16-662 Robot Autonomy 2020 Spring Final Report Team 4 - Play Jenga

Eva Mo(szuyum), Shaun Liu(hsuanchl), Alex Wu(chichiawu)

Motivation

In this project, we are exploring an automatic resetting procedure for Jenga in simulation using the RLBench environment. Robot resetting would allow human players to enjoy the game without having to perform the tedious task of rebuilding the tower. Furthermore, the process of training robots to perform forward pass can be sped up and done automatically when utilizing our resetting approach. The reinforcement learning (RL) pipeline we implemented could be applied to various scenarios, and our results could act as a basis and be scaled to different applications.

Overview of Approach

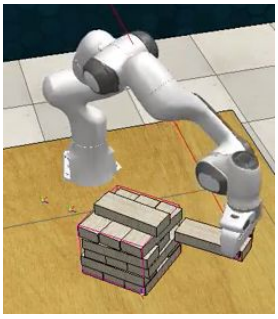

Figure 1. Forward Pass

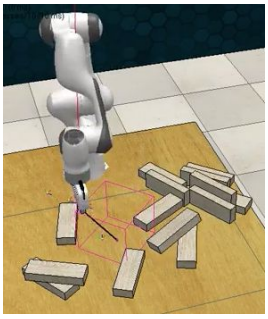

Fig 2. Clearing Procedure

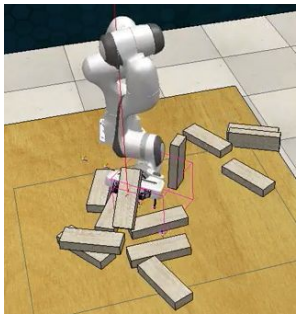

Fig 3. Resetting Phase

We split the task into three sections - the forward pass to play Jenga, the clearing procedure to knock down the tower and clear the table, and the resetting process to rebuild the tower. Visualizations of these sections are shown in Figures 1, 2, and 3. In order to simplify these tasks, we decided to apply reinforcement learning only during the resetting phase. The Franka robot starts the forward pass and tries to remove blocks and place them on the top for as long as possible. As soon as the tower falls or the robot is unable to find a path, the robot knocks down the entire tower, enters the resetting procedure, and starts the reinforcement learning process.

Forward Pass:

In the forward pass, we implement a state machine to have the robot perform different tasks at each state. First, the robot randomly chooses a valid cuboid to poke. Constraints are added to prevent the selection of blocks at the lowest level, where the robot gripper would hit the table, and also at the highest level, where it is against the rules of the game. After a valid cuboid is chosen, the robot first pushes the cuboid from one side and moves to the other side to pull it out. The block is then placed on the top level slightly protruding from the edge of the tower. The reason for this protrusion is to leave space for the gripper to hold onto the block. After releasing the block, the gripper will push it inward to align the edges. The way trajectories are planned is by having multiple waypoints for the robot to follow and providing the planner with absolute end-effector poses. Additionally, in order to prevent the robot from colliding with the Jenga tower, we sometimes separate one trajectory into multiple sections. The exit condition for the forward pass is when the robot could not find a possible path, or when the tower falls.

Clearing Procedure:

To prepare for the resetting phase, the robot arm will sweep through the workspace to knock down any remaining portions of the tower. Blocks will then be pushed aside to provide a clean area where the tower will be rebuilt. After the clearing procedure, the resetting phase starts and the robot will try to build up the Jenga tower.

Resetting:

With cluttered blocks on the workspace, the Franka robot first will search for a random cuboid and grasp it. With a block on the gripper, the agent in the RL environment will pick an action according to the current policy, which would be an absolute pose of the end effector consisting of position (x, y, and z) and rotation in quaternion form (qw, qx, qy, and qz). We expect the robot arm to learn to transport the cuboid to the desired pose in order to rebuild the whole tower. After the cuboid is moved to the desired position, the gripper is released.

To implement RL in simulation, we use the RLzoo package, which provides us with the environment and algorithm. The algorithm implemented is Clip Proximal Policy Optimization (PPO). Our state space is currently the robot arm configurations and our action space is the absolute pose of the end effector. The robot is rewarded if the sum of the height of all cuboids increases after an action. The episode terminates when the sum exceeds a certain limit, indicating the tower has been built completely.

Key Challenges

Our two key challenges are accurately manipulating cuboids and setting up the reinforcement learning environment. Manipulation tasks include poking, grasping, moving, and placing. The constraints of the robot arm, as well as inverse kinematics calculations in PyRep, often causes the error “no path found”. The position and orientation of the tower are initialized randomly, and in some cases, the tower is too far for the Franka arm to reach. Since the cuboid poses are obtained from the noisy cameras, there is variance when performing the same movements, which causes intermittent failures for certain tasks.

Setting up the environment for reinforcement learning was also convoluted. Many functions that we require are not included inside the libraries provided. In addition to our script, we had to modify the source code in RLBench, RLzoo, and PyRep. Much time was spent on familiarizing ourselves with the environment and how packages interact with each other.

Methods Implemented

One of the reasons that the robot could not find a path was due to unreachable waypoints. When the Jenga tower is initialized too far or too close to the robot, the waypoints we specified, such as pulling out a cuboid, can exceed configuration limitations. To resolve this issue, we restricted the spawn boundaries (location and orientation) during random initialization sampling to ensure all poses required are reachable. When a path can not be found due to inverse kinematics issues, we force the robot to enter the clearing procedure to reset the environment.

When specifying our waypoints, we often have to move the end effector along a direction with respect to a cuboid frame. Therefore, we had to calculate the transformation matrix between frames to convert the movements into the robot frame to maneuver the robot properly in the workspace. To accurately manipulate the cuboids, we call our state to obtain object poses 50 times in a row and take the average to filter noise. This is done whenever we need information about the environment. To calculate the correct pose to stack the cuboid on the top of the tower, we need to determine the number of cuboids on the highest level. If there are one or two blocks on the top of the tower, we stack the next block in the same level and with the same orientation as the blocks on the top. Otherwise, we stack the block one level higher and rotate by 90 degrees.

When setting up the RL environment, we dug deep into the libraries, including RLBench, RLzoo, and PyRep. To implement the custom reward, we had to design a custom success condition class followed by a reward function in the RLBench library. Since our action space is different from the default, joint velocity, we also had to define a custom exploration sampling with constraints in the RLzoo library. Another challenge is that the robot sometimes releases the cuboid in mid-air. The cuboid being released does not reach the table at the time we query the system for the sum of cuboids, meaning that the target cuboid is still in the air and would contribute a relatively high value of height. Our agent will receive a reward when it is not supposed to. Our solution to this issue is to move the robot arm to a homing position before calculating the reward so that there is sufficient time for the target cuboid to fall to the table.

Demo/Evaluation

We have successfully implemented the forward pass stack which enables the Franka robot to poke and grasp a random cuboid and to stack and align it on the top of the tower. The process can be performed smoothly and without collision as shown in our demo video. However, in the resetting part of our demo, the robot arm is moving cuboids to relatively random locations. Our robot does not perform well because the parameters of the network have not been fine tuned and the RL model has only trained for a short duration. We have successfully set up the training environment for the resetting task, but further training and adjustments are required.

Future Work

For the forward pass, we will improve grasp precision and make path planning more efficient. This will provide us with a more robust building process capable of grasping the bottom level of the Jenga tower and aligning the cuboids more accurately. For resetting, we need to resolve the issue that cuboids sometimes stick to the gripper in simulation even though the gripper is open. Fixing this problem will benefit our training process since for each RL step, the robot arm gains a different experience with a different cuboid. We also plan to train the network for longer periods of time and evaluate the performance. Different reward functions will be tested and hyperparameters (discount factor, batch size, max iterations) will also be tuned to make the model converge faster.