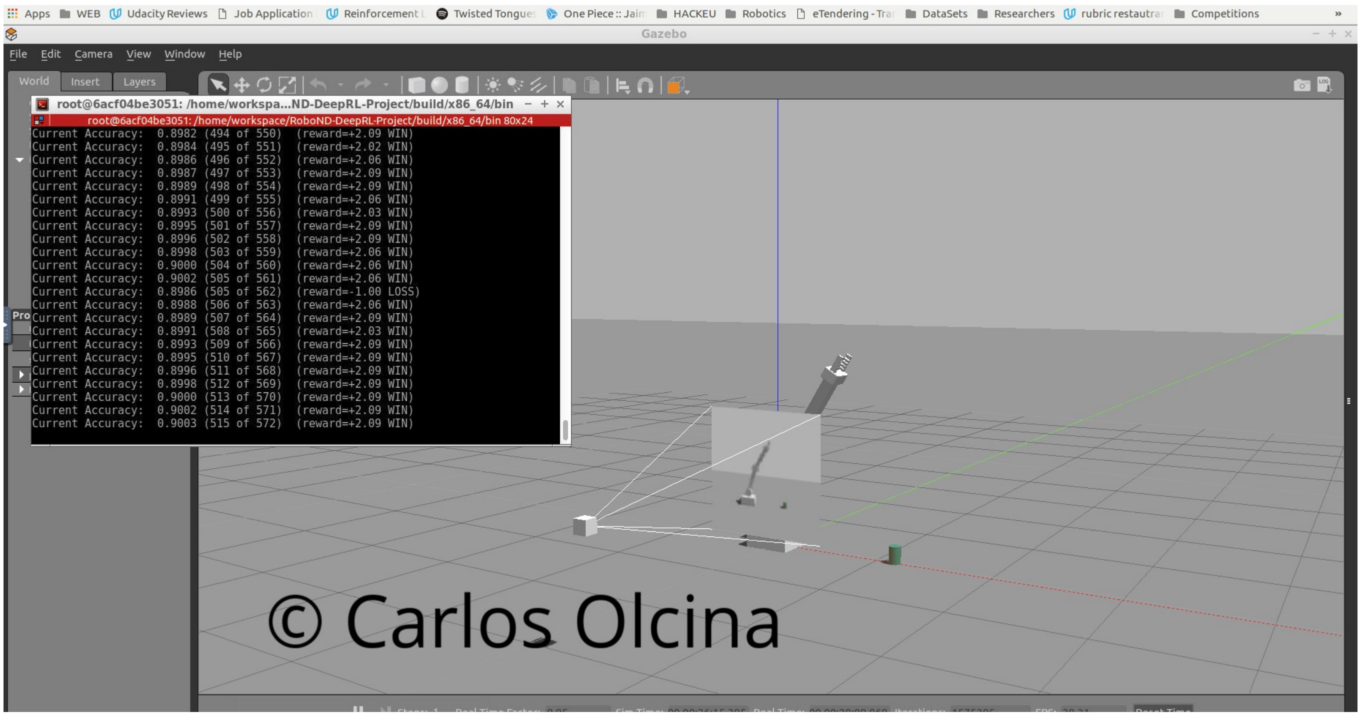
The hyperparameters chosen for the second objectives were the same as in the first objective.

Results: Explain the results obtained for both objectives. Include discussion on the DQN agent's performance for both objectives. Include watermarked images, or videos of your results.

90% accuracy for Arm Collision results

A water-marked screenshoot of the arm accuracy after ~400 tries is shown below.

Additionally the status of the project when the accuracy was reach can be found in this Repository (https://github.com/Olcina/RoboND-DeepRL-Arm.git) commit "a3f57c7"

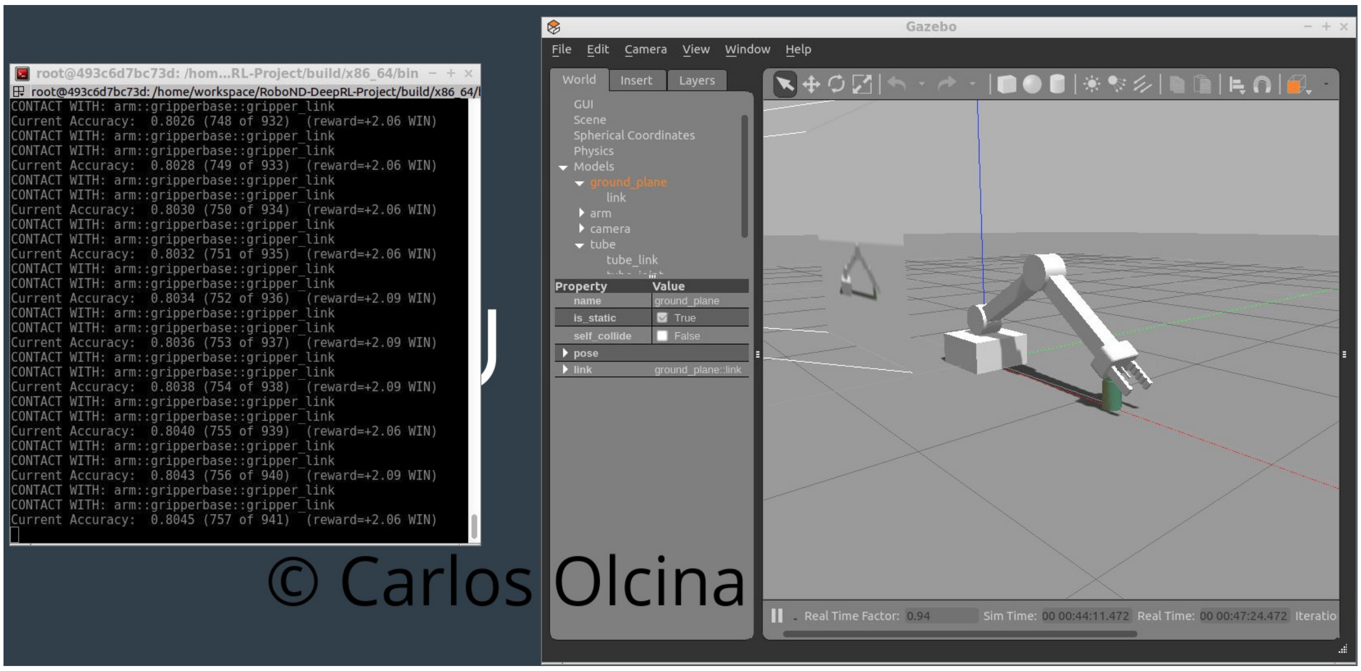


root@6acf04be3051: /home/workspace/RoboND-DeepRL-Project/build/x86_64/bin 80x24

```
Current Accuracy:  0.8982 (494 of 550)  (reward=+2.09 WIN)
Current Accuracy:  0.8984 (495 of 551)  (reward=+2.02 WIN)
Current Accuracy:  0.8986 (496 of 552)  (reward=+2.06 WIN)
Current Accuracy:  0.8987 (497 of 553)  (reward=+2.09 WIN)
Current Accuracy:  0.8989 (498 of 554)  (reward=+2.09 WIN)
Current Accuracy:  0.8991 (499 of 555)  (reward=+2.06 WIN)
Current Accuracy:  0.8993 (500 of 556)  (reward=+2.03 WIN)
Current Accuracy:  0.8995 (501 of 557)  (reward=+2.09 WIN)
Current Accuracy:  0.8996 (502 of 558)  (reward=+2.09 WIN)
Current Accuracy:  0.8998 (503 of 559)  (reward=+2.06 WIN)
Current Accuracy:  0.9000 (504 of 560)  (reward=+2.06 WIN)
Current Accuracy:  0.9002 (505 of 561)  (reward=+2.06 WIN)
Current Accuracy:  0.8986 (505 of 562)  (reward=-1.00 LOSS)
Current Accuracy:  0.8988 (506 of 563)  (reward=+2.06 WIN)
Current Accuracy:  0.8989 (507 of 564)  (reward=+2.09 WIN)
Current Accuracy:  0.8991 (508 of 565)  (reward=+2.03 WIN)
Current Accuracy:  0.8993 (509 of 566)  (reward=+2.09 WIN)
Current Accuracy:  0.8995 (510 of 567)  (reward=+2.09 WIN)
Current Accuracy:  0.8996 (511 of 568)  (reward=+2.09 WIN)
Current Accuracy:  0.8998 (512 of 569)  (reward=+2.09 WIN)
Current Accuracy:  0.9000 (513 of 570)  (reward=+2.09 WIN)
Current Accuracy:  0.9002 (514 of 571)  (reward=+2.09 WIN)
Current Accuracy:  0.9003 (515 of 572)  (reward=+2.09 WIN)
```

80% accuracy for Arm Gripper results

Similarly, the status of the project when the 80% accuracy was reach can be found in the same Repository (https://github.com/Olcina/RoboND-DeepRL-Arm.git) same commit "a3f57c7"



root@493c6d7bc73d: /home/workspace/RoboND-DeepRL-Project/build/x86_64/

```
CONTACT WITH: arm::gripperbase::gripper_link
Current Accuracy:  0.8026 (748 of 932)  (reward=+2.06 WIN)
CONTACT WITH: arm::gripperbase::gripper_link
CONTACT WITH: arm::gripperbase::gripper_link
Current Accuracy:  0.8028 (749 of 933)  (reward=+2.06 WIN)
CONTACT WITH: arm::gripperbase::gripper_link
CONTACT WITH: arm::gripperbase::gripper_link
Current Accuracy:  0.8030 (750 of 934)  (reward=+2.06 WIN)
CONTACT WITH: arm::gripperbase::gripper_link
CONTACT WITH: arm::gripperbase::gripper_link
Current Accuracy:  0.8032 (751 of 935)  (reward=+2.06 WIN)
CONTACT WITH: arm::gripperbase::gripper_link
CONTACT WITH: arm::gripperbase::gripper_link
Current Accuracy:  0.8034 (752 of 936)  (reward=+2.09 WIN)
CONTACT WITH: arm::gripperbase::gripper_link
CONTACT WITH: arm::gripperbase::gripper_link
Current Accuracy:  0.8036 (753 of 937)  (reward=+2.09 WIN)
CONTACT WITH: arm::gripperbase::gripper_link
CONTACT WITH: arm::gripperbase::gripper_link
Current Accuracy:  0.8038 (754 of 938)  (reward=+2.09 WIN)
CONTACT WITH: arm::gripperbase::gripper_link
CONTACT WITH: arm::gripperbase::gripper_link
Current Accuracy:  0.8040 (755 of 939)  (reward=+2.06 WIN)
CONTACT WITH: arm::gripperbase::gripper_link
CONTACT WITH: arm::gripperbase::gripper_link
Current Accuracy:  0.8043 (756 of 940)  (reward=+2.09 WIN)
CONTACT WITH: arm::gripperbase::gripper_link
CONTACT WITH: arm::gripperbase::gripper_link
Current Accuracy:  0.8045 (757 of 941)  (reward=+2.06 WIN)
```

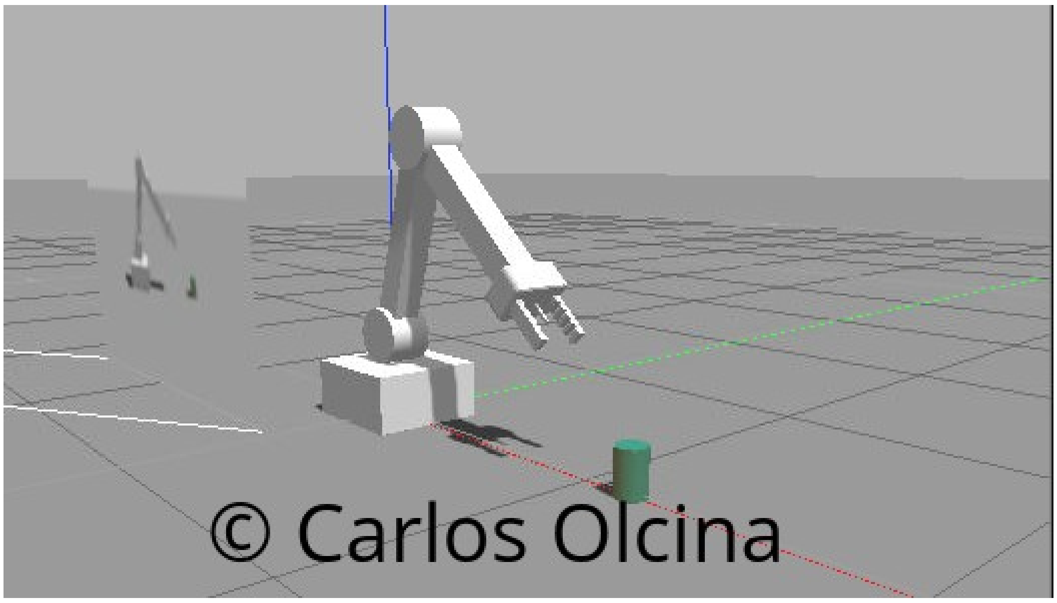
Even when the solution for both objectives was reached with the same hyperparameter selection the performance in the second case is much worse than in the first one. 941 tries against 572, that suggest that the results can be dramatically improved. The reward function used was indeed very simple and did not take into account variables such as time or joint angle values. Even with that, this agent reaches good performances after several tries.

There are different behaviors for the different objectives. For example, in the first objective, the arm generally overreached the object because of those where a valid solution, touching the object whit the arm. On the other hand, in the second objective, the agent learns quickly that a minimum angle between the two tubes of his arm is necessary to get positive rewards. Nevertheless, for both cases when the mentioned angle was two high the arm is not able to reach the objective and get stuck in a position that the average moving based regards cannot recover.

Future Work: Briefly discuss how you can improve your current results.

As discussed before, the results can be dramatically improved. For example, we could define a reward function that counts the time passed between the begin and the end of the simulation and gives more reward when the arm reaches the objective quickly. This reward function could lead to better and smoother solutions. But maybe, in such case, a limit to joint forces should be applied, or even penalized.

Another possible reward function could award the arm when touching with the arm but give more reward when touching whit the gripper, this could lead to avoiding more quick movements that lead to the arm touching the ground. In the same train of tough, penalize the arm when is close to the ground can avoid those situations two. In that case, an equilibrium between penalization for avoiding the ground and getting close to the object should be carefully chosen.



Lastly, the arm sometimes tends to bend himself too much, a penalization could be applied for delete this state (see image) but at the same time we would be limiting the arm's reach artificially and should not be used for a moving objective.

Ref: Code for playing videos in jupyter notebook taken from: stackoverflow