report.md

Deep RL Arm Manipulation

Introduction

Reinforce Learning is an old topic in Robotics/AI, which enable a robot discover behavior/Strategy through trial-error with its environment, value-action function can be learned in the reward-maximization process.

In Deep-rl, A neural network can be used to approximate this function, we will utilize a DQN agent which implents transition prediction model based CNN and LSTM, a robot arm is trained to perform specific tasks

In this project there are two primary objectives

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

Reward Functions:

Position Control was implemented here

The current value for a joint's position is stored in the array ref with the array lengths 3; and there are two outputs for every action - increase or decrease in the joint angles, step is actionJointDelta

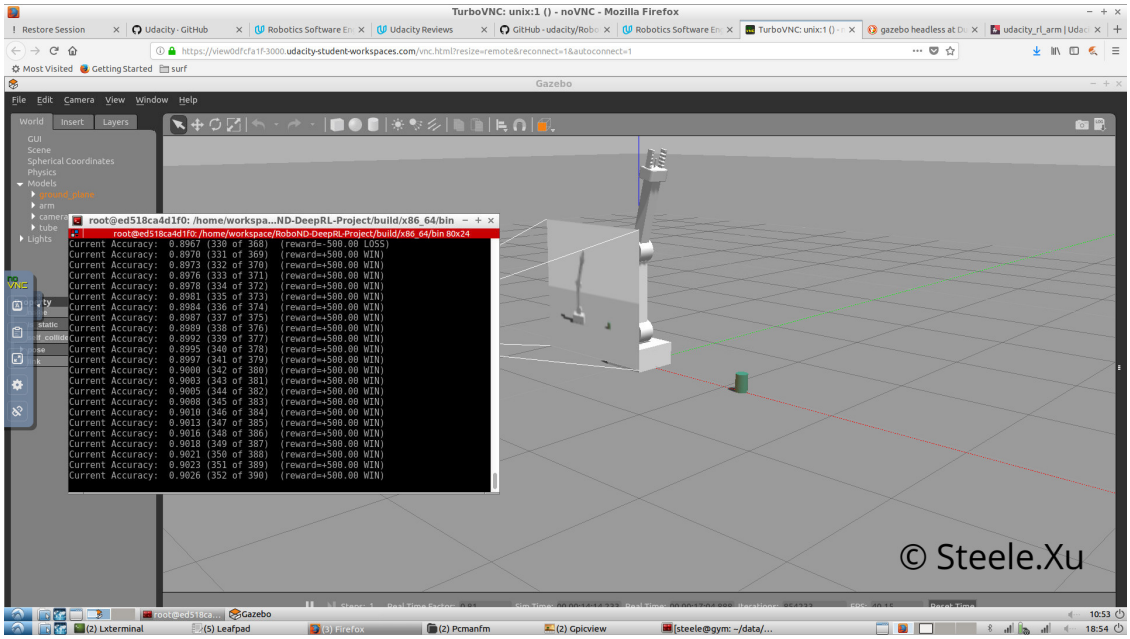
```
ref[action/2] += actionJointDelta * ((action % 2 == 0) ? 1.0f : -1.0f);
```

based on whether the action is even or odd

task1

- Reward for robot gripper hitting the ground, value as REWARD_LOSS(-500)
    - criteria: the distance between gripper_middle's box and prop, 0.05 as threshold
- Issue a reward based on collision between the arm(any part) and the object.
    - value as REWARD_WIN(500)
- Issue an interim reward based on the distance to the object
    - using moving average of delta of distance to the goal(distDelta)
    - `avgGoalDelta = (avgGoalDelta * 0.15) + (distDelta * (1.0f - 0.15))`
    - reward value as avgGoalDelta * REWARD_MULTIPLIER

here's the task1 result



task2

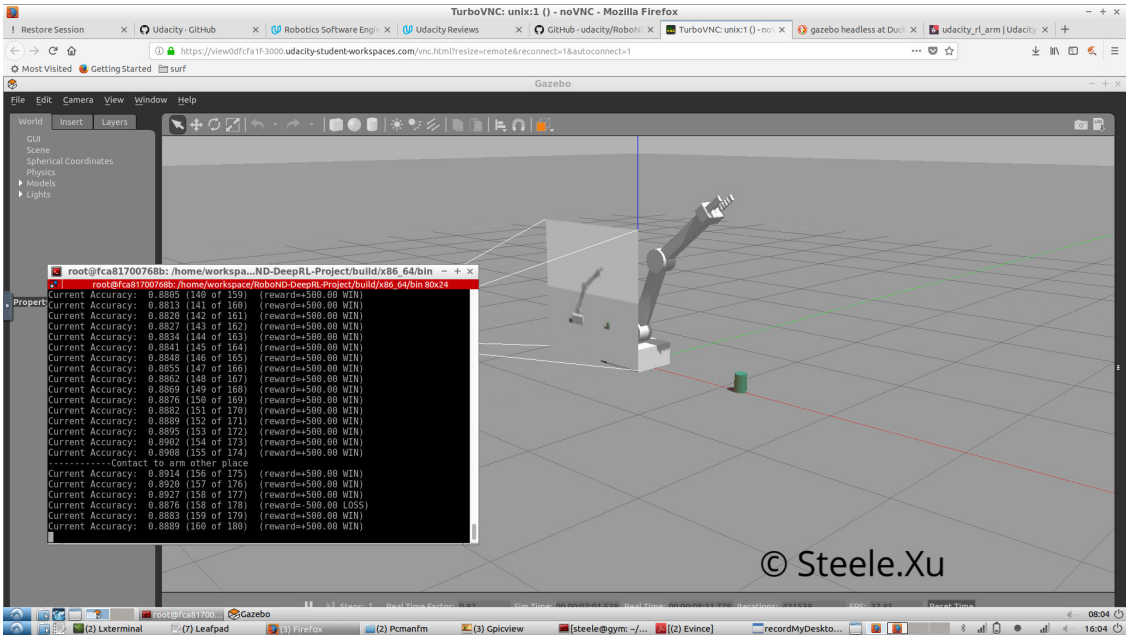
based on task1, just change the way handling non-correct collision(i.e when other part of arm collide with Prop)

In the begining of training(first 50 episode),try to encourage the arm to touch the object without call it end: reward also given if collision happened in first 30 frame.

```
        if(episodeFrames < 30){
                rewardHistory = 0.3*REWARD_WIN;
        }else {
          rewardHistory = 0.1*REWARD_LOSS;
          //endEpisode = true;
        }

        newReward  = true;
        if(totalRuns > 50){
                endEpisode = true;
        }
```

here's the task1 result, also check ArmPlugin-task2.cpp in the package



Hyperparameters

using LSTM here for better understanding the image sequence

to quicken the process, decrease the EPS_DECAY from 200 to 162; lower EPS_END may help to explore the state space

- EPS_END 0.01f
- EPS_DECAY 162

task1

- OPTIMIZER "RMSprop"
- LEARNING_RATE 0.001f
- REWARD_MULTIPLIER 5.0f

task2

- OPTIMIZER "Adam"
- LEARNING_RATE 0.005f
- REWARD_MULTIPLIER 660.0f

task 2 can't be reproduced very easily, so changed the optimizer making it stable; even try to increase REPLAY_MEMORY 20000. Another try to give more award to encourage arm moving towards the object via larger REWARD_MULTIPLIER

Results:

task1 stablely reach to 90% , by just setting the constraints according objective, DQN performed very good, decreasing EPS_DECAY also quicken the process;

When deal with task2, initialy nothing positive reward got. Trying to encourage the agent making any collision with prop object, with first successful touch, following action will be more sensefule. Though Sometime I got very good result (see the video task2out.ogv) it's not total controllable . DQN still lack some explicite way to make training smoothly like CNN

Future Work:

The task2 still not stable, more effort need to improve it-- increase the interim rewards by distance(from debug mode, sometime the interim award is too small), another is improve the encouraging mechanism which give more chance to get a good hit.

Another thought is on the imitation training : via human action demo/instruction in simulation environment, how quickly can an agent complete the task? This field is worth to have exploration.