Project Status Update: Reinforcement Learning in Robot Stacking Problem

Wei Lin Pai wpai@stanford.edu
Irmak Sivgin isivgin@stanford.edu
Yu Wei Lin ywlin@stanford.edu
AA228/CS238, Stanford University

1. Goal

Our project aims to develop a reinforcement learning (RL) solution for a robot manipulation task where a robotic arm learns to stack rectangular objects (i.e., boxes) on top of each other (Zhang et al., 2020) accurately, which can be potentially applied to the field of warehouse robotics. The objects will be constrained on the gripper of the robot arm (i.e., perfect grasping) to focus on the decision making process for the stacking problem only. The reward is divided into two parts: When the robot successfully stacks the objects, there is a positive reward proportional to the stability and balance of the objects we measure with packing efficiency. When the robot places objects next to each other (or the objects fall down from the stack), there will be a negative reward, which again can be scaled by the packing efficiency metric as this will lead the robot to place objects closer together (Xiong et al., 2023).

2. Problem Formulation

2.1 Assumptions

We assume the robotic system has some cameras to monitor the bounded area for stacking (e.g. side view camera) and the image processing yields approximate positions and dimensions of the rectangular objects. Thus, we formulate the problem as an MDP with uncertainty in the state of the objects. For our implementation in the simulation system, we use the readouts from the simulation environment for 3D coordinates / dimensions of the objects, and add Gaussian noise to model the uncertainty stemming from the camera-based monitoring system (which we don't simulate). Secondly, we note that the problem is symmetric in (x, y) coordinates (z axis denotes the height). Therefore, we assume a constant y coordinate, which corresponds to a single layer placement of objects (as in (Zhang et al., 2020)) and fix the depth dimension of the generated objects. This leads to simplifications in our action and state spaces, where the y coordinate becomes obsolete (and can be ignored).

2.2 MDP formulation

1. **State (S)**: We handle the state space in two parts. It includes the dimensions and positions (ground truth from simulation plus noise) of all previously placed objects as well as the new generated object dimensions.
2. **Action (A)**: Discretized (target) gripper position (x, z). The set of actions $\mathcal{A} = \{(x_1, z_1), (x_1{+}\Delta x, z_1), \ldots, (x_1{+}(n{-}1)\Delta x, z_1), (x_1, z_1{+}\Delta z), \ldots, (x_1, z_1{+}(n{-}1)\Delta z), \ldots, (x_1{+}(n-1)\Delta x, z_1 + (n-1)\Delta z)\}$ (over an n-by-n grid).
3. **Reward (R)**: $r = \eta V_{boxes}/V_{boundingbox}$ where η is a large positive constant, when the

objects are stacked; $r = -V_{boundingbox}/V_{boxes}$ otherwise. There is penalty added to r when there's collision.
4. **Uncertainty**: The position (x, y) and dimensions (w, h) of the objects are noisy to model the uncertainty in camera perception. Also, the dimensions of the next object are unknown (random height h and width w, fixed depth d).

2.3 Methodology

Since we have discretized action and state space, by limiting the number of objects and the bounds of the action space grid, we can apply Q-learning to this problem as a baseline:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

Our main method is using a DQN, where a fully-connected MLP will be parameterizing the Q-function as $Q_\theta(s,a)$. The parameters will be updated with

$$\theta \leftarrow \theta + \alpha(r + \gamma \max_{a'} Q_\theta(s,a') - Q_\theta(s,a))\nabla_\theta Q_\theta(s,a).$$

2.4 Simulation Environment

The stacking environment is implemented in PyBullet, consisting of a UR5 robotic arm and randomly generated rectangular objects with fixed width (0.1m) but varying length and height (0.1m-0.2m). Object manipulation is achieved through PyBullet's constraint system, where a fixed constraint is created between the robot's end-effector and the target object during grasping. To simulate real-world perception uncertainty, Gaussian noise ($\sigma =$ 0.005m) is added to object position measurements. The reward function combines stacking efficiency (ratio of object volume to bounding box volume, weighted at 0.7) and collision penalties (based on contact forces, weighted at 0.3), encouraging both stable stacking and gentle manipulation.

3. Current Progress

1. Changed the problem formulation and solidified the approach.
2. Set up simulation in Pybullet with UR5 robot (see Fig. 1).

4. Timeline

Week 9 (11/18 - 11/24): Complete simulation environment setup and start exploration on the simulation.
Week 10 (11/25 - 12/1): Curate a comprehensive dataset and save as a csv file for the training. Get baseline scores with Q-learning, start training the DQN.
Week 11: Train DQN and explore different architectures. Relax assumptions about fixed y if time permits, or try adding more flexibility to $\mathcal{A}$ by adding rotation of the gripper. Write the report.

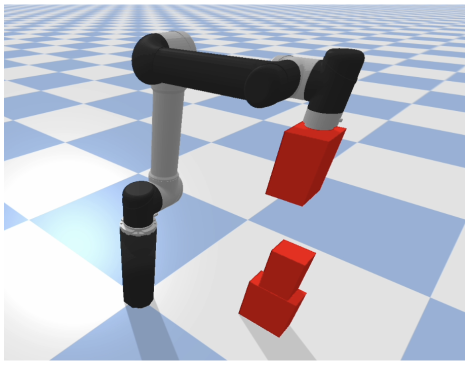

Figure 1: Simulation environment for robot stacking

References

Heng Xiong, Kai Ding, Wan Ding, Jian Peng, and Jianfeng Xu. Towards reliable robot packing system based on deep reinforcement learning. *Advanced Engineering Informatics*, 57:102028, 2023.

Junhao Zhang, Wei Zhang, Ran Song, Lin Ma, and Yibin Li. Grasp for stacking via deep reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2543–2549. IEEE, 2020.