Application of Reinforcement Learning for UAV Trajectory Planning in Complex Terrain

IOTA5201 (L01) - RL for Intelligent Decision Making in Cyber-Physical Systems

Shuaijun Liu
Student ID: 50047151
The Hong Kong University of Science and Technology (Guangzhou)
GitHub: https://github.com/Shuaijun-LIU/Final_for_5201

***Abstract*—This report presents my work on applying reinforcement learning (RL) for 3D trajectory planning of Unmanned Aerial Vehicles (UAVs) in complex terrain environments. Building upon a previous research project that successfully applied Double Deep Q-Network (DDQN) for UAV attitude control through a Fuzzy-Enhanced Adaptive Reinforcement PID (FEAR-PID) controller, I extend the RL framework to trajectory planning using Twin Delayed Deep Deterministic Policy Gradient (TD3). The work is conducted within a UAV-assisted fog computing system framework, where the UAV must navigate from User1 to User6 while avoiding obstacles including buildings, trees, lakes, and terrain height variations. I design an RL environment with an 8-dimensional state space capturing position, relative distance to goal, and terrain height information, along with a 3-dimensional continuous action space for 3D displacement control. I implement TD3 training infrastructure with comprehensive logging and visualization capabilities, and analyze the challenges encountered in applying RL to complex 3D navigation tasks. While I successfully implement the framework and training infrastructure, I acknowledge that I have not achieved satisfactory training results, with the model failing to learn effective obstacle avoidance strategies. This report discusses the methodology, implementation details, challenges identified, and potential improvements for future work.**

I. Introduction

Unmanned Aerial Vehicles (UAVs) have been identified as versatile platforms that connect IoT devices and servers via the network edge [3], with some UAVs equipped with computational capabilities serving as "moving fog nodes" for offloading computational tasks. To fully utilize UAVs in fog computing, key considerations include attitude control, trajectory planning, and task assignment.

I consider a fog computing environment with a single UAV deployed at the network edge, aiming to maximize energy efficiency through collaborative control of attitudes, trajectory planning, resource allocation, and task assignment. Attitude control ensures stable UAV orientation [6], while trajectory planning reduces power consumption by identifying efficient paths [2]. Task assignment determines whether tasks should be handled locally, processed in the fog layer, or offloaded to the cloud [7].

Existing studies have applied reinforcement learning (RL) to UAV control and trajectory planning. Deep RL approaches such as TD3 have been used for trajectory planning [9], while RL has been integrated with PID controllers for attitude control [8], [12].

In my previous research project, I successfully applied RL to UAV attitude control using a Fuzzy-Enhanced Adaptive Reinforcement PID (FEAR-PID) controller[1]. This approach integrates Double Deep Q-Network (DDQN) with fuzzy logic to dynamically optimize PID gain parameters. The success of DDQN in attitude control (handling discrete action spaces) motivated me to extend RL to trajectory planning. However, trajectory planning requires continuous 3D displacement vectors, necessitating algorithms like TD3 designed for continuous action spaces.

In this work, I apply TD3 to 3D obstacle-avoiding trajectory planning, extending the RL framework from attitude control (DDQN) to trajectory planning (TD3). I design an RL environment with appropriate state and action spaces, implement TD3 training infrastructure, and identify challenges in applying RL to complex 3D terrain environments. While I successfully implemented the framework, I acknowledge that I did not achieve satisfactory training results, with the model failing to learn effective obstacle avoidance strategies. This report discusses the methodology, implementation details, challenges identified, and potential improvements.

II. System Model and Framework

A. Network Structure and Components

I consider a UAV-assisted fog computing system in a three-dimensional Euclidean space. The system consists of a quadrotor UAV, a remote data center (DC) in the cloud, and K mobile IoT devices (MDs). A demonstration of the key structures in the system is shown in Figure 1. The quadrotor UAV is equipped with computational and communication capabilities, serving as a mobile fog node that can offload computational tasks from IoT devices. The K mobile IoT devices are randomly distributed on the terrain surface according to a Poisson Point Process (PPP), with the coordinate of the jth MD denoted by $\boldsymbol{M}_j = [x_j, y_j, z_j]$. Each MD is capable of generating computational tasks that require processing. The remote data center provides additional computational resources

[1]A simple game-like UAV flight control demo in my environment is available at https://shuaijun-liu.github.io/Final_for_5201/

in the cloud, enabling tasks to be offloaded for execution when local or fog-layer processing is insufficient.

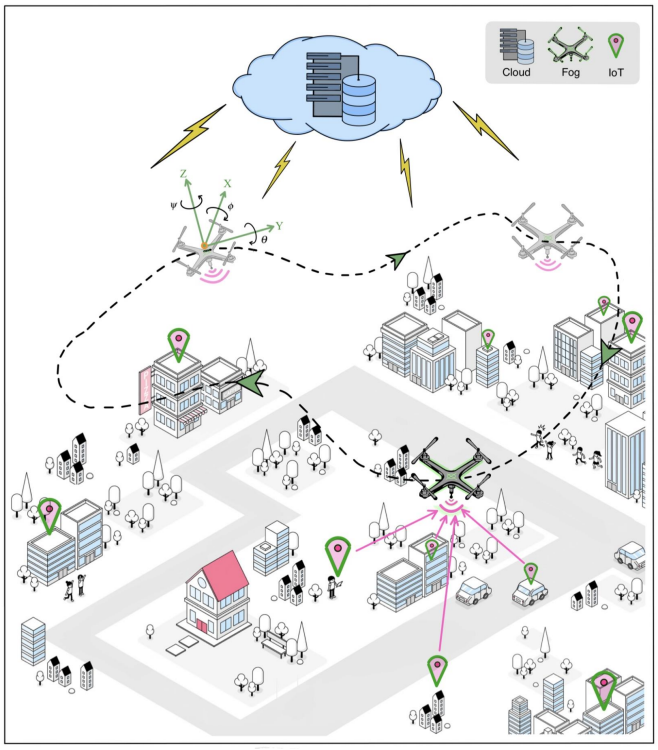

Fig. 1: The structure of UAV-assisted fog computing network, consisting of a quadrotor UAV, K mobile IoT devices (MDs), and a remote data center (DC).

The system operates in a three-dimensional space with dimensions $S \times S \times Z$, where terrain height varies continuously. I discretize a continuous time horizon with length T uniformly into N timeslots, where each timeslot has a length of $L = \frac{T}{N}$. I assume that N is sufficiently large, or equivalently, the timeslots are sufficiently short, such that the position of the UAV can be considered to be fixed within each timeslot. I use $[x(t), y(t), z(t)]$, where $t \in \{1, 2, \cdots, N\}$, to denote the UAV position at the tth timeslot. The UAV is powered by a battery with a maximum capacity of B_c. The height and speed are restricted to not exceed $z^{\max} = 150$ m and $v^{\max}$, respectively, at all times.

B. Holistic Optimization Framework

The UAV-assisted fog computing system employs a holistic optimization framework with four integrated modules that are inherently linked and mutually dependent. The first module is attitude control, which uses a Fuzzy-Enhanced Adaptive Reinforcement PID (FEAR-PID) controller that integrates DDQN with fuzzy logic to optimize PID parameters for stable flight. By adjusting pitch, roll, and yaw, attitude control ensures a stable and precise orientation of the UAV during operation, which is necessary to maintain high-quality communication with IoT devices and other fog nodes, while also facilitating effective computation and storage tasks as a fog node.

TABLE I: KEY NOTATIONS

Notation	Definition
s_t	State vector at time t
a_t	Action vector at time t
(x_t, y_t, z_t)	UAV position in 3D space
(dx_t, dy_t, dz_t)	Relative vector to goal
d_t	Euclidean distance to goal
h_{terrain}	Terrain height at (x, z)
$\Delta x, \Delta y, \Delta z$	Displacement components
Δp	Actual displacement vector
λ_a	Action scale factor
h_s	Safety margin
$R(s_t, a_t)$	Reward function
R_{GOAL}	Goal reward
$R_{\text{COLLISION}}$	Collision penalty
R_{STEP}	Step penalty
R_{SMOOTH}	Smoothness penalty
γ	Discount factor
τ	Soft update coefficient
K	Number of mobile IoT devices
S, Z	Spatial dimensions
T	Time horizon length
N	Number of timeslots
L	Timeslot length
B_c	Battery capacity
$z^{\max}$	Maximum altitude
$v^{\max}$	Maximum speed
α, β	Reward weighting factors

The second module is trajectory planning, which is the focus of this work. I use TD3 to plan obstacle-avoiding paths that minimize energy consumption while navigating through complex 3D terrain. The third module is task assignment, which uses Particle Swarm Optimization (PSO) to determine whether each computational task should be executed locally on the IoT device, processed in the fog layer by the UAV, or offloaded to the remote data center. The fourth module is resource allocation, which also employs PSO to efficiently allocate communication and computational resources, including transmission power and processing frequency, among different tasks and devices.

These modules are interdependent: attitude control affects flight stability and energy consumption, which in turn influences the efficiency of trajectory planning. Trajectory planning determines movement paths and efficiency, affecting how quickly the UAV can reach IoT devices to collect tasks. Both attitude control and trajectory planning interact with task assignment and resource allocation decisions, as the energy consumption and latency for the UAV to reach the proximity of an IoT device initiating a task along a planned trajectory must be considered when deciding task offloading and resource allocation. This holistic approach enables the system to optimize overall performance by jointly considering all four aspects rather than optimizing each module independently.

C. Existing RL Application: FEAR-PID for Attitude Control

In my previous work, I developed a Fuzzy-Enhanced Adaptive Reinforcement PID (FEAR-PID) controller[2] that integrates reinforcement learning with fuzzy logic to enhance the adaptive capabilities of conventional PID control. The architecture is shown in Figure 2. The system uses Double Deep Q-Network (DDQN) to optimize PID gain parameters based on real-time feedback, demonstrating the successful application of RL in UAV attitude control.

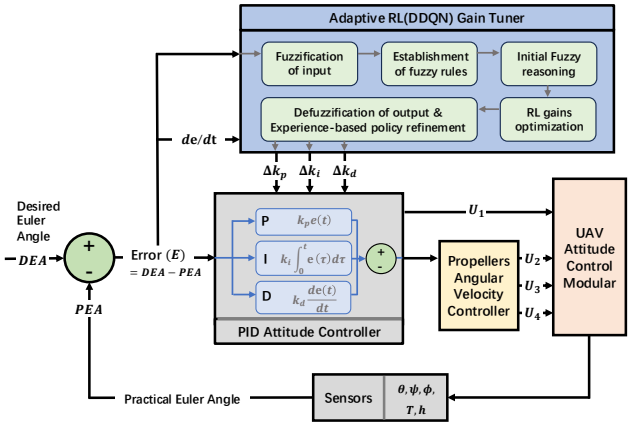

Fig. 2: Schematic structure of FEAR-PID control system for UAV attitude control. The Double Deep Q-Network (DDQN) module optimizes PID gain coefficients based on real-time feedback, demonstrating the successful application of RL in UAV attitude control.

The FEAR-PID controller design incorporates fuzzy logic to provide initial expert knowledge for PID parameter adjustment, while DDQN learns to refine these parameters adaptively based on control performance. The state space $S_t = [E, EC, \Delta k_p, \Delta k_i, \Delta k_d]$ consists of the control error E, error rate EC, and current PID parameter adjustments Δk_p, Δk_i, and Δk_d. The action space $A_t = [\Delta k_p, \Delta k_i, \Delta k_d]$ represents discrete adjustments to the PID parameters, resulting in updated PID gains $k_{p-FEAR} = k_p + \Delta k_p$, $k_{i-FEAR} = k_i + \Delta k_i$, and $k_{d-FEAR} = k_d + \Delta k_d$. The reward function is designed to minimize control error and overshoot while maintaining stability, guiding the RL agent toward optimal PID parameter configurations.

DDQN is well-suited for this application because it efficiently handles discrete action spaces, making it ideal for parameter optimization problems where PID gains can be adjusted in discrete increments. The fuzzy logic component provides initial expert knowledge through preset fuzzy rules, while DDQN learns to refine and adapt these rules based on real-time control performance, enabling the system to handle dynamic environmental changes and maintain stability during takeoff, cruising, and landing phases.

[2]A simple game-like UAV flight control demo in my environment is available at https://shuaijun-liu.github.io/Final_for_5201/

D. Trajectory Planning Problem Formulation

The trajectory planning problem requires finding a feasible 3D path from User1 (the starting position) to User6 (the goal position) while satisfying multiple constraints and optimizing performance objectives. The constraints include terrain clearance requirements, where the UAV must maintain a minimum clearance of 8.0 m above the terrain height at any given position. I also require the system to avoid collisions with various obstacles, including buildings modeled as axis-aligned bounding boxes, trees represented as cylindrical obstacles, and lakes defined as elliptical regions. Physical limits constrain the maximum altitude to 150 m and the maximum episode length to 800 steps, while boundary constraints ensure that the UAV position remains within the defined map boundaries at all times.

The primary objective is to minimize energy consumption while ensuring safe navigation through complex 3D terrain. The problem complexity stems from several factors: the continuous nature of both state and action spaces in 3D, requiring fine-grained control over position and movement; the presence of multiple constraint types (terrain height variations, diverse obstacle geometries, and boundary limits) that must be simultaneously satisfied; the need to consider both horizontal navigation (avoiding obstacles laterally) and vertical navigation (maintaining appropriate altitude above terrain) in a coordinated manner; and the sparse nature of reward signals, where positive rewards are only received upon reaching the goal and negative rewards are given upon collision, making it challenging for learning algorithms to gather informative feedback during exploration.

III. TD3-based Trajectory Planning Methodology

A. From Discrete to Continuous RL: Why TD3?

While DDQN successfully handles discrete action spaces in attitude control, trajectory planning requires continuous control over 3D displacement vectors. This fundamental difference necessitates a different RL algorithm. The comparison between DDQN and TD3 highlights their complementary roles: DDQN handles discrete actions (PID parameter adjustments) and is suitable for parameter optimization problems, while TD3 handles continuous actions (3D displacement vectors) and is suitable for path planning tasks. Both algorithms complement each other, forming a complete RL-driven control framework that covers both discrete parameter optimization and continuous trajectory planning.

TD3 (Twin Delayed Deep Deterministic Policy Gradient) [10] is chosen for several key advantages that make it particularly suitable for continuous control tasks like trajectory planning. First, TD3 provides direct continuous action support, outputting 3D displacement vectors without the need for action space discretization, which would introduce quantization errors and limit fine-grained control. Second, TD3 employs a deterministic policy, which is suitable for deterministic environments where consistent and reproducible paths are desired, enabling stable trajectory planning behavior. Third, TD3

addresses the overestimation bias problem common in actor-critic methods by using twin Q-networks that independently estimate action values and taking the minimum of the two estimates during policy updates, thereby improving training stability. Fourth, TD3 implements delayed policy updates, updating the policy network less frequently than the Q-networks, which prevents value function divergence and contributes to more stable learning. Finally, TD3 incorporates target policy smoothing, adding noise to target actions during value function updates to reduce value function estimation errors and further improve training stability. These characteristics make TD3 particularly suitable for continuous control tasks like trajectory planning, where smooth and consistent actions are essential for safe and efficient navigation.

B. Reinforcement Learning Framework

I formulate the trajectory planning problem as a Markov Decision Process (MDP), defining the state space, action space, state transition model, and reward function. The state space $\mathcal{S}$ is an 8-dimensional vector that captures essential information for navigation:

$$s_t = [x_t, y_t, z_t, dx_t, dy_t, dz_t, d_t, h_{\text{terrain}}] \tag{1}$$

where (x_t, y_t, z_t) represents the current UAV position in 3D space, (dx_t, dy_t, dz_t) is the relative vector from the current position to the goal position, d_t is the Euclidean distance to the goal, and h_{terrain} is the terrain height at the current (x, z) position, which is crucial for maintaining safe altitude above the terrain.

The action space $\mathcal{A}$ is a 3-dimensional continuous vector representing the desired displacement in each spatial dimension:

$$a_t = [\Delta x, \Delta y, \Delta z] \in [-1, 1]^3 \tag{2}$$

The action values are normalized to the range $[-1, 1]$ and then scaled by $\lambda_a = 10.0$ m to obtain the actual displacement in meters:

$$\Delta p = a_t \times \lambda_a \tag{3}$$

This normalization and scaling approach allows the policy network to output actions in a standardized range while enabling meaningful movement distances in the environment.

The state transition follows a deterministic physics model, where the next position is computed by adding the scaled action to the current position:

$$(x_{t+1}, y_{t+1}, z_{t+1}) = (x_t, y_t, z_t) + a_t \times \lambda_a \tag{4}$$

with the altitude y_{t+1} clamped to the range $[0, 150]$ m to ensure it remains within physical limits.

The reward function is designed with multiple objectives to guide the agent toward safe and efficient navigation:

$$R(s_t, a_t) = \begin{cases} +500 & d_t < 5.0 \text{ m} \\ -500 & \text{collision} \\ -1.0 & \text{per step} \\ -0.1 \times ||a_t||_2 & \text{smoothness} \end{cases} \tag{5}$$

The reward structure provides a large positive reward ($+500$) when the agent reaches within 5.0 m of the goal, a large negative reward (-500) upon collision to strongly discourage unsafe behavior, a small per-step penalty (-1.0) to encourage efficient paths, and a smoothness penalty ($-0.1 \times ||a_t||_2$) based on the L2 norm of the action to promote smoother trajectories. The reward function balances multiple objectives including goal achievement, safety, efficiency, and trajectory smoothness.

C. Environment Design

I implement the environment using Gymnasium (formerly OpenAI Gym) for compatibility with Stable-Baselines3 [15], providing a standardized interface for RL algorithm integration. The environment incorporates several key components to accurately model the 3D navigation task. The height field component loads terrain height data from `terrain_data.json` and provides interpolated height queries for any (x, z) position, enabling the agent to query terrain elevation at arbitrary locations for collision checking and safe altitude determination.

The collision detection system implements comprehensive collision checking for multiple obstacle types. For terrain collisions, the system checks if the UAV altitude y falls below the terrain height plus the safety margin ($y < h_{\text{terrain}} + h_s$), where $h_s = 8.0$ m is the safety margin, ensuring the UAV maintains adequate clearance above the terrain surface. Building collisions are detected using Axis-Aligned Bounding Box (AABB) collision detection, which efficiently checks whether the UAV position intersects with any building's bounding volume. Tree collisions employ cylindrical collision detection based on each tree's radius and height, treating trees as vertical cylinders in 3D space. Lake collisions use elliptical collision detection to check if the UAV position falls within elliptical boundaries defined for each lake. Additionally, boundary checking ensures that the UAV position remains within the defined map limits, preventing navigation outside the valid operational area.

The environment implements three termination conditions that determine when an episode ends. Success termination occurs when the distance to the goal falls below 5.0 m, indicating that the UAV has successfully reached the target location. Failure termination is triggered when any collision is detected, including terrain, building, tree, lake, or boundary violations, immediately ending the episode to reflect unsafe navigation. Timeout termination occurs when the episode length exceeds MAX_STEPS = 800, preventing episodes from continuing indefinitely and encouraging efficient path planning. Upon any termination condition, the environment resets to the initial state, allowing the agent to begin a new episode and continue learning.

D. TD3 Algorithm Configuration

I use Stable-Baselines3's TD3 implementation [15] with hyperparameters configured for the trajectory planning task. The hyperparameter configuration is summarized in Table II. The learning rate is set to 10^{-3} for both the actor (policy) and critic (Q-value) networks, providing a balance between learning speed and stability. The batch size of 128 is used for experience replay, sampling transitions from the replay buffer to update the networks. The training frequency is set to 1 step, meaning the networks are updated after every environment step, and gradient steps per update is set to 1, indicating a single gradient descent step per update cycle.

TABLE II: TD3 Hyperparameters

Parameter	Value
Learning Rate	10^{-3}
Batch Size	128
Train Frequency	1 step
Gradient Steps	1 per update
Policy Delay	2
Target Policy Noise	0.2
Action Noise	0.1
Discount Factor (γ)	0.99
Soft Update Coefficient (τ)	0.005

The policy delay parameter is set to 2, meaning the policy network is updated once every two Q-network updates, implementing the delayed policy update mechanism characteristic of TD3. The target policy noise is set to 0.2, controlling the amount of noise added to target actions during value function updates to smooth the target policy. The action noise is set to 0.1 for exploration during training, adding Gaussian noise to actions sampled from the policy to encourage exploration of the action space. The discount factor $\gamma = 0.99$ determines the importance of future rewards relative to immediate rewards, with values close to 1 indicating that long-term planning is important. The soft update coefficient $\tau = 0.005$ controls the rate at which target networks are updated toward the main networks, using a moving average approach for stable target estimation. The policy and value networks use Multi-Layer Perceptrons (MLPs) with default architectures from Stable-Baselines3, which typically consist of two hidden layers with 256 units each and ReLU activation functions.

IV. Experimental Setup

A. Environment Configuration

I configure the experimental environment to evaluate TD3's performance in 3D trajectory planning within a complex terrain setting. The key configuration parameters are summarized in Table III. The start point is set to User1 (index 0), and the goal point is set to User6 (index 5), establishing a challenging navigation task across the terrain. The safety margin is configured at 8.0 m, ensuring the UAV maintains adequate clearance above the terrain surface. The maximum altitude is constrained to 150.0 m, representing a reasonable operating ceiling for UAV flight. The action scale is set to 10.0 m, determining the maximum displacement per step in the environment. The maximum episode length is set to 800 steps, providing sufficient time for the agent to navigate from start to goal while preventing excessively long episodes. The observation dimension is 8, corresponding to the state space vector defined in the Reinforcement Learning Framework subsection, and the action dimension is 3, representing the 3D displacement vector.

TABLE III: Environment Configuration

Parameter	Value
Start Point	User1 (index 0)
Goal Point	User6 (index 5)
Safety Margin	8.0 m
Max Altitude	150.0 m
Action Scale	10.0 m
Max Episode Steps	800
Observation Dimension	8
Action Dimension	3

The terrain environment is loaded from `terrain_data.json`, which provides a comprehensive representation of the 3D navigation space. The terrain data includes a height map with 2 m resolution, enabling precise terrain height queries at any location. The environment contains 1,500 buildings modeled with Axis-Aligned Bounding Box (AABB) collision boxes, creating a dense urban-like obstacle distribution. Additionally, 4,000 trees are represented with cylindrical collision volumes, each defined by radius and height parameters. Multiple lakes are included with elliptical boundaries, adding further complexity to the navigation challenge. The dataset also includes 6 user positions (User1 through User6), representing potential start and goal locations for trajectory planning tasks. A visualization of this complex terrain environment is shown in Figure 3, illustrating the distribution of buildings, trees, lakes, and user positions, highlighting the significant challenges for trajectory planning due to complex obstacle distribution and terrain height variations. The combination of multiple obstacle types and terrain height variations creates a highly challenging navigation environment.

B. Training Configuration

I train the TD3 agent for 20,000 timesteps using the hyperparameter configuration detailed in Table II. The training process logs key performance metrics including episode rewards, episode lengths, success/failure indicators, and collision statistics by type. All training data is automatically saved to `outputs/training_log.json` for post-analysis and comparison with future training runs.

V. Experimental Results and Challenges

A. Training Outcomes

Unfortunately, the training results were not satisfactory. The TD3 agent failed to learn effective obstacle avoidance strategies, as summarized in Table IV. Over the course of 20,000 training timesteps (corresponding to 20,000 episodes due to immediate terminations), the agent achieved a success rate of 0%, meaning it never successfully navigated from the start position to the goal position. The average episode

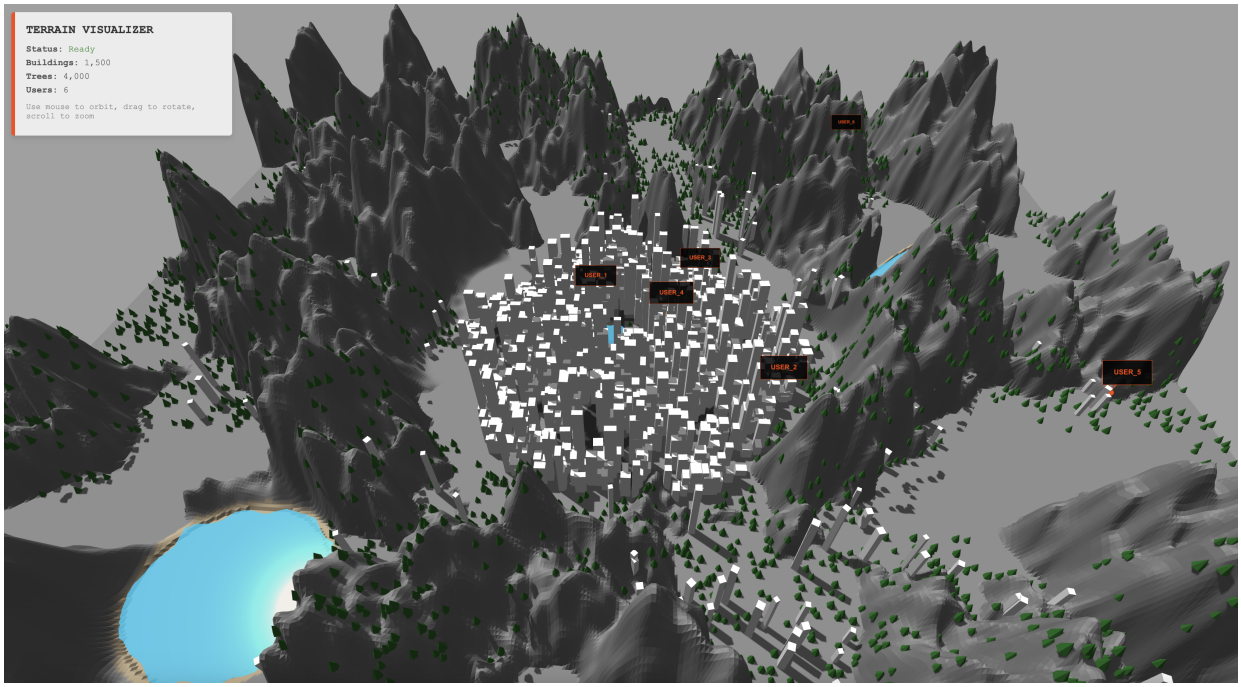

Fig. 3: 3D terrain visualization showing buildings, trees, lakes, and user positions. The environment presents significant challenges for trajectory planning due to complex obstacle distribution and terrain height variations.

length was exactly 1.0 steps, indicating that all episodes terminated immediately after the first action. The average episode reward was -501.17, which can be precisely explained as the sum of the collision penalty (-500) and the step penalty (-1.0), confirming that episodes ended instantaneously due to collisions.

TABLE IV: Training Results Summary

Metric	Value
Total Episodes	20,000
Success Rate	0%
Average Episode Length	1.0 steps
Average Episode Reward	-501.17
Collision Statistics	
Building Collisions	20,000
Terrain Collisions	0
Tree Collisions	0
Lake Collisions	0

The collision statistics reveal a critical pattern: all 20,000 episodes resulted in building collisions, with zero collisions occurring for terrain, trees, or lakes. This 100% building collision rate at the first step indicates that the agent consistently collided with buildings immediately upon taking its first action, preventing it from ever accumulating meaningful navigation experience. This catastrophic failure pattern suggests fundamental issues with the training setup, specifically: (1) the starting position may be invalid or immediately adjacent to obstacles, (2) the action scale is too large for the environment's obstacle density, and (3) the reward function provides no intermediate guidance signals.

B. Analysis of Challenges

I identify several factors contributing to the training failure:

1) Starting Position Issues: The starting position (User1) may be located inside or immediately adjacent to a building. This causes immediate collision at the first step, preventing the agent from accumulating any meaningful learning experiences. The agent receives only negative rewards and terminates immediately, creating a sparse reward problem where positive learning signals are extremely rare.

2) Reward Function Design Limitations: The current reward function design suffers from several critical issues that prevent effective learning. The reward structure exhibits severe sparsity problems: positive reward (+500) is only received when reaching the goal, which occurred with 0% frequency in my experiments, meaning the agent never receives positive reinforcement. Negative reward (-500) is received immediately upon collision, terminating the episode and providing only a binary failure signal without intermediate feedback. The step penalty (-1) provides minimal guidance, being overwhelmed by the large collision penalty and offering insufficient incentive for efficient path planning.

More fundamentally, the reward function lacks essential guidance signals that could help the agent learn incrementally. There is no reward for distance reduction, meaning the agent cannot learn that moving toward the goal is beneficial, as it receives no feedback until actually reaching the goal (which never occurred in any episode). There is no soft penalty for approaching obstacles, only a hard collision penalty that terminates the episode immediately, preventing the agent from learning to maintain safe distances gradually. Consequently,

the agent cannot learn incrementally; it only receives binary success/failure signals (which are exclusively failure signals in this case), making it impossible to improve through gradual refinement of behavior. This represents a classic sparse reward problem in RL, where the lack of intermediate feedback prevents effective learning.

3) Action Space Configuration: The action scale $\lambda_a = 10.0$ m appears to be too large for this environment, contributing significantly to the training failure. With such a large step size, the first action taken by the agent often moves it directly into an obstacle, leading to immediate collision before any learning can occur. Fine-grained navigation is impossible with 10-meter steps, as the agent cannot make subtle adjustments necessary to navigate through narrow passages or maintain precise distances from obstacles. The agent cannot learn to make small adjustments near obstacles, which is essential for safe navigation in cluttered environments, because every action represents a substantial displacement that frequently results in collision. A smaller action scale (e.g., 2.0-5.0 m) would allow for more precise control, enabling the agent to make incremental movements and learn safe navigation behaviors through gradual refinement.

4) Environment Complexity: The environment presents significant challenges: 3D navigation requires simultaneous consideration of horizontal and vertical movement, multiple obstacle types (buildings, trees, lakes, terrain) have different collision geometries, and terrain height variations require maintaining clearance above varying terrain. The 8-dimensional state space may not capture sufficient local obstacle information.

5) Training Strategy Limitations: Several training strategy issues contributed to the learning failure. The training duration of 20,000 timesteps may be insufficient for such a complex environment. The training setup lacks curriculum learning, starting immediately with full environmental complexity. Poor exploration capability results from immediate episode terminations, preventing the collection of diverse experiences necessary for learning.

C. Discussion of Limitations

The results highlight fundamental challenges in applying RL to complex 3D navigation tasks. Reward design difficulty represents a core challenge: designing effective reward functions for sparse-reward problems requires careful balance between providing sufficient guidance and maintaining simplicity. The exploration-exploitation trade-off manifests as a critical dilemma: the agent needs to explore to discover successful paths, but exploration leads to immediate termination, preventing the accumulation of positive learning experiences. The complexity scaling challenge suggests that the environment's complexity may exceed what can be learned with standard RL approaches without additional engineering efforts such as curriculum learning, reward shaping, or hierarchical decomposition.

VI. Proposed Improvements and Future Work

Based on the challenges identified, I propose the following improvements for future work:

A. Reward Function Enhancement

The most critical improvement is to redesign the reward function to provide dense, incremental feedback. The proposed enhanced reward function incorporates distance reduction and safety bonuses:

$$\begin{aligned} R(s_t, a_t) =& \alpha \cdot (d_{t-1} - d_t) + \beta \cdot \text{safety\_bonus} \\ &+ R_{\text{GOAL}} + R_{\text{COLLISION}} + R_{\text{STEP}} \end{aligned} \tag{6}$$

where $\alpha \cdot (d_{t-1} - d_t)$ represents a distance reduction reward that encourages the agent to move toward the goal, and $\beta \cdot$ safety_bonus provides a soft safety reward based on clearance to the nearest obstacle, encouraging safe navigation. The remaining terms (goal reward, collision penalty, and step penalty) are retained from the original design. This enhanced reward structure provides dense, incremental rewards that continuously guide the agent toward the goal while maintaining safety, addressing the sparse reward problem identified in the current implementation.

B. Hyperparameter Adjustments

Several hyperparameter adjustments are necessary to improve learning stability and convergence. The action scale λ_a should be reduced from 10.0 to 2.0–5.0 m to enable finer control and prevent immediate collisions. The training duration should be substantially increased to 100,000+ timesteps to provide sufficient experience for learning in this complex environment. Additionally, exploration noise should be increased to facilitate better state space exploration, and adaptive learning rate scheduling should be considered to stabilize training dynamics throughout the learning process. These adjustments address the action space configuration issues and training duration limitations identified in the experimental results.

C. Training Strategy Improvements

Training strategy improvements are essential for successful learning in this complex environment. Curriculum learning should be implemented, starting with simple scenarios featuring fewer obstacles and flat terrain, then gradually increasing complexity as the agent demonstrates proficiency. Starting position validation is crucial to ensure initial positions are safe, or to automatically adjust to the nearest safe location if the designated starting point is problematic. Pre-training using imitation learning or expert demonstrations could provide a strong initialization for the policy network. Furthermore, state augmentation should be considered to add local obstacle information such as nearest building distance, nearest tree distance, and other relevant spatial features to enhance the agent's awareness of its immediate surroundings.

D. Alternative Approaches

Future work could explore several alternative approaches that may prove more effective than direct end-to-end RL training. Hierarchical RL could decompose the problem into high-level path planning and low-level control, simplifying the learning task at each level. Hybrid approaches combining RL

with classical path planning methods such as A* could provide initialization or serve as a baseline, leveraging the strengths of both paradigms. Multi-agent RL employing multiple agents with different objectives could explore the state space more effectively through diverse exploration strategies. Transfer learning approaches that pre-train on simpler environments and then transfer knowledge to the complex terrain could also reduce the sample complexity and training time required.

VII. Conclusion

In this work, I applied TD3 to 3D trajectory planning for UAVs in complex terrain environments, extending my previous successful application of RL (DDQN for attitude control) to continuous action space trajectory planning. I successfully implemented the RL framework, environment design, and comprehensive training infrastructure. However, I did not achieve satisfactory training results, with the model failing to learn effective obstacle avoidance strategies.

The key contributions include extending the RL framework from discrete control (DDQN) to continuous control (TD3), designing appropriate state and action spaces for 3D terrain navigation, and identifying key challenges in applying RL to complex 3D environments. The training failures revealed critical insights: sparse reward problems require careful reward function design, complex 3D environments demand sophisticated training strategies, and action space configuration significantly affects learning success.

Future work should focus on reward function redesign incorporating distance reduction and safety bonuses, hyperparameter optimization, curriculum learning, and potentially hybrid approaches combining RL with classical path planning methods. This work provides a foundation for such improvements and contributes to understanding the challenges in applying RL to complex cyber-physical systems.

References

[1] K. Ashton, "That 'Internet of Things' Thing," *RFiD Journal*, vol. 22, no. 7, pp. 97-114, 1999.

[2] E. Besada-Portas, L. de la Torre, J. M. de la Cruz, and B. de Andrés-Toro, "Evolutionary Trajectory Planner for Multiple UAVs in Realistic Scenarios," *IEEE Transactions on Robotics*, vol. 26, no. 4, pp. 619-634, 2010.

[3] O. Ghdiri, W. Jaafar, S. Alfattani, J. B. Abderrazak, and H. Yanikomeroglu, "Offline and Online UAV-Enabled Data Collection in Time-Constrained IoT Networks," *IEEE Trans. Green Commun. Netw.*, vol. 5, no. 4, pp. 1918–1933, 2021.

[4] N. Hu, Z. Tian, X. Du, N. Guizani, and Z. Zhu, "Deep-Green: A Dispersed Energy-Efficiency Computing Paradigm for Green Industrial IoT," *IEEE Transactions on Green Communications and Networking*, vol. 5, no. 2, pp. 750-764, 2021.

[5] J. Pei, H. Chen, and L. Shu, "UAV-assisted connectivity enhancement algorithms for multiple isolated sensor networks in agricultural Internet of Things," *Computer Networks*, vol. 207, pp. 108854, 2022.

[6] P. Poksawat, L. Wang, and A. Mohamed, "Gain Scheduled Attitude Control of Fixed-Wing UAV With Automatic Controller Tuning," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 4, pp. 1192-1203, 2018.

[7] C. Rottondi, F. Malandrino, A. Bianco, C. F. Chiasserini, and I. Stavrakakis, "Scheduling of emergency tasks for multiservice UAVs in post-disaster scenarios," *Computer Networks*, vol. 184, pp. 107644, 2021.

[8] B. G. Shri Varu, N. Jayarajan, and T. Ganesan, "Optimizing Quadcopter Trajectory Tracking with Deep-Q Reinforcement PID Controller in Uncertain Environments," in *2024 IEEE Third International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES)*, 2024, pp. 341-346.

[9] L. Tan, S. Guo, P. Zhou, Z. Kuang, S. Long, and Z. Li, "Multi-UAV-Enabled Collaborative Edge Computing: Deployment, Offloading and Resource Optimization," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 11, pp. 18305-18320, 2024.

[10] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning*, 2018, pp. 1587-1596.

[11] X. Wei, L. Cai, N. Wei, P. Zou, J. Zhang, and S. Subramaniam, "Joint UAV Trajectory Planning, DAG Task Scheduling, and Service Function Deployment Based on DRL in UAV-Empowered Edge Computing," *IEEE Internet of Things Journal*, vol. 10, no. 14, pp. 12826-12838, 2023.

[12] B. Xia, I. Mantegh, and W.-F. Xie, "Hybrid Framework for UAV Motion Planning and Obstacle Avoidance: Integrating Deep Reinforcement Learning with Fuzzy Logic," in *2024 10th International Conference on Control, Decision and Information Technologies (CoDIT)*, 2024, pp. 2662-2669.

[13] P. Qin, Y. Fu, Y. Xie, K. Wu, X. Zhang, and X. Zhao, "Multi-Agent Learning-Based Optimal Task Offloading and UAV Trajectory Planning for AGIN-Power IoT," *IEEE Transactions on Communications*, vol. 71, no. 7, pp. 4005-4017, 2023.

[14] C. Zheng, K. Pan, J. Dong, L. Chen, Q. Guo, S. Wu, H. Luo, and X. Zhang, "Multi-Agent Collaborative Optimization of UAV Trajectory and Latency-Aware DAG Task Offloading in UAV-Assisted MEC," *IEEE Access*, vol. 12, pp. 42521-42534, 2024.

[15] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-Baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1-8, 2021.