Joint CSCE Construction Specialty & CRC Conference 2025
Conférence conjointe spécialisée en construction de la SCGC et CRC-2025

Montreal, Quebec
July 28-31, 2025 / *28-31 juillet 2025*

Generative Adversarial Imitation Learning to Train Rebar-Tying Agents for Agent-Based Simulation

Yusheng Huang[1] and Amin Hammad[2*]

[1]Department of Building, Civil and Environmental Engineering, Concordia University, Canada
[2]Concordia Institute for Information Systems Engineering, Concordia University, Canada

Abstract: Agent-based simulation (ABS) focuses on the interactions among agents and the environment, offering detailed insights to construction management. With the as-is model of the construction site, ABSs can be performed to explore what-if scenarios. However, previous implementations of construction ABS have been hindered by the large amount of effort required for modeling the agent under the dynamic and complex nature of construction sites. Moreover, interactions between different entities are difficult to model in the simulation. Previous studies proposed methods for modeling agents using simplistic probabilistic models, leading to artificial agent behaviors and less convincing simulation results. A significant gap exists in the method of modeling construction agents capable of making realistic decisions. Generative adversarial imitation learning (GAIL) has the potential to address this limitation. It enables agents to learn task performance from demonstration datasets, establishing a mapping between observations and actions. In this paper, a novel method that incorporates GAIL for training the agent of a construction worker to perform rebar-tying tasks is proposed. A case study is conducted to validate the efficiency of the proposed method.

Keywords –
Machine Learning; Generative Adversarial Imitation Learning; Agent-Based Simulation; Productivity; Rebar-Tying.

1. INTRODUCTION

Agent-based simulation (ABS) uses fully integrated models and focuses on the interactions among agents and the environment, offering more detailed insights to construction management (Khodabandelu and Park, 2021). However, the previous implementation of construction ABS has been hindered by the large amount of effort required for modeling the agent under the dynamic and complex nature of construction sites (Abdelmegid et al., 2020). With the as-is model of construction site, ABSs can also be used to explore what-if scenarios (Madni et al., 2019). However, interactions between different entities are difficult to model in the simulation. Previous studies proposed modeling agents using simplistic probabilistic models, leading to artificial agent behaviors and less convincing simulation results. A significant gap exists in the method of modeling construction agents capable of making realistic decisions. By training agents to imitate the behavior of real workers, agent-based simulations can be used to predict task progress based on the sequence of the remaining tasks and to perform productivity analysis.

Reinforcement learning (RL) and imitation learning (IL) are trending methods for training agents (Hua et al., 2021; Hussein et al., 2017). Kim et al. (2023) has proposed a method that applies deep RL to train construction agent, which can select its path in construction site, to conduct ABS for site layout optimization. In addition, the Previous studies have also applied RL/IL algorithms focusing on training agents of robots to complete construction tasks (Apolinarska et al., 2021; Lee et al., 2022; Li and Zou, 2023; Yu et al., 2024). However, compared to the human body, robots (e.g., robotic arms) have significantly lower degrees of freedom. To address this challenge, previous studies have proposed several promising methods using IL. Peng et al. (2018) proposed a method named *DeepMimic* which trains the humanoid agent to move in the style and appearance according to given motion clips, for example, jumping, walking, back-flipping, etc. Aiming to achieve more natural movement of the agent, *DReCon* (Bergamin et al., 2019) is proposed. These methods can significantly train the agent to imitate the behavior styles from the demonstrations. However, its limitations are also obvious. The trained policies often lack generalization so the agent may not learn how to perform in a different situation (Hua et al., 2021). In addition, it is not robust enough to recover well from any errors that are overlooked in the demonstrations.

The incorporation of the generative adversarial network (GAN) with IL, which is known as generative adversarial imitation learning (GAIL) (Torabi et al., 2019), can address this limitation. The method trains the agent to learn from the dataset containing multiple demonstrations in different scenarios rather than a single demonstration. The imitation reward is calculated from a discriminator, which is trained in parallel to distinguish the trajectories from the learned policy against the demonstrated trajectories. GAIL has been proven to give excellent results for complex and high-dimensional physics-based control tasks (Ho and Ermon, 2016). By incorporating GAIL, based on their previous IL work of *DeepMimic*, Peng et al. (2021) proposed a new method to train the agent to finish a task while imitating the behavior styles according to the locomotion datasets. Environment interaction features have been added to their latest work (Hassan et al., 2023), which allow agents to finish the scene interaction tasks without manual annotation.

The paper introduces a novel method that integrates GAIL framework of the previous studies from Peng et al. (2021) and Hassan et al. (2023) to train a construction worker agents for conducting agent-based simulations focusing on rebar-tying tasks. A case study is conducted to validate the efficiency of the proposed method.

2. PROPOSED METHOD

This section proposes a method to train construction worker agents using GAIL to support ABSs for construction management. Figure 1 shows the workflow of training a rebar-tying agent.

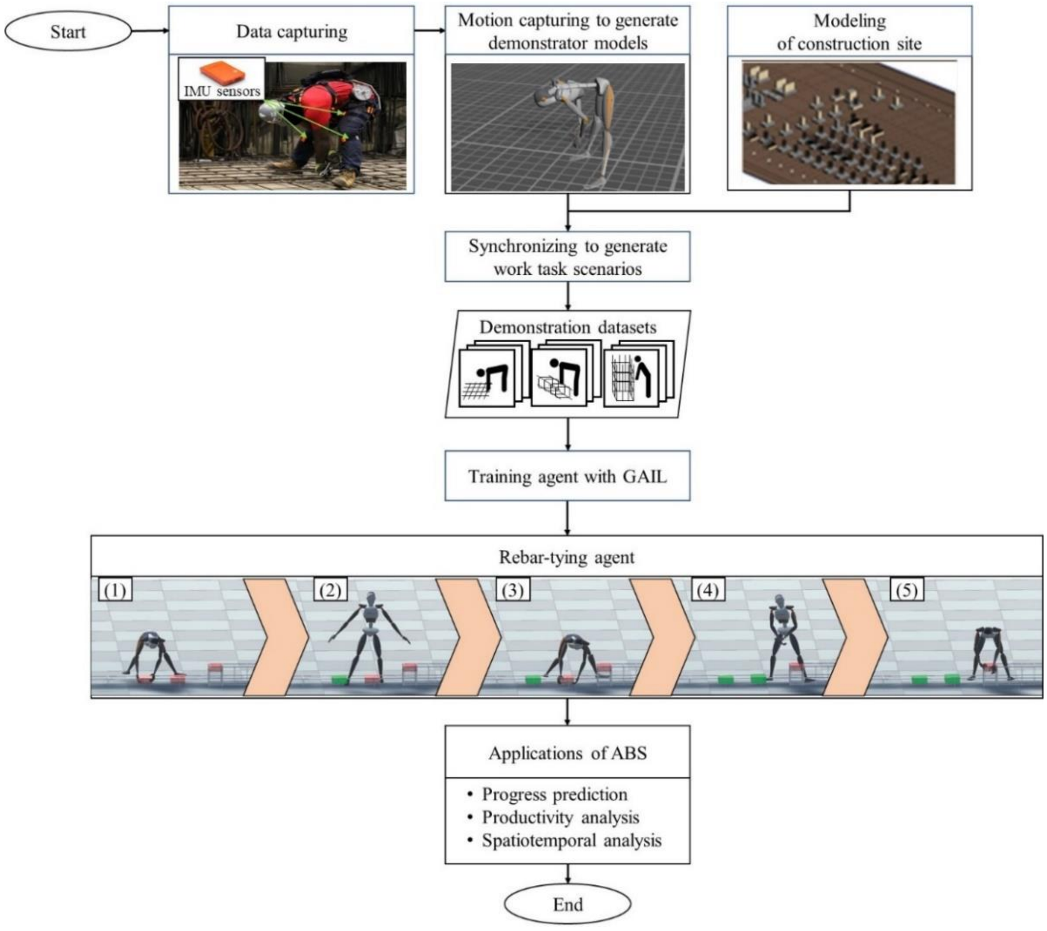

Figure 1: Workflow of the proposed method

Data from inertial measurement units (IMUs) is collected to capture the rebar-tying worker's motion to generate the demonstrator models for training. On the other hand, the simulation environment is updated to as-built by updating the site geometry and information about the upcoming tasks and schedules. The demonstrator models and the construction site model are synchronized to replicate work task scenarios when the worker moves to a position near the work area to complete the rebar-tying task, which may vary in height. These scenes are saved in the demonstration dataset. Then, GAIL algorithm is applied to train rebar-tying agents using the demonstrations as references. The trained agent should be able to perform rebar-tying tasks in the virtual environment. When the next task is identified, the agent moves to the appropriate position near the new work area and begins the task. The agent adjusts its posture according to the task's height while maintaining the style of behavior observed in the demonstration dataset. An example in Figure 1 shows how an agent ties rebars at three adjacent work areas with different heights. At last, with the trained agent, agent-based simulation can be performed to assist construction management based on the sequence of remaining tasks of the as-built construction site model for progress prediction and productivity analysis purposes. Additionally, this approach facilitates the identification of spatiotemporal issues in the construction site. Consequently, project managers can make decisions and necessary adjustments to improve productivity.

2.1 GAIL to Train Construction Worker Agents

Figure 2 shows the GAIL system for training agents which is developed by adapting the GAIL workflow proposed by Peng, et al. (2021) and Hassan et al. (2023). This system comprises three primary components: the agent and the environment and the discriminator.

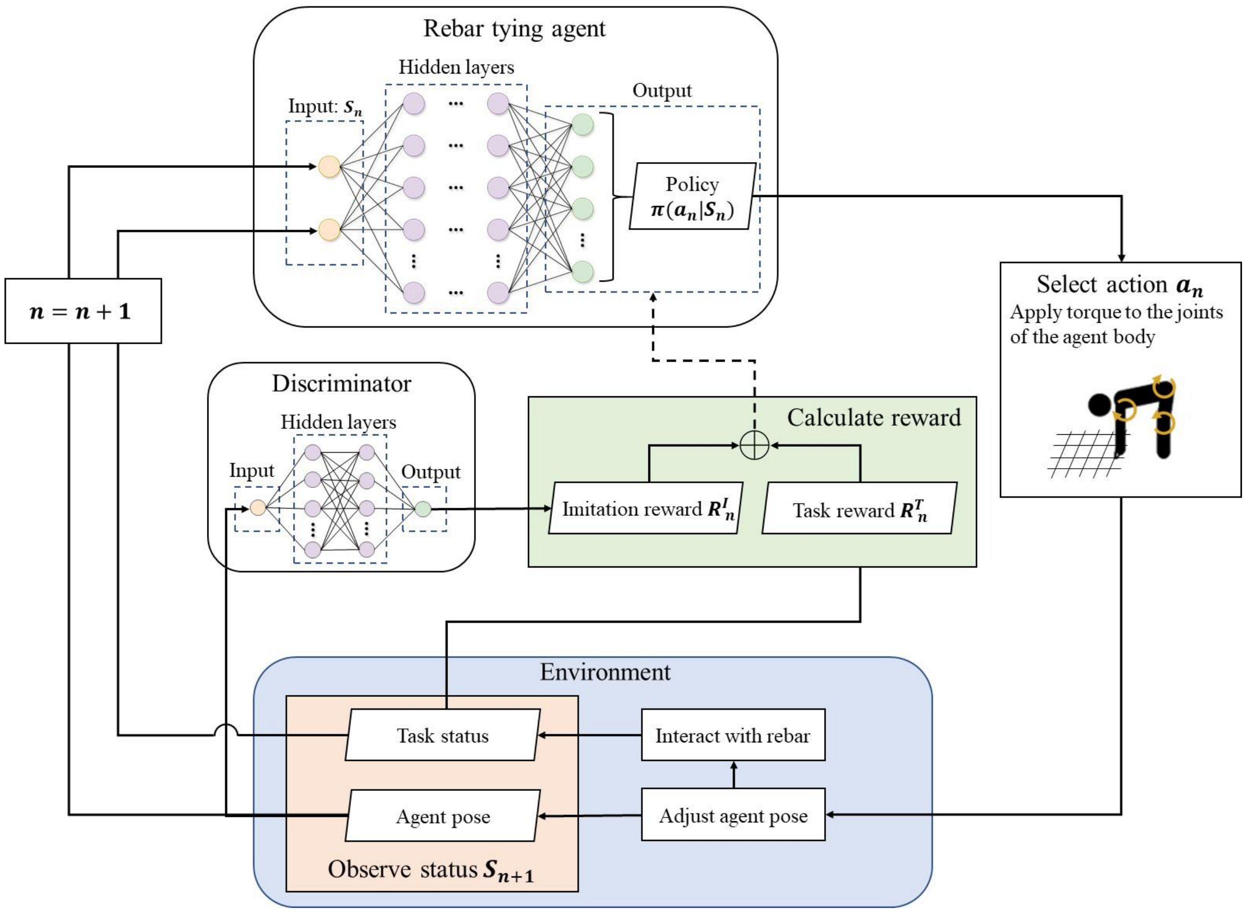

Figure 2: GAIL system for training rebar tying agents

The agent contains a deep neural network (DNN) to articulate the policy ($\pi_i(a_n|s_n)$) for selecting actions a_n within a given state S_n. In the context of training a rebar worker agent, a_n denotes the torque applied to the agent's body joints.

The simulation environment serves as the platform for agent interaction. Upon executing action a_n within the environment, the agent's pose updates to facilitate interaction with the rebars, consequently transitioning the environment to a new state S_{n+1}. The new observables in S_{n+1}, encompassing the agent's pose and task status, are then fed into the agent's DNN to select the subsequent action a_{n+1} within the state S_{n+1}. The observables in each state is further explained in Section 2.2.

These observables are used for calculating the reward from action a_n. Rewards consist of two parts: (a) the task reward, signifying the agent's task performance; and (b) the imitation reward, reflecting the fidelity of the agent's motion. The imitation reward is calculated by a discriminator, which uses a DNN to distinguish between the behaviors generated by the agent and the behavior recorded in the demonstration dataset. A detailed explanation of reward functions is provided in Section 2.3. These rewards are fed back to the agent to evaluate the state-action pair (S_n, a_n). The agent's DNN parameters are adjusted accordingly to optimize performance.

During the training process, the agent and the discriminator are trained parallelly. The framework for performing training with the GAIL system is illustrated in Figure 3. At the beginning of the training, the random-initialized agent and discriminator are used in the GAIL system to conduct simulations. The simulation generates multiple state-action-reward pairs (S_n, a_n, R_n), which are saved in a buffer. Whenever the size of the buffer reaches the maximum buffer size, the DNN parameters of both the agent and the discriminator will be updated to optimize the action selection policy and the performance of distinguishing the generated motion against the demonstrations, respectively. Then, a new round of simulation is conducted in the GAIL system with the updated agent and discriminator.

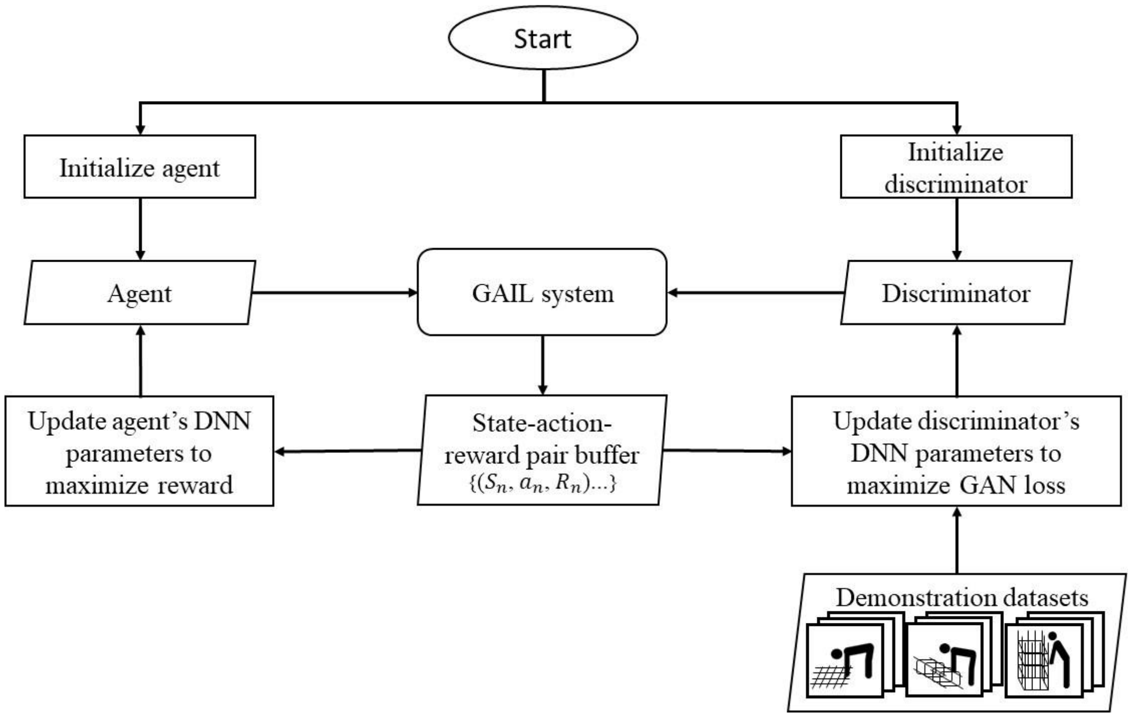
Start
Initialize agent
Initialize discriminator
Agent
GAIL system
Discriminator
Update agent's DNN parameters to maximize reward
State-action-reward pair buffer $\{(S_n, a_n, R_n)\ldots\}$
Update discriminator's DNN parameters to maximize GAN loss
Demonstration datasets

Figure 3: Training framework

2.2 Status Observation

The observations from the environment in each state consider two aspects: (a) the observables related to the task status, and (b) the observables related to the agent's motion features. The observables are listed in Table 1. Various observables pertaining to task status are collected, including the task's relative position, progress level, and whether it is actively being worked on (i.e., progression). The relative position refers to the task's location relative to the root joint of the agent. The task progress level is observed on a scale of 0 to 100%, denoting no progress and completion, respectively. Finally, task progression is observed by checking whether the agent's hands are interacting with the task's work area.

In terms of the agent's motion features, the focus lies on the status of each body part's movement. These features serve as criteria for the discriminator to evaluate the similarity between the agent's motion and the demonstrated behaviors at corresponding states. The feature observation is done by tracking the status of each joint, as shown in Figure 4.

Table 1: Observations in each state

Task status	Motion features
- The relative position of the task	- Velocity and angular velocity of the root joint of the avatar in the world coordinate system
- Task progress	- Relative rotation and angular velocity of other joints with respect to their parent joint
- Whether the agent is currently working on the task	- Position of the hands and feet relative to the root joint

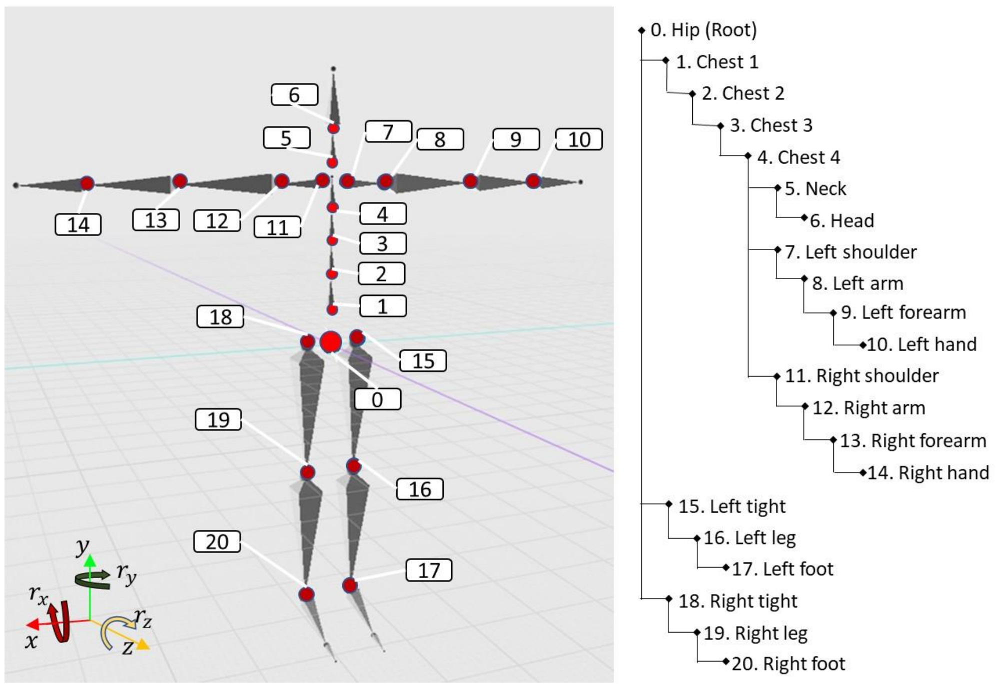

Figure 4: Hierarchy of joints

For the root joint, which corresponds to the hip center point in this study, the velocity and angular velocity are observed. These features serve to represent the overall motion status of the human body, allowing the discriminator to identify any abnormal motion speed of the agent behavior compared to the demonstrations in the dataset. For other joints, relative rotation and angular velocity in relation to their parent joint according to the hierarchy, are observed to capture the pose and motion of the agent's body parts. Furthermore, the positions of hands and feet are monitored to ensure that the combination of joint rotations by the agent results in hands and feet placements similar to the demonstrated behaviors in the dataset.

2.3 Reward Functions

In designing the reward functions, two aspects are taken into account: (a) the reward for finishing the task objectives (R_n^T), and (b) the reward for locomotion imitation (R_n^I), as shown in Eq. (1) adapted from (Peng et al., 2021).

$$R_n = \omega^T R_n^T + \omega^I R_n^I \tag{1}$$

where ω^T and ω^I are the weights for the reward related to the two aspects. It is worth noting that the sum of ω^T and ω^I is 1.

The task reward for the rebar-tying task is design to incentivize the agent's behavior to interact with the environment and finish the rebar-tying task. The task is considered to be in progress when both hands are in the work area. Eq. (2) shows the reward functions for the task considering two scenarios: a reward of 1 is given if the task is completed, while an unfinished task yields a reward based on the location of the work area and the pose of the agent.

$$R_n^T = \begin{cases} 1, & \text{task finished} \\ w^r r_n^r + w^h r_n^h, & \text{not finished} \end{cases} \tag{2}$$

where r_n^r refers to the reward when the agent is getting close to the task zone (Eq. (3)), r_n^h is the reward for the agent to put its hands near the task zone (Eq. (4)), w^r and w^h are the weights assigned to r_n^r and r_n^h, respectively.

$$r_n^r = \frac{\exp(-a_r \|x_n^* - x_n^{root}\|^2)}{2} \tag{3}$$

$$r_n^h = \sum_{hand=1}^{2} \frac{\exp(-a_h \|x_n^* - x_n^{hand}\|^2)}{2} \tag{4}$$

where x_n^*, x_n^{root} and x_n^{hand} are the position vectors of the work area, root joint and the hands in step *n* in the world coordinate system, respectively. a_r and a_h are the coefficients that are determined according to the expected reward ($\hat{r}_L$) at a specific distance baseline (*L*) as shown in Eq. (5). For example, in this rebar-tying task, assuming the distance from the root to the work area is 3 m, $\hat{r}_D$ of 0.1 is assigned. For the example of hands, assuming both hands are at a distance of 0.5 m from the work area, $\hat{r}_L$ of 0.1 is assigned.

$$a = -\frac{\ln(\hat{r}_L)}{L^2} \tag{5}$$

The reward of behavior imitation (R_n^I) is calculated by using a state-only discriminator. The discriminator D takes a transition from state s_n to s_{n+1} as input and outputs the probability that this state transition is derived from the demonstrations. A high probability infers that the discriminator cannot differentiate the behavior generated by the agent (fake data) from the behaviors recorded in the demonstration dataset (real data). As mentioned in Section 2.1, during the training of the agent, the discriminator is also concurrently trained to distinguish between the real data and fake data, by using the objective function (Eq. (6)) which maximizes the GAN loss (Eq. (7)) according to Torabi et al. (Torabi et al., 2019).

$$\mathrm{D} = argmax_{\mathrm{D}}(\text{GAN loss}) \tag{6}$$

$$\text{GAN loss} = \mathbb{E}_M\left[\log\left(\mathrm{D}(S_n^M, S_{n+1}^M)\right)\right] + \mathbb{E}_\pi\left[\log\left(1 - \mathrm{D}([S_n^\pi, S_{n+1}^\pi])\right)\right] \tag{7}$$

where S^Mrefers to the state recorded in the demonstration dataset M, S^π refers to the state recorded in the trajectories generated by the policy π. $\mathbb{E}_M()$ and $\mathbb{E}_\pi()$ refers to the functions that compute the expectations over the demonstration dataset and the trajectories generated from policy the policy π, respectively.

Conversely, to improve the policy to the level that the agent behavior can fool the discriminator, the agent will need to minimize the GAN loss. While the trained policy has no impact on the states in the demonstration, it could only generate behaviors that minimize the second part of Eq. (7). As such, the imitation reward can be defined as shown in Eq. (8).

$$R_n^I = D(s_n, s_{n+1}) \tag{8}$$

3. IMPLEMENTATION AND CASE STUDY

To validate the feasibility of the proposed method, a case study is conducted to generate a rebar-tying construction agent, which is capable to finish the interacting tasks with rebars in the construction environment.

3.1 Collecting Demonstration Dataset

In this test, data were collected within a controlled laboratory environment, where a worker undertook rebar-tying exercises in three distinct scenarios: (a) tying rebars on the slab; (b) tying rebars for the formwork; (c) tying rebars for a component in the prefabrication factory. The three scenarios may induce varying poses due to the difference in the rebar heights. These three situations yield separate datasets capturing rebar tying at varying heights, introducing differences in the pose of the worker as shown in Figure 5. The human demonstrator is equipped with IMUs from Xsense (2024), which will subsequently generate a digital model of the demonstrator. For each motion clip, the demonstrator is asked to stand in a regular pose and then bend down to finish the rebar tying tasks.

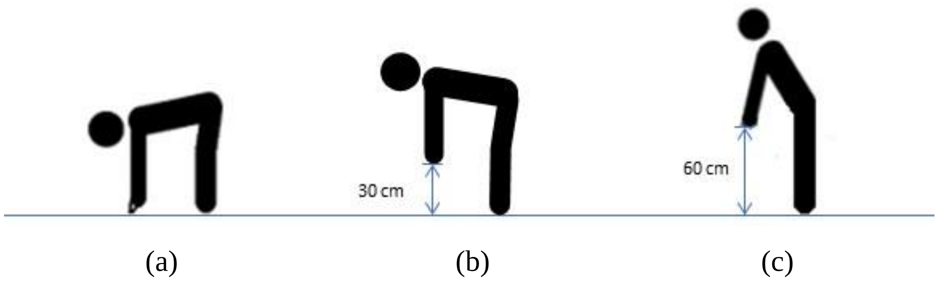

Figure 5: Rebar-tying scenarios with different poses

3.2 GAIL System Development

An interactive simulation environment was developed using Unity3D (2025), featuring the Movella Avatar. The environment includes a task zone reset function at the start of each simulation episode. In this training system, the agent is asked to finish the rebar-tying tasks under the replicated virtual scenarios where the demonstrator was asked to perform rebar-tying tasks. The rebar-tying task is considered to be in progress when both of the agent’s hands are in the rebar-tying work area.

The developing of the GAIL system uses the open-source package Modular Agent (2025), which provides proportional-derivative controller to control the agent based on MuJoCo simulation (2025). To train the agent, the trainer type of proximal policy optimization (PPO) is selected.

The discriminator is developed using Pytorch (2025). The GAN has two fully connected layers. The stochastic gradient descent optimizer with momentum of 0.9 is select. A threading user datagram protocol (UDP) server is developed to enable the communication between the Unity environment and the discriminator in Python.

3.3 Results

The policy is trained under 10 parallel simulation environments with the hyperparameters as shown in Table 2 on 16 CPU cores computer with two GTX 1070 GPU. The network setting and the hyperparameter selection for the agent in this preliminary test were primarily based on prior experience (Peng et al., 2021) and the guidelines provided in the ML-Agents documentation (2025) for PPO. For the discriminator, parameters, such as the number of epochs, batch size, and buffer size were aligned with those chosen for the agent. Since the training process of the discriminator tends to be more stable than that of the agent, a higher learning rate can be used. Additionally, a higher learning rate makes the discriminator more capable than the agent to ensure it could effectively distinguish between real and generated data, so that the agent will be forced to imitate the demonstration to get higher imitation reward.

Figure 6 shows the preliminary results with snapshots of the sequence of motions of the agent from the trained policy (the blue avatar) for Scenario (b) in Figure 5, compared with the demonstration (the yellow avatar). The unfinished task area is colored in red and the finished task is colored in green. The figure shows that the agent is almost synchronized with the demonstrator when it is getting down to finish the task. However, upon completing the task, the agent exhibits a slower pace in getting up compared to the demonstrator.

Table 2: Hyperparameters of the DNN

	Agent	Discriminator
Number of Epochs	3	3
Beta	2×10^{-4}	N.A.
Epsilon	0.1	N.A.
Discount factor	0.995	0
Learning rate	5×10^{-5}	1×10^{-3}
Batch size	2048	2048
Buffer size	40960	40960
Network setting		
-Number of Layer	3	2
-Hidden units	(512, 512, 512)	(1024, 512)

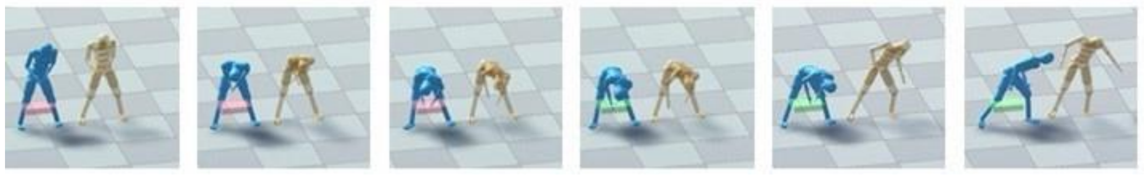

Figure 6: The sequence of agent motions from the trained policy compared with the demonstration

4. CONCLUSIONS AND FUTURE WORK

This paper proposed a novel method that incorporates the GAIL framework to train rebar-tying worker agents for interaction tasks. The trained agents can complete construction tasks while maintaining the behavioral style observed in the demonstration dataset. A case study is conducted using a prototype GAIL system developed using Unity3D game engine and Python. The results demonstrate that the method can effectively train agents to learn task performance from demonstration datasets, allowing them to perform construction tasks while preserving the behavior style of the demonstrator.

Several challenges need to be addressed in the future. Currently, training an agent to complete a simple task, such as tying rebars, takes a long time. To address this, future work will include sensitivity analysis to optimize hyperparameters, aiming to improve both training efficiency and agent performance. In addition, the task used in this study is relatively simple. Future research will focus on training construction agents to perform a variety of more complex, interactive tasks in a more challenging environment. For example, the agent should be able to deal with rebar-tying at different heights. Furthermore, future research will focus on developing methods to deploy the trained agent for agent-based construction simulation, enabling progress prediction and productivity analysis for rebar-tying tasks.

REFERENCES

Abdelmegid, M. A., González, V. A., Poshdar, M., O'Sullivan, M., Walker, C. G., & Ying, F. (2020). Barriers to adopting simulation modelling in construction industry. Automation in Construction, 111, 103046. https://doi.org/10.1016/j.autcon.2019.103046

Apolinarska, A. A., Pacher, M., Li, H., Cote, N., Pastrana, R., Gramazio, F., & Kohler, M. (2021). Robotic assembly of timber joints using reinforcement learning. Automation in Construction, 125, 103569. https://doi.org/10.1016/j.autcon.2021.103569

Balint-H/modular-agents: Extensions to the ML-Agents toolkit, focusing on humanoid control in Unity. (n.d.). Retrieved January 19, 2025, from https://github.com/Balint-H/modular-agents

Bergamin, K., Clavet, S., Holden, D., & Forbes, J. R. (2019). DReCon: Data-driven responsive control of physics-based characters. ACM Trans. Graph., 38(6), 206:1-206:11. https://doi.org/10.1145/3355089.3356536

Hassan, M., Guo, Y., Wang, T., Black, M., Fidler, S., & Peng, X. B. (2023). Synthesizing Physical Character-Scene Interactions (arXiv:2302.00883). arXiv. https://doi.org/10.48550/arXiv.2302.00883

Ho, J., & Ermon, S. (2016). Generative Adversarial Imitation Learning. Advances in Neural Information Processing Systems, 29.

Hua, J., Zeng, L., Li, G., & Ju, Z. (2021). Learning for a Robot: Deep Reinforcement Learning, Imitation Learning, Transfer Learning. Sensors, 21(4), Article 4. https://doi.org/10.3390/s21041278

Hussein, A., Gaber, M. M., Elyan, E., & Jayne, C. (2017). Imitation Learning: A Survey of Learning Methods. ACM Computing Surveys, 50(2), 21:1-21:35. https://doi.org/10.1145/3054912

Khodabandelu, A., & Park, J. (2021). Agent-based modeling and simulation in construction. Automation in Construction, 131, 103882. https://doi.org/10.1016/j.autcon.2021.103882

Kim, M., Ham, Y., Koo, C., & Kim, T. W. (2023). Simulating travel paths of construction site workers via deep reinforcement learning considering their spatial cognition and wayfinding behavior. Automation in Construction, 147, 104715. https://doi.org/10.1016/j.autcon.2022.104715

Lee, D., Lee, S., Masoud, N., Krishnan, M. S., & Li, V. C. (2022). Digital twin-driven deep reinforcement learning for adaptive task allocation in robotic construction. Advanced Engineering Informatics, 53. Scopus. https://doi.org/10.1016/j.aei.2022.101710

Li, R., & Zou, Z. (2023). Enhancing construction robot learning for collaborative and long-horizon tasks using generative adversarial imitation learning. Advanced Engineering Informatics, 58, 102140. https://doi.org/10.1016/j.aei.2023.102140

Madni, A. M., Madni, C. C., & Lucero, S. D. (2019). Leveraging Digital Twin Technology in Model-Based Systems Engineering. Systems, 7(1), Article 1. https://doi.org/10.3390/systems7010007

Motion Capture | Xsens. (n.d.). Retrieved January 19, 2025, from https://www.movella.com/

Overview—MuJoCo Documentation. (n.d.). Retrieved January 21, 2025, from https://mujoco.readthedocs.io/en/stable/overview.html

Peng, X. B., Abbeel, P., Levine, S., & van de Panne, M. (2018). DeepMimic: Example-guided deep reinforcement learning of physics-based character skills. ACM Transactions on Graphics, 37(4), 143:1-143:14. https://doi.org/10.1145/3197517.3201311

Peng, X. B., Ma, Z., Abbeel, P., Levine, S., & Kanazawa, A. (2021). AMP: Adversarial motion priors for stylized physics-based character control. ACM Transactions on Graphics, 40(4), 144:1-144:20. https://doi.org/10.1145/3450626.3459670

PyTorch. (n.d.). PyTorch. Retrieved January 19, 2025, from https://pytorch.org/

Technologies, U. (n.d.). Unity Real-Time Development Platform | 3D, 2D VR & AR Engine. Retrieved February 15, 2022, from https://unity.com/

Torabi, F., Warnell, G., & Stone, P. (2019). Generative Adversarial Imitation from Observation (arXiv:1807.06158). arXiv. https://doi.org/10.48550/arXiv.1807.06158

Unity ML-Agents Toolkit. (2024). https://unity-technologies.github.io/ml-agents/

Yu, H., Kamat, V. R., & Menassa, C. C. (2024). Cloud-Based Hierarchical Imitation Learning for Scalable Transfer of Construction Skills from Human Workers to Assisting Robots. *Journal of Computing in Civil Engineering*, *38*(4), 04024019. https://doi.org/10.1061/JCCEE5.CPENG-5731