Introduction

The goal of the arm is to approach and consistently touch the cylinder with any part of the arm, so the first reward function gives rewards based on whether the goal was achieved over 100 frames. If the arm touches the ground or the wrong part of the arm, a negative reward is given instead. There are interim rewards based on whether or not the arm is moving towards the objective. One is a reward directly proportional to the movement while the other is based on a slower moving average of the velocity. I also added a small negative interim reward based on goal distance to encourage it to find the object quicker. The joint control was position based

Background

The first parameter tuned was the image input size. They were set at 64x64 to be as low as possible as to still retain the information necessary for the job while optimizing for training speed. Learning rate was initially set to 0.1 since I wanted it to learn faster to hit the 90% accuracy with less episodes. Batch size was set to 64 to speed up learning as well.

I experimented with LSTM and size but it didn't seem to help too much. It gives the benefit of looking at previous frames to decide the next action, but it ends up being more parameters to learn, slowing down the training process. It could be more useful in velocity based joint control.

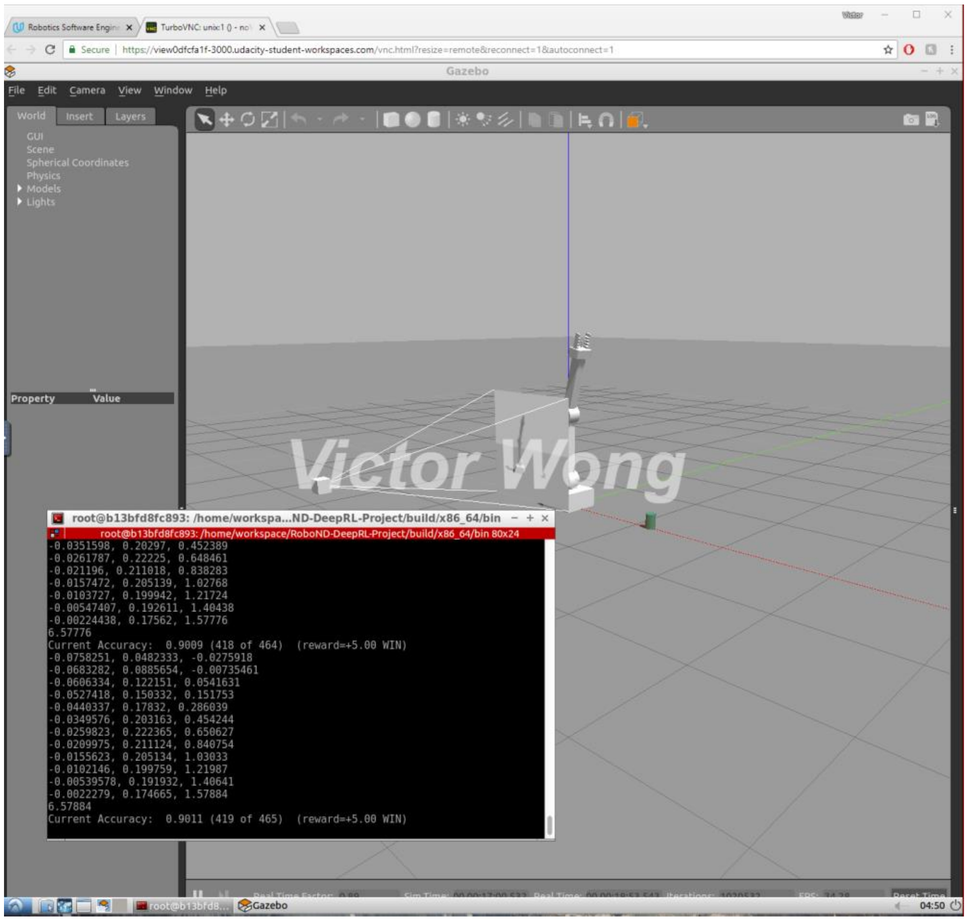
I used a gamma of 0.95 since I wanted to slightly encourage quicker episodes. But setting it too low would give less weight to the precision at the end of the episode, needed to get the gripper collision. I set the episode to start at .95 randomness and decay to .01. The goal here was to have high randomness in the beginning to encourage exploration and to dial it down once the more efficient paths have been hammered out. I lowered the ending randomness to .01 to reduce the likelihood of failing at the end when high precision is key.

In order to achieve the second task, I had to slightly decrease the learning rate and increase the delay for randomness decay. But most of it boiled down to simplifying my interim reward functions and normalizing their rewards so that no reward would overshadow the other ones. I changed all of the interim reward functions to be negative because I noticed it would get stuck trying to maximize marginal gains from a good move. Especially since the moving average reward takes a couple iterations to decay.

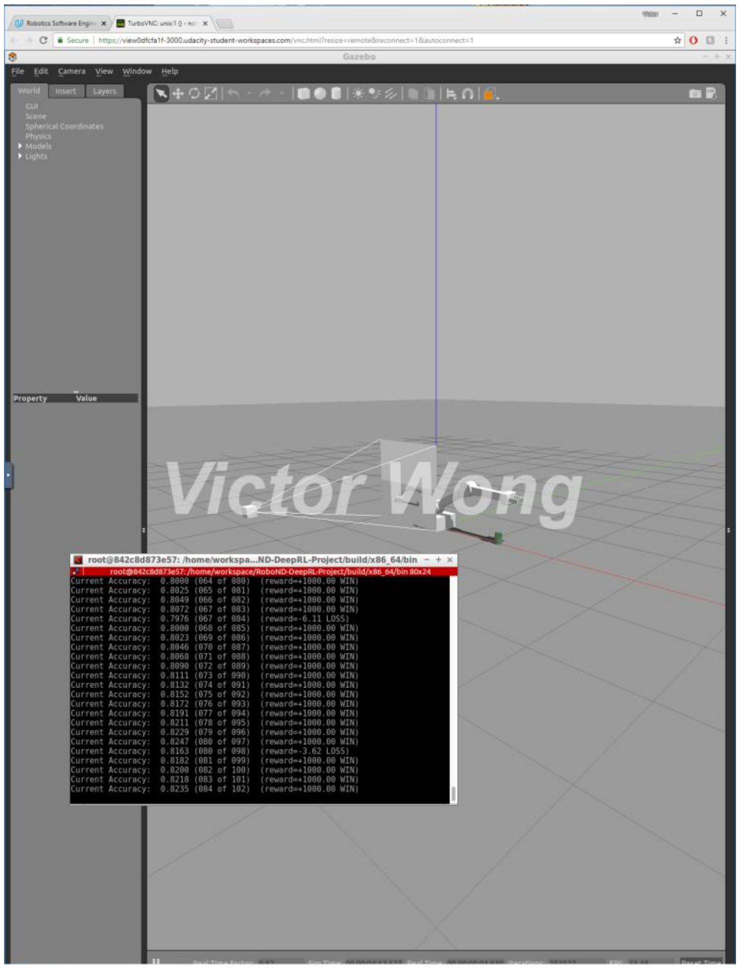
Results

I was able to hit just a little over a 90% success rate on the first task and an 82% success rate on the second task. The first one goal was pretty straightforward to hit, since it learned to just learned to apply torque to the first joint and swing the arm against the object. It didn't have to learn how to manipulate the 2nd joint so it was easier to get a higher precision.

The second goal took a lot more trial and error to achieve. It boiled down to getting good training sessions in the beginning and locking those actions in. I could have lowered the learning rate and randomness decay even more but it would accrue a lot more failures in the beginning which would take proportionally more wins to offset.


Robotics Software Engine
TurboVNC: unix1 ()
Secure | https://view0dfcfa1f-3000.udacity-student-workspaces.com/vnc.html?resize=remote&reconnect=1&autoconnect=1
Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Models
Lights
Property Value
Victor Wong
root@b13bfd8fc893: /home/workspa...ND-DeepRL-Project/build/x86_64/bin
root@b13bfd8fc893: /home/workspace/RoboND-DeepRL-Project/build/x86_64/bin 80x24
root@b13bfd8...
Gazebo
04:50

```
-0.0351598, 0.20297, 0.452389
-0.0261787, 0.22225, 0.648461
-0.021196, 0.211018, 0.838283
-0.0157472, 0.205139, 1.02768
-0.0103727, 0.199942, 1.21724
-0.00547407, 0.192611, 1.40438
-0.00224438, 0.17562, 1.57776
6.57776
Current Accuracy:  0.9009 (418 of 464)  (reward=+5.00 WIN)
-0.0758251, 0.0482333, -0.0275918
-0.0683282, 0.0885654, -0.00735461
-0.0606334, 0.122151, 0.0541631
-0.0527418, 0.150332, 0.151753
-0.0440337, 0.17832, 0.286039
-0.0349576, 0.203163, 0.454244
-0.0259823, 0.222365, 0.650627
-0.0209975, 0.211124, 0.840754
-0.0155623, 0.205134, 1.03033
-0.0102146, 0.199759, 1.21987
-0.00539578, 0.191932, 1.40641
-0.0022279, 0.174665, 1.57884
6.57884
Current Accuracy:  0.9011 (419 of 465)  (reward=+5.00 WIN)
```



```
root@842c8d873e57: /home/workspace/RoboND-DeepRL-Project/build/x86_64/bin 80x24
Current Accuracy:  0.8000 (064 of 080)  (reward=+1000.00 WIN)
Current Accuracy:  0.8025 (065 of 081)  (reward=+1000.00 WIN)
Current Accuracy:  0.8049 (066 of 082)  (reward=+1000.00 WIN)
Current Accuracy:  0.8072 (067 of 083)  (reward=+1000.00 WIN)
Current Accuracy:  0.7976 (067 of 084)  (reward=-6.11 LOSS)
Current Accuracy:  0.8000 (068 of 085)  (reward=+1000.00 WIN)
Current Accuracy:  0.8023 (069 of 086)  (reward=+1000.00 WIN)
Current Accuracy:  0.8046 (070 of 087)  (reward=+1000.00 WIN)
Current Accuracy:  0.8068 (071 of 088)  (reward=+1000.00 WIN)
Current Accuracy:  0.8090 (072 of 089)  (reward=+1000.00 WIN)
Current Accuracy:  0.8111 (073 of 090)  (reward=+1000.00 WIN)
Current Accuracy:  0.8132 (074 of 091)  (reward=+1000.00 WIN)
Current Accuracy:  0.8152 (075 of 092)  (reward=+1000.00 WIN)
Current Accuracy:  0.8172 (076 of 093)  (reward=+1000.00 WIN)
Current Accuracy:  0.8191 (077 of 094)  (reward=+1000.00 WIN)
Current Accuracy:  0.8211 (078 of 095)  (reward=+1000.00 WIN)
Current Accuracy:  0.8229 (079 of 096)  (reward=+1000.00 WIN)
Current Accuracy:  0.8247 (080 of 097)  (reward=+1000.00 WIN)
Current Accuracy:  0.8163 (080 of 098)  (reward=-3.62 LOSS)
Current Accuracy:  0.8182 (081 of 099)  (reward=+1000.00 WIN)
Current Accuracy:  0.8200 (082 of 100)  (reward=+1000.00 WIN)
Current Accuracy:  0.8218 (083 of 101)  (reward=+1000.00 WIN)
Current Accuracy:  0.8235 (084 of 102)  (reward=+1000.00 WIN)
```

Future Work

Exploring lower learning rates and longer training times would make the solution more robust. Also adding control to the gripper's fingers or unlocking and restricting the base joint would allow the robot to learn to swing into the object which would be a more robust way of colliding with only the object and not the ground.