Towards RL Control for Flexible Robotic Applications in Construction: Grasping Static Objects

Mohab Hassaan[1], **Florian Noichl**[1] **and André Borrmann**[1]

[1]Chair of Computing in Civil and Building Engineering,
TUM Georg Nemetschek Institute,
Technical University of Munich, Germany

mohab.hassaan@tum.de, florian.noichl@tum.de, andre.borrmann@tum.de

Abstract -

With recent advancements in robotics, robots have been successfully utilized to automate processes in various sectors. However, their use in the construction industry has not mirrored this trend, due to the complexity of construction processes and the fact that they take place on highly dynamic construction sites. Therefore, construction robots must be robust to changing environments, and their design must involve ways to mitigate problems such as health hazards. Recently, there has been an increasing research interest in the field with a focus on automating processes on construction sites, including the transport of material via automated cranes, and smaller robotic platforms such as UAVs and UGVs. In the case of cranes, however, the studies did not consider the stabilization of the objects carried by cranes, as this is conventionally done by hand once the object is close to its intended target position. Automating this process requires a controller robust to changing environments. Therefore, we introduce a project aimed at designing a Reinforcement Learning (RL) controller that guides a robot arm to stabilize crane loads. This paper presents the results of the first milestone of the project, in which an RL-controlled robot arm is tasked with grasping a cuboid at an arbitrary pose in space. Initial experiments expose the policy to previously unseen cuboid positions, demonstrating robustness with respect to position, while highlighting reduced performance in specific regions of the domain that require further investigation. Additionally, the policy was shown to be robust with respect to variations in cuboid size.

Keywords -

Construction Robotics; Reinforcement Learning; Proximal Policy Optimization; Robotic Grasping

1 Introduction

Industrial automation through the use of robotics has become increasingly popular over the past decades, as robots have been integrated into industrial processes in many sectors including the automotive, agricultural, and healthcare sectors. Mirroring this integration of robots into the Architectural, Engineering, and Construction (AEC) industries, however, will require addressing AEC-specific difficulties. For instance, construction sites are highly unpredictable and dynamic, and unlike manufacturing plants, each construction site is unique [1]. Transport-related tasks, in particular, pose a challenge in construction robotics research. The relatively large scale of construction sites limits the use of robotic arms for various tasks due to their reach, i.e., the maximum length of the robot arm. Similarly, mobile alternatives like unmanned ground vehicles (UGVs) and unmanned aerial vehicles (UAVs) face limitations due to their battery life and restricted payload capacity. At the same time, the utilization of cranes for the transportation of objects on construction sites is well-established, as evidenced by their increased usage over the last two decades [2]. Therefore, recent research has focused on the use of robotized cranes, instead of adapting existing robots toward such tasks. Robotized cranes have since been successfully used for transport applications in simulations and real-world experiments [3, 4].

Still, a persistent problem is the oscillatory movement of objects transported by cranes, which can arise from external conditions, such as wind, or through the crane's own movement. As a crane carrying an object can resemble a moving, multi-link pendulum, crane-induced movements can create oscillations that are exacerbated by external conditions. These oscillations are usually dealt with by hand as the object approaches its target location; in this case, one or more construction workers grab the object, stabilize it, and then guide it toward its intended final location. This proximity to oscillating objects poses a danger to the worker, as failure or sudden and erratic movements can lead to injuries. A robot can, therefore, automate this process, moving workers away from the crane and eliminating this safety hazard.

Designing controllers for this robotic task, however, is non-trivial due to the dynamic environment in which the robots will operate. On construction sites, the controller must be able to handle varying locations and terrains, as well as obstacles and crane loads that differ in shape and path. Using classical control in this case would require

constant parameter tuning for every setup, deeming them inadequate for such tasks. Reinforcement Learning (RL) offers an alternative to classical control methods for robot control. In an RL formulation, an agent, representing the robot in this case, interacts with an environment through the use of actions that are chosen based on the agent's observations. Each interaction results in a reward, a scalar value that quantifies the success of the agent's interaction. Repeated interactions with the environment result in increasingly successful actions. Therefore, unlike classical control formulations, RL-based controllers allow for robot learning without human feedback [5, 6].

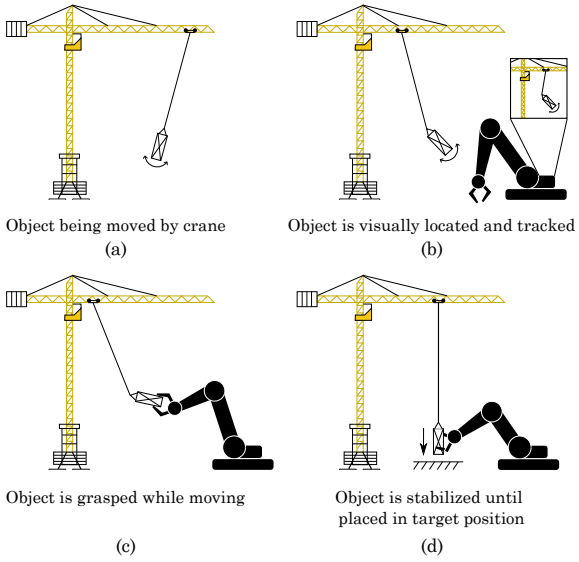

Figure 1. A schematic illustrating the concept of the proposed project, where alphabetical labels indicate the chronological order of all sub-tasks. Oscillations in subfigures (a) and (b) are visually depicted via double-sided arrows.

The aim of the presented work is to design an RL controller for a robot arm that stabilizes objects carried by cranes, thereby eliminating the need for workers near the crane's target position and reducing the risk of accidents. The robot, in this case, will be placed at the crane's target position; as the object comes within reach, it grasps the object, suppresses oscillations, and then places the object safely at its target location (Figure 1). The controller will be trained in robot simulators and deployed on a real robot. The goal is to train the policy to handle varying locations and crane paths, mimicking real construction-site conditions and allowing for possible future on-site deployment.

The project is organized into milestones of increasing complexity. The final milestone will employ a mobile robot arm that navigates to the crane's target location, then grasps and stabilizes the object. Grasping and stabilization will rely solely on vision and force-torque inputs, while coordinates will be provided for the navigation task.

This paper presents the results of the project's first milestone. Given a stationary cuboid at an arbitrary pose in space, the RL policy is tasked with guiding the robot to the cuboid and, once in reach, grasping the object. Successful grasps are restricted to those in which the approach direction is collinear with the normal vector of one of the cuboid's six faces, and the grasp must lie within one of the three grasping planes of the cuboid. This grasping definition was chosen as it is generalizable to more complex shapes. For instance, an appropriately selected normal vector and grasp plane can allow the robot to grasp an arbitrary object, e.g., formwork. The work reported here is confined to simulation; the next milestone involves deploying the trained policy on a real robot.

2 Related Work

An earlier instance of RL usage in the field of construction robotics was presented by Chu [7], who developed a state-space model aimed at resolving redundancies for mobile robotic arms with 9 Degrees of Freedom: 6 for the robot arm and 3 for the mobile base. The author then recommended using RL to generate solutions to the model. Later, Barros dos Santos et al. [8] used RL to train a quad-rotor to assemble truss structures. In this case, A* was used for path planning, and RL was used for learning valid assembly sequences. Real-robot experiments showed the approach's validity, although grasping was aided by magnetized components and, therefore, the problem was effectively reduced to one of correct placement. Subsequent work introduced custom structures that allowed users to specify various shapes and assemblies as input to the policy [9].

RL-based robot control has additionally been used for wall assembly. Barros dos Santos et al. [10] used an RL/Particle Swarm Optimization hybrid to assemble bricks into various 3D structures. As in the authors' previous work, the problem was simplified by using magnetized bricks. Liu et al. [11] used deep Q-learning for dry-stacking 2D irregular objects. The trained policy was deployed onto a UR5 robot arm.

Multi-agent cooperative construction was investigated by Zhu et al. [12] and Barros dos Santos et al. [13], where workflows for single construction tasks were detailed. Similarly, Liu et al. [14] used a deep Q-learning algorithm to train a dual-arm robot for slab installation in simulation. The policy used force sensor data as input, and the filling material was modeled using elastic springs. All studies, however, did not investigate the sim-to-real gap of their approaches. Moreover, Zhu et al. [12] focused on two-dimensional domains.

Imitation Learning (IL) is another agent-based learning paradigm that has become increasingly popular for

construction robotics. Instead of learning through feedback obtained via a reward function, IL agents learn from "expert" demonstrations, from which the agent infers desirable behaviors [15]. Several studies have used IL/RL hybrid setups that leverage human or simulated demonstrations and RL for training, allowing policies to be trained for more complex tasks. Additionally, IL/RL hybrids have achieved faster convergence speeds relative to pure RL formulations.

Apolinarska et al. [16] used simulated demonstrations to train an end-effector for construction tasks with Ape-X DDPG, a variant of the RL algorithm DDPG with distributed prioritized experience replay. Training was carried out in simulation, after which the policy was deployed onto an ABB robot arm. Similarly, Liang et al. [17] used a context-translation model for feature extraction and translation of human demonstrations and used the resultant state/action pairs together with TRPO to learn the control policy. In the latter study, however, the work was confined to simulations.

Recent studies have focused more directly on sim-to-real transfer. Li et al. [18] used teleoperation data in an IL/RL hybrid setup for robot learning, before deploying the learned policies onto a real robot for excavation tasks. Duan et al. [19] used vision-based hand gestures and deployed the policy onto a robotic arm for glass-panel installation tasks. The deployment was confined to a controlled testing laboratory.

Despite the growing body of work on RL- and IL/RL-based construction robotics, several key limitations remain. Existing studies have almost exclusively focused on manipulating static objects and have generally been developed and validated either purely in simulation or in highly controlled, confined experimental setups. As a result, the dynamic, unstructured, and partially observable conditions typical of real construction sites, such as changing layouts and varying weather conditions, are largely omitted from previous research. Tackling these limitations is essential for deploying robots on construction sites.

3 Methodology

The aim of the present work is to train an RL policy that guides a 6-DOF robot arm to successfully grasp a cuboid placed within its reach. This section details the RL setup used in this study, focusing on the RL algorithm and environment.

3.1 RL Algorithm

In the context of RL, an agent is characterized by its state ($s_t \in S$), which encodes information that can aid the agent in performing the RL task. During each time step t, the agent interacts with the environment by selecting an action ($a_t \in A$), the effect of which is an updated state (s_{t+1}) and a reward ($r_t \in \mathbb{R}$). Actions are outputted by a policy $\pi(a|s)$, which maps states to actions. A background to RL can be found in [5].

In this project, RL problems are modeled as Markov Decision Processes (MDPs), i.e., they satisfy the Markov property (Equation (1)). An MDP, here, is defined by the quintuple $\{S, A, P, r, \gamma\}$, where $P(s_{t+1}|s_t, a_t)$ is the state transition probability function, r is the reward function, and $\gamma \in]0, 1]$ is the discount factor. The robot's observation ($o_t \in O$) represents the information made available to the agent. In this work, RL problems are fully observable, i.e., $O = S$.

$$P(s_{t+1}|s_0, a_0, \ldots, s_t, a_t) = P(s_{t+1}|s_t, a_t) \tag{1}$$

The objective is to find an optimal policy π^* that maximizes the return ($R \in \mathbb{R}$; Equation (2)) over the course of one episode. An episode, here, refers to the finite number of time steps at which the agent interacts with the environment. Various RL algorithms are used to find π^*, the choice of which is dependent on the nature of the problem, e.g., discrete vs. continuous actions, high dimensionality, etc. A notable subset of these algorithms employs an actor-critic approach. Here, the critic estimates the output of a value function ($V_\pi(s)$), whose expectation corresponds to R, and is trained through the use of environment interactions ($\{s_t, r_t, a_t\} \forall t \in \{0, 1, \ldots, n-1\}$, where n is the number of environment steps collected before each update). The actor, π, is then updated through the critic's value function estimates.

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{2}$$

The RL algorithm used in this study is Proximal Policy Optimization (PPO), an on-policy, actor–critic trust region algorithm that aims to mitigate the effect of overly large policy updates, which can lead to unstable learning. PPO achieves this by clipping the policy update using Equation (3), where ϵ is a hyperparameter that limits the policy update. A more detailed description of the algorithm can be found in [20].

$$|\rho_t - 1| \leq \epsilon, \quad \text{where } \rho_t = \frac{\pi_t}{\pi_{t-1}} \tag{3}$$

In this study, the policy and value function are represented as multilayer perceptrons (MLPs), with three hidden layers, each with dim = 256 and an ELU activation function following each layer. Additionally, the policy network's output was passed through a tanh(.) function to bound the actions to the range $[-1, 1]$. Policy training was conducted using 1024 parallel environments and an adaptive learning rate $\in [1 \times 10^{-5}, 5 \times 10^{-5}]$. Training was considered to have converged once the episode-mean

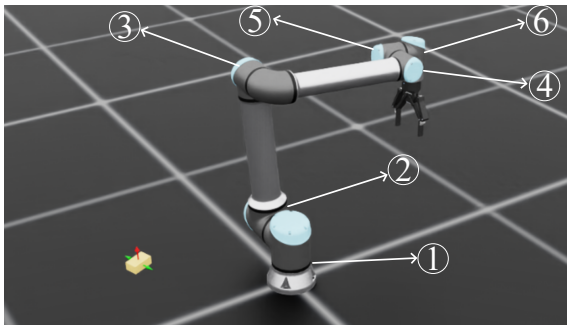

Figure 2. An image of the RL environment used in this study, where the numbers are the joint numbers in Table 1.

reward stagnated for more than 300 consecutive training iterations. The maximum number of iterations was set to 3×10^4.

3.2 RL Environment

The 6-DOF robot used here is a UR10e robot arm. Attached to its end effector is a Robotiq 2F-140, a two-finger gripper. The environment (Figure 2) comprises a single UR10e and a cuboid of dimensions: 3cm×4.5cm×7.5cm. The robot's task is to successfully grasp the object at one of its faces. Accordingly, the object is represented by its centroid, the center of the grasp face, a unit normal vector at this face center, and a tangent vector defining the gripper's closing direction (see Figure 3). The grasp face is determined as a function of the robot's pose: as the robot approaches the object, the face whose normal is most aligned with the robot's approach direction is selected. The grasping (closing) direction is determined in a similar manner (green arrow in Figure 3). To prevent frequent switching between faces and grasping directions, a hysteresis margin ($\phi = 0.1$) is added to stabilize the reward function. Moreover, since Robotiq 2F-140 is a two-finger gripper, the grasping direction is treated as unsigned; for each selected face and tangent, there are two valid grasping configurations.

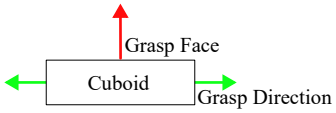

Figure 3. A schematic of the cuboid described in the main text, along with the grasping face's normal direction (red arrow) and the grasp direction (green double-sided arrow).

During training, the problem is split into two events: "has reached" and "has grasped" (Figure 4). A successful grasp triggers the "has grasped" event and terminates the episode. If "has grasped" is not triggered within 5 seconds, the episode is terminated, and the task is considered incomplete. Success rate is, then, defined as presented in Equation (4), where N is the number of episodes used to calculate the mean.

$$\overline{\psi} = \frac{1}{N}\sum_{i=1}^{N}\psi_i,\ \psi_i = \begin{cases} 1 & \text{if event "has grasped" triggers} \\ 0 & \text{else} \end{cases} \tag{4}$$

At the start of each RL episode, the cuboid's position is sampled from within a cylindrical shell with r, θ, and $z \in [0.3\text{m}, 0.83\text{m}]$, $[0, 2\pi[$, and $[0.15\text{m}, 1\text{m}]$, respectively. Additionally, the cuboid's orientation is fully randomized, and the robot is reset to a default joint configuration (Table 1), upon which zero-mean noise (from a uniform distribution) is superimposed.

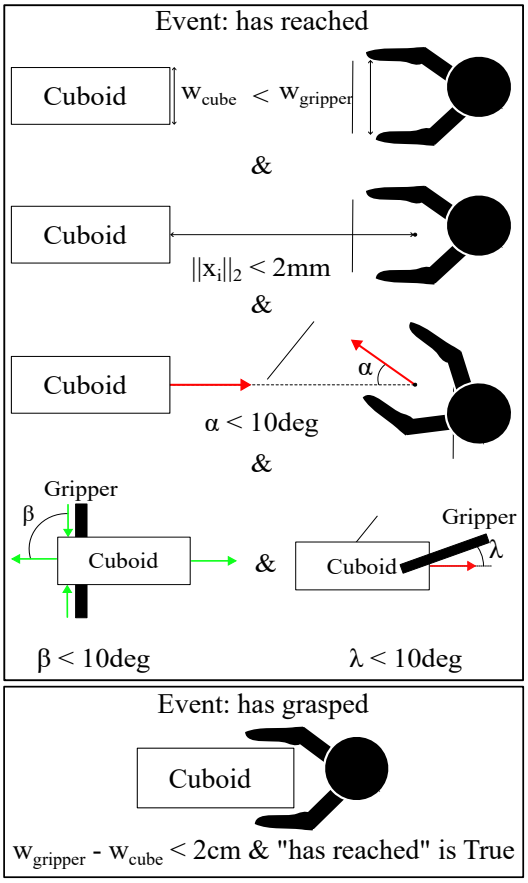

Figure 4. Visual description of the events "has reached" and "has grasped", where "&" represents the binary AND operator, $w_{gripper}$ and w_{cube} are the widths of the gripper and the cuboid's grasp face, respectively; α, β, and λ are the alignment angles.

Table 1. Default UR10e joint positions used in this study. Location of joints can be found in Figure 2

Joint name	Position (radians)
1	π
2	-0.5π
3	0.5π
4	-0.5π
5	-0.5π
6	0

3.3 Observations, Actions, and Rewards

The observation space is a single, 58-dimensional array that is bounded ($o_t \in [-1, 1]$) and contains the variables detailed in Table 2. Similarly, the action space is modeled as a seven-dimensional continuous vector that represents joint velocities (Equation (5)). The first six components correspond to the six joints of the UR10e, while the seventh component controls the Robotiq 2F-140 gripper.

$$a_t = [a_t^1, a_t^2, a_t^3, a_t^4, a_t^5, a_t^6, a_t^g];\ a_t^i \in [-1, 1] \quad (5)$$

In this work, the policy operates at 60Hz. During each time step Δt, the actions are scaled by the corresponding joint and gripper velocity limits such that $a_t = -1$ and $a_t = 1$ represent maximum admissible velocity in either direction. The scaled actions are, then, used to update the joint positions according to Equation (6), where q^i and $\dot{q}^i$ are the joint positions and velocities, respectively, defining the robot's new state. An analogous update is applied to the gripper state using $\dot{q}^g$.

Table 2. Variables added to the robot's observation along with their dimension.

Variable	Dimension
UR10e Joint state	12
Gripper State	8
Selected Face	6
Selected Face's Orientation	9
Cuboid velocity	6
Collision Forces	7
Alignment Angles	2
Grasp Ready	1
Gripper's Relative Pose	7

$$q_{t+1}^i = q_t + \dot{q}_t^i \Delta t \quad (6)$$

The reward function is a weighted sum of bounded functions ($r = \sum w_i r_i$) that is designed to guide the robot through both sub-tasks described in Section 3.2. Its components are presented in Table 3.

4 Experiments and Results

This section presents the results of training the UR10e's RL policy, whose setup was described in Section 3, and the testing of the trained policy on previously unseen cuboid positions and sizes. RL training was conducted using Isaac Lab [21].

Table 3. Reward functions used in this study along with their corresponding weight. * denotes a reward a function that is gated by the event "has reached". ** denotes rewards that are weighted by the end effector's proximity to cuboid.

Reward function	Function type	Weight
Time	Constant	-0.1
Action regularization	Quadratic	-0.01
Self-collision forces	Linear	-0.02
Cuboid velocity	Linear	-1
L_2 distance	Polynomial	0.2
L_2 distance ($t-1 \rightarrow t$)	Linear	0.5
Distance components	Linear	-0.5
Alignment angles	Cosine	0.05
Alignment bonus	ReLU**	0.2
Gripper gap	ReLU*	-0.5
Gripper action	ReLU*	2
Gripper width ($t-1 \rightarrow t$)	Linear*	2
"has reached" event	Sparse	0.05
"has grasped" event	Sparse	100

4.1 Training

Figure 5 presents the variation of the episode-mean reward and mean episode length during training. Since shorter episodes indicate quicker grasps, the initial plateau in episode length suggests that the sharp early reward increase is driven by alignment rewards. The subsequent exponential-like decay (iteration $\gtrapprox$ 700) is a resultant of more frequent, successful grasps. Post-convergence (not shown here), the two curves remain coupled: decreases in the reward are accompanied by increases in the mean episode length, while the alignment terms continue to improve.

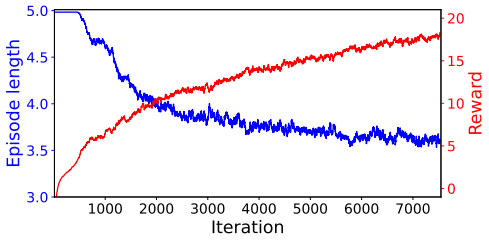

Figure 5. Variation of the mean episode length (in seconds; blue line) and the episode-mean reward (red line) during training. Both variables are plotted on separate y-axes. Episode length: left axis. Reward: right axis. All post-convergence iterations are excluded from the figure. Note: Values were averaged using an exponential moving average (25 iterations).

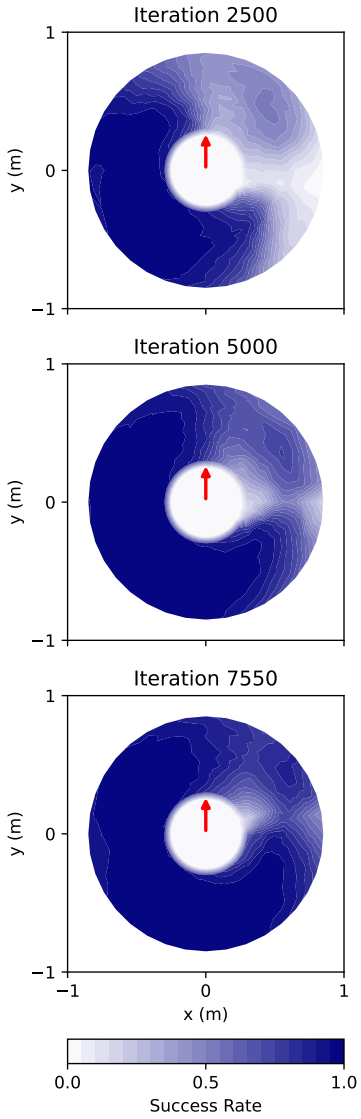

Figure 6. Contours of the success rate (Equation (4)) obtained during training on the $z = 0.5$ m plane, presented for the policies obtained at iterations (from top to bottom): 2500, 5000, and 7550. Cuboid positions were sampled from a cylindrical shell whose in-plane dimensions are identical to those described in Section 3.2, with $[\Delta r, \Delta\theta] = [5\,\text{cm}, 10°]$. Red arrows depict the robot's initial direction.

Figure 6 shows contours of the success rate (Equation (4); $N = 1000$) on the $z = 0.5$m plane, obtained using the policies after 2500, 5000, and 7550 iterations. Each point in Figure 6 denotes the cuboid's position at the start of the episode. During each episode, the cuboid's orientation was selected at random. A clear contrast between $x < 0$ and $x > 0$ can be observed. For $x < 0$, the success rate rises quickly and is high for all three policies. The main differences between iterations are thus concentrated in the $x > 0$ region: comparing the three policies shows a notable improvement in this half-space. Although at iteration 7550, a band of low success rate persists near $y \approx 0$. Note that the number of sampled cuboid positions is the same for $x < 0$ and $x > 0$. Therefore, this asymmetry cannot be attributed to an imbalance in the training samples.

4.2 Success Rate

To analyze the policy's robustness with respect to the cuboid position, the success rate was evaluated in RL environments with previously unseen cuboid positions, i.e., positions that were not encountered during training. Cuboid positions were sampled from the domain $r \in [0.83\,\text{m}, 1.2\,\text{m}]$, $\theta \in [0, 2\pi[$, and $z = 0.5\,\text{m}$, while cuboid orientations were randomized. Success rates were then calculated using Equation (4), with $N = 1000$. Note that larger values of r would be beyond the robot's reach. The resulting contours of the success rate are shown in Figure 7.

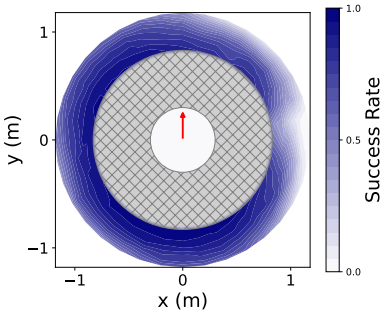

Figure 7. Contours of the success rate (Equation (4)) on the $z = 0.5$m plane for the trained policy obtained at iteration 7550. Cuboid positions were sampled from a cylindrical shell with $r \in [0.83, 1.2]$ m, $\theta \in [0, 2\pi[$, and $[\Delta r, \Delta\theta] = [5\text{ cm}, 10°]$. Red arrow depicts the robot's initial direction. Gray meshed area is the training domain.

It can be seen that the minimum success rates occur for $x > 0$ and $y \approx 0$. In this region, success rates range from 0% to ~ 40% and are the lowest across all values of r. In contrast, success rates in the remaining regions are higher and exhibit a smaller variability with respect to r, ranging from ~ 70% to 99%. The maximum, in this case, occurred at $[x, y] = [-0.28\,\text{m}, -0.78\,\text{m}]$.

The spatial distribution of the success rate closely mirrors that observed during training (Figure 6), indicating that the policy generalizes well to previously unseen cuboid positions and does not simply overfit to the training samples. In both the training and test evaluations, however, the policy consistently underperforms in the $x > 0$,

$y \approx 0$ region, suggesting that this area of the workspace is particularly difficult for the trained policy. Further training that specifically targets this region could, therefore, improve overall robustness with respect to the cuboid position. Though the performance in all other regions demonstrates a high degree of robustness and generalization.

4.3 Cuboid Size Variation

To test the policy's robustness with respect to the cuboid size, the cuboid position was fixed at $[x, y, z] = [-0.22\text{m}, -0.61\text{m}, 0.5\text{m}]$ and its dimensions were then uniformly scaled using Equation (7), such that a scaling factor of $\sigma = 1$ corresponds to the cuboid size used during training. For each σ, the cuboid's orientation was randomized, and the success rate was computed according to Equation (4) with $N = 1000$.

$$[\text{length, width, depth}] = [3\sigma, 4.5\sigma, 7.5\sigma]\ \text{cm} \quad (7)$$

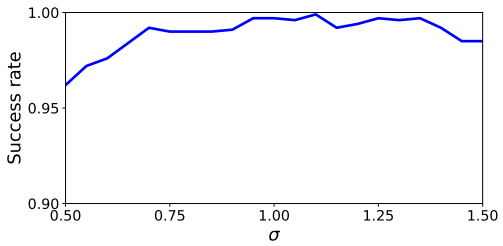

Figure 8. Variation of the success rate (Equation (4)) as a function of the cuboid size for the trained policy obtained at iteration 7550. Size values were normalized such that $\sigma = 1$ corresponds to the cuboid size used during training. Data were sampled for a fixed cuboid position at $[x, y, z] = [-0.22\text{m}, -0.61\text{m}, 0.5\text{m}]$, with $\Delta\sigma = 0.05$.

Figure 8 shows the variation of the success rate as a function of $\sigma \in [0.5, 1.5]$. Overall, the policy is highly robust to changes in cuboid size, with success rates ranging from 96% to 99% across the tested range. The slight drop in performance observed for $\sigma \lesssim 0.75$ is likely due to the smaller cuboids requiring more precise alignment of the end-effector during grasping, which makes the task more demanding for the trained policy. A stricter definition of "has reached" (Figure 4) will likely result in a more uniform success rate.

5 Conclusion

In this paper, an RL policy was trained for controlling a UR10e robot with a Robotiq 2F-140 gripper in an RL environment where its task was to grasp a cuboid positioned at an arbitrary pose in space. The algorithm Proximal Policy Optimization (PPO) was used here, with both the policy and value functions represented by identical MLPs.

Training results showed that the grasp success rate is not axisymmetric: initial cuboid positions at $x < 0$ exhibit notably higher success rates than those at $x > 0$, particularly near the $y = 0$ line. This disparity was also observed for previously unseen cuboid positions, indicating that it is a persistent feature of the learned policy rather than an imbalance with the training samples. With the exception of this region, however, the policy demonstrates robust performance, with grasp success rates ranging from 70% to 99% across the evaluated cuboid positions. Similarly, varying the cuboid size had no significant effect on the grasp success rate, indicating that the policy is robust with respect to changes in cuboid dimensions.

This work corresponds to the first milestone of the overall project, whose aim was summarized in Figure 1. Future work will focus on the next milestones and will involve deploying the trained policy onto a real robot, followed by incorporating vision input into the observation space, enabling the investigation of more complex object geometries, such as formwork. Additionally, later milestones involving the stabilization of moving crane loads will incorporate domain knowledge, e.g., crane-load dynamics, to improve robustness and convergence.

6 Acknowledgments

We thank the German Research Foundation (DFG) for funding this project under grant FOR 5672, subproject A, "Generic Framework for Simulating and Executing Robotized Construction" (grant number 517965147), and the TUM Innovation Network "CoConstruct."

References

[1] Stefana Parascho. Construction Robotics: From Automation to Collaboration. *Annual Review of Control, Robotics, and Autonomous Systems*, 6(Volume 6, 2023):183–204, May 2023. doi:10.1146/annurev-control-080122-090049.

[2] Yuanshen Ji and Fernanda Leite. Automated tower crane planning: leveraging 4-dimensional BIM and rule-based checking. *Automation in Construction*, 93:78–90, 2018. doi:10.1016/j.autcon.2018.05.003.

[3] M R Kolani, Stavros Nousias, and André Borrmann. Robotized tower crane: Simulation and high-level action planning system. In *Proceedings of the 42nd International Symposium on Automation and Robotics in Construction (ISARC)*, 2025.

[4] Yifei Xiao, T. Y. Yang, and Fan Xie. Autonomous construction framework for crane control with enhanced soft actor–critic algorithm

and real-time progress monitoring. *Computer-Aided Civil and Infrastructure Engineering*, 2025. doi:10.1111/mice.13427.

[5] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[6] Qiqi Zhou, Hui Ding, Teng Chen, Luxin Man, Han Jiang, Guoteng Zhang, Bin Li, Xuewen Rong, and Yibin Li. ALARM: Safe Reinforcement Learning With Reliable Mimicry for Robust Legged Locomotion. *IEEE Robotics and Automation Letters*, 10(7):6768–6775, July 2025. doi:10.1109/LRA.2025.3572427.

[7] Baeksuk Chu. Modeling of a mobile manipulator for redundancy resolution. In *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, volume 29, page 1. IAARC Publications, 2012.

[8] Sérgio Ronaldo Barros dos Santos, Sidney N. Givigi, and Cairo Lúcio Nascimento. Autonomous construction of structures in a dynamic environment using reinforcement learning. In *2013 IEEE International Systems Conference (SysCon)*, pages 452–459, 2013. doi:10.1109/SysCon.2013.6549922.

[9] Sérgio R. Barros dos Santos, Sidney N. Givigi, and Cairo L. Nascimento. Autonomous construction of multiple structures using learning automata: Description and experimental validation. *IEEE Systems Journal*, 9(4):1376–1387, 2015. doi:10.1109/JSYST.2014.2374334.

[10] Sérgio R. Barros dos Santos, Diego O. Dantas, Sidney N. Givigi, Luciano Buonocore, Areolino A. Neto, and Cairo L. Nascimento. A stochastic learning approach for construction of brick structures with a ground robot. *IFAC-PapersOnLine*, 50(1):5654–5659, 2017. doi:10.1016/j.ifacol.2017.08.1114. 20th IFAC World Congress.

[11] Yifang Liu, Seyed Mahdi Shamsi, Le Fang, Changyou Chen, and Nils Napp. Deep q-learning for dry stacking irregular objects. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1569–1576, 2018. doi:10.1109/IROS.2018.8593619.

[12] Aiyu Zhu, Pieter Pauwels, and Bauke de Vries. Robot construction simulation using deep reinforcement learning. In *EG-ICE 2020 Workshop on Intelligent Computing in Engineering, Proceedings.*, 2020.

[13] Sérgio Ronaldo Barros dos Santos, Cairo Lúcio Nascimento Júnior, and Sidney N. Givigi. Planning and learning for cooperative construction task with quadrotors. In *2014 IEEE International Systems Conference Proceedings*, pages 57–64, 2014. doi:10.1109/SysCon.2014.6819236.

[14] Dong Liu, Jianfu Cao, and Xiaokang Lei. Slabstone installation skill acquisition for dual-arm robot based on reinforcement learning. In *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1298–1305, 2019. doi:10.1109/ROBIO49542.2019.8961805.

[15] Juan Manuel Davila Delgado and Lukumon Oyedele. Robotics in construction: A critical review of the reinforcement learning and imitation learning paradigms. *Advanced Engineering Informatics*, 54:101787, October 2022. doi:10.1016/j.aei.2022.101787.

[16] Aleksandra Anna Apolinarska, Matteo Pacher, Hui Li, Nicholas Cote, Rafael Pastrana, Fabio Gramazio, and Matthias Kohler. Robotic assembly of timber joints using reinforcement learning. *Automation in Construction*, 125:103569, 2021. doi:10.1016/j.autcon.2021.103569.

[17] Ci-Jyun Liang, Vineet R. Kamat, and Carol C. Menassa. Teaching robots to perform quasi-repetitive construction tasks through human demonstration. *Automation in Construction*, 120:103370, 2020. doi:10.1016/j.autcon.2020.103370.

[18] Yan Li, Songyang Liu, Mengjun Wang, Shuai Li, and Jindong Tan. Teleoperation-Driven and Keyframe-Based Generalizable Imitation Learning for Construction Robots. *Journal of Computing in Civil Engineering*, 38(6):04024031, November 2024. doi:10.1061/JCCEE5.CPENG-5884.

[19] Kangkang Duan, Zhengbo Zou, and T. Y. Yang. Training of construction robots using imitation learning and environmental rewards. *Computer-Aided Civil and Infrastructure Engineering*, 40(9):1150–1165, 2025. doi:10.1111/mice.13394.

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

[21] Mayank Mittal et al. Isaac Lab: A GPU-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025. URL `https://arxiv.org/abs/2511.04831`.