Automatic Generation of Inspection Routes for Multiple Building Facades with Reinforcement Learning using Drones

Takumi Shibata[1] and Satoshi Yamada[2]
[1]*Nipponkoei, Business Strategy Operations, Tokyo, Japan*
takumi.shibata8195@gmail.com, ORCID: 0009-0009-1073-1134
[2]*Ritsumeikan University.*
[2]*sy@fc.ritsumei.ac.jp, ORCID: 0009-0009-3653-7196*

Abstract. Drones are being increasingly used for infrastructure and building inspections. However, significant challenges remain, such as the need for human interventions to determine inspection routes. Route-generation approaches using rule-based algorithms often fail to address the complexities of large-scale scenarios, such as inspecting buildings across an entire city. In response, this paper proposes a route-generation strategy for building inspections using deep reinforcement learning. In a game engine, we created a simplified virtual environment comprising sixty box-shaped structures representing buildings. This approach aimed to elucidate the conditions and design considerations necessary for implementing an effective reinforcement-learning model to inspect all building façades. Unity's ML-Agents Toolkit was employed for reinforcement learning. Drone movements and wall layout were mapped onto a 40-cell grid to simplify the environment and facilitate model training. After 1.8 billion learning steps, the model enabled drones to inspect all designated building surfaces, achieving an average flight path of 18,392 m across simulations, which is comparable to the route-generation performance of traditional rule-based methods. This study demonstrates the potential of reinforcement learning to tackle complex tasks, such as citywide building inspections. It serves as a pioneering example of agent-based research, highlighting its significance in encouraging further studies in this field.

Keywords. Drone inspection, path planning, wall surface inspection, reinforcement learning

1. Introduction

1.1. SOCIAL CONTEXT

The infrastructure in Japan is aging rapidly, necessitating regular inspections. However, challenges such as shortages of skilled engineers and financial resources persist. Additionally, as Japan is prone to earthquakes, quick inspection of buildings and infrastructure following disasters is critical. Thus, routine and emergency inspections of infrastructure are becoming increasingly necessary. Drone-based

ARCHITECTURAL INFORMATICS, Proceedings of the 30th International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA) 2025, Volume 3, 39-48.

inspection has been introduced to address this issue. However, in most cases, human operations are required, which pose ongoing challenges relating to safety, cost, and efficiency. Drone-based inspection has been introduced as a solution to address this issue. However, in many cases, human intervention is still required, which introduces persistent challenges in terms of safety, cost, and efficiency. Additionally, planning an inspection route for infrastructure over a broad area proves to be an enormous and complex task. Automated inspections face further challenges due to the difficulty in determining a unique inspection route. The sheer number of potential routes makes this problem particularly complex. Even when attempting to determine a route that visits n locations, a vast number of possible paths must be considered. Therefore, research has been conducted to derive routes using methods such as approximate solutions (e.g., greedy algorithms, local search methods) and exact solutions. However, numerous conditions must be considered for drone flights, including building shapes, the presence of obstacles, site conditions, the characteristics of individual drones, and weather conditions such as wind and rain. Developing an algorithm that can adapt to all these changing factors is impractical. Hence, this study focuses on reinforcement learning, which has the capability to learn autonomously according to various conditions. Reinforcement learning allows an agent to explore data and learn independently, enabling it to gain rich representations through exposure to diverse patterns. Additionally, inference models trained using reinforcement learning offer high generalization performance, allowing them to adapt to a wide range of conditions.

1.2. RESEARCH OBJECTIVE

This study investigated the generation of inspection routes using reinforcement learning. This study focused on large-scale and complex scenarios, such as the inspection of building exteriors during disasters, routine infrastructure inspections, and citywide building assessments. The aim was to create a flight path for drones equipped with cameras, and simulations were conducted within a game engine environment. As previously mentioned, route planning is complex owing to the vast number of possible routes. Therefore, a reinforcement learning approach, where an agent autonomously explores and learns, was adopted to generate inspection routes.

1.3. PREVIOUS RESEARCH

During wall inspection using drones, path planning traditionally relies on rule-based approaches. For instance, Xu et al. (2024), Zhang et al. (2024) and Zhou et al. (2021) demonstrated that a rule-based approach using heterogeneous drones facilitates efficient inspection of complex structures, such as Marina Bay Sands. However, when considering the inspection of building surfaces across an entire city, rule-based path planning approaches encounter limitations. Recent studies have investigated path planning using deep reinforcement learning to address these challenges (Muñoz, 2019; Tylkin, 2022). However, applications specifically focusing on wall inspections using reinforcement learning have not yet been reported. Therefore, this study aimed to explore the potential of reinforcement learning for inspections by focussing on the

following two objectives and novel contributions:

- Assess the feasibility of inspection path planning using reinforcement learning.
- If feasible, clarify the optimal conditions and design parameters required during training.

2. Research Overview

The workflow of the analytical method is illustrated in Figure 1. A reinforcement learning environment was developed using Anaconda and the ML-Agents Toolkit from Unity. After setting up the environment, parameters and other conditions in Unity were adjusted to follow the reinforcement-learning process. Flight simulations were conducted using Unity to evaluate the performance of the trained inference model.

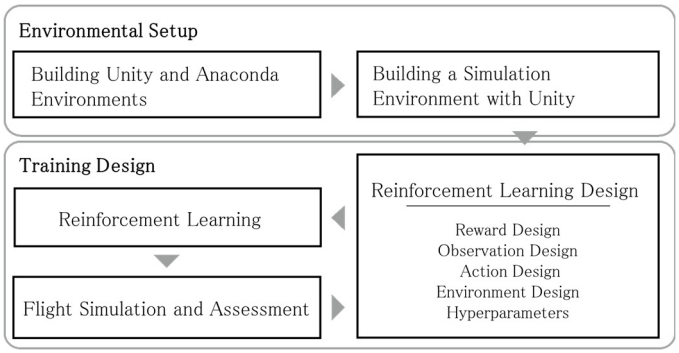

Figure 1. Flowchart of analytical method

2.1. RESEARCH OBJECTIVE

Reinforcement learning (Sutton, 1998) is a machine-learning method that enables learning of the optimal policy to take the best possible action based on the state of the environment and receive rewards. Figure 2 presents an overview of the reinforcement-learning process, which involves an iterative cycle of the following steps:

(1) The agent acquires the current state at each time step t.

(2) The agent selects an action based on the acquired state.

(3) A reward is assigned based on the performed action.

This iterative process continues as the agent seeks to optimise its long-term reward.

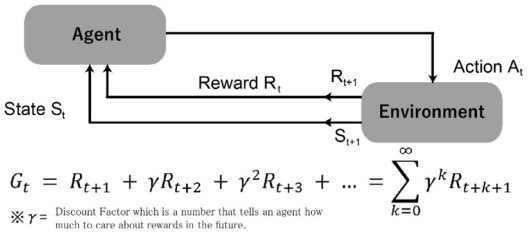

Figure 2. Learning overview

2.2. TRAINING ENVIRONMENT

An overview of the constructed training environment and the system versions used are shown in Figure 3. The design of states, actions, and rewards, as well as training and result verification, were conducted using Unity (version 2020.3.48f1). Reinforcement learning was implemented using the machine learning toolkit of Unity, Unity Machine Learning Agents (ML-Agents v2.0.1) (Juliani, 2018). The ML-Agents toolkit was built using the PyTorch library and Python 3.8.13. Training was performed using a graphics processing unit (GPU). The computer used in this study had a Core i7 8700 K processor, an NVIDIA GeForce RTX 2070 SUPER GPU, and a 32 GB DDR4 RAM. The off-policy proximal policy optimisation (PPO) method (Schulman, 2017) was employed for the reinforcement-learning algorithm. Although PPO has a low sample efficiency and requires numerous training steps, it is stable during training.

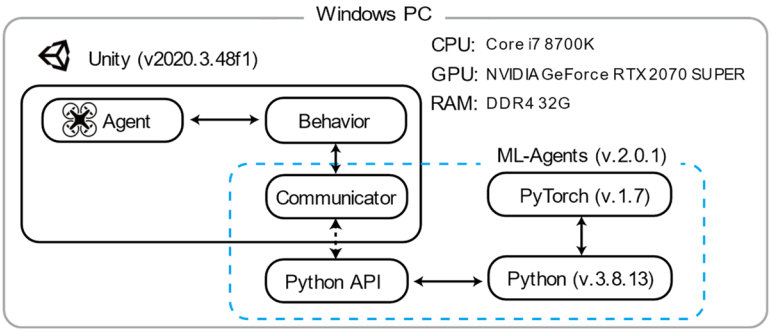

Figure 3. Overview of environmental setup

2.3. WALL INSPECTION ENVIRONMENT

A drone-based wall-inspection environment was created using the Unity game engine. The movement of the drone and wall structures were simplified to streamline the training. The drone was represented as a cube and programmed to move in six directions: up, down, left, right, forward, and backward, with rotational capabilities to turn left and right. The wall was modelled as a single surface composed of 40 cells (Figure 4), with each cell measuring w = 1 m × h = 2 m. Subsequently, the surface of the entire wall was $w = 10$ m × $h = 8$ m, whereas the entire volume of the structure was $w = 10$ m × $d = 10$ m × $h = 8$ m. This structure was positioned in a three-dimensional space for training.

Wall inspection was deemed successful when the following conditions were satisfied: (1) the drone faced the wall cell directly, and (2) the distance between the drone and the cell was within a specific threshold. These criteria were established based on advancements in camera technology, as visual inspections can be effectively performed if a camera captures a clear image of the wall surface. In the simulation environment, colours of the inspected wall sections were changed to indicate completion, and the flight path of the drone was displayed in purple (Figure 4).

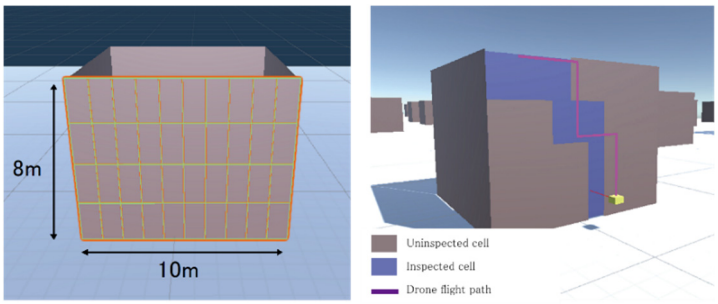

Figure 4. Left: Building model in Unity Right: Overview of flight simulation

3. Design Method

Reinforcement learning was applied to a group of 60 buildings arranged in a 340 m × 390 m field (Figures 6 and 7). Adjustments were made to the rewards, observations, environmental settings, and hyperparameters to ensure that all wall inspections were completed. The details of the design parameters are presented in Table 1.

Table 1. Detailed design for reinforcement learning （※distance: Refers to the distance between the previously and currently inspected cell, with a minimum value of 1.0. ※Normalise the dot product, as shown in Figure 10）

Observation	Drone's xyz coordination	3 observations
	Drone's y-axis rotation angle	1 observations
	Distance and vector between the drone and the cell	3 observations
	RayCast Observation	15 rays
Reward	When a horizontal wall is detected	$+1/(10^2 \times distance^{※})$
	When other walls are detected	$+1/(10^3 \times distance^{※})$
	When all walls have been detected	$+1.0$
	Out of range / per step	$-10^{-3} \cdot -10^{-6}$
	When moving toward a distant target cell	$(\vec{a} \cdot \vec{b})_{norm}^{※} \times 10^{-6}$

3.1. OBSERVATION DESIGN

In the observation design (information accessible to the drone for training), the drone gathers data regarding its own status and that of the wall. The includes recording its position coordinates and rotation angles. The position of the wall was detected using the Raycast component in Unity. When a ray (light beam) emitted from the drone contacted a wall, its distance to the wall was recorded. Additionally, the drone acquired the distance and vector of the target cell. The cell to be observed was defined as the cell closest to the last inspected position of the drone (Figure 5), and the target cell for

observation was determined through training using the following three methods:

- Method A: The cell closest to the current position of the drone is observed at each time step.
- Method B: The cell closest to the last inspected cell is observed until inspection.
- Method C: The cell closest to the position of the drone is observed after each cell inspection.

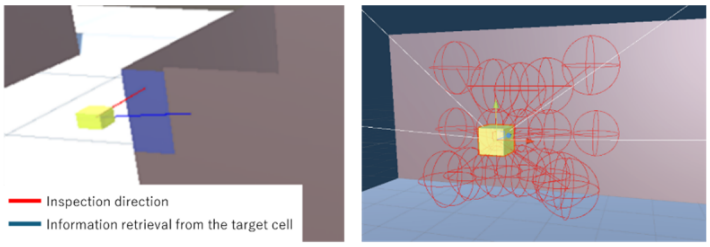

Figure 5. Method for obtaining wall information

Among these methods, Method C yielded the best results; therefore, this approach was adopted to determine the target cell for observation.

3.2. REWARD DESIGN

Positive rewards were given when a cell was inspected and upon the completion of all wall inspections. Negative rewards were assigned when the drone moved outside the designated area or over time at regular intervals. In addition, rewards were structured to encourage efficient inspection by increasing the number of cells adjacent to or close to previously inspected cells. This reward structure, based on previous trials, was designed to improve the learning efficiency and optimise the inspection paths. Figure 9 shows the conditions and amounts of reward.

The episode concluded either when all wall inspections were completed or when the drone moved out of bounds, and the drone was reset to its initial position. The flight area and starting point are shown in Figure 6. Because the drone did not naturally proceed to the next building after completing an inspection, rewards were designed to encourage movement toward subsequent buildings. The amount of reward was adjusted based on the dot product of the vector representing the movement of the drone (a) and vector from the drone to the target cell (b) (Figure 7). Consequently, rewards increased when the drone moved directly toward the target cell.

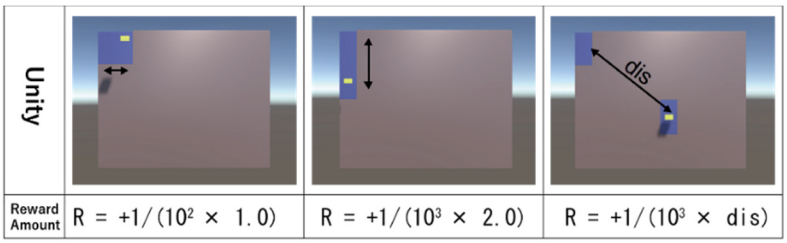

Figure 6. Amount of reward: Left: large; Centre: medium; Right: small

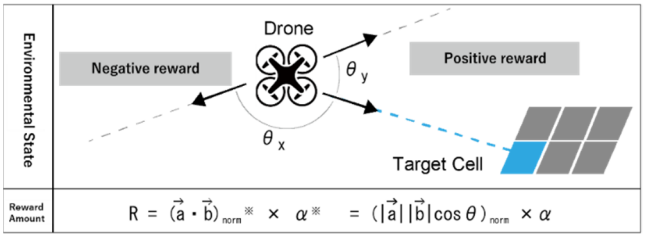

Figure 7. Reward based on the dot product (※ The dot product is normalised, and a constant α (10-6) is applied to adjust the amount of reward)

3.3. HYPERPARAMETER DESIGN

The details of the hyperparameters used in reinforcement learning are listed in Table 2. The beta parameter (entropy, which controls the randomness of action selection) was set to 1/100. When the beta value was set lower (beta = 1/5000), learning stalled, showing no progress.

Table 2. Hyperparameters set during the training process

Learning Hyperparameter
bach_size : 128
buffe_size : 2048
learning_rate : 0.0003
beta : 0.01
epsilon : 0.2
lambd : 0.95
num_epoch
learning_rate_schedule : liner

Network_settings
normalize : false
hidden_units : 256
num_layers : 2
vis_encode_type : simple
Reward_signals
gamma : 0.99
strength : 1.0

3.4. INSPECTION RESTRICTIONS PER BUILDING

The system was designed to prevent the inspection of the next building until all wall inspections of the current building were completed. Immediately after the inspection of the building was completed, the drone was permitted to inspect the nearest adjacent building. In the initial designs, where the drone could move to the next building without completing inspections, the learning process stalled as the drone engaged in repetitive movements between buildings rather than focused inspections. Thus, these restrictions were implemented to maintain efficient learning progress.

4. Route Generation for Inspecting 60 Buildings

4.1. CURRICULUM LEARNING

Curriculum learning was used in this study. This approach gradually increases the task difficulty to facilitate the acquisition of basic behaviours, eventually enabling the agent to tackle more complex tasks. In this study, the task difficulty was incrementally

increased by increasing the number of buildings from three walls to 1, 2, 3, 10, 20, 30, and 60 buildings. The training process consisted of 1.8 billion steps. Figure 8 shows the environmental settings, coverage area, and training steps for each stage.

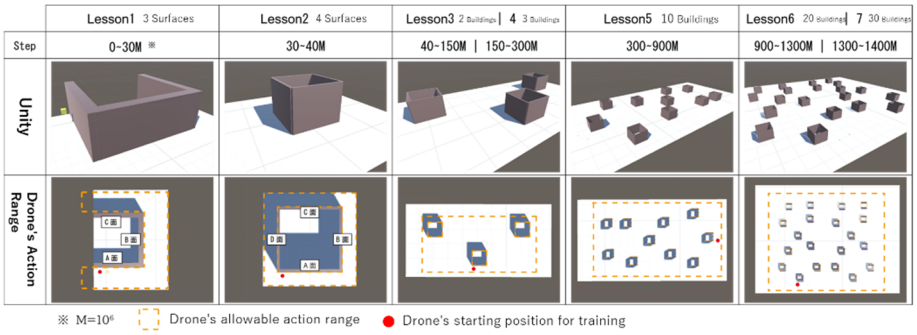

Figure 8. Curriculum-learning process

4.2. TRAINING RESULTS

Figure 9 and Figure 10 show the progressions of the average cumulative rewards, episode lengths, and average loss of the value function during the training process. One of the flight paths of the drone during training is shown in Figure 11. The drone successfully completed inspections of all the building walls. The average flight distance required to complete the inspections was 18,392.1 m, based on 10 simulation flights.

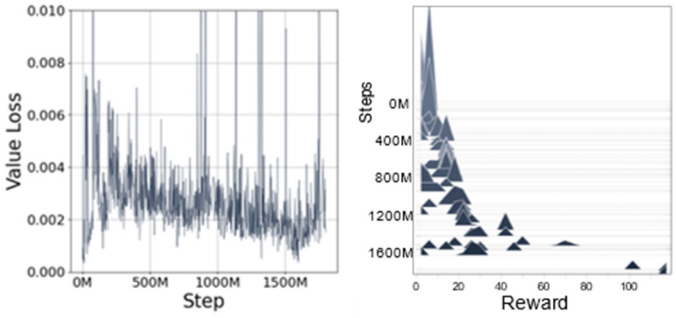

Figure 9. Progression of value loss and steps of average cumulative rewards

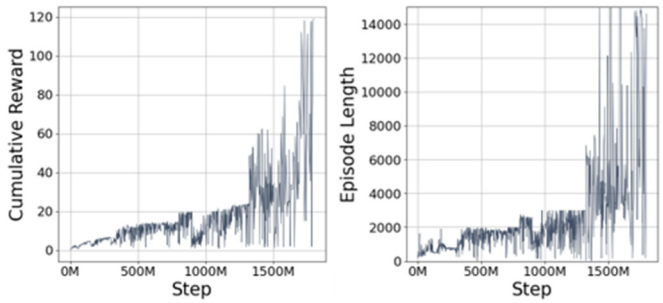

Figure 10. Progression of cumulative rewards and episode length of steps

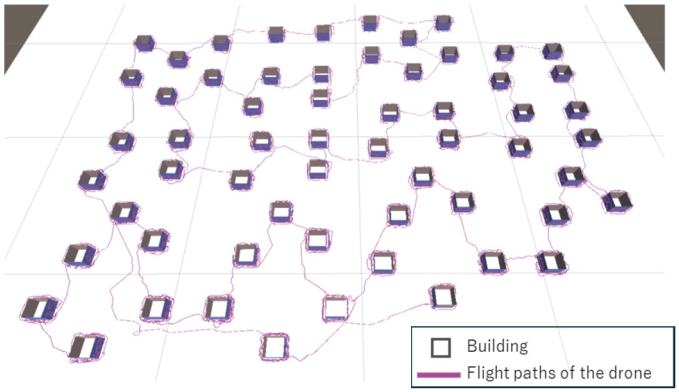

Figure 11. Inspection path of drone after training

4.3. ROUTE GENERATION USING A COMBINATION OF SIMULATED ANNEALING

For additional evaluation, route generation was conducted by combining reinforcement learning with simulated annealing, a search algorithm used to solve global optimisation problems. An approximate solution for the shortest inspection route among buildings was derived using simulated annealing. After completing the inspection of one building, the next building was selected based on the optimised route. A previously trained inference model for inspecting 60 buildings was applied without modification.

Consequently, inspections of all 60 buildings were completed following the optimised route. The average flight distance over 10 simulation flights was 18,444.8 m. Compared to using only reinforcement learning, this approach resulted in a slight increase in flight distance, with a ratio of 1.002865 (Figure 12). However, challenges still exist, such as the suboptimal inspection route distance after training and lengthy training time required to inspect subsequent buildings.

	A : Reinforcement Learning only	B : Method combining with Simulated Annealing
Average	18392.11	18444.81
Shortest distance	17758.23	17132.19
Longest distance	18932.02	19263.37

Comparison based on the ratio of RL only and Simulated Annealing Learning only		
Average value ratio	(B ÷ A)	1.002865
Shortest distance ratio	(B ÷ A)	0.964746
Longest distance ratio	(B ÷ A)	1.017502

Figure 12. Comparison of path and flight distance using simulated annealing

5. Conclusion

In this study, a method for generating drone-based wall inspection routes was evaluated using reinforcement learning. The results demonstrated that all wall surfaces of the 60 buildings could be successfully inspected through reinforcement learning. Moreover, the results demonstrated that reinforcement learning can be used to perform inspections on a scale comparable to that of past rule-based methods. This finding provides an effective solution for automating large-scale building inspections, potentially catalysing further research on agent-based models. Future studies should focus on enhancing reinforcement learning for more complex inspection targets, designing a learning environment to improve efficiency, and developing methods to reduce training time.

References

Juliani, A., Berges, V.-P., Teng, E., Cohen, A., Harper, J., Elion, C., Goy, C., Gao, Y., Henry, H., Mattar, M., & Lange, D. (2018). Unity: A general platform for intelligent agents, arXiv:1809.02627. https://arxiv.org/abs/1809.02627

Muñoz, G., Barrado, C., Çetin, E., & Salami, E. (2019). Deep reinforcement learning for drone delivery. Drones, 3(3), 72. https://doi.org/10.3390/drones3030072.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms, arXiv:1707.06347. https://arxiv.org/abs/1707.06347.

Sutton, R. S., & Barto, A. G. (1998). Reinforcement learning: An introduction. MIT Press, Cambridge, UK.

Tylkin, P., Wang, T.-H., Palko, K., Allen, R., Siu, H. C., Wrafter, D., Seyde, T., Amini, A., & Rus, D. (2022). Interpretable autonomous flight via compact visualizable neural circuit policies. IEEE Robotics and Automation Letters, 7(2), 3265–3272. https://doi.org/10.1109/LRA.2022.3146555.

Xu, X., Cao, M., Yuan, S., Nguyen, T. H., Nguyen, T. -M., & Xie, L. (2024). A cost-effective cooperative exploration and inspection strategy for heterogeneous aerial system, In 18th International Conference on Control & Automation (ICCA) (pp. 673–678). IEEE, Reykjavík, Iceland.

Zhang, M., Feng, C., Li, Z., Zheng, G., Luo, Y., Wang, Z., Zhou, J., Shen, S., & Zhou, B. (2024). SOAR: Simultaneous exploration and photographing with heterogeneous UAVs for fast autonomous reconstruction. arXiv preprint arXiv:2409.02738. https://arxiv.org/abs/2409.02738

Zhou, B., Zhang, Y., Chen, X., & Shen, S. (2021). FUEL: Fast UAV exploration using incremental frontier structure and hierarchical planning. IEEE Robotics and Automation Letters, 6(2), 779–786. https://doi.org/10.1109/LRA.2021.3051563.