Reinforcement learning with construction robots: A preliminary review of research areas, challenges and opportunities

X. Xu [1] and B. García de Soto [1]

[1] S.M.A.R.T. Construction Research Group, Division of Engineering, New York University Abu Dhabi (NYUAD), Experimental Research Building, Saadiyat Island, P.O. Box 129188, Abu Dhabi, United Arab Emirates

E-mail: xx927@nyu.edu; garcia.de.soto@nyu.edu

Abstract

The interest in the use of robots in the construction industry has been around for several decades; however, the advancement of technology for related applications has been slow. Considering that most construction robots are not fully automated and require extra guidance and instructions from the operators, an autonomous way for robots to understand how to execute specific construction tasks is needed. Reinforcement learning (RL) is a possible solution to this problem. Instead of explicitly detailing the solution to a problem, RL algorithms enable a robot to autonomously discover an optimal behavior through trial-and-error interactions with its environment. It constructs a learning model to solve various sequential decision-making problems. RL algorithms could help construction robots establish a learning process based on the feedback from the construction site and lead to an optimal strategy to finish the sequential construction work. Nowadays, on-site construction robots still require tedious work in preprogramming. Many other areas have used RL algorithms on different tasks, such as dexterous manipulation, legged locomotion, or pathing planning; thus, there is great potential for combining construction robot applications with RL algorithms.

Despite these achievements, most works investigated single-agent settings only. However, many real-world applications naturally comprise multiple decision-makers that interact simultaneously, such as traffic modeling for autonomous vehicles and networking communication packages for multi-robot control. These applications have faced significant challenges when dealing with such high-dimensional environments, not to mention the challenges for the on-site construction robots. The recent development of deep learning has enabled RL methods to drive optimal policies for a sophisticated high-dimensional environment. To the best of our knowledge, there is currently no extensive survey of the applications of RL techniques within the construction industry. This study can inspire future research into how best to integrate powerful RL algorithms to achieve a higher-level autonomous control of construction robots and overcome resource planning, risk management, and logistic challenges in the industry.

Keywords –

Construction robot; Deep reinforcement learning; Multi-agent; Task allocation; Path planning

1 Introduction

With the expansion of technologies, the construction industry has seen significant attempts at robotization. Some single-purpose robots were designed to conduct specific tasks in highly controlled environments. These applications have shown promising results for single repetitive tasks. However, all these robotic applications do not have the capabilities to adapt their operations to unique work environments. Their motions implement mobility, and manipulations are preprogrammed by the expertise considering the characteristics of tasks and work environments, which limited the wide usage of such robotic systems. Thus, instead of preprogramming all the details, a way for the robot to learn by itself and adopt different scenarios on the construction site is necessary to develop construction robot applications. Reinforcement learning (RL) seems to be a promising solution to these problems.

RL and optimization control theory are used to solve a wide range of tasks using a simple architecture where the agent operates in an environment that models the task it needs to fulfill. RL studies how to use past data to enhance the future manipulation of a dynamical system that adequately adapts to environmental changes [1]. Over the past decades, advances in RL have led robotics to be highly automated and intelligent, with safe operation instead of manual work for many challenging tasks. As an essential branch of machine learning, RL can realize sequential decision-making and has made a series of significant breakthroughs in robot applications. It has led to a wide range of impressive progress in various domains, such as industrial manufacturing [2], board games [3], robot control [4], and autonomous driving [5]. Although so much success has occurred in various robotic domains, few studies have focused on robots in construction applications.

The lack of attention for RL in the construction industry is due to its complexity [6]: with too many uncertainties on the job site and massive data from the dynamic environment, it is tough for the robot to model the tasks, set up the simulator, and get predictions from past results. However, considering the development of the digitalization of the construction industry, with the more standardized construction process and more modular products coming into the market, the construction work is far more predictable than it used to be. Besides, with BIM models becoming the norm in most job sites, the numerous information stored in the digitalized model can provide reliable data for simulation of the robot's behavior. Thus, this study aims at establishing the connections between RL and construction robotic applications, trying to advance the functionality and reliability of the current construction robot systems.

Therefore, a review of reinforcement learning applications in construction is presented herein to guide subsequent research. This paper aims to help researchers working in this field quickly place their work within the current spectrum, bearing in mind the current challenges and potential. The rest of this paper organizes as follows: Section 2 summarizes the state of the art of the algorithms and gives a brief overview of RL applications. Section 3 provides a framework to figure out how to apply these algorithms to construction robot applications. Section 4 presents a simple case study to adopt the RL algorithms and explains how to set up the algorithms and the environment. Section 5 discusses the results, and Section 6 summarizes the findings and provides further advancement for future research.

2 State-of-the-art RL

2.1 RL Overview

RL is about training an agent to interact with its environment. The agent arrives at different scenarios known as states by performing actions. Actions lead to rewards which could be positive or negative [8].

Some key terms that describe the essential elements of an RL problem are as shown in Figure 1:

a. *Environment* — Physical world in which the agent operates
b. *State* — Current situation of the agent S_t, Next situation S_{t+1} after taking an action.
c. *Action* — Step A_t taken by the agent when in a particular state.
d. *Reward* — Feedback from the environment R_t.
e. *Policy* — Method to map an agent's state to actions.
f. *Q-Value* — Expected return $q_\pi(s, a)$ starting from state S_t, following policy π, taking action A_t, used to determine how good an action is.

The RL process is a cycle that begins with the observation of a current state, choosing an action, observing the received state, and updating the evaluation of its value function based on the action taken. After that, the next cycle begins.

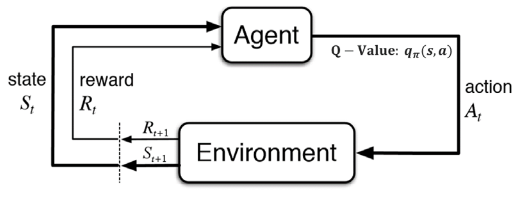


Figure 1. Reinforcement learning scheme

If the action's highest quality value (Q-value) is selected in every state, it results in the optimal policy. Q-learning finds an optimal policy in the sense of maximizing the expected value of the total reward over any successive steps, starting from the current state.

Policy Gradient is another popular method. It does not calculate the Q-value but instead uses a policy. The policy learns a mapping from every state to act, and its objective is to find which actions lead to higher rewards and increase their probability. The policy gradient observes the environment, acts in it and keeps updating its policy based on its rewards. After multiple iterations, the policy converges to a maximum value.

In section 4, our example that illustrates how to adopt Deep Reinforcement Learning (DRL), and Multi-agent Reinforcement Learning (MARL) on construction robots, will mainly focus on these two most used methods.

2.2 DRL to handle the dynamic environment

In situations with extensive state and action spaces, it is not feasible to learn the policy for each state and action pair. DRL allows agents to make decisions from high-dimensional and unstructured input data [9] using neural networks to represent policies. DRL combines both the technique of giving rewards based on actions from RL and the idea of using a neural network for learning feature representations from deep learning [10]. DRL is among the most promising algorithms when a predefined training dataset is required, which suits robotic manipulation and control tasks [11].

DRL is firstly used in video games and simulated control, which does not connect with the constraints of learning in natural environments. In 2015, Mnih et al. [7] used a Deep Q-Network (DQN) structure to create an agent that outperformed a professional player in 49 classic Atari games. With the development of algorithms, DRL has also demonstrated promises in enabling physical robots to learn complex skills in the real world [12]. For example, DRL algorithms allow robots to learn tasks such as door opening [13] and to grasp and assemble objects using manipulators [14] with low-dimensional state observations.

With the exact logistics, by adopting the DRL algorithms to construction robots, robots can be trained in a dynamic environment. with simulation or execution in the real world, the robot will generate its dataset autonomously by accumulating its experience. And then, with iterations of training, the robot could find the optimal actions to take from the dataset generated when given an initial state.

2.2.1 MARL to solve multi-agent situations

As real-world problems have become increasingly complicated, there are many situations where a single DRL agent cannot cope. Traditional RL approaches such as Q-Learning or policy gradient are poorly suited for multi-agent environments. One issue is that each agent's policy changes as the training progresses, and the environment becomes non-stationary from any individual agent's perspective [15]. Learning among the agents sometimes causes changes in an agent's policy and can affect the optimal policy of other agents. The estimated potential rewards of action would be inaccurate, and therefore, good policies at a given point in the multi-agent setting could not remain so in the future. To solve this problem, based on the current literature, generally, two ways are used chiefly to solve this multi-agent framework.

The centralized learning approach assumes a joint model for all the agents' actions and observations [16]. A centralized policy maps the joint observation of all the agents to joint action. A significant drawback of this approach is that it is centralized in both training and execution and leads to an exponential growth in the observation and actions spaces with the number of agents.

The centralized paradigm is usually a beneficial choice for exchanging information between agents, and problems like non-stationarity, partial observability, and coordination can be diminished [17]. In 2016, Mnih et al. [18] introduced a lightweight approach based on actor-critic architecture by simultaneously training multiple agent-environment instances. As a result, many other works have started to extend the state-of-the-art single-agent asynchronous advantage actor-critic algorithm to enable multi-agent training to accomplish more sophisticated tasks. Elfakharany et al. [19] proposed another framework using the policy gradient method known as Proximal Policy Optimization (PPO) to allocate path planning globally in a shared environment among multiple robots.

The second method uses decentralized learning. In the decentralized execution paradigm, the agents make decisions independently according to their individual policies [16]. Multiple agents work together to learn a homogeneous, distributed policy toward a common goal without explicitly interacting. One of the advantages of this approach is that it makes the learning of heterogeneous policies easier. It can be beneficial in domains where agents may need to take on specific roles to coordinate and receive a reward. For example, Sartoretti et al. [20] proposed an approach that relied on decentralized policy in an entirely visual system and succeeded in multi-agent-based brick construction. With a similar actor-critic architecture. However, training individual policies does not scale to many agents. Because the agents do not share the experience, this approach adds additional sample complexity to the RL task. Second, as the agents are learning and adjusting their policies, the change in the policies make the environment dynamics non-stationary. Stored experiences can be quickly meaningless due to the changing dynamics of other agents.

In addition to these two generally used methods, to speed up the learning procedure, another method can be used as a supplement by adding prior knowledge to guide the multi-agent learning process. Prior knowledge can dramatically help guide the learning process. These approaches significantly reduce the search space and, thus, speed up the learning process [21]. For example, in [22, 23], both the prior knowledge and prior rules are used to improve the DQN algorithm to solve the multi-robot path-planning problem.

3 RL with construction robots

RL has been widely adopted in different industries such as video games, marketing, public services, and manufacturing; however, there is no review of combining RL techniques with the construction industry. Aiming to inspire future research into solving the single-agent and multi-agent construction task execution problem, this section presents a survey discussing the possible challenges, opportunities, and research areas for the adoption of RL with construction robots based on previous studies and applications in relevant areas.

3.1 Challenges of construction robot training

3.1.1 Hard to build up the training environment

As an essential branch of machine learning, In comparison to supervised and unsupervised learning, few works have been published using the RL algorithm in the construction research field. The main reason for this may be that RL is a trial and error-based algorithm, which is hard for researchers to build up the training scheme in construction scenarios [24]. It is hard to initialize the agent(s) and the environment. To address that, RL research has generally been divided into two types of works [10]. The first type uses simulated control by testing virtual data in a simulated environment to verify the feasibility of robotic manipulation in the real world.

The second type is to ask the robot to sense the natural environment and learn directly in a real-world environment through trials and errors.

Due to the existence of many complex and random factors on the construction site, it is often impractical to develop an explicit construction environment based on the real-world situation. Construction sites are dynamic. Tracking changes and simulating real-world physics is hard in real-time. Besides, the simulation of robot models is also time-consuming. Generally, the simulation can only mimic the situation in small-scale scenarios, which is insufficient for large-scale construction operations. For real-world RL learning, errors or collision happens, and construction robotic systems are fragile and expensive. Therefore, it is not wise to test the robot directly on a dangerous construction site.

3.1.2 Construction tasks are hard to simulate

As the construction site evolves, it is hard to model the dynamics between states and actions and set up constraints and reward functions. In addition, construction tasks are unique for the construction application. No generalized building principles can be applied to all tasks or projects. Understanding the mapping between actions and states requires much expertise and prior knowledge to solve the physical equations and write down the scripts to control the robot, not to mention writing out the temporally and spatially coupled operational constraints or setting up specific goals for a characterized structure.

Moreover, it is challenging to solve large-scale construction work in real-time. When optimization is needed, these methods must compute all possible solutions entirely or partially and choose the best one, so the computation process is time-consuming when the solution space is vast [25].

3.1.3 Multi-workers working in the same environment with different roles

The construction industry is overwhelmed with resource planning, risk management, and logistic challenges, resulting in design defects, project delivery delays, cost overruns, and contractual disputes. Systematically putting all these factors into robotic learning is also a challenging job.

A construction job is a typical scenario that involves interaction among multiple agents, where emergent behavior and complexity arise from agents evolving together. For example, in collaborative construction activity, different agents have different tasks in the same environment to complete the activity. We need to define the working area for each individual and show them the path of their work routine to avoid physical collisions. Besides, because of the resource or material limitations, we need to wisely allocate the resources, arrange the sequence of operations for each agent, and build up a scheduling network. All these happen in a multi-agent domain. Successfully scaling RL to environments with multiple agents is crucial to improving construction robots' eligibility.

3.2 Opportunities

3.2.1 Mega data generated and stored in the construction industry

The Building Information Model (BIM) is a crucial contribution to the construction industry. The BIM consists of a three-dimensional graphical reproduction of the building geometry and a related database in which all data, properties, and relations are stored [26]. BIM provides digital models for RL simulation and massive data stored to represent components on the job site. With this information fed to the robot, we could build up a real-time simulator for RL training, and a more accurate construction process could be simulated to represent the physical feedback. The virtual infrastructure represented in augmented reality (AR) will improve the user interaction with intelligent infrastructure for specific applications such as maintenance, training, and wayfinding. This could solve the first challenge that the construction industry lacks task planning simulators.

3.2.2 Development of RL algorithms

As stated in section 2, combined with developments in deep learning, DRL has emerged as an accelerator in related fields. DRL can help solve the problem of finding the optimal policy between state and actions through neural networks. Also, it can speed up the training process to make the simulation of construction tasks feasible. Besides, from the well-known success in single-agent DRL, such as Mnih et al. [7], we now witness a growing interest in its multi-agent extension to simulate and find the optimal strategy in a higher dimension. This can better represent the real situations on the job site and show the potential to combine different roles of robotic workers to collaborate just like human beings.

3.2.3 Human-robot interaction

As stated in Section 2, one can ease the training process by combining it with the proper operation in advance, just as the supervised learning, but somehow it will not influence the general structure of the DRL algorithm. The only contribution is to speed up the convergence and add the correct episode into the simulation process with a user-defined map. As [27] stated, there has been a rising demand to provide human feedback in the agent training process. Another more feasible method is proposed by creating a goal map, which integrates human strategies before the training process [28]. RL algorithms provide other choices in human-robot collaboration with only limited prior knowledge from human beings. New formats such as augmented reality (AR) or game-engine-based simulation can be further developed to teach the robot how to initialize the learning process, prevent unsafe behavior, and define the goal or reward accordingly.

3.3 Research areas

After identifying some of the challenges and opportunities for RL-based construction robot applications, we put forward some possible research areas to guide future research on such innovative systems.

3.3.1 Simulator set up

First, as we identified in the Challenges and Opportunities sections, capturing the real-world environment into simulated augmented ones is a priority for RL-based construction robots to run the simulation. BIM-based digital twins could provide detailed information and the possibility to track real-time changes during construction. The communication between such models with the robotic system needs a platform to connect robotic training scripts with the BIM environment. A possible solution is integrating the ROS system with game engines such as Gazebo, Unreal or Unity to render the environment and imitate the physics from the real world. The combination between ROS and BIM (digital twin) allows the user-defined scripts for robotic control and visualization in real-time. This could be a critical topic for future research.

3.3.2 DRL algorithm advancing

Second, developing RL algorithms for robotic control is always a trend for advancing such systems. For example, algorithms that allow a mobile robot to reach target positions and navigate safely are open research fields [29]. Besides, many researchers are working on robotic manipulation, such as grasping and door-opening robots. In this way, there is great potential for us to develop the RL algorithms based on construction activities. Many researchers have succeeded in implementing the single repetitive work of construction robots, such as bricklaying and drilling. With the formula representing the construction dynamics, we can quickly adapt the RL algorithm structure to allow construction robots to fulfill such tasks in a more dynamic environment. The robot will then learn how to modify its actions to adapt to the changes in the environment, making the construction robot brighter and more easily implemented in the industry.

3.3.3 MARL algorithm advancing

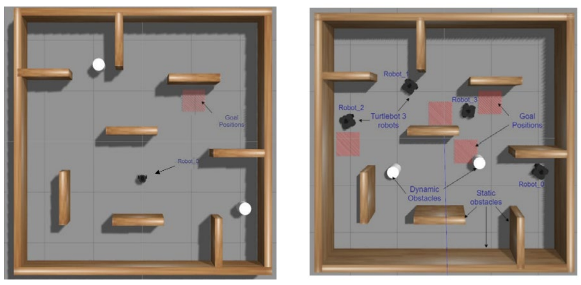
Finally, instead of low dimensional observation on the single-agent repetitive tasks, a higher dimensional network for MARL will automate construction tasks in collaborative ways just like human beings. As [30] stated, a higher level of on-site automated robots should fulfill perception, mobility, and manipulation tasks. Based on this and the essential characteristics of construction work, we can categorize the MARL algorithms into two parts [19]. Multi-Robot Task Allocation (MRTA) and Multi-Robot Path Planning (MRPP) as two separate steps, each with its own set of algorithms. The MRTA algorithm assigns each robot to a task to determine the optimal way to allocate tasks and resources, follow the constraints and manipulate specific tasks under a predefined schedule. The MRPP algorithm guides each robot through the environment towards the assigned goal position while avoiding both static and dynamic obstacles such as temporary structures and moving laborers or other robot agents.

4 Case study

This section presents a path planning mobile robot application developed based on the [31] to set up a foundation for RL-based construction robots in single-agent and multi-agent scenarios.

4.1 Simulator set up

We developed an experimental test for goal-oriented navigation and obstacle avoidance tasks using a TurtleBot3 Waffle Pi in the Gazebo simulation environment (Figure 2a) for behavior learning in autonomous agents.



(a) (b)

Figure 2. Gazebo environment for single-agent (a) and multi-agent robot environment path planning (b)

In the Gazebo simulation, to define the state of the robot, we need to observe the environment and describe the spatial relationship between them. Besides, to measure the right movement of the robot, we also need to sense the position of the virtual markers. Thus, with 24 sensors embedded in the robot model in the simulated environment, we define the state size for this single robot setting to be 26, 24 LDS (Laser Distance Sensor) values, along with the other two values: distances to goal and angle to goal.

Where LDS denotes the (24) values that the LIDAR sensor emits, the Distance represents the distance to the goal, and the Angle is the angle between the robot heading and vector to the goal.

4.2 DRL algorithm for single-agent robot navigation

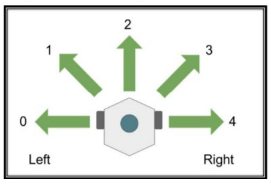
The reinforcement training algorithms were run on a GPU with NVIDIA GeForce GTX 2060 (the CPU was an

8-Core Intel (R) Xeon (R) CPU E7) and implemented using the OpenAI Baselines package with DQN and DDPG. The DDPG and DQN algorithms were modified to process a weighted reward.

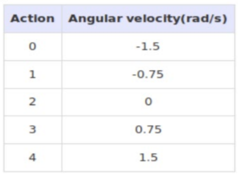
The goal is to train a Deep Q-Networks (DQN) agent to learn an optimal policy to navigate the robot from the initialized position to a goal position (user set up) with minimum effort.

Besides, we also adopted the Deep Deterministic Policy Gradient (DDPG) agent-based training algorithm, which allows continuous control of a robot. In our case we have linear velocity (0 to 0.22 m/s) and angular velocity (-1 to 1 rad/s) as outputs.

The robot has five actions that can act on depending on the type of state (Figure 3). The robot has a fixed linear velocity of 0.15 m/s, which determines the angular velocity. The linear velocity has discrete actions, as shown in Figure 3a. The angular velocity is determined by the state and the linear velocity, as shown in Table 1. The corresponding values are shown in Figure 3b. Table 1 also shows the reward functions.



(a)



Action	Angular velocity(rad/s)
0	-1.5
1	-0.75
2	0
3	0.75
4	1.5

(b)

Figure 3. Direction of the different actions (a) and corresponding angular velocity (b)

Table 1. Reward function equation and parameters

Reward function	Parameter
$\theta = \frac{\pi}{2} + act * \frac{\pi}{8} + \varphi$	$\theta:$ *Angle from goal*
	$\varphi:$ *Yaw of robot*
$R_\theta = 5 * 1^{-\theta}$	R_θ : *Reward from angle*
$R_d = 2^{\frac{D_c}{D_g}}$	R_d : *Reward from distance*
	D_c: *Current distance from goal*
	D_g *Absolute distance from goa*
$R = R_\theta * R_d$	$if -\frac{1}{2}\pi < \theta < \frac{1}{2}\pi$
	$R_\theta \geq 0$ *else* < 0

4.3 MARL algorithm for robot navigation

For the MARL algorithm, we used the DQN network to train four TurtleBot3 robots in the same working environment to achieve different goals (Figure 2b). We used the centralized learning or decentralized execution paradigm as discussed in the literature, in which each robot has a copy of the policy network, and each robot collects its data (O_i^t, a_i^t, r_i^t) from the environment. Each robot (i) at time step (t) receives an observation (O_i^t) and calculates the output action (a_i^t) that drives the robot from the start position towards the goal position. The observation for each robot is composed of several parts: The first part is the relative positions of all the goals in the robot's local polar coordinates. The second part is the relative positions of all the goals in the other moving robots' local polar coordinates. Last but not least is the laser scanner measurements. The reward function r_i^t is designed to ensure each robot move towards the unique zone set up separately. It penalizes getting near obstacles, colliding with other robots, or reaching the wrong goal position.

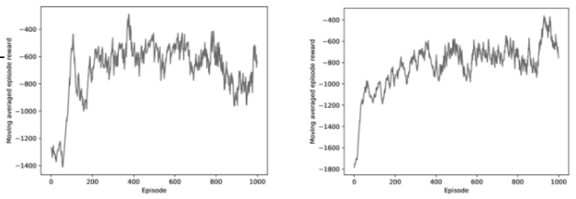
After each episode, it sends data rollouts to a centralized copy of the policy. The gradients are then calculated on the centralized policy, and the centralized policy is updated. After that, each robot receives a copy of the updated policy weights to start collecting a new batch of data. The episode ends when either the robots have a collision, when all the robots reach all the goal positions, or when the episode duration is exhausted.

5 Results

5.1 Single-agent DRL

To compare DDPG and DQN, it was necessary to define metrics. The reward was chosen as the primary indicator. Simulation results in a world of fixed obstacles and random start and end positions are shown in Figure 4. DQN achieved the average target score somewhere after 200 episodes, with each episode consisting of a maximum of 120 time steps (Figure 4a). Training a DDPG generally tends to take more time compared to DQN. In this case, it took about 600 episodes to achieve the average score (Figure 4b).

One of the reasons is that the number of parameters to deal with in DDPG is much higher than in DQN and requires more computation resources. DQN is a value-based learning method, whereas DDPG is an actor-critic method. With DDPG, we have to fine-tune not one but two neural network models. Moreover, since the performance of the actor model strongly depends on the critic's performance, they both must have proper stable growth, which is quite challenging to assure.



(a) (b)

Figure 4. DQN reward (Temporal difference) (a) and DDPG reward (Actor-critic) (b)

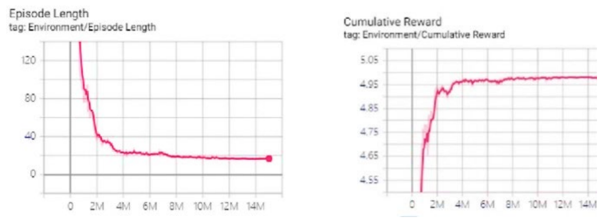
The second reason is that DQN is training under discrete action space, with less uncertainty than the DDPG ones and continuous action selections. Although DQN was successful in more significant dimensional problems, such as the Atari game, the action space is still discrete. However, for many tasks of interest, especially the physical control tasks of robots, the action space is mostly continuous.

5.2 Multi-agent DRL (MARL)

5.2.1 Decentralize learning

Figure 5b shows that the cumulative reward increases over time and reaches a stabilized value, which means the decentralized training succeeds in collaborative path planning. The episode length, Figure 5a, shows the length of each episode in the environment after all agents reach the goals. The agents struggle to complete the tasks within the episode limit for the initial few episodes, but we observe a drop in the episode length as the agents learn. Finally, it reaches a platform, which indicates that it takes 20 s for all the robots to reach the assigned goals in each episode. For decentralized learning, it is easy to get to the desired reward with limited time steps. The steps to take are gigantic, requiring almost 4 million episodes of training periods; this is because multi-agent collaboration leads to an exponential growth in the observation and actions spaces with the number of agents.

Nevertheless, the results are faster converge than the centralized training. Because all the agents do not need to share the experience, which eliminates additional sample complexity to set up the neural networks, it will take shorter to find the optimal strategies. Besides, the final result seems robust with a static platform. Less interface leads to a less dynamic environment, which simplifies the whole process.



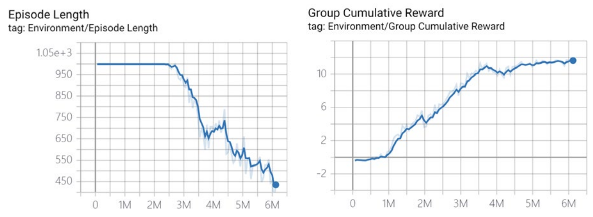
(a) (b)

Figure 5. Decentralized training result (DQN) Episode Length (a) and Cumulative Reward (b)

5.2.2 Centralized learning

Unlike decentralized learning, centralized learning could not reach the expected platform after a long training period in the first 2 million episodes (Figure 6a). The robot could not find the desired goals because the robot needed to find the desired joint observations in the same joint actions. This is because agents do share the experience, add additional sample complexity, and have significant computational burdens to set up the neural networks, which means it will take longer to initialize the simulation and find the optimal strategies. Thus, it takes longer for the robot to initialize the mapping between states and action, which takes longer for the robot to find the optimal trajectory in the beginning. As shown in Figure 6a, it takes almost 1,000 s for the first 3 million episodes to start. We can also see that the result from centralized learning is not as good as the one in decentralized learning. After a long training period, 5 million episodes (Figure 6b), the final result still does not converge. This is because some mapping between state and actions is not correctly build-up and the interface between different agents makes it harder to map the states to the actions. The change in the policies makes the environment's dynamics change, which will cause deviations in finding the optimal strategy and may take a longer time for the robot to adjust to reach its goals.

The summary of the results (metrics used for comparison purposes) for each scenario is shown in Table 3. In general, for a single-agent reinforcement learning task, it is accessible for the robot to find the optimal solution in a short period. However, as more agents come into the same environment, the variables such as states and actions lead to explosive growth, which results in significant computational burdens and sample complexities. One of the robust ways to solve this problem is to ask the robots to share their experiences and get feedback in the same reference domain. If not all information is handled and transformed, the robot could also learn, but it is pretty hard to find the optimal strategy.



(a) (b)

Figure 6. Centralized training result (DQN) Episode Length (a) and Cumulative Reward (b)

Table 3: Results for the metrics from each scenario.

Metric	DRL (DQN)	MARL (Decentralized)	MARL (Centralized)
Episode to converge	400	4e6	5e6 (Not converge)
Episode length	20 s	20 s	450 (Not converge)

6 Conclusions and outlook

This study gives an overview of RL and DRL algorithms, discusses the challenges, opportunities, and research areas of the integration of RL with construction robot applications, and implements a part of them in a simple case study to illustrate how they can be used in construction robots applications for path planning. This study shows that, as expected, for simple tasks such as navigation, single-agent RL algorithms (e.g., DQN and DDPG) effectively solve the problem, with few iterations and durations to train. However, the computational resource (training period, episode length) grows exponentially when more agents come into the same environment (more realistic for construction tasks). Using a decentralized learning algorithm is feasible to train multi-agents in those cases. The only problem is efficiency, as it takes a long training time. Still, the results are robust because this method provides a suitable evolving environment without too many changes to deal with, unlike centralized training. Therefore, it can be said that for the case study investigated, the decentralized learning approach could be better suited for complex construction task simulations.

Ongoing work by the authors includes the investigation of other RL algorithms for single task manipulation and execution, such as bricklaying, painting, door installation, etc., to prove the eligibility of such a combination of construction robot applications with RL. Our future goals include: 1) Adopting and developing current DRL and MARL algorithms for construction robot applications. 2) Adopting the multi-robot framework to solve complex tasks (e.g., laying bricks to build up a user-defined structure) through the cooperation of individual agents. 3) Assign different characters that allow different robots to work in different roles, such as manipulator, inspector, material delivery, etc. 4) Conducting the simulation in game engines such as Gazebo, Unreal, or Unity, and setting up communication among different platforms by using BIM models, this set up a great foundation for future verification in a real environment. 5) Benchmarking for efficiency of other algorithms to see which one is the best fit for the construction robot applications.

References

[1] Recht B., A tour of reinforcement learning: the view from continuous control. *Ann Rev Control Robot Autonom Syst* 2019; 6: 253–279.

[2] Mahadevan S. and Theocharous G. Optimizing production manufacturing using reinforcement learning. In: *FLAIRS Conference*, 18 May 1998, pp. 372–377. AAAI Press. 3.

[3] Silver D., Hubert T., Schrittwieser J., A general reinforcement learning algorithm that master's chess, shogi, and go through self-play. *Science 2018*; 362(6419): 1140–1144. 4.

[4] Kober J., Bagnell JA., and Peters J. Reinforcement learning in robotics: a survey. *Int J Robot Res* 2013; 32(11): 1238–1274. 5.

[5] Isele D., Rahimi R., Cosgun A., Navigating occluded intersections with autonomous vehicles using deep reinforcement learning. *ICRA*, Brisbane, QLD, Australia, 21–25 May 2018, pp. 2034–2039.

[6] Yayin X., Ying Z., Przemyslaw S., Lieyun D., Machine learning in construction: From shallow to deep learning, *Developments in the Built Environment*, Volume 6, 2021, 100045.

[7] Mnih, V., Kavukcuoglu, K., Silver, D. et al. Human-level control through deep reinforcement learning. *Nature* 518, 529–533 (2015).

[8] Shopov V., and Markova V., "A Study of the Impact of Evolutionary Strategies on Performance of Reinforcement Learning Autonomous Agents", *ICAS 2018*, p.56-60, 2018.

[9] Dresp-Langley B., Ekseth O.K., Fesl J., Gohshi S., Kurz M., Sehring H.W., Occam's Razor for Big Data? On detecting quality in large unstructured datasets. *Appl. Sci.* 2019, 9, 3065.

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

[11] Rongrong L., Florent N., Philippe Z., Michel D.M., Birgitta D., (2021). Deep Reinforcement Learning for the Control of Robotic Manipulation: A Focussed Mini-Review. *Robotics.* 10. 1-13.

[12] Zhang T., Mo H.. Reinforcement learning for robot research: A comprehensive review and open issues. *Journal of Advanced Robotic Systems.* May 2021.

[13] Gu S., Holly E., Lillicrap, T., Levine S. (2017) Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. *ICRA*, 3389–3396.

[14] Haarnoja T., Pong V., Zhou A., Dalal M., Abbeel P., Levine S., Composable deep reinforcement learning for robotic manipulation. *ICRA* (2018a).

[15] Lowe R., Wu Y., Tamar A., Multi-agent actor-critic for mixed cooperative-competitive environments. *arXiv preprint* arXiv:1706.02275, 2017.

[16] Lyu, X et al., Contrasting Centralized and Decentralized Critics in Multi-Agent Reinforcement Learning, 20th *AAMAS.* 2021

[17] Gronauer S., Multi-agent deep reinforcement learning: a survey. *Artif Intell Rev* (2021).

[18] Mnih V., Badia A. P., Mirza M., Graves A., et al., Asynchronous methods for deep reinforcement learning. *In International Conference on Machine Learning*, pages 1928–1937, 2016.

[19] Elfakharany A., Towards Multi-Robot Task Allocation and Navigation using Deep Reinforcement Learning 2020 *J. Phys.:* Conf. Ser. 1447 012045

[20] Sartoretti G., Wu Y., Paivine W., Distributed reinforcement learning for multi-robot decentralized collective construction. *Springer*, Cham, 2019: 35-49.

[21] Kober J., Bagnell J. A., and J. Peters J., 2013. Reinforcement learning in robotics: A survey. *Int. J. Rob.* Res. 32, 11, 2013, 1238–1274.

[22] Yang Y., Juntao L., Lingling P., (2020), multi-robot path planning based on a deep reinforcement learning DQN algorithm. CAAI Trans. *Intell. Technol.*, 5: 177

[23] Li B., Liang H., Multi-Robot Path Planning Method Based on Prior Knowledge and Q-learning Algorithms. 2020 *J. Phys.*: Conf. Ser. 1624 042008

[24] Chung, H. et al. Brick-by-Brick: Combinatorial Construction with Deep Reinforcement Learning, *35th NeurIPS 2021*

[25] Yu L. et al., "Multi-agent deep reinforcement learning for HVAC control in commercial buildings", *IEEE Trans. Smart Grid,* vol. 12, no. 1, 407-419, Jan. 2021.

[26] Will S., (2019). Deep Reinforcement Learning Algorithms in Intelligent Infrastructure. *Infrastructures.* 4. 52.

[27] Long P., Fan T., Liao X., Liu W., Zhang H., and Pan J., "Towards Optimally Decentralized Multirobot Collision Avoidance via Deep Reinforcement Learning," in *ICRA,* 2018.

[28] Zhou B., Khosla A., Lapedriza A., Oliva A., and Torralba A., Learning deep features for discriminative localization. In *CVPR*, pages 2921–2929, 2016.

[29] Xu X. and Garcia de Soto B., "On-site Autonomous Construction Robots: A review of Research Areas, Technologies, and Suggestions for Advancement," 37th *ISARC*, Kitakyushu, Japan, 2020.

[30] Christiano P. F., Leike J., et al.. Deep reinforcement learning from human preferences. *Advances in Neural Information Processing Systems*, pages 4302, 2017.

[31] ROS Packages for TurtleBot3 Machine Learning, *https://github.com/ROBOTIS-GIT/turtlebot3_machine_learning.git*