CS 6301 - Intro to Robot Manipulation & Navigation December 13, 2023

Final Project

Team Members: Abhishek Chauhan, Shreeprasasd Sonar, Vedang Prasad Wartikar

About the Project

In our project, our main goal was to explore reinforcement learning algorithms and apply these algorithms to pickup specific shaped objects. While exploring the pybullet library, we found some examples that we could learn from. We based out setup one of these examples that has a Kuka Robot arm and a sample Gym environment. We have devised a reward function that rewards if the user-specified object is picked up, else if other object is grasped, a penalty is applied. We have added specific shapes like cube and cuboid for the robot arm to attempt pickups.

How to Execute

Directory

Our directory structure looks like,

```
|-- config.json
|-- environment.yml
|-- metrics.csv
|-- models
|   |-- DQN_25k_cube
|   |   |-- best_model.zip
|   |   |-- monitor.csv
|   |   |-- tensorboard_DQN_12-08-2023_20-47-58
|   |       |-- DQN_1
|   |           |-- events.out.tfevents.1702090080.Abhishek-Machine
|-- references.txt
|-- src
    |-- env
    |   |-- kuka_diverse_gym_env.py
    |   |-- kuka.py
    |-- plot.py
    |-- test.py
    |-- train.py
```

Files

The *environment.yml* (created from our conda env) contains all the requirements needed to run this project.

For training the DQN / DDPG model, *config.json* should be modified. For example, if the user wants to train the DQN model on a cube for *25,000* timesteps, the following configuration is required:

- specific_object: cube_small.urdf
- algorithm: DQN
- timesteps: 25000

After this, one can execute the training by running - **python train.py**. It makes use of the PyBullet's Gym environment which includes custom objects, custom written reward function and the Kuka robot configuration. It will save the model with its timestamp in the models/ directory.

For testing a model, on say *100* timesteps, following changes in the *config.json* are needed:

- model_dir: ../models/DQN_75k_cube/best_model.zip
- test_episodes: 100

After this, one can execute the testing by running - **python test.py** which will print out the success ratio of the 100 grasp attempts. Since the training of most of the models takes more than *10+* hours, we have added a few models so that it's easy to test on those - DDPG *(10k, 25k)* and DQN *(10k, 25k, 50k, 75k)* timesteps.

The script *plot.py* plots the average success ratios of the DQN model from the *metrics.csv* which is generated by *test.py*.