15th Canadian Masonry Symposium
Ottawa, Canada
June 2-5, 2025

Reinforcement Learning Assisted Robotic Construction for Masonry Block Dry-Stacking in Simulation Environments

Bowen Zeng[i], Yuxiang Chen[ii], and Yong Li[iii]

ABSTRACT

Traditional masonry construction methods face significant challenges in tasks such as masonry block stacking, including labor intensity, quality variability, and the demand for high precision. These challenges often result in inefficiencies and inconsistent outcomes. Robotic construction technology presents a promising alternative by automating repetitive and complex tasks, which can improve efficiency, consistency, and accuracy. However, conventional industrial robots heavily rely on the pre-programming and human insights. Moreover, they are limited by the need for precise control and struggle to adapt to varied construction environments. To overcome these limitations, this study introduces a reinforcement learning (RL) strategy to optimize robotic masonry block dry-stacking. In this approach, the robotic arm autonomously learns and refines its stacking techniques through iterative interactions within a simulated environment that replicates the construction conditions. Performance evaluations indicate that the RL is capable of facilitating the block stacking process during the dry masonry construction, marking a step forward in automated masonry process.

KEYWORDS

Reinforcement Learning, Robotics, Block Dry Stacking, Masonry Construction

[i] Postdoctoral Fellow, University of Alberta, Edmonton, Canada, bzeng1@ualberta.ca
[ii] Associate Professor, University of Alberta, Edmonton, Canada, ychen5@ualberta.ca
[iii] Associate Professor, University of Alberta, Edmonton, Canada, yong9@ualberta.ca

INTRODUCTION

Masonry has long been valued for its durability, versatility, and aesthetic appeal and serves as the backbone for countless structures worldwide. From the perspective of structural engineering, masonry systems offer exceptional strength and resilience against external loads and environmental stressors. Moreover, their inherent thermal and acoustic properties enhance functionality, contributing to sustainable and energy-efficient construction practices.

The tradition of masonry dates back to ancient civilizations, as exemplified by iconic structures such as the Egyptian pyramids and Roman aqueducts. Early constructions were realized with the precision through the expertise of skilled artisans and the use of rudimentary tools. Over time, advancements in materials and techniques, such as reinforced masonry and modular block systems, have broadened its applications in modern architecture. Nevertheless, masonry construction remains labor-intensive and requires highly skilled labor to ensure precise block placement and structural integrity. This reliance on manual craftmanship presents challenges, particularly in large-scale projects where maintaining consistency and quality control is demanding [1–3].

In recent decades, the construction industry has increasingly embraced automation to address labor shortages. For example, robotic arms equipped with specialized grippers and advanced control mechanisms can position blocks according to predefined patterns, thereby reducing human error. Such systems utilize programmable motion paths and real-time adjustments to ensure proper structural alignment. Slocum and Schena [4] made early efforts to automate masonry construction through the development of Blockbot, a six degree-of-freedom (DOF) robotic manipulator designed to retrieve blocks from a delivery system. Subsequent research by Pritschow et al. [5] formalized key requirements for bricklaying robots and introduced a modular control system tailored to masonry tasks. Later, Pritschow et al. [6] proposed a mobile robotic system for automated on-site masonry construction by integrating thin-bed mortar application via a dipping method. Yu et al. [7] evaluated the feasibility of robotic bricklaying by co-optimizing manipulation trajectories (i.e., kinematic motion paths) and laying patterns (i.e., spatial brick configurations) to balance efficiency and structural integrity. Feng et al. [8] addressed material-handling challenges in unstructured environments by developing algorithms for the autonomous identification, grasping, and assembly of prismatic building components stored arbitrarily on-site. Giftthaler et al. [9] introduced a mobile robotic platform with high-performance control and planning algorithms for on-site digital fabrication. Bruun et al. [10] presented and validated three cooperative fabrication approaches using two or three robots for the scaffold-free construction of a stable masonry arch, from which a medium-span vault was subsequently built.

Despite these advances, these robotic systems rely heavily on pre-programming and human insights, which limits their flexibility and adaptability. To address these limitations, researchers have begun to integrate intelligent control strategies into robotics. Reinforcement learning (RL) techniques, for example, enable robots to learn and adapt through trial and error in response to new scenarios. Although RL holds significant promise, its application in construction robotics has only recently begun to receive attention [11–13]. Huang et al. [11] trained RL-based robots for window panel pick-and-transport tasks. Apolinarska et al. [12] applied RL to control robot movements in contact-rich and tolerance-prone assembly tasks and presented its fesibility in architectural construction with timber joints. Luo et al. [13] combined RL with force/torque feedback via an operational space force controller for robot arm manipulation. Nevertheless, RL has received limited attention in masonry construction, with only a few studies addressing small-scale block assembly tasks [14,15].

This study explores the feasibility of applying RL to automate robotic block manipulation tasks within a dry-stacking context. The task does not replicate the full masonry process, including mortar application and tolerance-based adjustments, but it serves a foundational step toward developing more flexible and adaptive construction automation systems. By formulating the problem within an RL framework, the robot is enabled to make autonomous decisions and adapt its actions based on environmental variations during the stacking process. The approach reduces reliance on pre-programmed control logic and explicit inverse kinematics, which are typically required in conventional robotic systems. Preliminary results from simulation indicate that the RL-trained agent can improve placement consistency and respond to position uncertainty, suggesting the potential for scalable and robust automation strategies in future masonry-related applications.

PROBLEM FORMULATION: MASONRY BLOCK STACKING IN REINFORCEMENT LEARNING

Basics of Reinforcement Learning (RL)

Reinforcement Learning (RL) is a machine learning paradigm in which an agent learns to make decisions by interacting with the environment. In RL, the problem is modeled as a Markov Decision Process (MDP), represented by the tuple (S, A, P, R, γ). Here, S denotes the set of possible states; A is the set of actions available to the agent; $P(s'|s, a)$ specifies the probability of transitioning from state s to s' after taking action a; and $R(s, a)$ is the reward function that quantifies the immediate feedback received upon executing action a in state s. The discount factor $\gamma \in [0, 1)$ is used to prioritize immediate rewards over those obtained in the more distant future. Figure 1 illustrates the MDP procedure for the specific masonry block stacking tasks involved in this study. In this context, the agent's overarching goal is to obtain higher rewards by selecting actions that lead to consistently improved performance, i.e., stacking more blocks. This action selection process is guided a policy π. Therefore, the primary objective in RL is to determine an optimal policy π^* that directs the agent's actions to maximize the cumulative expected reward over time. The cumulative reward is defined as Eq. (1):

$$(1)\quad G_t = \sum_{k=0}^{\infty} \gamma^t R_{t+k+1}$$

where G_t represents the sum of discounted future rewards starting from time step t.

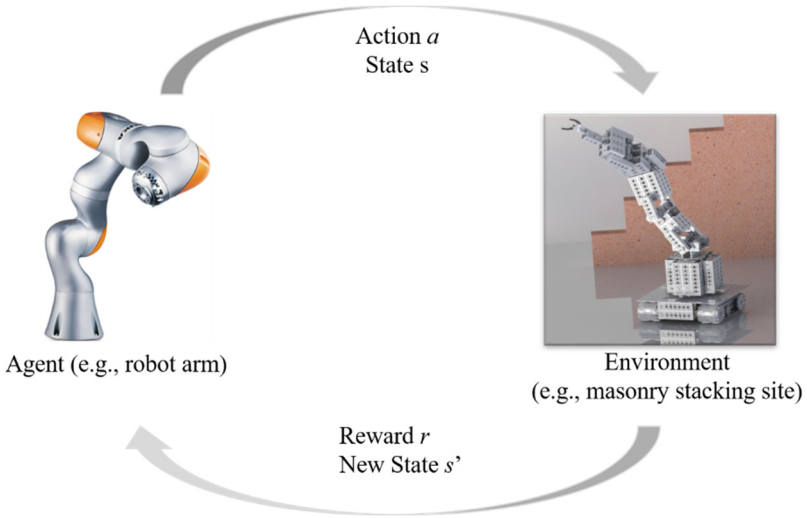


Figure 1: MDP procedure in the masonry block stacking task

Environment Setup and Formulation of Block Stacking Problem

The simulation environment for robotic masonry block stacking is developed in Pybullet [16], an open-source physics engine renowned for its high-fidelity simulation of rigid body dynamics. The environment is implemented in the style of an OpenAI Gym environment [17]. OpenAI Gym is a widely adopted toolkit for creating and comparing RL algorithms and Gym Interface standardizes the simulation structure by managing state observations, action execution, and episode management.

In the context of masonry block stacking, the state space encompasses the positions and orientations of both masonry blocks and robot arm, as well as their interactions. The action space consists of the essential robotic movements required for block manipulation, including pick-up and placement. In this study, a 7-DOF KUKA LBR iiwa robot arm [18] is used due to its widespread application in similar manipulation tasks. Figure 2 demonstrates the simulated environment, including the robot arm, nine blocks with random initial positions (with three groups of two blocks stacking together), and a target location for constructing the masonry wall. The masonry block considered have typical dimensions of those used in Canada: 390 mm × 190 mm × 190 mm.

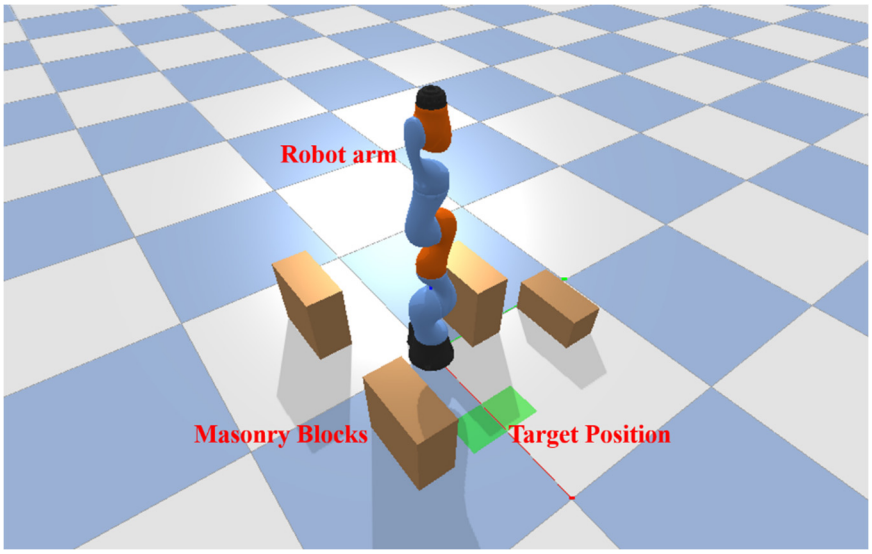


Figure 2. Simulated environment for block stacking problem

The masonry block stacking task is divided into two subtasks: pick-up and placement. During the pick-up phase, the robot arm learns to identify and securely grasp a masonry block. This process requires precise alignment between the robotic end-effector and the block, followed by safe extraction without inducing unintended displacement. Once a block is successfully grasped, the task transitions to the placement phase, in which the robot arm needs to accurately position the block at a designated location (i.e., green region in Figure 1).

For each subtask, three key components are defined: the observation space, the action space, and the reward function. In the pick-up subtask, the observation space is characterized by the coordinate difference between the end-effector and the block, denoted as $d_{ee\text{-}b}$. Note that the actual gripping action is neglected in the simulation. A block is considered successfully picked up when $d_{ee\text{-}b}$ falls below a predetermined threshold. In the placement subtask, the observation space is defined by the coordinate difference between the block and the target position $d_{b\text{-}t}$. The objective for the robot arm is to place the block such that $d_{b\text{-}t}$ is less than a specified threshold. In both subtasks, the action space comprises the seven joint rotations of the robot arm. The formulations for $d_{ee\text{-}b}$ and $d_{b\text{-}t}$ are presented in Equations (2-3):

(2) $d_{ee-b} = \sqrt[2]{(x_{ee} - x_b)^2 + (y_{ee} - y_b)^2 + (z_{ee} - z_b)^2}$

(3) $d_{b-t} = \sqrt[2]{(x_b - x_t)^2 + (y_b - y_t)^2 + (z_b - z_t)^2}$

Here, x, y, and z denote the Cartesian coordinates; d represents the Euclidean distance; and the subscript 'ee', 'b', and 't' refer to the end effector, block, and target position, respectively.

The reward functions are formulated based on the observation indicators $d_{ee\text{-}b}$ and $d_{b\text{-}t}$. As $d_{ee\text{-}b}$ and $d_{b\text{-}t}$ decrease, indicating proximity between the end effector and the block or between the block and the target, the robot arm receives a higher reward. Conversely, larger distances result in lower rewards. Upon successful completion of each subtask, the robot arm is granted a substantial positive reward. The reward functions R_1 and R_2, corresponding to the pick-up and placement subtasks, respectively, are formulated in Equations. (4-5):

(4) $R_1(t) = \begin{cases} \alpha[d_{ee-b}(t-1) - d_{ee-b}(t)] + R_c(t) & d_{ee-b} > grip\_threshold \\ 0.5 & d_{ee-b} \ll grip\_threshold \end{cases}$

(5) $R_2(t) =$
$\begin{cases} \beta[d_{b-t}(t-1) - d_{b-t}(t)] + R_c(t) & d_{b-t} > place\_threshold \\ 0.5 & d_{b-t} \ll place\_hreshold(first\ time) \\ 0.0001 & d_{b-t} \ll place\_hreshold\ (sucsequent\ steps) \end{cases}$

In Equations. (4-5), $\boldsymbol{\alpha}$ and β are two coefficients that weigh the importance of distance-based reward (determined as 10); $\boldsymbol{d_{ee-b}(t-1)}$ and $d_{ee-b}(t)$ denote the distance between the end-effector and block at the training step t - 1 and t, respectively; $\boldsymbol{R_c(t)}$ is a curiosity based learning function, aiming to encourage to explore more unfamiliar actions, defined as $\boldsymbol{R_c(t)} = k_{curiosity}\|obs(t) - obs(t-1)\|$. Here, $obs(t)$ and $obs(t-1)$ are the observation spaces at the training step t - 1 and t; $k_{curiosity}$ is a coefficient, determined as 0.0001 in this study.

Training Algorithm: Proximal policy Optimization (PPO)

As discussed earlies, the policy in RL is a function that maps states to actions and guides the agent's behavior. Typically, this policy is parameterized by a neural network, denoted as $\pi_\theta(a|s)$, where θ represents the network parameters. The objective of training is to learn an optimal policy that maximizes the cumulative reward through iterative updates based on interactions with the environment.

For the block stacking task, the policy is updated iteratively to achieve higher rewards. The Proximal Policy Optimization (PPO) algorithm [14] is employed due to its balance of sample efficiency and ease of implementation. At the core of PPO is the clipped surrogate objective, which limits large policy updates. At the step t, the probability ratio $r_t(\theta)$ compares the likelihood of action a_t under the current policy $\pi_\theta(a_t|s_t)$ to that under the previous policy $\pi_{\theta_{old}}(a_t|s_t)$:

is defined as Equation (6):

(6) $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$

The objective function $L^{CLIP}(\theta)$, i.e., clipped surrogate function, limits policy updates by bounding $r_t(\theta)$ within $[1-\epsilon, 1+\epsilon]$, as shown in Equation (7):

(7) $L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[min\left(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right]$

Here, $\hat{A}_t$ represents the estimated advantage at time t and ϵ *is* a small hyperparameter that controls the clipping range, determined as 0.2. This formulation prevents overly significant updates by clipping the probability ratio, i.e., $r_t(\theta)$ is too large.

The total loss function $L(\theta)$ with the optimization parameter θ combines the clipped objective with a value function loss $L^{VF}(\theta)$ to approximate state values and an entropy bonus $S[\pi_\theta](s_t)$ to encourage exploration, as shown in Equation (8):

$$(8)\quad L(\theta) = L^{CLIP}(\theta) - c_1 L^{VF}(\theta) + c_2 S[\pi_\theta](s_t)$$

In Equation (8), c_1 and c_2 are coefficients that balance these components. By iteratively updating the policy parameters θ using gradient ascent on $L(\theta)$, the policy parameters θ are refined to enable an effective policy learning to achieve a higher reward.

NUMERICAL RESULTS

The training was conducted by considering a small level of uncertainties of target block positions, i.e., $d + e$ with $e \sim$ U (-0.1, 0.1). d is the target position and e is the uncertainty. The training curve in Figure 3 illustrates the average cumulative reward across 1,000 training episodes. During the initial phase (the first 100 episodes), the reward is relatively small. This is a typical exploration phase of RL in which the robot arm tests various actions without a well-formed policy. As training continues, the agent discovers more effective strategies for block stacking, leading to a steady increase in the reward signal between episodes 0 and 200.

Beyond training episode 200, the average reward plateaus at a higher level, indicating that the agent has acquired a reasonably stable policy. Occasional fluctuations and sharp drops occur, primarily due to ongoing exploration and stochasticity in both the policy and the environment. Despite these variations, the upward trend demonstrates progressive policy refinement and improved adaptation to simulated construction tasks.

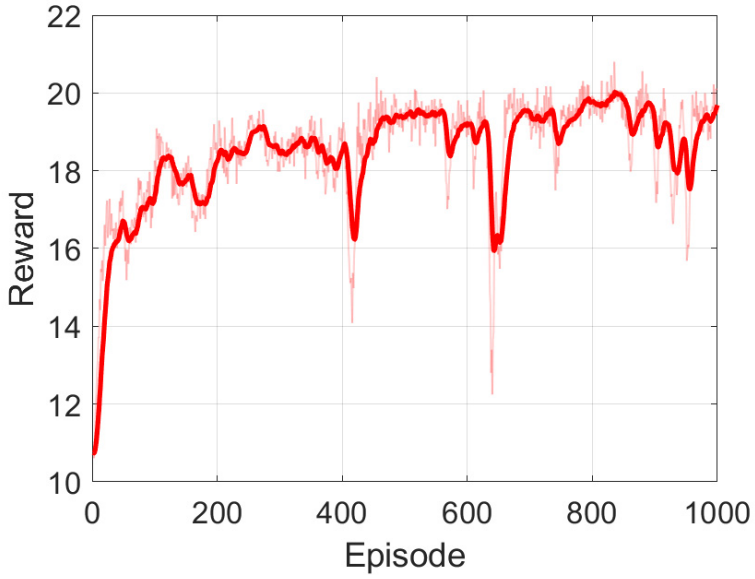
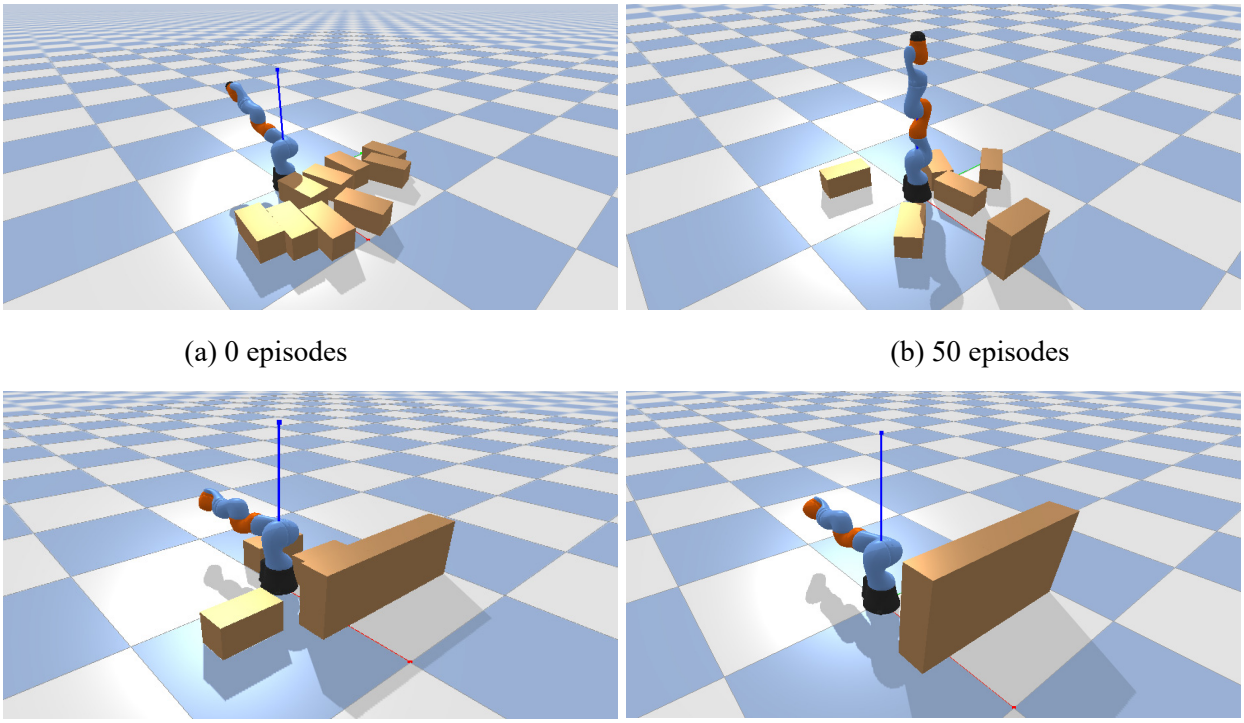


Figure 3. Training reward vs episode

Figure 4 illustrates the learning process at four distinct training stages: 0, 50, 300, and 1,000 episodes. At 0 training episodes without any training (Figure 4(a)), the robot performs random actions without a coherent strategy for block manipulation. Frequent collisions lead to mostly negative rewards, as also shown in Figure 3. The blocks are scattered on the floor, illustrating the robot's lack of coordination during this initial exploration phase. By 50 episodes (Figure 4(b)), the robot's performance exhibits partial improvement. Although its actions remain erratic, the robot arm demonstrates an emerging ability to grasp blocks. Some blocks have been repositioned in a semi-organized manner, suggesting that the policy is beginning to incorporate basic stacking strategies. Nevertheless, misplaced blocks and incomplete structures persist as the robot arm continues to refine its decision-making process.

Figure 4(c) and Figure 4(d) correspond to 300 and 1,000 training episodes, respectively. The results reveal a marked progression in performance. At 300 episodes, the robot consistently arranges blocks into a wall-like structure, indicating a more stable stacking approach despite not completing the entire task. By 1,000 episodes, the robot arm succeeds in constructing a complete wall. This progression underscores the effectiveness of the RL method. Through repeated interactions, the robot adapts its behavior to optimize block stacking. Although occasional negative rewards remain due to the misplacements and collisions, the overall trend toward higher rewards confirms the policy convergence.



(a) 0 episodes (b) 50 episodes

(c) 300 episodes (d) 1000 episodes

Figure 4. Training results with different training episodes

CONCLUSION

This study presents an approach for automating masonry block stacking process through reinforcement learning (RL) technique. By formulating the problem within the RL framework and leveraging the Proximal Policy Optimization (PPO) algorithm, a robust policy for block stacking problem was developed. The integration of a simulation environment using Gym and PyBullet enabled the modeling of construction dynamics and facilitated systematic training of the robotic system.

Preliminary training results demonstrate improvements in stacking precision and construction efficiency under dynamic conditions. The application of RL-based methods represents an advancement toward more flexible, adaptive, and efficient masonry construction practices. This work lays a foundation for future developments in automated masonry, with potential applications across a wide range of architectural and construction scenarios. Despite these encouraging outcomes, challenges remain in translating simulation success to real-world applications. Real construction environments involve dynamic obstacles, increased uncertainty, and greater environmental variability compared to the controlled conditions of simulation presented in this study. In addition to these environmental complexities, real-world implementation requires addressing various software and hardware limitations, including the integration of real-time vision systems for perception, dealing with sensor noise and calibration, managing controller latency, ensuring sufficient onboard computational capacity, and executing learned policies safely on physical hardware. Future research should focus on refining the simulation framework to better approximate real-world conditions and developing robust sim-to-real transfer strategies. Validation through physical testing in full-scale construction scenarios will also be critical for demonstrating practical feasibility.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the financial support provided by the Natural Sciences and Engineering Research Council (NSERC) in Canada through the Alliance Grant ALLRP 567205-21.

REFERENCES

[1] L. FLOREZ, Daniel CASTRO-LACOUTURE, Optimal Crew Design for Masonry Construction Projects Considering Contractor's Requirements and Workers' Needs, in: Constr. Res. Congr. 2014, 2011: pp. 140–149. https://doi.org/10.1088/1751-8113/44/8/085201.

[2] U.C. Gatti, G.C. Migliaccio, S.M. Bogus, S. Schneider, An exploratory study of the relationship between construction workforce physical strain and task level productivity, Constr. Manag. Econ. 32 (2014) 548–564. https://doi.org/10.1080/01446193.2013.831463.

[3] T. Elbashbishy, I.H. El-adaway, Skilled Worker Shortage across Key Labor-Intensive Construction Trades in Union versus Nonunion Environments, J. Manag. Eng. 40 (2024) 1–18. https://doi.org/10.1061/jmenea.meeng-5649.

[4] A.H. Slocum, B. Schena, Blockbot: A robot to automate construction of cement block walls, Rob. Auton. Syst. 4 (1988) 111–129. https://doi.org/10.1016/0921-8890(88)90020-6.

[5] G. Pritschow, M. Dalacker, J. Kurz, Configurable Control System of a Mobile Robot for On-Site Construction of Masonry, in: 10th Int. Symp. Robot. Autom. Constr., 1993. https://doi.org/10.22260/isarc1993/0012.

[6] G. Pritschow, M. Dalacker, J. Kurz, M. Gaenssle, Technological aspects in the development of a mobile bricklaying robot, Autom. Constr. 5 (1996) 3–13. https://doi.org/10.1016/0926-5805(95)00015-1.

[7] S.N. Yu, B.G. Ryu, S.J. Lim, C.J. Kim, M.K. Kang, C.S. Han, Feasibility verification of brick-laying robot using manipulation trajectory and the laying pattern optimization, Autom. Constr. 18 (2009) 644–655. https://doi.org/10.1016/j.autcon.2008.12.008.

[8] C. Feng, Y. Xiao, A. Willette, W. McGee, V.R. Kamat, Vision guided autonomous robotic assembly and as-built scanning on unstructured construction sites, Autom. Constr. 59 (2015) 128–138. https://doi.org/10.1016/j.autcon.2015.06.002.

[9] M. Giftthaler, T. Sandy, K. Dörfler, I. Brooks, M. Buckingham, G. Rey, M. Kohler, F. Gramazio, J. Buchli, Mobile robotic fabrication at 1:1 scale: the In situ Fabricator, Constr. Robot. 1 (2017) 3–14. https://doi.org/10.1007/s41693-017-0003-5.

[10] E.P.G. Bruun, R. Pastrana, V. Paris, A. Beghini, A. Pizzigoni, S. Parascho, S. Adriaenssens, Three cooperative robotic fabrication methods for the scaffold-free construction of a masonry arch, Autom. Constr. 129 (2021) 103803. https://doi.org/10.1016/j.autcon.2021.103803.

[11] L. Huang, Z. Zhu, Z. Zou, To imitate or not to imitate: Boosting reinforcement learning-based construction robotic control for long-horizon tasks using virtual demonstrations, Autom. Constr. 146 (2023) 104691. https://doi.org/10.1016/j.autcon.2022.104691.
[12] A.A. Apolinarska, M. Pacher, H. Li, N. Cote, R. Pastrana, F. Gramazio, M. Kohler, Robotic assembly of timber joints using reinforcement learning, Autom. Constr. 125 (2021) 103569. https://doi.org/10.1016/j.autcon.2021.103569.
[13] J. Luo, E. Solowjow, C. Wen, J.A. Ojea, A.M. Agogino, A. Tamar, P. Abbeel, Reinforcement learning on variable impedance controller for high-precision robotic assembly, Proc. - IEEE Int. Conf. Robot. Autom. 2019-May (2019) 3080–3087. https://doi.org/10.1109/ICRA.2019.8793506.
[14] Y. Lin, A.S. Wang, E. Undersander, A. Rai, Efficient and Interpretable Robot Manipulation with Graph Neural Networks, IEEE Robot. Autom. Lett. 7 (2022) 2740–2747. https://doi.org/10.1109/LRA.2022.3143518.
[15] S.K. Kang, J.G. Dy, M.B. Kane, Stone masonry design automation via reinforcement learning, Artif. Intell. Eng. Des. Anal. Manuf. AIEDAM. 37 (2023). https://doi.org/10.1017/S0890060423000100.
[16] E. Coumans, Y. Bai, Pybullet, a python module for physics simulation for games, robotics and machine learning, (2016).
[17] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, Openai gym, ArXiv Prepr. ArXiv1606.01540. (2016).
[18] V. Chawda, G. Niemeyer, Toward torque control of a KUKA LBR IIWA for physical human-robot interaction, IEEE Int. Conf. Intell. Robot. Syst. 2017-September (2017) 6387–6392. https://doi.org/10.1109/IROS.2017.8206543.