Results in Engineering 30 (2026) 110150

Contents lists available at ScienceDirect

Results in Engineering

journal homepage: www.sciencedirect.com/journal/results-in-engineering

Research paper

A hybrid framework for UAV obstacle avoidance integrating reactive sensing, LiDAR planning, and deep reinforcement learning in real time

Alaa H. Ahmed [a,b,c,*], Henrietta Tomán [a]

[a] Department of Data Science and Visualization, Faculty of Informatics, University of Debrecen, Debrecen, H-4032, Hungary
[b] Doctoral School of Informatics, University of Debrecen, Debrecen, H-4032, Hungary
[c] Department of Information Technology, College of Computer Science and Information Technology, University of Kirkuk, Kirkuk, 30061, Iraq

ARTICLE INFO

Keywords:
Unmanned aerial vehicles
Obstacle avoidance
DQN
LiDAR
Unreal engine
AirSim

ABSTRACT

This work presents LT-DQN, a three-tier, safety-gated obstacle avoidance framework that combines a learned DQN controller, LiDAR-based A* planning, and a lightweight Three-Beam reactive fallback for UAV navigation using only onboard sensing. The hybrid method is evaluated in Unreal Engine with Microsoft AirSim across two environments with static and dynamic obstacles under both single-UAV and multi-UAV settings, and is compared against its individual component baselines. Across the tested configurations, LT-DQN outperforms these baselines by achieving a 100% success rate with zero collisions in both single-UAV and multi-UAV trials, while maintaining favorable safety and efficiency metrics, including a 5.17 ± 2.09 m minimum clearance and a 40.3 ± 6.0 s time-to-goal in the multi-UAV setting at 6 m/s. Real-time feasibility is supported by a low control-cycle time of 2.60 ± 0.31 ms, with occasional overhead during safety-layer switching. Overall, these findings indicate that LT-DQN achieves an effective balance between safety and efficiency, enabling robust UAV navigation in cluttered environments.

1. Introduction

Unmanned Aerial Vehicles (UAVs), also known as drones, have become indispensable across numerous sectors due to their rapid deployment capabilities and adaptable maneuverability. Their ability to operate autonomously without human intervention has expanded their use in the last few years [1]. They have become an essential component of search and rescue (SAR) operations, particularly in disaster situations such as earthquakes, floods, or human-made crises. Since human access is limited in some dangerous places, several national emergency agencies have relied on drones as a core component of their SAR strategies, such as those in Philippines [2]. The deployment of drones also results in a measurable reduction in pilot injuries and fatalities [3].

Despite these advantages, UAVs still face the challenge of autonomous navigation, especially in complex environments filled with obstacles of varying sizes, shapes, and motions. Avoiding these obstacles is considered one of the fundamental requirements for safe UAV navigation [4]. Obstacles can be static, such as trees, walls, and buildings, or dynamic, such as aircraft, birds, or other drones [5]. The dynamic category is more challenging due to its speed and motions.

Complementary to obstacle avoidance is path planning, which identifies the optimal trajectory from the starting point to the target position. It can be performed offline before the flight mission or online during the flight mission, or as a hybrid of both. The process of finding the best path is often influenced by real-world factors such as turbulence, sudden environmental changes, and dynamic obstacles [6].

To address these challenges, drones are commonly equipped with diverse sensors for obstacle avoidance and path planning. There are several types of sensors, some are vision-based sensors such as first-person-view (FPV) cameras; and some are laser beam sensors like LiDAR which can create 3D maps proficiently [7,8]. There are also other sensors which rely on radio waves such as Multiple-Input Multiple-Output Radar (MIMO) [9]; and some rely on sound waves such as ultrasonic sensors [10]. These sensors create 2D or 3D maps that can help in detecting and avoiding obstacles. Nevertheless, each one of these sensors has certain limitations such as sensitivity to lighting, weather conditions, high computational cost, or range constraints. Thus, combining multiple sensors improves reliability and efficiency and this process is known as sensor fusion [11–13]. Meanwhile, machine learning methods such as Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) have been employed with drones to manage complex situations [14,15], but they require large annotated datasets, which are usually difficult to obtain. Alternatively, Reinforcement learning (RL) trains agents to develop optimal policies through a reward-and-penalty

* Corresponding author.
E-mail address: alaa.ahmed@inf.unideb.hu (A.H. Ahmed).

https://doi.org/10.1016/j.rineng.2026.110150
Received 11 December 2025; Received in revised form 13 March 2026; Accepted 17 March 2026
Available online 20 March 2026

Table 1
Comparison of UAV obstacle avoidance methods and safety mechanisms.

Method	Sensor Type	RL	Hybrid Approach	Dynamic Obstacles	Multi-UAV	Simulator	Safety Fallback
Miera et al. [19] (2023)	LiDAR	PPO	Partial	No	No	Gazebo/ROS	No
Sütő et al. [21] (2023)	Remote supervisory	Classic controller	Yes	No	Yes	Custom	Partial
Chao et al. [23] (2024)	Event camera	DQN	No	No	No	AirSim/UE	No
Skarka et al. [18] (2024)	Multi-sensor	RL + Classic	Yes	Partial	No	Custom	Partial
Yu et al. [24] (2025)	Depth Sensor	Double DQN + LSTM	No	No	No	AirSim/UE4	No
Lei et al. [25] (2025)	LiDAR + RGB-D	SAC-P	Yes	Partial	No	Gazebo/ROS	No
AlMahamid et al. [26] (2025)	Camera	DQN + Attention	No	No	No	AirSim/UE	No
LT-DQN (This work)	3-Beams + LiDAR	DQN	Yes (3-layer)	Yes	Yes	AirSim/UE4	Yes (deterministic)

mechanism, enabling them to achieve their required destination without relying on pre-existing training datasets [16].

L. Xu et.al [17] studied the role of deep reinforcement learning (DRL) in the field of path planning and controlling in dynamic environments. They reported specifically that the Deep Q-Network (DQN) variants developed to overcome common practical issues in UAV navigation, such as training instability, Q-value overestimation, and insufficient exploration in clustered environments. Algorithm level improvement is typically paired with careful reward shaping and state design for safety-aware control. The suggestion of the study is that the DQN-style methods remain viable and practical for safe UAV navigation when engineered for stability, reliability, and low latency control. In parallel, W. Skarka et.al [18] highlight the effectiveness of hybrid navigation that couples reinforcement learning techniques with classical planners for autonomous UAV navigation. This type of integration can improve reliability within dynamic environments and limited onboard computational resources. They reported that a purely learning-based controller may suffer from generalization and rare case failures, motivating the use of RL policy with safety fallback to achieve reliable recovery and low-latency control. Recently in 2023, P. Miera et al. [19], integrated LiDAR signals with reinforcement learning policy to navigate a single drone and avoid trees within Gazebo/ROS simulation. They trained the drone to fly 30 m through the forest in a given coordinate path, avoiding all the trees. Their success rate reached 91% in avoiding trees during flying period. For swarm UAV settings, B. Zhao et al [20] used multi-agent reinforcement learning (MARL) to control a large-scale UAV swarm safety. The swarm was modeled as a graph using graph neural network, where each UAV is considered a node connected to nearby UAV or obstacle. This connection helps the UAV to understand nearby obstacles and the local environment. Then, MARL controller learns cooperative behaviors such that moving towards the goal, keeping group motion, and avoiding collisions. Similarly, Á. Sütő et. al, [21] implemented a control and path planning system for multi-drone using baseline controller and filter running for each drone. They used remote supervisory controller to avoid obstacles such that it computed the smallest trajectory deviation to ensure safety and then transmitted this signal to the drones, which introduced communication delays and limited adaptability to dynamic obstacles. Nevertheless, these studies are constrained by simplified environments with single start-goal configuration and having notable limitations such as the absence of dynamic obstacles, which reduces their applicability in real-world conditions. Therefore, the challenges of enabling UAVs to navigate autonomously and safely without crashing with obstacles remain an open problem due to the diversity of obstacle types and varying environment conditions [22]. Table 1 provides a comparative overview of the prior UAV obstacle avoidance approaches and our proposed framework LT-DQN, highlighting sensor types, reinforcement learning components, hybridization, dynamic obstacle handling, multi-UAV capabilities, simulation platform, and safety-fallback mechanism.

Despite significant progress which has been made in RL-based navigation, several fundamental limitations remain. First, most DRL-based approaches including DQN variants and PPO-based controllers, operate with a single learned policy, which can be brittle under distribution shift and rare-event conditions, particularly in cluttered environments. Second, although hybrid approaches enhance robustness by coupling learning with classical planners; however, they still lack deterministic safety fallbacks and show limited capability to handle dynamic obstacles. Third, as summarized in Table 1, many existing works are evaluated in simplified scenarios, such as single mission, static obstacles only, or single-agent settings. Even the studies with swarm settings focused on cooperative control without addressing deterministic recovery/safety mechanism within a unified framework. Crucially, none of the comparable studies jointly address all these dimensions, including safety-oriented multi-UAV navigation which supports static and dynamic obstacles while maintaining low control latency and deterministic recovery strategies. In contrast, the proposed framework LT-DQN introduces a three-tier, safety gated navigation hierarchy that explicitly targets these gaps: 1) DQN policy for adaptive, experience driven control, 2) LiDAR-based planning layer for structured path planning, 3) Three-Beam sensing as a reactive layer for immediate collision avoidance. Importantly, LT-DQN integrates a deterministic safety fallback, and operates without pre-collected dataset or pre-built maps. It enables a robust multi-UAV navigation system among both static and dynamic obstacles. To the best of our knowledge, the combination of adaptive learning, and deterministic safety recovery has not been demonstrated jointly in prior studies. The main contributions of this work are:

- Hybrid safety-gated architecture: This study proposes LT-DQN, a three-tier obstacle-avoidance framework which integrates DQN control, LiDAR-based path planning, and Three-Beam sensing within a deterministic fallback architecture.
- Comprehensive evaluation in cluttered environments: The proposed framework was validated using Unreal Engine/AirSim under static and dynamic obstacles in both single-UAV and multi-UAV settings, demonstrating reliable goal reaching including a 100% success rate with zero collisions in the evaluated configuration.
- Ablation-driven evidence and real-time feasibility: Ablation results show that the three-layer hierarchy provides a superior safety-efficiency trade-off over two-layer variants, including faster target reaching ($\approx$ 45s vs. 49s) while maintaining low control-cycle latency ($\approx 2-3$ ms per cycle), supporting real-time operations. Taken together, these results support LT-DQN as a robust hybrid navigation solution, offering a practical alternative to single-policy DQN and partially hybrid approaches for safety-critical UAV autonomy.

The remainder of this paper is organized as follows. Section 2 presents the proposed methodology, including the environment setup and the sensor types used. Section 3 provides a detailed description of the individual obstacle-avoidance techniques and the LT-DQN framework. Section 4 discusses the experimental setup, evaluation measures, and performance results. Section 5 examines the key findings, implications, and limitations of the study. Finally, Section 6 summarizes the main conclusions and outlines directions for future work.

2. Methodology

For simulation, Unreal Engine was integrated with AirSim, an open-source UAV simulator. This platform enables the creation of multiple

scenarios with varied levels of complexity for training drones [27]. The training process focused on obstacle avoidance and goal reaching while minimizing computational cost. Moreover, simulation provides safe, cost-effective, and risk-free training alternative to real-world training, which is more expensive, time consuming, and risky in terms of drone crashes. During the training phase, Unreal Engine and AirSim were interfaced via their APIs using Python programming language.

2.1. Environment

The experiments were conducted in two distinct environmental settings. The first environment, specifically designed for this study, consists of a realistic green field with multiple walls, trees, birds, and drones. Walls and trees were modeled as static obstacles, and any collision with them was considered a crash. The walls have different shapes and complex structures, containing narrow geometric apertures that pose an increased challenge for drone navigation.

In addition, the environment includes dynamic obstacles including birds and drones, which were implemented as physical actors and configured with collision components. To make their movements realistic, each obstacle was assigned either circling or straight-line motion along spline paths and updated in real time via timeline functions. This ensured continuous and smooth movement of these physical actors, enabling drones to perceive them as dynamic objects. By positioning those dynamic obstacles in strategic locations inside the environment, the required avoidance maneuvers were introduced to test the robustness of the algorithms. Furthermore, the environment included a car object to serve as the target destination, as illustrated in Fig. 1.

The planar operating area is 504 x 504 m, corresponding to 50,400 x 50,400 Unreal Engine units under the default Unreal Engine scale (1 unit = 1 cm). Object dimensions reported below correspond to the Unreal Engine static mesh “Approx Size” in centimeters. The environment contains static obstacles comprising 10 wall blocks (87 x 23 x 112 cm each) and 10 tree meshes (502 × 562 × 813 cm each). It also includes dynamic obstacles consisting of two UAV agents with (98 x 98 x 29 cm each) and three bird agents (43 x 63 x 8 cm each) that move along predefined splines. The navigation target is a single vehicle object with (911 x 407 x 230 cm) dimensions. The UAVs controlled by LT-DQN framework have an approximate physical size of (98 x 98 x 29 cm) each.

As a second testbed, we chose the Landscape Mountain environment to evaluate the robustness of the proposed LT-DQN method on a terrain that is completely different from our custom flat field. The mountain scene contains strong elevation variations, and slopes help in creating more near-collision situations and affect the controller’s reactive switching behavior especially when traveling hills and valleys. Thus, this environment supports testing the method’s generality and robustness to unseen maps under more realistic conditions. Nevertheless, all quantitative evaluations and the reported results in this study were conducted in our custom environment to ensure controlled conditions and fair comparison. The reason for using two distinct environmental setting is to assess generalization and robustness under different conditions, consistent with the prior work in [28], which highlights that systems should handle different environment and changing conditions.

2.1.1. Sensors

The drones were equipped with two types of sensors that are highly flexible and not affected by the lighting conditions.

a) Three-Beam sensors (Type-5 in AirSim): These are time-of-flight (TOF) sensors oriented at 0°, ±90°. Each beam is configured with a detection rang eof 0.1 m to 50 m, providing coarse two-dimensional information about obstacles directly in front of the drone as well as on its lateral sides.
b) LiDAR sensor (Type-6 in AirSim): This sensor operates in conjunction with the A* algorithm to guide the drone safely. The A* algorithm is responsible for computing the path plan from the initial point to the target position [29].

3. Obstacle avoidance techniques

3.1. Three-Beam reactive controller

Three-Beam is a simple reactive, bug-style algorithm which is considered one of the most widely adopted algorithms in obstacle avoidance. It offers several advantages, such that it can operate under different light conditions, since it relies on emitted rays, unlike camera-based sensors. It is also significantly more cost-effective compared to LiDAR sensors.

Each drone is equipped with three single-ray distance sensors installed onboard. The three sensors are configured as:

- Frontal sensor to detect obstacles in front of the drone with yaw = 0°.
- Left sensor measures the distance to obstacles on the left-hand side of the drone with yaw = +90°.
- Right sensor measures the distance to obstacles on the right-hand side of the drone with yaw = −90°.

If an obstacle is detected, then the sensors measure the distance in meters; otherwise, they return ∞ if no obstacle is detected. The process of obstacle avoidance depends on the Frontal sensor initially such that:

First: if the Frontal sensor reads no obstacles, then the drone proceeds directly to the goal.

Second: if the Frontal sensor detects an obstacle closer than the threshold (emergency distance), then the drone avoids the detected obstacle by comparing left distance and right distance to strafe to the best direction offering greater clearance. This process continues in a fixed step until the drone reaches the goal threshold. Algorithm 1 explains the procedure in more detail.

Algorithm 1 Three-Beam (f, l, r).

Take off the UAV
for $j = 1$ to N_{steps} **do**
  Read position (x, y, z); goal (g_x, g_y)
  $dx \leftarrow g_x - x,\ dy \leftarrow g_y - y,\ D \leftarrow \sqrt{dx^2 + dy^2}$
  if $D < d_{\text{goal}}$ **then**
    Land and **exit**
  else
    Read beams: $f \leftarrow$ DistanceFront(), $l \leftarrow$ DistanceLeft(), $r \leftarrow$ DistanceRight()
    if $f < d_{\text{emg}}$ **then**
      if $l > r$ **then**
        Strafe left: $(v_x \leftarrow 0,\ v_y \leftarrow +v)$ for step_duration
      else
        Strafe right: $(v_x \leftarrow 0,\ v_y \leftarrow -v)$ for step_duration
      end if
    else
      $\psi \leftarrow \text{atan2}(dy, dx)$; rotate to ψ
      Fly forward: $(v_x \leftarrow +v,\ v_y \leftarrow 0)$ for step_duration
    end if
  end if
end for
Reach the goal and land

Where d_{emg} defines the emergency threshold, and d_{goal} defines the goal threshold.

In conclusion, the algorithm continuously reads the front, left, and right sensors in each control step. Based on this input, it decides whether to move forward towards the goal or strafe right or left, choosing the direction with the greatest available clearance. This process ends when the drone reaches the goal threshold and initiates landing.

An important advantage of the Three-Beam is that it requires neither mapping nor path planning, the entire process relies solely on the reading of the three sensors. Moreover, it remains effective even under low-lighting conditions, since it depends on direct sensor output rather

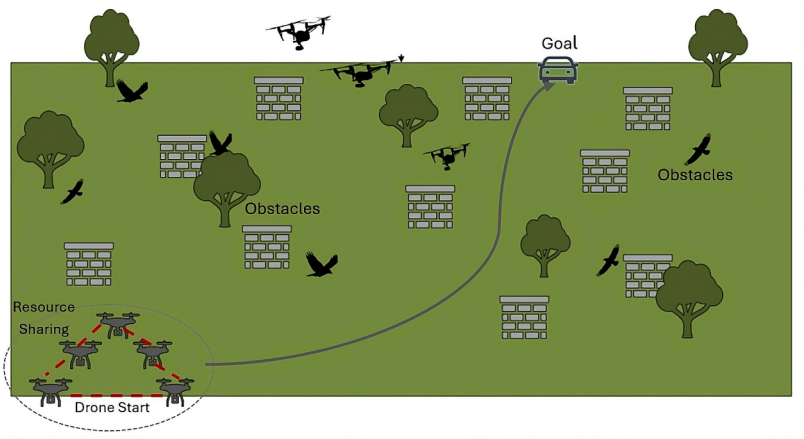


Fig. 1. Illustration of drones' path to the target avoiding the obstacles. Trees and walls are static obstacles, whereas other drones and birds are dynamic obstacles.

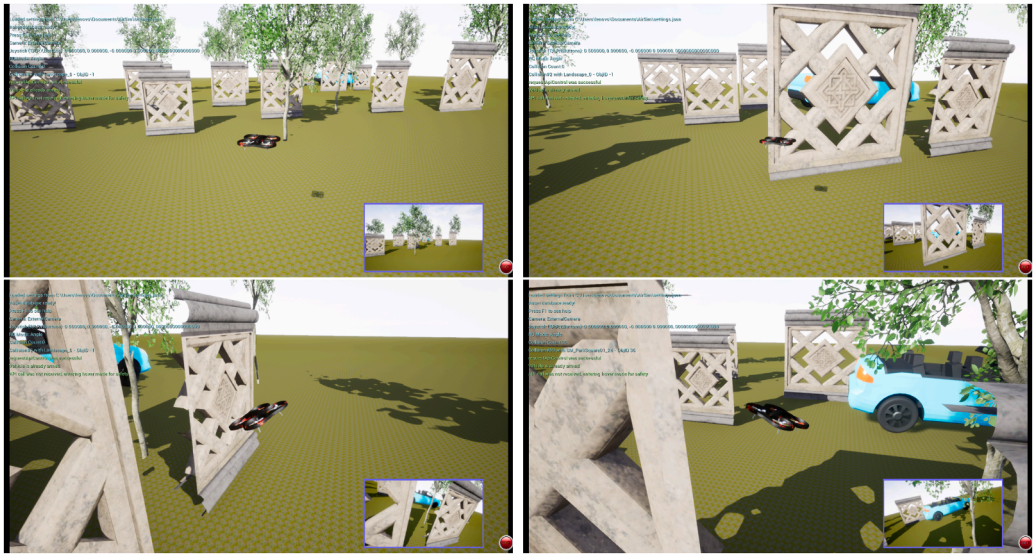


Fig. 2. Illustration of a drone avoiding obstacles and navigating towards the target.

than visual input. The process of Three-Beam is demonstrated in Fig. 2. In this example, the drone detects the wall in its path, and selects the left direction based on the side with more available space. Thus, it avoids collision with the wall and continues heading toward the target.

3.2. LiDAR with A* algorithm

Light detection and ranging (LiDAR) has emerged as one of the most effective and reliable sensing modalities for obstacle avoidance. It uses laser pulses to measure distances to the nearest objects, creating three-dimensional models of the surrounding environment. LiDAR is independent of lighting conditions and is widely used in autonomous aerial and robotic navigation [30]. In this study, the LiDAR sensor was integrated with A* algorithm to perform real-time path planning. LiDAR sweep was filtered to a narrow altitude band around the drone, then converted into a two-dimensional grid at fixed resolution. This grid was then processed by the A* planner which generated a collision-free path towards the goal. At each control point, the UAV updated its position and recomputed the distance to the target. At the same time, the grid was refreshed to ensure safe navigation since the environment is dynamic. Accordingly, if A* planner returned a valid path, then the drone chooses the best yaw and moves towards the target; otherwise, the drone hovers and reinitializes the occupancy grid again until a valid path is found without colliding with the obstacles. The detailed steps of LiDAR with A* are illustrated in Algorithm 2.

3.3. Deep reinforcement learning

Reinforcement learning (RL) is a subset of machine learning (ML), which enables agents to learn through interaction with the environment rather than relying on pre-collected datasets [31]. It gains knowledge through a trial-and-error process in a dynamic environment [32], where autonomous agents make decisions independently without human intervention and that makes it well-suited for obstacle avoidance tasks. Among various RL algorithms, Deep Q-Networks (DQN) have gained prominence for its effectiveness in addressing real-world challenges such as robotics, UAV navigation, and gaming. In Roghair et al. [33], DQN was applied to increase the flight performance of drones by minimizing collisions in a three-dimensional environment. While the work in [34] demonstrates that integrating sensors into drones improves DQN performance in avoiding obstacles. Reinforcement learning enables agents to adapt to their surroundings in a manner analogous to human learning, which allows them to handle complex and uncertain tasks.

3.3.1. Deep Q-network

Deep Q-Network (DQN) is an extended version of Q-learning algorithm, designed to handle continuous state-space environments and

Algorithm 2 LiDAR with A* path planning.

Take off the UAV
Initialize a 2D occupancy grid at the UAV start position
for $i = 1$ to N_{steps} **do**
  Read UAV position (x, y, z) and goal position (g_x, g_y)
  $dx \leftarrow g_x - x, \quad dy \leftarrow g_y - y$
  $D \leftarrow \sqrt{dx^2 + dy^2}$
  if $D < d_{\text{goal}}$ **then**
    Stop and land
    Exit
  end if
  Get the LiDAR scan (point cloud)
  if LiDAR finds enough points ($P \geq N_{\text{min}}$) **then**
    Clear the grid map
    Choose the points that are near to the drone's height
    Mark occupied cells in grid
    Run A* planner to goal on grid
    if A* returns valid next waypoint **then**
      Compute yaw $\psi \leftarrow \text{atan2}(dy, dx)$
      Rotate UAV to yaw ψ
      Move forward one step toward waypoint
    else
      Hover and reinitialize grid
    end if
  else
    Hover and reinitialize grid
  end if
end for
Reach the goal and land

Where P defines LiDAR point cloud, and N_{min} defines minimum liDAR points

represent them using a deep neural network. It takes raw inputs from sensors or cameras and turns them into a set of Q-values, one for each possible action. Algorithm 3 explains the process of DQN in more detail.

In the beginning, the environment is initialized along with two neural networks Q_{online} and Q_{target}. Q_{online} is used for selecting actions, while Q_{target} is used to provide stable training updates. For each flight episode, initial state s was observed and the agent selects an action using an ϵ-greedy policy. The environment then provides the next state s', the reward r, and the done flag. The experience is then stored in replay buffer for the training process.

Subsequently, updating Q_{network} takes place using these samples, where the state is replaced by $s \leftarrow s'$, and the rewards are accumulated over the episode. For the reward, the system assigns the maximum reward when the goal is reached. Additional rewards are given for moving closer to the target and for staying safe distance from obstacles. For the penalties, the system assigns them for crashing and for flying too close to obstacles.

In this study, vanilla DQN is adopted as a lightweight value-based baseline because the primary contribution is the proposed hybrid, safety-gated LT-DQN architecture. Vanilla DQN is well suited to the considered UAV navigation problem, which involves a small discrete action set (forward, left, right, rotate-to-goal), enabling efficient real-time inference via a single forward pass. Although modern RL models (e.g., Double /Dueling DQN) and actor-critic methods (e.g., PPO/SAC) can improve sample efficiency and stability, they generally introduce additional complexity and tuning. Accordingly, vanilla DQN was adopted as a simple, reproducible experimental backbone. This baseline policy is used as the nominal control layer within the LT-DQN framework described in the Section 3.4.

Algorithm 3 Deep Q-network (DQN).

Initialize environment; initialize Q_{online} and Q_{target}
Initialize replay buffer D
for episode = 1 to N_{episodes} **do**
  Reset environment and take off UAV
  Observe initial state s
  $R_{\text{total}} \leftarrow 0$
  for step = 1 to N_{steps} **do**
    if uniform$(0, 1) < \epsilon$ **then**
      Choose random action a
    else
      $a \leftarrow \arg\max_a Q_{\text{online}}(s, a)$
    end if
    Execute a; observe s', reward r, done flag
    Store $(s, a, r, s', \text{done})$ in D
    Sample minibatch from D
    for each $(s, a, r, s', \text{done})$ in minibatch **do**
      if done **then**
        $y \leftarrow r$
      else
        $y \leftarrow r + \gamma \max_{a'} Q_{\text{target}}(s', a')$
      end if
      $\mathcal{L} \leftarrow \left(Q_{\text{online}}(s, a) - y\right)^2$
      Update Q_{online} by minimizing $\mathcal{L}$
    end for
    $s \leftarrow s', \quad R_{\text{total}} \leftarrow R_{\text{total}} + r$
    if done **then**
      break
    end if
  end for
  Decay ϵ (exploration → exploitation)
  if episode mod $N_{\text{update}} = 0$ **then**
    $Q_{\text{target}} \leftarrow Q_{\text{online}}$
  end if
end for

Algorithm 4 LT-DQN hybrid navigation.

Initialize environment, Q_{online}, Q_{target}, and LiDAR mapper
Take off the UAV
for $i = 1$ to N_{steps} **do**
  Read UAV position (x, y, z) and goal (g_x, g_y)
  $dx \leftarrow g_x - x, \quad dy \leftarrow g_y - y, \quad D \leftarrow \sqrt{dx^2 + dy^2}$
  if $D < d_{\text{goal}}$ **then**
    Stop and land
    Exit
  end if
  Read frontal sensor distances (f, l, r)
  Compute clearance $c \leftarrow \min(f, l, r)$
  if $c < d_{\text{brake}}$ **then**
    Set $(v_x, v_y) \leftarrow (0, 0)$ for Δt; **continue**
  end if
  if $c > d_{\text{emg}}$ **then**
    Use DQN: follow Algorithm 3
    continue
  end if
  Obtain LiDAR point cloud P
  if $|P| \geq N_{\text{min}}$ **then**
    Use LiDAR + A*: follow Algorithm 2
    continue
  end if
  Use Three-Beam: follow Algorithm 1
end for
Land and exit

Where d_{brake} defines the brake threshold.

3.3.2. Model training

The steps used for training the Deep Q-Network to lead the drone from starting point to the required destination are as follows:-

- State: $s = [f, l, r, dx, dy]$, where f, l, r represent front, left, right distance-sensor reading, while dx, dy represent the coordinator of the distance to the target.
- Q-network architecture: fully connected input layer (5-64 units), fully connected 2 hidden layers (64-64 units) with ReLU activation function, followed by fully connected output layer (64-4 units).
- Actions: $a = [0, 1, 2, 3]$, where $0, 1, 2, 3$ represent forward, strafe left, strafe right, rotate to the goal, representing the four possible discrete actions.
- ϵ-greedy policy: with probability ϵ choose random actions (exploration), with probability $1 - \epsilon$ choose argmax-Q-values (exploitation).
- Replay buffer: with capacity 5000 with a minibatch size 64
- Loss function: mean squared error (MSE) between the predicted Q-values and the bootstrap target values.
- Optimizer: Adam ($lr = 1 \times 10^{-3}$) adjusts the weights to minimize errors.
- Reward and penalty design: based on the drone's movement, it gets feedback as a reward or punishment after each movement:

The reward function is defined as:

$$r = \begin{cases} +2 & \text{for steps towards the goal} \\ +1 & \text{for being safe from the obstacles} \\ +100 & \text{for reaching the goal} \\ -5 & \text{for flying too close to obstacles} \\ -0.1 & \text{for every step (forcing the drone to choose shortest path)} \\ -100 & \text{for crashing with obstacles} \end{cases}$$

The model was trained for 1000 episodes, with a maximum of 70 steps per episode. At the beginning of training, drones started from their initial state and selected action either randomly in order to explore, or from the learned policy to exploit existing knowledge. After executing each action, the drones observed the resulting state and received a reward or penalty depending on their movement and stored the experience in the memory. The network is then repeatedly updated using samples drawn from memory over many episodes. Over time, the drones learn which actions lead to penalties and which yield higher rewards. The DQN policy was trained only in the custom environment. During the evaluation phase, the learned model parameters were kept fixed (no further updates or fine-tuning), and the trained policy was tested in separate evaluation scenarios featuring different start-goal configurations and dynamic obstacle patterns. The trained policy was then evaluated without random exploration.

3.4. LT-DQN

We introduced a novel framework called LT-DQN for UAV obstacle avoidance. LT-DQN is a three-layer hybrid navigation system that combines three techniques: DQN, LiDAR, and the Three-Beam to improve safety and efficiency. It addresses the limitations of using the components individually by finding the best solution for computational cost, path planning, and the best distance to reach the desired target safely. The proposed algorithm consists of three main layers.

At takeoff, the drone relies on the DQN policy to perform discrete and reactive control under nominal, safe conditions. Specifically, DQN actions are applied only when the measured clearance exceeds the emergency threshold. If the clearance value drops below this threshold, the controller activates the LiDAR + A* module that constructs a local occupancy grid and computes a collision-free motion command using A*. In case there is insufficient point density for LiDAR planning or no valid A* path, then the controller activates Three-Beam reactive layer which includes emergency brake and a lateral sidestep rules. The process of LT-DQN is explained in more detail in Algorithm 4.

The advantage of using LT-DQN lies in its ability to combine the strengths of the three obstacle avoidance algorithms. Each individual method has specific advantages and limitations, especially when implemented in a complex environment. Since the Three-Beam algorithm operates without path planning or prior training, it performs basic obstacle avoidance but often results in many collisions. LiDAR, on the other hand, increases computational cost while planning the path especially when the environment is big and complex filled with dynamic obstacles. DQN works very well with the known environment, but it sometimes crashes when new obstacles are introduced. LT-DQN reduces these limitations by using DQN when there is enough clearance preventing risky actions near obstacles and providing low-latency control.

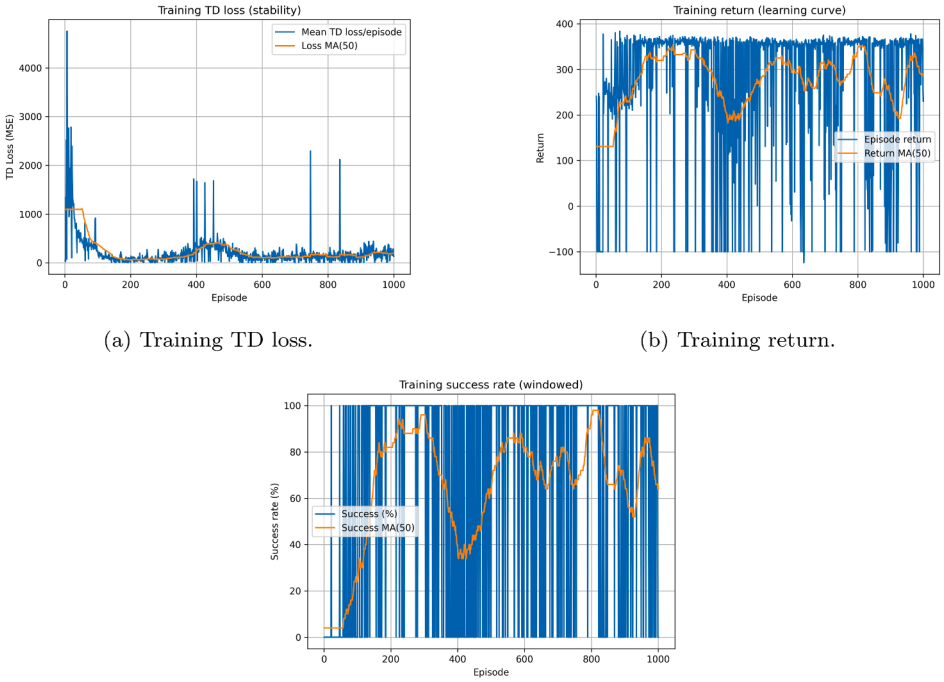
Layer switching criteria in the proposed hybrid framework depend on the clearance value at each control cycle. The controller computes the clearance value by taking the minimum of the Three-Beam distance measurements as $c = \min(f, l, r)$. The switching mechanism operates deterministically via a threshold-based gating policy. When $c > d_{\text{emg}}$, the controller activates the DQN layer to generate a low-latency navigation command toward the target. When $c \leq d_{\text{emg}}$, the controller escalates to the LiDAR with A* layer that constructs a local occupancy grid from the LiDAR point cloud around the current altitude and A* algorithm produces a collision-free direction for the next step towards the goal. To reduce computational cost, the planner restricts its operation to a local area and executes only the next move derived from the A* algorithm. In other terms, the LiDAR + A* layer is triggered only when the clearance value drops below the emergency threshold or when the DQN layer does not provide a valid action. If the LiDAR planning is not feasible because of insufficient point density or no valid A* path, then the Three-Beam layer gets invoked, serving as a final fallback layer. The Three-Beam layer applies an emergency brake for critical proximity and performs a lateral maneuver toward the side with larger clearance. Overall, the proposed LT-DQN framework guarantees reliable and fault-tolerant navigation, enabling the drones to reach their target with optimal safety and efficiency.

4. Experiments and results

We conducted two types of experiments to evaluate four types of obstacle-avoidance techniques (Three-Beam, LiDAR with A* algorithm, DQN, and the proposed hybrid framework (LT-DQN). The experiments were conducted using Unreal Engine, AirSim, and Python to control the APIs. All the simulations were executed on a workstation equipped with a 13th Gen Intel(R), Core (TM) i7-13620H 2.40 GHz processor, with memory size 16.0 GB, and an NVIDIA GEFORCE RTX 4050 6 GB GPU.

As described in Section 2, experiments were conducted in two distinct environments to evaluate robustness and generalization. The first environment was developed specifically for this study and contains diverse obstacle types, including static and dynamic obstacles, supporting controlled and reproducible evaluation under multi-agent interactions. The second environment was a mountain landscape used only to assess generalization of the trained policy.

In both environments, the drones were deployed using Microsoft AirSim, and the initial positions of the drones were determined such that the lead drone was positioned at (x = 0, y = 0, z = 0), while other drones slightly shifted to prevent inter-drone collision along the x and y axis. They were displaced by 5 m along the negative x-axis and ±3 along the y-axis reflecting a V-shaped formation strategy. For example, Drone2's coordinates were (x = -5, y = -3, z = 0) and Drone3's coordinates were (x = -5, y = 3, z = 0). The target's position was changed in each scenario several times to validate the generality and robustness of the algorithms. The essential configuration parameters included a fixed altitude of −6.0 m, an emergency threshold of 4.0 m, a brake threshold of 1.0 m, a goal threshold of 3.0 m, a minimum LiDAR point count of 30, a control frequency of 10 Hz, and a forward velocity of 5.0 or 6.0 m/s. These



(a) Training TD loss.

(b) Training return.

(c) Training success rate (windowed).

Fig. 3. DQN training curves showing TD loss, episode return, and success rate over 1000 episodes.

configurations were applied to achieve the main objective of enabling the drones to reach the required goal within the shortest safe path and avoid all obstacles with maximum clearance.

4.1. Evaluation metrics

The evaluation metrics were determined based on the main objective of this study: enabling the drones to avoid obstacles and reach the target safely while minimizing path length. Each algorithm was implemented to guide the drone toward the target position, and the performance was then assessed using the following metrics:

1. Drone Velocity: defines the drone operating speed which is either 5 m/s or 6 m/s during experimental trials.
2. Success Rate: reflects whether the drone reached the target, with success expressed as 1 and failure as 0.
3. Time to Goal: represents the duration required for the drone to reach the target, measured in seconds.
4. Path Length: the total distance traveled by the drones to reach the goal, measured in meters.
5. Minimum Clearance: the smallest recorded distance between the UAV and any obstacles encountered.
6. Collision Count: the number of collisions that occurred between the drone and the obstacles.
7. Path Planning: the process duration to determine the shortest path from the starting point of the drone to the target, measured in milliseconds.

4.2. Learning curves and convergence

The DQN control layer within the LT-DQN framework was trained for 1000 episodes, with a maximum of 70 steps per episode, using a replay buffer of 5000 transitions and mini-batches of size 64. The learning rate was set to 1×10^{-3}, with an ε-greedy exploration schedule initialized at $\varepsilon = 1.0$, decayed by 0.995 per episode, and clipped at a minimum of 0.05. Fig. 3 shows the learning curves of the DQN control layer, including training return, success rate, and TD loss.

Table 2
Late-training statistics over the last 100 episodes.

Metric	Value
Episode return	290.05 ± 135.64
Success rate (%)	71.0%
TD-loss	204.79 ± 98.68
Return MA50	268.19 ± 49.22
Success MA50	$69.44\% \pm 11.37\%$
TD-loss MA50	184.10 ± 30.41
TD-loss spike count	22 episodes with TD-loss > (global mean + 3σ)

Convergence was assessed via a sustained-plateau criterion based on the 50-episode moving average (MA50) of training return. Convergence was defined as the earliest episode where MA50 achieved at least 95% of the final plateau and remained above that level for 100 consecutive episodes. Convergence occurred at episode 112 (plateau MA50 = 268.19, 95% threshold = 254.78). The MA50 success rate also remained $\geq 70\%$ at episode 150 and $\geq 80\%$ at episode 176.

Late training stability was assessed over the last 100 episodes. Although per-episode returns remain noisy (std. = 135.64), the smoothed MA50 curves are substantially more stable (return MA50 std. = 49.22; TD-loss MA50 std. = 30.41). Rare TD-loss spikes were observed (22 episodes above mean + 3σ), indicating occasional high TD-error updates without persistent divergence. Table 2 summarizes late-training stability over the last 100 episodes using mean ± std.

Rapid improvement in return and success rate, together with stable high-performance plateau, demonstrate the effectiveness of the learning process. Training remains occasionally unstable, with TD-loss spikes, reflecting intermittent high-error updates in partially stochastic non-stationary environments.

4.3. Ablation study

We conducted a set of controlled experiments in both single-UAV and multi-UAV configuration within the custom environment to

Table 3
Results of applying OA techniques in the single-drone setting.

Algorithm	Velocity (m/s)	Success Rate (%)	Time to Goal (s)	Path Length (m)	Minimum Clearance (m)	Collision Count	Path Planning (ms)
Three-Beam	5	100 ± 0 CI90 [100; 100]	63.6 ± 14.4 CI90 [55.3; 71.9]	147.5 ± 39.8 CI90 [124.5; 170.6]	4.2 ± 1.9 CI90 [3.1; 5.3]	3.6 ± 5.8 CI90 [0.2; 7.0]	N/A
	6	80 ± 42 CI90 [56; 104]	55.2 ± 15.7 CI90 [44.7; 65.7]	152.0 ± 52.7 CI90 [116.7; 187.3]	2.6 ± 2.2 CI90 [1.1; 4.0]	21.9 ± 19.9 CI90 [8.5; 35.2]	N/A
LiDAR + A*	5	100 ± 0 CI90 [100; 100]	70.8 ± 15.4 CI90 [61.8; 79.7]	145.1 ± 39.2 CI90 [122.3; 167.8]	3.7 ± 1.0 CI90 [3.1; 4.3]	0.0 ± 0.0 CI90 [0.0; 0.0]	0.02 ± 0.03 CI90 [0.00; 0.04]
	6	90 ± 32 CI90 [72; 108]	60.4 ± 12.9 CI90 [52.4; 68.4]	154.9 ± 47.6 CI90 [125.4; 184.4]	2.8 ± 0.9 CI90 [2.2; 3.3]	11.0 ± 6.1 CI90 [7.2; 14.8]	0.05 ± 0.04 CI90 [0.02; 0.08]
DQN	5	100 ± 0 CI90 [100; 100]	63.2 ± 16.4 CI90 [53.7; 72.7]	136.8 ± 36.2 CI90 [115.8; 157.7]	1.3 ± 0.8 CI90 [0.8; 1.8]	0.0 ± 0.0 CI90 [0.0; 0.0]	N/A
	6	90 ± 32 CI90 [72; 108]	59.8 ± 16.2 CI90 [49.8; 69.9]	155.4 ± 46.7 CI90 [126.5; 184.4]	1.7 ± 0.9 CI90 [1.2; 2.3]	3.3 ± 5.0 CI90 [0.2; 6.4]	N/A
LT-DQN	5	100 ± 0 CI90 [100; 100]	61.9 ± 14.2 CI90 [53.7; 70.1]	137.3 ± 32.8 CI90 [118.3; 156.3]	5.0 ± 1.0 CI90 [4.4; 5.6]	0.0 ± 0.0 CI90 [0.0; 0.0]	0.00 ± 0.00 CI90 [0.00; 0.00]
	6	100 ± 0 CI90 [100; 100]	52.7 ± 12.5 CI90 [45.4; 59.9]	137.3 ± 35.3 CI90 [116.8; 157.7]	6.5 ± 3.4 CI90 [4.6; 8.5]	0.0 ± 0.0 CI90 [0.0; 0.0]	0.00 ± 0.00 CI90 [0.00; 0.00]

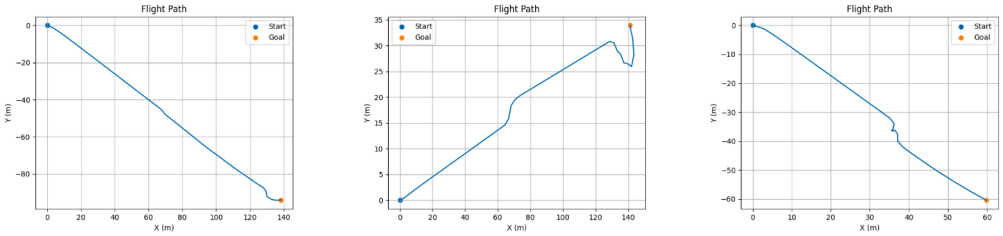


Fig. 4. Experiments using the Three-Beam sensor technique.

evaluate the proposed method against the baselines, analyze component contribution, and assess robustness under varying conditions.

4.3.1. Single-UAV and multi-UAV evaluation

Scenario 1: a single UAV tasked with navigation from a fixed start point to a predefined target by using obstacle avoidance (OA) methods to generate an efficient and safe trajectory. Each method was evaluated across 10 trials at two velocity settings (5 and 6 m/s), using four navigation methods. Results are presented in Table 3 as mean (μ) ± standard deviation (σ) along with 90% confidence intervals (CI) to evaluate the robustness and consistency of performance across repeated trials. The mean, standard deviation, and 90% CI were calculated using the equations below:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{1}$$

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu)^2} \tag{2}$$

$$\mathrm{CI}_{90\%} = \mu \pm t_{0.95,n-1}\frac{\sigma}{\sqrt{n}}. \tag{3}$$

where n denotes the number of repetitions in the study and x_i represents the measured value of the ith experiment.

Figs. 4–7 illustrate the single drone's trajectory under four OA techniques across three different target positions.

Scenario 2: multiple UAVs operated in the same environment and navigated toward a predefined target using the OA methods, which introduces inter-UAV interactions and makes the task more challenging than scenario 1. Each method was evaluated across 10 trials at two velocity settings (5 and 6 m/s), across four navigation methods. Results are presented in Table 4 as mean (μ) ± standard deviation (σ) along with 90% confidence intervals (CI) to quantify performance stability and consistency across repeated trials. In multi-UAV experiments, coordination is limited to the initial deployment: UAVs are spawned with a fixed V-shaped offset to prevent overlap at takeoff. Each UAV runs the same obstacle-avoidance controller independently using only its own onboard sensing. No inter-drone communication is used, and there is no centralized planner or shared policy. In Unreal Engine, dynamic obstacles, such as birds, and other UAVs, have collision enabled and are therefore perceived as moving obstacles, which may trigger LT-DQN layer switching. This setup evaluates the scalability and robustness of the proposed controller under multi-UAV interaction, rather than coordinated formation or cooperative planning.

Tables 3-4, present a comparison and demonstrate a noticeable difference between the three obstacle avoidance strategies (Three-Beam, LiDAR, Deep Reinforcement Learning) and the proposed hybrid framework in a single-drone and multi-drone settings. The results show that the Three-Beam has the weakest reliability and safety, particularly as the speed increases and in the presence of other UAVs. Even though it reaches the goal in some cases, the performance degrades at higher speed. This is because it collides several times and reduces the success rates which indicates limited capability to handle dense obstacle interactions under this purely reactive sensing strategy. LiDAR with A* improves safety compared to Three-Beam because it leverages an explicit planning module; however, collision remains non-zero and the success rate decreases at higher speed especially in the more complex multi-UAV setting. Moreover, LiDAR + A* introduces measurable planning latency because of the computational cost of re-planning, which can reduce responsiveness under more demanding dynamic conditions. DQN, on the other hand, achieves competitive performance in goal-reaching, but also it tends to operate with smaller minimum clearance and can collide under more challenging conditions. This indicates that a single learned policy can be inadequate for consistently safe behavior across all scenarios and speeds.

In contrast, the hybrid LT-DQN achieved the most reliable and safest performance in both single-UAV and multi-UAV setting. It consistently achieves a high success rate with zero collision count across both velocity levels. Moreover, LT-DQN attains the largest clearance among the compared methods, indicating a strong safety margin while remaining efficient in time-to-goal. Lastly, the reported planning time is negligible,

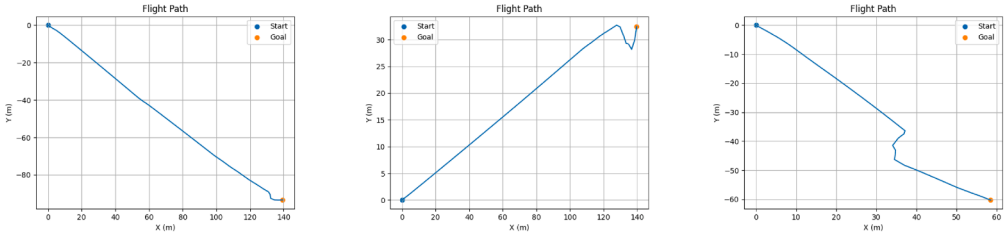


Fig. 5. Experiments using the LiDAR-based planning technique.

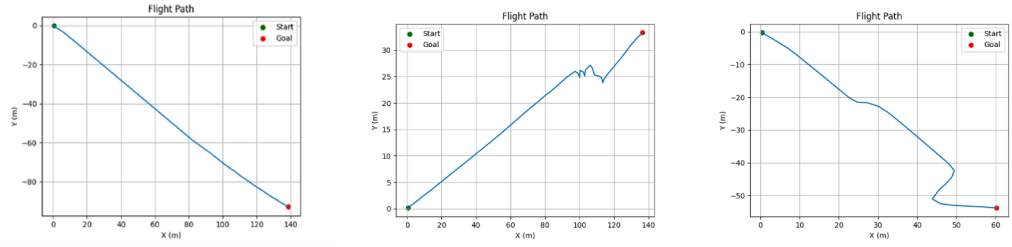


Fig. 6. Experiments using the DQN technique.

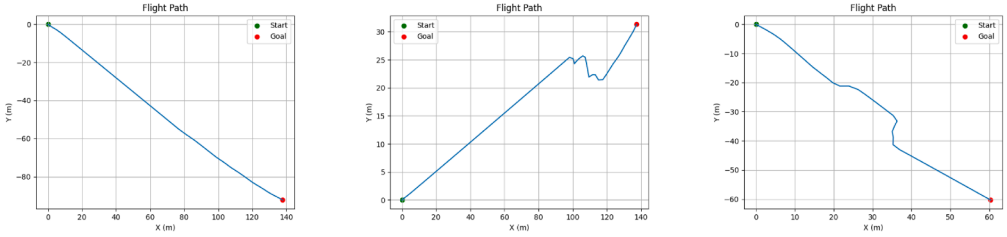


Fig. 7. Experiments using the LT-DQN technique.

Table 4
Results of applying OA techniques in the multi-drone setting.

Algorithm	Velocity (m/s)	Success Rate (%)	Time to Goal (s)	Path Length (m)	Minimum Clearance (m)	Collision Count	Path Planning (ms)
Three-Beam	5	90 ± 31 CI90 [81; 99]	56.1 ± 11.3 CI90 [52.4; 59.8]	121.9 ± 23.7 CI90 [114.1; 129.6]	2.46 ± 1.67 CI90 [1.91; 3.01]	1.4 ± 1.9 CI90 [0.8; 2.0]	N/A
	6	80 ± 41 CI90 [67; 93]	54.8 ± 9.4 CI90 [51.7; 57.9]	123.4 ± 18.1 CI90 [117.5; 129.3]	2.56 ± 1.44 CI90 [2.09; 3.03]	11.2 ± 18.2 CI90 [5.3; 17.2]	N/A
LiDAR + A*	5	100 ± 0 CI90 [100; 100]	58.3 ± 11.0 CI90 [54.9; 61.7]	137.2 ± 30.9 CI90 [127.6; 146.8]	2.56 ± 1.46 CI90 [2.11; 3.02]	0.9 ± 1.9 CI90 [0.4; 1.5]	0.19 ± 0.13 CI90 [0.15; 0.23]
	6	90 ± 31 CI90 [81; 99]	47.9 ± 8.6 CI90 [45.1; 50.7]	133.6 ± 28.9 CI90 [124.1; 143.1]	4.06 ± 2.38 CI90 [3.28; 4.84]	1.7 ± 3.3 CI90 [0.6; 2.8]	0.14 ± 0.13 CI90 [0.09; 0.18]
DQN	5	90 ± 31 CI90 [81; 99]	70.8 ± 18.3 CI90 [64.8; 76.8]	129.1 ± 19.7 CI90 [122.7; 135.6]	2.77 ± 2.30 CI90 [2.01; 3.52]	3.9 ± 12.4 CI90 [-0.1; 8.0]	N/A
	6	100 ± 0 CI90 [100; 100]	54.8 ± 10.8 CI90 [51.5; 58.2]	128.0 ± 18.5 CI90 [122.3; 133.8]	2.26 ± 2.05 CI90 [1.62; 2.89]	0.9 ± 2.9 CI90 [0.0; 1.8]	N/A
LT-DQN	5	100 ± 0 CI90 [100; 100]	49.6 ± 9.3 CI90 [46.7; 52.5]	126.1 ± 20.5 CI90 [119.8; 132.5]	4.33 ± 1.72 CI90 [3.80; 4.87]	0.0 ± 0.0 CI90 [0.0; 0.0]	0.00 ± 0.00 CI90 [0.00; 0.00]
	6	100 ± 0 CI90 [100; 100]	40.3 ± 6.0 CI90 [38.4; 42.2]	126.8 ± 18.4 CI90 [121.1; 132.5]	5.17 ± 2.09 CI90 [4.53; 5.82]	0.0 ± 0.0 CI90 [0.0; 0.0]	0.00 ± 0.01 CI90 [0.00; 0.00]

showing that this safety-efficiency balance is obtained without imposing additional computational burden, which explains its superior robustness and consistency across repeated trials. Figs. 8–11 demonstrate the multi-drone trajectory using four OA techniques across three different target positions.

4.3.2. *Runtime and switching behavior*

An explicit analysis was conducted to quantify the computational delay incurred during switching between LT-DQN control layers. We augmented the controller with high-resolution timing to measure the switching delay directly. In each timestep, time.perf_counter() was used to report compute time of the full decision cycle.

- **No-switch cycle time:** the average computation time per step when no layer change occurs.
- **Switch cycle time:** the average computation time per step when a switch occurs between layers.
- **Switching overhead:** the additional time introduced by switching, defined as

$$\Delta = T_{\text{switch}} - T_{\text{no-switch}}.$$

Table 5 shows the results as mean ± standard deviation using 10 independent trials in a denser obstacle environment with more challenging conditions to assess real time feasibility. The results indicated that LT-DQN remains low latency during steady-state operations, but the

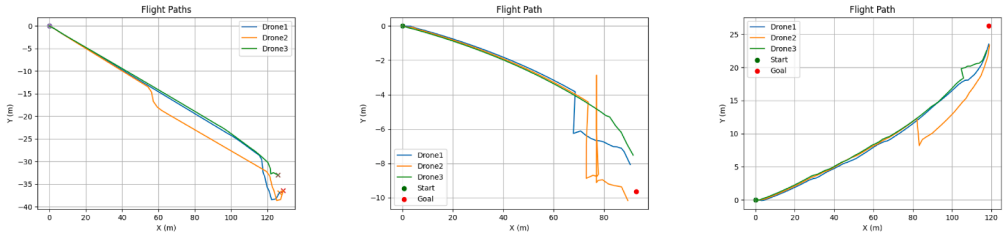


Fig. 8. Experiments using the Three-Beam sensor technique with three drones.

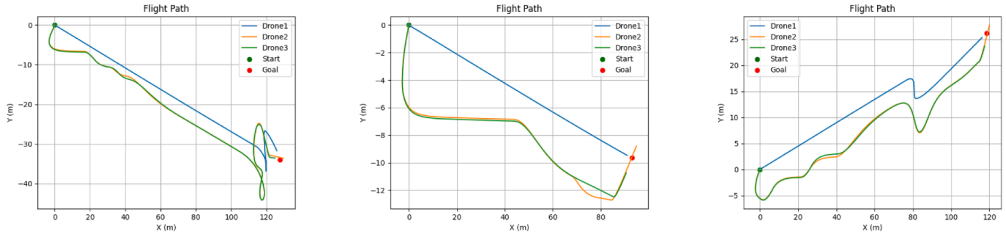


Fig. 9. Experiments using LiDAR-based planning technique with three drones.

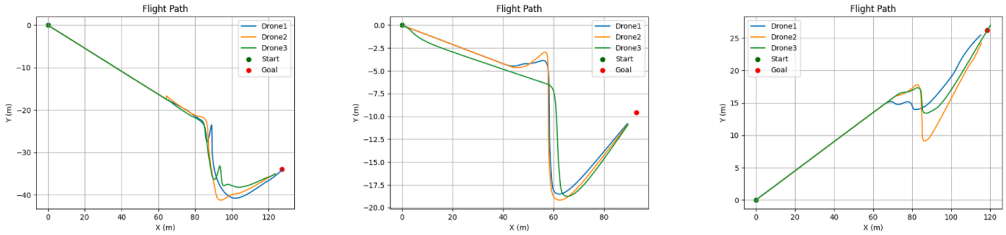


Fig. 10. Experiments using DQN technique with three drones.

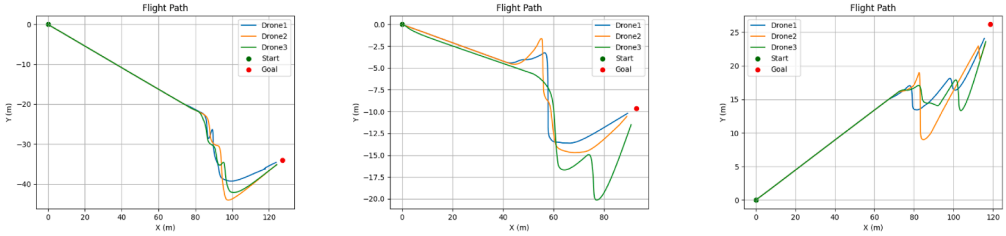


Fig. 11. Experiments using LT-DQN technique with three drones.

Table 5
LT-DQN computational latency and layer-switching delay.

Metric	Result
Control-cycle compute time	2.60 ± 0.31 ms
DQN layer latency	0.276 ± 0.023 ms
Three-beam latency	0.80 ± 0.28 μs
LiDAR + A* latency	0.770 ± 0.410 ms
No-switch cycle time	2.35 ± 0.22 ms
Switch cycle time	29.12 ± 13.25 ms
Switching delay Δ	26.77 ± 13.16 ms
Switch events per run	8.10 ± 6.67
LiDAR activations per run	4.40 ± 3.98
Three-Beam activations per run	8.50 ± 18.24
DQN activations per run	693.3 ± 189.6

infrequent switching events introduce measurable overhead, which occasionally increases the compute time. To make it clearer, we also reported the number of switch events per run, and the activation count for each layer.

Table 6
Two-layer vs. three-layer ablation results.

Metric	LT-DQN	DQN + LiDAR	DQN + Three-Beam
Success Rate	100 ± 0	80 ± 41	100 ± 0
Time to Goal (s)	45.00 ± 2.29	49.30 ± 3.47	48.31 ± 3.42
Path Length (m)	142.94 ± 4.17	147.10 ± 4.21	142.38 ± 5.24
Min Clearance (m)	5.07 ± 3.05	5.07 ± 3.51	4.92 ± 2.87
Collisions Count	0.00 ± 0.00	3.25 ± 6.02	0.33 ± 1.05
Avg Cycle Time (ms)	2.34 ± 0.33	7.50 ± 4.11	2.59 ± 0.51

4.3.3. *Layer contribution and hierarchical structure*

1) Two-layer vs. three-layer ablation.

We compared the proposed LT-DQN with two reduced-hierarchy variants (DQN + LiDAR and DQN + Three-Beam) to assess how different hierarchical designs impact reliability, efficiency and safety. Table 6 summarizes the ablation results of 10 trials comparing LT-DQN against two reduced hierarchies.

The results indicate the efficiency and reliability of the proposed method in achieving 100% success rate, and reaches the goal faster than other architectures with lower average cycle time (2.34 ± 0.33). Also, in

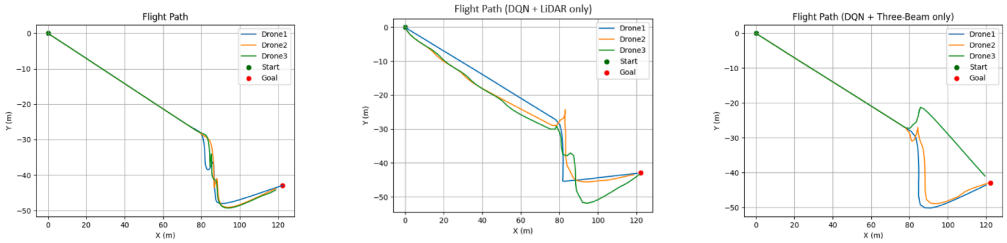


Fig. 12. Flight paths of three UAVs toward the target: (a) LT-DQN (three-layer), (b) two-layer DQN + LiDAR, and (c) two-layer DQN + Three-Beam.

Table 7
Comparison of different hierarchy orders and LT-DQN.

Metric	DQN + LiDAR + Three-Beam	LiDAR + DQN + Three-Beam	Three-Beam + DQN + LiDAR
Success Rate	100 ± 0	80 ± 45	100 ± 0
Time to Goal (s)	45.00 ± 2.29	37.94 ± 2.29	50.27 ± 8.21
Path Length (m)	142.94 ± 4.17	131.71 ± 2.26	147.24 ± 3.83
Min Clearance (m)	5.07 ± 3.05	3.88 ± 2.04	3.29 ± 2.47
Collisions Count	0.00 ± 0.00	0.58 ± 1.73	0.93 ± 2.58
Avg Cycle Time (ms)	2.34 ± 0.33	38.65 ± 14.08	2.69 ± 0.44
Path Planning (ms)	0.00 ± 0.01	1.47 ± 0.44	0.00 ± 0.00

terms of safety, it reaches the goal with (5.07 ± 3.05) clearance without any collisions. The path of the drones from the start point to the goal using LT-DQN, and the 2-layer experiments are shown in Fig. 12.

2) Hierarchy order ablation

The proposed framework relies on a safety-gated hierarchy that switches the control among modules whenever hazard is identified. Since the design relies on the priority order of the switching logic, an ablation study was performed to test whether the reported performance is sensitive to the layer ordering. Table 7 shows the results of three different hierarchies prioritizing each component at a time using the same environment.

The results in Table 7 show that LT-DQN hierarchy achieves 100% success rate with zero collisions maintaining low real-time control latency (2.34 ± 0.33). In contrast, the hierarchy with LiDAR-first layer reduces path length and time to goal but it increases computational cost (1.47 ± 0.44) and degrades robustness (0.80 ± 0.45). While the hierarchy with Three-Beam as the first layer maintains low control cycle latency (2.69 ± 0.44) but suffers from higher collision rate (0.93 ± 2.58) with slower time to goal (50.27 ± 8.21s). These outcomes confirm that switching order is not a neutral design choice: assigning highest priority to LiDAR planning increases planning cost and reduce responsiveness during rapid interactions; while prioritizing the purely reactive Three-Beam overemphasizes short-term avoidance that raises collision risk. Overall, the Lt-DQN hierarchy achieves the best safety, efficiency, and latency balance, supporting the selected priority structure for safety-critical navigation.

4.3.4. Robustness under harder conditions

1)Dynamic obstacles speed stress test

To evaluate the responsiveness of the proposed method under dynamic interactions, we varied the speed of dynamic obstacles using PlayRate parameters (0.5x, 1.0x, and 2.0x). For each PlayRate setting, 10 independent trials were performed and the results are reported as mean ± standard deviation in Table 8. The results indicate that the LT-DQN maintains 100% success rate across all the cases; however, higher obstacle speed increases collision rates especially in the 2.0x PlayRate. Moreover, increasing obstacle speed leads to longer trajectories, and increases the required time to goal, but the average control-cycle time stays nearly constant (≈ 2.2–2.4 ms). This indicates that the observed performance degradation is driven by increased dynamic complexity rather than computational slowdown.

2) Threshold sensitivity

Table 8
LT_DQN performance under varying dynamic obstacle speed.

Metric	Fast (2.0×)	Medium (1.0×)	Slow (0.5×)
Success (%)	100 ± 0	100 ± 0	100 ± 0
Collisions	0.30 ± 0.60	0.00 ± 0.00	0.00 ± 0.00
Time to Goal (s)	44.54 ± 9.66	38.17 ± 4.45	36.94 ± 1.13
Path Length (m)	120.16 ± 4.42	117.00 ± 3.39	118.87 ± 3.77
Min Clearance (m)	0.39 ± 0.51	0.50 ± 0.50	1.08 ± 0.27
Avg Cycle Time (ms)	2.37 ± 0.16	2.41 ± 0.20	2.20 ± 0.34

A sensitivity analysis was performed on the emergency distance threshold, which decides when the model switches to emergency behaviors if an obstacle becomes too close. Across the tested values (1m, 4m, and 7m), the drone's behavior in the scenarios does not strongly change. The navigation remains following the same path with a similar path length. The main visible difference is on the minimum clearance. Across tested thresholds, the minimum clearance remained in the same range (≈ 4.5–5.1 m), with a small mean decrease (≈ 0.6m) between the highest and lowest threshold. This indicates that the emergency threshold has limited impact on obstacle separation under the tested conditions.

3)Generalization to unseen environment

To assess generalization, LT-DQN was evaluated in an unseen environment (Landscape Mountain), which differs substantially from the training domain. Unlike the custom flat field, the mountain scene contains strong elevation variations, allowing us to verify that the method is not overfitted to the custom environment with obstacles and narrow passages. So, this out-of-distribution setting provides a more realistic test of environmental-level and map-independence. Across 10 trials in a multi-drone setting, the agents achieved 100% goal-reaching success. The mountain environment is predictably more challenging than the custom environment due to the terrain complexity and obstacle structure; additionally, the target was intentionally placed farther to create a more demanding long-horizon navigation task. Consequently, the averaged time to goal and averaged path length increased to (≈ 90.27 s) and (≈ 280.79 m) respectively, compared to (≈ 41.34 s) and (≈ 132.00 m) in the custom environment with a low collision count of (0.133 ± 0.434) as is shown in Table 9. These results demonstrate that the proposed hybrid method maintains stable navigation and transfers effectively beyond the training environment, while the

Table 9
Comparison of navigation performance in custom and mountain environments using the same frozen trained policy.

Metric	Custom environment	Mountain environment
Success rate	100 ± 0	100 ± 0
Time-to-goal (s)	41.34 ± 4.40	90.27 ± 12.85
Path length (m)	132.00 ± 12.50	280.79 ± 17.88
Min clearance (m)	4.04 ± 2.57	3.26 ± 1.91
Avg cycle time (ms)	2.27 ± 0.41	2.89 ± 0.60
Collision count	0.00 ± 0.00	0.133 ± 0.434

Table 10
Statistical significance tests and effect sizes at different velocities.

Speed	Metric	Test	p-value	Effect size
5	Success	Cochran's Q	0.112	—
5	Time to Goal	Friedman	< 0.001	$W = 0.859$
5	Path Length	Friedman	< 0.001	$W = 0.433$
5	Min Clearance	Friedman	0.021	$W = 0.134$
5	Collisions	Friedman	0.007	$W = 0.167$
5	Avg Planning	Friedman	< 0.001	$W = 1.000$
6	Success	Cochran's Q	0.012	—
6	Time to Goal	Friedman	< 0.001	$W = 0.653$
6	Path Length	Friedman	< 0.001	$W = 0.577$
6	Min Clearance	Friedman	< 0.001	$W = 0.262$
6	Collisions	Friedman	0.002	$W = 0.211$
6	Avg Planning	Friedman	< 0.001	$W = 0.835$

few collision cases indicate the increased complexity of the mountain scene.

4.3.5. Statistical testing

Significance testing was performed across methods for each velocity setting, to determine whether the observed performance differences are statistically significant. Cochran's Q test was used for binary results (success rate), Friedman test was applied for continuous metrics and then the p-values were reported with Kendall's W effect size. The results show a significant difference between methods for most metrics at both velocities as summarized in Table 10.

5. Discussion

The experimental results demonstrate that the proposed hybrid framework LT-DQN's main advantage extends beyond aggregate performance, highlighting the mechanism that enables robust reliability across diverse conditions. The proposed system decomposes navigation into complementary modes: adaptive learned control for nominal motion, a planning-based recovery mode for structured avoidance when risk increases, and a reactive fallback for extreme proximity or when planning is infeasible. This mode decomposition matters most in cluttered environments containing narrow passages and dynamic obstacles, where a single controller often faces an inherent trade-off between safety (avoiding collision with sufficient clearance) and efficiency (short path with fast progress). LT-DQN's gating mechanism selects efficient learned control when risk is low, and escalates to more conservative behaviors when risk increases, explaining its robust and consistent goal-reaching in both single-UAV and multi-UAV settings.

The timing analysis shows that the proposed model maintains low latency during steady-state conditions, with brief computational spikes occurring during layer transitions. This is expected because switching adds additional operations which are not executed at every step. The overhead is therefore event-driven: the lightweight policy handles most cycles, and heavier computation is triggered only when safety requires it.

The component-level analysis provides design-level evidence that the reported performance depends on the three-tier structure and the chosen priority order, rather than any single component alone. The results of comparison against the reduced two-layer variants show that eliminating any one of the safety layers changes the safety-efficiency balance. In case the design depends heavily on the planner, then it incurs a higher control-cycle time and reduced robustness during difficult interactions, while a design that depends on reactive avoidance increases collision risks even though it preserves low latency. Likewise, the hierarchy-order experiments show that prioritization is not a neutral choice: prioritizing LiDAR-based planner as the first-tier controller increases computational burden and reduces responsiveness during dynamic interaction, whereas prioritizing reactive control can favor immediate, short-range maneuvers at the expense of safety. Overall, these results support the architectural choice of utilizing learned control to handle nominal conditions, with deterministic escalation to planning and subsequently to reactive behavior as risk increases.

Robustness analyses indicate that LT-DQN remains stable in the presence of several stressors, yet it also clarifies the conditions that remain challenging. As dynamic-obstacle speed increases, the system remains reliable in goal reaching although collision probability can rise slightly under the most aggressive dynamics. Since the average control-cycle time stays nearly constant with respect to obstacle speed, the observed degradation is better explained by shortened reaction windows and increased interaction complexity rather than computational slowdown. Moreover, the threshold sensitivity analysis indicates that the controller is not fragile to the distance settings under the evaluated range. Changing the threshold primarily affects escalation timing and the corresponding safety margins, rather than inducing qualitatively different navigation behaviors. Furthermore, evaluation in an unseen mountain environment demonstrates that the hybrid policy transfers beyond the training domain, although the increased terrain complexity naturally leads to longer trajectories and occasional contacts, highlighting the value of testing across varied maps and dynamic conditions.

Limitation and Generalization: The proposed method was evaluated in simulation, and deploying it in the real world requires staged validation. Firstly, it requires hardware-in-the-loop (HIL) testing for timing and sensor latency. Secondly, it needs controlled stress tests under wind and sensor disturbances. Thirdly, autopilot-level safety must be ensured such as geofencing and fail-safe behaviors. Lastly, the switching logic has to be verified and systematically tested to ensure reliable fallback under abnormal conditions. In the current design, if none of the layers can generate a safe motion command under extreme conditions, then the system defaults to a conservative safe brake-and-hover behavior with zero command velocity. Once integrated with a flight controller, this safe behavior should invoke a higher-level fail-safe such as controlled landing or return-to-home depending on the mission requirements.

From a generalization standpoint, although the present study was conducted in simulation, the proposed method was evaluated across diverse environments and dynamic obstacle conditions, providing encouraging evidence of robustness beyond a single experimental scenario. Real-world implementation may still be influenced by factors including sensor noise, wind disturbances, and differences between simulated and physical obstacle motion. Nevertheless, the safety-gated hybrid system reduces reliance on the learned policy alone and provides a practical foundation for future real-world deployment. Accordingly, future work will include disturbance-injection experiments, hardware-in-the-loop evaluation for end-to-end timing analysis, and realistic outdoor trials.

Lastly, the use of autonomous multi-drone systems can raise safety and social concerns, such as misuse in restricted areas or the risk of harm to people or property. Therefore, real-world deployment should follow applicable regulations and safety protocols, including operator oversight and the maintenance of flight logs for traceability and post-flight auditing. These considerations further motivate conservative fail-safe design and careful pre-deployment validation.

6. Conclusion

This study introduces LT-DQN, a hybrid safety-gated obstacle avoidance framework that combines a lightweight DQN controller, LiDAR-based planning, and a reactive fallback for UAV navigation using only onboard sensing in cluttered environments. Across single-UAV and multi-UAV experiments, the proposed framework demonstrated superior performance compared with individual components, achieving a 100% goal-reaching rate with zero collisions in the main tested configurations, while maintaining safe clearance and low real-time control latency (2.60 ± 0.31 ms per cycle). These results demonstrate that the proposed framework provides an effective balance between safety, efficiency, and computational feasibility for UAV navigation in dynamic environments. Comparative statistical analysis further confirmed that the observed performance differences between methods are significant for key continuous metrics (time to goal, path length, collision counts, minimum clearance, and path planning) at both tested velocities. Several metrics exhibit moderate-to-large effect sizes such as success-rate differences which are statistically significant at velocity 6 m/s but not at 5 m/s, indicating that reliability gains are most pronounced under the more demanding settings.

At the same time, all evaluations were simulation-based and conducted in Unreal Engine with AirSim, and real-world deployment may be influenced by wind disturbance, sensor noise, and differences in obstacle dynamics. Accordingly, future work will focus on staged validation and progressive transfer to real-world flight conditions. This includes disturbance injection (wind and sensor noise models), hardware-in-the-loop timing checks, and progressively realistic field flight trials with autopilot-level safety measures, along with systematic verification of the switching logic and fail-safe behavior.

CRediT authorship contribution statement

Alaa H. Ahmed: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization; **Henrietta Tomán:** Supervision, Project administration, Funding acquisition.

Data availability

Data will be made available on request.

Declaration of competing interest

The authors whose names are listed immediately below certify that they have NO affiliations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers' bureaus; membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent-licensing arrangements), or non-financial interest (such as personal or professional relationships, affiliations, knowledge, or beliefs) in the subject matter or materials discussed in this manuscript.

References

[1] D. Hanover, A. Loquercio, L. Bauersfeld, A. Romero, R. Penicka, Y. Song, G. Cioffi, E. Kaufmann, D. Scaramuzza, Autonomous drone racing: a survey, 2023, arXiv:2301.01755.

[2] M.A.C. Montalvo, K.K. Batoy, C.M.G. Villame, Development of a mission planning and obstacle avoidance algorithm for unmanned aerial vehicle (UAV) using liDAR and potential field method within ROS-Gazebo, IAENG Int. J. Comput. Sci. (2023). Accepted / IAENG IJCS.

[3] G.-T. Tu, J.G. Juang, UAV path planning and obstacle avoidance based on reinforcement learning in 3D environments, Sensors 22 (3) (2022) 1059. https://doi.org/10.3390/s22031059

[4] Y. Xie, C. Yu, H. Zang, F. Gao, W. Tang, J. Huang, J. Chen, B. Xu, Y. Wu, Y. Wang, Multi-UAV behavior-based formation with static and dynamic obstacles avoidance via reinforcement learning, 2024, arXiv:2410.00000.

[5] A. Merei, H. Mcheick, A. Ghaddar, D. Rebaine, A survey on obstacle detection and avoidance methods for UAVs, Drones 9 (3) (2025) 203.

[6] J. Wu, Y. Ye, J. Du, Multi-objective reinforcement learning for autonomous drone navigation in urban areas with wind zones, Autom. Constr. 158 (2024) 105253. https://doi.org/10.1016/j.autcon.2023.105253

[7] T. Stefansson, 3D obstacle avoidance for drones using a realistic sensor setup, Master's thesis in Computer Science, KTH Royal Institute of Technology, School of Electrical Engineering and Computer Science 2018. https://kth.diva-portal.org/smash/get/diva2:1242725/FULLTEXT01.pdf

[8] E. Ferrera, A. Alcántara, J. Capitán, A.R. Castaño, P.J. Marrón, A. Ollero, Decentralized 3D collision avoidance for multiple UAVs in outdoor environments, Sensors 18 (12) (2018) 4101.

[9] H. Yu, F. Zhang, P. Huang, C. Wang, L. Yuanhao, Autonomous obstacle avoidance for UAV based on fusion of radar and monocular camera, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020, pp. 5954–5961. https://doi.org/10.1109/IROS45743.2020.9341432

[10] Y. Yu, W. Tingting, C. Long, Z. Weiwei, Stereo vision based obstacle avoidance strategy for quadcopter UAV, in: 2018 Chinese Control and Decision Conference (CCDC), 2018, pp. 490–494. https://doi.org/10.1109/CCDC.2018.8407182

[11] T. Kenny, Optical and radiation sensors, in: J.S. Wilson (Ed.), Sensor Technology Handbook, Newnes/Elsevier, Burlington, MA, USA; Cambridge, UK, 2005, pp. 307–320.

[12] A.H. Ahmed, H. Tomán, Stochastic fusion techniques for state estimation, Computation 12 (10) (2024) 209. https://doi.org/10.3390/computation12100209

[13] A.H. Ahmed, H. Tomán, Swarm drones with QR code formation for real-time vehicle detection and fusion using unreal engine, Automation 6 (4) (2025) 87. https://doi.org/10.3390/automation6040087

[14] M.A. Arshad, S.H. Khan, S. Qamar, M.W. Khan, I. Murtza, J. Gwak, A. Khan, Drone navigation using region and edge exploitation-based deep CNN, IEEE Access 10 (2022) 95441–95450. https://doi.org/10.1109/ACCESS.2022.3205959

[15] T. Lee, S. McKeever, J. Courtney, Flying free: a research overview of deep learning in drone navigation autonomy, Drones 5 (2) (2021) 52. https://doi.org/10.3390/drones5020052

[16] C. Tang, B. Abbatematteo, J. Hu, R. Chandra, R. Martín-Martín, P. Stone, Deep reinforcement learning for robotics: a survey of real-world successes, Annu. Rev. Control Rob. Auton. Syst. 8 (2025) 153–188. https://doi.org/10.1146/annurev-control-030323-022510

[17] L. Xu, W. Zhang, Survey on path planning based on deep reinforcement learning, in: Proceedings of the 2025 2nd International Conference on Machine Learning and Intelligent Computing, 278 of *Proceedings of Machine Learning Research*, PMLR, 2025, pp. 685–695. https://proceedings.mlr.press/v278/xu25a.html.

[18] W. Skarka, R. Ashfaq, Hybrid machine learning and reinforcement learning framework for adaptive UAV obstacle avoidance, Aerospace 11 (11) (2024). https://www.mdpi.com/2226-4310/11/11/870. https://doi.org/10.3390/aerospace11110870

[19] P. Miera, H. Szolc, T. Kryjak, LiDAR-based Drone Navigation with Reinforcement Learning, 2023, arXiv:2307.14313.

[20] B. Zhao, M. Huo, Z. Li, Z. Yu, N. Qi, Graph-based multi-agent reinforcement learning for large-scale UAVs swarm system control, Aerosp. Sci. Technol. 150 (2024) 109166. https://doi.org/10.1016/j.ast.2024.109166

[21] Á. Sütő, L. Kovács, I. Rudas, Optimal control of multiple drones for obstacle avoidance, IFAC-PapersOnLine 56 (2) (2023) 5475–5481. https://doi.org/10.1016/j.ifacol.2023.10.200

[22] Y. Lu, Z. Xue, G.-S. Xia, L. Zhang, A survey on vision-based UAV navigation, Geo-Spatial Inf. Sci. 21 (1) (2018) 21–32. https://doi.org/10.1080/10095020.2017.1420509

[23] Y. Chao, R. Dillmann, A. Roennau, Z. Xiong, E-DQN-based path planning method for drones in airsim simulator under unknown environment, Biomimetics 9 (4) (2024) 238. https://doi.org/10.3390/biomimetics9040238

[24] R. Yu, Q. Li, J. Ji, T. Wu, J. Mao, S. Liu, Z. Sun, Improved double DQN with deep reinforcement learning for UAV indoor autonomous obstacle avoidance, Sci. Rep. 15 (1) (2025). https://doi.org/10.1038/s41598-025-02356-6

[25] B. Lei, W. Hu, Z. Ren, S. Ji, DRL-based UAV autonomous navigation and obstacle avoidance with LiDAR and depth camera fusion, Aerospace 12 (9) (2025) 848. https://doi.org/10.3390/aerospace12090848

[26] F. AlMahamid, K. Grolinger, Agile DQN: adaptive deep recurrent attention reinforcement learning for autonomous UAV obstacle avoidance, Sci. Rep. 15 (1) (2025). https://doi.org/10.1038/s41598-025-03287-y

[27] S. Shah, D. Dey, C. Lovett, A. Kapoor, AirSim: high-fidelity visual and physical simulation for autonomous vehicles, in: Proceedings of the Field and Service Robotics (FSR), Springer, 2017, pp. 621–635. https://doi.org/10.1007/978-3-319-67361-5_40

[28] S. Menon, S.R. Addula, A. Parkavi, C. Subbalakshmi, V.B. Dhandayuthapani, K.S. Pokkuluri, A. Soni, Streamlining task planning systems for improved enactment in contemporary computing surroundings, SN Comput. Sci. 5 (2024) 993. https://doi.org/10.1007/s42979-024-03267-5

[29] P.E. Hart, N.J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE Trans. Syst. Sci. Cybern. 4 (2) (1968) 100–107. https://doi.org/10.1109/TSSC.1968.300136

[30] N. Steinke, D. Göhring, R. Rojas, GroundGrid: LiDAR point cloud ground segmentation and terrain estimation, IEEE Rob. Autom. Lett. 9 (1) (2024) 420–426. https://doi.org/10.1109/LRA.2023.3333233

[31] M.N.A. Al-Hamadani, M.A. Fadhel, L. Alzubaidi, B. Harangi, Reinforcement learning algorithms and applications in healthcare and robotics: a comprehensive and systematic review, Sensors 24 (8) (2024) 2461. https://doi.org/10.3390/s24082461

[32] Q. Wang, Z. Zhan, Reinforcement learning model, algorithms and its application, in: Proceedings of the IEEE Conference, IEEE, 2011.
[33] J. Roghair, A. Niaraki, K. Ko, A. Jannesari, A vision based deep reinforcement learning algorithm for UAV obstacle avoidance, in: Proceedings of the SAI Intelligent Systems Conference, Springer, Berlin/Heidelberg, Germany, Virtual Conference, 2021, pp. 115–128. https://doi.org/10.1007/978-3-030-82199-9_9
[34] S.Y. Shin, Y.W. Kang, Y.G. Kim, Obstacle avoidance drone by deep reinforcement learning and its racing with human pilot, Appl. Sci. 9 (22) (2019) 5571. https://doi.org/10.3390/app9225571