Proceedings of the 2022 Winter Simulation Conference
B. Feng, G. Pedrielli, Y. Peng, S. Shashaani, E. Song, C.G. Corlu, L.H. Lee, E.P. Chew, T. Roeder, and P. Lendermann, eds.

ACCELERATING TRAINING OF REINFORCEMENT LEARNING-BASED CONSTRUCTION ROBOTS IN SIMULATION USING DEMONSTRATIONS COLLECTED IN VIRTUAL REALITY

Lei Huang
Zhengbo Zou

Department of Civil Engineering
University of British Columbia
6250 Applied Science Lane
Vancouver, BC V6T 1Z4, CANADA

ABSTRACT

The application of construction robots is crucial to mitigate challenges faced by the construction industry, such as labor shortages and low productivity. Reinforcement learning (RL) enables robots to take actions based on observed states, improving flexibility over traditional robots pre-programmed to follow determined sequences of instructions. However, RL-based control is time-consuming to train, hindering the wide adoption of RL-based construction robots. This paper proposes an approach that utilizes expert demonstrations collected from virtual reality to accelerate the RL training of construction robots. For evaluation, we implement the approach for the task of window pickup and installation on a virtual construction site. In our experiment, out of 10 RL agents trained using virtual expert demonstrations, 7 agents converge to an optimal policy faster than the baseline RL agent trained without demonstrations by around 40 epochs, which proves adding expert demonstrations can effectively accelerate the training of robots learning construction tasks.

1 INTRODUCTION

The global construction industry is anticipated to increase its output from US$10.7 trillion to US$15.2 trillion between 2020 and 2030 (OE 2021). Despite being an engine for economic growth, the construction industry has been afflicted with persistent challenges such as unsafe working conditions, poor productivity, and workforce shortage (Pradhananga et al. 2021). For example, the construction industry has the highest rate of fatal accidents, responsible for around 20% of occupational deaths in the U.S. (OSHA 2019) while only employing 6% of the national workforce (BLS 2022). The dangerous working conditions make construction less appealing to the next generations. It subsequently results in workforce shortages (Tonnon et al. 2017) along with the massive retirement of the aging workforce. More than 60% of the construction companies in the U.S. reported that they experienced difficulties in hiring skilled construction workers as of the fourth quarter of 2021 (USCC 2021). Insufficient workforce and physically intensive manual tasks further reduce construction productivity (Karimi et al. 2018).

Extensive surveys show that adopting robotics in construction to assist workers in various construction tasks has the potential to overcome challenges in the construction industry by improving safety, quality, and productivity (de Soto et al. 2018). Since the early 1980s, researchers have been seeking the integration of robots into construction (Haas et al. 1995) so that dangerous and arduous tasks can be conducted or facilitated by robots. In doing so, the responsibility of on-site workers can be altered from task operators to supervisors (Liang et al. 2021), which reduces the risk of workers being exposed to dangerous scenarios.

A traditional and straightforward type of construction robot is the pre-programmed robot (Liang et al. 2021). With specialists domain knowledge, pre-programmed robots can perform tasks with high success rates and precision (Lee and Chung 2008). Practical applications of pre-programmed robots, such as

assembly robots (Iturralde and Bock 2018) and painting robots (Seriani et al. 2015), have been deployed on construction sites. Although being the most prevailing form of robots in construction and having achieved success, pre-programmed robots suffer from the deficiency of being unable to adapt to unexpected scenarios (Cully et al. 2015). The reason for this deficiency lies in that the carefully handcrafted program is designed to work perfectly in specific scenarios (Carlson and Murphy 2005). For example, in the task of picking up a window panel, the robot arm would likely fail when the location of the window panel is slightly changed from its pre-programmed origin, since the robot is not aware of the surroundings and is unable to adapt.

The integration of reinforcement learning (RL) into construction robots promises more robust and generalized robots (Levine 2018; Eysenbach et al. 2021). RL equips robots with adaptability and flexibility. For example, a robot arm can adjust its trajectories according to a window panels location to finish the pickup task. Some researchers directly apply RL algorithms to train their robot arms to carry out tasks such as timber assembly (Apolinarska et al. 2021; Belousov et al. 2022). But direct RL via trial-and-error on robots usually requires days of training and considerable computing resources. Thus, methods based on imitation learning (IL) are developed to enable robots to learn a control policy by watching demonstrations (Liang et al. 2020). IL-based methods expedite the training process by extracting and leveraging information from expert demonstrations while reducing unreasonable explorations (Cheng et al. 2018). However, they require researchers to set up physical experiment scenes for expert demonstrations, which causes extra work and can be dangerous to demonstrators resulting from fatigue (Yu et al. 2019).

To address the issues of long training time, extra work for setting up experiment scenes, and possible dangers resulting from expert demonstrations collection, we propose an approach that boosts construction robots training process by adding additional demonstration data to the replay buffer used by the RL agent (i.e., a container that stores trajectories of experience when applying a control policy). The demonstrations are collected in a virtual reality (VR) environment (i.e., a virtual construction site), where the expert can conveniently demonstrate construction tasks using a pair of handheld VR controllers while avoiding possible dangers. During demonstrations, states (i.e., joint rotations of the robot arm and spatial information of the window panel) and actions (i.e., the changes of joint rotations of the robot arm) are extracted simultaneously. Each logged demonstration is treated as one trajectory and can be added to the replay buffer for RL training. The proposed approach is based on the hypothesis that the demonstrations will accelerate the training process of construction robot. To highlight, our main contributions are threefold:

- We build a virtual construction site in Pybullet and collect virtual expert demonstrations using an intuitive way, which is similar to directly maneuvering a robot arm in the real world.
- We develop a novel approach that augments RL with virtual expert demonstrations to accelerate the training of construction robots learning tasks. Our approach adds demonstrations to the replay buffer during training, following deep deterministic policy gradient from demonstrations (DDPGfD) (Vecerik et al. 2017), while replacing the demonstrations collected in the real world with virtual demonstrations collected in VR.
- We conduct the experiment in simulation and show that with demonstrations in the first N (N is searched in the range of 1 to 20) training epochs, the robot arm can learn the construction task faster with higher stability.

2 RELATED WORK

Based on the control policy (or a controlling program) being used, we can generally divide existing construction robots into pre-programmed robots and RL-based robots.

Pre-programmed robots usually require experienced civil, mechanical, and electrical engineers, as well as seasoned expert workers, to design the whole procedure of how a robot should perform a specific task in specific scenarios, which is then written as an executable program (Lauria et al. 2002; Christensen et al. 2008; King et al. 2014). As one of the most prevalent types of robots in construction, pre-programmed robots have achieved great success in many tasks such as bolting, transportation, maintenance, and assembly

(Salmi et al. 2018). For example, Bruzl, Usmanov, Svoboda, and Sulc (2016) used an optimizing algorithm to calculate an efficient painting trajectory for painting robots; Iturralde and Bock (2018) provided assembly robots with pre-programmed trajectories to assemble timber frames. Due to the nature of only taking defined actions, pre-programmed robots are unable to change their behaviors in response to various situations while working, even if the pre-programmed behaviors may cause failure or danger in new situations.

To develop adaptable construction robots with higher flexibility and intelligence, researchers started to apply RL for construction robot control (Xu and Garcia de Soto 2020). In contrast to pre-programmed robots executing deterministic pre-defined programs, RL-based construction robots learn a control policy of how to take actions based on the states abstracted from observations of the environment (e.g., the information of the robots location and surrounding objects) so as to finish a task and maximize the total expected reward (Sutton and Barto 2018; Ibarz et al. 2021). Thus, RL-based robots have the intelligence to make decisions step by step (Janner et al. 2021) according to the immediate and expected feedbacks from the environment. A well-trained RL-based construction robot can adapt to numerous situations. For example, a robot arm can always successfully pick up a window wherever the window is located (assuming that the window is within the range of the robot arm). Though RL is an effective paradigm for training robots, the training process is usually time-consuming and unstable, requiring millions of interactions between the agent and the environment (Nachum et al. 2018). Thus, to the best of our knowledge, all existing RL-based construction robots are first trained in simulation unanimously (Liang et al. 2020; Apolinarska et al. 2021; Belousov et al. 2022).

Because of the prosperity of off-site modular construction, a majority of RL-based construction robots focus on the task of assembly (Yin et al. 2019). For example, Apolinarska et al. (2021) used a variant of the Deep Deterministic Policy Gradient algorithm (DDPG) (Horgan et al. 2018) to train a control policy for the assembly of lap joints for custom timber frames in simulation and successfully deployed the control policy on a real robot arm. Belousov et al. (2022) used Twin Delayed DDPG (TD3) (Fujimoto et al. 2018) to train a control policy to place a building block (e.g., brick). Both RL-based methods used low-dimensional data (e.g., part positions and orientations, robot state, sensor readings, etc.) as the state information for faster training, but they still require hours or even days of training to learn a control policy that is adaptive and flexible enough to guide the robot arm to conduct a task.

Inspired by humans ability to master skills in a short time by watching demonstrations, researchers explored the idea of training robots to learn a control policy from demonstration videos (Liu et al. 2018). Based on this IL scheme, Liang et al. (2020) proposed a method for training a construction robot to perform quasi-repetitive construction tasks (e.g., ceiling tile installation) using visual demonstrations, which were collected on a real site set up in a laboratory. Since their robot learned from videos, high-dimensional images were used as states for the agent. Images usually are more informative than low-dimensional states, but they are also prone to distractors such as light variance and moving background (Zhang et al. 2021) and are more computation-intensive. Besides, collecting demonstrations in a physical scene can be costly and time-consuming.

To accelerate the training process, we propose a novel RL-based approach that trains a construction robot more efficiently by taking demonstration data as catalysts during training. To avoid irrelevant information affecting the control policy, we use low-dimensional states following Apolinarska et al. (2021) and Belousov et al. (2022). Furthermore, to avoid extra costs and the potential dangers during demonstrations, we built a virtual construction site for the expert to demonstrate using handheld VR controllers.

3 METHODOLOGY

Our approach collects trajectory data from expert demonstrations on a virtual construction site and uses it to accelerate the training of RL-based construction robots learning how to conduct construction tasks. The approach consists of three steps. First, as shown in Figure 1(a), we design and build a virtual construction site using Pybullet, which is a widely used real-time physics simulation engine that can simulate physical properties such as collision and dynamics of objects (Coumans and Bai 2021). Second, as shown in Figure

1(b), we collect virtual expert demonstrations using a VR headset and a pair of handheld VR controllers from human demonstrators. From the virtual demonstrations, we log the trajectory data in the form of state-action pairs. The data is further pre-processed and fed into the replay buffer for RL training. Third, as shown in Figure 1(c), we train the agent (robot arm) in the environment (i.e., the virtual construction site) using the policy gradient RL method (Sutton and Barto 2018). Meanwhile, we add trajectories of virtual expert demonstrations collected in the second step as additional experience replay into the replay buffer to accelerate training.

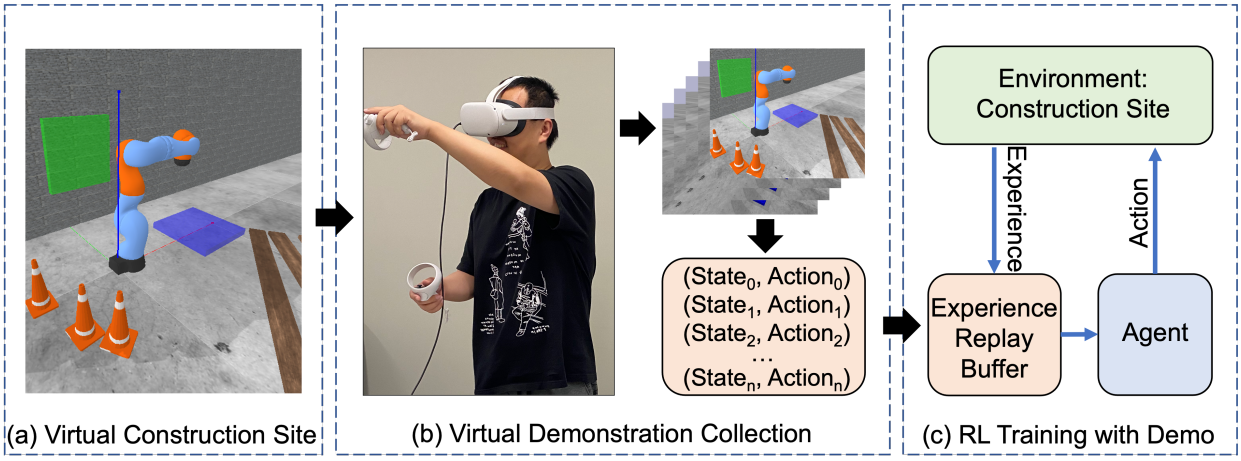


Figure 1: Overarching architecture of methodology.

3.1 Virtual Construction Site Design

The first step of our approach is to build a construction site environment in Pybullet (Coumans and Bai 2021). The virtual construction site contains several objects, including a robot arm, a blue transparent window panel on the ground, and a green transparent cuboid marking the target (i.e., the desired opening for the window panel) in which the window should be installed, with a few common objects on-site (e.g., reflective cones and wood planks). Figure 2 shows the overall virtual construction site in detail.

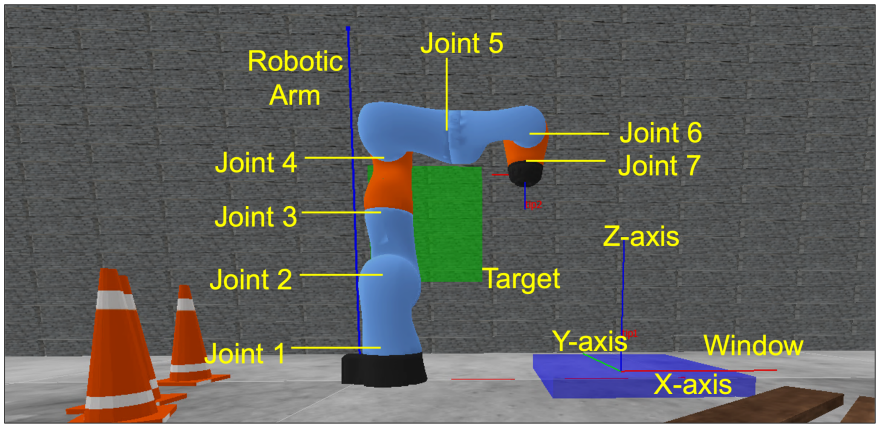


Figure 2: Virtual construction environment (robot arm, window, target, reflective cones, and wood planks).

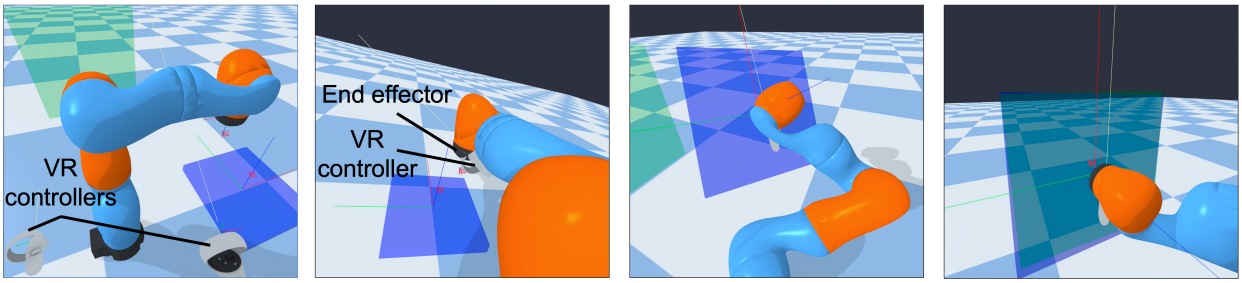
The robot arm model in the virtual construction site is the KUKA LBR iiwa robot because it is a widely used robot for executing tasks such as object pick and place, assembly, and human-robot collaboration (Kyjanek et al. 2019). The robot arm has seven degrees of freedom (i.e., seven joints) and has a load

capacity ranging from 7 kilograms to 14 kilograms. To make the space information (e.g., coordinates) simple, the base of the robot arm is initialized to be at the origin of the cartesian world coordinate system with no rotations along the x, y, and z-axis. To make the robot arm start with a normal pose like in the real world, the rotations of the fourth joint and the sixth joint are set to 1.57 radians and -1.57 radians, respectively, with the rest of the joints having zero rotations. The window and the target are located within the reach of the robot arm so that the robot arm can operate on the window.

We can control the robot arm in two modes. The first mode is using handheld VR controllers to manipulate the robot arm intuitively, which will be used in virtual demonstration collection. The other mode is to directly send actions to the robot, which will be used in training and testing the robot.

3.2 Virtual Demonstration Collection

The second step is to collect virtual expert demonstrations of the construction task (i.e., window installation in our experiment) in the as-built virtual construction site. Through a VR headset, the demonstrator views the virtual construction site along with two virtual handheld VR controllers. The demonstrator first moves a VR controller towards the mass center of the end effector of the robot arm. When the VR controller and the end effector overlap, they are automatically bound together so that the demonstrator can control the end effector by moving the VR controller. As shown in Figure 2, we add additional visual annotations for objects mass centers and axes to assist the demonstrator. The mass center of the window is annotated with tip1. The mass center of the end effector is annotated with tip2; the x, y, z-axis of the window is annotated with a red, green, and blue line, respectively. The demonstrator then moves the end effector to the center of the window. When the distance between the end effector and the window is under a threshold, the robot arm automatically picks up the window. Finally, the demonstrator moves the window towards the target position with proper orientation. Figure 3 shows the entire process from the demonstrators view. The expert demonstration collection is object-centric, i.e., the data of demonstrations solely focuses on the information of objects (window, target, and robot arm). On the other hand, human motion data is not a focus in our approach even though the robot arm is controlled by human.



(a) Initial State (b) Reach Window (c) Pick Up Window (d) Move To Target

Figure 3: Virtual expert demonstration process.

3.3 RL Training with Demonstration

After collecting the virtual expert demonstrations, we train the robot arm to learn a control policy to accomplish a construction task, i.e., window installation, using the RL method. A control policy is a function mapping states of an agent (e.g., a robot) to actions for the agent to take with the objective of maximizing the cumulative reward (i.e., immediate reward in the current step and the future reward). An RL agent learns an optimal control policy by interacting with the environment iteratively. In every interaction, after taking an action based on the current state and the latest control policy, the agent receives a reward from the environment indicating how beneficial the action is in that state. The control policy is then updated to make the agent more likely to pick an action that might result in higher rewards.

3.3.1 Policy Gradient Method

We utilize policy gradient methods (Sutton and Barto 2018) for RL training because it works well with continuous control of agents such as robots. Moreover, with the continued research efforts into development of policy gradient methods, there exist powerful algorithms based on policy gradient, such as proximal policy gradient (PPO) (Schulman et al. 2017) and trust region policy optimization (TRPO) (Schulman et al. 2015). Policy gradient methods model the control policy π directly using parameterized functions (e.g., neural network), $\pi_\theta(a|s)$, where a is the action chosen and sent to the environment; s is the current state; and θ is the set of parameters of the function. The differentiable policy is optimized using gradient ascent so as to output actions that yield high rewards.

To optimize the control policy, we first generate a set of trajectories from the environment by running the latest policy (step 3 in Algorithm 1). For each trajectory, we compute rewards-to-go (immediate reward and future reward) starting from the end of the trajectory (step 4 in Algorithm 1). Using the trajectories and calculated rewards, we compute a policy gradient and an advantage term A_t for optimizing the control policy (step 5 and 6 in Algorithm 1). We then update the policy using gradient ascent algorithms such as Adam (Kingma and Ba 2015) (step 7 in Algorithm 1). Finally, we fit the value function (for computing the advantage term) through regression of the mean-squared error using gradient descent (step 8 in Algorithm 1). Algorithm 1 shows the general procedure of the vanilla policy gradient method.

Algorithm 1 Vanilla Policy Gradient Algorithm

1: Input: initial policy parameters θ_0, initial value function parameters ϕ_0
2: **for** $k = 0, 1, 2, ...$ **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, $\hat{A}_t$ based on the current value function V_{ϕ_k}.
6: Estimate policy gradient as

$$\hat{g}_k = \frac{1}{|\mathcal{D}_k|} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)|_{\theta_k} \hat{A}_t.$$

7: Compute policy update, using gradient ascent algorithm like Adam, $\theta_{k+1} = \theta_k + \alpha_k \hat{g}_k$.
8: Fit value function by regression on mean-squared error via some gradient descent algorithm:

$$\phi_{k+1} = \arg\min_\phi \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} (V_\phi(s_t) - \hat{R}_t)^2.$$

9: **end for**

3.3.2 Proximal Policy Optimization

However, the vanilla policy gradient algorithm is prone to collapse in performance especially when using large step sizes, because it has no constraints on the deviation between the new and previous policies. To overcome this limitation of the vanilla policy gradient method, we adopt PPO (Schulman et al. 2017) as the RL algorithm in our approach. PPO is identical to the vanilla policy gradient algorithm except for the loss objective in step 6 of Algorithm 1. PPOs loss objective function (1) eliminates the shortcoming of vanilla policy gradient by introducing constraints to gradients. When advantage value A is positive, the ceiling is $(1+\epsilon)A^{\pi_{\theta_k}}(s,a)$. When advantage value A is negative, the ceiling is $(1-\epsilon)A^{\pi_{\theta_k}}(s,a)$. The ceilings guarantee that the control policy always takes a small step safely. Moreover, PPO also has the

advantages of easier implementation and tuning compared to other policy gradient-based algorithms such as TRPO (Schulman et al. 2015).

$$L(s, a, \theta_k, \theta) = min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a), g(\epsilon, A^{\pi_{\theta_k}}(s, a))\right), \tag{1}$$

where

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A & A \geq 0 \\ (1-\epsilon)A & A < 0. \end{cases}$$

3.3.3 Trajectory Set with Virtual Demonstrations

The key point of our approach lies in adding virtual expert demonstrations into the set of trajectories $\mathcal{D}_k$ in step 3 of Algorithm 1. We hypothesize that adding additional demonstrations will accelerate the training process of the RL agent. In the virtual expert demonstrations, the state and action pairs are usually close to optimal with larger rewards-to-go. The demonstrations provide the control policy with examples to imitate (Rajeswaran et al. 2018) and hence allow the agent to learn faster.

4 EXPERIMENTS AND RESULTS

4.1 Experiments

To validate our approach, we implemented it on the construction task of window pickup and installation. Using the virtual construction site created in Pybullet, we collected a total of 10 virtual expert demonstrations. The data collection was done as an alpha test within the research group to validate the viability of the approach. During each demonstration, the state information at each timestep was logged. Each state included seven-dimensional joint rotations of the robot arm, seven-dimensional position and orientation of the window, and seven-dimensional position and orientation of the target. Table 1 shows the lengths of the 10 collected demonstration trajectories.

Table 1: Lengths of demonstration trajectories (each timestep takes $\frac{1}{60}$s).

Demo Index	1	2	3	4	5	6	7	8	9	10
Length (timestep)	6224	6876	5833	7851	7000	6093	6613	7206	7639	5794

The lengths of raw demonstrations are excessively long for the replay buffer. Thus, we kept three key states in each demonstration, including the initial state, the state of pickup, and the state of installation (i.e., termination state). As for the other timesteps, we randomly sampled from timesteps between the key states to ensure that the length of demonstration data does not exceed the capacity of the replay buffer.

To make the demonstration data compatible with trajectories τ_i in $\mathcal{D}_k$, we further complemented extra 5 columns for each demonstration: action (a), immediate reward (r), expected future reward (v), the logarithm of a probability of taking an action ($\log p$), and the next state (s').

For the action, we calculated the seven-dimensional incremental action (i.e., the seven joints rotation changes) taken by the robot arm by subtracting the joints' reading in the current timestep from those in the previous timestep. The action in the final timestep of each demonstration was set to all rotation changes being zeros because this timestep marked the termination of the demonstration with no further changes. For the immediate reward, it was based on the reward function defined for the RL algorithm. In our experiment,

we defined the reward function as follows:

$$R(s,a) = \begin{cases} 1 & \text{Window panel pickup} \\ 2 & \text{Window panel installation} \\ -1 & \text{Robot arm and window panel collide} \\ -\frac{1}{3000} & \text{Otherwise.} \end{cases}$$

The reward function was designed this way so that the cumulative reward in each trial was always bounded from above by 3 and below by -2. Since we assumed the demonstrations were optimal strategies, there was no collision between the robot arm and the window panel. We only needed to fill the immediate reward of each entry with 1, 2, or $-\frac{1}{3000}$. For the expected future reward, we recursively conducted the calculation starting from the final timestep of each demonstration using the formula $V(s) = R(s,a) + \lambda V(s')$, where V is the function mapping the current s to v, R is the function mapping current s and a to r, and λ is the discount factor being 0.99. For the logarithm of the probability of taking an action, we assigned the value of the natural logarithm of 0.75 in all timesteps, assuming that this action was taken with high confidence with minor stochasticity. For the next state, it was straightforward that the state in the next timestep is the s' with respect to the current timestep. The s' of in the termination timestep was the same as the termination state since there would not be any state changes. After data complementation, each entry in the dataset of a demonstration is a tuple containing 6 elements.

We trained our agents in two modes. The first mode (M_1) was training the robot arm using PPO without additional demonstration added to the set of trajectories $\mathcal{D}_k$. The second mode (M_2) was training the robot arm using PPO with virtual expert demonstrations added to the set of trajectories $\mathcal{D}_k$ during the early phase (e.g., demonstrations were used only in the first 20 epochs in Algorithm 1; after that, the training was the same as in M_1). The demonstration data only occupied part of the space in $\mathcal{D}_k$, while the remaining space was filled with the data from the agent's interactions with the environment.

We determined a set of hyperparameters that made the agent trained using M_1 successfully converge. When training agents using M_2, we applied the same set of hyperparameters without any further tuning, except that there was an additional hyperparameter representing the number of epochs using demonstration for training in M_2. In total, we trained 10 agents by adding 10 different demonstrations using M_2 respectively and trained one agent using M_1.

4.2 Results and Discussion

Among the 10 agents trained using 10 different demonstrations in M_2, seven agents successfully outperformed the agent without demonstrations trained in M_1 by faster convergence and higher stability, as shown in Figure 4(a) and Figure 4(b); two agents' performance was on par with the agent without demonstrations, as shown in Figure 5(a) and Figure 5(b); and one agent showed an unstable and slower learning process despite eventual convergence, as shown in Figure 6.

To get a more general sense of the benefit brought by demonstrations, we aggregated the seven reward-epoch lines in Figure 4(a) to generate Figure 4(b) by bootstrapping the data per epoch (i.e., random sampling with replacement). The solid line in the middle represented the mean value, and the faded area represented the 95% confidence interval. For agents using demonstrations, their rewards started to increase rapidly from the beginning of the training. Their rewards reached above zero after around 12 epochs, signifying that the agents had reliably learned the task of window panel pickup. Then, the rewards kept increasing steadily with minor fluctuations and converged to a high reward of over 2.97 after around 50 epochs, signifying that the agents had reliably learned the task of window pickup and installation. In the entire training process, the gradient of the reward curve was stable until convergence. On the other hand, for the agent without demonstrations, the initial increment of the reward in the early phase was relatively modest, which resulted in the reward reaching above zero after around 25 epochs. Once the agent succeeded in finishing the task of window pickup and installation during exploration, the agent rapidly adjusted the control policy so that it

could be more likely to take such actions. Thus, the reward of the agent without demonstrations increased greedily from epoch 25 to epoch 35. However, its reward did not converge until after around 100 epochs. Between epoch 35 and epoch 100, the agent without demonstrations experienced numerous fluctuations while adjusting its control policy, leading to longer time until convergence. Essentially, the outperformance of agents with demonstrations was decomposed into threefold: an early start with the knowledge of the optimal control policy provided by demonstrations, a stable and consistent gradient throughout, and fewer fluctuations in the learning process.

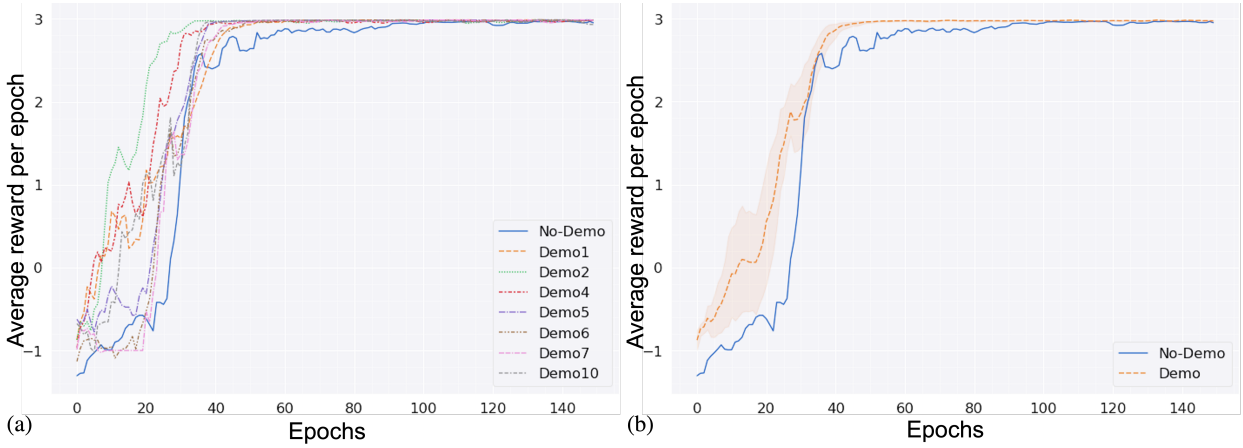


Figure 4: (a) 7 demonstrations make agents learn faster. (b) Aggregate of the 7 agents' performance.

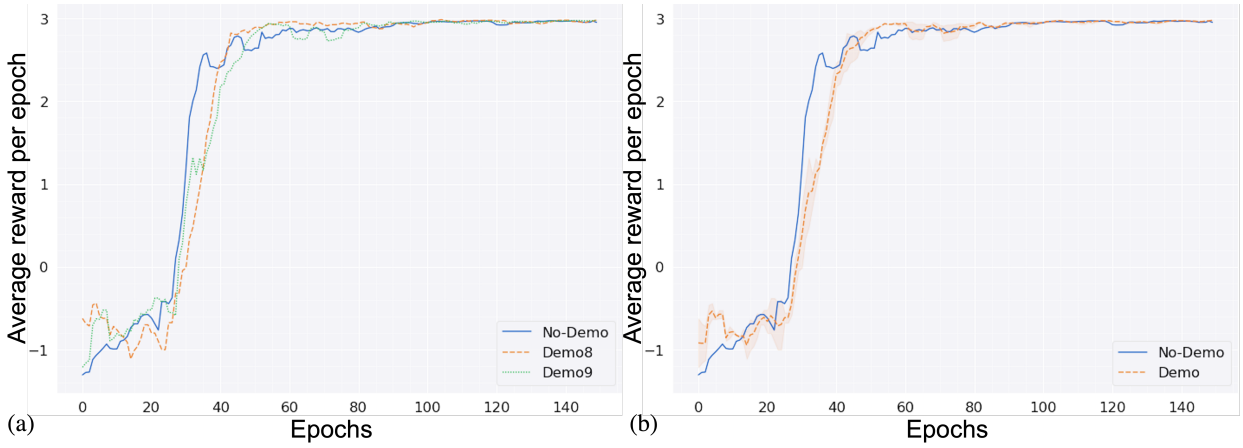


Figure 5: (a) 2 demonstrations make agents on par with the agent without demonstration. (b) Aggregate of the 2 agents' performance.

As for the two agents whose convergence time was on par with the agent without demonstrations (Figure 5(a) and Figure 5(b)) and the one agent that performed worse than the agent without demonstrations (Figure 6), we conjecture this was mainly due to the quality of demonstration data collected in VR. The ways for improvement include sampling states from demonstrations using carefully handcrafted methods instead of using random sampling between key states in the experiment, collecting more demonstrations, and checking data quality before adoption. Besides, since the set of hyperparameters was determined based on the agent without demonstrations, fine-tuning the models of agents with demonstrations may also help

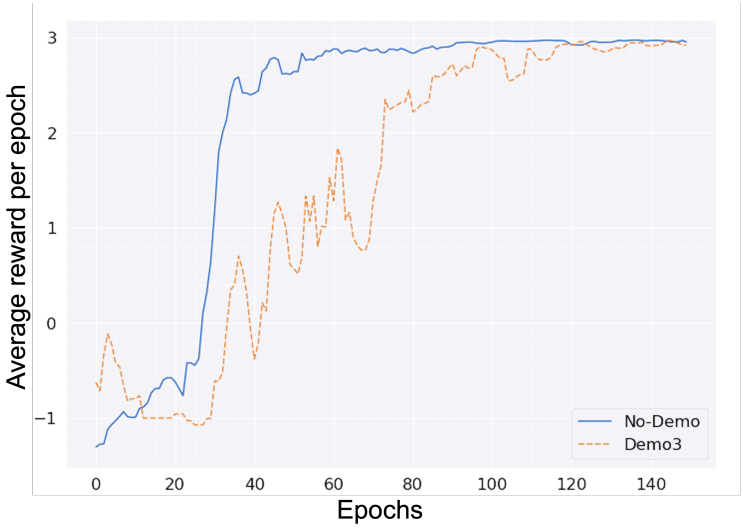


Figure 6: 1 demonstration makes agent learn slower..

them achieve improved performances. But these are not the focus of this paper, so we prioritize illustrating the benefits brought by demonstrations.

Overall, our results show that utilizing demonstration data in the replay buffer as we proposed can accelerate the policy training resulting in faster convergence.

5 CONCLUSION

In this paper, we proposed an approach aiming to accelerate the training for the control policy of RL agents by adding virtual expert demonstrations to the replay buffer during training. Results showed that the virtual demonstration trajectories could successfully and effectively assist the robot arm in expediting the learning of construction tasks (i.e., window pickup and installation) resulting in a more stable learning process and a shorter convergence time. The choice of the window installation task in the experiment was used as an example. Our approach can be scaled to training construction robots learning a repertoire of diverse construction tasks, such as welding and flooring. To conclude, RL algorithms enable construction robots to adaptively learn tasks from interactions with environments, but the learning process is time-intensive; and our proposed approach alleviates this issue and lays the groundwork for faster development of RL-based construction robots. Although we have not tested our approach in reality, we foresee that the objects state information can be collected using sensors, and the robot arms state information can be retrieved directly via built-in packages. Thus, our approach is expected to work properly in reality as well. For future work, we aim to explore more imitation-based methods of developing RL-based construction robots. Potential forms of demonstrations for imitation include videos, human intervention while training, and verbal instructions. Besides, we will apply the approach to a real robot and dive into the transferring problem of simulation to reality.

REFERENCES

Apolinarska, A. A., M. Pacher, H. Li, N. Cote, R. Pastrana, F. Gramazio, and M. Kohler. 2021. "Robotic assembly of timber joints using reinforcement learning". *Automation in Construction* 125:103569.

Belousov, B., B. Wibranek, J. Schneider, T. Schneider, G. Chalvatzaki, J. Peters, and O. Tessmann. 2022. "Robotic architectural assembly with tactile skills: Simulation and optimization". *Automation in Construction* 133:104006.

BLS 2022. "US Bureau of Labor Statistics: Construction NAICS 23". https://www.bls.gov/iag/tgs/iag23.htm, accessed 08.04.2022.

Bruzl, M., V. Usmanov, P. Svoboda, and R. Sulc. 2016. "Optimizing the Trajectory of the Painting Robot". In *Proceedings of the 33rd International Symposium on Automation and Robotics in Construction (ISARC)*, edited by A. A. U. Sattineni,

S. A. U. Azhar, and D. G. T. U. Castro, 605–612. Auburn, Alabama, USA: International Association for Automation and Robotics in Construction (IAARC).

Carlson, J., and R. R. Murphy. 2005. "How UGVs physically fail in the field". *IEEE Transactions on robotics* 21(3):423–437.

Cheng, C.-A., X. Yan, N. Wagener, and B. Boots. 2018. "Fast Policy Learning through Imitation and Reinforcement". In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018*, edited by A. Globerson and R. Silva, 845–855. Monterey, California, USA: AUAI Press.

Christensen, A. L., R. OGrady, and M. Dorigo. 2008. "SWARMORPH-script: a language for arbitrary morphology generation in self-assembling robots". *Swarm Intelligence* 2(2):143–165.

Erwin Coumans and Yunfei Bai 2016–2021. "PyBullet, a Python module for physics simulation for games, robotics and machine learning". http://pybullet.org.

Cully, A., J. Clune, D. Tarapore, and J.-B. Mouret. 2015. "Robots that can adapt like animals". *Nature* 521(7553):503–507.

de Soto, B. G., I. Agustí-Juan, J. Hunhevicz, S. Joss, K. Graser, G. Habert, and B. T. Adey. 2018. "Productivity of digital fabrication in construction: Cost and time analysis of a robotically built wall". *Automation in construction* 92:297–311.

Eysenbach, B., R. R. Salakhutdinov, and S. Levine. 2021. "Robust Predictable Control". In *Advances in Neural Information Processing Systems*, edited by M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Volume 34, 27813–27825. Virtual: Curran Associates, Inc.

Fujimoto, S., H. Hoof, and D. Meger. 2018. "Addressing function approximation error in actor-critic methods". In *International conference on machine learning*, 1587–1596. Stockholm, Sweden: PMLR.

Haas, C., M. Skibniewski, and E. Budny. 1995. "Robotics in civil engineering". *Computer-Aided Civil and Infrastructure Engineering* 10(5):371–381.

Horgan, D., J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. van Hasselt, and D. Silver. 2018. "Distributed Prioritized Experience Replay". In *6th International Conference on Learning Representations*. April 30th-May 3rd, Vancouver, Canada.

Ibarz, J., J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine. 2021. "How to train your robot with deep reinforcement learning: lessons we have learned". *The International Journal of Robotics Research* 40(4-5):698–721.

Iturralde, K., and T. Bock. 2018. "Integrated, Automated and Robotic Process for Building Upgrading with Prefabricated Modules". In *Proceedings of the 35th International Symposium on Automation and Robotics in Construction (ISARC)*, edited by J. Teizer, 340–347. Taipei, Taiwan: International Association for Automation and Robotics in Construction (IAARC).

Janner, M., Q. Li, and S. Levine. 2021. "Offline Reinforcement Learning as One Big Sequence Modeling Problem". In *Advances in Neural Information Processing Systems*, edited by M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Volume 34, 1273–1286. Virtual: Curran Associates, Inc.

Karimi, H., T. R. Taylor, G. B. Dadi, P. M. Goodrum, and C. Srinivasan. 2018. "Impact of skilled labor availability on construction project cost performance". *Journal of Construction Engineering and Management* 144(7):04018057.

King, N., M. Bechthold, A. Kane, and P. Michalatos. 2014. "Robotic tile placement: Tools, techniques and feasibility". *Automation in Construction* 39:161–166.

Kingma, D. P., and J. Ba. 2015. "Adam: A Method for Stochastic Optimization". In *3rd International Conference on Learning Representations (ICLR)*. May 7th-9th, San Diego, California, USA.

Kyjanek, O., B. Al Bahar, L. Vasey, B. Wannemacher, and A. Menges. 2019, May. "Implementation of an Augmented Reality AR Workflow for Human Robot Collaboration in Timber Prefabrication". In *Proceedings of the 36th International Symposium on Automation and Robotics in Construction (ISARC)*, edited by M. Al-Hussein, 1223–1230. Banff, Canada: International Association for Automation and Robotics in Construction (IAARC).

Lauria, S., G. Bugmann, T. Kyriacou, and E. Klein. 2002. "Mobile robot programming using natural language". *Robotics and Autonomous Systems* 38(3-4):171–181.

Lee, K., and W. Chung. 2008. "Calibration of kinematic parameters of a car-like mobile robot to improve odometry accuracy". In *2008 IEEE International Conference on Robotics and Automation*, edited by S. Hutchinson, 2546–2551. Pasadena, California, USA: IEEE.

Levine, S. 2018. "Reinforcement learning and control as probabilistic inference: Tutorial and review". *arXiv preprint arXiv:1805.00909*.

Liang, C.-J., V. R. Kamat, and C. C. Menassa. 2020. "Teaching robots to perform quasi-repetitive construction tasks through human demonstration". *Automation in Construction* 120:103370.

Liang, C.-J., X. Wang, V. R. Kamat, and C. C. Menassa. 2021. "Human–Robot Collaboration in Construction: Classification and Research Trends". *Journal of Construction Engineering and Management* 147(10):03121006.

Liu, Y., A. Gupta, P. Abbeel, and S. Levine. 2018. "Imitation from observation: Learning to imitate behaviors from raw video via context translation". In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 1118–1125. Brisbane, Australia: IEEE.

Nachum, O., S. S. Gu, H. Lee, and S. Levine. 2018. "Data-Efficient Hierarchical Reinforcement Learning". In *Advances in Neural Information Processing Systems*, edited by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Volume 31, 3303–3313. Montral, Canada: Curran Associates, Inc.

OE 2021. "Oxford Economics: Future of Construction A Global Forecase for Construction to 2030". https://resources.oxfordeconomics.com/hubfs/Future%20of%20Construction_Full%20Report_FINAL.pdf, accessed 08.04.2022.

OSHA 2019. "Occupational Safety and Health Administration, United States Department of Labor: Commonly Used Statistics". https://www.osha.gov/data/commonstats, accessed 08.04.2022.

Pradhananga, P., M. ElZomor, and G. Santi Kasabdji. 2021. "Identifying the challenges to adopting robotics in the US construction industry". *Journal of Construction Engineering and Management* 147(5):05021003.

Rajeswaran, A., V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. 2018. "Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations". In *Robotics: Science and Systems XIV*. June 26th-28th, Pittsburgh, Pennsylvania, USA.

Salmi, T., J. M. Ahola, T. Heikkilä, P. Kilpeläinen, and T. Malm. 2018. "Human-robot collaboration and sensor-based robots in industrial applications and construction". In *Robotic Building*, edited by H. Bier, 25–52. Cham: Springer International Publishing.

Schulman, J., S. Levine, P. Abbeel, M. Jordan, and P. Moritz. 2015, July 07–09. "Trust Region Policy Optimization". In *Proceedings of the 32nd International Conference on Machine Learning*, edited by F. Bach and D. Blei, Volume 37 of *Proceedings of Machine Learning Research*, 1889–1897. Lille, France: PMLR.

Schulman, J., F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. 2017. "Proximal policy optimization algorithms". *arXiv preprint arXiv:1707.06347*.

Seriani, S., A. Cortellessa, S. Belfio, M. Sortino, G. Totis, and P. Gallina. 2015. "Automatic path-planning algorithm for realistic decorative robotic painting". *Automation in Construction* 56:67–75.

Sutton, R. S., and A. G. Barto. 2018. *Reinforcement learning: An introduction*. Second ed. Bradford Books. http://www.incompleteideas.net/book/RLbook2020.pdf, accessed 17.03.2022.

Tonnon, S. C., R. Van Der Veen, M. J. Westerman, S. J. Robroek, H. P. Van Der Ploeg, A. J. Van Der Beek, and K. I. Proper. 2017. "The employer perspective on sustainable employability in the construction industry". *Journal of Occupational and Environmental Medicine* 59(1):85–91.

USCC 2021. "U.S. Chamber of Commerce: Q4 2021 Commercial Construction Index". https://www.uschamber.com/assets/documents/Q4-2021-CCI-Report.pdf, accessed 08.04.2022.

Vecerik, M., T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller. 2017. "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards". *arXiv preprint arXiv:1707.08817*.

Xu, X., and B. Garcia de Soto. 2020. "On-site Autonomous Construction Robots: A review of Research Areas, Technologies, and Suggestions for Advancement". In *Proceedings of the 37th International Symposium on Automation and Robotics in Construction (ISARC)*, edited by H. Osumi, H. Furuya, and K. Tateyama, 385–392. Kitakyushu, Japan: International Association for Automation and Robotics in Construction (IAARC).

Yin, X., H. Liu, Y. Chen, and M. Al-Hussein. 2019. "Building information modelling for off-site construction: Review and future directions". *Automation in Construction* 101:72–91.

Yu, Y., H. Li, X. Yang, L. Kong, X. Luo, and A. Y. Wong. 2019. "An automatic and non-invasive physical fatigue assessment method for construction workers". *Automation in construction* 103:1–12.

Zhang, A., R. T. McAllister, R. Calandra, Y. Gal, and S. Levine. 2021. "Learning Invariant Representations for Reinforcement Learning without Reconstruction". In *9th International Conference on Learning Representations (ICLR)*. May 3rd-7th, Vienna, Austria.

AUTHOR BIOGRAPHIES

LEI HUANG is a Ph.D. student in the Department of Civil Engineering, Faculty of Applied Science at The University of British Columbia. He holds a M.Sc. degree in Civil Engineering from Columbia University in the City of New York. His e-mail address is lei.huang@ubc.ca.

ZHENGBO ZOU is an Assistant Professor in the Department of Civil Engineering, Faculty of Applied Science at The University of British Columbia. He holds a Ph.D. degree in Civil Engineering from New York University. His research focuses on advancing the knowledge of automation and intelligence technologies in the life-cycle of design, construction, and maintenance of civil infrastructures. His email address is zhengbo@civil.ubc.ca. His website is https://civil.ubc.ca/faculty/zhengbo-zou/.