CPSC 533v Project Report: RL-Based Construction Robot Arms

Lei, Huang[*1,2], Weijia, Cai[*1,3]
[1] University of British Columbia, Canada
[2] lh2921@student.ubc.ca
[3] andycai@student.ubc.ca

Abstract: The construction industry faces challenges of skilled labor shortage, low productivity, continuous cost and schedule overruns, and unsafe working conditions for construction workers. The application of construction robots is a potential solution to alleviate these problems. Existing construction robotic solutions, such as bricklaying robots and grid drawing robots, are typically preprogrammed to follow specific sequences of instructions designed beforehand. Recent advancement in robot control algorithms (e.g., Imitation Learning) have enabled robots to learn sequences of optimal actions for executing a construction task from videos of construction experts demonstrating the process. However, these demonstrations are mostly collected on real construction sites, which can be costly and potentially dangerous to experts, especially when more than a few demonstrations are required. In this project, we propose a novel approach that leverages VR to generate expert demonstrations, allowing them to illustrate the procedure of a construction task using handheld controllers. In the meantime, both direct (e.g., coordinates) and indirect (e.g., images) parameters are extracted from the virtual environment to generate a control policy that imitates the behavior of the expert. We implement the proposed approach on the window installation task as validation, where the learned control policy is applied to a robot arm in the virtual environment.

1. INTRODUCTION

As one of the most traditional industries, the construction industry still heavily relies on manual human labor on site. It has been lagging behind in the adoption of advanced technologies in comparison with others such as manufacturing. To some extent, this leads to problems of low productivity, labor crisis, highly uncertain progress, etc. In other words, construction could benefit significantly from a higher level of applications of automation. In our project, we investigate how a robotic arm can be utilized to perform various construction tasks (e.g., window installation, welding, heavy lifting, etc.) by learning from human demonstrations via Reinforcement Learning (RL). To make the project more doable in such a short period, we decompose the whole work into some subtasks, including environment design and implementation, environment connection to Virtual Reality (VR), demonstrations collection in VR, and agent training using collected demonstrations. Due to time limitations, a few simplifications are also made. First, instead of high dimensional data like images, we directly adopt the states of joints of the robotic arm as observations. For actions, we also design such that the joints are maneuvered directly. Second, instead of learning from expert demonstrations, we let the robotic arm directly learn via deep reinforcement learning, because demonstrations are not collected yet. Third, the task of window installation in our case is simplified to random object reaching, which is a starting point of picking up a window and then installing it. The pictures and videos are in the project webpage: https://github.com/YESAndy/cpsc533vproject/blob/gh-pages/index.md.

2. RELATED WORK

RL has not been used in the construction industry until recently. Thus, there are only a handful of related papers. Liang et al. [1] applied Learning from Demonstration methods in teaching robot apprentices to perform quasi-repetitive tasks on construction sites. They collected a set of real-world human demonstration videos and translated the context to what could be understood from the robotic arm's perspective, based on which the control policy was generated. Their work was evaluated in the Robot Operating System (ROS) Gazebo Simulator using a KUKA mobile industrial robotic arm emulator. Apolinarska et al. [2] applied RL to robotic construction to assemble timber structures. Their control policy was trained entirely in simulation and deployed in reality. Belousov et al. [3] investigated how a robotic arm can be used to assemble a structure from predefined discrete building blocks autonomously via a combination of RL and planning.

The aforementioned studies either directly use RL algorithms or learn from video demonstrations to generate control policies.

In our work, the demonstrations are created in VR by experts, from which the information of all components is approachable. The robotic arm then learns a control policy from these demonstrations using Imitation Learning. Despite the long-term goal of our project described above, we investigate an RL-based method for the reaching task on a construction site [4].

3. METHODOLOGY

3.1. Setup

The RL training environment in our experiment is built mainly upon two packages: the NiryoOne Robot Arm package and Unity Machine Learning Agent Toolkit (ML-Agents). NiryoOne is a 6 Degree-of-Freedom (DoF) robot arm with a gripper as its end effector. Each joint has one DoF. To create the training environment, we first create a simplified construction workspace where there contains a robot arm, grass terrain, a wall frame, and a cube as the target to reach (as shown in Fig.1.). We then implement a control script to control the robot arm based on the action we designed. At every step, we select one joint and add force to it. ML-Agent is a package that enables users to create RL training environments based on Unity Game Engine and to train various RL algorithms within this platform. The workflow of ML-Agents is that the agent in Unity requests the policy from the trained RL algorithm and then applies the received action. It allows an action space to contain both discrete and continuous values simultaneously.

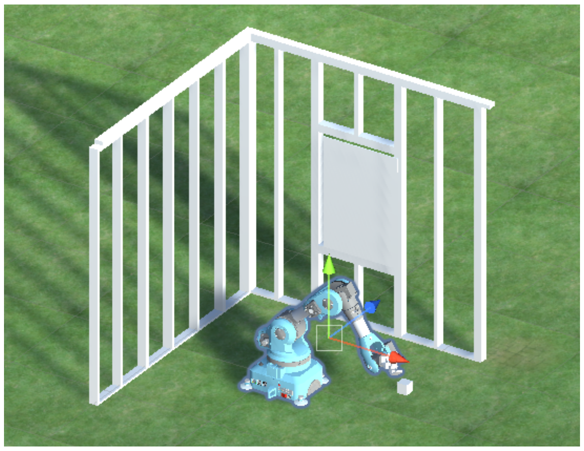

Fig. 1. Robotic arm on the construction site

3.2. Observations/States

We tested with two sets of observation spaces. One has 15 dimensions, including the 6-dimensional joint position and 9-dimensional gripper's velocity, position, and rotation. The other has 9 dimensions consisting of the gripper's velocity, position, and rotation.

3.3. Actions

The action space is 2-dimensional, consisting of a discrete index of the joint to move next and a continuous force applied to the selected joint. The force is calculated from the current value of the drive using the formula:

$$\text{Force} = \text{Stiffness} \times (\text{CurrentPosition} - \text{Target}) - \text{Damping} \times (\text{CurrentVelocity} - \text{TargetVelocity}),$$

3.4. Reward

Every time an action is taken, a reward of $-1.0/500$ is added. If the distance from the gripper to the target is below the pre-defined threshold, we consider the agent succeeds at the task. Accordingly, a reward of 150 is given and the episode ends. Otherwise, a reward of $-0.25 \times \text{DistanceFromGripperToTarget}$ is given.

3.5. Algorithms

Proximal Policy Optimization (PPO) algorithm is adopted in our task. PPO explores the biggest possible improvement step on a policy using the currently available data without stepping so far that we accidentally cause performance collapse [5]. It is a first order-based optimization method that is simpler to implement while behaves at least as well as TRPO. There are two main variants of PPO: PPO-Penalty and PPO-Clip. Here we adopt PPO-Clip. The equation for the policy update is:

$$\underset{\theta}{argmax} \underset{s,a\sim\pi_{\theta_k}}{E} [L(s,a,\theta_k,\theta)],$$

The loss function is:

$$L(s,a,\theta_k,\theta) = min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s,a),\ clip(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\varepsilon, 1+\varepsilon) A^{\pi_{\theta_k}}(s,a)\right),$$

where epsilon is a small hyperparameter which roughly indicates how far away the new policy is allowed to move from the previous policy. The overall algorithm is shown in Fig.2. below:

```
Algorithm 1 PPO-Clip
1: Input: initial policy parameters $\theta_0$, initial value function parameters $\phi_0$
2: for $k = 0, 1, 2, \ldots$ do
3:     Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
4:     Compute rewards-to-go $\hat{R}_t$.
5:     Compute advantage estimates, $\hat{A}_t$ (using any method of advantage estimation) based
       on the current value function $V_{\phi_k}$.
6:     Update the policy by maximizing the PPO-Clip objective:
           $\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left( \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t),\ \ g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$
       typically via stochastic gradient ascent with Adam.
7:     Fit value function by regression on mean-squared error:
           $\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left( V_\phi(s_t) - \hat{R}_t \right)^2,$
       typically via some gradient descent algorithm.
8: end for
```

Fig. 2. PPO algorithm

In our project, we set the initial epsilon to 0.5 and it decades linearly as the training proceeds. A neural network with two hidden layers with size 128 is used for the policy network. Lambda and learning rate are set to 0.99 and 0.001 respectively. The maximum step is 1 million and every episode has a maximum of 10,000 steps.

4. EXPERIMENTS AND RESULTS

In this project, we aim to study the performance of the robot arm under different observabilities. Motivated by this goal, we mainly conduct two experiments: using PPO and state1; using PPO and state2. From Fig.3. and Fig.4., the agent with the additional joint information has a higher cumulative reward over time and requires fewer steps to complete the task. From Fig.5. and Fig.6., however, the value loss of the agent with the additional joint information is higher than the other agent. The difference probably stems from the additional joint information added to the first agent. Moreover, Fig.7. shows that the agent with additional information indeed has a higher policy entropy, which indicates it contains more unknown information.

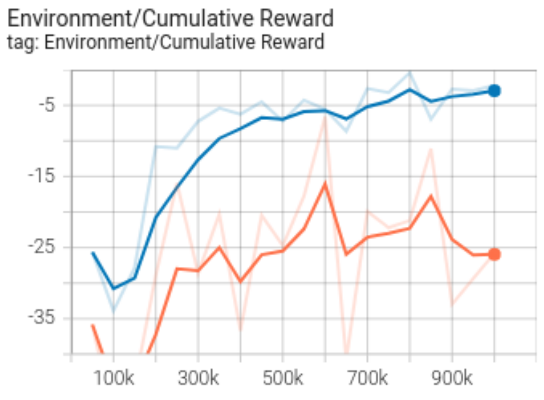

Fig. 3. Cumulative reward from the environment. The blue line is the agent with the additional joint information, the orange line is the other agent.

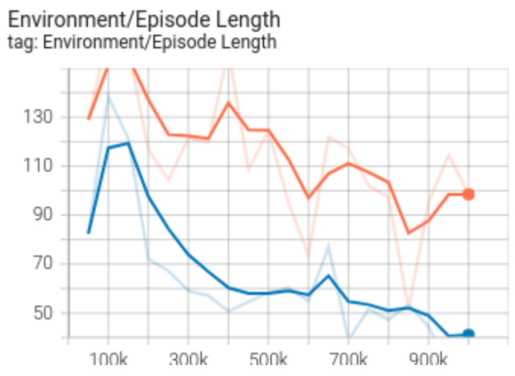

Fig. 4. Episode length for each episode

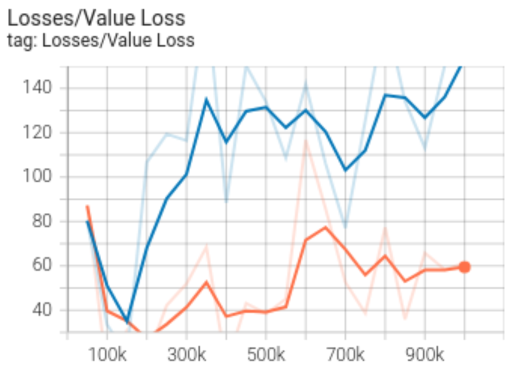

Fig. 5. Loss value for value network

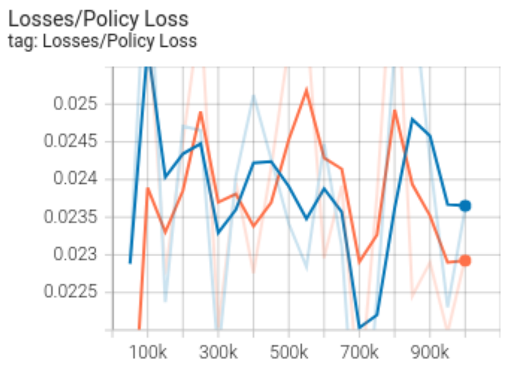

Fig. 6. Loss value for policy network

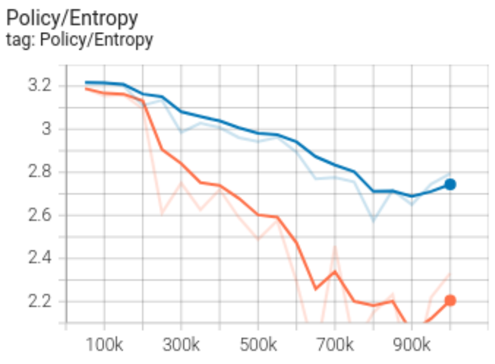

Fig. 7. Policy Entropy

5. DISCUSSION AND FUTURE WORK

Our experiments confirm the capacity of PPO in the robotic reaching task. The experiment results also show that the additional information help the robot agent to learn motion planning with respect to the designed reward. However, the training still cost lots of time (more 3 hours per agent) to converge. Based on the behaviors of the agent during the training, we observe that a good initiation of the robot arm can make the learning converge faster, where Imitation Learning might be of help. The work in this report is a small portion and a starting point of our long-term objective. In the future, we will replace the cube to be picked with real construction-related objects such as windows, bricks, timbers, etc. The task for agent to learn will be much more complicated than reaching (e.g., first pick up a window, then install it into a window frame), in which case, the observation space, the action space, and the reward function need to be studied in more depth. Following the simulation work, everything will then be transferred from simulation to real-world. Our robotic arms should be able to conduct construction tasks on real construction sites.

References

[1] Liang, Ci-Jyun, Vineet R. Kamat, and Carol C. Menassa. "Teaching robots to perform quasi-repetitive construction tasks through human demonstration." *Automation in Construction* 120 (2020): 103370.

[2] Apolinarska, Aleksandra Anna, Matteo Pacher, Hui Li, Nicholas Cote, Rafael Pastrana, Fabio Gramazio, and Matthias Kohler. "Robotic assembly of timber joints using reinforcement learning." *Automation in Construction* 125 (2021): 103569.

[3] Belousov, Boris, Bastian Wibranek, Jan Schneider, Tim Schneider, Georgia Chalvatzaki, Jan Peters, and Oliver Tessmann. "Robotic architectural assembly with tactile skills: Simulation and optimization." *Automation in Construction* 133 (2022): 104006.

[4] Aumjaud, Pierre, David McAuliffe, Francisco Javier Rodríguez-Lera, and Philip Cardiff. "Reinforcement learning experiments and benchmark for solving robotic reaching tasks." In *Workshop of Physical Agents*, pp. 318-331. Springer, Cham, 2020.

[5] Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. "Proximal policy optimization algorithms." *arXiv preprint arXiv:1707.06347* (2017).