Final Project / Assigment 4 (due 6/11)

Welcome to the final assignment of CSE 276F. The goal is to build a custom robotics environment and use modern reinforcement learning tools to solve and make progress on that environment. The final submission is a short report detailing your environment and experiments.

A simulated robot environment is generally comprised of the following elements:

- Defining what robot(s) to use (we recommend just sticking with panda robot)
- Defining the scene, what objects are loaded in, which ones can be moved around and which cannot
- Defining the initialization conditions of all objects/robots. Where do they spawn, are their positions randomized?
- Defining the success and/or failure conditions of the task. You need to code to evaluate the environment state at each time step to determine if the task is solved or not.
- Defining a dense reward function. For this report a dense reward function is required. Dense reward functions can be used to verify a task is solvable, and also learn neural network policies via RL to control the robot to solve the task.

Report Requirements (15 pts):

- Description of the robot environment(s) you have built (5 pts):
  1. Share a photo of what it looks like when reset randomly and a photo of what it looks like when solved.
  2. Describe how objects and robot(s) are initialized and randomized.
  3. Describe what the success and fail conditions are for the task.
  4. Further describe what do you include as part of the observations by default.
    - 5 pts if the task does not exist in ManiSkill and is different enough compared to existing tasks (Check here: https://maniskill.readthedocs.io/en/latest/tasks/index.html). If you are unsure you can ask on piazza. Generally making a task where the solution trajectory looks visually different to existing tasks is sufficient.

        - Example of too similar tasks: Inserting a peg inside a box (exists) compared to inserting a peg inside a cylinder.
        - Example of tasks that different enough: Pick a cube to reach a goal (exists) compared to push a cube to hit some other cube (does not exist).
        - Need ideas? Check out RLBench and Metaworld https://github.com/Farama-Foundation/Metaworld, https://github.com/stepjam/RLBench. Some concrete task ideas include:
            - Grab a tool and manipulate a cube from far away from the robot to some goal
            - Pick an object from one box and place/drop it into another box
            - Grab a plate and place it on a dish-rack object.
            - Rotate a wine bottle up-right
        - Remember that the task you build must include some kind of robot manipulation (otherwise its generally too easy).
    - 2 pts if the task does not include any robot manipulation (e.g. the task is to make the robot arm reach a goal instead of controlling an object).
    - 0 pts if this is not done.
- Reward Verification / RL (5 pts): Run an RL algorithm such as PPO (recommended) and verify the task is solvable using the reward function you designed with state based observations. Visual RL is also okay but it will take longer to do. Provide a figure showing the success rate and return curves over training time (training time being environment samples here). Furthermore, detail in this section the reward function design, explain how each component/stage encourages a RL agent to learn to solve the task.
    - 5 pts if PPO successfully solves the task you designed, and there are correctly formatted figures.
    - 3-4 pts if PPO successfully/partially solves the task but figures/some explanations are lacking.
    - 2 pts for writing a reward function and explaining how it works
    - 0 pts if this is not proven/is empty.

- Experimentation: A huge part of robot machine learning research centers on experimentation these days, we want you to learn how to conduct a proper experiment and report it. You can pick at least 1 component of your RL algorithm and/or robotics environment to modify and vary. Then conduct an ablation study on this component and share the results of the experiments. How does the variation of this component affect results?
    - 5 pts if a clearly written ablation study is made to show the performance differences when different changes are made to the environment. There is also clear discussion about potential reasons why there are changes in performance.
        - Some ideas you can try: Experimenting with different activation functions, neural net architectures, loss schedulers, noisy simulation observations etc.
    - 3-4 pts if the figues/graphs have some minor issues and/or some discussion does not logically follow the results.
    - 1-2 pts if the section is lacking significant discussion and/or figures to backup results.
    - 0 pts if the section is empty.

We require you to include in the end of your report a list of python commands that can directly reproduce your report. Namely

1. A demo_random_actions.py file that when run, will save a video of a robot taking random actions in your environment
2. A train.py file that when run will solve your task with PPO and generate the return and success rate curves (it is okay if this file is the exact same as publicly available code, but you may need to tune hyper-parameters).

If we find that your results are not mostly reproducible, you will receive 0 points on the results section.

Note you only need to make a single environment. We also do not expect massive ablation studies, focus on writing a high quality report covering a few hyper-parameters / environment design decisions.

If you are concerned about project scope / what is sufficient in terms of a report, you may ask on Piazza.

Code To Help Get Started

Follow the tutorial at https://maniskill.readthedocs.io/en/latest/user_guide/tutorials/custom_tasks/index.html to learn how to build custom tasks/environments

If you have limited compute be wary of how difficult of a task you make. Anything beyond the difficulty of PickCube-v1 might give you too little time to finish this project. We recommend copying the PushCube task code as a starting point and modifying that to make your task.

For tasks you want to make that involve non primitive shapes (like a wine bottle, a dishrack), you can try and find .glb files of those objects online.

For PPO code we recommend you use: https://github.com/haosulab/ManiSkill/tree/main/examples/baselines/ppo. First try running one of the baselines (e.g. solve PickCube-v1 with PPO) before trying it on your own task / modifying the PPO training script. You are allowed to copy/paste the PPO code directly. Note that there are 2 state-based PPO training files, `ppo.py` and `ppo_fast.py`. `ppo_fast` trains faster but is unstable and may not work on your system.

For those without their own GPUs, please start early to gain access to UCSD Datahub resources. The GPUs there are far faster than Google Colab and can help accelerate your progress in the assignment.

Submission Requirements

- PDF of your report. Try to limit it to 5 pages max.
    - Ensure the exact python commands to run your environment (e.g. `python demo_random_action.py`) and to train a PPO agent that solves the environment (e.g. `python train.py --env-id your-custom-env-v1`) are provided in the report.
- Zip of your code / notebook

The report does not need to be structured like a conference paper although you are welcome to do so.