

Autonomous Path Selection of Unmanned Aerial Vehicle in Dynamic Environment Using Reinforcement Learning

Komsun Tamanakijprasart*,
Adolfo Perrusquia[†],
Sabyasachi Mondal[‡],
and Antonios Tsourdos[§]

Cranfield University, Cranfield MK430AL, United Kingdom

The Unmanned Aerial Vehicle (UAV) is an emerging area within the aviation industry. Currently, fully autonomous UAV operations in real-world scenarios are rare due to low technology readiness and a lack of trust. However, Artificial Intelligence (AI) offers powerful tools to adapt to changing conditions and handle complex perceptions. In autonomous vehicles, automotive self-driving technologies have made significant advances. To enhance the level of autonomy in aviation, it is beneficial to analyze these frameworks and extend autonomous driving principles to autonomous flying. This research introduces a novel solution for ensuring safe navigation in UAVs by adopting the concept of autonomous lane or path selection strategies used in cars. The approach employs deep reinforcement learning (DRL) for high-level decision-making in selecting the appropriate path generated by various established algorithms that consider different scenarios. Specifically, the Interfered Fluid Dynamical System (IFDS) [1] is utilized for guidance and the PID for the flight control system. The UAV can choose between global and local paths and determine the appropriate speed for following these paths. This proposed framework lays the foundation for future research into practical and safe navigation strategies for UAVs.

Nomenclature

DDQN	Double Deep Q-Network
DRL	Deep Reinforcement Learning
IFDS	Interfered Fluid Dynamical System
UAS	Unmanned Aircraft Systems
UAV	Unmanned Aerial Vehicle

I. Introduction

IN recent years, researchers have explored various approaches to enhance unmanned aerial vehicle (UAV) navigation. One key challenge lies in efficiently avoiding dynamic obstacles while ensuring smooth trajectories. Path planning algorithms offer long-term considerations, but their effectiveness depends on the specific environment. On the other hand, detect-and-avoid (DAA) focuses on immediate safety but may not optimize long-term objectives. Striking the right balance between these two aspects is essential for robust UAV navigation. In this context, artificial intelligence (AI) techniques play a crucial role.

In many pieces of literature, the UAV navigation solution combines navigation algorithms with a hierarchy of scenarios and rules to ensure safety. However, in practice, it is impossible to guarantee the sufficiency of such an

*PhD candidate in Aerospace, Faculty of Engineering and Applied Science, Cranfield University, United Kingdom. Email: komxun@gmail.com

[†]Lecturer in Reinforcement Learning for Engineering, Faculty of Engineering and Applied Science, Cranfield University, Email: adolfo.perrusquia-guzman@cranfield.ac.uk

[‡]Lecturer in Autonomous Systems, Faculty of Engineering and Applied Science, Cranfield University, Email: sabyasachi.mondal@cranfield.ac.uk

[§]Professor and Director of AI, Autonomy and Robotics, Faculty of Engineering and Applied Science, Cranfield University, Email: a.tsourdos@cranfield.ac.uk, AIAA Senior Member.

approach to tackle all possible real-world environment cases. The learning-based approach, such as reinforcement learning, is more suitable for tackling unpredictable real-world environments. This approach can use the historical flight data and iteratively interact with the environment through actions.

A. Related Studies

Pente-Castro et al.[2] conducted a comprehensive review of AI techniques applied to UAV path planning, highlighting the importance of integrating AI methods into navigation systems. Among these techniques, reinforcement learning (RL) has emerged as a powerful tool in aviation, with successful applications in decision-making, trajectory planning, and control. Razzaghi et al.[3] provided an in-depth review of RL's role in aviation, covering areas such as attitude control, flight planning, air traffic management, flight delay prediction, and aircraft performance parameter estimation.

Yan et al. [4] use a DRL method to formulate the UAV navigation in a dynamic environment with the provided map. However, they studied a 2D scenario where the area is divided into grids, and the action space is eight adjacent grids to which the UAV can move. As a result, the generated trajectory is not smooth, and extending such an approach into navigation in 3D space is difficult. Brittain et al.[5–7] investigated multi-agent deep reinforcement learning (MARL) for resolving conflicts among multiple UAVs. Targeting large-scale air traffic management applications, their studies also employ 2D scenarios with speed adjustments as the primary control variable.

More recently, Guo et al. [8] proposed a novel approach to dynamic obstacle avoidance of UAVs using Deep Reinforcement Learning (DRL). They formulated the navigation problem using three interconnected networks: the avoid network for obstacle avoidance, the acquired network for reaching the destination, and the packed network, which selects actions from either network. Du et al. [9] introduced a hybrid hierarchical two-layer motion planning framework, leveraging DRL exclusively for local path planning while relying on the conventional A* algorithm for global planning. This hybrid approach demonstrated superior performance to traditional motion planning methods, offering an efficient solution for complex navigation tasks.

In addition to UAV navigation, DRL has also been extensively studied in the field of autonomous driving [10–12]. Concepts from autonomous driving, such as hierarchical decision-making and trajectory optimization, offer valuable insights for unmanned aircraft systems (UAS).

B. Contributions

This study focuses on collision avoidance of a single UAV in a 3D dynamic environment. We extend our previous dynamic auto-routing framework [13] by integrating state-of-the-art reinforcement learning into the behavioural decision-making process. Drawing inspiration from [14], where the behavioural problem of on-road vehicles is formulated as a sequence of lane selections, we similarly formulate the UAV navigation problem. In our formulation, the agent can select a predefined global path or local paths computed from the dynamic auto-routing and determine an appropriate speed. The trajectory planning and control tasks are delegated to underlying algorithms, including IFDS [1] and the PID controller.

Our approach simplifies UAV navigation into two primary behaviours:

- **Path Selection:** We offer two path-planning modes: the global path and the local path. The agent can dynamically choose between these modes based on the environment and situations.
- **Speed Control:** Besides path selection, the agent determines an optimal speed to avoid collisions and reach the destination efficiently.

This methodology offers several key advantages:

- 1) **Leveraging proven techniques:** Existing navigation methods, such as the IFDS algorithm and the PID controller, have undergone rigorous testing and validation. By incorporating them, we benefit from their proven effectiveness in navigation and control tasks.
- 2) **Reduced complexity:** Delegating detailed path planning and collision avoidance tasks to existing algorithms simplifies the reinforcement learning problem. The agent only needs to focus on high-level discrete decisions, such as between global and local paths.
- 3) **Robustness and safety:** The global-local path adaptation is especially useful in real-world scenarios where airspace regulations are in place. It represents a practical step towards using air corridors for UAS traffic management.
- 4) **Strategic learning:** By training for high-level decision-making, the agent learns to make strategic choices without knowing the intricate details of low-level control tasks, obstacle avoidance, and path planning.

II. Problem Formulation

A. Aim and Objectives

This study aims to tackle the collision avoidance problem of the Unmanned Aircraft Systems (UAS) by extending the concept of lane-changing from autonomous vehicles to the UAS. Collision-free navigation for UAS can be formulated as a sequential decision-making process, where the UAV dynamically selects alternate paths to ensure safety. The concept is illustrated in Fig.1. Deep reinforcement learning is utilized to enable this intelligent decision-making. Specifically, we employ the Double Deep Q-Network (DDQN) with a dueling network architecture [15].

We aim to demonstrate proof-of-concept and point out the advantages of this approach over traditional methods that rely solely on trajectory planning. The problem is systematically broken down in Table 1. By formulating the problem in this way, we can leverage the power of Artificial Intelligence (AI) to find optimal policies that balance the trade-off between reaching the destination quickly, avoiding collisions, and adhering to the predefined route. This formulation lays the new foundation for developing efficient and safe navigation strategies for UAVs.

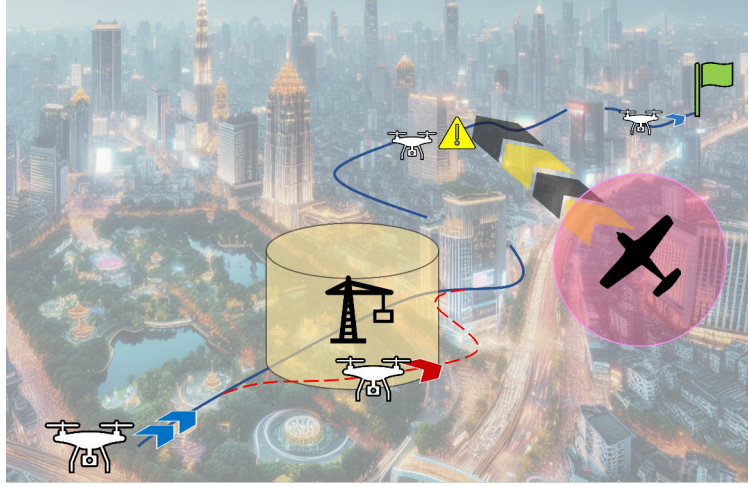


Fig. 1 Problem scenario. Intelligent behaviour includes: 1) UAV temporarily switching from the optimal mission route to a secondary route to avoid threats, and 2) UAV remaining on the same route but decelerating to wait until the threat subsides.

Table 1 Problem formulation breakdown

Topic	Content
Requirements	<ul style="list-style-type: none"> • The UAV shall reach its destination as quickly as possible • The UAV shall avoid both static and dynamic obstacles
Constraints	<ul style="list-style-type: none"> • Static obstacles such as buildings • Dynamic obstacles such as other UAVs • Limited detection range of the UAV • Performance of the UAV
Assumptions	<ul style="list-style-type: none"> • The global mission route is predefined and fixed • The local route is computed onboard the UAV • Obstacle locations are only known upon detection by the UAV • No uncertainty in all data

III. Method and Implementation

A. Interfered Fluid Dynamical System

Inspired by how water avoids rocks in nature, the Interfered Fluid Dynamical System (IFDS) algorithm [1] generates a collision-free path with the modified sink velocity. With closed-form analytical solutions, the IFDS algorithm can be computed for dynamic obstacle avoidance in a 3D environment in real time.

The core of the IFDS algorithm is the modification of the velocity field by the modulation matrix $\mathbf{M}(\xi)$, which responds to the shape of the obstacles. The matrix $\mathbf{M}(\xi)$ is given as follow:

$$\mathbf{M}(\xi) = \mathbf{I} - \frac{\mathbf{n}(\xi)\mathbf{n}(\xi)^\top}{|\Gamma(\xi)|^{\frac{1}{\rho(\xi)}} \mathbf{n}(\xi)^\top \mathbf{n}(\xi)} + \frac{\mathbf{t}(\xi)\mathbf{n}(\xi)^\top}{|\Gamma(\xi)|^{\frac{1}{\sigma(\xi)}} \|\mathbf{t}(\xi)\| \|\mathbf{n}(\xi)\|}$$

where $\xi = (x, y, z)$ represents the position in 3D space, and $\Gamma(\xi)$ is the object modeling function. The vectors $\mathbf{n}(\xi)$ and $\mathbf{t}(\xi)$, along with the parameters $\rho(\xi)$ and $\sigma(\xi)$, are defined as follows:

$$\begin{aligned} \mathbf{n}(\xi) &= \left[\frac{\partial \Gamma(\xi)}{\partial x}, \frac{\partial \Gamma(\xi)}{\partial y}, \frac{\partial \Gamma(\xi)}{\partial z} \right]^\top & \mathbf{t}(\xi) &= \mathbf{R} \mathbf{t}'(\xi) \\ \rho(\xi) &= \rho_0 \exp \left(1 - \frac{1}{d_0(\xi) d(\xi)} \right) & \sigma(\xi) &= \sigma_0 \exp \left(1 - \frac{1}{d_0(\xi) d(\xi)} \right) \end{aligned}$$

Here, $\mathbf{t}(\xi)$ is the tangential vector. To modify the path's tangential angle α , $\mathbf{t}(\xi)$ is transformed into the tangential coordinate system using the rotation matrix $\mathbf{R}$. The vectors $\mathbf{t}_i^1$, $\mathbf{t}_i^2$, and $\mathbf{n}$ define the new x' , y' , and z' axes, which are used to construct the tangent coordinate system.

$$\begin{aligned} \mathbf{t}_i^1(\xi) &= \left[\frac{\partial \Gamma(\xi)}{\partial y}, \frac{-\partial \Gamma(\xi)}{\partial x}, 0 \right]^\top \\ \mathbf{t}_i^2(\xi) &= \left[\frac{\partial \Gamma(\xi)}{\partial x} \frac{\partial \Gamma(\xi)}{\partial z}, \frac{\partial \Gamma(\xi)}{\partial y} \frac{\partial \Gamma(\xi)}{\partial z}, -\left(\frac{\partial \Gamma(\xi)}{\partial x} \right)^2 - \left(\frac{\partial \Gamma(\xi)}{\partial y} \right)^2 \right]^\top \\ \mathbf{t}'(\xi) &= \begin{bmatrix} \cos \alpha & \sin \alpha & 0 \end{bmatrix}^\top \end{aligned} \tag{1}$$

Finally, the rotation matrix $\mathbf{R}$ to transform the coordinate from $O'_{x'y'z'}$ to O_{xyz} can be expressed as follow:

$$\begin{aligned} \mathbf{R} &= \begin{bmatrix} \mathbf{t}_i^1(\xi) & \mathbf{t}_i^2(\xi) & \mathbf{n}(\xi) \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial \Gamma(\xi)}{\partial y} & \frac{\partial \Gamma(\xi)}{\partial x} \frac{\partial \Gamma(\xi)}{\partial z} & \frac{\partial \Gamma(\xi)}{\partial x} \\ \frac{-\partial \Gamma(\xi)}{\partial x} & \frac{\partial \Gamma(\xi)}{\partial y} \frac{\partial \Gamma(\xi)}{\partial z} & \frac{\partial \Gamma(\xi)}{\partial z} \\ 0 & -\left(\frac{\partial \Gamma(\xi)}{\partial x} \right)^2 - \left(\frac{\partial \Gamma(\xi)}{\partial y} \right)^2 & \frac{\partial \Gamma(\xi)}{\partial z} \end{bmatrix} \end{aligned}$$

B. Routes Option

Whenever the UAV detects a pop-up threat or dynamic obstacle, it can temporarily switch from following a global path to a local path to avoid obstacles. Then, it returns to the global path after the threat subsides. The global and local path-switching concept is illustrated in Fig.2.

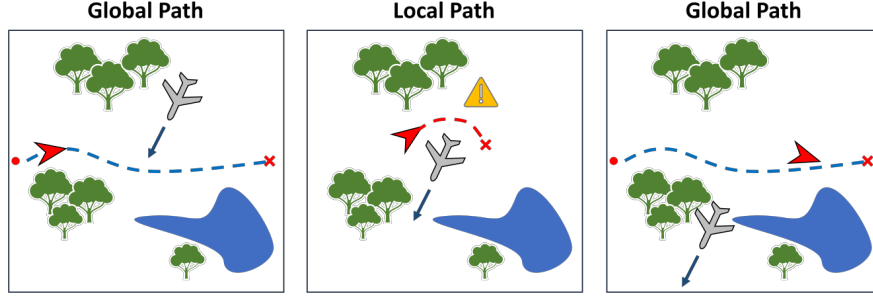


Fig. 2 Scenarios illustrating global and local path adaptation

1. Global Path

The global path represents the authorized route given to each UAV before it takes off on each mission. We assume the global route is static and does not react to the changing environment. In this work, the global route for the UAV agent is set to be a single straight line in Euclidean space from the UAV's initial location to its destination.

2. Local Path

Local paths are the unauthorized routes instantly generated by the UAV to deal with the risk of collision. In this work, local routes are computed in real-time using the IFDS path-planning algorithm incorporating current object detection data. The variation of each route results from eight configurations of tangential angle by substituting Eq.1 with $\alpha \in \{0^\circ, 70^\circ, 90^\circ, 110^\circ, 180^\circ, 250^\circ, 270^\circ, 290^\circ\}$. This allows UAVs to evade obstacles from eight different angles in three-dimensional space, as shown in Fig.3

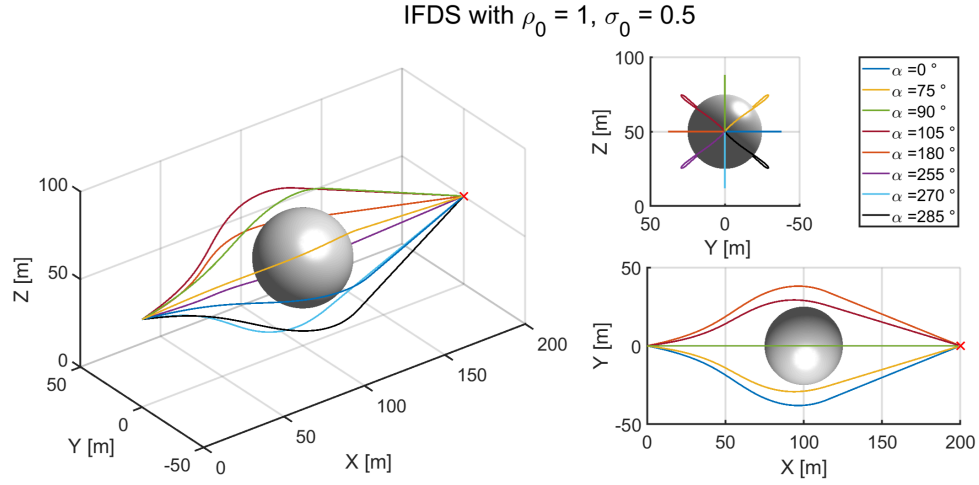


Fig. 3 Local path using IFDS algorithm with varying tangential angle α

C. Detection with Batch Ray-Casting and ADS-B data

While the IFDS algorithm offers computational efficiency and real-time capabilities, it relies on the strict assumption that accurate ground truth information about all obstacles is available beforehand. This limitation can be mitigated by integrating two complementary sources of obstacle detection: batch ray-casting for local sensing and ADS-B data for situational awareness.

Ray casting is a technique in 3D rendering which involves sending out imaginary lines or rays from a point in space, as shown in Fig.4. The information on relative position can be obtained if the objects are hit by these rays. Real-world sensors, including LIDAR, SONAR, RADAR, ultrasonic sensors and IR sensors, operate similarly to surveillance

broadcasts, emit directional signals such as light, sound, or radio waves and detect how those signals interact with objects. In this work, rays are cast uniformly outward the UAV in all directions in three dimensions, creating a sphere as shown in Fig.5

Automatic dependent surveillance-broadcast (ADS-B) is the standard of aircraft broadcasting their identity, position, and other information from their onboard systems. ADS-B can be captured on board other aircraft for situational awareness.

In this work, we assume the information, such as the position of all aircraft in the mission area, is provided via ADS-B data and can be retrieved by the UAV at any time. The owner will only see and retrieve aircraft data once detected by ray cast to make it memory efficient and scalable to high-density airspace.

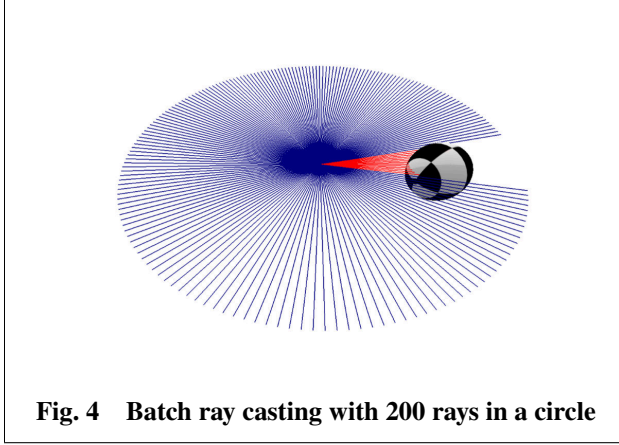


Fig. 4 Batch ray casting with 200 rays in a circle

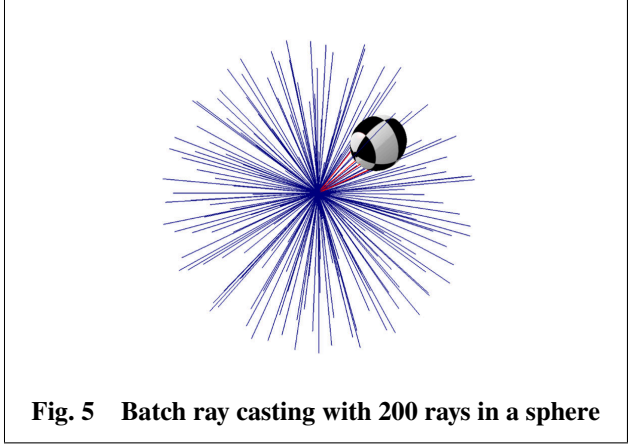


Fig. 5 Batch ray casting with 200 rays in a sphere

D. Reinforcement Learning

Reinforcement learning (RL) is a rapidly growing area in artificial intelligence that focuses on training agents to make sequences of actions. By interacting with an environment, an agent learns to achieve a goal in uncertain and complex circumstances. The agent's learning process is guided by a policy, denoted as π , which essentially acts as a strategy for selecting actions, given the current state of the environment. This policy aims to maximize the cumulative reward, which signals the agent about the value of its actions.

The formal framework used to model the RL problem is the Markov Decision Process (MDP). An MDP is defined by a set of states (S), a set of actions (A), a state transition probability function (T), a reward function (R), and a discount factor (γ). The state transition probability function, $T : S \times A \rightarrow S$, determines the likelihood of transitioning to a new state, given the current state and action taken. The reward function, $R : S \times A \times S \rightarrow \mathbb{R}$, defines the reward received after transitioning from one state to another due to an action taken.

The Markov property is a core concept in MDPs, stating that the future is independent of the past, given the present. This means that the probability distribution of future states depends solely on the current state and action, not on the sequence of events that preceded it. This property simplifies the complexity of the decision-making process, as the agent does not need to consider the entire history of states but only the current situation. At each time step, the agent evaluates its situation and selects an action expected to yield the highest future discounted return. The return, denoted as G , is the total accumulated discounted reward that the agent can expect to receive over the future, starting from the current state. The return is expressed as:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \quad (2)$$

In an RL scenario, the agent's ultimate goal is to discover the optimal policy, π^* , that maximizes the expected return from any given state. Finding this policy is a complex challenge and involves methods such as dynamic programming, Monte Carlo methods, and temporal-difference learning, each with strengths and applications.

E. Integration of Dynamic Autorouting with Deep Reinforcement Learning

The integration of the Interfered Fluid Dynamical System (IFDS) algorithm with artificial intelligence methods is limited. Wang et al. [16] utilized neural networks to adaptively adjust the coefficients of the IFDS algorithm, resulting

in an optimal path. Similarly, Zhang et al. [17] employed reinforcement learning techniques, including Proximal Policy Optimization (PPO), Twin Delayed DDPG (TD3), Deep Deterministic Policy Gradient (DDPG), and Soft Actor-Critic (SAC), to enable the agent to adjust the IFDS tuning parameters dynamically. While these studies demonstrate success in the path-planning phase, they do not address how UAVs navigate using these paths in the actual scenario.

There is a significant gap in the treatment of how UAVs navigate using these planned paths. Even the best path may not ensure safety, particularly in rapidly changing environments. The crucial factor is how effectively the UAV utilizes the path. Therefore, in this work, we extend the application of the IFDS algorithm by integrating it with deep reinforcement learning to improve real-time decision-making and adaptability. Figure 6 illustrates an overall design structure.

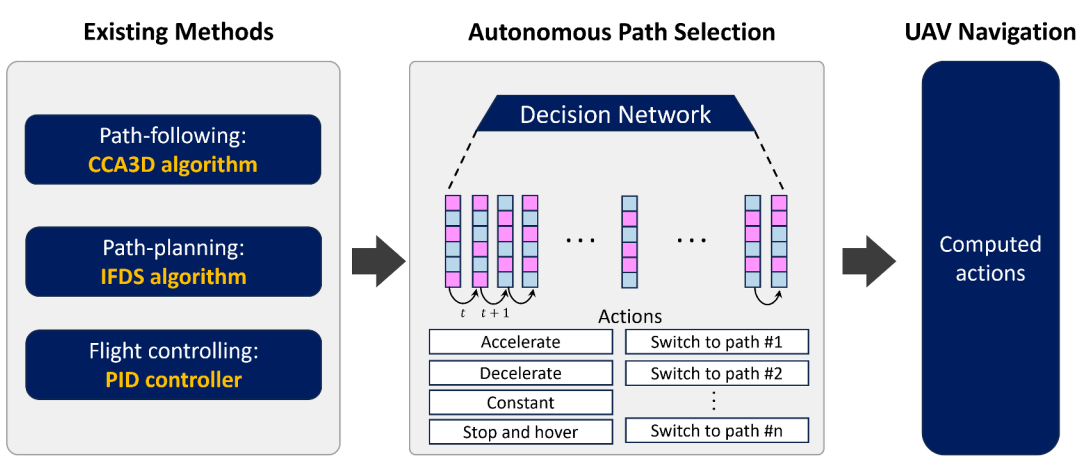


Fig. 6 Autonomous path selection design for UAV navigation

1. Observation Space

The observation is the set of variables the agent perceives at any given time. This work assumes that dynamic obstacles are not pre-known, but agents can detect them with the onboard sensors. This makes the problem into a Partially Observable Markov Decision Process (POMDP). In our case, the agent has the information shown in Table 2.

The key parameter is the number of rays. A higher number of rays means the agent has a smaller blind spot and can detect smaller objects. However, this comes with a higher computational cost and increased training time. In this study, $n_{ray} = 200$, meaning there are 1013 states observed at every step.

Table 2 Observation space

Number of state	Variables	Description
3	(x, y, z)	UAV's current location in Cartesian world coordinates
3	(ϕ, γ, ψ)	UAV's current Euler's angles
3	(v_x, v_y, v_z)	UAV's current linear velocity
3	(w_x, w_y, w_z)	UAV's current angular velocity
1	$\ d_g\ $	Displacement magnitude from UAV's current location to the destination
$1 \times n_{ray}$	U_{id}	Unique id of the hit object from ray casting
$1 \times n_{ray}$	r_{hit}	Hit fraction in range [0,1] along the ray
$3 \times n_{ray}$	$(x_{hit}, y_{hit}, z_{hit})$	Hit position in Cartesian world coordinates

2. Action Space

While we can divide the actions of UAVs into continuous (speed control) and discrete actions (path selection), this approach may complicate the episode evaluation. Many studies effectively integrated continuous and discrete actions, for instance, combining flight control with weapon firing [18]. These studies typically assign separate rewards for each task. In contrast, our approach uses a unified reward system; path selection and speed control will be combined together in a single training network. Consequently, a set of discrete actions is defined as detailed in Table 3. The specific magnitudes of acceleration and deceleration will be determined based on the UAV's performance characteristics and flight envelope.

Table 3 Action space for autonomous path selection

Action	Description
a_1	Keep the current path, follow with an acceleration
a_2	Keep the current path, follow with a deceleration
a_3	Stop and hover at the current position
a_4	Follow the global path with a constant velocity
a_5	Follow the local path no.1 with a constant velocity
...	
a_{12}	Follow the local path no.8 with a constant velocity

3. Reward

Inspired by a study from [19] where the reinforcement learning is formulated for collision avoidance of UAVs in 2D, the reward function in this work is defined in Eq. 3.

$$R_i(s_t) = \begin{cases} -2 & \text{mid-air collision,} \\ -1 & \text{near mid-air collision.} \\ 2 & \text{reach the destination.} \\ r_t & \text{otherwise.} \end{cases} \quad (3)$$

The near mid-air collision (NMAC) occurs if any of the rays detect obstacles within 20% of their length. In air traffic management, this range should align with the formal definition of NMAC. The reward r_t is given to the agent in the nominal state where no threats are detected and is defined as follows:

$$r_t = \begin{cases} r_G & \text{when UAV follows the global path,} \\ 0.25 \cdot r_G & \text{when UAV follows the local path} \end{cases}$$

$$r_G = \frac{d_g/T_{max}}{d_{g,max}}$$

where d_g is the UAV's distance to its destination at the current time step, and $d_{g,max}$ is the total displacement from the UAV's starting position to its destination. This reward formulation encourages forward progress by penalizing behaviour, such as hovering (action a_3), which does not contribute to reaching the destination. In addition, the reward r_t depends on the type of path the UAV is currently following. The reward will be reduced whenever the agent follows the local route, as this poses safety concerns in real scenarios.

F. Dueling Double Deep Q-Learning

The action space in this study (Table 3) includes many actions with potentially similar values, as different paths may yield only slight variations in rewards. In such scenarios, the use of action-advantages is particularly beneficial. To address this, we employ the Dueling Double Deep Q-Network (Dueling DDQN), which allows the agent to effectively

compare similarly valued actions, such as evaluating alternative path options. Consequently, Dueling DDQN provides a more stable and robust value-based deep reinforcement learning (DRL) method compared to traditional approaches like Double Deep Q-Network (DDQN) [20] and Deep Q-Network (DQN) [21].

Dueling DDQN shares many components with DDQN and DQN but introduces a key enhancement in its neural network structure. Its primary objective is to approximate the optimal action-value function $q^*(s, a)$. Like DDQN, double learning is adopted where two instances of the action-value function from the online and target networks are used to cross-validate the estimates. This reduces the chance of positive bias.

The unique feature of Dueling DDQN lies in its dueling network architecture [15] which splits the Q-function into two streams: one estimates the state-value function $V(s)$, and the other estimates the action-advantage function $A(s, a)$. This design improves sample efficiency, as a single experience tuple can enhance the value estimates of all possible actions for a given state. The components and configurations of the Dueling DDQN used in this study are detailed in Table 4.

Table 4 Dueling DDQN components and configurations

Component	Configuration
Neural network architecture	Dueling network with state-input and value-output layers (nodes: 1013, 512, 128, 1; 12, 12),
Policy evaluation	Off-policy TD targets: $r + \gamma \max_{a'} Q(s', a'; \theta)$ with $\gamma = 0.9$, using double learning
Loss function	Mean Squared Error (MSE).
Optimizer	Root mean square propagation (RMSprop) with a learning rate of 0.0007.
Policy improvement	Exponentially decaying epsilon-greedy policy: epsilon decays from 1.0 to 0.3 over approximately 100,000 steps. Greedy action selection for evaluation.
Replay buffer	Capacity between 320 and 50,000 samples; batch size of 64.
Target network update frequency	Updates after every step.

IV. Simulation Setup

A. Quadcopter Configurations

1. Quadcopter Dynamics

This study adopts the PyBullet-drone framework [22], a general-purpose multi-agent *Gym* environment [23] designed for quadcopters simulation. Specifically, the dynamics of the Bitcraze Crazyflie 2.x [24] are modelled, incorporating six degrees of freedom dynamics. The simulation is run at the frequency of 60 Hz and includes forces generated by each of the four motors and the resulting torques. Additionally, aerodynamic effects such as drag, ground effect, and downwash are considered, providing a realistic representation of quadcopter behaviour.

2. Quadcopter Configurations

For this study, all UAVs are assumed to be identical, equipped with the same sensor configuration capable of producing 200 rays uniformly distributed in a spherical pattern, as illustrated in Fig.5. The critical parameters for the UAVs are summarized in Table5.

Table 5 UAV parameters setups.

Parameter	Value
Speed limit	0.3 m/s
Acceleration and deceleration magnitude	0.05 m/s ²
UAV boundary box	1 m × 1 m × 1 m
Number of rays per UAV	200 rays
Ray detection range	1.5 m

B. Training and Evaluation Scenario

The random scenarios of multiple UAVs flying are generated using two concentric circles representing possible initial locations and destinations for each UAV.

For each UAV, points are selected from each circle, and slight variability is added to the base radii. This variability results in each UAV having a slightly different radial distance from the origin, creating a spread. A small random offset is also applied to the angle, so the UAVs are not evenly spaced along the circle. Finally, random offsets in the z-axis introduce variation in vertical positioning.

Let R_{init} , R_D , and O be the base radius for the initial position, the base radius for the destination, and the origin of the base circle to deploy drones. We introduce the randomness factor δR , $\delta\theta$, and δZ for the variation of radius, angle, and height, respectively. Here, the initial position for the i^{th} drone can be expressed as follow:

$$\begin{aligned} x_{i,init} &= O_x + R_i \cdot \cos(\theta_{i,init}) \\ y_{i,init} &= O_y + R_i \cdot \sin(\theta_{i,init}) - R \\ z_{i,init} &= O_z + \text{random}(-\delta Z, \delta Z) \end{aligned}$$

where the random radius and angle offset are represented as

$$\begin{aligned} R_i &= R_{init} + \text{random}(-\delta R, \delta R) \\ \theta_{i,init} &= \theta_i + \text{random}(-\delta\theta, \delta\theta) \end{aligned}$$

and $\theta_i = \frac{2\pi i}{N} + \frac{\pi}{2}$ is the base angle of each drone and N is the total number of drones. Similarly, the random destination of each drone i can be expressed as follows:

$$\begin{aligned} x_{i,dest} &= O_x + R_{i,dest} \cdot \cos(\theta_{i,dest}) \\ y_{i,dest} &= O_y + R_{i,dest} \cdot \sin(\theta_{i,dest}) - R_D \\ z_{i,dest} &= O_z + \text{random}(-\delta Z, \delta Z) \end{aligned}$$

with the following random radius and angle offset for the destination:

$$\begin{aligned} R_{i,dest} &= R_D + \text{random}(-\delta R, \delta R) \\ \theta_{i,dest} &= \theta_{i,init} + \frac{\pi}{2} + \text{random}(-\delta\theta, \delta\theta) \end{aligned}$$

1. Training setup

For every training and evaluating episode, we random a scenario of 20 drones with $R = 4$ m, $R_D = 2$ m, $\delta R = 1$ m, $\delta Z = 0.2$ m. Among these, only the agent-controlled UAV can choose different actions, while the other UAVs follow predefined routes without collision avoidance capabilities. An example of a training scenario is shown in Fig.7.

Training is conducted in three sessions, each lasting 60 minutes of wall-clock time. After each session, the model is evaluated over 100 episodes. The model achieving the highest evaluation score across all three training sessions is

selected as the final model for the agent. The training is performed on a Lenovo ThinkPad P1 laptop with 32 GB of memory, an 8 GB NVIDIA GeForce RTX 4060 laptop GPU, and a 13th-generation Intel Core i7 processor.

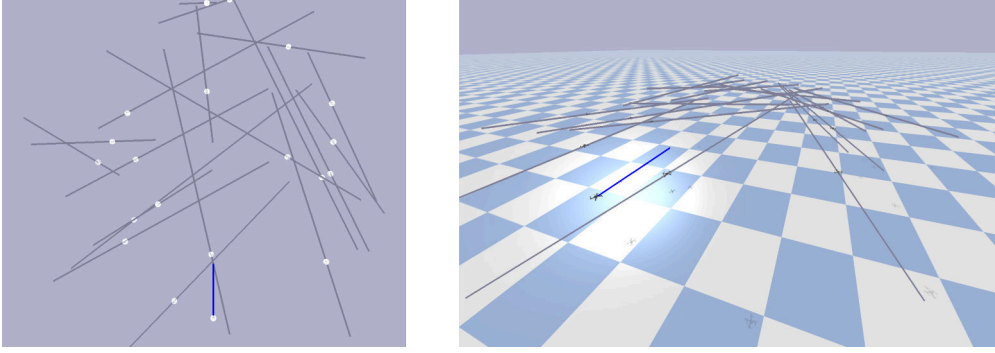


Fig. 7 Random scenario for training with 20 drones with $R = 4\text{ m}$, $R_D = 2\text{ m}$, $\delta R = 1\text{ m}$, $\delta Z = 0.2\text{ m}$

2. Training configuration Trade off

Several key parameters influence the training process and require careful consideration of trade-offs between computational efficiency and scenario complexity:

- **Total number of UAVs:** Increasing the number of UAVs leads to denser airspace with more diverse scenarios and a higher risk of collisions. While this enriches the training environment, it also requires greater computational costs and longer training times as the agent must solve more complex problems.
- **Number of rays:** Reducing the number of rays decreases computational overhead and speeds up the training process. However, fewer rays increase the blind spots of the UAVs, which compromise their ability to detect and avoid smaller obstacles.
- **Mission area:** A smaller mission area leads to higher UAV density, increasing the likelihood of collisions. This challenges the agent to make rapid decisions to avoid obstacles while maintaining mission efficiency.

C. Testing Scenario

This study compares the proposed autonomous path selection method to a traditional trajectory planning approach, specifically the Interfered Fluid Dynamical System (IFDS) algorithm. The test scenario incorporates a combination of static and dynamic obstacles, as illustrated in Fig. 8. The dynamic obstacles, represented by spheres, move back and forth along the x-axis, perpendicular to the UAVs' heading direction along the y-axis.

For the IFDS method, the algorithm activates only when the UAV detects an obstacle; otherwise, it follows the shortest path to its destination. In contrast, the AI-based method enables the agent to dynamically select between global paths, local paths, and speed adjustments at any time, regardless of obstacle detection.

Preliminary results indicate that the IFDS algorithm and the proposed AI-based method successfully resolve collision avoidance challenges in this environment. The scenario will be tested across 100 episodes to evaluate the two approaches comprehensively. This benchmarking highlights each method's relative performance, strengths, and limitations, offering insights into their suitability for real-world applications.

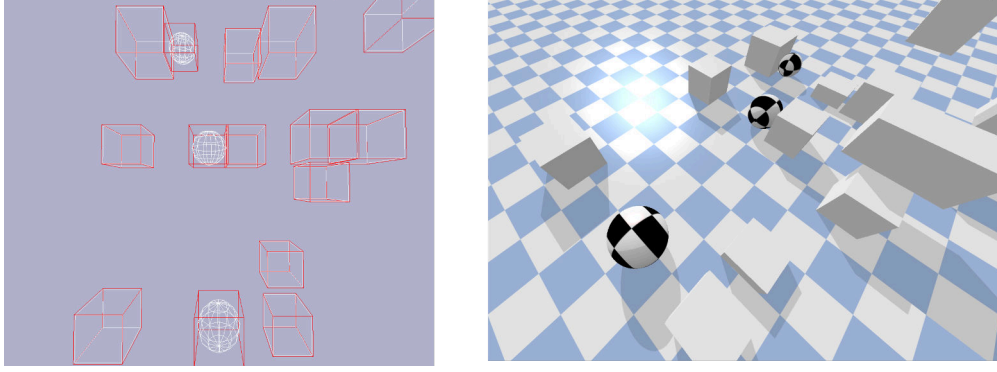


Fig. 8 Testing scenario: cubes represent static obstacles, and spheres represent dynamic obstacles

V. Results and Discussion

An example of a successful trial for the trained autonomous path selection agent (AI-based method) and the IFDS method is shown in Figs.9 and 10. These figures illustrate each method's response to dynamic obstacles, while detailed trajectory metrics are provided in Table 6. While both methods successfully avoid all obstacles, the AI-based method demonstrates a more efficient flight path with reduced deviation. From Fig.10, it can be observed that the AI agent adheres to the global path most of the time, only temporarily switching to a local path to avoid obstacles.

Figure 11 compares both methods' trajectory deviations in the X and Z directions. The AI-based method primarily adjusts in the X-direction, whereas the IFDS method relies more heavily on adjustments in the Z-direction. This results in a significant reduction in the airspace volume occupied by the AI-based trajectory, at $0.59 m^3$, which represents a 61.18 % reduction compared to the IFDS method, which occupies $1.52 m^3$.

Consequently, the AI-based method achieves a shorter overall travel distance, with an average of $10.63 m$ compared to $10.95 m$ for the IFDS method, as shown in Fig. 12.

Despite these advantages, the AI-based method required slightly more time to reach the destination, averaging 585 steps compared to 515 steps for the IFDS method. Since the simulation runs at 60 Hz, this 70-step difference corresponds to an additional 1.167 seconds. While this delay may seem negligible, further investigation is needed to assess its impact in larger-scale scenarios with more dynamic obstacles.

Table 6 Performance comparison of IFDS and AI-Based path selection methods over 100 testing episodes

Parameter	IFDS method	AI-based method
Average total time steps	515 steps	585 steps
Average flight path length	$10.95 m$	$10.63 m$
Flight path span along X-axis	$[-0.08, 0.08] m$	$[-0.14, 0.12] m$
Flight path span along Z-axis	$[1.00, 1.85] m$	$[0.96, 1.16] m$
Volume occupied by UAV trajectory	$1.52 m^3$	$0.59 m^2$

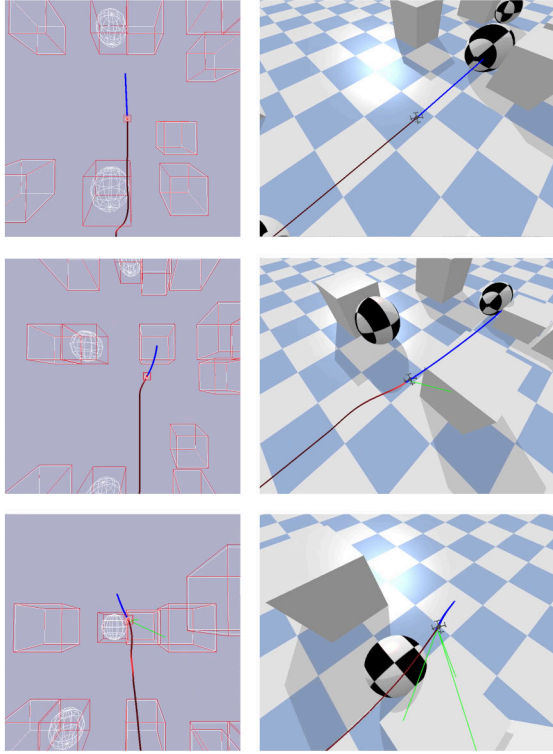


Fig. 9 The IFDS response to three dynamic obstacles

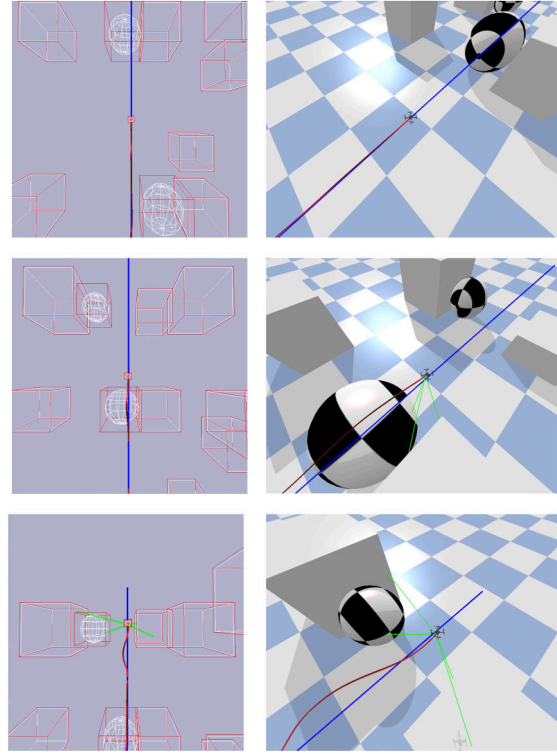


Fig. 10 The AI-based response to three dynamic obstacles

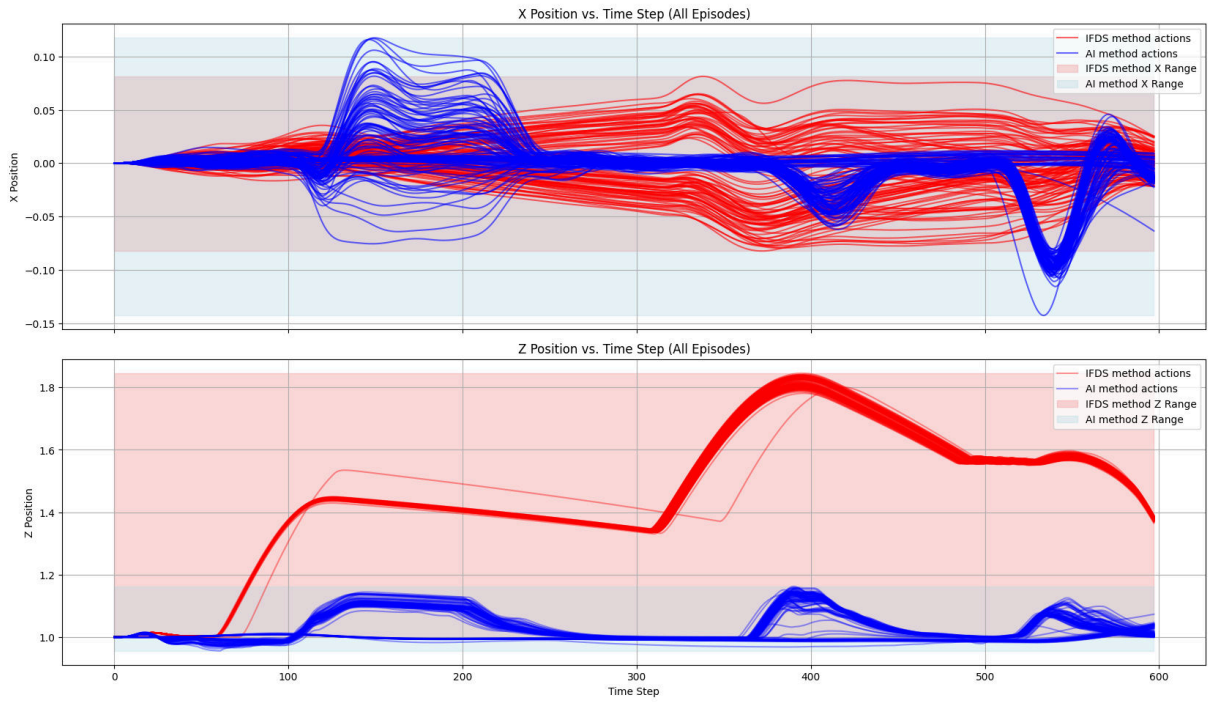


Fig. 11 Flying route deviation across 100 testing episodes

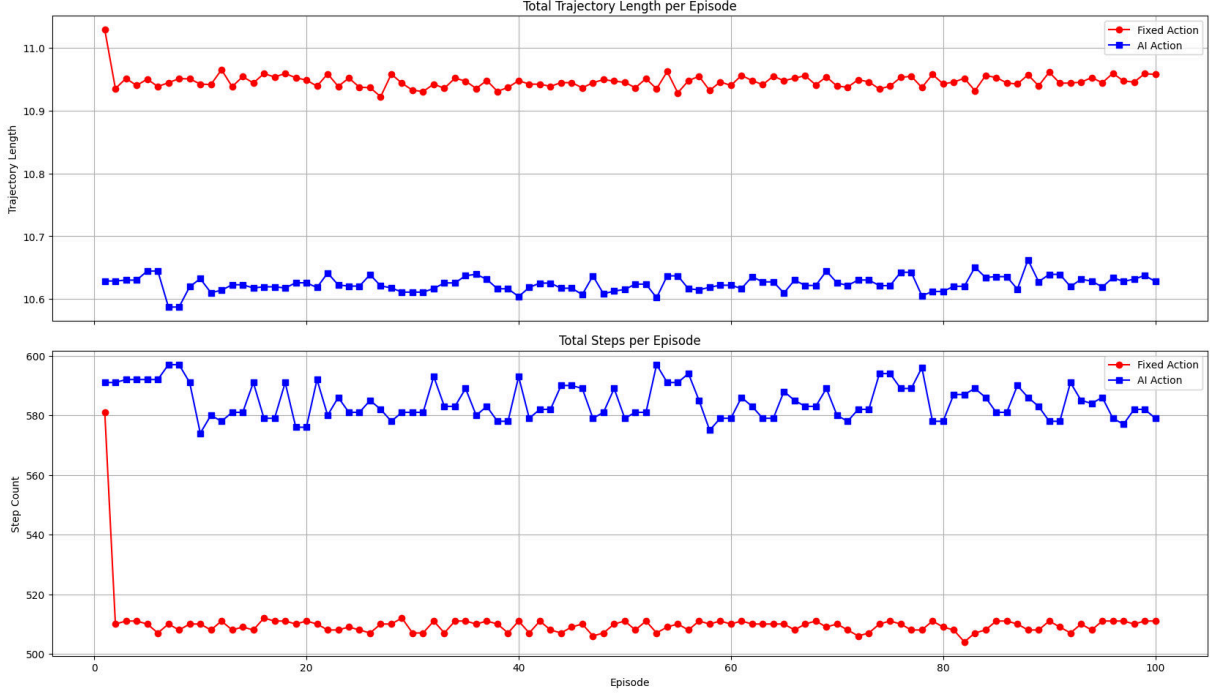


Fig. 12 Traveling distance and flight time across 100 testing episodes

A. Training performance

Figure 13 shows the moving average reward for both training and evaluation, which increases and converges to -43.18 ± 5.40 . The evaluation plot follows a similar pattern, suggesting that the agent’s performance is stable and consistent over time. It is noticed that the reward reaches a stable value after approximately 40 episodes.

An important observation is that the wall clock time is approximately twice the actual training time. This computational overhead arises from the evaluation process, which runs 100 episodes after each training phase to assess the agent’s performance. These evaluation episodes are critical for validating the model’s learned policy and ensuring that it generalizes well beyond the training set. The relatively high computational cost associated with the evaluation process could be a limitation in scenarios requiring fast training cycles, especially as the number of training episodes or complexity of the environment increases.

VI. Conclusion

This paper introduces a novel approach to collision-free navigation for unmanned aerial vehicles (UAVs), inspired by the concept of autonomous lane-changing in cars. The problem is framed as a sequence of path selections, each with an appropriate speed, where the UAV can choose between one global path and twelve local paths.

We employ the Dueling Deep Double Q-Network (Dueling DDQN) to automate high-level decision-making while relying on established algorithms for low-level tasks. Specifically, we use the Interfered Fluid Dynamical System (IFDS) algorithm for path planning and a PID controller for flight control. The reward function encourages the UAV to reach its destination quickly while avoiding collisions. Additionally, the reward function favours using the global path, aligning with safety regulations typically required in real-world scenarios. Training involves generating 20 randomly routed UAVs in a 3D environment for each episode, while testing incorporates both static and dynamic obstacles within the same space.

The results show that the proposed autonomous path selection method not only reduces the overall flight distance but also significantly minimizes the occupied airspace volume, achieving a 61.18 % reduction compared to the IFDS method alone. However, the AI-based approach took 515 more steps, or 1.167 additional seconds, to reach the destination. While this delay may appear insignificant, further testing in larger-scale missions with more complex scenarios is necessary to assess the performance and feasibility of the method in real-world applications.

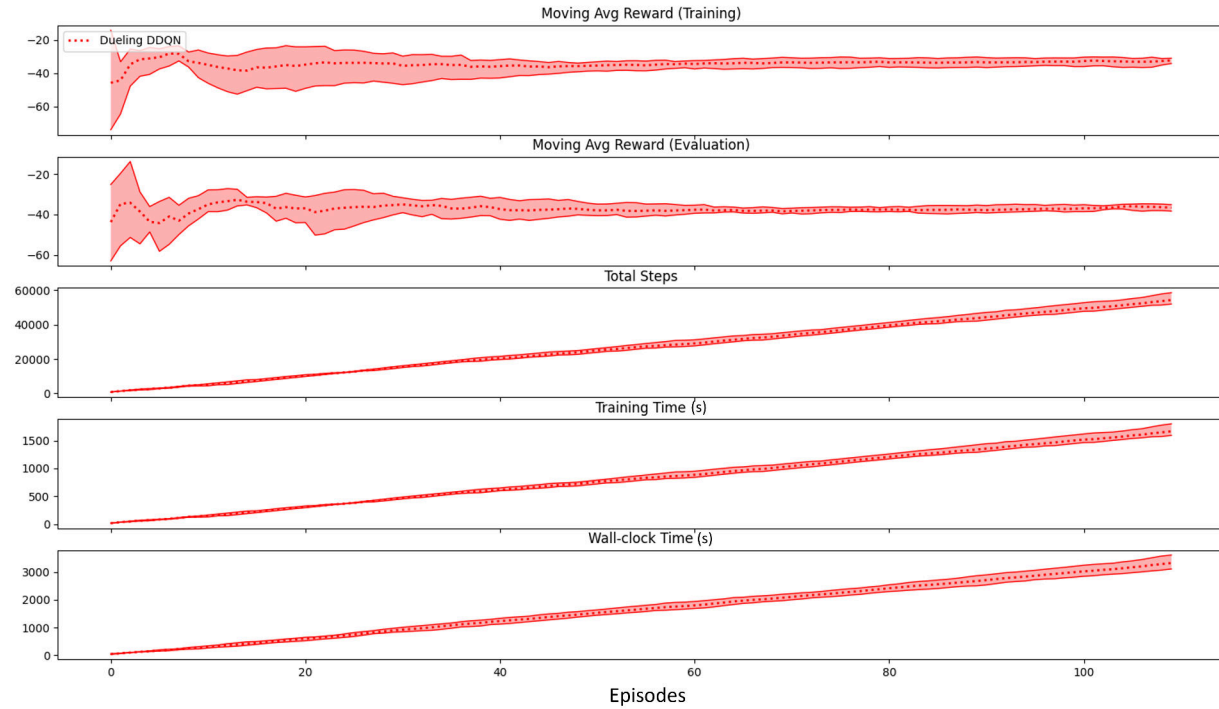


Fig. 13 Training performance of the Dueling DDQN

References

- [1] Wang, H., Lyu, W., Yao, P., Liang, X., and Liu, C., "Three-dimensional path planning for unmanned aerial vehicle based on interfered fluid dynamical system," *Chinese Journal of Aeronautics*, Vol. 28, 2015, pp. 229–239. <https://doi.org/10.1016/J.CJA.2014.12.031>.
- [2] Puente-Castro, A., Rivero, D., Pazos, A., and Fernandez-Blanco, E., "A review of artificial intelligence applied to path planning in UAV swarms," *Neural Computing and Applications*, Vol. 34, No. 1, 2022, pp. 153–170.
- [3] Razzaghi, P., Tabrizian, A., Guo, W., Chen, S., Taye, A., Thompson, E., Bregeon, A., Baheri, A., and Wei, P., "A Survey on Reinforcement Learning in Aviation Applications," , 2022.
- [4] Yan, C., Xiang, X., and Wang, C., "Towards real-time path planning through deep reinforcement learning for a UAV in dynamic environments," *Journal of Intelligent & Robotic Systems*, Vol. 98, 2020, pp. 297–309.
- [5] Brittain, M., Yang, X., and Wei, P., "A Deep Multi-Agent Reinforcement Learning Approach to Autonomous Separation Assurance," , 2020.
- [6] Brittain, M. W., and Wei, P., *One to Any: Distributed Conflict Resolution with Deep Multi-Agent Reinforcement Learning and Long Short-Term Memory*, ??? <https://doi.org/10.2514/6.2021-1952>, URL <https://arc.aiaa.org/doi/abs/10.2514/6.2021-1952>.
- [7] Brittain, M., and Wei, P., "Scalable autonomous separation assurance with heterogeneous multi-agent reinforcement learning," *IEEE Transactions on automation science and engineering*, Vol. 19, No. 4, 2022, pp. 2837–2848.
- [8] Guo, T., Jiang, N., Li, B., Zhu, X., Wang, Y., and Du, W., "UAV navigation in high dynamic environments: A deep reinforcement learning approach," *Chinese Journal of Aeronautics*, Vol. 34, No. 2, 2021, pp. 479–489. <https://doi.org/https://doi.org/10.1016/j.cja.2020.05.011>, URL <https://www.sciencedirect.com/science/article/pii/S1000936120302247>.
- [9] Du, H., You, M., and Zhao, X., "Globally Guided Deep V-Network-Based Motion Planning Algorithm for Fixed-Wing Unmanned Aerial Vehicles," *Sensors*, Vol. 24, No. 12, 2024, p. 3984.
- [10] Haydari, A., and Yilmaz, Y., "Deep Reinforcement Learning for Intelligent Transportation Systems: A Survey," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 23, No. 1, 2022, pp. 11–32. <https://doi.org/10.1109/TITS.2020.3008612>.

- [11] Sallab, A. E., Abdou, M., Perot, E., and Yogamani, S., “Deep Reinforcement Learning framework for Autonomous Driving,” *Electronic Imaging*, Vol. 29, No. 19, 2017, p. 70–76. <https://doi.org/10.2352/issn.2470-1173.2017.19.avm-023>, URL <http://dx.doi.org/10.2352/ISSN.2470-1173.2017.19.AVM-023>.
- [12] Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Sallab, A. A. A., Yogamani, S., and Pérez, P., “Deep Reinforcement Learning for Autonomous Driving: A Survey,” *IEEE Transactions on Intelligent Transportation Systems*, Vol. 23, No. 6, 2022, pp. 4909–4926. <https://doi.org/10.1109/TITS.2021.3054625>.
- [13] Tamanakijprasart, K., Mondal, S., and Tsourdos, A., *Dynamic Path Planning of UAV in Three-dimensional Complex Environment Based on Interfered Fluid Dynamical System*, 2024. <https://doi.org/10.2514/6.2024-2091>, URL <https://arc.aiaa.org/doi/abs/10.2514/6.2024-2091>.
- [14] Liu, S., Li, L., Tang, J., Wu, S., and Gaudiot, J.-L., *Creating autonomous vehicle systems*, Springer, 2018.
- [15] Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., and de Freitas, N., “Dueling Network Architectures for Deep Reinforcement Learning,” , 2016. URL <https://arxiv.org/abs/1511.06581>.
- [16] Wang, Y., Wang, H., Wen, J., Lun, Y., and Wu, J., “Obstacle Avoidance of UAV Based on Neural Networks and Interfered Fluid Dynamical System,” *2020 3rd International Conference on Unmanned Systems (ICUS)*, 2020, pp. 1066–1071. <https://doi.org/10.1109/ICUS50048.2020.9274988>, URL <https://ieeexplore.ieee.org/document/9274988>.
- [17] Zhang, Y., and Wang, H., “Adaptive Interfered Fluid Dynamic System Algorithm Based on Deep Reinforcement Learning Framework,” *Proceedings of 2021 International Conference on Autonomous Unmanned Systems (ICAUS 2021)*, edited by M. Wu, Y. Niu, M. Gu, and J. Cheng, Springer Singapore, Singapore, 2022, pp. 1388–1397.
- [18] Li, S., Jia, Y., Yang, F., Qin, Q., Gao, H., and Zhou, Y., “Collaborative Decision-Making Method for Multi-UAV Based on Multiagent Reinforcement Learning,” *IEEE Access*, Vol. 10, 2022, pp. 91385–91396. <https://doi.org/10.1109/ACCESS.2022.3199070>.
- [19] Zhao, Y., Guo, J., Bai, C., and Zheng, H., “Reinforcement learning-based collision avoidance guidance algorithm for fixed-wing uavs,” *Complexity*, Vol. 2021, 2021, pp. 1–12.
- [20] van Hasselt, H., Guez, A., and Silver, D., “Deep Reinforcement Learning with Double Q-learning,” , 2015. URL <https://arxiv.org/abs/1509.06461>.
- [21] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al., “Human-level control through deep reinforcement learning,” *nature*, Vol. 518, No. 7540, 2015, pp. 529–533.
- [22] Panerati, J., Zheng, H., Zhou, S., Xu, J., Prorok, A., and Schoellig, A. P., “Learning to Fly – a Gym Environment with PyBullet Physics for Reinforcement Learning of Multi-agent Quadcopter Control,” , 2021. <https://doi.org/10.48550/arXiv.2103.02142>, URL <http://arxiv.org/abs/2103.02142>.
- [23] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W., “OpenAI Gym,” , 2016. URL <https://arxiv.org/abs/1606.01540>.
- [24] Silano, G., and Iannelli, L., “CrazyS: A software-in-the-loop simulation platform for the crazyflie 2.0 nano-Quadcopter,” *Robot Operating System (ROS) The Complete Reference (Volume 4)*, 2020, pp. 81–115.



DSPACE

<https://dspace.org/>

Autonomous path selection of unmanned aerial vehicle in dynamic environment using reinforcement learning

Tamanakijprasart, Komsun; Perrusquía, Adolfo; Mondal, Sabyasachi; Tsourdos, Antonios

2025-01-06

American Institute of Aeronautics and Astronautics (AIAA)

<https://doi.org/10.2514/6.2025-1998>

Tamanakijprasart K, Perrusquia A, Mondal S, Tsourdos A. (2025) Autonomous path selection of unmanned aerial vehicle in dynamic environment using reinforcement learning. In: AIAA SCITECH 2025 Forum, 6-10 January 2025, Orlando, FL, USA. Paper number AIAA 2025-1998