ME5418 OpenAI Gym Report
Group 1

1 Introduction

In this report, the gym environment `PushEnv` would be introduced, which alighs with the objective of the previous project proposal: precisely positioning various objects through pushing, with a primary focus on preventing the toggling of objects while achieving the manipulation objectives. Specifically, the `PyBullet` physics engine is employed to simulate object dynamics and collision events.

2 Environment Description

2.1 Overall Configuration

In this environment, the control frequency (simulation step) is set as 20 Hz, and the simulated image is generated using the OpenGL renderer provided by `PyBullet`.

2.2 State Space

- **Camera Image**: An array of type `np.uint8` with dimensions of $240 \times 240 \times 3$, which represents the visual observation captured by a camera within the environment.
- **End-Effector Position**: An array of type `np.float32` with dimensions of 3×1 and describes the position of the robot's end-effector.
- **Object Position**: An array of type `np.float32` and indicates the position of the object that needs to be pushed.
- **Goal Position**: An array of type `np.float32` with dimensions 3×1, representing the target position of the object.
- **Touch Information**: An boolean variable that indicates whether the end-effector is in contact with the object.

2.3 Action Space

The action space of the environment is defined as a continuous vector of shape 2×1, representing the instantaneous velocity of the end-effector in the (x, y) plane. The range of each dimension is normalized [-1, 1], corresponding to the velocity ranging from -2 (m/s) to 2 (m/s).

2.4 Episode Randomization (the `reset()` method)

When the environment is reset, the positions of the robot, object, and target location are initialized and visually represented within the environment. In this environment, two candidate objects are available: a cube and an upright cuboid. At the start of each episode, one of these objects is randomly selected.

2.5 Episode Step (the `step()` method)

At each time step, the robot's end-effector is controlled to follow the input velocity, and the robot's inverse kinematics and dynamics are calculated using the built-in solver in `PyBullet`.

2.6 Termination Conditions

The termination conditions and the corresponding reward configuration are as follows:

- **Timeout**: The timeout signal will be triggered after 300 steps of simulation. At the last time step, if the robot is touching the object, a reward of 2 is given; if the object is not being touched, a reward of -1 is given.
- **Object being toppled**: If the absolute value of the x or y component of the object's XYZ Euler angle exceeds $\pi/4$, the episode is terminated with a reward penalty of 1.
- **Object falling down**: If the z component of the object's position is below -0.1 (m), the episode is terminated with a reward penalty of 1.
- **Goal reached**: If the distance between the object and the target is within 0.1 (m), the episode terminates with a reward of 50.
- **End-effector out of workspace**: If the position of the robot's end-effector is detected outside the desk, the episode terminates with a reward of -5.

2.7 Reward Shaping

In addition to the reward assigned at the terminal state, we also implemented reward shaping as outlined below:

1. Time penalty of -0.005 per step
2. when the object is moving, a reward that is proportional to the dot product between the object's velocity vector and the displacement vector from the object to the target will be given
3. If the end-effector is detected not in contact with the object, a reward that is proportional to the dot product between the end-effector's velocity vector and the displacement vector from the end-effector to the object will be given

3 Reflection and Conclusion

In comparison with the environment proposed in the previous project proposal, the following modifications were made while constructing the OpenAI Gym environment:

1. To facilitate testing and troubleshooting of the learning agent in the later stages of this project, we expanded the agent's observations to include not only the originally defined visual and touch information but also the exact positions of the object and the goal.
2. We restricted the robot's movement to the (x, y, 0) plane rather than the entire space. This constraint was implemented to narrow down the search space and potentially expedite the algorithm's convergence.
3. To alleviate the problem of reward sparsity explained in the previous project prosoal, where no rewards are provided when the end-effector is not in contact with the object, we addressed this concern by introducing rewards that encourage the end-effector to approach the object, as elaborated in Section 2.7.

In conclusion, this report presents the PushEnv environment, offering a challenging scenario for training a manipulator to push an object on a desk. When training deep reinforcement algorithms in this environment, the learning agent is expected to utilize visual and touch observations to accomplish the primary objective: successfully pushing the object to the designated target area without toppling the object. This environment serves as a valuable platform for exploring the capabilities of reinforcement learning techniques in complex manipulation tasks.